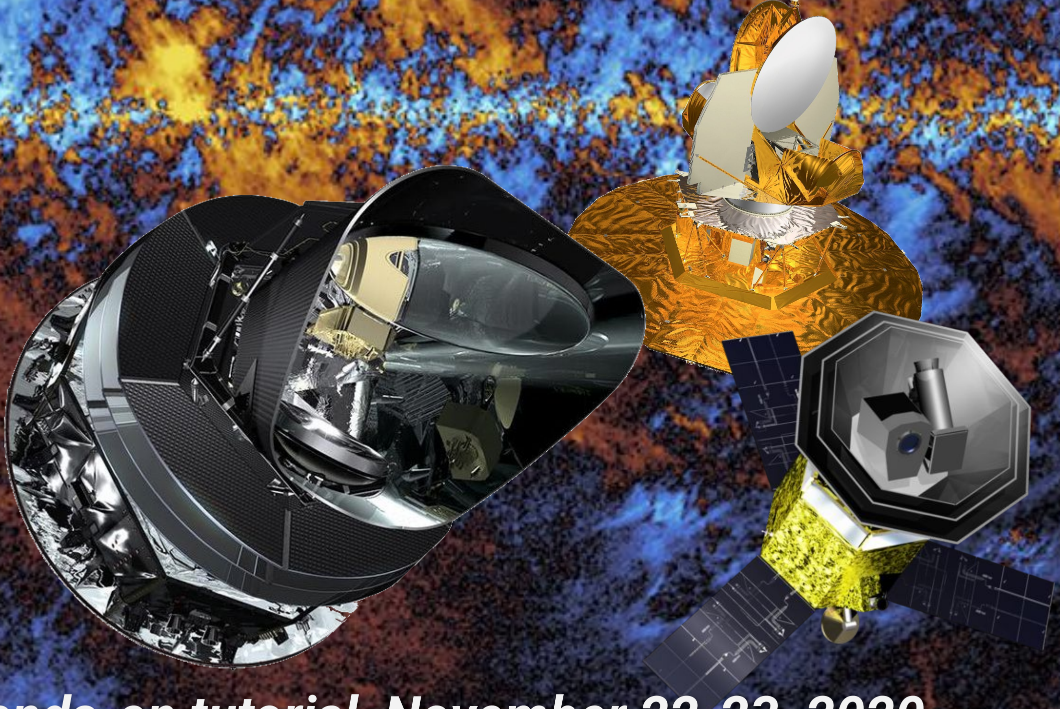
The Commander parameter file

Kristian Joten Andersen



BeyondPlanck online hands-on tutorial, November 22-23, 2020

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 776282

What is the Commander parameter file?

- All information going into any Commander run
  - The “recipe”

- Three main parameter types
  - Infrastructure parameters; these control the behaviour of the algorithms and IO
  - Data set parameters; properties of each data set
  - Model parameters; properties of each sky component

- Commander documentation is available through the project home page: https://docs.beyondplanck.science/#/README (direct link)
- Caveats:
  - This is a software platform for cutting-edge research, and therefore by nature a continuous work-in-progress.
  - “Alpha state”
  - Support is provided on a strictly voluntary basis; there is no designated “help desk”
  - If you find information missing, please contact us.

General notes about read-in

- Any line starting with # is assumed as a comment
- Blank lines are allowed
- Any specific path should be given inside apostrophes
  - ‘path/to/some/data/data.dat‘
- Every parameter takes the form:

  PARAMETER_NAME = value
- The ***first*** occurance of a parameter is the one that will be read

General notes about read-in

@INCLUDE

- Recurse into the given file
- “@INCLUDE parfile2.txt” will include parameters from parfile2.txt at the current position in the original parameter file

Command line read-in

- Any parameter may be specified using the command line
- --PARAMETER_NAME=value   (no spaces)
- Will override the parameter file for the specified parameter

Error checking and debugging

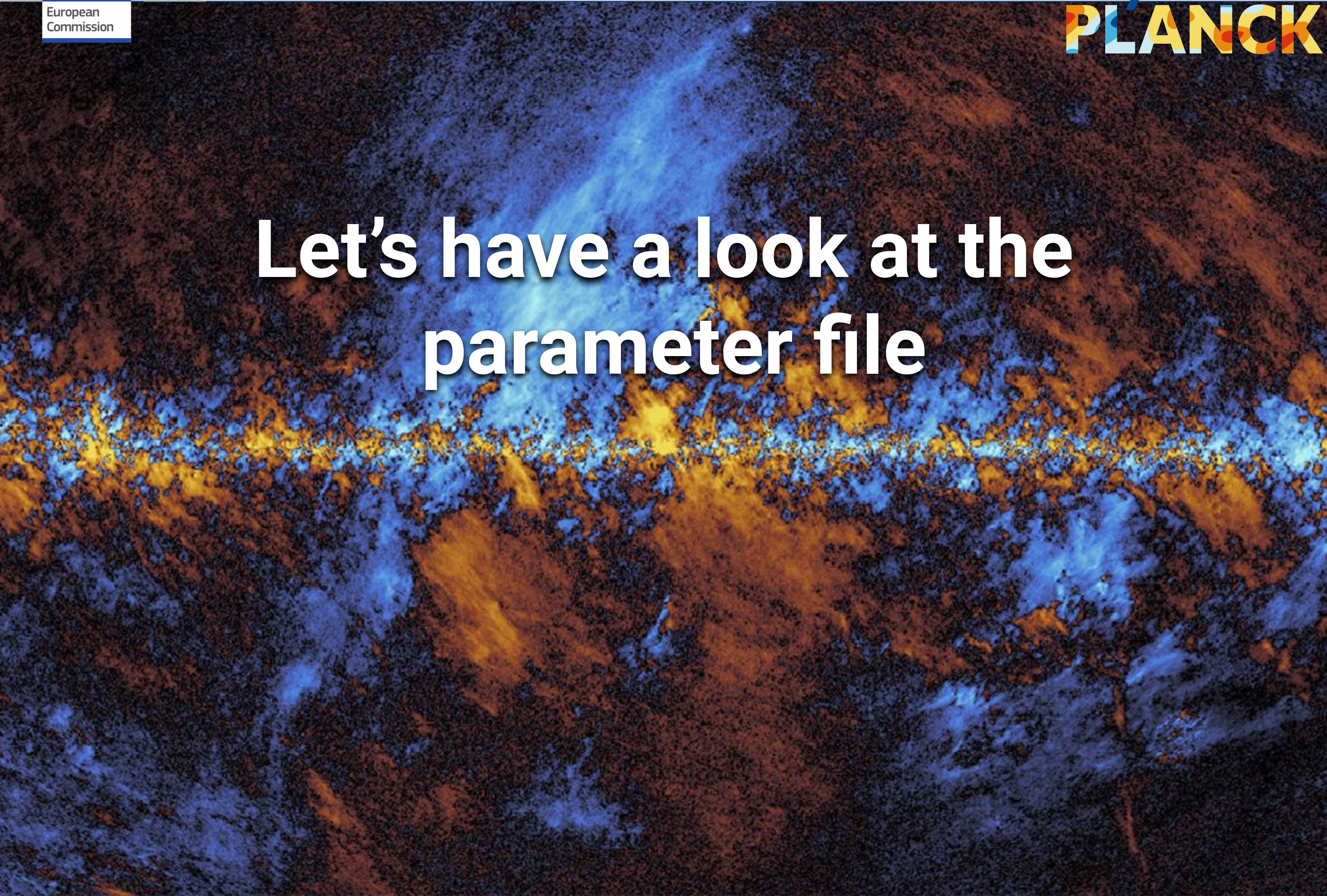
Parameter file is parsed at the beginning of the run

