Inpainting

Srini Raghunathan

Apr 22, 2025

CONTENTS

CHAPTER

ONE

FLATSKY MODULE

flatsky.**cl2map**(*flatskymapparams*, *cl*, *el=None*)

cl2map module - creates a flat sky map based on the flatskymap parameters and the input power spectra. Look into make_gaussian_realisation for a more general code.

Parameters

- **flatskymyapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.
- **cl** (*array*) – 1d vector of Cl power spectra: temp / pol. power spectra
- **el** (*array (optional)*) – Multipole over which the signal / noise spectra are defined. Default is None and el will be np.arange(len(cl_signal))

Returns

flatskymap – flatskymap with the given underlying power spectrum cl.

Return type

array

See also

make_gaussian_realisation

flatsky.**cl_to_cl2d**(*el*, *cl*, *flatskymapparams*, *left=0.0*, *right=0.0*)

Interpolating a 1d power spectrum (cl) defined on multipoles (el) to 2D assuming azimuthal symmetry (i.e:) isotropy.

Parameters

- **el** (*array*) – Multipoles over which the power spectrium is defined.
- **cl** (*array*) – 1d power spectrum that needs to be interpolated on the 2D grid.
- **flatskymyapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.
- **left** (*float*) – value to be used for interpolation outside of the range (lower side). default is zero.
- **right** (*float*) – value to be used for interpolation outside of the range (higher side). default is zero.

Returns
cl2d – interpolated power spectrum on the 2D grid.

Return type
array, shape is (ny, nx).

flatsky.**convert_eb_qu**(*map1*, *map2*, *flatskymapparams*, *eb_to_qu=1*)

Convert EB/QU into each other.

Parameters

- **map1** (*array*) – flatsky map of E or Q.
- **map2** (*array*) – flatsky map of B or U.
- **flatskymyapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.
- **eb_to_qu** (*bool*) – Either EB–>QU or QU–>EB. Default is EB–>QU.

Returns

- **map1_mod** (*array*) – flatsky map of E or Q.
- **map2_mod** (*array*) – flatsky map of B or U.

flatsky.**get_lpf_hpf**(*flatskymapparams*, *lmin_lmax*, *filter_type=0*)

Get 2D Fourier filters. Supports low-pass (LPF), high-pass (HPF), and band-pass (BPF) filters.

Parameters

- **flatskymyapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.
- **lmin_lmax** (*list*) – Contains lmin and lmax values for the filters. For low-pass (LPF), lmax = lmin_lmax[0]. For high-pass (HPF), lmin = lmin_lmax[0]. For band-pass (BPF), lmin, lmax = lmin_lmax.
- **filter_type** (*int*) – 0: LPF 1: HPF 2: BPF Default is LPF.

Returns
fft_filter – Requested 2D Fourier filter.

Return type
array

flatsky.**get_lxly**(*flatskymapparams*)

return lx, ly modes (kx, ky Fourier modes) for a flatsky map grid.

Parameters
flatskymyapparams (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.

Returns
lx, ly

Return type
array, shape is (ny, nx).

flatsky.**get_lxly_az_angle**(*lx*, *ly*)

azimuthal angle from lx, ly

Parameters

- **lx** (*array*) – lx modes
- **ly** (*array*) – ly modes

Returns
phi – azimuthal angle

Return type
array

flatsky.**make_gaussian_realisation**(*mapparams*, *el*, *cl*, *cl2=None*, *cl12=None*, *cltwod=None*, *tf=None*, *bl=None*, *qu_or_eb='qu'*)

Make gaussian realisation of flat sky map or 2maps based on the flatskymap parameters and the input power spectra. Look into cl2map for a simple version.

Parameters

- **flatskymyapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.
- **el** (*array*) – Multipoles over which the power spectrum is defined.
- **cl** (*array*) – 1d vector of Cl auto-power spectra for map1.
- **cl2** (*array (optional)*) – 1d vector of Cl2 auto-power spectra for map2. Default is None. Used to generate correlated maps.
- **cl12** (*array (optional)*) – 1d vector of Cl2 cross-power spectra of map1 and map2. Default is None. Used to generate correlated maps.
- **cltwod** (*array*) – 2D version of cl. Default is None. Computed using 1d vector assuming azimuthal symmetry.
- **tf** (*array*) – 2D filtering. Default is None. Used to removed filtered modes.
- **bl** (*array*) – 1d beam window function. Default is None. Used for smoothing the maps.
- **qu_or_eb** (*array*) – Generates TQU or TEB maps if cl, cl2, cl12 are supplied. Default is 'QU'.

