# FLASK:
# Full-sky Lognormal Astro-fields Simulation Kit

Usage and installation manual

**Written by:** Henrique S. Xavier
**Contact:** `hsxavier@if.usp.br`

University College London, UK
Universidade de São Paulo, Brazil

January 7, 2016

# Contents

Chapter 1

# Disclaimer

Thank you for using this software. Please report any bugs to `hsxavier@if.usp.br` and acknowledge FLASK by citing the publication Xavier et al. (2016).

Chapter 2

# Installation

## 2.1 Requirements

FLASK is a code written in C++ that uses OPENMP for parallelization; HEALPIX[1] for mapping the sky and performing harmonic transforms; the Gnu Scientific Library[2] (GSL) to generate pseudo-random numbers, to perform Cholesky decompositions and other tasks; and the CFITSIO[3] library to input and output FITS files. Apart from HEALPIX, all the remaining dependencies should be available on Linux and Mac repositories. HEALPIX however should be easy to install (just follow their instructions); FLASK requires version 3.11 or later of the C++ HEALPIX implementation due to specific functions implemented only on these versions.

So far FLASK has been tested only on Linux distributions – Scientific Linux 5.6 and Ubuntu 14.04 – and compiled with g++ (GCC) versions 4.8.1 and 4.8.4. Besides the main code and other routines in C++, FLASK also includes auxiliary PYTHON and SHELL scripts. These were tested in BASH shell and PYTHON versions 2.7.3 and 2.7.6. The PYTHON scripts require the a few packages like NUMPY, SCIPY and HEALPY (which can be installed by following HEALPIX instructions). Check the scripts for further dependencies.

## 2.2 Compiling

Before compiling, you will have to modify the `Makefile` in the `src` sub-directory of FLASK. The following lines must be changed according to the location of the HEALPIX installation directory:

```
HEALDIR = <path to Healpix directory>
CXXHEAL = -I<path to Healpix header files>
LDHEAL = -L<path to Healpix library files>
```

In case other libraries such as CFITSIO, GSL are installed in non-standard locations, you might need to specify their locations with similar compiler flags. The default compiler

[1] http://healpix.jpl.nasa.gov
[2] http://www.gnu.org/software/gsl
[3] http://heasarc.gsfc.nasa.gov/fitsio

in the `Makefile` is `g++`, you might need to change the keyword `COMP` as well if using a different compiler.

After changing the `Makefile`, run the command `make` in the `src` directory. The executables will be built in FLASK `bin` sub-directory. You may add this sub-directory to your PATH environment variable in order to be able to run FLASK anywhere; otherwise, you will require to run it from there or to provide the full path as usual.

Chapter 3

# Usage

## 3.1 Using FLASK

### 3.1.1 Quick start

To get a feeling of FLASK and to test if things are working fine, go to the FLASK directory and run the command:

```
./bin/flask example.config
```

This should result in a clean run (no errors or warnings) and should make FLASK use typical data stored in the `data` sub-directory to create the most relevant outputs into the `example` sub-directory (which should be empty), including maps and catalogs simulations. Since writting files to the hard-disk is a slow process, this run will take much longer than an usual FLASK run that only outputs one or two files. To figure out what the output files are, check the file `example.config` and Sec. 3.1.4.

### 3.1.2 Basic operation

FLASK is executed through the command line, and it always requires a configuration file (e.g. `example.config` in the FLASK directory):

```
flask <config file>
```

Any keyword in the configuration file can be altered from the command line by specifying its name `<keyword_i>` and value `<value_i>` after passing the configuration file to FLASK:

```
flask <config file> <keyword_1>:  <value_1> <keyword_2>:  <value_2>...
```

For example,

```
flask example.config RNDSEED: 334 MAP_OUT: ./map-002.dat
```

FLASK outputs to the screen a comprehensive description of what it is doing. In case you want to store this information for later, you might want to redirect the output to a file,

