Cosmology with CosmoMC
Solution to Exercises

Antonio J. Cuesta, ICCUB Barcelona

17–21 April 2016

Abstract

This is a compilation of the exercises from the Tutorial session "Cosmology with CosmoMC" at the 1st Mexican AstroCosmoStatistics School, held in April 17-21 in León, Guanajuato, Mexico. The code CosmoMC [Lewis and Bridle, 2002] is a Bayesian parameter inference code for Cosmology. The website is `http://cosmologist.info/cosmomc`

Installation

As a requirement for completing the exercises, one has to install the July 2015 version of CosmoMC. The following script will install Anaconda Python distribution[1], ESA/Planck 2015 likelihoods and data[2], and CosmoMC[3]. The installer for Anaconda will ask you some questions, to which you should answer 'yes'. Some commands below are minor modifications to the original CosmoMC files to make them work on the supercomputer used during the School. Also, before installing CosmoMC you must agree to the terms and conditions of its license, which you can do through the following website:
`http://cosmologist.info/cosmomc/submit.html`.

```
#!/bin/bash

# Setting up environment
module load intel/14.0.1
module load openmpi/1.8.1_intel
source /opt/apps/compilers/intel/composer_xe_2013_sp1.1.106/mkl/bin/mklvars.sh intel64
echo -e "\nmodule load intel/14.0.1" >> ~/.bashrc
echo -e "\nmodule load openmpi/1.8.1_intel" >> ~/.bashrc
echo -e "\nsource /opt/apps/compilers/intel/composer_xe_2013_sp1.1.106/mkl/bin/mklvars.sh intel64" >> ~/.bashrc

# Downloading and installing Anaconda+PySide
wget https://3230d63b5fc54e62148e-c95ac804525aac4b6dba79b00b39d1d3.ssl.cf1.rackcdn.com/Anaconda2-4.0.0-Linux-x86_64.sh
bash Anaconda2-4.0.0-Linux-x86_64.sh
export PATH="$HOME/anaconda2/bin:$PATH"
conda install pyside

# Downloading and installing Planck2015
wget http://irsa.ipac.caltech.edu/data/Planck/release_2/software/COM_Likelihood_Code-v2.0.R2.00.tar.bz2
tar xvjf COM_Likelihood_Code-v2.0.R2.00.tar.bz2
wget http://irsa.ipac.caltech.edu/data/Planck/release_2/software/COM_Likelihood_Data-baseline_R2.00.tar.gz
tar xvzf COM_Likelihood_Data-baseline_R2.00.tar.gz
cd plc-2.0/
./waf configure --lapack_mkl=${MKLROOT} --install_all_deps
./waf install
source ./bin/clik_profile.sh
echo -e "\nsource $(pwd)/bin/clik_profile.sh" >> ~/.bashrc
cd ..

# Downloading and installing CosmoMC
wget https://github.com/cmbant/CosmoMC/archive/July2015.zip
unzip July2015.zip
cd CosmoMC-July2015/
ln -s ~/plc_2.0 ./data/clik
export PYTHONPATH="$HOME/CosmoMC-July2015/python:$PYTHONPATH"
echo -e "\nexport PYTHONPATH=$HOME/CosmoMC-July2015/python:\$PYTHONPATH" >> ~/.bashrc
sed -i -e 's/-fpp/-fpp -msse3/g' source/Makefile #to run in AMD nodes
sed -i -e 's/+= -xHost/+=/g' source/Makefile     #to avoid xHost to overrule msse3
sed -i -e  "s/items\[0\]/items\[0\]+'.atocatl-hpc.astroscu.unam.mx'/g" python/paramgrid/jobqueue.py #fix for runningJobs.py
sed -i -e 's/-U/-u/g' python/paramgrid/jobqueue.py #fix for runningJobs.py
make all
mpirun -np 1 ./cosmomc test_planck.ini
cd ..
```

[1] `http://www.continuum.io`
[2] `http://pla.esac.esa.int/pla/#cosmology`
[3] `http://cosmologist.info/cosmomc`

Exercise 1: Running a simple case

This is meant to be a first approach to CosmoMC in which we set up a cosmological model and choose a combination of data sets by editing the parameter file. The goal of this exercise is to run a simple case in which we determine the cosmological parameter constraints by CMB+BAO on the ΛCDM+m_ν model. This will be used in Exercise 3.