Returns
sim_map_arr – sim_map1: T-map. if cl2 and cl12 are provided: sim_map2: Q or E map. sim_map3: U or B map.

Return type
array.

See also

cl2map

flatsky.**map2cl**(*flatskymapparams*, *flatskymap1*, *flatskymap2=None*, *binsize=None*)

map2cl module - get the auto-/cross-power spectra of map/maps

Parameters

- **flatskymyapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.
- **flatskymap1** (*array*) – flatskymap1 with dimensions (ny, nx).
- **flatskymap2** (*array (Optional)*) – flatskymap2 with dimensions (ny, nx). Default is None. If None, compute the auto-spectrum of flatskymap1. If not None, compute the cross-spectrum between flatskymap1 and flatskymap2.
- **binsize** (*int*) – el bins. computed automatically based on the fft grid spacing if None.

Returns

- **el** (*array*) – Multipoles over which the power spectrum is defined.
- **cl** (*array*) – auto/cross power spectrum.

flatsky.**radial_profile**(*z*, *xy=None*, *bin_size=1.0*, *minbin=0.0*, *maxbin=10.0*, *to_arcmins=1*, *get_errors=1*)

get the radial profile of an image (both real and fourier space). Can be used to compute radial profile of stacked profiles or 2D power spectrum.

Parameters

- **z** (*array*) – image to get the radial profile.
- **xy** (*array*) – x and y grid. Same shape as the image z. Default is None. If None, x, y = np.indices(image.shape)
- **bin_size** (*float*) – radial binning factor. default is 1.
- **minbin** (*float*) – minimum bin for radial profile default is 0.
- **maxbin** (*float*) – minimum bin for radial profile default is 10.
- **to_arcmins** (*bool*) – If set, then xy are assumed to be in degrees and multipled by 60 to convert to arcmins.
- **get_errors** (*bool*) – obtain scatter in each bin. This is not the error due to variance. Just the sample variance. Default is True.

Returns

radprf – Array with three elements cotaining radprf[:,0] = radial bins radprf[:,1] = radial binned values if get_errors: radprf[:,2] = radial bin errors.

Return type

array.

flatsky.**wiener_filter**(*flatskymyapparams*, *cl_signal*, *cl_noise*, *el=None*)

Get 2D Wiener filter.

$$W(\ell) = \frac{C_\ell^{\rm signal}}{C_\ell^{\rm signal} + C_\ell^{\rm noise}}$$

Parameters

- **flatskymyapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.
- **cl_signal** (*array*) – Power spectrum of the signal component.
- **cl_noise** (*array*) – Power spectrum of the noise component.

- **el** (*array (optional)*) – Multipole over which the signal / noise spectra are defined. Default is None and el will be np.arange(len(cl_signal))

Returns
wiener_filter – 2D Wiener filter.

Return type
array

CHAPTER

TWO

INPAINT MODULE

inpaint.**calccov**(*sim_mat*, *noofsims*, *npixels*)

Computer pixel-pixel covarinace based on simulations.

Parameters

- **sim_mat** (*array*) – nd array multiple sim maps used for covariance.
- **noofsims** (*int*) – number of simulations.
- **npixels** (*int*) – Map dimension (number of pixels).

Returns

cov – covarinace array. Same as np.cov(sim_mat).

Return type

array

inpaint.**get_covariance**(*ra_grid*, *dec_grid*, *mapparams*, *el*, *cl_dic*, *bl*, *nl_dic*, *noofsims*, *mask_radius_inner*, *mask_radius_outer*, *low_pass_cutoff=1*, *maxel_for_grad_filter=None*)

Compute the covariance between regions R1 and R2 required for inpainting.

$$\hat{T}_1 = \tilde{T}_1 + \hat{\mathbf{C}}_{12}\hat{\mathbf{C}}_{22}^{-1}(T_2 - \tilde{T}_2)$$

