Chapter 4

Peer-reviewed and published cosmological code

4.1 Introduction

In this chapter we describe some pieces of code which were developed in order to be able to tackle some of the problems presented in the previous chapters, and to prepare the tools for future research.

Calculating is a fundamental part of Theoretical Physics research. The very act of carrying out the algebraic or numeric integral of a 2-loop process is arguably not Physics *per se* – as some of my professors would say, it is just cranking the *mathematical coffee mill*. Physics is to be found in the motivation of the problem, in the choice of the integration method, in the approximations made and those deliberately not made, in the method used to test the result, and in the interpretation of it in the context of the theory.

Together with algebraic calculations, computer-assisted calculations are essential in Theoretical Physics, specially when heavy numerical computations, randomisation or simulations are involved. The challenge in this cases is often carrying out the calculations in a manageable amount of time, controlling all possible errors.

Creating the tools for computer-assisted calculations, *programming* or *coding*, *is* Physics research in the same way as algebraic calculations are: Physics is in the choice of the algorithms, the approximations, the means to control the errors, the interpretation of the result... all that based on the nature of the problem and its physical context.

In order to carry out the research presented in this thesis, and also in order to facilitate similar research by our peers, we created a small number of tools that have been published under an open source license as an integral part of two Cosmological codes, CLASS and Monte Python, having previously been reviewed by the authors of the respective codes. In the next sections we describe them briefly.

4.2 Arbitrary Primordial Power Spectrum for CLASS

In chapter 2, we have attempted to fit a modified shape of the slow-roll, single field primordial power spectrum to the Planck CMB data. In order to accomplish that, we needed to integrate into a cosmological code the ability to start from an arbitrary primordial power spectrum. It would be desirable for the primordial power spectrum to be at the same time (1) calculated on-the-fly from a series of parameters, (2) easily modified without the need to maintain many different compiled versions of the cosmological code, and (3) the relevant parameters should be easily handled by a Monte Carlo sampler.

We found those requirements to be satisfied more easily when integrating them into the Boltzmann code CLASS [2], due, among other things, to the cleanness and modularity of its code and the effective support from its authors in implementing the modifications presented here.

With the requirements stated above at hand, we opted for modifying CLASS minimally to allow it to launch an external program and retrieve from it the primordial power spectrum as a table in plain text, from which CLASS creates an internal interpolating function. The external power spectrum generator gets the values of the necessary parameters through CLASS itself, either as defined in a CLASS input file or fed by a Monte Carlo sampler. This way, CLASS does not need to be recompiled when modifying the shape of the primordial power spectrum, and the external generator can be coded in any language that appeals the researcher, following minimum requirements. Even the case of a precomputed table of values of the primordial power spectrum is trivially included, through the use of the `cat` command of Unix-like systems.

The main disadvantage of this external-program approach, as opposed to hard coding the power spectrum, is the delay inherent to interfacing through plain text. Fortunately, this delay is small compared to the most time-consuming part of the cosmological code: the calculation of the transfer functions.

This modification of CLASS, together with the necessary documentation written by the author of this thesis, was reviewed by one of the authors of CLASS, Julien Lesgourgues, and integrated into the main code and released to the community in CLASS version 2.1, on March 2014.

As a simple example we show here how easy it is to implement an oscillatory feature on top of the primordial scalar power spectrum:

$$\mathcal{P}(k) = \left(A_\mathrm{s}\left(\frac{k}{k_0}\right)^{n_\mathrm{s}-1}\right)\left(1 + B\sin\left(2\pi\frac{k}{k_l}+\phi\right)\right)\ . \tag{4.1}$$

```
P_k_ini type = external_Pk
command = python external_Pk/generate_Pk_sin.py
custom1 = 0.05      # Pivot scale
custom2 = 2.215e-9  # Amplitude of the power spectrum
custom3 = 0.9624    # Spectral index of the power spectrum
custom4 = 0.0       # Amplitude of the feature
custom5 = 0.002     # Wavelength of the feature
custom6 = 0         # Phase of the feature
```

```python
# Retrieving the necessary parameters
import sys
k_0 = float(sys.argv[1]) # Pivot scale
A   = float(sys.argv[2]) # Amplitude of the power spectrum
n_s = float(sys.argv[3]) # Spectral index of the power spectrum
B   = float(sys.argv[4]) # Amplitude of the feature
k_l = float(sys.argv[5]) # Wavelength of the feature
phi = float(sys.argv[6]) # Phase of the feature

# P(k) calculation
from math import exp, sin, pi
def P(k):
    return (A * (k/k_0)**(n_s-1.) *
            (1 + B * sin(2*pi*k/k_l + phi)))

# Limits for k and precision
k_min, k_max  = 1.e-6, 0.75
k_per_decade_primordial = 1000.

# Producing the values of P(k)
k = k_min
while k <= k_max:
    print "%.18g %.18g" % (k, P(k))
    k = k * 10.**(1./float(k_per_decade_primordial))
```

Figure 4.1: **Top:** lines of the CLASS input file defining the use of the oscillatory spectrum. **Bottom:** Python script that generates the spectrum of eq. (4.1).