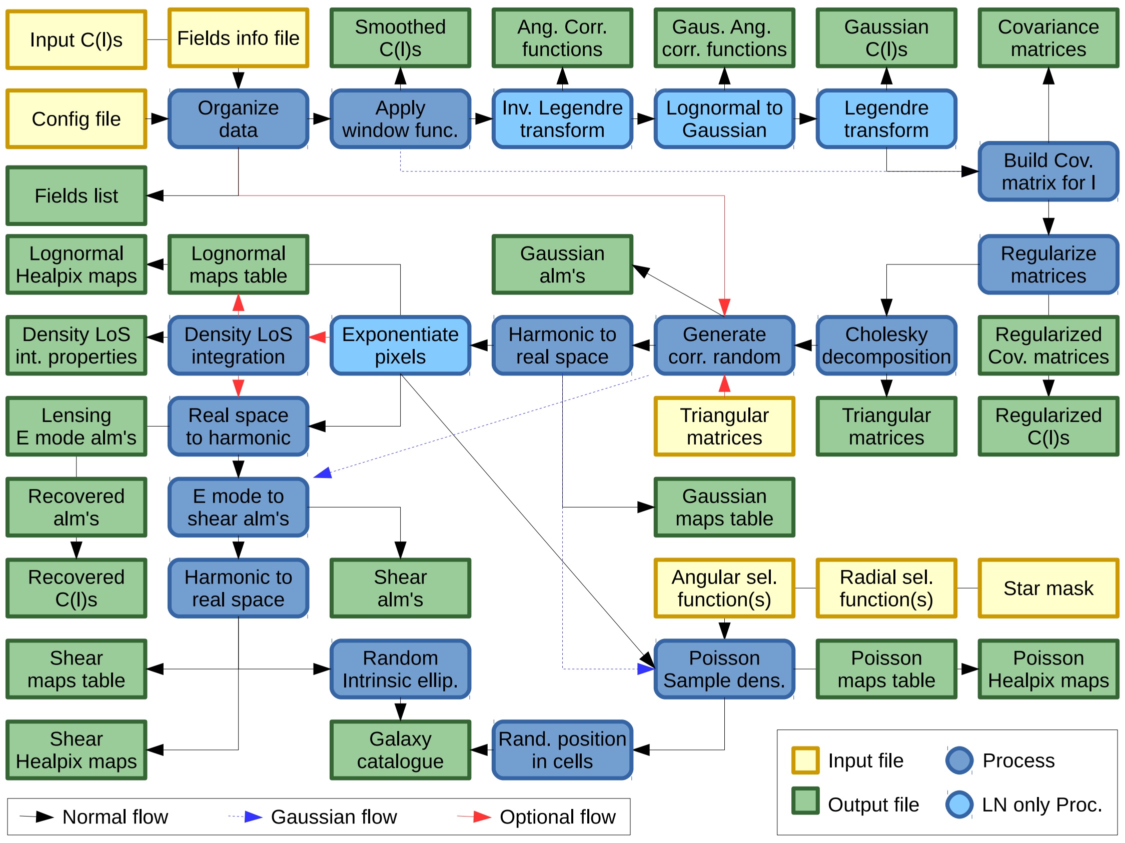

**Figure 3.1:** FLASK basic flow chart.

e.g.:

```
flask example.config RNDSEED: 334 MAP_OUT: ./map-002.dat > r334.log
```

## 3.1.3 Inner workings

The internal process followed by FLASK is described in Xavier et al. (2016). Fig. 3.1 shows a flow chart that roughly described the operations sequences. Some processes are only performed when simulating lognormal fields (Gaussian simulations may skip a few of them). Other processes like creating triangular matrices used for generating correlated random variables and performing a density line-of-sight (LoS) integration to get the convergence are optional (they are executed according to the specifications in the configuration file).

## 3.1.4 Configuration file keywords

FLASK is built in a way that all the existing keywords in the code must be present in the configuration file otherwise a warning message is issued; this is to avoid setting keywords to values unknown by the user and to disclose to the user all available options in the code. FLASK will also complain if keywords that do not exist in the code are present in the configuration file or are passed through the command line; keywords are identified by a colon (:) after it, so this should not be used for other purposes inside configuration files. Although the hash (#) is used to identify comments, this is only for aesthetic reasons and does not have any effect on the parsing of the configuration file (`# DENS2KAPPA:` is still identified as the `DENS2KAPPA` keyword). More information about this can be found in the `example.config` file.

FLASK goes through a series of sequential computations that transforms the simulated data somehow; after each step, the code can output the current status of the data. These possible outputs are presented in the `example.config` file in the exact order they are produced. Thus, one can have an idea of the simulation process by reading the output options in that file. All outputs can be turned off by assigning `0` to the respective keyword.

We now provide a description of every keyword available in the code (a brief description of each one is also provided in `example.config`):

- `DIST:` This takes as value one string that can be either `GAUSSIAN`, `LOGNORMAL` or `HOMOGENEOUS`, specifying what kind of distribution all the simulated random fields will follow. The `HOMOGENEOUS` in fact ignores all input power spectra $C(\ell)$s as if they were all zero and creates homogeneous field maps. Density fields can be later Poisson sampled according to the selection function, so the `HOMOGENEOUS` choice is useful to create the random catalogues used in Landy and Szalay (1993), for instance. Weak lensing shear produced under this choice is currently zero, and ellipticities are completely random according to `ELLIP_SIGMA`. For simulating Gaussian and lognormal correlated fields simultaneously, one should use the `LOGNORMAL` choice and adjust the input information so to approximate the desired fields to a Gaussian distribution.

- `RNDSEED:` This is a single integer number, the seed for the pseudo-random number generator. In fact, the seed is transformed into $N_{\text{proc}}$ seeds, where $N_{\text{proc}}$ is the number of processors available for parallel computing with OPENMP. This transformation is such that every seed provided to `RNDSEED` will result in different and exclusive set of $N_{\text{proc}}$ so that different seeds truly result in independent realisations. Note that a given seed passed to `RNDSEED` will only result in the same realisation as long as $N_{\text{proc}}$ remains the same.

- `POISSON:` This can be `1` or `0`, and specifies if the galaxy density fields should be Poisson sampled or not, respectively. If not sampled (useful for debugging reasons), the value attributed to the HEALPIX map is the expected number of galaxies (a non-integer number). This map can then be output through the `MAPWER_OUT` and `MAPWERFITS_PREFIX` keywords. Generating a galaxy catalogue from this map (with `POISSON: 0`) would not make much sense and is likely to cause the code to crash.

- `OMEGA_m:` This takes a real number that represents the total matter (dark matter plus baryons) density parameter. It is only used if the user sets: (1) `DENS2KAPPA: 1`; or (2) `SELEC_TYPE: 1` or `SELEC_TYPE: 3`.

- `OMEGA_L:` This takes a real number that represents the dark energy density parameter. It is only used if the user sets: (1) `DENS2KAPPA: 1`; or (2) `SELEC_TYPE: 1` or `SELEC_TYPE: 3`.

- `W_de`: This takes a real number that represents the $w_0$ parameter of a constant dark energy equation of state. It is only used if the user sets: (1) `DENS2KAPPA: 1`; or (2) `SELEC_TYPE: 1` or `SELEC_TYPE: 3`. Currently this is the only kind of dark energy model implemented in the code.

- `ELLIP_SIGMA`: This takes a real number that represents the standard deviation of the zero mean Gaussian distribution from which each component of the galaxy intrinsic ellipticity $\varepsilon_\mathrm{s}$ is randomly drawn. The final ellipticity in the catalogue is $\varepsilon = \varepsilon_1 + i\varepsilon_2$:

$$\varepsilon = \begin{cases} \dfrac{\varepsilon_\mathrm{s} + g}{1 + g^*\varepsilon_\mathrm{s}}, & |g| \leq 1; \\ \dfrac{1 + g\varepsilon_\mathrm{s}^*}{\varepsilon_\mathrm{s}^* + g^*}, & |g| > 1; \end{cases} \tag{3.1}$$

  where $g \equiv \gamma/(1+\kappa)$ is the reduced shear, $\gamma = \gamma_1 + i\gamma_2$ is the shear, $\kappa$ is the convergence, and $\varepsilon_\mathrm{s} = \varepsilon_{\mathrm{s},1} + i\varepsilon_{\mathrm{s},2}$ is the galaxy intrinsic ellipticity (we follow Bartelmann and Schneider, 2001, eq. 4.12).