If you use `test.ini` as a sample parameter file and do the required modifications, you will easily obtain something like this. Note that the position of `batch2/common.ini` with respect to the other likelihoods is important, e.g. if placed at the top of the file then `use_BAO` will be set by default to False.

```
#Planck 2015, default just include native likelihoods (others require clik)
DEFAULT(batch2/plik_dx11dr2_HM_v18_TTTEEE.ini)
DEFAULT(batch2/lowTEB.ini)

#Other Likelihoods
DEFAULT(batch2/BAO.ini)

#general settings
DEFAULT(batch2/common.ini)

#high for new runs
MPI_Max_R_ProposeUpdate = 30

propose_matrix= planck_covmats/base_mnu_BAO_TTTEEE_lowTEB_plik.covmat

#Folder where files (chains, checkpoints, etc.) are stored
root_dir = chains/base_mnu/

#Root name for files produced
file_root= plikHM_TTTEEE_lowTEB_BAO
#action= 0 runs chains, 1 importance samples, 2 minimizes
#use action=4 just to quickly test likelihoods
action = 0

num_threads = 0

start_at_bestfit =F
feedback=1
use_fast_slow = T

checkpoint = T

#sampling_method=7 is a new fast-slow scheme good for Planck
sampling_method = 7
dragging_steps  = 3
propose_scale = 2

#Set >0 to make data files for importance sampling
indep_sample=10

#these are just small speedups for testing
get_sigma8=T

num_massive_neutrinos= 3
#to vary parameters set param[name]= center, min, max, start width, propose width
param[mnu] = 0.02 0 5 0.1 0.03
```

Then we can run CosmoMC with the script `python/runMPI.py`. Note that the directory `chains/base_mnu` needs to be created before running the code.

```
python python/runMPI.py --chainsPerNode 8 --coresPerNode 24 --walltime 120:00:00 --queue lpq1a base_mnu_plikHM_TTTEEE_lowTEB_BAO.ini
```

where `base_mnu_plikHM_TTTEEE_lowTEB_BAO.ini` is the name of the parameter file above.

Exercise 2: Running a grid of cases

The BOSS collaboration[4] has compiled their cosmological results in a table of parameter constraints from the data set combinations CMB+BAO and CMB+BAO+SN for six different models, i.e. the (flat) ΛCDM, wCDM, w_0w_aCDM models, as well as their corresponding versions with non-zero spatial curvature $o\Lambda$CDM, owCDM, ow_0w_aCDM. The goal here is to run this set of 12 cases using the Python tools in CosmoMC without going to through the work of writing 12 parameter files. Use the file `python/settings_sample.py` as a guide.

```
from paramgrid import batchjob

# Directory to find .ini files
ini_dir = 'batch2/'

# directory to look for existing covariance matrices
cov_dir = 'planck_covmats/'

# ini files you want to base each set of runs on
defaults = ['common.ini']
importanceDefaults = ['importance_sampling.ini']

# set up list of groups of parameters and data sets
groups = []

# make first group of runs (all parameter variations with all data combinations)
g = batchjob.jobGroup('main')

g.params = [[], ['omegak'], ['w'], ['omegak','w'], ['w','wa'], ['omegak', 'w', 'wa'] ]

g.datasets = []

# lists of dataset names to combine, with corresponding sets of inis to include
g.datasets.append(batchjob.dataSet(['plikHM', 'TTTEEE', 'lowTEB', 'BAO'],
                                   ['plik_dx11dr2_HM_v18_TTTEEE.ini', 'lowTEB.ini', 'BAO.ini']))
g.datasets.append(batchjob.dataSet(['plikHM', 'TTTEEE', 'lowTEB', 'BAO', 'JLA'],
                                   ['plik_dx11dr2_HM_v18_TTTEEE.ini', 'lowTEB.ini', 'BAO.ini','JLA.ini']))


# add importance name tags, and list of specific .ini files to include (in batch1/)
g.importanceRuns = []
g.importanceRuns.append([['HST'], ['HST.ini']])

groups.append(g)

# ranges for parameters when they are varied (can delete params if you just want to use defaults)
params = dict()
params['w'] = '-0.99 -3. 1 0.02 0.02'
params['wa'] = '0 -3 2 0.05 0.05'
params['omegak'] = '-0.0008 -0.3 0.3 0.001 0.001'  # starting exactly on flat seems to confuse minimizer


# try to match each new run name to exisitng covmat (covariance matrix for efficient exploration)
# e.g. try to map to get name without particular data combinations
covWithoutNameOrder = ['lensing', 'BAO']
# or try replacing various names (these are standard for the provided planck_covmats)
covNameMappings = {'plikHM': 'plik', 'plikLite': 'plik'}
# for mapping to nominal mission names try
# covNameMappings = {'plikHM':'planck','TT':'', 'lowTEB':'lowLike'}
```

