Planck Sky Model : User Manual

Jacques Delabrouille & the PSM development team

Release version 1.7.8

Contents

1 Foreword

1.1 Overview

The Planck Sky Model is a set of programs and data for the simulation or the prediction of sky emission at frequencies ranging from about 3 GHz (10 cm) to about 3 THz (100 microns).

The software is developed mostly in the IDL programming language. It uses the HEALPix sky-pixellisation package (with calls to C++ and F90 binaries), the astron library, the CAMB and/or CLASS softwares, and the MPFIT fitting library. The package comes as a collection of component-specific simulation codes put together by driver routines, automating fastidious tasks like parameter processing and component map coaddition.

The present document is a user manual, that describes some of the PSM features and possibilities. The document is regularly updated, but is not fully complete as, in particular, new developments are not documented before their interfaces with the rest of the software have stabilised, and before they are tested and validated at a reasonable level.

The PSM is (permanently) under development. Visit the web site regularly for releases of simulation products and of the software package.

1.2 Authors

The following people have contributed to the PSM project: Mark Ashdown, Jonathan Aumont, Carlo Baccigalupi, Anthony Banday, Soumen Basak, Jean-Philippe Bernard, Marc Betoule, François Bouchet, Guillaume Castex, David Clements, Antonio Da Silva, Gianfranco de Zotti, Jacques Delabrouille, Jean-Marc Delouis, Clive Dickinson, Fabrice Dodu, Klaus Dolag, Franz Elsner, Lauranne Fauvet, Gilles Faÿ, Giovanna Giardino, Joaquin Gonzalez-Nuevo, Maude Le Jeune, Hugo Jiménez-Pérez, Samuel Leach, Julien Lesgourgues, Michele Liguori, Juan Macías, Marcella Massardi, Sabino Matarrese, Pasquale Mazzotta, Jean-Baptiste Melin, Marc-Antoine Miville-Deschênes, Ludovic Montier, Sylvain Mottet, Roberta Paladini, Bruce Partridge, Rocco Piffaretti, Gary Prezeau, Simon Prunet, Sara Ricciardi, Matthieu Roman, Björn Schäfer, Sibylle Téchené, Luigi Toffolatti.

1.3 About this version

The version presented in this document, version 1.7.8, is the first public release of the PSM code. It is identical to v1.7.7, except for a bug fix in one program (`read_camb_cl.pro`), and minor changes in the documentation. This version has been used to generate (part of) FFP6 simulations for the Planck collaboration.

1.4 Contact

For questions about the PSM, to report bugs, or to suggest modifications, please contact Jacques Delabrouille (delabrouille@apc.univ-paris7.fr).

1.5 Credits

Whenever the PSM software and/or simulations are used, please acknowledge the usage of the PSM as follows: *"The authors acknowledge the use of the PSM, developed by the Component Separation Working Group (WG2) of the Planck Collaboration"*, and cite the PSM paper (Delabrouille et al., Astronomy & Astrophysics, Volume 553, id.A96), as well as the papers that describe the particular model you have been using (see section 17.1).

2 Installation Procedure

2.1 Requirements

2.1.1 Operating System

The PSM requires a UNIX or LINUX operating system (for running shell commands and scripts). The PSM software often calls UNIX line commands in IDL programs using the IDL command `SPAWN`.

2.1.2 wget

During software execution, the PSM uses `wget` to download useful data sets from the PSM data repository, if such data are not already present on the local machine. The present PSM version is compatible with `wget` version 1.12, which should be installed on the machine used for running the PSM.

2.1.3 IDL

The PSM is mainly composed of IDL scripts which require an IDL development environment. The present PSM version has been developed and tested mostly with IDL v7.1.1. Earlier versions of IDL cause a problem when the `VISU` keyword is set to anything else than 0, because of the usage of the /DECOMPOSED keyword to calls to the IDL DEVICE routine. In principle, nothing should prevent the PSM to run under any IDL version between 7.1.1 and 8.1.

2.1.4 astron

The PSM uses the IDL Astronomy User's Library (`astron`), downloadable from:
http://idlastro.gsfc.nasa.gov/ftp/
The present version of the PSM has been developed with the dec. 2010 version of `astron`. Earlier versions can be the source of problems in the handling of fits headers (errors in calls to the `SXPAR` function of the `astron` package).

2.1.5 HEALPix

A fully functional installation of the `HEALPix` package is required. In particular the `anafast_cxx` and `alm2map_cxx` binaries should be in the execution path. The `HEALPix` package can be downloaded from:
http://healpix.jpl.nasa.gov/
The present version of the PSM is compatible with version 2.14 and 2.15a of `HEALPix`.

2.1.6 MPFIT

The PSM uses the `MPFIT` fitting library by Craig Markwardt, which can be downloaded from:
http://www.physics.wisc.edu/∼craigm/idl/fitting.html

2.1.7 CAMB

For generating a CMB model using CMB power spectra computed from user-specified cosmological parameters, a fully functional installation of a Boltzman code is needed. The default option is to use the `CAMB` package, which can be downloaded from:
http://camb.info/
The present version of the PSM has been developed and tested with the jan. 2010 version of `CAMB`.

2.1.8 CLASS

`CLASS` can be used by the PSM as an alternative to `CAMB` for computing CMB and matter power spectra. For ease of use, the proper version of `CLASS` to be used with a particular PSM release is included in the PSM distribution, in the `libraries/class/` subdirectory of the PSM software directory. The `CLASS` software is required to generate far infrared background fluctuations according to the castex2012 model. The appropriate version of the `CLASS` software for the present PSM release is included in the PSM distribution, in the `external` subdirectory of the psm software directory. Further information about `CLASS` can be found in the `CLASS` website: http://class-code.net

2.1.9 CGIS

Some software tools, included in the PSM software but not used in normal PSM runs, call routines from the CGIS library (COBE analysis software). For full consistency, you may opt to include this library in your IDL path. The library can be downloaded from:
http://lambda.gsfc.nasa.gov/product/cobe/cgis.cfm

2.2 Getting and installing the code

The PSM is made available as version-tagged and documented releases available at the following URL:
http://www.apc.univ-paris7.fr/∼delabrou/PSM/psm.html.

To install the PSM,

- Retrieve and extract the tarball of the code.
- Adapt your `IDL_STARTUP` file as follows :
  - Define the `PSMROOT` IDL system variable to point to the root of the package. This can be done, for instance, with the command line :

    ```
    DEFSYSV, '!PSMROOT', '/Path/Towards/PSMROOT/'
    ```

    The PSM software itself should be installed in a subdirectory called `Soft/` of this PSM root directory (the root directory will also contain a `Data/` subdirectory, in which PSM input data will be copied.
  - Add the routines of the package to your path, for example :

    ```
    !path=!path+':'+expand_path('+'+!PSMROOT+'/Soft/')
    ```
  - Add the `astron`, `HEALPix`, `MPFIT` and `CGIS` IDL routines to your path – preferably in that order.
- Make sure a compiled version (executables) of `HEALPix` C++ and F90 routines are in your unix execution path.
- Optionally, download the `CAMB`, compile it, and make sure that the executable is in your unix execution path. This is necessary for running a PSM simulation with the `RUN_CAMB` parameter set to `yes` (see section 6.2).
- Optionally, compile `CLASS` and/or `ilens` packages, both provided in the `Soft/libraries/` subdirectory of the PSM software distribution, and make sure that the executables are in your unix execution path. This is useful for some of the advanced options in the PSM, respectively when you set the `RUN_CLASS` parameter to `yes` (see section 6.2) and when you set the `CMB_LENSING` parameter to `ilens` (see section 8).

2.3 Getting PSM data

The PSM software uses a large set of miscellaneous data (observations, simulations, data files describing instruments, etc...) which must be downloaded to the local machine for proper PSM run. By default, during program execution, the PSM checks for the availability of the PSM data sets needed for the simulation. If any required data set is not available, a request is made to the PSM data base (using `wget`). The data is copied into a subdirectory of the `PSMROOT` directory called `Data/`. PSM data accumulates there as consecutive PSM runs are made (see, however, the use of the `GET_DATA` parameter in section 4.3). We recommend that users that normally use the PSM on machines connected to the web use this particular feature to let the PSM download only those files that they need for the type of simulations they are doing.

An alternate option is to retrieve PSM data before any PSM run. This is recommended if you install the PSM with the objective or running many PSM simulations with various configuration files. Then, use the `GET_PSM_DATA` procedure as follows:

```
IDL> GET_PSM_DATA
```

This, however, will download *the full PSM data directory*, which contains several sample simulations made with different seeds, and amounts to hundreds of GBytes of data. It should be avoided if you have only limited storage on your computer (e.g. avoid using this command on your laptop...).

Under normal operation of the PSM software, the `PSMROOT` directory contains at least the PSM software directory `Soft/`, and the PSM data directory `Data/`.

Note that some PSM data is presently restricted to the Planck collaboration. Retrieval of such data (limited in the present version to somewhat more detailed description of the Planck instrument for Planck-specific simulations) requires a username and a password. Restricted PSM data is obtained with the command:

```
IDL> GET_PSM_DATA, /private
```

Upon this command, you will be asked for a username and password (available only to members of the Planck collaboration).

3 Running the code

3.1 Running the PSM

All the parameters of the simulation code are set in the single text *configuration file* specified as an argument to PSM_MAIN. By default, if no argument is present, the PSM runs using the **config.psm** configuration file which is found in the PSMROOT/Soft/psm/config/ directory. For using your own configuration file, copy **config.psm** to your favorite directory for PSM configuration files, rename it if you wish (e.g. **myconfig.psm**), edit it to suit your needs, and use it as input to PSM_MAIN as specified below.

To run the simulation, within the IDL environment, call the PSM_MAIN routine with the name of the parameter file (provide either full path, or relative path), e.g.:

```
IDL> PSM_MAIN, '/full/path/towards/config/file/myconfig.psm'
```

or, for instance, if the configuration file you wish to use (here **myconfig.psm**) is in your working directory, simply

```
IDL> PSM_MAIN, 'my_config.psm'
```

Upon execution, the PSM software looks for the specified configuration file on the local disk. If a simple filename is provided, the configuration file is looked for in the working directory first, and if no such file is present the PSM looks for the configuration file in the $PSMROOT/Soft/psm/config/ directory.

3.2 Optional keywords

The PSM_MAIN routine can be launched with a number of optional keywords, which bypass any equivalent definition of the same keywords in the configuration file.

3.2.1 The output_dir keyword

All products of the PSM run are written in a directory (the output directory) specified by the user. This output directory is created by the PSM process if it does not already exist.

The name of the PSM output directory can be passed to PSM_MAIN in two ways. Either it can be written in the configuration file (OUTPUT_DIRECTORY parameter), or it can be supplied as a keyword in the call of PSM_MAIN. If the keyword is set, it overwrites whatever is written in the configuration file. The PSM is launched with the output directory supplied by keyword with the command:

```
IDL> PSM_MAIN, 'config.psm', output_dir='path/towards/output/directory/'
```

3.2.2 The check_param keyword

If the check_param keyword is set in the call to PSM_MAIN, the PSM simply prints out information about the input parameters, without actually running the code to produce PSM outputs. This permits one to check easily, prior to running the PSM, what parameters will actually be used during the run. This is achieved by entering at the IDL prompt the command:

```
IDL> PSM_MAIN, 'config.psm', /check_param
```

3.2.3 The `debug` keyword

There is also a `debug` mode, which bypasses the error catch in component pipes. The debug mode (as the name indicates) is useful for debugging, as it permits to find where an error actually occurred. The `PSM_MAIN` routine is launched in debug mode with the command:

```
IDL> PSM_MAIN, 'config.psm', /debug
```

3.2.4 The `carefulness` keyword

The `PSM_MAIN` routine can be run with different levels of carefulness. This is done by setting the `carefulness` keyword. By default, the PSM is moderately careful, overwriting existing files if any, and basically trusting the user to know what he/she is doing. By setting `carefulness` to 1, moderate care is taken – the PSM performs some elementary checks during its execution, stopping if an unexpected situation is encountered. By setting the `carefulness` parameter to 2, the PSM is very careful during its execution (in particular, no existing file is erased). If carefulness is set to 3, no missing parameter is set to a default value, and consistency checks are performed all along the PSM run. The carefulness is set by assigning a value to the `carefulness` keyword, e.g.:

```
IDL> PSM_MAIN, 'config.psm', carefulness=2
```

Note: At present, setting `carefulness` to a non-zero value causes the PSM to crash, because of missing information in some file headers. This will be fixed ASAP.

3.2.5 The `verbosity` keyword

The `PSM_MAIN` routine can be run with different levels of verbosity. This is done by setting the `verbosity` keyword. By default, the PSM writes little information on the ongoing processes. Set the `verbosity` keyword to 1 or 2 for increasing amount of information about the run, e.g.

```
IDL> PSM_MAIN, 'config.psm', verbosity=1
```

3.3 Outputs of the PSM

The outputs of the code are sets of maps, catalogues of objects, IDL save sets, and text files. The various outputs are organized in subdirectories of the output directory, as described in section 15.1.

3.4 Consecutive PSM runs

It is possible to run the PSM several times in a row, using different configuration files, but using the same output directory. Each consecutive run then updates the content of the PSM output directory, according to the instructions given in the parameter file. The history of PSM runs for a given output directory is written in the `psminfo` subdirectory of the output directory. All individual parameter files are copied, with a naming of the form: `config-n_xxxxxxxxxxxxxxxx.psm`, where, `n` stands for the order of the run in the consecutive run list, and `xxxxxxxxxxxxxxxx` is a 16-ASCII character code assigned to each PSM run, that helps trace the run that produced any particular data product.

3.5 Monte-Carlo simulations using the PSM

At present, there is no standard way to generate Monte-Carlo PSM simulations, i.e. make many runs with varying input parameters. For doing MC simulations, one has to write an external piece of software that writes or modifies PSM configuration files, and launches the `PSM_MAIN` procedure with these different configuration files as inputs.

4 Configuration files

Except for the keywords of the `PSM_MAIN` procedure, described in section 3.2, the PSM input parameters setting the configuration of the run are collected into one single parameter file for a given run.

For a given version of the PSM, the parameter file defines completely the output of the code. Simulations are then reproducible – one merely has to run the same version of the PSM with the same input parameters. As mentioned above, for this purpose, a special version of the input parameter file, `config.psm`, with all parameters explicitly set to their used values, is stored with the output data.

4.1 Syntax for editing configuration files

Parameter files consist in a set of couples keyword-value separated by the symbol '='. Keyword names are case insensitive (but keyword values are not, *e.g.* mJy/sr $\neq$ MJy/sr).

All lines starting with a `#` are considered as comment lines and ignored (as well as empty lines) when the configuration file is read.

4.2 User-ready configuration files

The file `config.psm` in the `Soft/psm/config` can be used as an example to set-up user-specific configuration files. Comments explain the role of most of the keywords, and the various options.

