CosmoMC

CosmoMC

Cosmological Monte Carlo

Main author: Antony Lewis

Documentation: README, plotting and analysis README

Main references:
Lewis, Bridle 2002
Lewis 2013

Outline

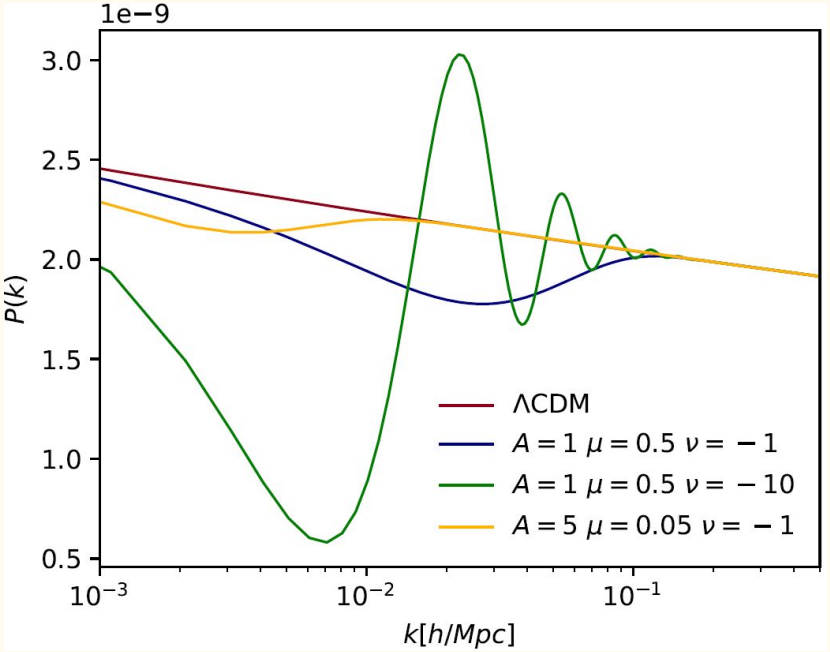
1. The black box approach: compile, running, analyzing chains
2. Exercise: run a simple chain, analyze pre made chains
3. A glimpse at the structure: parametrizations, CAMB interface
4. Exercise: add modification of CAMB to cosmomc

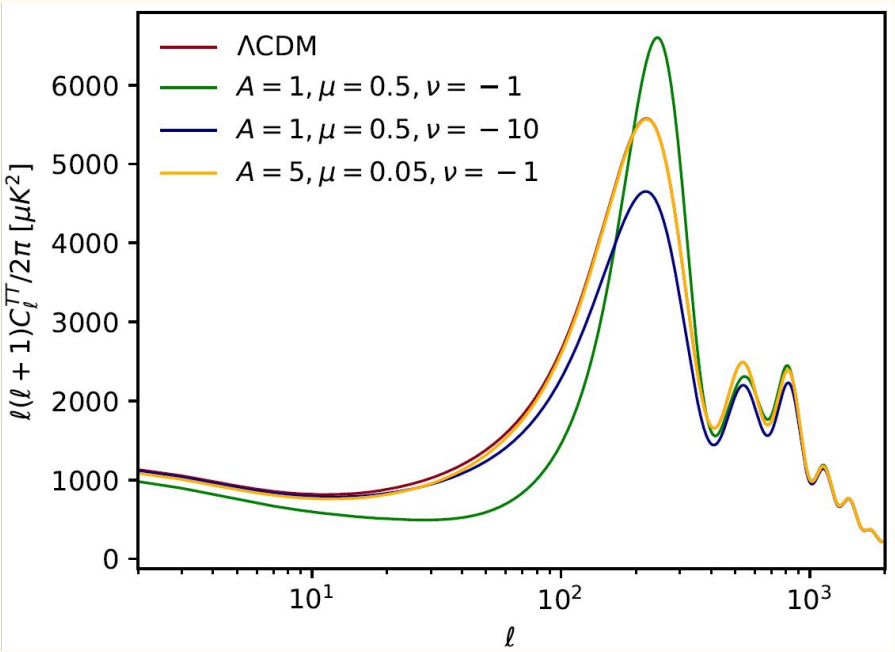
   If we have enough time...

5. More on the structure: likelihoods
6. Exercise: write a new likelihood

The Black Box Approach

Why CosmoMC?





Can we fit the low-l tail without spoiling everything else?

What's CosmoMC?

CosmoMC is a Markov-Chain Monte-Carlo engine used to perform cosmological parameter estimation.

Language: fortran + python wrapper

The code calls CAMB and performs MCMC over user selected parameters.

It calls CAMB for each sampled point of the parameter space, obtains cosmological predictions for that specific point and compares it to observations

Downloading

CosmoMC is available at http://cosmologist.info/cosmomc/ or on github

Get the code with:

> git clone https://github.com/cmbant/CosmoMC.git

CosmoMC has several cosmological likelihoods built-in.

Extra download needed to obtain the Planck likelihood (check Planck readme)

Compiling

CosmoMC Makefile runs compilation both for CosmoMC and the associated CAMB

source/Makefile can choose CAMB options:

```
camb:
	cd ../camb && \
	$(MAKE) --file=Makefile_main libcamb OUTPUT_DIR=$(OUTPUT_DIR) \
	RECOMBINATION=$(RECOMBINATION) EQUATIONS=equations_ppf NONLINEAR=halofit_ppf
```

Compile!

Schematic structure

batch1/2/3:
Contains .ini files with settings for the available likelihoods.

data:
Contains the actual datasets

camb:
Contains the camb code (or modifications of it)

python:
Python scripts used to run chains, make parameter grids, analysis....

source:
Source code of CosmoMC

Setup

Following the provided test.ini, the necessary options for cosmological and settings parameters are given in:

- test.ini
- batchX/common.ini
- batchX/params_CMB_defaults.ini

General ini file (test.ini)

- Experiments:
  Decides which likelihoods will be considered, calling likelihoods ini files

- General settings:
  Calls general options file (default common.ini)

- CosmoMC options:
  chains names
  actions
  method
  feedback

```
#general settings
#Bicep-Keck-Planck, varying cosmological parameters
#DEFAULT(batch2/BKPlanck.ini)

#Planck 2015, default just include native likelihoods (others require clik)
DEFAULT(batch2/plik_dx11dr2_HM_v18_TT.ini)
DEFAULT(batch2/lowTEB.ini)
#DEFAULT(batch2/lowl.ini)
#DEFAULT(batch2/lensing.ini)

#Other Likelihoods
#DEFAULT(batch2/BAO.ini)
#DEFAULT(batch2/WiggleZ_MPK.ini)
#DEFAULT(batch2/MPK.ini)
#DEFAULT(batch2/WL.ini)
```

Choose data to use

```
#general settings
DEFAULT(batch2/common.ini)

#e.g. to vary r in addition to standard 6:
#(for r>0 also need compute_tensors=T)
#compute_tensors = T
#param[r] = 0.03 0 2 0.04 0.04

#high for new runs
MPI_Max_R_ProposeUpdate = 30

propose_matrix= planck_covmats/base_TT_lowTEB_plik.covmat

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

#Root name for files produced
file_root=pk15_features
#action= 0 runs chains, 1 importance samples, 2 minimizes
#use action=4 just to quickly test likelihoods
action = 0

#expected result for -(log like)
test_check_compare = 28.337

num_threads = 0

#if you want to get theory cl for test point
#test_output_root = output_cl_root

start_at_bestfit =F
feedback=2
use_fast_slow = T

checkpoint = F

#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
```

Parameters and options

Choose what to do!