If you save the above script as `python/settings_boss.py`, this grid can then be created and run (with `OMP_NUM_THREADS=2`) via the following two commands:

```
python python/makeGrid.py boss_grid settings_boss
python python/runbatch.py --queue lpq1a --chainsPerNode 4 --coresPerNode 24 --runsPerJob 3 --walltime 240:00:00 boss_grid/
```

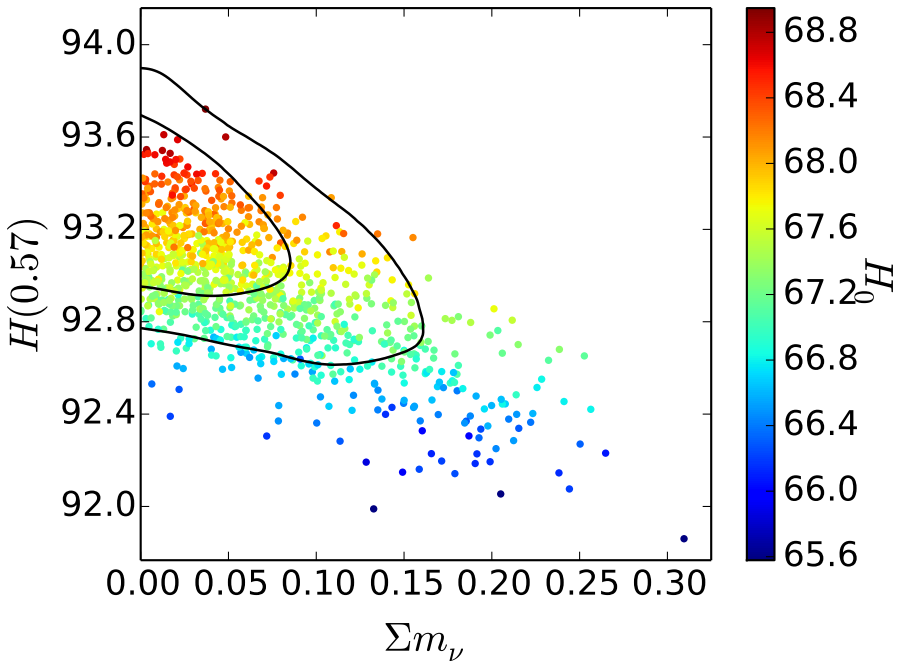
where `boss_grid` should be a symbolic link to a folder somewhere in your `/scratch` directory.

[4] `http://arxiv.org/abs/1312.4877`

Exercise 3: Importance sampling

In this exercise, we use the chains run for Exercise 1 and use the latest Hubble constant measurement by Riess et al.(2016)[5]. Just replace the 2011 values in `batch2/HST.ini` with the new measurement, make a copy of the original parameter file you used in Exercise 1, replace `action=0` in the parameter file with `action=1`, add this updated HST likelihood to the parameter file with `DEFAULT(batch2/HST.ini)` at the top of the file, and run CosmoMC using this new parameter file.

If you need to create the parameter file from scratch, note that the default is `redo_add=F` so you need to explicitly specify all likelihoods again, and if there is any difference between the parameters in the old and the new chains (for example, if you did not include Planck nuisance parameters explicitly in the parameter file), then the code will complain. Those warnings are flagging that the likelihoods do not match the chains you want to post-process.



The color dots are from the original chains, and the contours are the post-processed chains that include the H_0 constraint. Adding the new measurement of the Hubble constant on top of the CMB+BAO determination of neutrino mass of $m_\nu < 0.17$eV results in a more constraining bound of $m_\nu < 0.13$eV.