4.3 Global parameters of the PSM run

This section describes the global parameter keywords used by the PSM run. Table 1 summarizes these parameters dedicated to the general setting.

keyword name	description / comments	accepted values	default
`OUTPUT_DIRECTORY`	Specify the path to the output directory (will be created)	any valid path	`./PSM_OUTPUT`
`PRECISION`	Floating point precision of the output maps	single, double	single
`FIELDS`	Model and process temperature only or both temperature and polarisation	T, TP	T
`CLEAR_ALL`	Erase all existing directories and files in the PSM output directory	yes, no	no
`VISU`	Level of visualisation of PSM outputs	0, 1, 2	0
`OUTPUT_VISU`	On what support is the visualisation made	screen, png, ps	screen
`SEED`	Specify the seed for random number generation	any positive long integer	1
`GET_DATA`	Option for getting PSM data during the run. - 0: no retrieval - 1: get data if missing on local disk - 2: update data if repository more recent	0, 1, 2	1

Table 1: Global parameters of the PSM run

4.3.1 `OUTPUT_DIRECTORY`

The following line sets the name of the output directory to be `My_PSM_run`, in your current working directory (the directory from which the PSM is launched):

```
OUTPUT_DIRECTORY = ./My_PSM_run
```

4.3.2 `PRECISION`

Setting `PRECISION` to `double` results in almost all PSM calculations being done in double precision (8 bytes per real number). In addition, some approximations are bypassed, integrals are computed using more integration points, and output data are written in double precision (requiring about twice the disk space). When `PRECISION` is set to `single`, some calculations are still made in double precision whenever necessary, but overall most of the calculations are performed in single precision (4 bytes per real number), and outputs are stored in single precision.

4.3.3 `FIELDS`

The PSM can be run in modes where either only temperature data, or both temperature and polarisation data, are produced.

When `FIELDS` is set to `TP`, the maps of diffuse sky emission are polarised when the corresponding sky emission is polarised, and when the relevant parameter is different for T, Q and U. For instance, the map of synchrotron amplitude produced by the PSM will be polarised, but the map of thermal SZ effect will not. Catalogues of points sources will be polarised.

When `FIELDS` is set to `T`, the observations of the sky with instruments (output in the `/observations` subdirectory of the PSM output directory) will be temperature only, irrespective of the polarisation state of the sky model (which could have been generated by a previous PSM run, in which the FIELDS keyword could have been set to T), or of the polarisation capability of the instrument(s).When FIELDS is set to TP, the polarisation state of the observations for each channel depends on both the sky model polarisation, and on the polarisation capability of the particular channel.

However, the polarisation state of band-integrated sky maps (output in the /skyinbands subdirectory of the PSM output directory) is set by the polarisation state of the sky model, irrespective of the polarisation capability of the instrument(s).

4.3.4 CLEAR_ALL

When the CLEAR_ALL keyword is set to yes, all the subdirectories of the PSM output directory that are created by the PSM (and listed in section 15.1) are emptied and erased, unless the carefulness keyword (described in section 3.2) is set to 2.

4.3.5 VISU

The VISU parameter sets whether the PSM run produces output visualisation of the modelled sky components and observed maps. It can range from 0 (no visualisation) to 2.

4.3.6 OUTPUT_VISU

The OUTPUT_VISU parameter sets the support for visualisation, which can be screen, png, or ps. The first two require the X window graphic system device to be operational for your ongoing IDL session. png or ps outputs are written in the figures/ subdirectory of the PSM output directory, with self-explanatory names. Visualisation is useful for checking that the outputs of the PSM run look as expected.

4.3.7 SEED

The PSM is designed in order to avoid, as much as possible, unintended correlation between random numbers drawn by different parts of the package, while keeping the ability to reproduce simulations. The SEED parameter is a long integer which provides the starting point for the random number generator. Details on how the seeds are handled by the PSM are given in section 17.4.

Note: Rerunning the PSM with the same seed but a different set of parameters (different nside, lmax, number of components, etc... generates a somewhat different sky, as the number of random draws depends on the parameters). Hence, reproducing the same sky requires not only the same seed, but the same parameter file in general.

4.3.8 GET_DATA

The input data sets used by the PSM are stored on a central repository. For proper operation of the PSM, some of these data should be copied onto the machine where the PSM is run (see section 2.3).

The GET_DATA keyword sets the way the PSM run deals with input data sets during the PSM normal execution. A value of 0 means that the PSM run stops (or fails) if required data is missing on disk. A value of 1 means that the PSM run automatically tries to download the required data from the PSM data directory, and the PSM run fails only if the data retrieval was unsuccessful (this is the default option). A value of 2 means that for each input file, the PSM is checking the data repository for the date-stamp of the data to be used. If the data stored on the central repository is more recent than the local copy, the latter is updated (and the previous version is erased). Although in principle no data set in the PSM data repository is replaced (new files with different names are created instead), avoid using GET_DATA = 2 if you wish to maintain traceability of preexisting simulations with one particular version of the PSM software.

PSM users are advised to use responsibly the options of GET_DATA. If you plan to run many simulations of the PSM, consider downloading the PSM data directory once and for all as explained in section 2.3.

5 The sky model

5.1 Global parameters of the sky model

This section described the general parameters that define the global properties of the modelled sky. Table 2 summarizes these parameters.

keyword name	description / comments	accepted values	default
`SKY_TASK`	Specify the sky modeling task	`new`, `restore`	`new`
`SKY_RESOLUTION`	Resolution of sky maps in arc-minutes (Gaussian)	any positive real number	15
`SKY_LMAX`	Maximum ℓ value of the modelled sky	any positive integer	1536
`SKY_PIXELISATION`	Pixelisation used to map the sky	`HEALPIX`	`HEALPIX`
`HEALPIX_NSIDE`	`nside` parameter for sky HEALPix maps	1, 2, 4, 8, 16 ...	512
`SKY_PIXWINDOW`	Whether sky maps are sampled at pixel centers (0) or averaged over pixel areas (1)	0, 1	1
`WRITE_ANCILLARY`	If set to `yes`, some ancillary data is written in the `/ancillary` subdirectory of the PSM output directory	`yes`, `no`	`no`

Table 2: Global parameters used by the PSM to model the emission of the sky

5.1.1 `SKY_TASK`

The keyword `SKY_TASK` sets whether the PSM run should produce a new model of the sky (`SKY_TASK = new`), or use a model of the sky already existing in the PSM output directory and simply perform the band-integration of this pre-existing model, and its observation with an instrument (`SKY_TASK = restore`).

5.1.2 `SKY_RESOLUTION`

The keyword `SKY_RESOLUTION` sets the resolution (understood as the size of an hypothetical Gaussian beam) at which the component maps should be created.

Note that the resolution of the map (i.e. an equivalent Gaussian smoothing applied to the full resolution sky) is different from the pixel size (which is set with `HEALPIX_NSIDE`) and from the resolution of the instrument(s) observing the sky. In the PSM, the `SKY_RESOLUTION` parameter exists for making sure that all maps are properly sampled. For a proper simulation pipeline, the resolution should be of the order of (or smaller than) the resolution of the final observations. The sky pixel size should then be about 1/3 of the map resolution or smaller. For instance a resolution of 5 arcminutes corresponds approximately to `lmax` = 4000 (but larger lmax is preferable for accurate simulations), and `nside` = 2048.

It is possible to set the sky resolution to 0. If this is done, it is recommended that you generate the sky model with large `HEALPIX_NSIDE` and large `SKY_LMAX`, to avoid aliasing due to improper sampling. Note that ringing around strong point sources is then be expected in the final observations, if strong point sources are co-added in the final observation maps.

5.1.3 SKY_LMAX

Many of the PSM maps are generated in the harmonic domain. Spherical harmonic transforms are used to change the resolution of the maps whenever necessary. The SKY_LMAX keyword sets the harmonic band limit of the PSM simulation. For specific components (e.g. the dipole) the actual maximum multipole order can be lower, but no map of the sky will have non-vanishing harmonic coefficients at ℓ larger than SKY_LMAX. For all PSM maps and harmonic coefficients files, the maximum ℓ value of the data is written in the fits header(s), using the fits parameter PSM_LMAX (see section 16).

5.1.4 SKY_PIXELISATION

This parameter sets the pixelisation scheme used to represent the sky model. At present, only HEALPIX is implemented.

5.1.5 HEALPIX_NSIDE

Set this parameter to the `nside` parameter of the HEALPix pixelisation. See the SKY_RESOLUTION parameter for a rule of thumb for an appropriate choice of HEALPIX_NSIDE as a function of the sky resolution and of the harmonic band limit SKY_LMAX of the sky model.

5.1.6 SKY_PIXWINDOW

Maps of the sky can be viewed as a sampled version of the underlying sky emission (in the usual sense of the sampling theorem). In this case, the map value assigned to a given pixel is the sky emission at the center of that pixel. Alternatively, a map can be viewed as a tiled approximation of the underlying sky, in which case each pixel contains the integral of the sky emission in the pixel area.

On flat 2-dimensional images with equally spaced samples or pixels, for properly sampled band-limited images, the tiled version of the image is obtained by sampling the image convolved with the pixel-shaped square kernel. The Fourier transform of the tiled image is obtained from the Fourier transform of the sampled image by simple multiplication by the Fourier transform of the kernel.

On the sphere, the integration of sky emission in HEALPix pixels is approximately equivalent to multiplying the $a_{\ell m}$ coefficients of its spherical harmonic transform by the HEALPix 'pixel window function'. Setting SKY_PIXWINDOW to 1 results in PSM maps to be averaged in pixels of the size specified by HEALPIX_NSIDE, and multiplying harmonic coefficients by the corresponding HEALPix pixel window function.

For all PSM maps and harmonic coefficients files, the pixel window function applied to the data is written in the fits header(s), using the parameter PXWIN in the fits header. PXWIN gives the value of `nside` corresponding to the pixel window function applied to the data, i.e. PXWIN=512 for a map (at any `nside`) for which the pixel window function applied is that of a nside=512 healpix map (see section 16).

5.1.7 WRITE_ANCILLARY

If the WRITE_ANCILLARY parameter is set to `yes`, the PSM writes (in the `ancillary` subdirectory of the PMS output directory) ancillary catalogues of the PSM point sources: infrared sources as they would be observed by IRAS at 100 and 60 microns, radio sources as they would be observed at 0.84, 1.4 and 4.85 GHz, and ultracompact H-II regions as they would be observed at all these frequencies. The measured ancillary fluxes comprise errors with respect to modelled source fluxes, which are compatible with the measurement errors in the actual data. These ancillary catalogues are written in the format of IDL save sets.

5.2 Models for all components

The main components generated by the PSM are the CMB dipole (`dipole`), CMB anisotropies (`cmb`), the thermal and kinetic SZ effects (`sz`), Galactic emission from the diffuse inter-stellar medium which comprises thermal dust, spinning dust, synchrotron, free-free, and CO lines (`galaxy`), emission from radio-sources, infrared sources, ultra-compact H-II regions, and WMAP unresolved sources, collectively denoted as point sources (`ps`), and the far infrared background (`firb`).

For each of these components, several different models are available, each of which specified with a number of model-specific parameters (detailed in sections 7 to 12). Table 3 summarises the available models for each of the main components.

component	**description / comments**	**accepted models**	**default**

dipole	The CMB 'cosmological' dipole, due essentially to the motion of the solar system	`no_dipole`
		`prediction`
		`generic`
			`no_dipole`

cmb	The CMB anisotropies, including lensing, ISW, and re-ionisation effects	`no_cmb`
		`prediction`
		`gaussian`
		`nongaussian_fnl`
			`no_cmb`

SZ effect	The Sunyaev-Zel'dovich emission, including thermal and kinetic effect	`no_sz`
		`prediction`
		`dmb`
		`nbody+hydro`
		`hydro+dmb`
			`no_sz`

galaxy	The emission of the galactic inter-stellar medium, including thermal dust, spinning dust, synchrotron, free-free, and CO emission lines	`no_galaxy`
		`prediction`
		`simulation`
			`no_galaxy`

point sources	The emission of galactic and extra-galactic point sources (radio and infrared, ultracompact H-II regions)	`no_ps`
		`prediction`
		`simulation`
			`no_ps`

firb	The emission of the background of blended extra-galactic infrared point sources	`no_firb` `simulation`	`no_firb`

Table 3: Available models for the different components of sky emission included in the PSM

6 Cosmology

The PSM produces sky simulations on the basis of an underlying assumed ΛCDM cosmological model. The main parameters of the model are listed in 6.1. These parameters are adjustable by the PSM user, and are used as input parameters for running `CAMB` and/or `CLASS` to compute CMB and/or matter power spectra, as well as to generate shells of density contrast at various redshifts (see 6.2).

6.1 Cosmological parameters

Cosmological parameters are used in various parts of the PSM sky simulation. As much as possible, the same set of parameters is used everywhere in the simulation. The only exception to this is when pre-computed maps of some component are used, in which case these specific maps use the cosmological model assumed for their generation, which can be different from the global cosmological parameters defined by the PSM user.

Cosmological parameters are used, in particular, for the computation of a CMB power spectrum with CAMB in the `gaussian` CMB model, and for the computation of cluster mass functions in the `dmb` and `hydro+dmb` models of SZ emission.

Table 4 summarizes the cosmological parameters used by the PSM. Note that in principle σ_8 can and should be computed from the other parameters. This is not fully implemented for the moment. It is up to the user to make sure that the scalar amplitude (used to normalise the CMB scalar anisotropies) and σ_8 (used to generate catalogues of SZ clusters), both provided as input, are compatible with the same cosmological model.

parameter	**description / comments**	**default**
`T_CMB`	CMB temperature (Kelvin)	2.725
`H`	Hubble parameter at present time ($\texttt{H} = h = H_0/100$, with H_0 in km/s/Mpc)	0.704
`OMEGA_M`	Matter parameter density	0.272
`OMEGA_B`	Baryonic matter parameter density	0.0456
`OMEGA_NU`	Neutrino matter parameter density	0
`OMEGA_K`	Curvature parameter Ω_k. This sets the dark energy density parameter as $\Omega_\mathrm{DE} = 1 - \Omega_\mathrm{k} - \Omega_\mathrm{m}$	0
`SIGMA_8`	Amplitude of matter perturbations at the scale of $8h^{-1}$ Mpc	0.809
`N_S`	Scalar spectral index n_s of primordial fluctuations	0.963
`N_S_RUNNING`	Running of the scalar spectral index n_s	0
`N_T`	Tensor spectral index n_t of primordial fluctuations	0
`N_T_RUNNING`	Running of the tensor spectral index n_t	0
`R`	Tensor to scalar ratio (primordial power at k_pivot)	0.05
`TAU_REION`	Reionisation optical depth	0.087
`HE_FRACTION`	Helium fraction (by mass)	0.24
`N_MASSLESS_NU`	Number of massless (i.e. relativistic) neutrino species	3.04
`N_MASSIVE_NU`	Number of massive neutrino species	0
`W_DARK_ENERGY`	w parameter for the equation of state of dark energy	-1
`K_PIVOT`	The comoving scale k_pivot (in Mpc^{-1}) at which the amplitudes of initial scalar and tensor power spectra are defined	0.002
`SCALAR_AMPLITUDE`	Amplitude of scalar modes	2.441e-9

Table 4: Cosmological parameters read-out from the configuration file by the PSM

6.2 Density fluctuations and cosmic structure

Perturbations of the spacetime metric along the cosmic history are of major importance for the PSM. Perturbations in the early universe, mostly around the epoch of last scattering ($z \simeq 1100$), give rise to CMB anisotropies. Late time perturbations ($z \leq 10$) are linked to cosmic structure, that defines the distribution of galaxies and clusters of galaxies. The PSM uses `CAMB` and/or `CLASS` to compute CMB temperature and polarisation power spectra, lensing potential, and matter power spectra at late time.

The following parameters are used to run `CAMB` and `CLASS` during the execution of a PSM run, to generate CMB and matter power spectra corresponding to the cosmological parameters listed in 6.1, and to select what is used to compute the matter power spectrum used in the PSM run. Table 5 lists the options.

parameter	**description / comments**	**accepted values**	**default**
`RUN_CAMB`	Whether to compute CMB and matter power spectra using `CAMB`	`yes`, `no`	`no`
`RUN_CLASS`	Whether to compute CMB and matter power spectra using `CLASS`	`yes`, `no`	`no`
`CMB_CL_SOURCE`	Source for CMB C_ℓ and lensing potential	`standard-LCDM`, `CAMB`, `CLASS`	`CAMB`
`COSMO_PK_SOURCE`	Source for computing the matter power spectrum $P_k(z)$	`EisHu`, `BBKS`, `CAMB`, `CLASS`	`EisHu`

Table 5: Parameters specifying whether CMB and matter power spectra are computed with none, either or both of `CAMB` and `CLASS`, and what is used to compute CMB, lensing, and matter power spectra in the PSM run.