If T allows to restart chains when they stop

Base sampling method is Metropolis-Hastings (sampling_method=1). If sampling_method=7 handles efficiently “fast” nuisance parameters

Likelihood ini files

These files contains:

- flags needed for cosmomc to use specific likelihoods
- Path of datasets
- options

```
use_mpk = T
use_wigglez_mpk = T

#if true, use HALOFIT for non-linear corrections (astro-ph/0207664).
nonlinear_wigglez = T

#use gigglez corrections; only set Use_gigglez = T if nonlinear_pk = T
Use_gigglez = T

#File with common Settings for all datasets
wigglez_common_dataset = data/wigglez_nov11_common.dataset

#filenames for WiggleZ data only
mpk_wigglez_numdatasets = 4
wigglez_dataset1 = data/wigglez_nov11a.dataset
wigglez_dataset2 = data/wigglez_nov11b.dataset
wigglez_dataset3 = data/wigglez_nov11c.dataset
wigglez_dataset4 = data/wigglez_nov11d.dataset
~
~
```

General settings

- common.ini
  generally doesn't need to be modified. Contains general mcmc settings

- params_CMB_default.ini
  Contains parameter values and/or variation range
  Modifying this changes which parameters are actually sampled by the MCMC

```
#to vary parameters set param[name]= center, min, max, start width, propose width
#for fixed can just fix fixed value

param[omegabh2] = 0.0221 0.005 0.1 0.0001 0.0001
param[omegach2] = 0.12 0.001 0.99 0.001 0.0005
param[theta] = 1.0411 0.5 10 0.0004 0.0002
param[tau] = 0.09 0.01 0.8 0.01 0.005

neutrino_hierarchy = degenerate
num_massive_neutrinos=1
param[mnu] = 0.06
param[meffsterile] = 0

param[omegak] = 0
param[w] = -1
param[nrun] = 0
param[nrunrun] = 0
param[r] = 0
```

Sampled parameters

Fixed parameters

Grid of models

If you have to do many parameter files, e.g. with different data combination or with different free parameters, things might get boring.

CosmoMC contains useful python script to generate a grid of model, e.g.

> python python/makeGrid.py planck_grid settings_planck_2015

Generates a grid of .ini files with all the analysis done by Planck 2015

A planck_grid folder is created, with a bunch of ini files and ready to host your chains

Run the code

Single chain:

> ./cosmomc test.ini

MPI jobs:

> mpirun -np 4 ./cosmomc test.ini

Python:

> python python/runMPI.py [options] test.ini

The output

Run test.ini after switching to action=0

> ./cosmomc test.ini

The chain folder will now contain the typical output files

Output files

- The program produces **file_root_1.txt**, **file_root_2.txt** etc, files listing each accepted set of parameters for each chain; the first column gives the number of iterations staying at that parameter set (more generally, the sample weight), and the second the likelihood.
- A **file_root.paramnames** file lists the names and labels of the parameters corresponding to the columns 3+ of the output chain files. By default this is just a copy of the params_CMB.paramnames file read in by the default parameterization specified in params_CMB.f90. Parameter names are strings of numbers and letters without spaces, used reference parameters in input files; the labels are used when making plot axes labels and various output files.
- A **file_root.ranges** lists the name tags of the parameters, and their upper and lower bounds (including parameters that are not varied, where upper and lower are equal to the fixed value)
- Files **file_root_1.dat**, **file_root_2.dat**, etc., if **indep_sample** is non-zero. These contain full computed model information at the independent sample points (power spectra etc). These can be used for quick importance sampling using action=1.
- A **file_root.log** file contains some info which may be useful to assess performance.
- For post-processing (action=1), the program reads in an existing .data file and processes according to the redo_xxx parameters. At this point the acceptance multiplicities are non-integer, and the output is already thinned by whatever the original **indep_sample** parameter was. The post-processed file are output to files with root **redo_outroot**.

What does a chain look like?