- Stored in-memory in a Fortran type called “cpar” ("Commander parameters")
- Active filenames are validated with respect to existence, but not content.
- Many types of parameter file errors are therefore (automatically) detected immediately, but not all.

Error checking and debugging

If you find that the code crashes with something that looks like a parameter error, typical things to check are the following:

- Is a given parameter of the correct/expected type?
  - Check which line causes the crash, and look it up in comm_param_mod.f90
- Does a given file contain the expected data type?
- If it is an ASCII input file, does it have the correct format?



Let’s have a look at the parameter file

Algorithm specification

→ = parameters that one is likely to change on a (semi-)regular basis when working with Commander, see the documentation for details.

```
→ OPERATION                       = sample   # {sample,optimize}
  VERBOSITY                       = 3        # [0,...,3]

  ##############################################################
  #                   Algorithm specification                  #
  ##############################################################

  # Monte Carlo options
  NUMCHAIN                        = 1        # Number of independent chains
→ NUM_GIBBS_ITER                  = 1000        # Length of each Markov chain
  BASE_SEED                       = 163425   # Seed for random number generator
→ CHAIN_STATUS                    = append    {append, new}
  NUM_INIT_CHAINS                 = 1
→ INIT_CHAIN01                    = "data_BP8/chain_init_BP8.15.h5:9"   {new}
                                      path      name                sample
→ NUM_GIBBS_STEPS_PER_TOD_SAMPLE = 1   How often to run TOD analysis

→ SAMPLE_ONLY_POLARIZATION        = .false. |
→ SAMPLE_SIGNAL_AMPLITUDES        = .true.  |   Component separation analysis
→ SAMPLE_SPECTRAL_INDICES         = .true.  |
→ SAMPLE_POWSPEC                  = .false. |

→ ENABLE_TOD_ANALYSIS             = .true.                                |
  TOD_OUTPUT_4D_MAP_EVERY_NTH_ITER = 10                                   |
  TOD_OUTPUT_AUXILIARY_MAPS_EVERY_NTH_ITER = 10                           |  TOD
  TOD_INCLUDE_ZODI                = .false.                               |
  FFTW3_MAGIC_NUMBERS             = 'data_BP8/fft3_magic_numbers_230810.txt'
→ * TOD_NUM_BP_PROPOSALS_PER_ITER  = 1    # 1 for sampling; >= 1 for optimize
```

* # bandpass proposals per TOD sample

Algorithm specification

```
  # Options for CMB resampling (for constrained realization production)
➔ RESAMPLE_CMB                  = .false.
  FIRST_SAMPLE_FOR_CMB_RESAMP   = 1
  LAST_SAMPLE_FOR_CMB_RESAMP    = 15
  NUM_SUBSAMP_PER_MAIN_SAMPLE   = 10

  # Numerical accuracy settings
➔ CG_CONVERGENCE_CRITERION       = fixed_iter # {residual, chisquare}
  CG_LMAX_PRECOND                =    -1      # lmax for low-l preconditioner
➔ CG_MAXITER                     =   300      # Conjugate gradients time out limit
➔ CG_MINITER                     =     5
➔ CG_TOLERANCE                   = 1.d-8      # Fractional CG convergence criterion
* CG_CONV_CHECK_FREQUENCY        =     1      # Check convergence every n'th iteration
  CG_PRECOND_TYPE                = diagonal   # {diagonal, pseudoinv}
**CG_INIT_AMPS_ON_ZERO           = .false.
  SET_ALL_NOISE_MAPS_TO_MEAN     = .false.
  NUM_INDEX_CYCLES_PER_ITERATION =     1
  IGNORE_GAIN_AND_BANDPASS_CORR = .false.
```

Conjugate Gradients (CG)