The `RUN_CAMB` and `RUN_CLASS` parameters are compatible, i.e. it is possible to run both `CAMB` and `CLASS` (for instance, for comparing their outputs), or none, or one of them only. The outputs actually used for generating CMB fluctuations are specified with the `CMB_CL_SOURCE` parameter. The outputs used to compute the matter power spectrum as a function of redshift z are specified with `COSMO_PK_SOURCE`.

6.2.1 `RUN_CAMB`

When the `RUN_CAMB` parameter is set, the PSM run calls the `CAMB` software to generate CMB and matter power spectra according to the cosmological parameters defined in section 6.1. The PSM generates an input parameter file (saved in the `cosmo/camb/` subdirectory of the PSM output directory) that is used for running `CAMB`. The outputs of `CAMB` are saved in the same directory. They are optionally used for generating the CMB and for computing the matter power spectrum used in some of the models of SZ emission.

6.2.2 `RUN_CLASS`

When the `RUN_CLASS` parameter is set, the PSM run calls the `CLASS` software to generate CMB and matter power spectra according to the cosmological parameters defined in section 6.1. The PSM generates an input parameter file for `CLASS` (saved in the `cosmo/class/` subdirectory of the PSM output directory). The outputs of `CLASS` are saved in the same directory. They are optionally used for generating the CMB and for computing the matter power spectrum used in some of the models of SZ emission.

6.2.3 `CMB_CL_SOURCE`

This parameter sets the origin of the CMB power spectrum (or spectra) used to generate the CMB. The `standard_LCDM` option uses the current best fit model CMB power spectrum described in section 8.1. The `CAMB` option uses the outputs of `CAMB`, and the `CLASS` option the outputs of `CLASS`, to generate the CMB and lensing potential C_ℓ. In the last two cases, the cosmological parameters described in section 6 are used to generate the CMB temperature and polarisation power spectra.

6.2.4 `COSMO_PK_SOURCE`

This parameter sets the origin of the matter power spectrum $P(k)$. The default value is `EisHu`, in which case the approximation of Eisenstein and Hu is used. The other cases are not fully implemented or tested yet.

7 The CMB dipole

The CMB dipole is an important term of sky anisotropies in the millimeter wavelengths range. It can been used for absolute or relative calibration of CMB observations. Two distinct models of CMB dipole emission are implemented in the PSM: `prediction` and `generic`. The output modelled dipole data are stored in the `components/dipole/` subdirectory of the PSM output directory.

7.1 `prediction` CMB dipole

The `prediction` model generates a dipole with user-defined amplitude, galactic longitude, and galactic latitude. By default, these parameters match the measurement of WMAP (7 year data release). Default values are listed in table 6.

keyword name	**description / comments**	**accepted values**	**default**
`DIPOLE_GLON`	Dipole galactic longitude (degrees)	angle in [0,360[	263.99
`DIPOLE_GLAT`	Dipole galactic latitude (degrees)	angle in [-90,90]	48.26
`AMPLITUDE`	Dipole amplitude (`mK_CMB`)	any positive value	3.355

Table 6: Parameters used by the PSM to model the CMB dipole

7.2 `generic` CMB dipole

The `generic` model draws at random the dipole amplitude and coordinates according to Gaussian laws. The default distributions are set using the same parameter names as in the `prediction` model and centered on the WMAP measurement by default (table 6). Standard deviations are set with the parameters listed in table 7, and have default values equal to the WMAP measurement error bars. Note that these uncertainties are used to generate a random dipole only if the dipole model is set to `generic`.

keyword name	**description / comments**	**accepted values**	**default**
`DIPOLE_GLON_ERROR`	Uncertainty (1σ) on dipole galactic longitude (degrees)	any positive value	0.14
`DIPOLE_GLAT_ERROR`	Uncertainty (1σ) on dipole galactic latitude (degrees)	any positive value	0.03
`AMPLITUDE_ERROR`	Uncertainty (1σ) on dipole amplitude (`mK_CMB`)	any positive value	0.017

Table 7: Parameters used by the `generic` CMB dipole model in the PSM to model random deviations of the dipole from its nominal amplitude and direction

8 Cosmic Microwave Background anisotropies

The PSM provides maps and power spectra of the CMB anisotropies (temperature and polarisation). The output modelled CMB data are stored in the `components/cmb/` subdirectory of the PSM output directory. The theoretical C_ℓ of the simulation are stored in the `cmb_cl.fits` file, and the CMB map and harmonic coefficients are stored in `cmb_map.fits` and `cmb_alm.fits` respectively.

8.1 `prediction` CMB model

The CMB prediction (selected in the PSM configuration file by setting `CMB_MODEL = prediction`) is derived from a CMB map obtained on WMAP 5-year data using a needlet ILC component separation method. The predicted CMB temperature anisotropies estimates 'best' (in a least square sense) the sky CMB emission at the target sky resolution. Note that the actual resolution of the map is set by the required sky resolution, the resolution of the available CMB observation, and the map signal-to-noise ratio. The `prediction` CMB is a result of a compromise between CMB error and noise contamination (in the Wiener sense). The input CMB map is the NILC5 map stored in the PSM input ancillary data in:

```
Data/ancillary/observations/WMAP/NILC-CMB-5yr/mapilc5yr.fits
```

This map is then Wiener-filtered to minimize the total RMS error for the target sky resolution.

The corresponding assumed CMB power spectrum is a default 'WMAP-fit' obtained from running CAMB online on the Lambda web site, using the following parameter file, available in the PSM ancillary data products (and, if not already available on the user computer, read from the PSM data base upon normal PSM execution):

```
Data/ancillary/models/cmb/standard_lcdm/camb_42481064.ini
```

The CMB power spectrum is read from the output of the CAMB run available in the same data directory, either in `camb_42481064_lensedcls.dat` or in `camb_42481064_scalcls.dat`, depending on whether the `CMB_LENSING` parameter is set to `cl` or something different in the PSM configuration file. This parameter is the only one specifically used for the predicted CMB (in addition to all global PSM and sky model parameters described in section 4.3 and 5.1).

Note that the choice of C_ℓ (lensed or not) has no impact on the output CMB map. It only impacts the model power spectrum written as an output of the PSM run in the `component/cmb/` subdirectory of the ouput directory.

Note also that on the basis of the theoretical correlation between E and T, the CMB `prediction` model produces not only a best guess for CMB temperature, but also for CMB E modes of polarisation. In the present version of the PSM, the B modes vanish.

8.2 gaussian CMB model

The `gaussian` CMB model is selected in the PSM configuration file by setting `CMB_MODEL = gaussian`. The CMB map produced by the PSM is a random generation of CMB harmonic coefficients $a_{\ell m}$ according to Gaussian statistics defined by an input CMB power spectrum (temperature, and polarisation).

The parameters of the model are specified in table 8. Their impact on the simulated CMB is detailed below.

keyword name	description / comments	accepted values	default
`CMB_CONSTRAINED`	Whether the CMB realisation should be constrained to match the observed CMB	`yes`, `no`	`no`
`CMB_LENSING`	Method used to do the CMB lensing, if any	`cl`, `ilens`, `none`	`cl`

Table 8: Parameters used by the `gaussian` CMB model in the PSM

8.2.1 CMB_CONSTRAINED

Setting this parameter to `yes` amounts to forcing the simulated CMB to match WMAP observations. The simulated CMB is then the sum of the predicted CMB described in section 8.1, and of randomly generated 'missing power'. On large scales, the CMB then essentially matches WMAP observations, whereas on small scales it is essentially random. Note that in spite of the name of the model, the constrained Gaussian CMB map may be detectably non-Gaussian, since the WMAP map used to constrain the CMB realisation is itself slightly non-Gaussian (by reason or low-level residual foregrounds in the map, for instance, if nothing else).

Setting this parameter to `no` will result in a totally random Gaussian CMB realisation.

8.2.2 CMB_LENSING

Lensing of the CMB by large scale structure generates small shifts of the CMB temperature and polarisation patterns on the sky. This in turn changes the power spectrum of temperature and polarisation anisotropies.

CMB lensing can be made in two ways with the PSM: either at the level of the theoretical power spectrum, or at the level of the maps. In the first case, the generated CMB will still be Gaussian, but the CMB power will be modified to take into account the impact of lensing. In the second case, unlensed CMB maps are generated first, and are subsequently lensed by shifting the temperature and polarisation patterns of the CMB anisotropies. Set `CMB_LENSING` to `cl` to generate Gaussian CMB maps with a lensed power spectrum, and to `ilens` to generate CMB maps with lensing effect implemented on maps. The second option makes use of a map of lensing potential generated on the basis of CAMB power spectra. It requires significant memory (of order 40 GBytes) for lensing polarised maps at HEALPix `nside`=2048 and for `SKY_LMAX`=4300.

8.3 nongaussian_fnl CMB model

The PSM can produce simulated CMB maps with non-gaussianity of the local type. Such CMB realisations have been precomputed and are stored in the PSM data repository, both at `lmax=1024` (1000 realisations), or at `lmax=3500` (100 realisations). The non-Gaussian CMB model assumes a linear-plus-quadratic model for Bardeen's gauge-invariant curvature potential, where the contribution of the quadratic term is given by a single parameter $f_{\rm nl}$.

The parameters of the non-Gaussian CMB model are specified in table 8. Their impact on the simulated CMB is detailed in the following paragraphs.

keyword name	**description / comments**	**accepted values**	**default**
NG_SIMUSET	What set of input simulations to use	`elsner1024`, `elsner3500`	`elsner3500`
HI_ELL_EXTEND_GAUSSIAN	Whether to add gaussian fluctuations at $\ell >$ 3500	`yes`, `no`	`no`
READJUST_NG_SPECTRUM	Whether the power spectrum of the simulated non-gaussian template should be readjusted to match the expectation for the input cosmological parameters	`yes`, `no`	`no`
DRAW_F_NL	Whether the value of $f_{\rm nl}$ is drawn at random	`yes`, `no`	`no`
F_NL_MIN	Minimum value of $f_{\rm nl}$ if `DRAW_F_NL = yes`	any number	-30
F_NL_MAX	Maximum value of $f_{\rm nl}$ if `DRAW_F_NL = yes`	any number	30
F_NL	Fixed value of $f_{\rm nl}$ if `DRAW_F_NL = no`	any number	10
DRAW_NG_MAP_NUMBER	Whether the number of the precomputed non-Gaussian CMB map used is drawn at random	`yes`, `no`	`no`
NG_MAP_NUMBER_MIN	lowest map number if the map number is randomly drawn	positive integer	1
NG_MAP_NUMBER_MAX	highest map number if the map number is randomly drawn	positive integer	1
NG_MAP_NUMBER	map number used if the map number is not randomly drawn	positive integer	1

Table 9: Parameters used by the `nongaussian_fnl` CMB model in the PSM

8.3.1 NG_SIMUSET

The non-Gaussian simulations available in the PSM comprise two sets of maps. The first one comprises 1000 maps at maximum harmonic ℓ of 1024, and the other one 100 maps at maximum harmonic ℓ of 3500. The `NG_SIMUSET` parameter is used to decide among which of these simulation sets the CMB map will be drawn.

8.3.2 HI_ELL_EXTEND_GAUSSIAN

Set this parameter to `yes` to extend the CMB to harmonic modes ℓ higher than the limit in the original simulation. The additional small scales will be Gaussian.

8.3.3 READJUST_NG_SPECTRUM

Setting this keyword to 'yes' results in scaling the linear scalar modes of the non-Gaussian simulation to match the power spectrum corresponding to the cosmology set by the PSM user (with the cosmological parameters discussed in section 6). The cosmological parameter set used to generate the non-Gaussian simulations is the default PSM cosmology (corresponding to WMAP 7year + BAO + H0). The corresponding CMB power spectrum (scalar modes) is $[C_\ell^{ng}]$. For a different cosmology, we should have instead $[C_\ell^{psm}]$. If READJUST_NG_SPECTRUM is set to yes, then the linear part of the scalar modes of the maps are scaled by $[C_\ell^{psm}]^{1/2}\,[C_\ell^{ng}]^{-1/2}$, so that the spectrum of the CMB is compatible with the cosmological parameter set used in the simulation. If however the keyword READJUST_NG_SPECTRUM is set to no, then the cosmological parameters set by the PSM user are overwritten to match those used in the non-Gaussian simulation.

Not also that whether or not READJUST_NG_SPECTRUM is set to yes, tensor modes are added to the simulation if the tensor to scalar ratio is non-vanishing.

The CMB_CL_SOURCE parameter (see section 6.2) sets the origin of the CMB power spectrum (or spectra) used to readjust the NG CMB power spectrum when the READJUST_NG_SPECTRUM parameter is set to yes.

8.3.4 DRAW_F_NL

Set DRAW_F_NL to yes to generate the value of $f_{\rm nl}$ at random between F_NL_MIN and F_NL_MAX with a uniform probability distribution. If DRAW_F_NL is set to no, the value of $f_{\rm nl}$ is set with the F_NL parameter.

8.3.5 F_NL_MIN

Set to minimum allowed value of $f_{\rm nl}$ if drawn at random. The default value is -30, but this number can also be positive.

8.3.6 F_NL_MAX

Set to maximum allowed value of $f_{\rm nl}$ if drawn at random. The default value is 30. Note that just as much as F_NL_MIN, this number can be negative if requested.

8.3.7 F_NL

This parameter is used to set $f_{\rm nl}$ to a fixed value (can be positive, negative, or zero).

8.3.8 DRAW_NG_MAP_NUMBER

The PSM uses a number of pre-generated non-Gaussian simulations. If this parameter is set to yes, the map used in the simulations is drawn at random among available maps.

8.3.9 NG_MAP_NUMBER_MIN

The lowest map number used for random selection of the non-Gaussian simulation. Can be fixed between 1 and 1000 for simulations at $l_{\rm max} = 1024$, and between 1 and 100 for simulations at $l_{\rm max} = 3500$.

8.3.10 NG_MAP_NUMBER_MAX

The highest map number used for random selection of the non-Gaussian simulation. Can be fixed between NG_MAP_NUMBER_MIN and 1000 for simulations at $l_{\rm max} = 1024$, and between NG_MAP_NUMBER_MIN and 100 for simulations at $l_{\rm max} = 3500$.

8.3.11 NG_MAP_NUMBER

This parameter is used to set the non-Gaussian map number used, if not drawn at random.

9 SZ effect

The SZ effect, due to the inverse Compton interaction of CMB photons with ionised gaz (primarily in clusters of galaxies) is simulated in the PSM with the superposition of thermal and kinetic SZ effects in a catalogue of galaxy clusters.

The generation of thermal and kinetic SZ effects in the sky model is turned on by setting to `yes` the SZ_INCLUDE_THERMAL and SZ_INCLUDE_KINETIC parameters respectively.

SZ effects are generated in the PSM by two means: either by post-processing large scale N-body and/or hydrodynamical simulations of Large Scale Structure to produce catalogues of clusters and maps of thermal and kinetic SZ effects, or on the basis of a cluster mass function which provides, for a given cosmology (as defined by the cosmological parameters described in section 6), the number density $dN/dMdz$ of clusters of mass M at redshift z.