[5] `http://arxiv.org/abs/1604.01424`

Exercise 4: Derived parameters

This exercise is the first case in which we will modify and recompile the source code, so it is strongly recommended to install again CosmoMC in a separate directory e.g. `myCosmoMC` and make any modifications to the original code in that directory. Feel free to use version control if you are familiar with it!

The goal of this exercise is to add another redshift output ($z = 0.32$) to BAO and LSS quantities. The first file you need to modify is `source/CosmologyTypes.f90`. Line 42 should look like this:

```
real(mcp) :: z_outputs(2) = [0.32_mcp,0.57_mcp]
```

Also you need to add the theory and LSS derived parameters at the new redshift outputs to make them correspond to the derived parameters that CosmoMC will generate. The new file `paramnames/derived_theory.paramnames` should look like this (note that the order of the redshifts is important).

```
age*          {\rm{Age}}/{\rm{Gyr}}
zstar*        z_*
rstar*        r_*
thetastar*    100\theta_*
DAstar*       D_{\rm{A}}/{\rm{Gpc}}
zdrag*        z_{\rm{drag}}
rdrag*        r_{\rm{drag}}
kd*           k_{\rm D}
thetad*       100\theta_{\rm{D}}
zeq*          z_{\rm{eq}}
keq*          k_{\rm{eq}}
thetaeq*      100\theta_{\rm{eq}}
thetarseq*    100\theta_{\rm{s,eq}}
rsDv032*       r_{\rm{drag}}/D_V(0.32)
Hubble032*     H(0.32) #now in km/s/Mpc
DA032*         D_A(0.32)
FAP032*        F_{\rm AP}(0.32)
rsDv057*       r_{\rm{drag}}/D_V(0.57)
Hubble057*     H(0.57) #now in km/s/Mpc
DA057*         D_A(0.57)
FAP057*        F_{\rm AP}(0.57)
```

and the file `paramnames/derived_LSS.paramnames` should look like this:

```
fsigma8z032*   f\sigma_8(0.32) #at redshift 0.32
sigma8z032*    \sigma_8(0.32)
fsigma8z057*   f\sigma_8(0.57) #at redshift 0.57
sigma8z057*    \sigma_8(0.57)
```

This exercise can be generalized to an arbitrary number of redshift outputs, which can then be used (for example) to constrain the expansion history of the Universe. In that case, you might need to modify `max_derived_parameters` at line 18 in `source/CosmologyTypes.f90`.

In the case of a generic derived parameter, then one should increase `num_derived` in line 25 of `source/CosmologyParameterizations.f90` as well as `ix` in line 219 by the number of parameters desired, then define the parameters themselves right above line 219 as new components of the `derived` array. The name and definition of the new derived parameters have to be included (with its correct position) in the file `paramnames/params_CMB.paramnames`.

Exercise 5: Writing a likelihood

In this exercise we want to define a simple likelihood that we want to include in our chains (either running chains or via post-processing). The likelihood to include is a simple combination of $(\Omega_m/0.27)^{0.46}\,\sigma_8 = 0.774 \pm 0.040$ as determined by the CFHTLens collaboration[6].

First we need to name our likelihood, e.g. `CFHTLens`. Taking the file `source/HST.f90` as a guide, the file `source/CFHTLens.f90` should look like this:

```
module CFHTLens
use CosmologyTypes
use CosmoTheory
use Likelihood_Cosmology
implicit none
private

type, extends(TCosmoCalcLikelihood) :: CFHTLensLikelihood
    real(mcp) :: Omegam = 0.27
    real(mcp) :: alpha  = 0.46
    real(mcp) :: Oms8, Oms8_err
contains
procedure :: LogLike => CFHTLens_LnLike
end type CFHTLensLikelihood

public CFHTLensLikelihood, CFHTLensLikelihood_Add
contains

subroutine CFHTLensLikelihood_Add(LikeList, Ini)
class(TLikelihoodList) :: LikeList
class(TSettingIni) :: ini
Type(CFHTLensLikelihood), pointer :: this

if (Ini%Read_Logical('use_CFHTLens',.false.)) then
    allocate(this)
    this%LikelihoodType = 'CFHTLens'
    this%name= Ini%Read_String('CFHTLens_name')
    this%Oms8 = Ini%Read_Double('CFHTLens_Oms8')
    this%Oms8_err = Ini%Read_Double('CFHTLens_Oms8_err')
    call Ini%Read('CFHTLens_Omegam',this%Omegam)
    call Ini%Read('CFHTLens_alpha',this%alpha)
    this%needs_background_functions = .true.
    call LikeList%Add(this)
end if

end subroutine CFHTLensLikelihood_Add

real(mcp) function CFHTLens_LnLike(this, CMB, Theory, DataParams)
Class(CFHTLensLikelihood) :: this
Class(CMBParams) CMB
Class(TCosmoTheoryPredictions), target :: Theory
real(mcp) :: DataParams(:)
real(mcp) :: omegam
real(mcp) :: theoryval

omegam = 1.d0 - CMB%omv - CMB%omk
theoryval = (omegam/this%Omegam)**(this%alpha)*Theory%sigma_8
CFHTLens_LnLike = (theoryval - this%Oms8)**2/(2*this%Oms8_err**2)

end function  CFHTLens_LnLike

end module CFHTLens
```