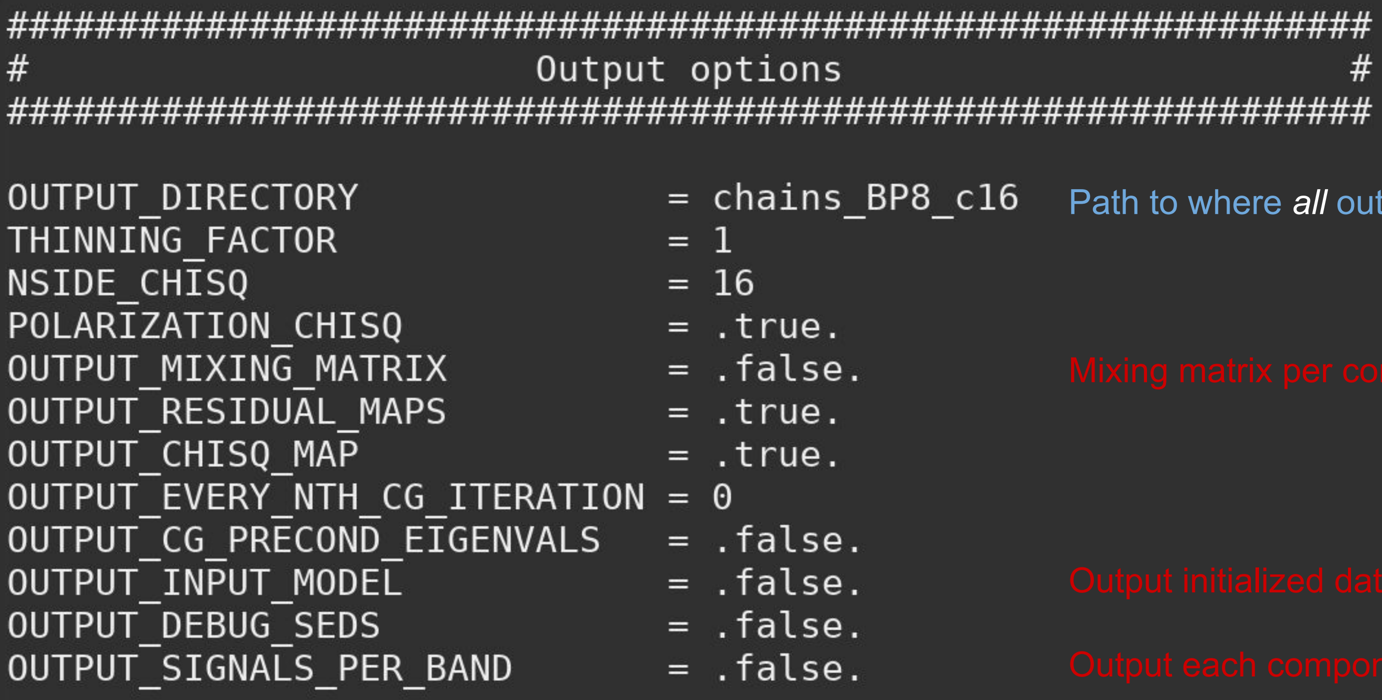
(residual or chisquare)

Seljebotn et al. 2019, A&A, 627, A98

* If using chisq: set to ~5 as this check is more expensive than residual

** Set all component amplitudes to zero for each CG search. Can lead to long convergence time

Output options



```
  ##############################################################
  #                       Output options                       #
  ##############################################################

→ OUTPUT_DIRECTORY              = chains_BP8_c16
→ THINNING_FACTOR               = 1
→ NSIDE_CHISQ                   = 16
→ POLARIZATION_CHISQ            = .true.
* OUTPUT_MIXING_MATRIX          = .false.
  OUTPUT_RESIDUAL_MAPS          = .true.
  OUTPUT_CHISQ_MAP              = .true.
**OUTPUT_EVERY_NTH_CG_ITERATION = 0
  OUTPUT_CG_PRECOND_EIGENVALS   = .false.
  OUTPUT_INPUT_MODEL            = .false.
  OUTPUT_DEBUG_SEDS             = .false.
* OUTPUT_SIGNALS_PER_BAND       = .false.
```

Path to where *all* output files are written

Mixing matrix per component per band

Output initialized data and exits

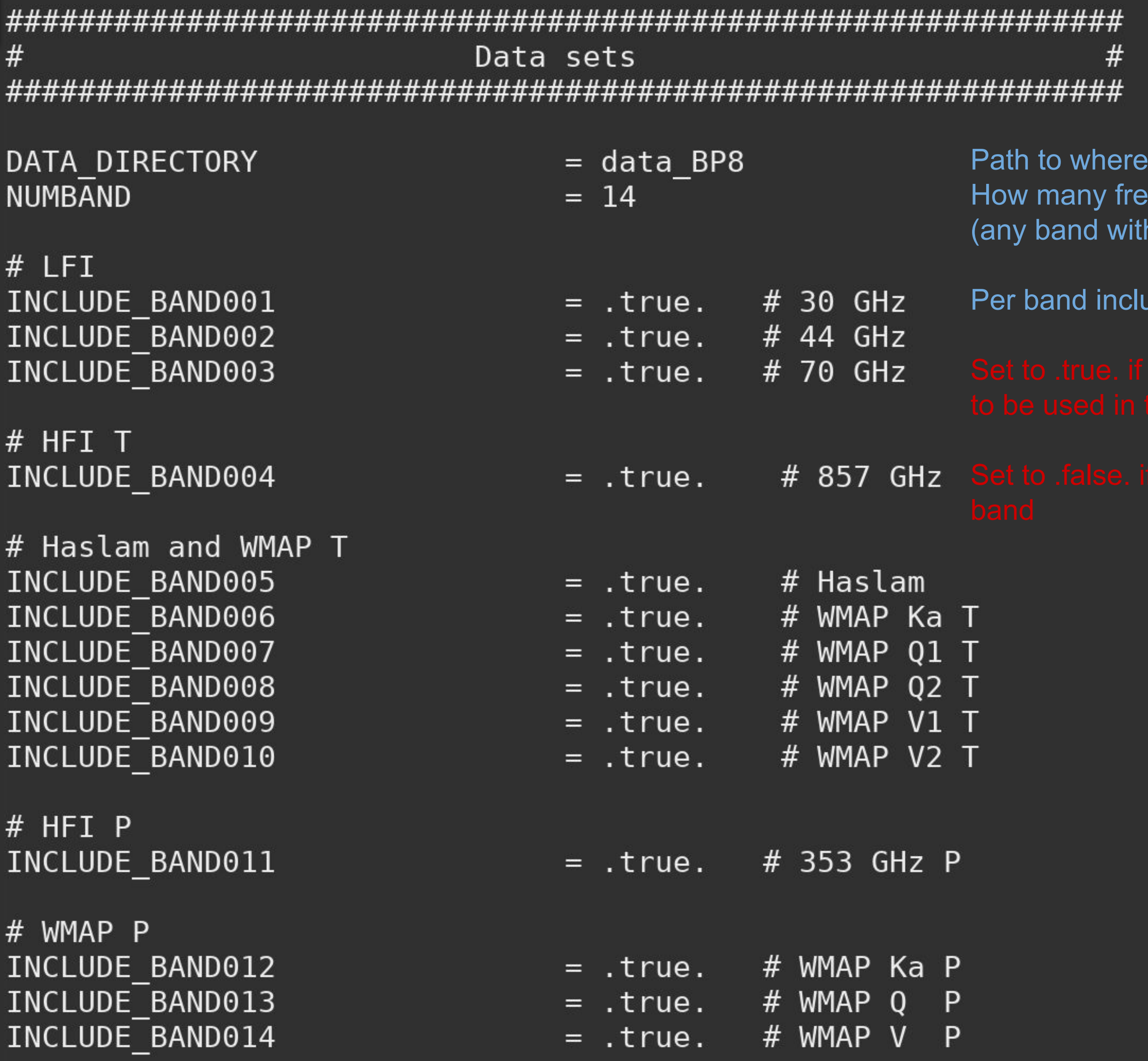
Output each component per frequency band