9.1 SZ: Cluster parameters

A number of parameters are used to convert mass and redshift into integrated Y parameter, or connect X-ray observations to Y. These are listed below.

keyword name	**description / comments**	**accepted values**	**default**
`CLUSTER_PROFILE`	Which type of profile is used to model clusters	`xmm`, `chandra`, `beta`	`xmm`
`NSTD_PROFILE`	Whether to use a non-standard profile	`yes`, `no`	`yes`
`NORM_PROFILE`	Whether the cluster profile is normalised to match the observations	`yes`, `no`	`yes`
`PROFILE_BOUNDS`	Boundary for the profile	`5r500`, `rvir`	`5r500`
`CLUSTER_T_STAR`	Normalisation parameter $T_\star$ to be used if `NORM_PROFILE` is set to `no`	any number	1.48

Table 10: Parameters used to model cluster profiles and normalise them.

9.1.1 `CLUSTER_PROFILE`

This parameter sets the type of profile used to model the cluster.
The three dimensional `beta` profile is given by

$$P(r) = \frac{P_0}{\left[1 + (\frac{r}{r_c})^2\right]^{3\beta/2}} \tag{1}$$

with the core radius r_c depending on the cluster mass and β being fixed to 2/3.
The `xmm` and `chandra` profile are Generalized Navarro Frenk and White profile of the form

$$P(r) = P_{500}\frac{P_0}{(c_{500}x)^\gamma(1 + (c_{500}x)^\alpha)^{\frac{\beta-\gamma}{\alpha}}} \tag{2}$$

where $x = r/R_{500}$, P_{500} an analytical formula depending on the cluster mass and redshift, and α, β, γ being fitted on XMM data (Arnaud, M., Pratt, G. W., Piffaretti, R., et al. 2010, A&A, 517, A92) and Chandra data (Nagai, D., Kravtsov, A. V., & Vikhlinin, A. 2007, ApJ, 668, 1) respectively.

9.1.2 NSTD_PROFILE

When set to `yes`, P_0, c_{500}, γ, α and β for the cluster profile (see eq. (2)) are set to the values of eq. (12) in Arnaud, M., Pratt, G. W., Piffaretti, R., et al. 2010, A&A, 517, A92. If set to `no`, then values are taken from eq. (B.2.) of the same paper. `NSTD_PROFILE` stands for non-standard profile because choosing the values from eq. (12) leads to a non-standard slope of the Y-M relation ($Y \propto M^{1.78}$) while choosing the values from eq. (B.2.) leads to a standard slope ($Y \propto M^{5/3}$). Values from eq. (12) are the best fit on XMM data while values from eq. (B.2.) are obtained when forcing the standard dependance of the scaling laws.

9.1.3 NORM_PROFILE

When set to `yes`, this parameter imposes a normalisation that matches the observations (and this, irrespective of the fact that the cosmological model used in the PSM simulation may be different from the real one). If, in contrast, `NORM_PROFILE` is set to `no`, then the normalisation is made according to a theoretical model, and uses the normalisation parameter `CLUSTER_T_STAR`. The normalisation is computed from the pressure profiles derived from X-ray observations i.e. using the P_0 parameter in the equation 2. If `NORM_PROFILE` is set to `no` then P_0 is ignored. The `NORM_PROFILE` parameter is not active if the `CLUSTER_PROFILE` parameter is set to `beta`.

9.1.4 PROFILE_BOUNDS

Distance from the cluster center at which it is assumed all the cluster mass is included. This also sets the angular distance (from the cluster center) at which the SZ emission of a single cluster will vanish in the SZ maps. The value `5r500` is 5 times the distance at which, in the `xmm` or `chandra` model of the cluster profile, the density of the cluster is 500 times the critical density. The value `rvir` is the virial radius (used for the `beta` model of cluster profile).

9.1.5 CLUSTER_T_STAR

Normalisation parameter to be used if `NORM_PROFILE` is set to `no` or `CLUSTER_PROFILE` is set to `beta`. `CLUSTER_T_STAR` is the value of the T_* parameter in equation 4 of Perpaoli et al., 2003, MNRAS, 342, 163:

$$\left(\frac{M(T,z)}{10^{15}\,h^{-1}\,\mathrm{M}_\odot}\right) = \left(\frac{T}{T_*}\right)^{3/2} \left(\Delta_c E^2\right)^{-1/2} \times \left(1 - 2\frac{\Omega_\Lambda(z)}{\Delta_c}\right)^{-3/2} ,$$

where T is in keV, Δ_c is the mean overdensity inside the virial radius in units of the critical density and E the Hubble parameter normalized to its present value. Under the assumption of clusters being isothermal, we use equation 3 (so the T_* parameter) to normalise equation 1 or 2.

9.2 SZ: Catalogue parameters

A few parameters are used for the generation of a cluster catalogue. They are used by the `dmb`, `hydro+dmb` and `prediction` SZ models, and are listed below.

9.2.1 MASS_FUNCTION

This parameter defines the model to use for the number density $dN/dMdz$ of clusters of mass M at redshift z. The references are:

- Press W. H., Schechter P., 1974, ApJ, 187, 425
- Sheth R. K., Tormen G., 1999, MNRAS, 308, 119
- Evrard A. E., et al., 2002, ApJ, 573, 7
- Jenkins A., et al., 2001, MNRAS, 321, 372
- Tinker J., et al., 2008, ApJ, 688, 709

keyword name	description / comments	accepted values	default
`MASS_FUNCTION`	The mass function used to generate the catalogue	`press_schechter`, `sheth_tormen`, `evrard`, `jenkins`, `tinker`	`tinker`
`CLUSTER_M_INF`	Lower mass limit of clusters included in the catalogue, in units of 10^{15} solar masses	a number, between 0.01 and 1 typically	`0.1`
`SZ_INPUT_CAT`	List of catalogues of known clusters to be included in the model sky emission	a list that can include `rosat`, `sdss`	`rosat sdss`
`SZ_RELATIVISTIC`	Order of relativistic corrections to the thermal SZ effect	0, 1, 2, 3, 4	0

Table 11: Parameters used to produce the cluster catalogue.

9.2.2 `CLUSTER_M_INF`

This parameter sets the lower mass limit of clusters included in the catalogue, in units of 10^{15} solar masses. The number of clusters included increases rapidly with decreasing lower mass. This impacts the time for generating the SZ maps, and the size of the cluster catalogue.

9.2.3 `SZ_INPUT_CAT`

This parameter is used to define the catalogues of known clusters that should be included in the simulation. It is used by the `prediction` model in all cases, as well as in the `dmb` model if the `SZ_CONSTRAINED` parameter of that model is set to `yes`. The `rosat` catalogue comprises 1743 galaxy clusters with the ROSAT X-ray satellite. The `sdss` catalog comprises 13,823 optically selected clusters extracted from the SDSS galaxy survey. 215 clusters are common to these two independent catalogues and their emission is modeled on the basis of the ROSAT observations if both catalogues are used.

9.2.4 `SZ_RELATIVISTIC`

This parameter sets whether relativistic corrections are taken into account in the model of thermal SZ, and at which order. The default is 0 (non-relativistic limit). The relativistic SZ effect is currently fully implemented only for the `prediction` and `dmb` SZ models. In the `hydro+dmb` model, high redshift clusters (which are generated from number counts with the `dmb` model), are modelled with relativistic correction included. The low redshift map, however, is computed at first order only (non-relativistic thermal SZ) for the moment.

9.3 SZ: `prediction`

The SZ `prediction` model includes only expected signals from the clusters included in the catalogues specified with the `SZ_INPUT_CAT` parameter. The parameters `CLUSTER_PROFILE`, `NORM_PROFILE`, `PROFILE_BOUNDS` and `CLUSTER_T_STAR` are active for this model. This model generates only thermal SZ effect.

9.4 SZ: `dmb`

The SZ `dmb` model generates first a catalogue of galaxy clusters according to the mass function specified by the `MASS_FUNCTION` parameter. For each cluster, the expected SZ signal is computed on the basis of a physical model linking mass and redshift to electron density and temperature, on the basis of the spherically symmetric profile specified with the `CLUSTER_PROFILE` parameter. Cluster are distributed at random over the 4π of the sky, with a uniform probability. To each cluster, a velocity is assigned as a function of its redshift (assuming linear growth of structures). The 3-D velocity vector is drawn at random given the variance of the velocity field at that redshift, for the given cosmological parameters. This model accepts two additional parameters: `SZ_CONSTRAINED` and `SZ_INCLUDE_POLARISED`.

keyword name	**description / comments**	**accepted values**	**default**
`SZ_CONSTRAINED`	Whether the catalogue contains real observed clusters	`yes`, `no`	`no`
`SZ_INCLUDE_POLARISED`	Whether to include polarised SZ effect	`yes`, `no`	`no`

Table 12: Parameters used by the `dmb` SZ emission model.

9.4.1 `SZ_CONSTRAINED`

When this parameter is set to `yes`, a catalogue of observed clusters as specified with the `SZ_INPUT_CAT` parameter is produced. Clusters in the simulated catalogue that match best the observed ones (same bin of mass and redshift, location on the sky as close as possible as that of the real cluster) are replaced by the observed ones.

9.4.2 `SZ_INCLUDE_POLARISED`

This parameter is used to generate SZ polarisation due to the transverse motion of the cluster. This feature needs revision (for band integration), and should not be used at present, but can be reactivated if needed. Contact Jacques Delabrouille and/or Jean-Baptiste Melin.

9.5 SZ: `hydro+dmb`

The SZ `hydro+dmb` model merges a low redshift ($z < 0.25$) full hydrodynamic simulation, containing the constrained local SZ map, with a high redshift model based on cluster number counts following the method implemented in the `dmb` model. The high redshift `dmb` part accepts the same keywords as the corresponding model. Note that the catalogue of clusters written by the PSM contains only the high redshift objects (no catalogue is available yet for the low redshift objects present in the hydro simulation).

9.6 SZ: `nbody+hydro`

The SZ `nbody+hydro` model uses a combination of hydro+N-body simulations of the distribution of baryons for redshifts $z < 0.025$ (the local universe), and of pure N-body simulations of dark matter structures in a Hubble volume. The model uses pre-generated SZ maps of thermal and kinetic SZ and uses them as templates of SZ emission. This model requires no additional parameter.

10 The Galaxy

Emission from the galactic interstellar medium is constituted of 5 main components: synchrotron, free-free, thermal dust, spinning dust, and CO molecular lines.

Subdirectories named `synchrotron/`, `freefree/`, `thermaldust/`, `spindust/` and `co/` of the `component/` subdirectory of the PSM output directory are created upon PSM execution, and contain maps and structures describing each of the emissions individually.

There are at present two models of galactic emission, the `prediction` and `simulation` models. Both of them are based on the same input galactic templates, but the `simulation` model generates random small scale structure that is added to the synchrotron, free-free and thermal dust templates if the sky resolution of the PSM run is smaller than the resolution of the available template.

10.1 Galactic polarisation

The polarisation of galactic diffuse emission can be modelled according to two main prescriptions: either following Miville-Deschênes et al. (2008), or Fauvet et al (2011). The models are tightly connected.

Table 13 highlights the parameters that specify the modelling of polarised galactic emission in the PSM.

keyword name	description / comments	accepted values	default
`GAL_POLAR_MODEL`	What model should be used for galactic polarised emission	`mamd2008`, `fauvet2011`	`mamd2008`
`DUST_INTRINSIC_POL`	Intrinsic polarisation fraction of the thermal dust emission	any number between 0 and 1	0.15
`GAL_BFIELD_PITCH_ANGLE`	Pitch angle of the galactic spiral arms	any angle in degrees	-30
`GAL_BFIELD_TURB_AMPL`	Amplitude of the turbulent component of the galactic magnetic field relative to the regular component	any positive number	0.2

Table 13: Parameters used for generating the model of galactic polarisation.

10.1.1 `GAL_POLAR_MODEL`

This parameter sets which model is used to generate the galactic polarised emission. The two possible options are based on a 3-D model of the galactic magnetic field. If `GAL_POLAR_MODEL` is set to `mamd2008`, polarisation templates have been constrained to match WMAP observations, and are fixed, except for the dust intrinsic polarisation level. If it is set to `fauvet2011`, two additional parameters (the galactic pitch angle, and the relative amplitude of the turbulent to regular part of the magnetic field), can be set by the PSM user.

10.1.2 `DUST_INTRINSIC_POL`

Intinsic polarisation fraction of thermal dust emission. This parameter will scale the dust polarisation templates for both the `mamd2008` and the `fauvet2011` galactic polarisation models. The default value corresponds to 15% intrinsic dust polarisation.

10.1.3 `GAL_BFIELD_PITCH_ANGLE`

This parameter is used only by the `fauvet2011` galactic polarisation model. It is used to set the geometry of the regular component of the galactic magnetic field used to produce templates of polarised galactic emission from observations of total intensity.

10.1.4 GAL_BFIELD_TURB_AMPL

This parameter sets the relative strength of the turbulent part of the galactic magnetic field as compared to the regular one (used only by the `fauvet2011` galactic polarisation model).

10.2 Synchrotron

Synchrotron emission is included in the sky model if the `INCLUDE_SYNCHROTRON` parameter is set to `yes`. The synchrotron model in the present PSM is modeled on the basis of a single template at 23 GHz, which is scaled in frequency with a pixel-dependent emission law (either power law, or power law with curvature). A power law synchrotron emission is implemented as

$$I_\nu \propto \left[\frac{\nu}{\nu_{\rm ref}}\right]^{(\beta_s+2)} \tag{3}$$

and a curved power law synchrotron emission as

$$I_\nu \propto \left[\frac{\nu}{\nu_{\rm ref}}\right]^{(\beta_s+2)+\beta_c \log_{10}(\nu/\nu_{\rm cur})} \tag{4}$$

where β_s is the synchrotron spectral index, $\nu_{\rm ref}$ is a reference frequency (for which the synchrotron template is available, currently 23 GHz), β_c is the curvature amplitude, and $\nu_{\rm cur}$ is a reference frequency for the curvature of the emission law.

The synchrotron 23 GHz map is stored in the `synchrotron_ampl.fits` file, and the spectral index map in the `synchrotron_specind.fits` file, both located in the `components/synchrotron/` subdirectory of the PSM output directory. Synchrotron emission is polarised if the `FIELDS` global parameter is set to `TP`.

keyword name	description / comments	accepted values	default
SYNCHROTRON_EMISSION_LAW	What emission law should be used to model synchrotron emission	`powerlaw`, `curvpowerlaw`	`powerlaw`
SYNCHROTRON_INDEX_MODEL	Model used for the variablility of the synchrotron spectral index over the sky	`giardino2002`, `mamd2008`, `uniform`	`mamd2008`
SYNCHROTRON_CURV_FREQ	Reference frequency for synchrotron curvature (if any), in GHz	any positive number (typically between 20 and 100)	23
SYNCHROTRON_CURV_AMPL	Amplitude of the curvature of the synchrotron emission law	any number (typically negative, for steepening)	-0.3

Table 14: Parameters used for generating the synchrotron emission model.

10.2.1 SYNCHROTRON_EMISSION_LAW

If this is set to `powerlaw`, then the synchrotron map template is extrapolated using a power law (that can be pixel-dependent). Set to `curvpowerlaw` for modelling synchrotron with an emission law that steepens at higher frequency.

10.2.2 SYNCHROTRON_INDEX_MODEL

Two different templates, `giardino2002` and `mamd2008`, can be used for modeling a space-varying synchrotron spectral index. The first model is based on Giardino et al., A&A, 387, 82 (2002). The second is based on Miville-Deschênes et al. A&A, 490, 1093 (2008). Finally, if `SYNCHROTRON_INDEX_MODEL` is set to `uniform`, the synchrotron spectral index is assumed to be uniform over the sky, and equal to -3 (in $K_{\rm RJ}$ units).