As we see, the whole spectrum is parametrised by the global amplitude A_s and spectral index n_s, together with the oscillation amplitude B, its wavelength k_l and its phase ϕ. In order for CLASS to use that as a primordial spectrum, it is enough to add to the CLASS input file the lines on the upper box of figure 4.1, where the contents of the file `generate_Pk_sin.py` can be seen in the bottom of the same figure. Such a simple code fulfils all the necessary requirements. The resulting primordial and CMB spectra can be seen in figure 4.2.

4.3 Interfacing of MultiNest into Monte Python

Bayesian statistics judges the adequacy of some physical models over others in terms of their predictivity: more predictive models are preferred over less predictive ones. Predictivity of a model is the joint effect of how well it describes the data together with how restricted is its range of predictions. The joint measurement of both aspects is the *Bayesian evidence*, equal to the integral of the

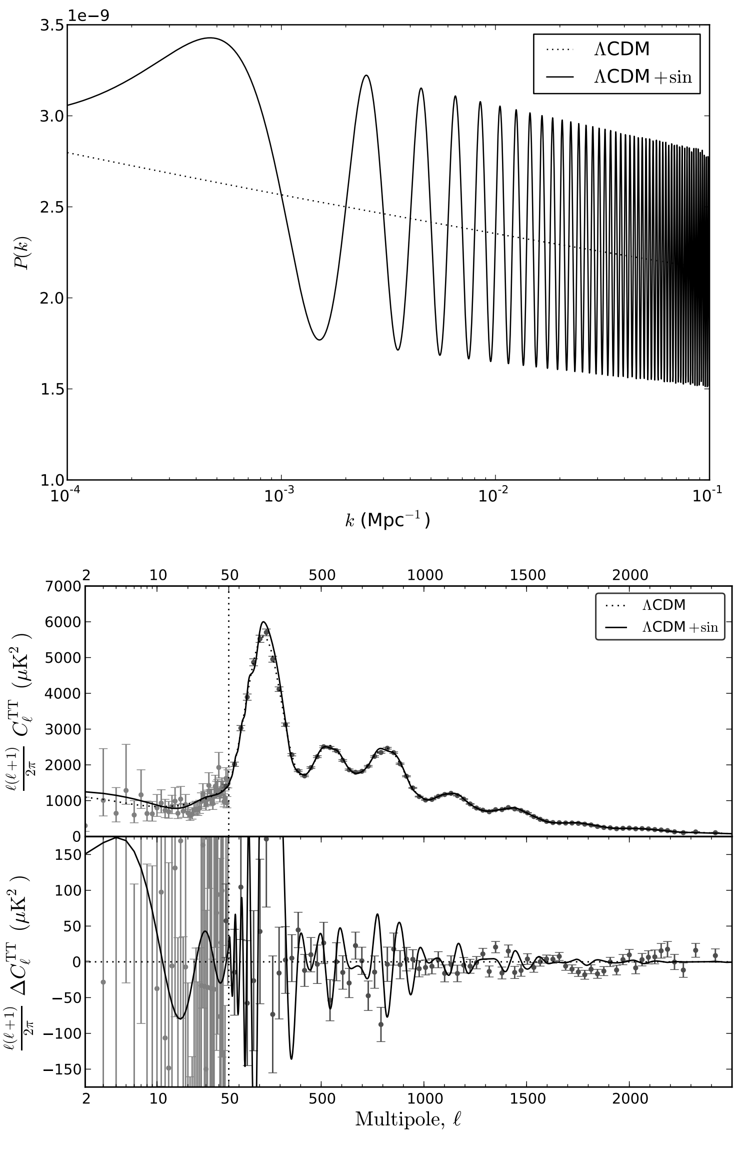

Figure 4.2: **Top:** Primordial scalar power spectrum of eq. (4.1), assuming Planck's best fit values for A_s and n_s, and some random values for the parameters of the oscillation. **Bottom:** Resulting (lensed) CMB temperature power spectrum.

likelihood and the *prior*:

$$B_{\mathcal{M}} = \int_{\Omega} \mathcal{L}(\mathcal{D}|\mathcal{M}(\omega))\, \pi(\omega|\mathcal{M})\, \mathrm{d}\omega \ , \tag{4.2}$$

$\mathcal{D}$ being a set of data, and $\mathcal{M}$ a physical model parametrised by the quantit*ies* ω, which take values on a set Ω with probability $\pi(\omega|\mathcal{M})$.