Different options of what to output

* Output each instance as a HEALPix map at the band resolution/pixelarization. Use with care as this will require a lot of disk space.

** Outputs non-converged CG iterations to files. Use with care as they will overwrite converged samples with the same numbering. Only use for debugging.

Data sets: inclusion



```
##############################################################
#                         Data sets                          #
##############################################################

DATA_DIRECTORY                 = data_BP8
NUMBAND                        = 14

# LFI
INCLUDE_BAND001                = .true.   # 30 GHz
INCLUDE_BAND002                = .true.   # 44 GHz
INCLUDE_BAND003                = .true.   # 70 GHz

# HFI T
INCLUDE_BAND004                = .true.    # 857 GHz

# Haslam and WMAP T
INCLUDE_BAND005                = .true.    # Haslam
INCLUDE_BAND006                = .true.    # WMAP Ka T
INCLUDE_BAND007                = .true.    # WMAP Q1 T
INCLUDE_BAND008                = .true.    # WMAP Q2 T
INCLUDE_BAND009                = .true.    # WMAP V1 T
INCLUDE_BAND010                = .true.    # WMAP V2 T

# HFI P
INCLUDE_BAND011                = .true.   # 353 GHz P

# WMAP P
INCLUDE_BAND012                = .true.   # WMAP Ka P
INCLUDE_BAND013                = .true.   # WMAP Q  P
INCLUDE_BAND014                = .true.   # WMAP V  P
```

Path to where (almost) *all* input data is stored.

How many frequency bands are defined (any band with # higher will not be read)

Per band inclusion flags.

Set to .true. if you want the frequency band to be used in the analysis.

Set to .false. if you want to omit the given band

Data sets: smoothing scales

```
 *  SOURCE_MASKFILE                = none    #bright_sources.txt
    PROCESSING_MASKFILE            = none    #procmask.fits
    PROCESSING_MASKFILE2           = none    #procmask2.fits
    PROC_SMOOTH_SCALE              = 30.     #arcmin; smoothing inside processing mask

    # Spectral index sampling options
 ➜  NUM_SMOOTHING_SCALES           = 1          Number of spectral parameter smoothing scales

 ** SMOOTHING_SCALE_FWHM01           = 300.    #
    SMOOTHING_SCALE_LMAX01           = 96
    SMOOTHING_SCALE_NSIDE01          = 32
    SMOOTHING_SCALE_PIXWIN01         = pixel_window_n0032.fits      HEALPix pixel window file
*** SMOOTHING_SCALE_FWHM_POSTPROC01 = 600.    # Smoothing FWHM after sampling
```

* Helpful to mask out problematic point sources / regions, read documentation for details.

** [arcmin] larger than any FWHM of bands included in sampling at this scale

*** [arcmin] Any sampled spectral parameter at this smoothing scale will be smoothed with a beam of this FWHM after sampling. For pixel-region sampling, this is the smoothing between regions!

Data sets: band specifics

General parameter syntax:
BANDᐟPARAMETERxxx = value
xxx = 3-digit band number

How to add new data bands will be shown in tomorrow’s tutorials

```
  # 30 GHz parameters
→ BAND_LABEL001                    = 030
  BAND_TOD_TYPE001                 = 'LFI' #{LFI, WMAP, none}
  BAND_OBS_PERIOD001               = 1
  BAND_POLARIZATION001             = .true.
  BAND_NSIDE001                    = 512
  BAND_LMAX001                     = 1500
  BAND_UNIT001                     = uK_cmb
  BAND_NOISE_FORMAT001             = rms
→ BAND_MAPFILE001                  = BP_030_map.fits
→ BAND_NOISEFILE001                = BP_030_rms.fits
→ BAND_REG_NOISEFILE001            = none   # BP_030_regnoise.fits
→ BAND_NOISE_RMS001_SMOOTH01       = BP_030_rms_smoothscale1.fits   {native, none}
→ BAND_NOISE_UNIFORMIZE_FSKY001    = 0.0
→ BAND_MASKFILE001                 = fullsky
  BAND_BEAMTYPE001                 = b_l   # {b_l, febecop}
→ BAND_BEAM_B_L_FILE001            = Bl_TEB_npipe6v19_30GHzx30GHz.fits
  BAND_BEAM_B_PTSRC_FILE001        = febecop_AT20G_GB6_NVSS_PCCS2_v6_030.h5
  BAND_PIXEL_WINDOW001             = pixel_window_n0512.fits
  BAND_SAMP_NOISE_AMP001           = .false.
```