- `GALDENSITY`: This takes a real number representing the 3D comoving galaxy number density in $(h^{-1}\mathrm{Mpc})^{-3}$. It is only used if `SELEC_TYPE: 1` or `SELEC_TYPE: 3`.

- `FIELDS_INFO`: This takes a string representing the path to the fields information file described in Sec. 3.1.5.

- `CHOL_IN_PREFIX`: This takes the character `0` or a string representing the path, including a file prefix, to the files written by the keyword `CHOLESKY_PREFIX` in a previous FLASK run. If set to `0`, the code: takes as input the power spectra $C_\mathrm{in}(\ell)$s specified by `CL_PREFIX`; process it to obtain the associated Gaussian power spectra $C_\mathrm{g}(\ell)$s [if `DIST: LOGNORMAL`, otherwise $C_\mathrm{g}(\ell) = C_\mathrm{in}(\ell)$]; build independent covariance matrices for each $\ell$; regularise the matrices if necessary and requested; and perform a Cholesky decomposition to obtain triangular matrices – which can be written to files by the `CHOLESKY_PREFIX` keyword – used to generate correlated Gaussian variables. If `CHOL_IN_PREFIX` is not set to `0`, FLASK therefore skips all the processing above and loads its final product – the triangular matrices, previously computed – into the memory.

- `CL_PREFIX`: This takes a string that represents the path (including a prefix) to all power spectra $C^{ij}(\ell)$ required to specify the statistical properties of the set of fields listed in the fields information file specified by `FIELDS_INFO`. Each $C^{ij}(\ell)$ must be in a separate file with two columns ($\ell$ and $C^{ij}(\ell)$). This string is only used if `CHOL_IN_PREFIX` is set to zero.

- `ALLOW_MISS_CL`: This can be `0` or `1`. If `0`, FLASK will abort if any required $C^{ij}(\ell)$ is not found. If `1`, the code will assume that the missing $C^{ij}(\ell)$s are zero. This works for cross power spectra but will cause the program to abort if auto power spectra are missing.

- WINFUNC_SIGMA: In case you want to simulate fields that have been convolved (smoothed) in each redshift slice with a 2D Gaussian, you can adjust the Gaussian's standard deviation here (this takes a real number). The power spectra will be adjusted accordingly. To skip applying this smoothing, set this value to something less than zero.

- APPLY_PIXWIN: To compute the value of a pixel from the map's harmonic coefficients, HEALPIX sums the multipoles weighted by their coefficients (i.e. performs an inverse harmonic transformation) and computes the result at the angular coordinates of the pixel's centre (the pixel value is not the field average inside the pixel). In case you want the pixel value to be the field average inside the pixel, you must apply the HEALPIX window function to the input power spectra; this is done by setting APPLY_PIXWIN to 1. To skip this multiplication, set APPLY_PIXWIN: 0.

- SUPPRESS_L: This takes a real number that represents the scale $\ell_{\rm sup}$ at which the input power spectra $C^{ij}_{\rm in}(\ell)$ will be exponentially suppressed with index $n$ given by SUP_INDEX. The suppressed power spectra $C^{ij}_{\rm sup}(\ell)$, used for all subsequent computations and simulations, will be:

$$C^{ij}_{\rm sup}(\ell) = C^{ij}_{\rm in}(\ell)\exp\left[-\left(\frac{\ell}{\ell_{\rm sup}}\right)^n\right] \tag{3.2}$$

  To avoid applying this suppression, set either SUPPRESS_L or SUP_INDEX (or both) to something less than zero.

- SUP_INDEX: This takes a real number that represents the index $n$ in Eq. 3.2 (see SUPPRESS_L description). You can avoid the use of such suppression by setting this value to a negative number.

- SELEC_SEPARABLE: This specifies if the galaxy selection function is separable between radial and angular parts (1) or not (0). If separable: the selection function value is the multiplication of the two; the keyword SELEC_PREFIX will actually take a single FITS filename representing the angular selection function as a HEALPIX map that will be used for all galaxy fields and all redshift slices; and the keyword SELEC_Z_PREFIX will take a prefix leading to one radial selection function for each galaxy field. If not separable, the keyword SELEC_PREFIX will take a path with a prefix that should lead to one HEALPIX map in FITS format for each galaxy field and redshift slice, representing the entire selection function. In this case, SELEC_Z_PREFIX is ignored.

- SELEC_PREFIX: This takes a string that represents either a path plus a prefix for a bunch of HEALPIX maps in FITS format representing the galaxies selection functions or the filename of a single HEALPIX map, representing the angular part of the selection function (see SELEC_SEPARABLE keyword). It can also be set to 0; in this case, the code works as if the selection function (or the angular selection if SELEC_SEPARABLE: 1) was equal to one in every pixel. For separable selection functions, this corresponds to a full-sky simulation; for non-separable selection function, this is just weird. In the current implementation, the selection functions must have the same $N_{\rm side}$ as the one passed to the NSIDE keyword.

- SELEC_Z_PREFIX: This takes a string representing the path plus a prefix for two column (redshift and selection function) text files, one for each galaxy field. This is ignored if SELEC_SEPARABLE: 0. The files can have headers if they start with #, and they have to be named according to the prefix, followed by f$[f_i]$.dat, where $[f_i]$ is the field number "naming" the field in the fields information file (see Sec. 3.1.5).

- SELEC_SCALE: This takes a real number that will serve as a re-scaling of the selection function (the selection function will be multiplied by this value). To use the exact values in the files specified by SELEC_PREFIX and SELEC_Z_PREFIX, set SELEC_SCALE: 1.0.

- SELEC_TYPE: There are two types of selection functions: one specifies the expected number of observed galaxies per unit redshift, per square arcmin (SELEC_TYPE: 0), and the other specifies the fraction of existing galaxies that are actually observed at each angular position and redshift (SELEC_TYPE: 1). In the latter case, the total existing galaxies is computed from the galaxy density given at GALDENSITY and the comoving volume calculated according to the pixel size (determined by NSIDE), the redshift slice width (set in the fields information file) and the cosmological parameters OMEGA_m, OMEGA_L and W_de. The option SELEC_TYPE: 1 is currently not fully implemented. On top of these two options, the user can specify that the angular part of the selection function is only used for bookkeeping (that is, it does not affect the number of galaxies generated and the simulation is actually full sky) by adding 2 to choice made, leading to the following possible options for SELEC_TYPE: 0, 1, 2 and 3. Currently the bookkeeping option only works for separable selection functions.