Computing the Bayesian evidence is a hard task to accomplish, which in general requires methods different from those used for *parameter inference* (i.e. estimating the parameter values that best fit the data and their distribution around those values). While Markov chain Monte Carlo methods (MCMC) are one efficient method for parameter inference, they fall short when used for computation of the Bayesian evidence, mostly due to their lack of sampling in the areas of the parameter space away from the best fit, which in practice do not affect the confidence intervals of the parameters, but can add up to a significant fraction of the total value of the evidence. An alternative sampling method is needed.

Nested sampling [8] is one of those methods aimed at calculating accurately the Bayesian evidence. Its adequacy arises, on the one hand, from how it samples the parameter space, climbing from the lowest to the highest values of the likelihood allowed by the prior ranges; and, on the other hand, on its slow scaling with the number of dimensions of the parameter space. One particular enhancement over nested sampling is the MultiNest algorithm [5, 6, 4]. MultiNest improves on nested sampling by adding an ellipsoidal-decomposition algorithm that allows both for sampling of weirdly-shaped distributions and for simultaneous sampling of multiple *modes*, or regions of high probability, in complicated probability distributions that may contain several of them (this is precisely our case, see chapters 2 and 3). In addition to an accurate calculation of the Bayesian evidence of complicated distributions, MultiNest is also able to produce a fair sample of the parameter space under the likelihood, that can be used for parameter inference in a similar way as the chains of samples coming from MCMC's.[1]

All that makes MultiNest a very interesting tool for data analysis, which would be desirable to use along the common MCMC methods. In order to do so, we wish to combine MultiNest with the necessary computational cosmological tools, mainly Boltzmann codes and experimental likelihoods. This is naturally achieved by interfacing MultiNest through an already established cosmological MCMC sampler, such as CosmoMC [7] or Monte Python [1]. Since the Boltzmann code used in this thesis was initially mostly CLASS [2], it made sense to create this interface for the companion sampler Monte Python.

The interface written by the author of this thesis and the main author of Monte Python, Benjamin Audren, starts from PyMultiNest, the Python wrapper of MultiNest written by Johannes Buchner [3], which takes care of the

[1]Since the samples of MultiNest are scarcer, but the sampling is faster, the sample from MultiNest can also be used to roughly estimate the covariance matrix of the likelihood, and then use it to make more efficient an MCMC, which will characterise the distribution around the minimum more thoroughly – a great idea of Benjamin Audren.

FORTRAN–Python communication in a neat way. MultiNest's input, both physical and sampling parameters, is passed through the standard input methods of Monte Python (parameter file and command line, respectively), and the output of MultiNest is simultaneously

- kept unmodified in its raw format in a separate folder for later additional analyses;
- used to create MCMC-like samples in Monte Python format from the fair samples of MultiNest, in order to use the same Monte Python analysis tools. The different modes are separated in different, independent *chain folders*.

In addition, our interface adds the capability, not present in the current version of MultiNest, of using as *clustering parameters* any combination of them, not necessarily those in the first positions.

This interface, together with the necessary documentation written by the author of this thesis, was integrated into the main code and released to the community with Monte Python version 2.0, on March 2014.

Together with the interface, we created a simple example likelihood with 3 distinct modes:

$$\mathcal{L}\left(\mathcal{D}\,|\,\boldsymbol{v}\right) = \prod_{i=1}^{3} \exp\left\{-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{\mu}_i)\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{v}-\boldsymbol{\mu}_i)\right\} \ , \tag{4.3}$$

where $\boldsymbol{v} := (H_0,\, \omega_{\rm b},\, \omega_{\rm CDM})$ are the parameters of the model and the data $\mathcal{D}$ enters through the central values $\boldsymbol{\mu}_i$ and the covariance matrices $\boldsymbol{\Sigma}_i$:

$$\boldsymbol{\mu}_1 = (67.0,\ 0.02225,\ 0.0120)\ , \qquad \boldsymbol{\Sigma}_1 = \begin{pmatrix} 0.1 & 2{\cdot}10^{-5} & 2{\cdot}10^{-4} \\ 2{\cdot}10^{-5} & 2{\cdot}10^{-7} & 2.4{\cdot}10^{-7} \\ 2{\cdot}10^{-4} & 2.4{\cdot}10^{-7} & 5{\cdot}10^{-6} \end{pmatrix}\ , \tag{4.4a}$$

$$\boldsymbol{\mu}_2 = (69.5,\ 0.02300,\ 0.0170)\ , \qquad \boldsymbol{\Sigma}_2 = \begin{pmatrix} 0.3 & 2{\cdot}10^{-5} & 3.5{\cdot}10^{-4} \\ 2{\cdot}10^{-5} & 1.2{\cdot}10^{-7} & 1.4{\cdot}10^{-7} \\ 3.5{\cdot}10^{-4} & 1.4{\cdot}10^{-7} & 2{\cdot}10^{-6} \end{pmatrix}\ , \tag{4.4b}$$