10.2.3 SYNCHROTRON_CURV_FREQ

Frequency $\nu_{\rm cur}$ (in GHz) for the steepening of the emission law (used only if SYNCHROTRON_EMISSION_LAW is set to curvpowerlaw).

10.2.4 SYNCHROTRON_CURV_AMPL

Amplitude β_c (unitless) for the steepening of the emission law (also used only if SYNCHROTRON_EMISSION_LAW is set to curvpowerlaw).

10.3 Free-free

Free-free emission is included in the sky model if the INCLUDE_FREEFREE parameter is set to `yes`. The free-free model uses a single free-free template, which is scaled in frequency using a specific emission law (close to a power law with spectral index -0.15). The free-free template map at 23 GHz is stored in the `freefree_ampl.fits` output file, located in the `components/freefree/` subdirectory of the PSM output directory. Free-free emission is not polarised in the present model.

The free-free model accepts the following additional parameters:

keyword name	description / comments	accepted values	default
FREEFREE_TEMPLATE	Template free-free map	`dickinson_h_alpha`, `wmap_mem`, `mamd2008`	`mamd2008`
FREEFREE_E_TEMP	Temperature of free-free electrons (in K)	any positive number (typically between 4000 and 10000)	7000

Table 15: Parameters used for generating the free-free emission model.

10.3.1 FREEFREE_TEMPLATE

Set this parameter to `dickinson_h_alpha` to use as a free-free template a map of Hα emission corrected for dust extinnction as derived in Dickinson et al, MNRAS, 341, 369 (2003), to `wmap_mem` to use the WMAP MEM free-free map from Bennett et al., , and to `mamd2008` to use a composite map that uses the former over most of the sky, but uses the latter in regions where the extinction is $E(B-V) \geq 2$ (or $A_V \geq 6$), and uses the WMAP MEM map also when it is lower than the free-free predicted from the Hα emission.

10.3.2 FREEFREE_E_TEMP

The free-free emission law depends slightly on the temperature of the warm medium. Set FREEFREE_E_TEMP to the assumed temperature, in Kelvin (7000 K is the default).

10.4 Thermal dust

Thermal dust emission is included in the sky model if the INCLUDE_THERMALDUST parameter is set to `yes`. It is modeled on the basis of the coaddition of two greybodies with fixed emissivity spectral indices, and with each an amplitude template map and a temperature map.

The amplitude maps are stored in `thermaldust_ampl1.fits` and `thermaldust_ampl2.fits`, and temperature maps are stored in `thermaldust_temp1.fits` and `thermaldust_temp2.fits`. All the above maps are written in the `components/thermaldust/` subdirectory of the PSM output directory. When the FIELDS global parameter is set to TP, the dust amplitude maps are polarised, but the temperature maps are not (the emission laws are assumed the same in temperature and polarisation).

keyword name	description / comments	accepted values	default
I100	Which 100-micron template is used to generate dust emission	SFD, SDFnoHII, FFP6-JD	SDFnoHII

Table 16: Parameters used for generating the thermal dust emission model.

10.4.1 I100

The I100 parameter sets the version of the 100 micron dust template used to generate dust emission. The SFD option corresponds to the Schlegel-Finkebiner-Davies map (in HEALPix format, at native nside=1024). The default option, SDFnoHII, corresponds to the same map with ultra-compact HII regions subtracted (note that the former parameter INCLUDE_H2REGION, used in previous versions, is now obsolete, and is replaced by the use of I100). The third option, FFP6-JD, is a template built from an extrapolation at 100 microns of Planck HFI 857 GHz observations, filtered to suppress cosmic infrared background anisotropies, and with point sources subtracted. That third option is restricted to the Planck collaboration.

10.5 Spinning dust

Spinning dust emission is included in the sky model if the INCLUDE_SPINDUST parameter is set to yes. The spinning dust model uses a single template, which is scaled in frequency using a specific emission law.

The spinning dust template map at 23 GHz is stored in the spindust_ampl.fits output file, located in the components/spindust/ subdirectory of the PSM output directory. Spinning dust emission is not polarised in the present model.

keyword name	description / comments	accepted values	default
SPINDUST_EMISSION_LAW	What emission law should be used to model spinning dust emission	dl98, dl98composition	dl98
SPD_CNM	Proportion of cold neutral medium for spinning dust emission	any number between 0 and 1	0
SPD_WNM	Proportion of warm neutral medium for spinning dust emission	any number between 0 and 1	0.962
SPD_WIM	Proportion of warm ionised medium for spinning dust emission	any number between 0 and 1	0
SPD_MOL	Proportion of molecular clouds for spinning dust emission	any number between 0 and 1	0
SPD_DRK	Proportion of dark gas for spinning dust emission	any number between 0 and 1	0
SPD_RN	Proportion of reflexion nebulae for spinning dust emission	any number between 0 and 1	0.038
SPD_EXTRA	Proportion of extra component for spinning dust emission	any number between 0 and 1	0

Table 17: Parameters used for generating the spinning dust emission model.

10.5.1 SPINDUST_EMISSION_LAW

There are two options for the emission law, which are selected with the SPINDUST_EMISSION_LAW parameter in the PSM configuration file. If this parameter is set to dl98composition, the average composition of the

ISM in terms of cold neutral medium (CNM), warm neutral medium (WNM) warm ionised medium (WIM), molecular clouds (MOL), dark component (DRK), reflection nebulae (RN), are set by the PSM user. The `dl98composition` emission law also accepts an extra component (EXTRA), the emission law of which is tabulated in the `emit4.jnu.extra_a` file, to be found in the `datafiles/spindust` directory of the PSM distribution. The PSM user can change the spinning dust emission law by modifying the corresponding `emit4.jnu.extra_a` data file and changing the proportion of extra emission (recommended only to experimented PSM users). The total spinning dust emission law is the sum of the individual emissions of all components, in proportions set by the parameters described next. The `dl98` spinning dust emission law (default) corresponds to 96.2% warm neutral medium and 3.8% reflection nebulae.

10.6 CO molecular lines

CO molecular line emission is included in the sky model if the `INCLUDE_CO` parameter is set to `yes`. Currently the model is rather simple: one single template, part sky coverage only, constant line ratio, no polarisation. The map used for generating CO emission has only part-sky coverage.

11 Point sources

Point sources in the PSM are separated into three categories: radio sources (`radiops`), infrared sources (`irps`) and ultra-compact HII regions (`uchii`). In addition, WMAP sources are treated as a special case of radio sources.

There are two point sources models implemented in the PSM: `prediction` and `simulation`. As many radio sources are variable, the `prediction` model comprises only infrared sources and ultra-compact HII regions, modelled on the basis of extrapolations of real IRAS sources. The `simulation` model comprises fake (faint) infrared sources to homogenize the IRAS coverage, and extrapolations of radio sources observed at frequencies ranging from 850 MHz to 4.85 GHz.

All PSM sources are divided in two additional categories: strong sources, and faint sources. Strong source observed maps are created directly at the sky resolution by drawing individual sources in pixel space, while faint point source maps are based on distributing the faint sources on single pixels, and then convolving the maps with the appropriate gaussian beam in harmonic space.

11.1 Parameters of the point source model

keyword name	description / comments	accepted values	default
`STRONG_PS_LIMIT_FREQ_GHZ`	Set of frequencies used to separate between strong and faint sources	a list of frequencies (in GHz)	20 1000
`STRONG_PS_LIMIT_FLUX_JY`	Flux limits in Jy, above which sources are considered as strong (must be a list of same size as the list of corresponding frequencies above)	a list of fluxes (in Jy)	0.1 0.5
`INCLUDE_RADIO_SOURCES`	Whether to include radio sources in the sky model	`yes`, `no`	`yes`
`INCLUDE_WMAP_SOURCES`	Whether to include WMAP sources in the sky model	`yes`, `no`	`no`
`INCLUDE_UCHII_SOURCES`	Whether to include ultira compact H-II regions in the sky model	`yes`, `no`	`yes`
`INCLUDE_IR_SOURCES`	Whether to include infrared sources in the sky model	`yes`, `no`	`yes`
`MEAN_IR_POLAR_DEGREE`	Mean degree of polarisation of infrared sources	any number between 0 and 1	0.01

Table 18: Parameters used for generating the point sources in the PSM.

11.1.1 `STRONG_PS_LIMIT_FREQ_GHZ`

Set this parameter to a set of frequencies that will be used to separate between strong and point sources.

11.1.2 `STRONG_PS_LIMIT_FLUX_JY`

Set this parameter to a set of limit fluxes, in Jy. Each of these fluxes corresponds to one of the frequencies set with `STRONG_PS_LIMIT_FREQ_GHZ`, so that the number of specified frequencies and fluxes should be the same.

Any source that has a flux in excess of the limit at any of the specified frequencies is considered as strong, the rest being considered as faint (i.e. those sources that exceed the limit at none of the specified frequencies). Maps and catalogues of observed strong sources will contain the same list of sources for all frequencies of observation. Note that depending on how these parameters are set, there is no guarantee that the sources selected as 'strong' are indeed the strongest ones in all the frequency bands of observation. An extreme example would be to set the limit strong vs. faint at radio frequencies (e.g. 5 GHz) but observing the sky in high frequency bands. The PSM would consider as 'strong' some strong radio sources, but not any of the strong infrared galaxies that are likely to be the strongest in the observations. For safety, it is recommended to have at least one radio frequency and one far infrared frequency in the `STRONG_PS_LIMIT_FREQ_GHZ` list (as is done by default).

11.1.3 `INCLUDE_RADIO_SOURCES`

Set this parameter to include the population of radio sources in the model.

11.1.4 `INCLUDE_WMAP_SOURCES`

Set this parameter to include the population of radio sources in the model. Note that when this is done, a number of sources from the radio catalogue produced from lower frequency data are replaced by sources that match the WMAP measurements. Most of these sources, however, are highly variable, so that an experiment observing the sky some years after WMAP is not likely to observe compatible emission from these sources.

11.1.5 `INCLUDE_UCHII_SOURCES`

Set this parameter to include a population of galactic ultra-compact H-ii regions.

11.1.6 `INCLUDE_IR_SOURCES`

Set this parameter to include a population of infrared sources (based on the IRAS observed sources).

11.1.7 `MEAN_IR_POLAR_DEGREE`

This parameter sets the level of polarisation of infrared sources (for polarised sky simulations).

11.2 Radio sources

Radio sources (`faintradiops` and `strongradiops` components) in the PSM are modelled as pointlike objects in the sky, with a Spectral Energy Distribution (SED) that depends on frequency as a set of band-limited power laws. Each radio source is modelled with four distinct power laws, that describe their emission below 4.85 GHz, between 4.85 and 20, between 20 and 100, and above 100 GHz. Each source has its own amplitude and spectral indices. Each source has its own polarisation fraction and angle, but both are constant (for each source) across frequencies. The model catalogues of radio point sources is stored in the format of an IDL save file in the `component/ps/` subdirectory of the PSM output directory. There is, in general, a catalogue for faint radio sources, and another one for strong radio sources.

The total number of modelled radio sources in the PSM is about 2,000,000.

11.3 Infrared sources

Infrared sources (`faintirps` and `strongirps` components) are modelled as pointlike objects, with an SED in the form of a single greybody each. Infrared sources are mostly galactic sources and local galaxies. Catalogues for strong and faint infrared sources are stored as IDL save sets in the `component/ps/` subdirectory of the PSM output directory.

11.4 WMAP sources

WMAP sources are considered as radiosources, and treated as such except that they are modelled with an emission al below 4.85 GHz, one between 4.85 and 23, one between each of the central frequencies of the WMAP frequency bands (23, 33, 41, 61, and 94 GHz), and one above 94 GHz. If the parameter `INCLUDE_WMAP_SOURCES` is set to `yes`, this replaces the modeling of some of the radio sources above.

11.5 UCH-II sources

Ulracompact H-II regions are modelled with the sum of two emission laws: a greybody for the thermal emission part, and a free-free emission at radio frequencies.

12 The Far Infrared Background

There is at present one single model of emission for the far infrared background, due to a collection of blended high redshift infrared sources. No parameter exists at present for this component.

13 Band-integration and simulated observations

Once a model of the sky is generated, the PSM performs band-integration of the emissions to generate band-integrated maps of components, at the resolution of the generated model sky. These-band-integrated sky emission maps are then used to generate simulated observations by instruments (with noise added, resolution changed, and possibly map format changed).

13.1 General parameters of sky observation

The general parameters specific for band-integration only are as follows:

keyword name	description / comments	accepted values	default
`STRONG_SOURCES_TO_CAT`	Whether a catalogue of strong point sources with fluxes integrated in the band is produced	`yes`, `no`	`yes`
`STRONG_SOURCES_TO_MAP`	Whether a map of strong point sources with fluxes integrated in the band is produced	`yes`, `no`	`no`
`GROUP_GALAXY`	Whether to co-add all diffuse galactic components into a single maps of galactic emission	`yes`, `no`	`yes`
`GROUP_FAINT_PS`	Whether to co-add all faint point sources (radio, infrared) and the FIRB into a single map of point source background	`yes`, `no`	`yes`
`GROUP_STRONG_PS`	Whether to co-add all strong point sources (radio, infrared, ultra-compact HII) into a single map of strong sources	`yes`, `no`	`yes`

Table 19: Parameters specifying the rules for band-integration.

Observation parameters, that impact the production of coadded maps as seen by the instruments, are:

keyword name	description / comments	accepted values	default
`OBS_TASK`	Specifies what observation is performed	`new`, `update`, `none`	`none`
`WHAT_OBS`	Whether to do only band integration, or full observation	`bandinteg` `fullobs`	`bandinteg`
`OBS_RES`	Whether the observations are smoothed or deconvolved to put them either at sky resolution, or at the resolution of the instrument	`instr`, `sky`	`sky`
`OBS_COADD`	Which coadded map(s) to produce in the `observations` subdirectory	see section 13.2	none

Table 20: Parameters specifying the rules for sky observation.

13.1.1 `STRONG_SOURCES_TO_CAT`

Strong sources, as selected on the basis of the values of the `STRONG_PS_LIMIT_FREQ_GHZ` and `STRONG_PS_LIMIT_FLUX_JY` parameters described in section 11.1, can be 'observed' in the format of a catalogue of observed sources. Set `STRONG_SOURCES_TO_CAT` to `yes` to produce, for each instrument channel, a catalogue of strong point source observations.

13.1.2 STRONG_SOURCES_TO_MAP

Set this parameter to `yes` to produce maps of strong point sources for each of the instrument channels.

13.1.3 GROUP_GALAXY

For the purpose of saving disk space, it is possible to avoid writing on disk the maps for individual galactic components (synchrotron, free-free, thermal dust, spinning dust, CO lines). If
tt GROUP_GALAXY is set to `'yes'`, then all these maps are co-added for each frequency band, and are are saved in a single file (per frequency band).

13.1.4 GROUP_FAINT_PS

This parameter is similar to
tt GROUP_GALAXY, except that it co-adds all faint source maps (including the far infrared background).

13.1.5 GROUP_STRONG_PS

This parameter is similar to
tt GROUP_GALAXY, except that it co-adds all strong source maps (including the far infrared background).

13.1.6 OBS_TASK

In addition to the default option of not observing the sky (i.e. doing no integration of the sky model into instrumental frequency bands) the PSM offers the possibility to generate a new integrated sky emission, or to update an existing one. This is set using the `OBS_TASK` parameter. If it is set to `update`, then the code checks for existing band-integrated sky maps in the relevant directories of the PSM output, and checks whether the current instrumental band is the same as the one stored. If the instrumental band is the same, and the band-integrated sky file already exists, then the band integration is not redone. Otherwise, if either the sky or the band have changed, then the band-integration is re-done, and the new band-integrated sky emission is saved in place of any already existing band-integrated sky emission. When instead the `OBS_TASK` parameter is set to `new`, then all existing observations are erased, and re-done.