- STARMASK: Besides the selection function specified in SELEC_PREFIX, the user can multiply the selection function by a HEALPIX map in FITS file format, passed to this keyword. All selection functions are affected by this angular mask. In the current implementation, this map must have the same $N_{\text{side}}$ as the one passed to the NSIDE keyword. To not use this extra angular selection function, the user can set STARMASK: 0.

- EXTRAP_DIPOLE: Many power spectra calculators like CLASS[1] or CAMB SOURCES[2] write $C(\ell)$s for $\ell \geq 2$. If this keyword is set to 1, then the dipole $C(\ell = 1)$, if missing, is linearly extrapolated from $C(\ell = 2)$; otherwise, if EXTRAP_DIPOLE: 0 and the dipole is missing, then the dipole is set to zero (NOTE: the monopole is always set to zero).

- LRANGE: This takes two integers separated by space representing the minimum $\ell_{\text{min}}$ and maximum $\ell_{\text{max}}$, respectively, that will be used to generate the associated Gaussian multipoles. They should obey $1 \leq \ell_{\text{min}} \leq \ell_{\text{max}} \leq \ell_{\text{in}}$, where $\ell_{\text{in}}$ is the highest $\ell$ provided in the input power spectra.

- SHEAR_LMAX: The shear is derived from the convergence multipoles using the equations from Hu (2000) and the $E$-mode definition from HEALPIX manual. In Gaussian simulations the shear multipoles are derived from the original convergence

[1] http://class-code.net/
[2] http://camb.info/sources

multipoles used to generate the maps; this is not possible in lognormal simulations since the convergence was generated from multipoles of the associated Gaussian field (a similar argument exists for convergences obtained from density LoS integration). Therefore, to compute the shear from lognormal (or density LoS integration) convergences one needs first to obtain their multipoles. The `SHEAR_LMAX` keyword takes one integer representing the maximum multipole that will be extracted from the lognormal convergence maps for this purpose (thus it does not affect Gaussian simulations) and the maximum multipole that will be available in shear maps. Using `SHEAR_LMAX` $>$ `NSIDE` introduces noise in this transformation.

- `NSIDE`: This takes an integer representing the HEALPIX $N_{\text{side}}$ parameter. The number of pixels $N_{\text{pix}}$ used in each redshift slice is $N_{\text{pix}} = 12N_{\text{side}}^2$. Any number can be provided here, although some applications depend on powers of 2.

- `USE_HEALPIX_WGTS`: This keyword states if the code should use (`1`) or not (`0`) the HEALPIX weights to perform the `map2alm` (harmonic transform) operation. The weights are only available for $N_{\text{side}} = 2^n$, where $n \in \mathbb{N}$. If `USE_HEALPIX_WGTS: 0`, the weights will all be set to unity.

- `BADCORR_FRAC`: This takes a real number $f$. In case some covariance matrix entries lead to forbidden correlations $\rho_{ij}$ ($\rho_{ij} > 1$ or $\rho_{ij} < -1$), FLASK will add this fractional change to both variances related to the forbidden correlation: $v_i^{\text{new}} = (1+f)v_i^{\text{old}}$ and $v_j^{\text{new}} = (1+f)v_j^{\text{old}}$.

- `REGULARIZE_METHOD`: In case some associated Gaussian multipoles covariance matrix is non-positive-definite, FLASK can make it positive-definite by changing its entries. There are two methods currently implemented for that, described in Xavier et al. (2016): option `REGULARIZE_METHOD: 1` performs an eigendecomposition of the matrix and sets the negative eigenvalues to a new value specified by `NEW_EVAL`; option `REGULARIZE_METHOD: 2` looks for a direction of greatest change in the negative eigenvalues and apply successive distortions (up to `REG_MAXSTEPS` times) of size given by `REGULARIZE_STEP` in that direction, until all eigenvalues are positive. You may set `REGULARIZE_METHOD: 0` to make FLASK abort in case of non-positive-definite matrices.

- `NEW_EVAL`: This takes a real number. If `REGULARIZE_METHOD` is set to `1`, covariance matrices negative eigenvalues will be replaced by this value. This value is not used for other `REGULARIZE_METHOD` options.

- `REGULARIZE_STEP`: This takes a real number. If `REGULARIZE_METHOD` is set to 2, this will be the size of the step taken in the direction in the covariance matrix elements space that changes the negative eigenvalues the most. This value is not used for other `REGULARIZE_METHOD` options.

- `REG_MAXSTEPS`: This takes an integer number that specifies the maximum amount of steps taken in the covariance matrix elements space when trying to regularise the matrix with `REGULARIZE_METHOD: 2`. This value is not used for other `REGULARIZE_METHOD` options.

- `ADD_FRAC`: If the covariance matrix regularisation fails or if it succeeds but the Cholesky decomposition still fails, FLASK will take the real value passed to this keyword, multiply by the smallest element of the covariance matrix diagonal and add the result to all diagonal elements of that matrix.

- `ZSEARCH_TOL`: When building the galaxy catalogue, FLASK will randomly select a redshift inside the galaxy's redshift slice according to the selection function. To do that, it has to find the selection function local maximum inside the redshift bin. The real value passed to `ZSEARCH_TOL` specifies the precision in redshift to which the maximum is found.

- `EXIT_AT`: This keyword takes a string that must be either `0` or any other keyword that specifies an output (without the colon, case sensitive). The code will stop right after the point where that output is produced, even if it is not set to be produced. If `EXIT_AT: 0`, FLASK will run until the end. The output keywords in the `example.config` file are ordered just like they are produced.

- `FITS2TGA`: This can take the options: `0`, all HEALPIX map outputs remain only in HEALPIX map FITS format; `1`, all HEALPIX map outputs also result in a TGA image format version; `2`, all HEALPIX map outputs are transformed into TGA images.