[6] http://arxiv.org/abs/1303.1808

We also need to create a data file for our likelihood. So this would be the contents of `batch2/CHFTLens.ini`:

```
CFHTLens_Omegam   = 0.27
CFHTLens_alpha    = 0.46
CFHTLens_Oms8     = 0.774
CFHTLens_Oms8_err = 0.040
CFHTLens_name = Heymans2013
use_CFHTLens=T
```

Then we just add these lines to the file `source/DataLikelihoods.f90`:

```
use CFHTLens
call CFHTLensLikelihood_Add(DataLikelihoods, Ini)
```

and similarly we add these lines to the file `source/Makefile`:

```
$(OUTPUT_DIR)/CFHTLens.o: $(OUTPUT_DIR)/Likelihood_Cosmology.o
DATAMODULES += $(OUTPUT_DIR)/CFHTLens.o
```

and then recompile the code (the new likelihood can be added to a parameter file with `DEFAULT(batch2/CFHTLens.ini)`. Note the subtlety that we used `LogLike` rather than `LogLikeTheory` since we needed to pass `Theory`, which is the structure that includes the value of σ_8.

Extra exercise: Background parameterization

In particular cases, one does not really need to use the full Planck likelihood, and the parameter space to be constrained differs from ΛCDM (for example, if the data sets to be used can only constrain the distance-redshift relation and they do not have any information about e.g. the primordial power spectrum).

In that case, one can simplify things by using `parameterization=background`. This replaces the usual ΛCDM parameters $\{\Omega_b h^2,\ \Omega_m h^2,\ \theta_*,\ \ln 10^{10} A_s,\ n_s,\ \tau\}$ with a parameter space described by the base parameters $\{\Omega_m,\ H_0\}$, with the possibility of adding $\{\Omega_K,\ \Sigma m_\nu,\ w,\ w_a,\ N_{\rm eff}\}$. The goal of this exercise is to combine the techniques in Exercise 4 and Exercise 5 to derive constraints on θ_* using Planck distance priors (as opposed to using the full likelihood of Planck).

As in the previous exercise, we will define a likelihood. In this case we are going to implement the Planck distance priors of Wang & Dai (2015)[7]. This likelihood `source/Planckprior.f90` should look like this:

```
module Planckprior
 use CosmologyTypes
 use Likelihood_Cosmology
 use MatrixUtils
 implicit none
 private

 real(mcp) :: CMB_ombh2, CMB_omc, CMB_omb, CMB_omnu, CMB_omk, CMB_omv, CMB_w, CMB_wa, CMB_h

 type, extends(TCosmoCalcLikelihood) :: PlanckpriorLikelihood
     real(mcp),dimension(4,4) :: datacov
     real(mcp),dimension(4)   :: dataval,dataerr
 contains
 procedure :: LogLikeTheory => Planckprior_LnLike
 end type PlanckpriorLikelihood


 public PlanckpriorLikelihood, PlanckpriorLikelihood_Add
 contains

 subroutine PlanckpriorLikelihood_Add(LikeList, Ini)
 class(TLikelihoodList) :: LikeList
 class(TSettingIni) :: ini
 Type(PlanckpriorLikelihood), pointer :: this
 character(LEN=:), allocatable :: string
 Type(TTextFile) :: F
 integer i,iopb

 if (Ini%Read_Logical('use_Planckprior',.false.)) then
     allocate(this)
     this%LikelihoodType = 'Planckprior'
     this%name= Ini%Read_String('Planckprior_name')
     string = Ini%Read_String('shiftparams_values')
     read (string,*) this%dataval
     string = Ini%Read_String('shiftparams_errors')
     read (string,*) this%dataerr
     string = Ini%ReadRelativeFileName('correlationmatrix_file')
     call F%Open(string)
     read (F%unit,*, iostat=iopb) this%datacov
     if (iopb /= 0) call MpiStop('Planckprior: Error reading Planckprior correlation matrix: ' &
            //trim(this%name))
     call F%Close()
     do i = 1,4
          this%datacov(i,:)=this%datacov(i,:)*(this%dataerr(i)*this%dataerr(:))
     enddo
     call matrix_inverse(this%datacov)
     this%needs_background_functions = .true.
     call LikeList%Add(this)
 end if

 end subroutine PlanckpriorLikelihood_Add
```