Parameters

- **ra_grid** (*array*) – ra_grid in degrees or arcmins for the flatsky map.
- **dec_grid** (*array*) – dec_grid in degress or arcmins for the flatsky map.
- **mapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.
- **el** (*array*) – Multipoles over which the power spectrum is defined.
- **cl_dic** (*dict*) – Signal power spectra dictionary. Keys are "TT" for T-only inpainting; "TT", "EE", "TE" for T/Q/U inpainting.
- **bl** (*array*) – 1d beam window function. Default is None. Used for smoothing the maps.
- **nl_dic** (*dict*) – Noise power spectra dictionary. Keys are "TT" for T-only inpainting; "TT", "EE", "TE" for T/Q/U inpainting.
- **noofsims** (*int*) – number of simulations used for covariance calculation.
- **mask_radius_inner** (*float*) – Inner radius of region R1 to inpaint in arcmins.
- **mask_radius_outer** (*float*) – Outer radius of region R2 to inpaint in arcmins.
- **low_pass_cutoff** (*bool*) – Low pass filter the maps before inpainting. Default is True.

- **maxel_for_grad_filter** (*int*) – lmax for the LPF above. Default is None in which it will be calculated based on the radius of the inner region.

Returns

sigma_dic – Covariance dictionary containing the covariance between R12, R22, and R22_inv. See the equation above.

Return type

dict (Optional)

See also

inpainting

inpaint.**get_mask_indices**(*ra_grid*, *dec_grid*, *mask_radius_inner*, *mask_radius_outer*, *square=0*, *in_arcmins=1*)

Get the pixel indices in regions R1 and R2 for inpainting.

Parameters

- **ra_grid** (*array*) – ra_grid in degrees or arcmins for the flatsky map.
- **dec_grid** (*array*) – dec_grid in degress or arcmins for the flatsky map.
- **mask_radius_inner** (*float*) – Inner radius of region R1 to inpaint in arcmins.
- **mask_radius_outer** (*float*) – Outer radius of region R2 to inpaint in arcmins.
- **low_pass_cutoff** (*bool*) – Low pass filter the maps before inpainting. Default is True.
- **square** (*bool*) – if True, returns a square. else, returns a cirucalr region. Default is Circle.
- **in_arcmins** (*bool*) – Supplied grid are in arcmins. default is True. If False, degrees is assumed and will be converted to arcmins.

Returns

- **inds_inner** (*array*) – pixel indices of the inner region R1.
- **inds_outer** (*array*) – pixel indices of the outer region R2.

inpaint.**inpainting**(*map_dic_to_inpaint*, *ra_grid*, *dec_grid*, *mapparams*, *el*, *cl_dic*, *bl*, *nl_dic*, *noofsims*, *mask_radius_inner*, *mask_radius_outer*, *low_pass_cutoff=1*, *maxel_for_grad_filter=None*, *intrp_r1_before_lpf=0*, *mask_inner=0*, *sigma_dic=None*, *use_original=False*, *use_cons_gau_sims=True*)

Perform inpainting. Can perform joint inpainting of T/Q/U maps using the cross-spectra between them. ###mask_inner = 1: The inner region is masked before the LPF. Might be useful in the presence of bright SZ signal at the centre.

$$\hat{T}_1 = \tilde{T}_1 + \hat{\mathbf{C}}_{12}\hat{\mathbf{C}}_{22}^{-1}(T_2 - \tilde{T}_2)$$

Parameters

- **map_dic_to_inpaint** (*dict*) – flatsky map dict to inpaint. Keys must be ["T", "Q", "U"].
- **ra_grid** (*array*) – ra_grid in degrees or arcmins for the flatsky map.
- **dec_grid** (*array*) – dec_grid in degress or arcmins for the flatsky map.
- **mapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.