- `LRANGE_OUT`: This takes to integers separated by space, representing the minimum and maximum $\ell$s that will be output in any $C(\ell)$ or $a_{\ell m}$ outputs; this range does not affect calculations, only outputs, and is has to be included in the range set by `LRANGE`.

- `MMAX_OUT`: This takes an integer. In $a_{\ell m}$ outputs, the multipoles will all be written out if this is set to a negative number. If this is set to $m_{\max}$, only $a_{\ell m}$s with $m \leq m_{\max}$ will be written. In the current implementation, $m_{\max}$ must be less than or equal to the minimum $\ell$ specified in `LRANGE_OUT`.

- `ANGULAR_COORD`: This can take three options: `0`, all text outputs referring to angular position will use the polar and azimutal angles $\theta$ and $\phi$, given in radians, as angular coordinates (HEALPIX style); `1`, these outputs will use $\theta$ and $\phi$ in degrees; `2`, these outputs will use right ascension and declination in degrees as angular coordinates. This choice does not affect `XIOUT_PREFIX` and `GXIOUT_PREFIX` outputs which are always in degrees.

- `DENS2KAPPA`: If set to `1`, FLASK computes a convergence field by integrating the matter density fields along the line of sight (see Xavier et al., 2016, for details). This option requires that density fields in the fields information file are specified in contiguous and ordered redshift bins. The resulting convergence fields are added to the field list and from now on are treated the same way as ordinary convergence fields. If `DENS2KAPPA` is set to `0`, no integration is performed and no new convergence is computed.

- `FLIST_OUT`: This keyword takes a string. If anything other than `0`, the code treat it as a filename and writes the list of fields in use by the code to this file.

- SMOOTH_CL_PREFIX: Path plus prefix for the power spectra output, after applying Gaussian and/or HEALPIX pixel window functions according to keywords WINFUNC_SIGMA and APPLY_PIXWIN. Set to 0 for no output.

- XIOUT_PREFIX: Path with prefix for angular correlation functions $\xi^{ij}(\theta)$, written as two column (angle $\theta$ in degrees and $\xi^{ij}(\theta)$) text files. Currently it can only be calculated if DIST: LOGNORMAL. Set to 0 for no output.

- GXIOUT_PREFIX: Path, including a prefix, for angular correlation functions $\xi_{\mathrm{g}}^{ij}(\theta)$ of the associated Gaussian fields, written as two column (angle $\theta$ in degrees and $\xi_{\mathrm{g}}^{ij}(\theta)$) text files. This is only calculated if the input fields are lognormal. Set to 0 for no output.

- GCLOUT_PREFIX: Prefix (including path) for angular power spectra $C_{\mathrm{g}}^{ij}(\ell)$ of the associated Gaussian fields. This is only calculated if DIST: LOGNORMAL. Set to 0 for no output.

- COVL_PREFIX: Path plus prefix for the Gaussian $a_{\ell m}^{i}$s covariance matrices [for a fixed $\ell$, $C_{\mathrm{g}}^{ij}(\ell)$ is the element $(i,j)$ of the covariance matrix]. Set to 0 for no output.

- REG_COVL_PREFIX: Path with prefix for the covariance matrices that would be output by COVL_PREFIX but after being regularised according to the choice in REGULARIZE_METHOD. Set to 0 for no output.

- REG_CL_PREFIX: Prefix (including path) for the regularised $C^{ij}(\ell)$s. In the current implementation this can only be computed for DIST: LOGNORMAL; therefore the $C^{ij}(\ell)$s output here are for the lognormal fields. For no output, set this keyword to 0.

- CHOLESKY_PREFIX: Path plus prefix for the outcome of the Cholesky decomposition as implemented by the *GNU Scientific Library* (GSL).[3] Set to 0 for no output. These files can be passed as input to FLASK by using the keyword CHOL_IN_PREFIX.

- AUXALM_OUT: Filename (with path) for the table with multipoles coefficients $a_{\ell m}^{i}$ of every associated Gaussian field and redshift slice $i$ being simulated. Since the fields are real, only $a_{\ell m}^{i}$ with $m \geq 0$ are written. Set to 0 for no output.

- AUXMAP_OUT: Filename (with path) for a text table containing the pixel's centre angular coordinates and the values at that point of every associated Gaussian field at every redshift slice being simulated. Set to 0 for no output.

- DENS2KAPPA_STAT: This can be set to 0 for no output, to 1 for output on the screen, or to a filename in case the user wants the output to be written to a file. The output is a table with statistical properties of the convergence obtained from density LoS integration. This information is only produced if DENS2KAPPA: 1.

[3] Check http://www.gnu.org/software/gsl/manual/html_node/Cholesky-Decomposition.html for details.

- MAP_OUT: Path plus filename for a text table containing the pixel's centre angular coordinates and the values at that point of every field (Gaussian or lognormal, according to DIST) at every redshift slice being simulated. This does not include the selection function nor noise apart from cosmic variance. Set to 0 for no output.

- MAPFITS_PREFIX: Prefix (including path) for a HEALPIX map FITS file for each simulated field and redshift slice. The output format can be altered by FITS2TGA. This does not include the selection function nor noise apart from cosmic variance. Set to 0 for no output.

- RECOVALM_OUT: Path plus filename for a text table of multipole coefficients $a^i_{\ell m}$ recovered, using the map2alm HEALPIX function, from the maps that can be output by MAP_OUT and MAPFITS_PREFIX. Set to 0 for no output.

- RECOVCLS_OUT: Path plus filename for a text table of power spectra $C^{ij}_{\rm rec}(\ell)$ recovered from the maps that can be output by MAP_OUT and MAPFITS_PREFIX. In other words, $C^{ij}_{\rm rec}(\ell)$ is the average over $m$ of $a^i_{\ell m}a^{j*}_{\ell m}$, where $a^i_{\ell m}$ can be output by RECOVALM_OUT. Set to 0 for no output.

- SHEAR_ALM_PREFIX: Prefix (including path) for files containing the weak lensing shear $E$-mode multipole coefficients $a^i_{\ell m}$, one for each lensing field and redshift slice. Set to 0 for no output.