Data sets: band specifics

```
   BAND_SAMP_BANDPASS001              = .false.
   BAND_BANDPASSFILE001               = LFI_instrument_v4.h5
→  BAND_SAMP_GAIN001                  = .false.
→  BAND_GAIN_PRIOR_MEAN001             = 1.
→  BAND_GAIN_PRIOR_RMS001              = 0.1
→  BAND_GAIN_CALIB_COMP001            = all 'cmb'
   BAND_GAIN_LMIN001                  = -1
   BAND_GAIN_LMAX001                  = -1
   BAND_GAIN_APOD_MASK001             = fullsky
   BAND_GAIN_APOD_FWHM001             = 120.
→  BAND_MASKFILE_CALIB001             = mask_common_dx12_n0512_TQU.fits
   BAND_DEFAULT_GAIN001               =   1.
   BAND_DEFAULT_BP_DELTA001           =   0.
   BAND_DEFAULT_NOISEAMP001           =   1.
   BAND_COMPONENT_SENSITIVITY001      = broadband
```

< 0 ~ no upper/lower limit

Data sets: band specifics

```
→ BAND_TOD_MAIN_PROCMASK001       = mask_proc_030_res_v5.fits
→ BAND_TOD_SMALL_PROCMASK001      = mask_smap6.fits
→ BAND_TOD_BP_INIT_PROP001        = bp_init_030_v1.dat
  BAND_TOD_RIMO001                = LFI_instrument_v4.h5
* BAND_TOD_FILELIST001            = filelist_30_v17.txt
  BAND_TOD_HALFRING001            = 0
→ BAND_TOD_START_SCANID001        = 3
→ BAND_TOD_END_SCANID001          = 44072
→ BAND_TOD_TOT_NUMSCAN001         = 45860
  BAND_TOD_FLAG001                = 6111232
  BAND_TOD_ORBITAL_ONLY_ABSCAL001 =.false.
→ BAND_TOD_DETECTOR_LIST001       = "27M,27S,28M,28S"
→ BAND_TOD_INIT_FROM_HDF001       = default            {default, none, path to chain+sample}
```

* TOD scan definition file. List of all compressed time-ordered data files with initial values. See documentation.

Global model parameters

```
  MJYSR_CONVENTION               = IRAS
  T_CMB                          = 2.7255d0

→ INSTRUMENT_PARAM_FILE          = instrument_params.dat
→ INIT_INSTRUMENT_FROM_HDF       = default
```

Initial band gain and bandpass corrections

```
  CMB_DIPOLE_PRIOR               = none
  # 'mask_common_dx12_n1024_TQU.fits; 3364.4; 263.998; 48.265' # LFI 2018

→ NUM_SIGNAL_COMPONENTS          = 4
→ INCLUDE_COMP01                 = .true.  # Cmb # CMB; no monopole
  INCLUDE_COMP02                 = .true.  # synch # Synch pow-law
  INCLUDE_COMP03                 = .true.  # dust # Thermal dust
  INCLUDE_COMP04                 = .true.  # md # Mono and dipoles
```

{.true., .false.}

Similar to frequency bands

```
→ OUTPUT_COMPS_TO_CHAINDIR = 'all'
```

CG

```
→ NUM_CG_SAMPLING_GROUPS = 2
  CG_SAMPLING_GROUP01           = 'md'
  CG_SAMPLING_GROUP_MASK01      = mask_common_dx12_n1024_TQU.fits
  CG_SAMPLING_GROUP_MAXITER01   = 3
→ CG_SAMPLING_GROUP02           = 'cmb,dust,synch'
→ CG_SAMPLING_GROUP_MASK02      = fullsky
→ CG_SAMPLING_GROUP_MAXITER02   = 50
```

Component labels

Sampling mask

Max CG iterations (# of iterations for fixed_iter)

```
  # Alm sampler settings
→ ALMSAMP_NSAMP_ALM            = 100  # of mcmc samples per gibbs
  ALMSAMP_BURN_IN              = 1     # of gibbs iterations with steplength adjustment
→ ALMSAMP_NSIDE_CHISQ_LOWRES               = 16
  ALMSAMP_PRIOR_FWHM                       = 0
  ALMSAMP_OPTIMIZE_ALM                     = .false. # save chisq from prev gibbs iter
→ ALMSAMP_APPLY_PRIOR                      = .true. # apply prior to alms
→ ALMSAMP_PIXREG                           = .true.
→ ALMSAMP_PRIORSAMP_FROZEN_REGIONS         = .true.
```

a_lm spectral parameter sampler

```
  #local sampler settings
→ LOCALSAMP_BURN_IN       = 1    # of gibbs iterations with steplength adjustment
```

Model specific parameters

General parameter syntax:
COMP_PARAMETERxx = value
xx = 2-digit band number

```
# CMB
COMP_LABEL01                  = cmb
COMP_TYPE01                   = cmb
COMP_CLASS01                  = diffuse    # {diffuse, ptsrc, template}
COMP_POLARIZATION01           = .true.
COMP_CG_SCALE01               = 1.d0

COMP_NSIDE01                  = 1024
COMP_MONOPOLE_PRIOR01         = none   "monopole:mask_common_dx12_n1024_TQU.fits"
COMP_DEFLATION_MASK01         = fullsky
COMP_L_APOD01                 = 2000
COMP_LMIN_AMP01               = 0
COMP_LMAX_AMP01               = 2000
COMP_LMAX_IND01               = 0
COMP_OUTPUT_FWHM01            = 14         # arcmin
COMP_UNIT01                   = uK_cmb
COMP_NU_REF_T01               = 1  100.
COMP_NU_REF_P01               = 1  100.
COMP_CL_TYPE01                = power_law  # {none, single_l, binned, power_law}
COMP_CL_POLTYPE01             = 1  # {1 = {T+E+B}, 2 = {T,E+B}, 3 = {T,E,B}}
COMP_CL_BETA_PRIOR_MEAN01     =  0.0
COMP_CL_BETA_PRIOR_RMS01      =  0.1
COMP_CL_L_PIVOT01             =  20          # Pivot multipole
COMP_CL_DEFAULT_AMP_T01       =  1000000        # D_l = amp * (l/lpivot)**beta
COMP_CL_DEFAULT_AMP_E01       =  1000
COMP_CL_DEFAULT_AMP_B01       =  1000
COMP_CL_DEFAULT_BETA_T01      =  0.d0
COMP_CL_DEFAULT_BETA_E01      = -0.5d0
COMP_CL_DEFAULT_BETA_B01      = -0.5d0
#COMP_CL_TYPE01                = binned     # {none, binned, power_law}
#COMP_CL_BIN_FILE01            = bins_lmax2000_TE.dat   # for binned type
#COMP_CL_DEFAULT_FILE01        = base_plikHM_TTTEEE_lowl_lowE_lensing.minimum.theory_cl
COMP_MASK01                   = fullsky
COMP_INPUT_AMP_MAP01          = init_cmb_amp_BP8.11.fits
COMP_PRIOR_AMP_MAP01          = none
COMP_OUTPUT_EB_MAP01          = .false.
COMP_INIT_FROM_HDF01          = default
```