The `update` mode is particularly useful for generating new observations of an existing sky, in which case it would be used with the `SKY_TASK` parameter set to `restore` (see section 5.1).

13.1.7 WHAT_OBS

The 'observation' of the PSM model sky is performed in two steps:

1. First, integration in frequency bands (at the resolution of the sky model). This is the `bandinteg` step;
2. Then, smoothing (if required) to the resolution of instrumental channels (or deconvolving, if the resolution of the instrument is better than that of the sky model, although this is not particularly recommended), coaddition, generation and addition of instrumental noise, and reprojection in the pixelisation schem of each instrumental channel. This is the `fullobs` step.

Set `WHAT_OBS` to `bandinteg` to stop at the end of step 1, and to `fullobs` to stop at the end of step 2.

13.1.8 OBS_RES

This parameter offers the possibility to produced final observations at the resolution of the sky model rather than that of the instrument. It is useful to generate maps at degraded resolution.

13.2 Coaddition rules

The PSM offers flexibility in the production of co-added or partially coadded output maps. Rules for coaddition are defined using the special parameter `OBS_COADD`. Unlike most parameters defined in the PSM configuration file, there can be several instances of `OBS_COADD` in the instructions.[1]

Each one of them will then be used to generate one single map of observation, containing the emission of one ore more sky components and/or of instrumental noise. The maps will be found in the `observations` subdirectory of the output, in subdirectories corresponding to individual instrument channels. Map names will start with `group1_map_...`, `group2_map_...`, etc. The list of components included in each map is written in the README file included in each channel subdirectory. For instance, consider the following lines in the PSM configuration file:

```
OBS_COADD = allsky
OBS_COADD = all
OBS_COADD = synchrotron freefree thermaldust spindust co
OBS_COADD = synchrotron freefree
OBS_COADD = faintps, strongps
OBS_COADD = noise
```

These five lines specify that the PSM should produce 5 maps of observed emission for each detector of each instrument: For each detector the first map, saved in the file named `group1_map_....fits`, will be the coaddition of all sky emission. The second map will be the coaddition of all sky emission and instrumental noise, the third the coaddition of the specified galactic components, etc.

Note that coaddition rules will look for the specified sky components in the `skyinbands` directory. If the maps are not present, they will not be coadded. For instance, if the parameter `GROUP_GALAXY` has been set to `yes`, individual band-integrated maps do not exist for synchrotron, free-free, etc. Instead, there exists a single map of galactic emission. It is not possible anymore to make a coadded map, of synchrotron and free-free only, and the coaddition of all galactic components should be specified by:

```
OBS_COADD = galaxy
```
instead of:
```
OBS_COADD = synchrotron freefree thermaldust spindust co
```

[1]The other exception is the `INSTRUMENT` parameter, see section 14.

14 Instruments

The PSM uses, for observing the simulated sky, simple models of a few relevant instruments. Each instrument is described on the basis of a number of channels or detectors, with each a specific beam, polarisation sensitivity, frequency band, (simplified) noise properties, and pixelisation scheme. Specific instruments implemented in the current version comprise a few different versions of the Planck LFI and HFI, of WMAP, and of IRAS (lowest two frequency channels). In addition, the software implements a generic, simple instrument called `PSM_IDEAL`, which permits the user to define a simplified instrument model.

Instruments used for band-integrating and observing sky emission in the PSM are specified in the PSM configuration file by lines such as:

```
INSTRUMENT = PSM_IDEAL
INSTRUMENT = LFI_BLUEBOOK
INSTRUMENT = HFI_RIMO
INSTRUMENT = WMAP
```

These keywords specify that in both the `skyinbands` and `observations` directories, if `OBS_TASK` is either `new` or `update` a subdirectory corresponding to each of these instruments will will contain the catalogues and/or maps of emission after band-integration (if `WHAT_OBS` is equal to `bandinteg` or `fullobs`), and as observed by the detectors of the corresponding instrument (if `WHAT_OBS` is equal to `fullobs`).

14.1 The `PSM_IDEAL` instrument

`PSM_IDEAL` is the easiest, simplest, and most flexible instrument implemented in the PSM. It is fully described by a set of frequencies, and corresponding beams, polarisation properties, and noise levels. Maps are always produced in the same pixelisation as the sky model itself, and frequency bands are infinitely thin.

keyword name	**description / comments**	**accepted values**	**default**
`OBS_FREQUENCIES`	A list of frequencies, in GHz	Any list of numbers	No default value
`OBS_RESOLUTION`	A list of beam sizes in arcminutes	Any list of numbers	No default value
`OBS_STOKES`	Specifies whether the observations in that band are polarised, or not	`T TQU`	`T`
`OBS_UNITS`	Units of the maps – See section 17.2 for details about PSM units	any PSM brightness unit `psmunit`	`mK_RJ`
`PSM_IDEAL_NOISE`	Noise for the `PSM_IDEAL` instrument	`nominal none`	`none`
`T_NOISE_LEVEL`	Noise level for temperature observations	Any list of numbers	0
`P_NOISE_LEVEL`	Noise level for polarisation observations	Any list of numbers	0
`NOISE_UNITS`	Units for the noise, per square degree – See section 17.2 for details about PSM units	String of the form `psmunit/deg`	`uK_RJ/deg`

Table 21: Parameters specifying the `PSM_IDEAL` instrument.

14.2 Specific instruments

Specific instruments implemented in the PSM are models of the Planck LFI, the Planck HFI, WMAP and IRAS. For all versions of any of these instruments, a single parameter list specifies the observation units for all the channels, and a single parameter list specifies whether the pixelisation for the observations is specific to the instrument, or the same as the pixelisation of the sky. These parameters are listed in Tables 22 and 23 respectively.

keyword name	**description / comments**	**accepted values**	**default**
`HFI_UNITS`	Units for all HFI instruments	A list of 6 `psmunits`	`K_CMB`
`LFI_UNITS`	Units for all LFI instruments	A list of 3 `psmunits`	`K_CMB`
`WMAP_UNITS`	Units for the WMAP instrument	A list of 5 `psmunits`	`mK_CMB`
`IRAS_UNITS`	Units for all IRAS instruments	A list of 6 `psmunits`	`MJy/sr`

Table 22: Parameters specifying the units for the various specific PSM instruments – See section 17.2 for details about PSM units.

keyword name	**description / comments**	**accepted values**	**default**
`HFI_PIX`	Pixelisation for all HFI instruments	`instr`, `sky`	`sky`
`LFI_PIX`	Pixelisation for all LFI instruments	`instr`, `sky`	`sky`
`WMAP_PIX`	Pixelisation for the WMAP instrument	`instr`, `sky`	`sky`
`IRAS_PIX`	Pixelisation for all IRAS instruments	`instr`, `sky`	`sky`

Table 23: Parameters specifying the pixelisation for the various specific PSM instruments. The `sky` option corresponds to maps in the same pixelisation as sky maps, specified with the parameters described in section 5.1.

The list of currently implemented specific instruments is:

For the Planck HFI: `HFI_IDEAL`, `HFI_BLUEBOOK`, `HFI_RIMO`
For the Planck LFI: `LFI_IDEAL`, `LFI_BLUEBOOK`, `LFI_RIMO`
For WMAP: `WMAP`
For IRAS: `IRAS_IDEAL`, `IRAS_RIMO`

They are described in more detail in the next sections.

14.2.1 `HFI_IDEAL`

The `HFI_IDEAL` instrument comprises the 6 HFI channels, with 6 monofrequency bands at 100, 143, 217, 353, 545 and 857 GHz. The resolution of each channel is that of the Planck Blue Book. Noise for this instrument is generated if the `HFI_IDEAL_NOISE` is set to `nominal` [default value is `none`, i.e. no noise]. The noise is uncorrelated and uniform over the whole sky. The noise level is taken from the Planck blue book, but can be scaled from the original nominal mission duration of 14 months using the `HFI_IDEAL_DURATION` keyword (in months).

Table 24 gives the main characteristics of the HFI channels. The last three columns give the multiplicative coefficients that permit to change the units of a map from y_{SZ} to $\mathrm{K_{CMB}}$ (SZ Compton parameter to thermodynamic temperature in Kelvin), from $\mathrm{K_{RJ}}$ to $\mathrm{K_{CMB}}$ (antenna temperature to thermodynamic temperature), and MJy/sr to $\mathrm{K_{CMB}}$. The numbers given here are obtained for single-precision integration, they may vary slightly for double precision simulations (typically by a fraction of a per cent). See section 17.2 for important precisions about the units used in the PSM.

channel	FWHM	YSZ2KCMB	KRJ2KCMB	MJYSR2KCMB
100GHz	10	-4.1091199	1.2867296	0.0041880799
143GHz	7.1	-2.8341320	1.6539019	0.0026324815
217GHz	5	-0.019536398	2.9923766	0.0020683517
353GHz	5	6.1090741	12.915154	0.0033734774
545GHz	5	15.259461	159.98662	0.017531469
857GHz	5	30.229624	15753.256	0.69813007

Table 24: `HFI_IDEAL` instrument: characteristics

14.2.2 `LFI_IDEAL`

The `LFI_IDEAL` instrument is very similar in spirit to `HFI_IDEAL`, except that it uses the `LFI_IDEAL_NOISE` and `LFI_IDEAL_DURATION` parameters instead. Table 27 gives the main characteristics of the LFI channels.

channel	FWHM	YSZ2KCMB	KRJ2KCMB	MJYSR2KCMB
30GHz	33	-5.3238063	1.0234751	0.037013687
44GHz	24	-5.1799698	1.0510483	0.017670341
70GHz	14	-4.7766109	1.1332425	0.0075275633

Table 25: `LFI_IDEAL` instrument: characteristics

14.2.3 HFI_BLUEBOOK

The HFI_BLUEBOOK instrument differs from HFI_IDEAL only through the shape of the frequency bands, which are square instead of monofrequency. The HFI_BLUEBOOK_NOISE and HFI_BLUEBOOK_DURATION parameters are used to specify the noise properties, in the same way as for the HFI_IDEAL instrument.

channel	FWHM	YSZ2KCMB	KRJ2KCMB	MJYSR2KCMB
100GHz	10	-4.0649414	1.2993981	0.0041912780
143GHz	7.1	-2.7749512	1.6818202	0.0026528442
217GHz	5	-0.0014196425	3.0654824	0.0020998274
353GHz	5	5.7909145	12.972937	0.0033580961
545GHz	5	14.024523	143.55841	0.015589777
857GHz	5	26.833689	9832.7070	0.43183285

Table 26: HFI_BLUEBOOK instrument: characteristics

14.2.4 LFI_BLUEBOOK

The LFI_BLUEBOOK instrument differs from HFI_IDEAL only through the shape of the frequency bands, which are square instead of monofrequency. The LFI_BLUEBOOK_NOISE and LFI_BLUEBOOK_DURATION parameters are used to specify the noise properties, in the same way as for the HFI_IDEAL instrument.

channel	FWHM	YSZ2KCMB	KRJ2KCMB	MJYSR2KCMB
30GHz	33	-5.3217640	1.0238649	0.036904771
44GHz	24	-5.1757340	1.0518942	0.017625811
70GHz	14	-4.7669721	1.1354356	0.0075170747

Table 27: LFI_BLUEBOOK instrument: characteristics

14.2.5 HFI_RIMO

14.2.6 LFI_RIMO

14.2.7 WMAP

14.2.8 IRAS_IDEAL

14.2.9 IRAS_RIMO

15 Description of the PSM outputs

15.1 The PSM output directory

All the products of a PSM run are organised in a hierarchy of subdirectories of the PSM output directory. A complete PSM output directory comprises the following directory structure:

```
OUTPUT_DIRECTORY/ --> .psm/
                  --> ancillary/
                  --> components/   --> cmb/
                                    --> co/
                                    --> dipole/
                                    --> freefree/
                                    --> ps
                                    --> spindust
                                    --> synchrotron
                                    --> sz
                                    --> thermaldust

                  --> cosmo/        --> camb/
                                    --> class/
                                    --> standard/
                  --> figures/
                  --> observations/ --> HFI_BLUEBOOK/
                                    --> HFI_IDEAL/
                                    --> HFI_RIMO/     --> detector_100_1a/
                                                      --> detector_100_1b/
                                                      --> ...
                                    --> IRAS_IDEAL/
                                    --> IRAS_RIMO/
                                    --> LFI_BLUEBOOK/
                                    --> LFI_IDEAL/
                                    --> LFI_RIMO/
                                    --> WMAP/         --> K/
                                                      --> Ka/
                                                      --> ...
                  --> psminfo/
                  --> skyinbands/   --> HFI_BLUEBOOK/
                                    --> HFI_IDEAL/
                                    --> HFI_RIMO/
                                    --> IRAS_IDEAL/
                                    --> IRAS_RIMO/
                                    --> ...
```

The main subdirectories are briefly described above (section). The most important directories from the user's point of view are the `components`, `cosmo`, `observations`, `psminfo`, and `skyinbands` directories. The `.psm` directory contains informations private to the PSM run, that are used in consecutive runs of the PSM on that output directory. Some of this is obsolete. The `ancillary` directory contains ancillary data produced during the PSM run (not used much for the moment). The `figures` directory contains some figures produced automatically by the PSM, but this will probably change in the near future.

15.1.1 The `.psm` directory

The `.psm` directory contains information that is private to the PSM code, and in principle is not useful to the PSM user (either duplicates information stored elsewhere, or is only useful for technical aspects of software implementation).

15.1.2 The `ancillary` directory

The `ancillary` directory contains ancillary data generated during the PSM run if the `WRITE_ANCILLARY` parameter is set to `yes` in the PSM configuration file. Such ancillary data is meant to mimic existing observables currently available, which could be used as ancillary data for analysing PSM outputs. They are simulations compatible with the model sky generated during the PSM run. This feature of the PSM is not fully operational at present (only limited ancillary data is generated, if any).

15.1.3 The `components` directory

The `components` directory contains all the information concerning the model sky: parameters, maps, catalogues. The content of this directory is completely independent of the instrument (or set of instruments) ultimately used to 'observe' the sky. This directory is itself organised in several sub-directories (one per component). A description of the component outputs can be found in section 15.2.

15.1.4 The `cosmo` directory

The `cosmo` directory contains the inputs and outputs of `CAMB` and `CLASS` runs, used by the PSM for generation CMB maps (and matter power spectra for upcoming versions of the code). It contains also files for the present default CMB power spectrum C_ℓ, for the current best-fit concordance cosmological model.

15.1.5 The `figures` directory

The `figures` directory contains any figures produced during the process of the PSM run. The production of these outputs are activated by setting the `VISU` parameter in the PSM configuration file to any non-zero integer, and setting the `OUTPUT_VISU` parameter to `png` or `ps`.

15.1.6 The `psminfo` directory

The `psminfo` directory contains information relative to the PSM run: duplicates of any configuration file used to produce the simulations present in the output directory, a file giving the bibliography relevant to the modelled sky and its observation, log files giving the details of the PSM run. The configuration files stored in this directory is directly reusable as input configuration files of the PSM to recompute the same outputs (see configuration file description in section 4).

15.1.7 The `observations` directory

The `observations` directory contains simulated, noisy observations of the PSM sky with an instrument, or a set of instruments. Each observation is stored in a format and in units which are specific to each channel of the instrument, and are set in the PSM configuration file.

15.1.8 The `skyinbands` directory

The `skyinbands` directory contains maps of sky diffuse components and/or catalogue of point sources as seen after integration of their emission in instrumental frequency bands.