- SHEAR_FITS_PREFIX: Prefix (including path) for HEALPIX maps of weak lensing attributes, obtained from lensing $a_{\ell m}$s ($E$-modes that can be output by SHEAR_ALM_PREFIX) by the HEALPIX function alm2map_spin. These FITS files contain three columns: convergence $\kappa$ and shear components $\gamma_1$ and $\gamma_2$. This does not include the selection function nor noise apart from cosmic variance. Set to 0 for no output.

- SHEAR_MAP_OUT: Path plus filename for a text table containing the pixel's centre angular coordinates and the values at that point of the two shear components $\gamma_1$ and $\gamma_2$ of every weak lensing field at every redshift slice. This does not include the selection function nor noise apart from cosmic variance. Set to 0 for no output.

- MAPWER_OUT: Path plus filename for a text table containing the pixel's centre angular coordinates and the values at that point of the fields after applying the selection function and Poisson sampling the galaxy density fields; that is, the values are the number of galaxies in each pixel in each redshift slice (i.e. in each cell). Convergence fields remain the same, without any noise apart from cosmic variance. Set to 0 for no output.

- MAPWERFITS_PREFIX: Prefix (including path) for the HEALPIX maps version of the MAPWER_OUT output, one file per field per redshift slice. Set to 0 for no output.

- CATALOG_OUT: Path plus filename for a galaxy catalogue in text, FITS or compressed FITS formats, chosen accordingly to the filename extension: .dat, .fits or .fits.gz, respectively. Each row is one galaxy from the map described in MAPWER_OUT and the columns are chosen by the user using the CATALOG_COLS keyword. Set to 0 for no output.

- `CATALOG_COLS`: This takes as input everything in the same line of the configuration file after it, and it specifies the columns to be included in the galaxy catalogue output by `CATALOG_OUT`. They can be:
  - `theta`, the polar angle $\theta$ of the galaxy position (or the declination, according to `ANGULAR_COORD`), randomly sampled inside the corresponding pixel;
  - `phi`, the azimutal angle $\phi$ of the galaxy position (or the right ascension, according to `ANGULAR_COORD`), randomly sampled inside the corresponding pixel;
  - `z`, the galaxy's redshift, which is randomly sampled inside the corresponding cell according to the selection function;
  - `galtype`, i.e. the field's name as in the fields information file, see Sec. 3.1.5;
  - `kappa`, the convergence $\kappa$ at the corresponding cell (pixel and redshift slice);
  - `gamma1`, the first component of the shear, $\gamma_1$, at the corresponding cell;
  - `gamma2`, the second component of the shear, $\gamma_2$, at the corresponding cell;
  - `ellip1`, the first component of the ellipticity of the galaxy $\varepsilon_1$, given by Eq. 3.1;
  - `ellip2`, the second component of the ellipticity of the galaxy $\varepsilon_2$, given by Eq. 3.1;
  - `pixel`, the HEALPIX map pixel number of the corresponding cell;
  - `maskbit`, returns a sum of the following integers: 1 if the galaxy would be removed by the angular selection function, i.e., if the angular selection function value at the corresponding pixel is zero (currently this only happens if the selection function is separable); 2 if the galaxy would be removed by the star mask given by `STARMASK`, i.e., if the star mask value at the corresponding pixel is zero; 4 if the angular selection function value $s$ obeys $0 < s < 1$ (only if the selection function is separable); and 8 if the star mask value $m$ obeys $0 < m < 1$. For instance, a `maskbit` of 0 means that the galaxy would be observed while a `maskbit` of 3 means that the galaxy would be blocked both by the angular selection function and the star mask.

  The column names have to be separated by spaces and they can appear in any number (no repeated columns, though) and order.

### 3.1.5 Fields information file

The fields information file is a text file that informs FLASK the fields and redshift slices it should simulate, along with their properties. In the current implementation, these files can have any number of lines serving as headers (that should always start with a `#`). Empty lines can also appear before the tabular data, but they cannot contain spaces. Once the tabular data appears, you should stick to it. The file can end at the last data line or with an empty line (no spaces allowed). An example of such file is given in Fig. 3.2.

```
# Field number, z bin number, mean, shift, field type, zmin, zmax
# Types: 1-galaxies 2-lensing

     1       1    0.0000    1.0000       1    0.2500    0.3500
     1       2    0.0000    1.0000       1    0.3500    0.4500
     1       3    0.0000    1.0000       1    0.4500    0.5500
     2       1    0.0000    0.0050       2    0.2500    0.3500
     2       2    0.0000    0.0084       2    0.3500    0.4500
     2       3    0.0000    0.0126       2    0.4500    0.5500
```

**Figure 3.2:** Example of a fields information file with two fields (one of type *galaxy* or *matter density* and the other of type *weak lensing convergence*) in three redshift slices. The column order is summarised in the header.

The fields information file columns must be, in this order:

- An arbitrary natural number “naming” the field (for multi-tracer simulations, they can identify different galaxy populations).
- An arbitrary natural number “naming” the redshift slice.
- The fields mean value at that redshift slice, usually zero (for density contrast and convergence, for instance);
- The fields shift parameter, if the field is lognormal (in case of a Gaussian simulation, this column still have to be present but it will be ignored).
- The field type, which determines how the field is treated inside the code. In the current implementation, this can be either 1 (CMB or gas temperature, matter density or matter density tracers like galaxies, quasars, etc.) or 2 (weak lensing convergence). These types affect how FLASK deals with selection functions, noise and field transformations. For details, one can look for `fshear` and `fgalaxies` in the code using the command `grep`.
- The lower boundary of the redshift slice;
- and the upper boundary of the redshift slice. The redshift slices may overlap, have any size and ordering (unless you are integrating the density, see the `DENS2KAPPA` keyword in Sec. 3.1.4). In the current implementation, these are only used either in the density LoS integration controlled by the `DENS2KAPPA` or to Poisson sample the density fields; in both cases, the redshift slice is assumed to be a top-hat redshift window function. If you are not performing these operations, these redshift boundaries are irrelevant and the slices can present arbitrary redshift window functions (e.g. Gaussians), according to the power spectra provided as input.