- **el** (*array*) – Multipoles over which the power spectrum is defined.
- **cl_dic** (*dict*) – Signal power spectra dictionary. Keys are "TT" for T-only inpainting; "TT", "EE", "TE" for T/Q/U inpainting.
- **bl** (*array*) – 1d beam window function. Default is None. Used for smoothing the maps.
- **nl_dic** (*dict*) – Noise power spectra dictionary. Keys are "TT" for T-only inpainting; "TT", "EE", "TE" for T/Q/U inpainting.
- **noofsims** (*int*) – number of simulations used for covariance calculation.
- **mask_radius_inner** (*float*) – Inner radius of region R1 to inpaint in arcmins.
- **mask_radius_outer** (*float*) – Outer radius of region R2 to inpaint in arcmins.
- **low_pass_cutoff** (*bool*) – Low pass filter the maps before inpainting. Default is True.
- **maxel_for_grad_filter** (*int*) – lmax for the LPF above. Default is None in which it will be calculated based on the radius of the inner region.
- **intrp_r1_before_lpf** (*bool*) – Interpolate R1 before inpainting. Default is False.
- **mask_inner** (*bool*) – Mask inner region before inpainting. Default is False.
- **sigma_dic** (*dict (Optional)*) – Covariance dictionary containing the covariance between R12, R22, and R22_inv. See the equation above. If None, this will be calculated on the fly.
- **use_original** (*bool*) – Do not inpaint. Return the same map. Default is False.
- **use_cons_gau_sims** (*bool*) – Use constrained Gaussian realisations or not. If False, fields with tilde will be set to zero and there will be no randomness. (i.e:) Just interpolate R1 based on R2 and the covariance. default is True.

Returns

- **cmb_inpainted_map** (*array*) – inpainted CMB region. All pixels other than R1 will be zero in this map.
- **inpainted_map** (*array*) – inpainted map.
- **map_to_inpaint** (*array*) – original map used for inpainting.

See also

get_covariance

inpaint.**masking_for_filtering**(*ra_grid*, *dec_grid*, *mask_radius=2.0*, *taper_radius=6.0*, *apodise=True*, *in_arcmins=True*)

Mask regions before filtering. Returns an apodised (or binary) mask. Default values correspond to ACT/SPT/SO/CMB-S4-like beams.

Parameters

- **ra_grid** (*array*) – ra_grid in degrees or arcmins for the flatsky map.
- **dec_grid** (*array*) – dec_grid in degress or arcmins for the flatsky map.
- **mask_radius** (*float*) – radius for the masked region in arcmins. default is 2 arcmins.
- **taper_radius** (*float*) – raptering radius in arcmins for the mask. default is 6 arcmins.

- **apodise** (*bool*) – If True, apodises the mask. Otherwise, return a binary mask. Default is True.
- **in_arcmins** (*bool*) – Supplied grid are in arcmins. default is True. If False, degrees is assumed and will be converted to arcmins.

Returns
mask – mask corresponding to the parameter.

Return type
array, shape is ra_grid.shape.

CHAPTER

THREE

TOOLS MODULE

tools.**get_blsqinv**(*beamval*, *el*, *make_2d=0*, *mapparams=None*)

Get the inverse of the beam window function sqaured.

Parameters

- **beamval** (*float*) – Beam FWHM in arcmins.
- **el** (*array*) – Multipoles over which the window function must be defined.
- **make_2d** (*bool*) – Convert to 2D if desired. Default is False.
- **mapparams** (*list*) – [nx, ny, dx, dy] where ny, nx = flatskymap.shape; and dy, dx are the pixel resolution in arcminutes. for example: [100, 100, 0.5, 0.5] is a 50' x 50' flatskymap that has dimensions 100 x 100 with dx = dy = 0.5 arcminutes.

Returns

blsqinv – 1/Bl^2 either in 1d or 2D.

Return type

array.

tools.**get_nl**(*noiseval_in_ukarcmin*, *el*, *beamval=None*, *elknee_t=-1*, *alpha_knee=0*)

Get the noise power spectra: White and 1/f spectrum. Can return beam deconvolved nl if desired.

$$P(f) = A^2 \left[1 + \left(\frac{\ell_{\rm knee}}{\ell}\right)^{\alpha_{\rm knee}}\right].$$

Parameters

- **noiseval_in_ukarcmin** (*float*) – White noise level in uK-arcmin.
- **el** (*array*) – Multipoles over which the window function must be defined.
- **beamval** (*float*) – Beam FWHM in arcmins. Default is None. If supplied, bl^2 will be divided from nl.
- **elknee_t** (*float*) – Knee frequency for 1/f (el_knee in the above equation).
- **alpha_knee** (*float*) – Slope for 1/f (alpha_knee in the above equation).

Returns

nl – (Beam deconvoled) noise power spectrum.

Return type

array.

PYTHON MODULE INDEX

INDEX