15.2 Sky model

The first step in a PSM run is the creation of the model sky. Each component is represented either using maps or catalogues of emission laws. The parameters of these emission laws are stored in the `components` subdirectory of the PSM output directory, in a specific subdirectory for each component. Outputs relevant to the cosmological model are written in the `cosmo` subdirectory of the PSM output directory.

For each component, an IDL save-set is written in the corresponding component subdirectory (e.g. for the CMB, a file `cmb.sav`). This save-set contains meta-information about the component, that is used in the following steps to produce maps of band-integrated emission. This meta-information is saved in the format of a structure that contains, in particular, information about the number of emission laws used to model the component, and about the type and parameter values (for instance, pointers to $a_{\ell m}$ or map files) of each such emission law.

15.2.1 The CMB component

The CMB component is saved in the `components/cmb` subdirectory of the PSM output directory. The structure produced and used by the PSM code is saved in the `cmb.sav` IDL save set. The structure is also printed out in the `cmb.txt` text file for easy checking by PSM users. A typical CMB structure print-out (obtained using the `PRTSTRUCT` procedure) is:

```
CMB | .NAME       |         |  --------->     STRING    = 'cmb'
CMB | .TYPE       |         |  --------->     STRING    = 'comp'
CMB | .ID         |         |  --------->     STRING    = 'MmFsg5TZju6WVbZV'
CMB | .CLASS      |         |  --------->     STRING    = 'diffuse'
CMB | .POLARISED  |         |  --------->     BYTE      =    1
CMB | .NLAW       |         |  --------->     INT       =        1
CMB | .E1         | .LAW    |  --------->     STRING    = 'cmb'
    |             | .NUMIN  | .VALUE --->    <PtrHeapVar550> FLOAT     =       0.00000
    |             |         | .UNIT ---->     STRING    = 'Hz'
    |             |         | .INFO ---->     STRING    = 'min. freq. range value'
    |             | .NUMAX  | .VALUE --->    <PtrHeapVar551> FLOAT     =           Inf
    |             |         | .UNIT ---->     STRING    = 'Hz'
    |             |         | .INFO ---->     STRING    = 'max. freq. range value'
    |             | .AMPL   | .VALUE --->    <PtrHeapVar552> STRING    = 'cmb_map.fits'
    |             |         | .UNIT ---->     STRING    = 'uK_CMB'
    |             |         | .INFO ---->     STRING    = 'flux at ref. freq.'
    |             | .NUREF  | .VALUE --->    <PtrHeapVar553> UNDEFINED = <Undefined>
    |             |         | .UNIT ---->     STRING    = 'Hz'
    |             |         | .INFO ---->     STRING    = 'reference frequency'
```

The structure shows that the CMB emission is modelled using one single emission law `cmb` (see section 15.2.4, and stores the name of the file in which the CMB map is written (note that only the base name of the file is stored, rather than the full file name including the path). The emission law is valid over the full frequency range (from 0 to infinity), the map is in μK thermodynamic (redundant information, as this is also written in the header of the fits file). The reference frequency for the emission law `nuref` is not needed here (and is not defined in this specific example).

15.2.2 The CMB dipole

The dipole is saved in the `components/dipole` subdirectory of the PSM output directory. The dipole structure is similar to that of the CMB.

15.2.3 CO emission lines

The CO emission is modelled using one map of emission intensity for each one of the (J=1-0), (J=2-1) and (J=3-2) transitions. A typical CO line emission structure print-out is as follows:

```
CO | .NAME       |         |  --------->     STRING    = 'co'
CO | .TYPE       |         |  --------->     STRING    = 'comp'
CO | .ID         |         |  --------->     STRING    = '2cKZ9hv53SasHb8L'
CO | .CLASS      |         |  --------->     STRING    = 'diffuse'
CO | .POLARISED  |         |  --------->     BYTE      =    0
CO | .NLAW       |         |  --------->     INT       =        3
CO | .E1         | .LAW    |  --------->     STRING    = 'dirac'
   |             | .NUMIN  | .VALUE --->    <PtrHeapVar140> DOUBLE    =    1.1520000e+11
   |             |         | .UNIT ---->     STRING    = 'Hz'
   |             |         | .INFO ---->     STRING    = 'min. freq. range value'
   |             | .NUMAX  | .VALUE --->    <PtrHeapVar141> DOUBLE    =    1.1530000e+11
   |             |         | .UNIT ---->     STRING    = 'Hz'
   |             |         | .INFO ---->     STRING    = 'max. freq. range value'
   |             | .AMPL   | .VALUE --->    <PtrHeapVar142> STRING    = 'co_ampl10.fits'
   |             |         | .UNIT ---->     STRING    = 'MJy/sr'
   |             |         | .INFO ---->     STRING    = 'flux at ref. freq.'
   |             | .NUREF  | .VALUE --->    <PtrHeapVar143> DOUBLE    =    1.1527100e+11
   |             |         | .UNIT ---->     STRING    = 'Hz'
   |             |         | .INFO ---->     STRING    = 'reference frequency'
CO | .E2         | .LAW    |  --------->     STRING    = 'dirac'
   |             | .NUMIN  | .VALUE --->    <PtrHeapVar144> DOUBLE    =    2.3050000e+11
   |             |         | .UNIT ---->     STRING    = 'Hz'
   |             |         | .INFO ---->     STRING    = 'min. freq. range value'
   |             | .NUMAX  | .VALUE --->    <PtrHeapVar145> DOUBLE    =    2.3060000e+11
   |             |         | .UNIT ---->     STRING    = 'Hz'
   |             |         | .INFO ---->     STRING    = 'max. freq. range value'
   |             | .AMPL   | .VALUE --->    <PtrHeapVar146> STRING    = 'co_ampl21.fits'
   |             |         | .UNIT ---->     STRING    = 'MJy/sr'
   |             |         | .INFO ---->     STRING    = 'flux at ref. freq.'
   |             | .NUREF  | .VALUE --->    <PtrHeapVar147> DOUBLE    =    2.3053800e+11
   |             |         | .UNIT ---->     STRING    = 'Hz'
   |             |         | .INFO ---->     STRING    = 'reference frequency'
CO | .E3         | .LAW    |  --------->     STRING    = 'dirac'
   |             | .NUMIN  | .VALUE --->    <PtrHeapVar148> DOUBLE    =    3.4570000e+11
   |             |         | .UNIT ---->     STRING    = 'Hz'
   |             |         | .INFO ---->     STRING    = 'min. freq. range value'
   |             | .NUMAX  | .VALUE --->    <PtrHeapVar149> DOUBLE    =    3.4590000e+11
   |             |         | .UNIT ---->     STRING    = 'Hz'
   |             |         | .INFO ---->     STRING    = 'max. freq. range value'
   |             | .AMPL   | .VALUE --->    <PtrHeapVar150> STRING    = 'co_ampl32.fits'
   |             |         | .UNIT ---->     STRING    = 'MJy/sr'
   |             |         | .INFO ---->     STRING    = 'flux at ref. freq.'
   |             | .NUREF  | .VALUE --->    <PtrHeapVar151> DOUBLE    =    3.4579600e+11
   |             |         | .UNIT ---->     STRING    = 'Hz'
   |             |         | .INFO ---->     STRING    = 'reference frequency'
```

15.2.4 Emission laws

PSM emission laws are parametric functions $F(\nu;\Theta)$, where ν is frequency and Θ a set of parameters. Implemented emission laws are listed in Table 28.

emission law	description / comments	parameters
`cmb`	CMB anisotropy emission law (derivative of blackbody w.r.t. temperature, at $T = T_{\rm CMB}$)	none
`blackbody`	Black body: $F(\nu; T) \propto B_\nu(T)$	temperature T
`greybody`	Modified black body: $F(\nu; \alpha, T) \propto \nu^\alpha B_\nu(T)$	spectral index α temperature T
`powerlaw`	Power law: $F(\nu; \alpha) \propto \nu^\alpha$	spectral index α
`curvpowerlaw`	Curved power law: $F(\nu; \alpha, a_c, \nu_c) \propto \nu^{\alpha + a_c \log_{10}(\nu/\nu_c)}$	spectral index α curv. amplitude a_c curv. ref. freq. ν_c
`freefree`	Free-free emission law	electron temp. T_e
`spindust`	Spinning dust emission law	see section 10.5
`thermalsz`	Thermal SZ emission law. The temperature parameter is optional. If present it is used to compute relativistic corrections, up to order 4 in T_e/mc^2	electron temp. T_e
`relatsz1`	Spectral dependence of the first order relativistic correction to thermal SZ	none
`relatsz2`	Spectral dependence of the second order relativistic correction to thermal SZ	none
`relatsz3`	Spectral dependence of the third order relativistic correction to thermal SZ	none
`relatsz4`	Spectral dependence of the fourth order relativistic correction to thermal SZ	none
`dirac`	Emission line (infinitely thin): $F(\nu; \nu_{\rm ref}) \propto \delta(\nu - \nu_{\rm ref})$	reference freq. $\nu_{\rm ref}$

Table 28: PSM emission laws.

15.3 Band-integrated sky emission

After generation of the model sky, that stores all parameters of all emission components, the PSM integrates those components in frequency bands specified by a list of instruments. Band-integrated components are stored in the `skyinbands` directory. For each instrument, there is a sub-directory named after the instrument, which contains maps and catalogues obtained after band-integration.

15.4 Observed sky emission

16 PSM headers for `fits` files

Significant effort is made to include, in the headers of all fits files produced by the PSM, the relevant information about the data stored. For this purpose, headers of fits files written by the PSM comprise a section that is specific to the PSM, and contains most of the information useful for describing what is in the data set. Software for manipulating PSM headers can be found in the `psm/fitshdr/` subdirectory of the PSM software distribution.

PSM headers are written between two standard delimiting lines, between which PSM header blocks carry each a part of the information, connected to one particular feature of the data. Base headers blocks store information about the PSM run that generated the data, and the data type and format. Specific headers blocks exist for alm, bands, beams, cl, components, maps, observations. Some are exclusive, i.e. map header blocks are specific to maps, alm header blocks to alm data, and cl header blocks to cl data. Component headers blocks are for component files (i.e. files that are part of a model of a component) while observation headers blocks are for observations (in a frequency band, or at a frequency). For example, a PSM header of an observed map contains a base header, an observation header, a map header, a beam header, a band header, as in the following example:

```
COMMENT  *** PSM header *******************************************************
COMMENT  --- PSM base header --------------------------------------------------
PSM_VERS= 'head    '           /PSM version used to create the data
PSM_RNID= 'BzkMjSHlWHyWlFzR'   /ID key of PSM run which produced the data
PSM_DTTP= 'OBS     '           /PSM data type ('COMP' or 'OBS')
PSM_DTFM= 'MAP     '           /PSM data format ('MAP', 'ALM', 'CL' or 'CAT')
PSM_DTID= '        '           /ID key of PSM data
PSM_FLID= 'PCcOEejcpXpXWSoL'   /ID key of this file
COMMENT  --- PSM observation header -------------------------------------------
PSM_OBID= 'rWfTLUpiDe6RJUEY'   /PSM observation ID key
CMIX1   = 'synchrotron'        /Component included in observation
CMIX2   = 'freefree'           /Component included in observation
CMIX3   = 'thermaldust'        /Component included in observation
CMIX4   = 'spindust'           /Component included in observation
CMIX5   = 'co      '           /Component included in observation
COMMENT  --- PSM map header ---------------------------------------------------
PSM_PXTP= 'HEALPIX '           /Pixelisation type
PSM_LMAX=                  200 /Maximum multipole number
PXWIN   =                    0 /Pixel window function
COMMENT  --- PSM beam header --------------------------------------------------
BEAMTYPE= 'GAUSSIAN'           /
BEAMSIZE=              180.000 /Beam size in arcminutes
COMMENT  --- PSM band header --------------------------------------------------
BD_SHP  = 'INSTR   '           /Band shape (e.g. 'DIRAC', 'TOPHAT', 'INSTR')
BD_INSTR= 'HFI_RIMO'           /Instrument for the specified band
BD_VERS = 'FFP4_bandpass-only' /Version for the specified band
BD_CHAN = '100_1a  '           /Channel for the specified band
COMMENT  *** End of PSM header ************************************************
```

16.1 PSM base header

A PSM base header block looks typically as follows:

```
COMMENT  --- PSM base header ----------------------------------------------------
PSM_VERS= 'head    '            /PSM version used to create the data
PSM_RNID= 'cULvKukObFipESwb'   /ID key of PSM run which produced the data
PSM_DTTP= 'COMP    '            /PSM data type ('COMP' or 'OBS')
PSM_DTFM= 'MAP     '            /PSM data format ('MAP', 'ALM', 'CL' or 'CAT')
PSM_DTID= 'pfF8cQoSlpogDcCS'   /ID key of PSM data
PSM_FLID= 'yjYAl7A8Arf4hMba'   /ID key of this file
```

The PSM base header comprises 6 keywords.

- PSM_VERS is the version of the PSM code. In the kinetic SZ example above, the head version of the PSM CVS repository was used, i.e. it is not a tagged and released version. Data generated with the 1.7.4 release will be tagged with PSM_VERS = '1_7_4 '.
- PSM_RNID is the identification key for the PSM run. All the data produced by the same run will share the same key, which will be written in the PSM headers of the fits files.
- PSM_DTTP is the data type, which can be 'COMP' for a component, and 'OBS' for an observation.
- PSM_DTFM is the data format, which can be 'MAP' for a map, 'ALM' for spherical harmonics, 'CL' for a (multivariate) power spectrum, and 'CAT' for a catalogue of objects.
- Disregard for the moment the other ID keys, 'PSM_DTID' and 'PSM_FLID'. They are meant to tag the data object (for data objects that have several files associated to them) and the file itself (for cross reference), but they are not handled consistently by the PSM yet.

16.2 PSM component header

A PSM component header block looks typically as follows:

```
COMMENT  --- PSM component header -----------------------------------------------
PSM_CPNM= 'kineticsz'          /PSM component name ('cmb', 'synchrotron', ...)
PSM_CPID= 'pfF8cQoSlpogDcCS'   /PSM component ID key
```

The PSM component header comprises two keywords. PSM_CPNM is the name of the component. Valid component names are:

- CMB components: `dipole cmb`
- Diffuse galactic components: `synchrotron freefree thermaldust spindust co`
- SZ effects: `thermalsz kineticsz polarsz`
- Strong point sources: `strongirps strongradiops stronguchii strongwmapps strongercscps`
- Faint point sources: `faintirps faintradiops faintuchii faintwmapps faintercscps`
- The Cosmic Infrared Background: `firb`

The ID key 'PSM_CPID' is a unique identification key that is given to the particular component when it is created by the PSM run.

16.3 PSM observation header

An example of a PSM observation header is given in the observed map header displayed on page 53:

```
COMMENT  --- PSM observation header ------------------------------------------
PSM_OBID= 'rWfTLUpiDe6RJUEY'   /PSM observation ID key
CMIX1   = 'synchrotron'        /Component included in observation
CMIX2   = 'freefree'           /Component included in observation
CMIX3   = 'thermaldust'        /Component included in observation
CMIX4   = 'spindust'           /Component included in observation
CMIX5   = 'co      '           /Component included in observation
```

The PSM observation header block comprises an ID key, and a number of keywords of the form `CMIX`x, where x is a number. Each one of the `CMIX` keywords is used to specify the name of one component present in the observed map.