To simulate the fields and redshift slices mentioned in a fields information file, FLASK requires their auto power spectra and all cross power spectra (unless the keyword `ALLOW_MISS_CL` is set to `1`). Each power spectrum has to be in a file with prefix given by the keyword `CL_PREFIX` followed by `f`$[f_i]$`z`$[z_i]$`f`$[f_j]$`z`$[z_j]$`.dat`, where $[f_i]$ and $[z_i]$ are the numbers (no zero-padding allowed) in the first and second columns of the fields information file. For instance, if the configuration file have `CL_PREFIX: data/testCl-` and `FIELDS_INFO` referring to the file in Fig. 3.2, FLASK will look for the files:

```
data/testCl-f1z1f1z1.dat  data/testCl-f2z1f1z1.dat
data/testCl-f1z1f1z2.dat  data/testCl-f2z1f1z2.dat
data/testCl-f1z1f1z3.dat  data/testCl-f2z1f1z3.dat
data/testCl-f1z1f2z1.dat  data/testCl-f2z1f2z1.dat
data/testCl-f1z1f2z2.dat  data/testCl-f2z1f2z2.dat
data/testCl-f1z1f2z3.dat  data/testCl-f2z1f2z3.dat
data/testCl-f1z2f1z1.dat  data/testCl-f2z2f1z1.dat
data/testCl-f1z2f1z2.dat  data/testCl-f2z2f1z2.dat
data/testCl-f1z2f1z3.dat  data/testCl-f2z2f1z3.dat
data/testCl-f1z2f2z1.dat  data/testCl-f2z2f2z1.dat
data/testCl-f1z2f2z2.dat  data/testCl-f2z2f2z2.dat
data/testCl-f1z2f2z3.dat  data/testCl-f2z2f2z3.dat
data/testCl-f1z3f1z1.dat  data/testCl-f2z3f1z1.dat
data/testCl-f1z3f1z2.dat  data/testCl-f2z3f1z2.dat
data/testCl-f1z3f1z3.dat  data/testCl-f2z3f1z3.dat
data/testCl-f1z3f2z1.dat  data/testCl-f2z3f2z1.dat
data/testCl-f1z3f2z2.dat  data/testCl-f2z3f2z2.dat
data/testCl-f1z3f2z3.dat  data/testCl-f2z3f2z3.dat
```

However, since the cross power spectra are symmetric (testCl-f$[f_i]$z$[z_i]$f$[f_j]$z$[z_j]$.dat $=$ testCl-f$[f_j]$z$[z_j]$f$[f_i]$z$[z_i]$.dat ), FLASK only requires 21 of the files listed above.

In case you use CLASS[4] (Blas et al., 2011; Dio et al., 2013) or CAMB SOURCES[5] (Challinor and Lewis, 2011) to compute the input power spectra, it is possible to use the PYTHON scripts called `class2info.py` and `camb2info.py` in FLASK's `src/scripts` folder to create the fields information file corresponding to CLASS and CAMB SOURCES input (`.ini`) files. See Sec. 3.2 and the scripts' docstring for more information.

### 3.1.6 Angular power spectra files

Apart from the distribution's PDF – determined by the `DIST` keyword in the configuration file – and the fields mean and shift parameters in the fields information file, all statistical properties of the simulated fields are fixed by the input angular power spectra $C^{ij}_{\rm in}(\ell)$, including their cross-correlations. All characteristics such as: tracer biases with respect to matter density; redshift space distortions; galaxy intrinsic alignments; integrated Sachs-Wolfe effect; tracer evolution with redshift; and magnification bias can be included in the simulations by providing the appropriate $C^{ij}_{\rm in}(\ell)$ with such information encoded in them. The step of computing these power spectra is not part of FLASK and is left to the user. Some public codes available to perform such calculations are: CLASS[6] (Blas et al., 2011; Dio et al., 2013) and CAMB SOURCES[7] (Challinor and Lewis, 2011).

[4] http://class-code.net/
[5] http://camb.info/sources
[6] http://class-code.net/
[7] http://camb.info/sources

In the current implementation, the $C_{\rm in}^{ij}(\ell)$ files must be named as explained in 3.1.5 and must have two columns: $\ell$ and $C_{\rm in}^{ij}(\ell)$ (no headers allowed here). The $C_{\rm in}^{ij}(\ell)$ is the power spectra itself and not something like $l(l+1)/(2\pi)\cdot C_{\rm in}^{ij}(\ell)$. They are interpolated by the code so it is not necessary to specify them at every $\ell$ or at integer $\ell$s. For the user's convenience, PYTHON scripts called `prepClassInput.py` and `prepCambInput.py` are available in FLASK's `src/scripts` folder that can translate CLASS and CAMB SOURCES output power spectra into FLASK input power spectra. See Sec. 3.2 and the scripts' docstring for more information.

## 3.2 Auxiliary codes

There are several smaller routines available in FLASK's directory that can help the user prepare its inputs and visualise or organise its outputs. Here we list the most important of them. More information can be found on their docstrings.

### 3.2.1 `Dens2KappaCls`

This is a C++ code compiled to the `bin` sub-directory that takes the input power spectra and configuration file that would be used by FLASK and compute the expected power spectra that would be recovered from convergence computed by integrating simulated lognormal densities along the line of sight (i.e., this code computes the red solid lines in Fig. 11 of Xavier et al. (2016).

The `Dens2KappaCls` usage is similar to that of FLASK: you run it by invoking the executable followed by the path to the configuration file, followed by keywords you want to change. The difference is that the last parameter passed to `Dens2KappaCls` must be the output files prefix for the $C(\ell)$s of the convergence obtained from density LoS integration, for instance:

```
Dens2KappaCls example.config NSIDE: 128 output/kappaCl-
```

Although this code takes the same configuration file as FLASK and will complain if any keyword is missing, the only keywords it actually uses are: `FIELDS_INFO`, `CL_PREFIX`, `ALLOW_MISS_CL`, `OMEGA_m`, `OMEGA_L` and `W_de` (therefore, changing `NSIDE` in the example above is a bit silly).

In the current version, `Dens2KappaCls` does not apply any window function or suppression to the input $C(\ell)$s before integrating them. In case you want to compare its output with recovered $C(\ell)$s from FLASK and have applied any of these changes to the FLASK input power spectra, you should use the `SMOOTH_CL_PREFIX` keyword in FLASK to output files to be used as input here.