prior
RMS

initialization map

prior mean, 'none' = no prior

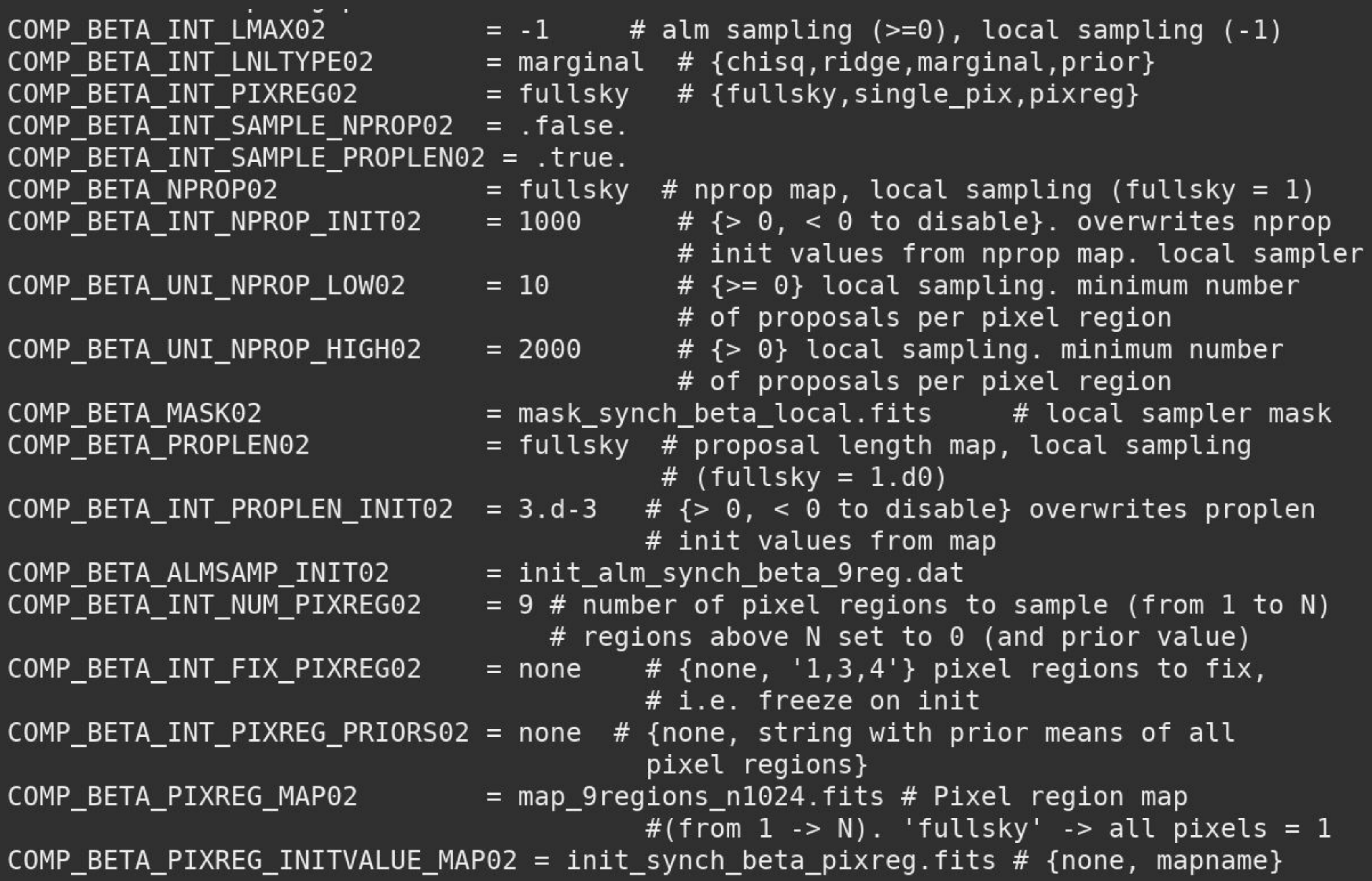
Model specific parameters

```
  # Synchrotron component
  COMP_LABEL02                = synch
  COMP_TYPE02                 = power_law
  COMP_CLASS02                = diffuse    # {diffuse, ptsrc}
  COMP_POLARIZATION02         = .true.
→ COMP_CG_SCALE02             = 1          Multiplicative amplitude scale used in CG search
→ COMP_CG_SAMP_GROUP_MAXITER02 = 40        Max CG iterations in amplitude sampling after marginal
  COMP_NSIDE02                = 1024            likelihood spectral index sampling
  COMP_MONOPOLE_PRIOR02       = none
  COMP_DEFLATION_MASK02       = fullsky
  COMP_L_APOD02               = 1500
  COMP_LMIN_AMP02             = 0
  COMP_LMAX_AMP02             = 1500
  COMP_OUTPUT_FWHM02          = 60         # arcmin
  COMP_UNIT02                 = uK_RJ
→ COMP_NU_REF_T02             = 30
  COMP_NU_REF_P02             = 30
  COMP_MASK02                 = fullsky
  COMP_CL_TYPE02              = gauss  # {none, single_l, binned, power_law}
  COMP_CL_POLTYPE02           = 2  # {1 = {T+E+B}, 2 = {T,E+B}, 3 = {T,E,B}}
  COMP_CL_BETA_PRIOR_MEAN02   = -0.5
  COMP_CL_BETA_PRIOR_RMS02    =  0.1
  COMP_CL_L_PIVOT02           =   100        # Pivot multipole
  COMP_CL_DEFAULT_AMP_T02     =  1e3        # D_l = amp * (l/lpivot)**beta
  COMP_CL_DEFAULT_AMP_E02     =  200
  COMP_CL_DEFAULT_AMP_B02     =  100
  COMP_CL_DEFAULT_BETA_T02    =  60d0
  COMP_CL_DEFAULT_BETA_E02    =  30d0
  COMP_CL_DEFAULT_BETA_B02    =  30d0
```