Multiplicity and likelihood

Minimal ΛCDM parameters

This is a single line

```
  0.2000000E+01   0.6035613E+04   0.2206191E-01   0.1209273E+00   0.1040932E+01   0.8500430E-01   0.3100886E+01   0.9611985E+00  -0.6169144E-01   0.1048932E+01   0.1240018E+00   0.1002521E+01   0.6659496E+02
0.9432532E-02   0.3215227E+01   0.2685844E+03   0.3629809E+02   0.3771139E+02   0.1196200E+03   0.7998695E+00   0.7428712E+01   0.9093039E+01   0.1479897E+02   0.1045109E+03   0.1000863E+01   0.9933816E+00   0.6
677979E+02   0.6779167E+00   0.3220833E+00   0.1436343E+00   0.6451439E-03   0.9591869E-01   0.8376820E+00   0.4754045E+00   0.6310608E+00   0.1025077E+01   0.2534577E+01   0.1069794E+02   0.2221763E+01   0.1874
407E+01   0.1242316E+04   0.5679107E+04   0.2517952E+04   0.8072349E+03   0.2280303E+03   0.9611985E+00   0.2452493E+00   0.2465753E+00   0.2649949E+01   0.1383213E+02   0.1090395E+04   0.1444272E+03   0.1041143
E+01   0.1387199E+02   0.1059284E+04   0.1471906E+03   0.1405234E+00   0.1611620E+00   0.3417006E+04   0.1042903E-01   0.8098303E+00   0.4477984E+00   0.7103850E-01   0.9264928E+02   0.1398791E+04   0.6786939E+0
0   0.4897231E+00   0.6205758E+00   0.1564901E+04   0.1049769E+05   0.8635542E+01   0.1206259E+05
  0.1000000E+01   0.5911105E+04   0.2204734E-01   0.1208468E+00   0.1040936E+01   0.8494210E-01   0.3100746E+01   0.9615178E+00  -0.5940046E-01   0.1039358E+01   0.1261409E+00   0.1002628E+01   0.6024200E+02
0.2235632E+00   0.1476084E+01   0.2545640E+03   0.3278571E+02   0.3032620E+02   0.1147198E+03   0.8576056E+01   0.7963180E+01   0.6005200E+01   0.1494029E+02   0.1032913E+03   0.1000338E+01   0.9932647E+00   0.6
679932E+02   0.6783181E+00   0.3216819E+00   0.1435392E+00   0.6451439E-03   0.9588324E-01   0.8375279E+00   0.4750208E+00   0.6307481E+00   0.1024739E+01   0.2533010E+01   0.1069489E+02   0.2221451E+01   0.1874
377E+01   0.1241559E+04   0.5678037E+04   0.2518423E+04   0.8074068E+03   0.2280568E+03   0.9615178E+00   0.2452422E+00   0.2465682E+00   0.2652771E+01   0.1383295E+02   0.1090406E+04   0.1444590E+03   0.1041155
E+01   0.1387488E+02   0.1059246E+04   0.1472279E+03   0.1404724E+00   0.1611869E+00   0.3414733E+04   0.1042210E-01   0.8102004E+00   0.4480029E+00   0.7106302E-01   0.9264864E+02   0.1398595E+04   0.6785937E+0
0   0.4895348E+00   0.6205584E+00   0.1316411E+04   0.1049758E+05   0.8223240E+01   0.1181399E+05
  0.1000000E+01   0.5843892E+04   0.2217067E-01   0.1202529E+00   0.1041182E+01   0.8277377E-01   0.3097929E+01   0.9638512E+00  -0.4022295E-01   0.1048096E+01   0.1870182E+00   0.1004361E+01   0.5566711E+02
0.5290828E-01   0.6675662E+01   0.3322266E+03   0.4426662E+02   0.4169546E+02   0.1191312E+03   0.1521160E+01   0.4300914E+01   0.5124822E+01   0.2055573E+02   0.1142977E+03   0.1000223E+01   0.9928416E+00   0.6
718994E+02   0.6830899E+00   0.3169101E+00   0.1430687E+00   0.6451439E-03   0.9612776E-01   0.8348058E+00   0.4699519E+00   0.6263534E+00   0.1018435E+01   0.2517036E+01   0.1045074E+02   0.2215202E+01   0.1877
228E+01   0.1239220E+04   0.5698832E+04   0.2526456E+04   0.8112787E+03   0.2293777E+03   0.9638512E+00   0.2453030E+00   0.2466292E+00   0.2629067E+01   0.1381067E+02   0.1090194E+04   0.1445182E+03   0.1041391
E+01   0.1387742E+02   0.1059475E+04   0.1472484E+03   0.1405487E+00   0.1610700E+00   0.3403485E+04   0.1038778E-01   0.8127286E+00   0.4492392E+00   0.7131732E-01   0.9285798E+02   0.1392982E+04   0.6773976E+0
0   0.4867566E+00   0.6196951E+00   0.1176119E+04   0.1049662E+05   0.1504322E+02   0.1167274E+05
  0.1000000E+01   0.5810824E+04   0.2213408E-01   0.1199150E+00   0.1041224E+01   0.7527736E-01   0.3084997E+01   0.9602536E+00  -0.7947986E-01   0.1042583E+01   0.2440424E+00   0.1004931E+01   0.5438471E+02
0.1594658E+00   0.8892490E+01   0.3384830E+03   0.4598640E+02   0.5124210E+02   0.1275316E+03   0.1850700E+01   0.3411823E+01   0.3360027E+01   0.2070570E+02   0.1112897E+03   0.1000584E+01   0.9928609E+00   0.6
729248E+02   0.6848820E+00   0.3151180E+00   0.1426942E+00   0.6451439E-03   0.9602246E-01   0.8273200E+00   0.4644191E+00   0.6198574E+00   0.1008533E+01   0.2504994E+01   0.9770203E+01   0.2186740E+01   0.1881
101E+01   0.1248753E+04   0.5734117E+04   0.2530058E+04   0.8106984E+03   0.2287027E+03   0.9602536E+00   0.2452850E+00   0.2466111E+00   0.2636058E+01   0.1381168E+02   0.1090211E+04   0.1446334E+03   0.1041439
E+01   0.1388784E+02   0.1059399E+04   0.1473756E+03   0.1403833E+00   0.1611386E+00   0.3394533E+04   0.1036046E-01   0.8142594E+00   0.4500674E+00   0.7142327E-01   0.9287436E+02   0.1391808E+04   0.6769462E+0
0   0.4819422E+00   0.6145722E+00   0.1098854E+04   0.1049681E+05   0.2597974E+02   0.1159567E+05
  0.1000000E+01   0.5764229E+04   0.2223390E-01   0.1185915E+00   0.1041392E+01   0.7601880E-01   0.3081758E+01   0.9648976E+00  -0.8073054E-01   0.1096209E+01   0.2753688E+00   0.1003703E+01   0.5764549E+02
0.7934956E-01   0.9269340E+01   0.3002466E+03   0.4690390E+02   0.4901151E+02   0.1199873E+03   0.1472540E+01   0.4674600E+01   0.4749035E+01   0.2247169E+02   0.1088302E+03   0.1000691E+01   0.9927520E+00   0.6
789795E+02   0.6931312E+00   0.3068688E+00   0.1414706E+00   0.6451439E-03   0.9605562E-01   0.8226189E+00   0.4556959E+00   0.6122614E+00   0.9983214E+00   0.2476928E+01   0.9782410E+01   0.2179668E+01   0.1872
239E+01   0.1235894E+04   0.5722949E+04   0.2526328E+04   0.8113176E+03   0.2290775E+03   0.9648976E+00   0.2453328E+00   0.2466591E+00   0.2617051E+01   0.1379096E+02   0.1089967E+04   0.1449005E+03   0.1041598
E+01   0.1391136E+02   0.1059513E+04   0.1476176E+03   0.1402086E+00   0.1610666E+00   0.3365284E+04   0.1027122E-01   0.8199456E+00   0.4529662E+00   0.7188240E-01   0.9312563E+02   0.1383760E+04   0.6748525E+0
0   0.4770845E+00   0.6130958E+00   0.1013235E+04   0.1049578E+05   0.1944646E+02   0.1150901E+05
  0.1000000E+01   0.5743559E+04   0.2223253E-01   0.1184740E+00   0.1041400E+01   0.7471377E-01   0.3078440E+01   0.9649881E+00  -0.7686451E-01   0.1094643E+01   0.2709278E+00   0.1003546E+01   0.5953202E+02
0.1885591E-01   0.9739003E+01   0.2958580E+03   0.4329475E+02   0.4298090E+02   0.1149038E+03   0.1935556E+01   0.5285233E+01   0.6055051E+01   0.2334127E+02   0.1057007E+03   0.1000475E+01   0.9924572E+00   0.6
793701E+02   0.6937416E+00   0.3062584E+00   0.1413517E+00   0.6451439E-03   0.9603009E-01   0.8208835E+00   0.4542820E+00   0.6106657E+00   0.9959288E+00   0.2471516E+01   0.9660339E+01   0.2172448E+01   0.1870
914E+01   0.1234587E+04   0.5721263E+04   0.2525175E+04   0.8109433E+03   0.2289194E+03   0.9649881E+00   0.2453322E+00   0.2466585E+00   0.2617310E+01   0.1379072E+02   0.1089959E+04   0.1449322E+03   0.1041594
E+01   0.1391447E+02   0.1059513E+04   0.1476495E+03   0.1401731E+00   0.1610715E+00   0.3362441E+04   0.1026255E-01   0.8204556E+00   0.4532333E+00   0.7191683E-01   0.9313580E+02   0.1383289E+04   0.6746965E+0
0   0.4759175E+00   0.6119533E+00   0.9739247E+03   0.1049560E+05   0.1758877E+02   0.1146953E+05
  0.2000000E+01   0.5720269E+04   0.2224755E-01   0.1180015E+00   0.1041187E+01   0.7770797E-01   0.3086366E+01   0.9650359E+00   0.2394186E-01   0.1136036E+01   0.2473434E+00   0.1004866E+01   0.6983785E+02
0.1960978E+00   0.9254166E+01   0.3807917E+03   0.4389392E+02   0.3937379E+02   0.9267191E+02   0.2438651E+01   0.3311178E+01   0.7137791E+01   0.2514468E+02   0.9590660E+02   0.1000737E+01   0.9919344E+00   0.6
804932E+02   0.6957395E+00   0.3042605E+00   0.1408942E+00   0.6451439E-03   0.9587755E-01   0.8223483E+00   0.4536058E+00   0.6107553E+00   0.9968824E+00   0.2477271E+01   0.9919739E+01   0.2189737E+01   0.1874
544E+01   0.1239754E+04   0.5747000E+04   0.2532152E+04   0.8128158E+03   0.2294356E+03   0.9650359E+00   0.2453389E+00   0.2466652E+00   0.2614464E+01   0.1379324E+02   0.1089898E+04   0.1450440E+03   0.1041388
E+01   0.1392795E+02   0.1059513E+04   0.1477589E+03   0.1400702E+00   0.1610338E+00   0.3351507E+04   0.1022919E-01   0.8223347E+00   0.4541977E+00   0.7201545E-01   0.9314731E+02   0.1382069E+04   0.6741850E+0
0   0.4762374E+00   0.6135423E+00   0.9236190E+03   0.1049595E+05   0.2097176E+02   0.1141957E+05
```