### 3.2.2 `GenStarMask`

This routine, written in C++, creates a fulls-sky random star mask with parameters given by the user and then outputs it to a HEALPIX map in FITS format. The parts without stars have value 1 and the parts with stars (which are modelled as disks) have value 0. If necessary, these values can be changed in the source code by redefining the `StarValue` and `EmptyValue` variables.

`GenStarMask` takes 6 inputs, in this order: a integer used as seed for random numbers; the $N_{\mathrm{side}}$ HEALPIX resolution parameter; a minimum angular radius for the stars $r_{\mathrm{min}}$, given in arcmin; a maximum angular radius for the stars $r_{\mathrm{max}}$, given in arcmin; the sky fraction to be covered by stars; and the filename for the output.

The stars (modelled as disks) are distributed homogeneously on the sky, and the natural logarithmic of the disk radius are homogeneously sampled between $\ln(r_{\mathrm{min}})$ and $\ln(r_{\mathrm{max}})$. The stars can be superimposed and are generated until the fraction of pixels covered by stars is greater than the value specified in the input.

### 3.2.3 `camb2info.py`

This PYTHON script takes a CAMB SOURCES input file and creates a FLASK fields information file. It takes as input, in this order: the CAMB SOURCES `.ini` filename and the output filename.

There are three methods for computing the shift parameter for convergence: a table read from a file which is interpolated; the formula from Hilbert et al. (2011); and a formula computed from FLASK density line of sight integration. The latter is currently used (the others are commented out in the script). To generate the `zmin` and `zmax` columns in the fields information file, this routine uses the `redshift_sigma` entry in the CAMB SOURCES `.ini` file as half of the redshift bin width. Note that CAMB SOURCES assumes, by default, that the redshift bins have Gaussian selection functions. If necessary, the user must modify CAMB SOURCES to use top-hat redshift bins.

### 3.2.4 `class2info.py`

This PYTHON script works in the same way as `camb2info.py` but for CLASS input (`.ini`) files (it uses the `selection_width` keyword as half of the redshift bin width).

### 3.2.5 `ChangeMapResolution.py`

This PYTHON script gets a HEALPIX map in FITS format, changes its resolution (given by $N_{\mathrm{side}}$) and write the new map to another FITS file. Its inputs are, in this order: the input map, $N_{\mathrm{side}}$ and the output map. When increasing the resolution, the pixels inside the original pixel all get the same value as the original one; and when decreasing the resolution, the new pixel takes the average value of the original pixels inside it.

### 3.2.6 `prepCambInput.py`

This PYTHON script translates the $C(\ell)$s written by CAMB SOURCES to the output file with `scalCovCls.dat` suffix to the $C(\ell)$ files used by FLASK as input: besides partitioning the $C(\ell)$s into separate and appropriately named files (see Sec. 3.1.6), it removes the $\ell(\ell+1)/2\pi$ factors to return the pure $C(\ell)$s.

Since CAMB SOURCES output files currently have no headers, this script requires as input the fields information file created by `camb2info.py`. Therefore, its input parameters are, in this order: the CAMB SOURCES $C(\ell)$s file (with suffix `scalCovCls.dat`); the fields information file; and the prefix for the output $C(\ell)$s.

### 3.2.7 `prepClassInput.py`

This PYTHON script is the analogous of `prepCambInput.py` but for CLASS output $C(\ell)$s. There are two more differences from `prepCambInput.py`. First, since CLASS output files have headers, this script does not require a fields information file and only take two input parameters, in this order: the CLASS $C(\ell)$s file (with `cl.dat` suffix); and the prefix for the output $C(\ell)$s. Second, since CLASS outputs lensing potential $C(\ell)$s instead of convergence $C(\ell)$s, this script converts the former into the latter (and also performs a similar transformation for the cross angular power spectra) using the factors described in Hu (2000). Check the script code for more information.

### 3.2.8 `summarizeData.py`

This PYTHON script takes a list of $N$ files (which can be specified using wildcards like `*`) that must all have the same shape (basically a table with the same number of columns and rows) and headers starting with `#` (or no headers) and compute the average of each entry $x_{ij}$ over all files. For example, the table element $\bar{x}_{ij}$ in the output file is given by:

$$\bar{x}_{ij} = \frac{1}{N}\sum_{n=1}^{N} x_{n,ij}. \tag{3.3}$$

This script also computes the standard deviation in an analogous fashion. Therefore, the two outputs (the average file and the standard deviation file) have the same shape as the input. This script input parameters are, in this order: the filename for the average output; the filename for the standard deviation output; and the files to be averaged over (which can be specified using wildcards).

### 3.2.9 `ViewMap.py`

This PYTHON script plots in Mollweide projection a HEALPIX map saved as a FITS file. It just takes one input: that file. The plot can be saved in different formats.

# Bibliography

Bartelmann, M. and Schneider, P. (2001). Weak gravitational lensing. *Physics Reports*, 340:291. astro-ph/9912508.

Blas, D., Lesgourgues, J., and Tram, T. (2011). The cosmic linear anisotropy solving system (class) ii: Approximation schemes. *Journal of Cosmology and Astroparticle Physics*, 1107:034. arXiv:1104.2933.

Challinor, A. and Lewis, A. (2011). The linear power spectrum of observed source number counts. *Physical Review D*, 84:043516. arXiv:1105.5292.

Dio, E. D., Montanari, F., Lesgourgues, J., and Durrer, R. (2013). The CLASSgal code for relativistic cosmological large scale structure. *Journal of Cosmology and Astroparticle Physics*, 1311:044. arXiv:1307.1459.

Hilbert, S., Hartlap, J., and Schneider, P. (2011). Cosmic shear covariance: The log-normal approximation. *Astronomy & Astrophysics*, 536:A85. arXiv:1105.3980.

Hu, W. (2000). Weak lensing of the CMB: A harmonic approach. *Physical Review Letters D*, 62:043007. astro-ph/0001303.

Landy, S. D. and Szalay, A. S. (1993). Bias and variance of angular correlation functions. *The Astrophysical Journal*, 412(1):64. doi:10.1086/172900.

Xavier, H. S., Abdalla, F. B., and Joachimi, B. (2016). Fundamental limitations of log-normal models for cosmological fields. *in prep.*