Model specific parameters: Synchrotron

```
COMP_INDMASK02               = mask_synch_beta_BP8_10deg_new_chisqmask.fits
COMP_LMAX_IND02              = 100
COMP_PRIOR_UNI_BETA_LOW02    =  -4.5
COMP_PRIOR_UNI_BETA_HIGH02   =  -1.5
COMP_PRIOR_GAUSS_BETA_MEAN02 =  -3.1      Prior mean
COMP_PRIOR_GAUSS_BETA_RMS02  =   0.1      Prior RMS
COMP_APPLY_JEFFREYS_PRIOR02  =  .false.
COMP_BETA_SMOOTHING_SCALE02  = 2
COMP_BETA_POLTYPE02          = 2       # index {1 = {T+Q+U}, 2 = {T,Q+U}, 3 = {T,Q,U}}
COMP_BETA_NU_MIN02           =   0.    # Lowest frequency for index estimation in GHz
COMP_BETA_NU_MAX02           =  80.    # Highest frequency for index estimation in GHz
COMP_INPUT_AMP_MAP02         = init_synch_amp.fits
COMP_PRIOR_AMP_MAP02         = none
COMP_INPUT_BETA_MAP02        = init_synch_beta.fits  default
COMP_DEFAULT_BETA02          = -3.1
COMP_OUTPUT_EB_MAP02         = .false.
COMP_INIT_FROM_HDF02         = default
```

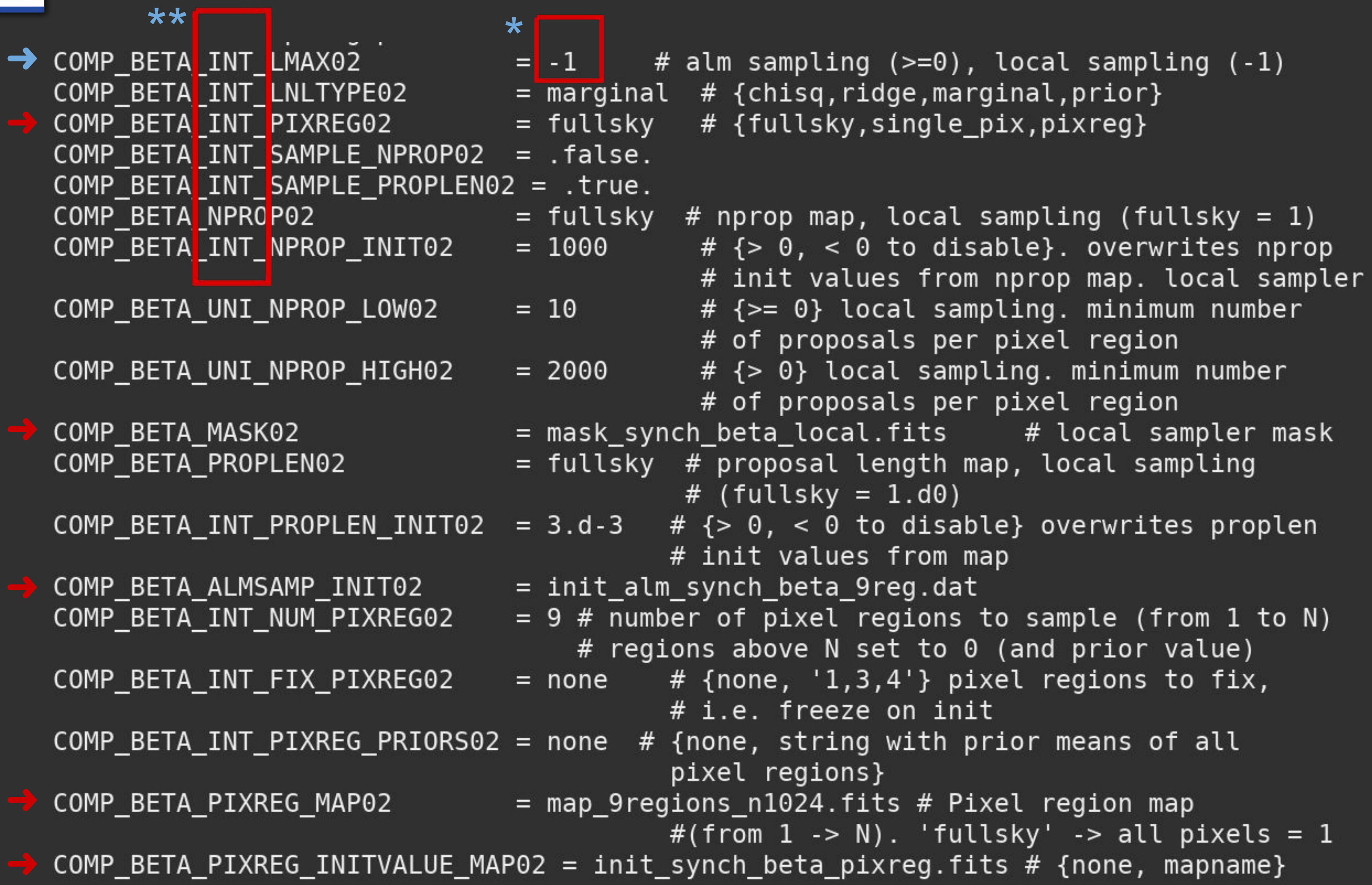
Model specific parameters



```
COMP_BETA_INT_LMAX02            = -1     # alm sampling (>=0), local sampling (-1)
COMP_BETA_INT_LNLTYPE02         = marginal  # {chisq,ridge,marginal,prior}
COMP_BETA_INT_PIXREG02          = fullsky   # {fullsky,single_pix,pixreg}
COMP_BETA_INT_SAMPLE_NPROP02    = .false.
COMP_BETA_INT_SAMPLE_PROPLEN02 = .true.
COMP_BETA_NPROP02               = fullsky  # nprop map, local sampling (fullsky = 1)
COMP_BETA_INT_NPROP_INIT02      = 1000      # {> 0, < 0 to disable}. overwrites nprop
                                            # init values from nprop map. local sampler
COMP_BETA_UNI_NPROP_LOW02       = 10        # {>= 0} local sampling. minimum number
                                            # of proposals per pixel region
COMP_BETA_UNI_NPROP_HIGH02      = 2000      # {> 0} local sampling. minimum number
                                            # of proposals per pixel region
COMP_BETA_MASK02                = mask_synch_beta_local.fits     # local sampler mask
COMP_BETA_PROPLEN02             = fullsky  # proposal length map, local sampling
                                           # (fullsky = 1.d0)
COMP_BETA_INT_PROPLEN_INIT02    = 3.d-3   # {> 0, < 0 to disable} overwrites proplen
                                          # init values from map
COMP_BETA_ALMSAMP_INIT02        = init_alm_synch_beta_9reg.dat
COMP_BETA_INT_NUM_PIXREG02      = 9 # number of pixel regions to sample (from 1 to N)
                                  # regions above N set to 0 (and prior value)
COMP_BETA_INT_FIX_PIXREG02      = none    # {none, '1,3,4'} pixel regions to fix,
                                          # i.e. freeze on init
COMP_BETA_INT_PIXREG_PRIORS02 = none  # {none, string with prior means of all
                                        pixel regions}
COMP_BETA_PIXREG_MAP02          = map_9regions_n1024.fits # Pixel region map
                                          #(from 1 -> N). 'fullsky' -> all pixels = 1
COMP_BETA_PIXREG_INITVALUE_MAP02 = init_synch_beta_pixreg.fits # {none, mapname}
```