16.4 PSM map header

A typical PSM map header block is:

```
COMMENT  --- PSM map header ----------------------------------------------------
PSM_PXTP= 'HEALPIX '           /Pixelisation type
PSM_LMAX=                 2000 /Maximum multipole number
PXWIN   =                    0 /Pixel window function
```

The PSM map header comprises a keyword that specifies the pixelisation scheme (the `PSM_PXTP` keyword, a keyword that specifies the maximum harmonic mode included in the map (). For HEALPix maps (the only implemented map type in the PSM at present) there also is a keyword that specifies the window function. The value of the latter can be 0 (if the map is sampled at the centers of the HEALPix pixels), or any power of 2 (i.e. any value of possible HEALPix `nside` parameter).

16.5 PSM alm header

A PSM fits file containing spherical harmonics typically comprises 1 or 3 extensions, depending on whether the data is polarised or not. The PSM alm header block looks typically as follows:

```
COMMENT  --- PSM alm header ----------------------------------------------------
ALM_FLD = 'T       '           /Name of alm field stored in this extension
PSM_LMAX=                 1200 /Maximum multipole number
PXWIN   =                  512 /Pixel window function
```

Fields are typically T, E, and B for polarised harmonic modes for a polarisation observation. Lensing potential alms are labelled with `ALM_FLD = 'P'`. The other keywords are the same as those used in map header blocks.

16.6 PSM cl header

A PSM fits file containing a multivariate power spectrum comprises a cl header block such as::

```
COMMENT  --- PSM cl header -----------------------------------------------------
PSM_LMAX=                 3500 /Maximum multipole number
PXWIN   =                    0 /Pixel window function
```

16.7 PSM band header

Band headers in the PSM store the information about the frequency band associated with the data stored in the fits file. This PSM header block is useful for data of 'observation' type (i.e. for which the `PSM_DTTP` keyword in the base header is `'OBS '`. A typical header for a tophat band, such as those used by the Planck HFI and LFI 'bluebook' instrument, is:

```
COMMENT  --- PSM band header ---------------------------------------------------
BD_SHP  = 'TOPHAT  '           /Band shape (e.g. 'DIRAC', 'TOPHAT', 'INSTR')
BD_LNU  =          9.00000E+10 /Band lower frequency
BD_UNU  =          1.15000E+11 /Band upper frequency
```

This header stores the shape of the band in the keyword `BD_SHP`, and the lower and upper bounds of the frequency band. Different header blocks are implemented for other types of bands (instrumental tabulated bands, for which `BD_SHP` = `'INSTR '` ; monofrequency bands, for which `BD_SHP` = `'DIRAC '`):

```
COMMENT  --- PSM band header ---------------------------------------------------
BD_SHP  = 'INSTR   '           /Band shape (e.g. 'DIRAC', 'TOPHAT', 'INSTR')
BD_INSTR= 'HFI_RIMO'           /Instrument for the specified band
BD_VERS = '20120124'           /Version for the specified band
BD_CHAN = 'F143    '           /Channel for the specified band
```

```
COMMENT  --- PSM band header ---------------------------------------------------
BD_SHP  = 'DIRAC   '           /Band shape (e.g. 'DIRAC', 'TOPHAT', 'INSTR')
BD_CNU  =          1.00000E+11 /Band central frequency
```

Software tools are available in the PSM to read this information in the fits file headers, and convert it into usable band 'objects' that can be used for unit conversion, color correction ,ad band-integration.

16.8 PSM beam header

The PSM beam header block stores the information about the beam associated with the data stored in the fits file.

17 Important technical aspects

17.1 Bibliographic information

Essential bibliographic information about the model generated is provided in two files, which are written at the time of PSM execution in the `psminfo/` subdirectory of the output directory.

The information about the model used is written in `psm_citations.txt`, and the corresponding bibliography in `psm_bibliography.txt`. Please use the information provided there to give proper credit to the original work that has been used to generate your particular sky model.

17.2 Units

The PSM uses strict unit conventions that are used in all output data sets. Conversion between these units is implemented in a single routine, `conversion_factor.pro`, in the `tools/units/` subdirectory of the PSM software distribution.

All units can be prefixed by any of the following `['n','u','m','k','M','G']` for nano, micro, milli, kilo, Mega, Giga, and optionally raised to an integer power, in which case the unit is in parentheses and postfixed by `['**2','**3',...]`. For instance, `'(mK_CMB)**2'` is a valid PSM unit.

17.2.1 Brightness units

The list of brightness units used by the PSM is: `['Jy/sr','K_CMB','K_RJ','K/KCMB','y_sz','W/m2/sr/Hz']` Conversion from one of these units to another is frequency dependent, except for the conversion between `'Jy/sr'` and `'W/m2/sr/Hz'`. The `conversion_factor.pro` program provides this conversion for any frequency, or for any frequency band.

Note that the MJy/sr units used in the PSM do not assume any spectral shape (contrarily to the 'IRAS convention' sometimes used among members of the Planck consortium). In the PSM, 1MJy/sr equals 10^{-20} W/m^2/sr/Hz, with no convention assumed (usual definition of units, as can be found on wikipedia or elsewhere).

17.2.2 Mass units

The list of mass units used by the PSM is: `['gram','Msun']`.

17.2.3 Angle units

The list of angle units used by the PSM is: `['rad','deg','arcmin','arcsec']`.

17.2.4 Length units

The list of length units used by the PSM is: `['meter','parsec']`. Hence, `'kmeter'` is a valid PSM unit, but `'kilometer'` or `'km'` are not – at least for this release.

17.3 Temporary files

The PSM requires writing and reading temporary files during its execution. The directory used for this is set by the environment variable `IDL_TMPDIR`. Temporary files can be large, and the execution time of the PSM can depend significantly on the I/O rate to write and read them.

Names for temporary files are generated automatically during the PSM run, and are of the form:

```
psm_tmpfile_{key}.ext,
```

where {`key`} is a randomly generated key comprising 16 characters, and `.ext` is the file extension. A typical temporary file name can be, for instance:

```
psm_tmpfile_4iT1o6c24yvr1QZ2.fits
```

If you interrupt the execution of the PSM, it is possible that a temporary file has been created, but the PSM process has been interrupted before the file has been erased. It is recommended to check for forgotten temporary files (created during the present PSM run) using:

```
IDL> PRINT, PSM_TMPFILES()
```

If necessary, erase any forgotten temporary files with the command:

```
IDL> ERASE_TMPFILES
```

This last command erases only files created during the present PSM run. To check for and/or erase all temporary PSM files, including those created by another run of the PSM, set the `/allpsm` keyword in the calls, e.g.:

```
IDL> PRINT, PSM_TMPFILES(/allpsm)
IDL> ERASE_TMPFILES, /allpsm
```

This will erase all files matching the standard PSM temporary file format (make sure neither you nor a colleague have any (other) PSM process(es) writing useful temporary files in the same `IDL_TMPDIR` before using this command).

When not set by the user, the `IDL_TMPDIR` default value is the standard directory used for this purpose by the current operating system. This can be changed, for instance, to a personal temporary directory. In bash, this is done for instance using a command such as:

```
export IDL_TMPDIR="/scratch/$USER.$(date +%s)"
```

Large parallel computers often provide such dedicated space. Check with your system administrator what is the correct place to use. On personal computers, the default value of `IDL_TMPDIR` is usually a good choice.

17.4 Seeds for random number generation

18 Some useful PSM software tools

The PSM software distribution comprises various tools that can be useful for various purposes besides the generation of simulations using the `PSM_MAIN` procedure. This section describes the most useful of them.

18.1 Documentation and online help

18.1.1 Documentation

Partial documentation about PSM programs can be generated using the `PSMDOCGEN` procedure. Simply type `PSMDOCGEN` in the IDL command line, and an html document named `psm_documentation.html` will be generated in the `/doc` subdirectory of the PSM software directory. Only partial information is available so far, however.

18.1.2 `PSMHELP`

A complete list of PSM procedures and functions is printed out in the IDL standard output by typing

```
IDL> psmhelp
```

The output can be limited to programs that contain a particular template in their names as follows, e.g.

```
IDL> psmhelp, 'instrument'
```

will print out all programs that contain `instrument` in their names. The output is:

```
A ---------------------------------------------------------------------------------------
  ALL_INSTRUMENTS
I ---------------------------------------------------------------------------------------
  INSTRUMENTAL_NOISE        INSTRUMENT_LIST
L ---------------------------------------------------------------------------------------
  LOAD_INSTRUMENT
P ---------------------------------------------------------------------------------------
  PRINT_INSTRUMENT
-----------------------------------------------------------------------------------------
```

18.1.3 `PROHELP`

For a large fraction of PSM procedures and function, short online help can be obtained using the `PROHELP` procedure, e.g., typing

```
IDL> prohelp, load_instrument
```

will print out:

```
--------------------------------------------------------------------------------------------
LOAD_INSTRUMENT: function which returns a structure describing an instrument
% ----> SYNTAX: result = LOAD_INSTRUMENT(instrument_name, channel_list=, version=, freqs=)
% ----> Choose instrument_name from the following list:
% ---->     PSM_IDEAL
% ---->     HFI_IDEAL
% ---->     LFI_IDEAL
% ---->     HFI_BLUEBOOK
% ---->     LFI_BLUEBOOK
% ---->     HFI_RIMO
% ---->     LFI_RIMO
% ---->     WMAP
% ---->     IRAS_IDEAL
% ---->     IRAS_TABLEBANDS
% ----> channel_list is the list of channels to be included
% ----> (default: all individual detectors, or all frequency bands if /freqs is set)
% ----> version sets the version for HFI_RIMO, LFI_RIMO, WMAP
----------------------------------------------------------------------------------------------
```

18.2 Instrument structures

An instrument is represented as a collection of channels, with each a (symmetric) beam, a spectral band, noise properties, polarisation properties, pixelisation properties. It is represented internally by the PSM as a (somewhat complex) structure, created with the `LOAD_INSTRUMENT` procedure. For instance, the following command line in IDL creates a PSM instrument that represents in a simplified way the WMAP instrument (7-year version), with here 5 channels only (one per frequency band), and ideal monochromatic spectral band approximation:

```
IDL> wmap = LOAD_INSTRUMENT('WMAP', version='7yr', /freqs)
```

Information about the corresponding WMAP instrument can be visualised using the `PRTSTRUCT` utility, e.g.

```
IDL> prtstruct, wmap, name='WMAP'
```

```
WMAP | .NAME       |              |  ------------------>    STRING    = 'WMAP'
WMAP | .VERSION    |              |  ------------------>    STRING    = '7yr'
WMAP | .NCHANNEL   |              |  ------------------>    INT       =        5
WMAP | .CHANNEL_1  | .INST        |  ------------------>    STRING    = 'WMAP'
     |             | .VERSION     |  ------------------>    STRING    = '7yr'
     |             | .NAME        |  ------------------>    STRING    = 'K'
     |             | .BAND        | .SHAPE ------------>    STRING    = 'DIRAC'
     |             |              | .NU_C ------------->    FLOAT     =   2.30000e+10
     |             |              | .INTEG_DNU -------->    DOUBLE    =        1.0000000
     |             | .BEAM        | .TYPE ------------->    STRING    = 'INSTR'
     |             |              | .INSTR ------------>    STRING    = 'WMAP'
     |             |              | .VERSION ---------->    STRING    = '7yr'
     |             |              | .CHANNEL ---------->    STRING    = 'K'
     |             | .PIX         | .PIXTYPE ---------->    STRING    = 'HEALPIX'
     |             |              | .NSIDE ------------>    LONG      =          512
     |             |              | .NPIX ------------->    LONG      =      3145728
     |             |              | .ORDERING --------->    STRING    = 'NESTED'
     |             |              | .OBJECT ----------->    STRING    = 'FULLSKY'
     |             |              | .FIRSTPIX --------->    LONG      =            0
     |             |              | .LASTPIX ---------->    LONG      =      3145727
     |             |              | .INDXSCHM --------->    STRING    = 'IMPLICIT'
     |             |              | .COORDSYS --------->    STRING    = 'G'
     |             | .STOKES      |  ------------------>    STRING    = 'TQU'
     |             | .OBS_UNITS   |  ------------------>    STRING    = 'mK_CMB'
     |             | .CONVERSION  | .YSZ2UNITS -------->    DOUBLE    =       -5375.6860
     |             |              | .KCMB2UNITS ------->    DOUBLE    =        1000.0000
     |             |              | .KRJ2UNITS -------->    DOUBLE    =        1013.7430
     |             |              | .MEGAJYSR2UNITS --->    DOUBLE    =        62.373463
...
```

Here, only the first channel information has been reprinted. Note that units used for the standard observations in each channel, as well as main unit conversion coefficients, are included in the description of each channel of the instrument. Less complete, but easier to read information can be printed out using the `PRINT_INSTRUMENT` utility, e.g.

```
IDL> print_instrument, wmap

        channel            FWHM        YSZ2KCMB        KRJ2KCMB      MJYSR2KCMB
 --------------  --------------  --------------  --------------  --------------
              K             N/A      -5.3756860       1.0137430     0.062373463
             Ka             N/A      -5.2974526       1.0284615     0.030738867
              Q             N/A      -5.2152334       1.0442069     0.020218388
              V             N/A      -4.9356367       1.0999517    0.0096214733
              W             N/A      -4.2586475       1.2503038    0.0046056137
```

Note that for this instrument, beams are not Gaussian, and hence are not described by a single FWHM per channel (hence the `N/A` in the FWHM column of the above printout).

18.2.1 Spectral bands

18.2.2 Detector beams

18.2.3 Noise description

18.3 Band integration and color correction

The PSM uses several types of emission laws, described above in Section 15.2.4, and uses structures that describe spectral bands of instruments such as Planck and IRAS. Procedures and functions that combine both types of data for band integration and color correction are implemented in the PSM, and are described below.

18.3.1 Band integration

18.3.2 Color correction coefficients

The `COLORCOR` function is a very simple tool for computing color correction for `POWERLAW` and `GREYBODY` PSM emission laws. the call is:

```
result = COLORCOR(band, emlaw, nuref=, specind=, temp=, /double)
```

where `band` is a structure representing a spectral band, `emlaw` is the name of the emission law, `nuref` is the reference frequency for color correction, and `specind` and `temp` are parameters of the emission law. For instance, first define a band using the `GET_BAND_STRUCT` procedure:

```
band = GET_BAND_STRUCT('INSTR', instr='HFI_RIMO', version='DX9-v1', channel=['857-1'])
```

then find the color correction at 857 GHz for a greybody (modified blackbody) with temperature and spectral index $T = 10\,\mathrm{K}$ and $\alpha = 1.6$, using:

```
PRINT, COLORCOR(band, 'GREYBODY', nuref=857e9, specind=1.6, temp=10.)
```

The result is 1.0140338. The brightness (in MJy/sr, or W/m^2/sr/Hz) at the reference frequency (857 GHz here) is obtained from the average brightness (in the same units) within the spectral band of detector `857-1` of version DX9-v1 of the HFI RIMO by multiplication by the output of `COLORCOR`, i.e.

$$I_\nu(857\,\mathrm{GHz}) = 1.0140338 \times \int_0^\infty h(\nu) I_\nu \, \mathrm{d}\nu \qquad (5)$$

where $h(\nu)$ is the normalised (i.e. $\int_0^\infty h(\nu)\,\mathrm{d}\nu = 1$) spectral band of interest.

Acknowledgements

The PSM project has benefitted from useful discussions with: Karim Benabed, Rodney Davis, François-Xavier Désert, Hans-Kristian Eriksen, Frode Hansen, Lauro Moscardini, Francesca Perrotta, Stephen Serjeant, Grazia Umana, Benjamin Wandelt.

We thank the PSM users who have been testing the consecutive versions of the package and have helped validating simulation outputs and finding/fixing some of the bugs. We thank in particular Charmaine Armitage, Julian Borrill, Jason Dick, Joanna Dunkley, Maxence Fournier, Frédéric Guilloux, Martin Reinecke, Mathieu Remazeilles, and Graça Rocha.