$$\boldsymbol{\mu}_3 = (71.5,\ 0.02180,\ 0.0100)\ , \qquad \boldsymbol{\Sigma}_3 = \begin{pmatrix} 0.2 & 5.5{\cdot}10^{-5} & 3{\cdot}10^{-4} \\ 5.5{\cdot}10^{-5} & 4{\cdot}10^{-7} & 5{\cdot}10^{-7} \\ 3{\cdot}10^{-4} & 5{\cdot}10^{-7} & 5{\cdot}10^{-6} \end{pmatrix}\ . \tag{4.4c}$$

The result of the sampling of this likelihood using MultiNest through the interface described in this section is show in fig. 4.3. The dashed lines show the cubic prior regions corresponding to each of the modes, cut automatically by MultiNest and stored as three different Monte Python chains.

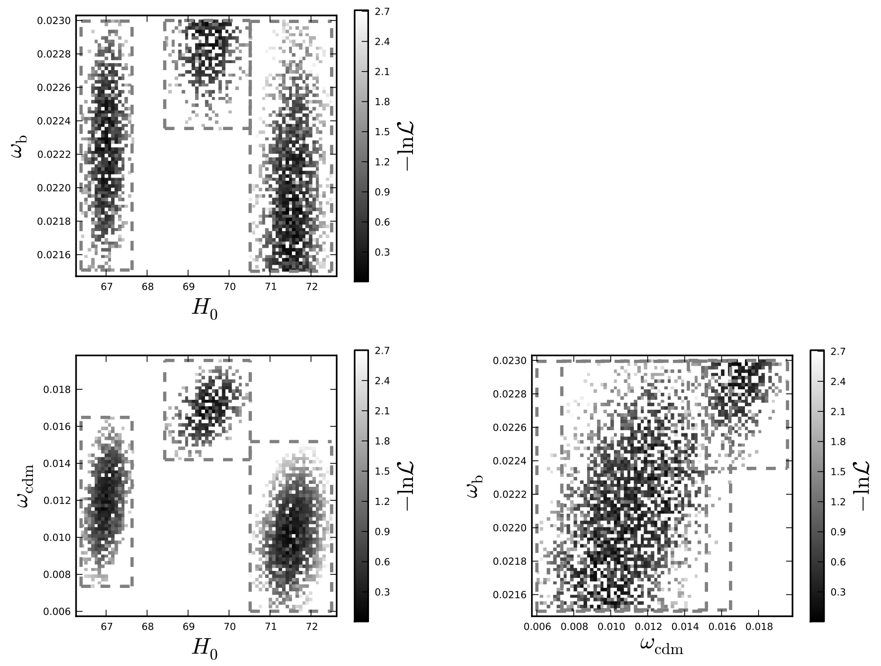

Figure 4.3: Profile of the sampling of the likelihood given by eq. (4.3), using MULTINEST interfaced through MONTE PYTHON.

Bibliography

[1] Benjamin Audren, Julien Lesgourgues, Karim Benabed, and Simon Prunet. Conservative Constraints on Early Cosmology: an illustration of the Monte Python cosmological parameter inference code. *JCAP*, 1302:001, 2013, 1210.7183.

[2] Diego Blas, Julien Lesgourgues, and Thomas Tram. The Cosmic Linear Anisotropy Solving System (CLASS) II: Approximation schemes. *JCAP*, 1107:034, 2011, 1104.2933.

[3] J. Buchner, A. Georgakakis, K. Nandra, L. Hsu, C. Rangel, et al. X-ray spectral modelling of the AGN obscuring region in the CDFS: Bayesian model selection and catalogue. 2014, 1402.0004.

[4] F. Feroz, M. P. Hobson, E. Cameron, and A. N. Pettitt. Importance Nested Sampling and the MultiNest Algorithm. *ArXiv e-prints*, June 2013, 1306.2144.

[5] F. Feroz, M.P. Hobson, and M. Bridges. MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Mon.Not.Roy.Astron.Soc.*, 398:1601–1614, 2009, 0809.3437.

[6] Farhan Feroz and M.P. Hobson. Multimodal nested sampling: an efficient and robust alternative to MCMC methods for astronomical data analysis. *Mon.Not.Roy.Astron.Soc.*, 384:449, 2008, 0704.3704.

[7] Antony Lewis and Sarah Bridle. Cosmological parameters from CMB and other data: a Monte- Carlo approach. *Phys. Rev.*, D66:103511, 2002, astro-ph/0205436.

[8] John Skilling. Nested sampling. *AIP Conference Proceedings*, 735(1):395–405, 2004.