[7] http://arxiv.org/abs/1509.02198

```
real(mcp) function rint(x)
real(mcp), INTENT(IN) :: x
real(mcp) :: Rb, Tcmb, omegam, zeq
Tcmb = 2.7255
Rb = 31500*CMB_ombh2*(Tcmb/2.7)**(-4.)
omegam = CMB_omc + CMB_omb + CMB_omnu
zeq = 2.5e4*(omegam*CMB_h**2)*(Tcmb/2.7)**(-4.)
rint = 1.d0/sqrt(3.*(1.+Rb*x)*(omegam*(x+1./(1.+zeq))+CMB_omk*x**2+ &
       CMB_omv*x**4.*x**(-3.0*(1.0+CMB_w+CMB_wa))*exp(3.0*CMB_wa*(x-1.0))))
end function rint

real(mcp) function Planckprior_LnLike(this, CMB)
Class(PlanckpriorLikelihood) :: this
Class(CMBParams) CMB
real(mcp), dimension(4) :: theoryval
real(mcp) :: obh2, omh2, g1, g2, zstar, rzstar, rszstar, theta_zstar
real(mcp)  :: rombint
external rombint
common /deriv/ theta_zstar

CMB_ombh2 = CMB%ombh2
CMB_omc = CMB%omc
CMB_omb = CMB%omb
CMB_omnu = CMB%omnu
CMB_omk = CMB%omk
CMB_omv = CMB%omv
CMB_w = CMB%w
CMB_wa = CMB%wa
CMB_h = CMB%h

obh2=CMB%ombh2
omh2=CMB%omch2+CMB%ombh2+CMB%omnuh2
g1=(0.0783*(obh2)**(-0.238))/(1.+39.5*(obh2)**(+0.763))
g2=0.560/(1.+21.1*(obh2)**(+1.81))
zstar = 1048.*(1.+0.00124*(obh2)**(-0.738))*(1.+g1*(omh2)**g2)
rzstar=this%Calculator%AngularDiameterDistance(zstar)*(1.+zstar)
rszstar=(299792.458/CMB%H0)*rombint(rint,0.d0,1./(1.+zstar),1.0d-5)
theta_zstar = 100.*rszstar/rzstar

theoryval(1)=pi*rzstar/rszstar
theoryval(2)=100.*sqrt(omh2)*rzstar/299792.458
theoryval(3)=CMB%ombh2
theoryval(4)=CMB%InitPower(ns_index)

! the parameter space used here  does not actually read ombh2 or ns so I only use the 2x2 matrix (1a,R)
Planckprior_LnLike = dot_product((this%dataval(1:2)-theoryval(1:2)), &
                     matmul(this%datacov(1:2,1:2),(this%dataval(1:2)-theoryval(1:2))) )/2.0d0
end function  Planckprior_LnLike

end module Planckprior
```

The corresponding data file `batch2/Planckprior.ini` would then be:

```
# Values from Wang & Dai (2015) http://arxiv.org/abs/1509.02198
use_Planckprior = T
lmin_store_all_cmb = 0
Planckprior_name = WangDai2015
shiftparams_values     = 301.76 1.7474 0.02228 0.9659
shiftparams_errors     = 0.0930 0.0051 0.00016 0.0048
correlationmatrix_file = data/WangDai2015.corr
```

where `data/WangDai2015.corr` is a text file containing the 4x4 matrix given by eq.13 in that paper.

```
+1.0000 +0.4529 -0.3507 -0.3576
+0.4529 +1.0000 -0.7000 -0.7780
-0.3507 -0.7000 +1.0000 +0.5296
-0.3576 -0.7780 +0.5296 +1.0000
```