GetDist: analyzing chains

Once your chains have run for some time you can assess their convergence and analyze them.

>python python/GetDist.py [distparams.ini] /path_to_chain/chain_root

```
novamaris [1058] $ python python/GetDist.py chains/pk15_features
producing files in directory  output/
chains/pk15_features_1.txt
chains/pk15_features_2.txt
chains/pk15_features_3.txt
chains/pk15_features_4.txt
chains/pk15_features_5.txt
chains/pk15_features_6.txt
chains/pk15_features_7.txt
chains/pk15_features_8.txt
Number of chains used =  8
 var(mean)/mean(var), remaining chains, worst e-value: R-1 =       0.13218
RL: Not enough samples to estimate convergence stats
using 27152 rows, 69 parameters; mean weight 6.40932527991, tot weight 174026.0
Approx indep samples (N/corr length): 315.0
Equiv number of single samples (sum w)/max(w): 667.0
Effective number of weighted samples (sum w)^2/sum(w^2): 10222
Best fit sample -log(Like) = 5631.682000
mean(-Ln(like)) = 5640.102223
-Ln(mean like)  = 5636.710777
```

GetDist options
Default is in
python/getdist/analysis_defaults.ini

MCMC convergence:
Gelman and Rubin statistics,
i.e. "variance of chain means"/"mean of chain variances" (-1)

GetDist products

- .covmat
  Covariance matrix you can use as input for future analysis
- .corr
  Correlation between parameters
- .converge
  Convergence diagnostics
- .likestats
  Best fit point, with its likelihood. Warning: action=0 not accurate
- **.margestats**
  Marginalized values of parameters, with standard deviation and limits.
  These are the values that you usually quote in papers!

Exercise(s):

Run and analyze chains

Create a grid and start a simple chain

Use the python scripts to create a grid of models to be tested

Copy the settings_sample.py and change it to include:

- Use BAO and JLA datasets
- Include w and wa on top of non standard parameters

Run one of the produced ini files

settings_sample

```
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 = [[], ['mnu'], ['nnu']]

g.datasets = []

# lists of dataset names to combine, with corresponding sets of inis to include
g.datasets.append(batchjob.dataSet(['plikHM', 'TT', 'lowTEB'], ['plik_dx11dr2_HM_v18_TT.ini', 'lowTEB.ini']))
g.datasets.append(batchjob.dataSet(['plikHM', 'TT', 'lowTEB', 'lensing'],
                                   ['plik_dx11dr2_HM_v18_TT.ini', 'lowTEB.ini', 'lensing.ini']))


# add importance name tags, and list of specific .ini files to include (in batch1/)
g.importanceRuns = []
g.importanceRuns.append([['BAO'], ['BAO.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['mnu'] = '0.02 0 5 0.1 0.03'
params['omegak'] = '-0.0008 -0.3 0.3 0.001 0.001'  # starting exactly on flat seems to confuse minimizer
params['nnu'] = '3.046 0.05 10 0.05 0.05'
params['nrun'] = '0 -1 1 0.005 0.001'
params['r'] = '0 0 3 0.03 0.03'
params['Alens'] = '1 0 10 0.05 0.05'
params['yhe'] = '0.245 0.1 0.5 0.006 0.006'
params['alpha1'] = '0 -1 1 0.0003 0.0003'
params['meffsterile'] = '0.1 0 3 0.1 0.03'


# extra parameters that are set only when specific parameters are varied. Can deleted to get defaults.
param_extra_opts = {
    'mnu': {'num_massive_neutrinos': 3},
    'meffsterile': {'param[mnu]': '0.06', 'param[nnu]': '3.1 3.046 10 0.05 0.05', 'num_massive_neutrinos': 1,
                    'accuracy_level': 1.2},
    'yhe': {'bbn_consistency': False},
    'r': {'compute_tensors': True},
    'nt': {'inflation_consistency': False, 'lmax_tensor': 1000}
}
```

Default ini files

Parameter combinations

Dataset names and .ini to use

Importance datasets

Extra parameters

Analyze chains: the GUI

Launch the GUI

> python python/GetDistGUI.py

Features:

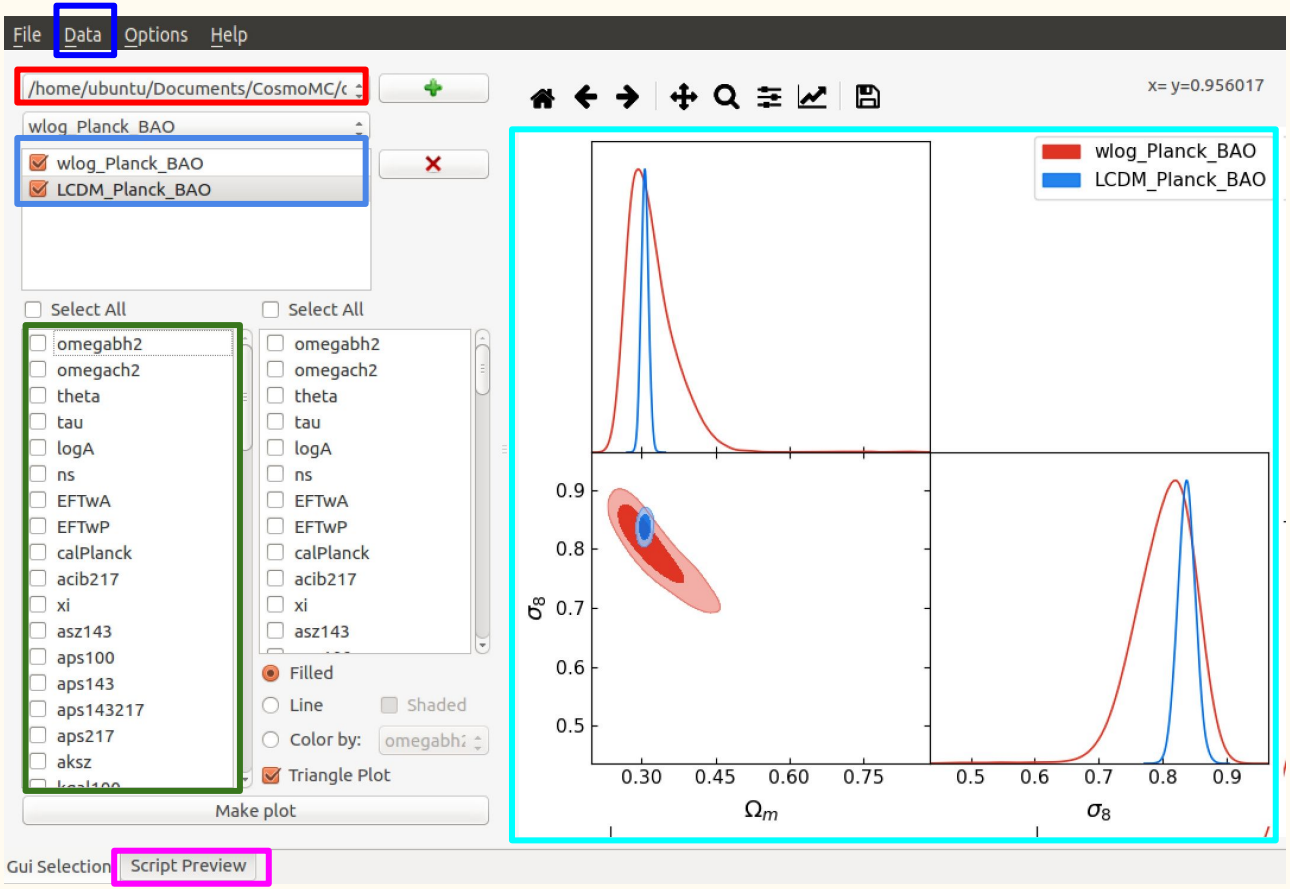
- Make plots: 1D, 2D and triangle
- Get convergence and marginalized statistics
- Copy the script for GetDist

Get statistics

Path to chains

Selected chains

List of parameters



The plot

Python script

Exercise(s):

Use the GUI to analyze chains

Use the GUI

You can use the chains provided in google drive for LCDM and the Log w parametrization.

- What is the convergence (R-1) of the chains?
- Look at H_0 and Ω_b h^2, what are the mean values and the 2σ bounds?
- Make a 2D plot using the chains from two different runs.
- Make a triangular plot with 6 parameters of your choice (you may have to set the Tight Layout in order to see the plot properly)
- Copy the script and run it with python. Remember to add the getdist path to your script!

```
import os, math, sys

here = os.path.dirname(os.path.abspath(__file__))
getdist_python_path = here+'/python/'
sys.path.insert(0, os.path.normpath(getdist_python_path))
```

CosmoMC's structure

Parameters and CAMB interface

Parameters and CAMB interface

CosmoMC is interfaced with standard CAMB.

A common change one usually does to CosmoMC is to include a modified CAMB. Using it with a modified version requires adapting this interface.

The relevant source files to change for this are

- CosmologyTypes.f90 -> define new parameters
- CosmologyParametrizations.f90 -> read new parameters and define new derived
- Calculator_CAMB.f90 -> pass the new parameters to the modified CAMB
- .paramnames files -> A dictionary for CosmoMC

CosmologyTypes

```
module CosmologyTypes
use settings
use likelihood
use GeneralTypes
use ObjectLists
implicit none

integer, parameter :: derived_age=1, derived_zstar=2, derived_rstar=3, derived_thetastar=4, derived_DAstar = 5, &
    derived_zdrag=6, derived_rdrag=7,derived_kD=8,derived_thetaD=9, derived_zEQ =10, derived_keq =11, &
    derived_thetaEQ=12, derived_theta_rs_EQ = 13 !index in derived parameters array

integer, parameter :: As_index=1, ns_index =2, nrun_index=3, nrunrun_index=4, amp_ratio_index = 5, &
    & nt_index= 6, ntrun_index = 7, Aphiphi_index = 8, Aini_index = 9, muini_index = 10, nuini_index = 11, &
    & last_power_index = nuini_index !MMmod additional parameters

integer, parameter :: max_inipower_params = 13 !MMmod 10+3 additional parameters

real(mcp), parameter :: cl_norm = 1e-10_mcp !units for As
integer, parameter :: max_derived_parameters = 30

integer :: num_hard, num_initpower
integer :: index_initpower
```

Initial power parameters defined as elements of a general array

If you include new parameters check that you are below the maximum or increase it

```
!set number of hard parameters, number of initial power spectrum parameters
call this%SetTheoryParameterNumbers(16,last_power_index)

end subroutine TP_Init
```

Global number of parameters

```
Type, extends(TTheoryParams) :: CMBParams
    real(mcp) InitPower(max_inipower_params)
    !These are fast paramters for the initial power spectrum
    !Now remaining (non-independent) parameters
    real(mcp) omb, omc, omv, omnu, omk, omdm
    real(mcp) ombh2, omch2, omnuh2, omdmh2
    real(mcp) zre, zre_delta, nufrac
    real(mcp) h, H0, tau
    real(mcp) w, wa
    real(mcp) YHe, nnu, iso_cdm_correlated, ALens, Alensf, fdm !fdm is dark matter annihilation, eg,. 0910.3663
    real(mcp) :: omnuh2_sterile = 0._mcp  !note omnhu2 is the sum of this + standard neutrinos
    real(mcp) :: sum_mnu_standard
    real(mcp) reserved(5)
end Type CMBParams
```