Many of these parameters are depending on each other. See documentation for details!

Model specific parameters



```
      **                  *
COMP_BETA_INT_LMAX02          = -1     # alm sampling (>=0), local sampling (-1)
COMP_BETA_INT_LNLTYPE02       = marginal  # {chisq,ridge,marginal,prior}
COMP_BETA_INT_PIXREG02        = fullsky   # {fullsky,single_pix,pixreg}
COMP_BETA_INT_SAMPLE_NPROP02  = .false.
COMP_BETA_INT_SAMPLE_PROPLEN02 = .true.
COMP_BETA_NPROP02             = fullsky  # nprop map, local sampling (fullsky = 1)
COMP_BETA_INT_NPROP_INIT02    = 1000      # {> 0, < 0 to disable}. overwrites nprop
                                          # init values from nprop map. local sampler
COMP_BETA_UNI_NPROP_LOW02     = 10        # {>= 0} local sampling. minimum number
                                          # of proposals per pixel region
COMP_BETA_UNI_NPROP_HIGH02    = 2000      # {> 0} local sampling. minimum number
                                          # of proposals per pixel region
COMP_BETA_MASK02              = mask_synch_beta_local.fits     # local sampler mask
COMP_BETA_PROPLEN02           = fullsky  # proposal length map, local sampling
                                         # (fullsky = 1.d0)
COMP_BETA_INT_PROPLEN_INIT02  = 3.d-3   # {> 0, < 0 to disable} overwrites proplen
                                        # init values from map
COMP_BETA_ALMSAMP_INIT02      = init_alm_synch_beta_9reg.dat
COMP_BETA_INT_NUM_PIXREG02    = 9 # number of pixel regions to sample (from 1 to N)
                                  # regions above N set to 0 (and prior value)
COMP_BETA_INT_FIX_PIXREG02    = none    # {none, '1,3,4'} pixel regions to fix,
                                        # i.e. freeze on init
COMP_BETA_INT_PIXREG_PRIORS02 = none  # {none, string with prior means of all
                                        pixel regions}
COMP_BETA_PIXREG_MAP02        = map_9regions_n1024.fits # Pixel region map
                                        #(from 1 -> N). 'fullsky' -> all pixels = 1
COMP_BETA_PIXREG_INITVALUE_MAP02 = init_synch_beta_pixreg.fits # {none, mapname}
```

* Affects input spectral parameter map. If pixel-by-pixel structure, this must be -1

** Defined for poltype (polarization type), “INT” = poltype index 1 {T or T+Q+U}, “POL” = poltype index 2 {Q+U or Q}, “POL3” = poltype index 3 {U}

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776282

- "*BeyondPlanck*"
  - COMPET-4 program
  - PI: Hans Kristian Eriksen
  - Grant no.: 776282
  - Period: Mar 2018 to Nov 2020

Collaborating projects:

- "*bits2cosmology*"
  - ERC Consolidator Grant
  - PI: Hans Kristian Eriksen
  - Grant no: 772 253
  - Period: April 2018 to March 2023
- "*Cosmoglobe*"
  - ERC Consolidator Grant
  - PI: Ingunn Wehus
  - Grant no: 819 478
  - Period: June 2019 to May 2024

European Commission

Questions?

Beyond PLANCK

Commander

UNIVERSITAS OSLOENSIS MDCCCXI

UNIVERSITAS STUDIORUM MEDIOLANENSIS

OSSERVATORIO ASTRONOMICO DI TRIESTE
INAF ISTITUTO NAZIONALE DI ASTROFISICA

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

planetek
hellas

VET TES EN NOV TAM TVM

JPL

BERKELEY LAB

HAVERFORD
COLLEGE

THE FRAMEWORK PROGRAMME FOR RESEARCH AND INNOVATION
HORIZON 2020

erc

Cosmoglobe

Beyond PLANCK