As in the previous exercise, we need to add the new likelihood to `source/DataLikelihoods.f90`:

```
use Planckprior
call PlanckpriorLikelihood_Add(DataLikelihoods, Ini)
```

And add the new source file to `source/Makefile`:

```
$(OUTPUT_DIR)/Planckprior.o: $(OUTPUT_DIR)/Likelihood_Cosmology.o
DATAMODULES += $(OUTPUT_DIR)/Planckprior.o
```

Before recompiling, we need to do minor edits of some other files. For example, in `source/CosmologyParameterizations.f90` we need to replace the line `CMB%omb=omegam - CMB%omnu` with a line assigning a fixed value e.g. `CMB%omb=0.04`[8] and, in order to avoid problems with CAMB, replace the line `CMB%omc=0` with `CMB%omc = omegam - CMB%omb - CMB%omnu`.

We need a file to set defaults for our new parameterization. In order to do this, we can just make a copy of `batch2/params_CMB_defaults.ini` into `batch2/params_background_defaults.ini` and comment out or remove the lines starting with `param[omegabh2]`, `param[omegach2]`, `param[theta]`, `param[tau]`, `param[ns]`, `param[logA]` (i.e. the ones corresponding to the ΛCDM parameters) and instead add two lines for `param[omegam]` and `param[H0]`.

Also, since we want to add `theta_zstar` as a derived parameter, we need to include it in `paramnames/params_background.paramnames` and also modify the function `BK_CalcDerivedParams` in `source/CosmologyParameterizations.f90`:

```
subroutine BK_CalcDerivedParams(this, P, Theory, derived)
class(BackgroundParameterization) :: this
real(mcp), allocatable :: derived(:)
class(TTheoryPredictions), allocatable :: Theory
real(mcp) :: P(:)
Type(CMBParams) CMB
real(mcp) theta_zstar
common /deriv/ theta_zstar

allocate(Derived(2))
call this%ParamArrayToTheoryParams(P,CMB)

derived(1) = CMB%omv
derived(2) = theta_zstar

end subroutine BK_CalcDerivedParams
```

And finally, the new likelihood can be added to a parameter file including the following lines (remember that the order is important!):

```
DEFAULT(batch2/Planckprior.ini)
DEFAULT(batch2/params_background_defaults.ini)
parameterization = background
```

also change `get_sigma8=T` to `get_sigma8=F`.

[8] Alternatively, you could add the parameter `omegab` as an extra parameter to `paramnames/params_background.paramnames` and set instead `CMB%omb=Params(n)` where n is the position of `omegab` in the paramnames file.

Acknowledgements

This tutorial would not have been possible without the support of the organizers of the 1st Mexican AstroCosmoStatistics School, held in April 17-21 in León, Guanajuato, Mexico. In particular I would like to thank Mariana Vargas-Magaña (IF-UNAM), Alma González-Morales (CONACYT/UGTO), and Iván Rodríguez Montoya (INAOE) for organizing this successful school. AJC is supported by the European Research Council under the European Community's Seventh Framework Programme FP7-IDEAS-Phys.LSS 240117 and the Spanish MINECO under projects AYA2014-58747-P and MDM-2014-0369 of ICCUB (Unidad de Excelencia 'María de Maeztu'). My deepest admiration goes to my students during this School: Miguel Ángel de Icaza Lizaola, Efrain Torres Lomas, Luis Enrique Padilla Albores, David Fernandez Arenas, Jairo Andrés Alzate Trujillo, Fidel Sosa Nuñez, Gabriela Alejandra Aguilar Argüello, Óscar Nicolás Gómez Giraldo, Metin Ata, and Bruno Villaseñor Álvarez, for their impressive work while completing this tutorial.

References

[Lewis and Bridle, 2002] Lewis, A. and Bridle, S. (2002). Cosmological parameters from CMB and other data: A Monte Carlo approach. *Phys. Rev.*, D66:103511.