Other parameters defined in variables

CosmologyParametrization

CosmoMC standard parametrization is slightly different from CAMB (θ instead of H0)

```
subroutine SetForH(Params,CMB,H0, firsttime,error)
use bbn
real(mcp) Params(num_Params)
logical, intent(in) :: firsttime
Type(CMBParams) CMB
real(mcp) h2,H0
integer, optional :: error

CMB%H0=H0
if (firsttime) then
    CMB%reserved = 0
    CMB%ombh2 = Params(1)
    CMB%tau = params(4) !tau, set zre later
    CMB%Omk = Params(5)
    CMB%w = Params(8)
    CMB%wa = Params(9)
    CMB%nnu = Params(10) !3.046
    !Params(6) is now mnu, where mnu is physical standard neutrino mass and we assu
    CMB%sum_mnu_standard = Params(6)
    CMB%omnuh2=Params(6)/neutrino_mass_fac*(standard_neutrino_neff/3)**0.75_mcp
    !Params(7) is mass_sterile*Neff_sterile
    CMB%omnuh2_sterile = Params(7)/neutrino_mass_fac
    !we are using interpretation where there are degeneracy_factor neutrinos, each
    !So internally 3.046 or 3.046/3 massive neutrnos. But mnu is the physical integ
    if (CMB%omnuh2_sterile >0 .and. CMB%nnu < standard_neutrino_neff) then
        if(present(error))then
            error=-1
        else
            call MpiStop('sterile neutrino mass required Neff>3.046')
        end if
    end if

    CMB%omnuh2 = CMB%omnuh2 + CMB%omnuh2_sterile
    CMB%omch2 = Params(2)
    CMB%omdmh2 = CMB%omch2+ CMB%omnuh2
    CMB%nufrac=CMB%omnuh2/CMB%omdmh2

    if (CosmoSettings%bbn_consistency) then
        CMB%YHe = BBN_YHe%Value(CMB%ombh2,CMB%nnu - standard_neutrino_neff,error)
    else
        !e.g. set from free parameter..
        CMB%YHe  =Params(11)
    end if

    CMB%iso_cdm_correlated =  Params(12)
    CMB%zre_delta = Params(13)
    CMB%ALens = Params(14)
    CMB%ALensf = Params(15)
    CMB%fdm = Params(16)
    call SetFast(Params,CMB)
end if

CMB%h = CMB%H0/100
h2 = CMB%h**2
CMB%omb = CMB%ombh2/h2
CMB%omc = CMB%omch2/h2
CMB%omnu = CMB%omnuh2/h2
CMB%omdm = CMB%omdmh2/h2
CMB%omv = 1- CMB%omk - CMB%omb - CMB%omdm
```

This subroutine reads parameters and obtains H0 iteratively

Define derived parameters

Gets initial power params as “fast parameters”

```
subroutine SetFast(Para
real(mcp) Params(num_Pa
Type(CMBParams) CMB

CMB%InitPower(1:num_ini
CMB%InitPower(As_index)
write(*,*) 'Aini=',CMB%
end subroutine SetFast
```

```
subroutine TP_CalcDerivedParams(this, P, Theory, derived)
class(ThetaParameterization) :: this
real(mcp), allocatable :: derived(:)
class(TTheoryPredictions), allocatable :: Theory
real(mcp) :: P(:)
Type(CMBParams) CMB
real(mcp) :: lograt
integer ix,i
real(mcp) z
integer, parameter :: derivedCL(5) = [40, 220, 810, 1420, 2000]

if (.not. allocated(Theory)) call MpiStop('Not allocated theory!!!')
select type (Theory)
class is (TCosmoTheoryPredictions)
    allocate(Derived(this%num_derived), source=0._mcp)

    call this%ParamArrayToTheoryParams(P,CMB)

    derived(1) = CMB%H0
    derived(2) = CMB%omv
    derived(3) = CMB%omdm+CMB%omb
    derived(4) = CMB%omdmh2 + CMB%ombh2
    derived(5) = CMB%omnuh2
    derived(6) = (CMB%omdmh2 + CMB%ombh2)*CMB%h

    derived(7) = Theory%Sigma_8
    derived(8) = Theory%Sigma_8*((CMB%omdm+CMB%omb))**0.5_mcp
    derived(9) = Theory%Sigma_8*((CMB%omdm+CMB%omb))**0.25_mcp
    derived(10)= Theory%Sigma_8/CMB%h**0.5_mcp

    derived(11) = Theory%Lensing_rms_deflect
    derived(12) = CMB%zre
    ix=13
    derived(ix) = cl_norm*CMB%InitPower(As_index)*1e9
    derived(ix+1) = derived(ix)*exp(-2*CMB%tau)  !A e^{-2 tau}
    ix = ix+2
```

From CosmoMC to CAMB

The interface between CAMB and CosmoMC is in Calculator_CAMB.f90
Where CAMB parameters are set and
routines are called

```
subroutine CAMBCalc_CMBToCAMB(this,CMB,P)
use LambdaGeneral
use camb, only: CAMB_SetNeutrinoHierarchy
use CAMBmain, only : ALens
use constants, only : default_nnu,delta_mnu21,delta_mnu31,mnu_min_normal
use lensing, only : ALens_Fiducial
use MassiveNu, only : sum_mnu_for_m1
class(CAMB_Calculator) :: this
class(CMBParams) CMB
type(CAMBParams)  P
real(dl) neff_massive_standard, mnu, m1, m3, normal_frac
real(dl), external :: Newton_raphson

P = this%CAMBP
P%omegab = CMB%omb
P%omegan = CMB%omnu
P%omegac = CMB%omc
P%omegav = CMB%omv
P%H0 = CMB%H0
P%Reion%redshift= CMB%zre
P%Reion%delta_redshift = CMB%zre_delta
w_lam = CMB%w
wa_ppf = CMB%wa
ALens = CMB%ALens
ALens_Fiducial = CMB%ALensf
P%InitialConditionVector(initial_iso_CDM) = &
    sign(sqrt(abs(CMB%iso_cdm_correlated) /(1-abs(CMB%iso_cdm_correlated))),CMB%iso_cdm_correlated)
P%Num_Nu_Massive = 0
P%Nu_mass_numbers = 0
P%Num_Nu_Massless = CMB%nnu
P%share_delta_neff = .false.
if (CMB%omnuh2>0) then
    call CAMB_SetNeutrinoHierarchy(P, CMB%omnuh2, CMB%omnuh2_sterile, CMB%nnu, &
        CosmoSettings%neutrino_hierarchy, CosmoSettings%num_massive_neutrinos)
end if

P%YHe = CMB%YHe
#ifdef COSMOREC
    if (CMB%fdm/=0._mcp) P%Recomb%runmode = 3
    P%Recomb%fdm = CMB%fdm * 1e-23_mcp
#else
    if (CMB%fdm/=0._mcp) call MpiStop('Compile with CosmoRec to use fdm')
#endif
    call this%SetCAMBInitPower(P,CMB,1)
```

P% = CAMB params

Calls CAMB subroutine

```
subroutine CAMBCalc_SetPkFromCAMB(this,M,Theory,error)
use Transfer
use camb, only : CP
class(CAMB_Calculator) :: this
class(TCosmoTheoryPredictions) Theory
Type(MatterTransferData) M
integer :: error
real(mcp), allocatable :: k(:), z(:), PK(:,:)
integer zix,nz,nk, nR
real(mcp), allocatable :: NL_Ratios(:,:)
real(mcp) :: dR, R, minR
integer i

!Free theory arrays as they may resize between samples
call Theory%FreePK()

nz=CP%Transfer%PK_num_redshifts

if (.not. allocated(Theory%growth_z)) allocate(Theory%growth_z, Theory%sigma8_z)
call Theory%growth_z%InitForSize(nz)
call Theory%sigma8_z%InitForSize(nz)
allocate(z(nz))

do zix=1,nz
    z(zix) = CP%Transfer%PK_redshifts(nz-zix+1)
    Theory%sigma8_z%F(zix) = M%sigma_8(nz-zix+1,1)
    Theory%growth_z%F(zix) = M%sigma2_vdelta_8(nz-zix+1,1)/M%sigma_8(nz-zix+1,1)
end do
Theory%sigma8_z%X=z
Theory%growth_z%X=z

if (CosmoSettings%use_matterpower) then
    nk=M%num_q_trans
    nz=CP%Transfer%PK_num_redshifts
    allocate(PK(nk,nz))
    allocate(k(nk))

    k = log(M%TransferData(Transfer_kh,:,1))

    call Transfer_GetUnsplinedPower(M, PK,transfer_power_var,transfer_power_var)
    PK = Log(PK)
    if (any(ieee_is_nan(PK))) then
        error = 1
        return
    end if
    allocate(Theory%MPK)
    call Theory%MPK%Init(k,z,PK)
end if
```

Exercise:

Features in the primordial power spectrum (again)

Use the modified CAMB

What to do?

- Substitute your modified CAMB to the camb folder (maybe in another copy of CosmoMC)
- Include the additional parameters in cosmomc (where?), change the number of maximum parameters
- Add them to the params.ini

Start a single chain...
Check .ranges and and .paramnames to see everything is going fine

Tips

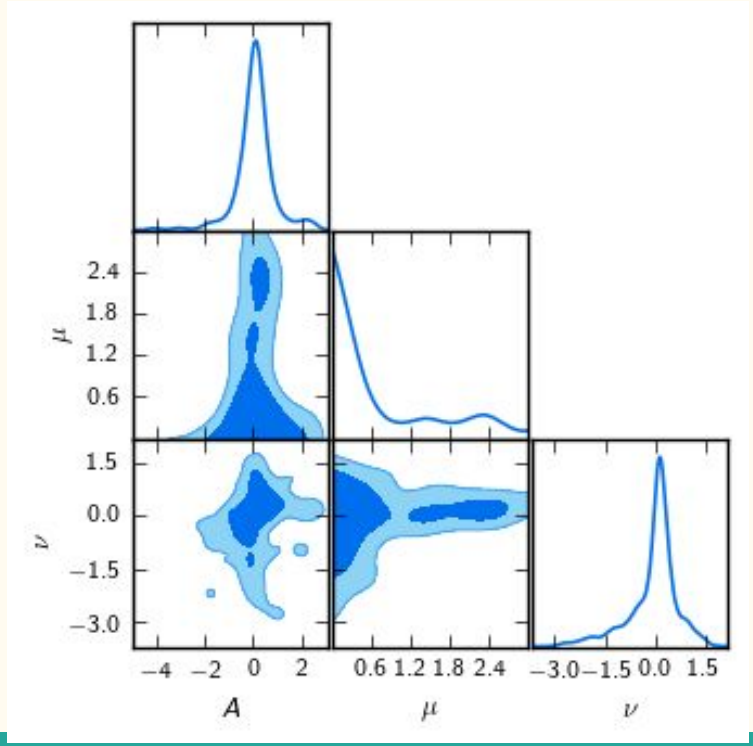
As for the CAMB modification, follow one of the existing parameters for the initial power spectrum, e.g. amp_ratio_index

- Increase the maximum index and define the new ones in CosmologyTypes.f90
- Add the new parameters in paramnames/params_CMB.paramnames
- Check that everything is fine in SetFast (CosmologyParametrization.f90)
- In Calculator_CAMB.f90 pass the new parameters to CAMB in the CAMBCalc_SetCAMBInitPower subroutine

Use the modified CosmoMC

If you have access to some cluster, try to install the Planck likelihood (readme here) and to run chains with this parametrization.
Probably this is not the best parametrization!

Chains not converged yet: R-1 ~0.5



CosmoMC's structure

likelihoods

Likelihood modules

CosmoMC calls different likelihood modules in DataLikelihoods.f90

Each subroutine checks if the flag to use that particular likelihood is set to true.

Some likelihood are called and then not added to the likelihood list.

```
module DataLikelihoodList
use likelihood
use settings
use CosmologyTypes
implicit none

contains

subroutine SetDataLikelihoods(Ini)
use HST
use snovae
use CMBLikelihoods
use bao
use mpk
use wigglez
use szcounts !Anna
use wl
use ElementAbundances
class(TSettingIni), intent(in) :: Ini

CosmoSettings%get_sigma8 = Ini%Read_Logical('get_sigma8',.false.)

call CMBLikelihood_Add(DataLikelihoods, Ini)

call AbundanceLikelihood_Add(DataLikelihoods, Ini)

call HSTLikelihood_Add(DataLikelihoods, Ini)

call SNLikelihood_Add(DataLikelihoods, Ini)

call MPKLikelihood_Add(DataLikelihoods, Ini)

if (use_mpk) call WiggleZLikelihood_Add(DataLikelihoods, Ini)

call BAOLikelihood_Add(DataLikelihoods, Ini)

call SZLikelihood_Add(DataLikelihoods, Ini) !Anna

call WLLikelihood_Add(DataLikelihoods, Ini)

end subroutine SetDataLikelihoods


end module DataLikelihoodList
```

Likelihood modules: an example

```
!! Hubble measurement constraint, based on corresponding angular diamter distance value and error

module HST
use CosmologyTypes
use Likelihood_Cosmology
implicit none
private

type, extends(TCosmoCalcLikelihood) :: HSTLikelihood
    real(mcp) :: angconversion = 11425.8d0
    !angconversion converts inverse of the angular diameter distance at z = zeff to H0
    !for the fiducial cosmology (omega_k = 0, omega_lambda = 0.7, w = -1)
    !likelihood is in terms of inverse of the angular diameter distance, so includes the tiny cosmological
    !dependence of the measurement (primarily on w) correctly.
    real(mcp) :: zeff = 0.04d0
    real(mcp) :: H0, H0_err
contains
procedure :: LogLikeTheory => HST_LnLike
end type HSTLikelihood

public HSTLikelihood, HSTLikelihood_Add
contains

subroutine HSTLikelihood_Add(LikeList, Ini)
class(TLikelihoodList) :: LikeList
class(TSettingIni) :: ini
Type(HSTLikelihood), pointer :: this

if (Ini%Read_Logical('use_HST',.false.)) then
    allocate(this)
    this%LikelihoodType = 'Hubble'
    this%name= Ini%Read_String('Hubble_name')
    this%H0 = Ini%Read_Double('Hubble_H0')
    this%H0_err = Ini%Read_Double('Hubble_H0_err')
    call Ini%Read('Hubble_zeff',this%zeff)
    call Ini%Read('Hubble_angconversion',this%angconversion)
    this%needs_background_functions = .true.
    call LikeList%Add(this)
end if

end subroutine HSTLikelihood_Add

real(mcp) function HST_LnLike(this, CMB)
Class(HSTLikelihood) :: this
Class(CMBParams) CMB
real(mcp) :: theoryval

theoryval = this%angconversion/this%Calculator%AngularDiameterDistance(this%zeff)
HST_LnLike = (theoryval - this%H0)**2/(2*this%H0_err**2)

end function  HST_LnLike

end module HST
```

Likelihood variables

```
# Riess et al (2011) value of H0 = 73.8 +/- 2.4 km/s/Mpc
# Riess et al: 1103.2976
Hubble_zeff = 0.04
Hubble_angconversion = 11425.8
#angconversion converts inverse of the angular diameter distance at z = zeff to H0
#for the fiducial cosmology (omega_k = 0, omega_lambda = 0.7, w = -1)
#likelihood is in terms of inverse of the angular diameter distance, so includes the tiny cosmological
#dependence of the measurement (primarily on w) correctly.
Hubble_H0 = 73.8
Hubble_H0_err = 2.4
Hubble_name = HST
use_HST=T
```

Read the ini file and add to the list of likelihoods

Added to the list of likelihoods

Likelihood computation

Some complication

Some likelihoods require more complicated settings

- higher number of input observational parameters
- higher number of data points
- several possible setups

Exercise:

Write a new likelihood

Extra exercise: a new likelihood

Write a new, easy likelihood to include in the code, e.g. redshift drift

Look at a distant source and measure its change in redshift with time (its spectroscopic velocity).
Assuming peculiar motion is negligible, this is due cosmological expansion

$$\frac{\Delta v}{c} = H_0 \Delta t \left[1 - \frac{E(z_s)}{1+z_s}\right]$$

Can we measure it?

Probably (or hopefully) we will measure it in the future, e.g. with SKA phase 2 or with the E-ELT.

For the latter, the planned CODEX experiment has simulated errors for distant QSO observations

$$\sigma_{\Delta v} = 1.35 \frac{2370}{S/N} \sqrt{\frac{30}{N_{QSO}}} \left(\frac{5}{1+z_{QSO}} \right)^{x} \; cm\; s^{-1}$$

$$x = 1.7 \text{ if } z \leq 4$$

$$x = 0.9 \text{ if } z > 4$$

What to do?

Take as example the HST.f90 likelihood

1. Write the new likelihood module, e.g. RD.f90
   - Define the new likelihood variables, i.e. what you will read from the ini file
   - Write the routine connecting to the likelihood ini file, e.g. RD_LikelihoodAdd
     This will have to read the observational quantities, i.e. redshifts,
     quantities needed to compute errors and usage flag (use_RD)
   - Write the actual likelihood function (pay attention to cm/s!). Use a simple chi2

2. Call the new module in DataLikelihoods.f90 and add it to the Makefile

3. Write your own .ini likelihood file

Likelihood function

$$P(d|\theta) = e^{-\frac{\chi^2}{2}}$$

$$\chi^2 = \sum_{z_i} \frac{\left[\Delta v(z_i) - \Delta v^{\mathrm{fid}}(z_i)\right]^2}{\sigma^2(z_i)}$$

Data points

Let’s keep it simple and almost realistic:

use 3 data points

1. z=2.5 Δv(fiducial)
2. z=3.5 Δv(fiducial)
3. z=5.0 Δv(fiducial)

Source per bin = 10
S/N = 3000
Δt = 20 yr

Results

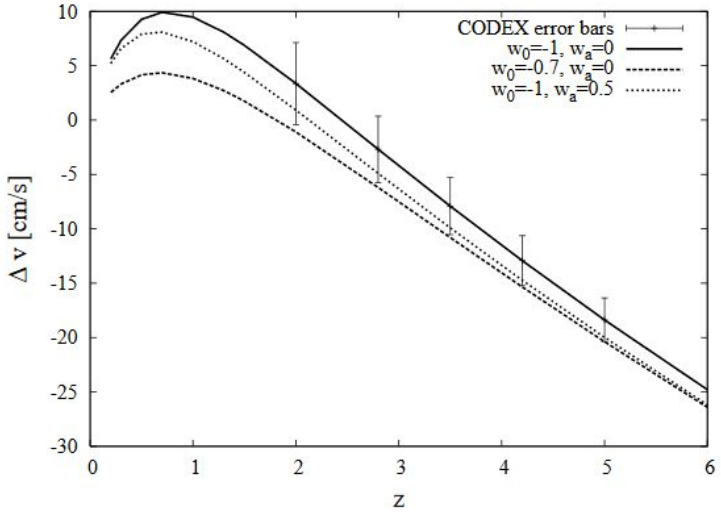


FIG. 2: Δv signal for different combinations of EoS parameters w_0 and w_a. The error bars refers to the CODEX spectrograph.

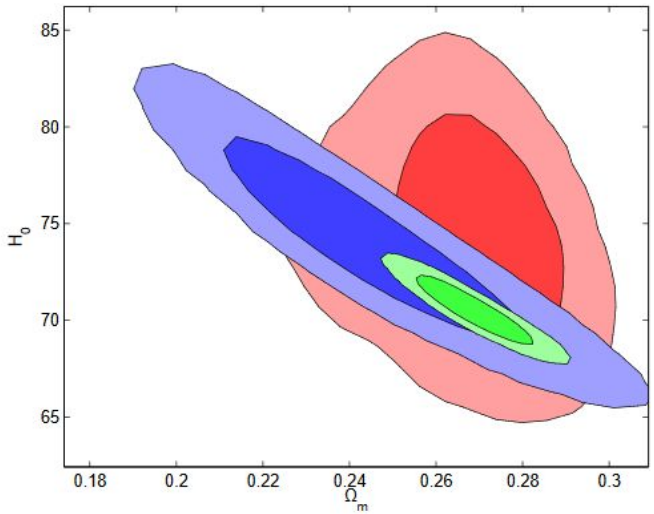


FIG. 5: 2-D constraints on H_0 and Ω_m using CMB (blue), SL (red) and combining the two probes (green).

Martinelli et al. 2017