Tools for data analyses in Cosmology

- Aula 11 -

Camila Novaes

Observatório Nacional

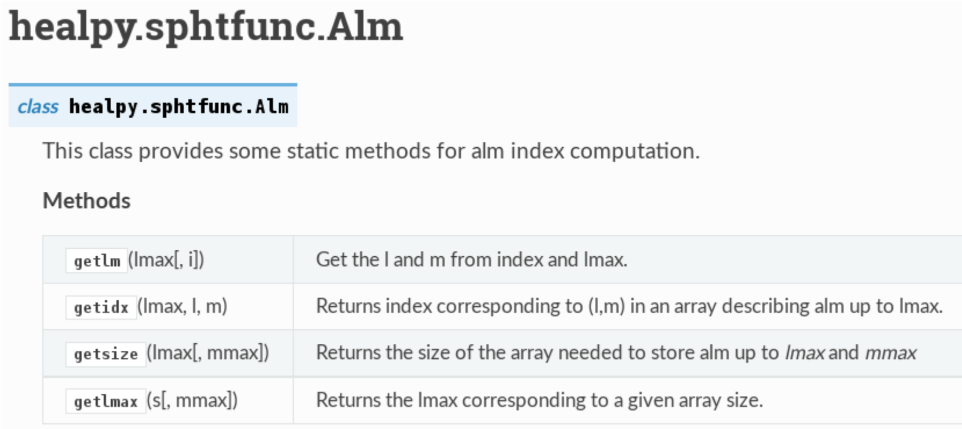
June 13, 2017

Healpy

Spherical harmonic transforms: tools

Previously: you could define $\ell_{\max}$, $m_{\max}$.

Now you can select them ...



healpy.sphtfunc.Alm

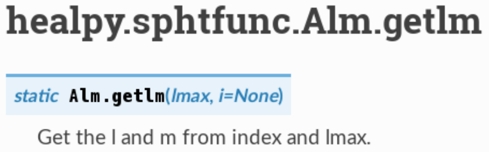
class **healpy.sphtfunc.Alm**

This class provides some static methods for alm index computation.

Methods

`getlm`(lmax[, i])	Get the l and m from index and lmax.
`getidx`(lmax, l, m)	Returns index corresponding to (l,m) in an array describing alm up to lmax.
`getsize`(lmax[, mmax])	Returns the size of the array needed to store alm up to *lmax* and *mmax*
`getlmax`(s[, mmax])	Returns the lmax corresponding to a given array size.

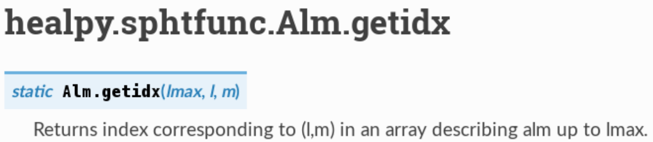
Spherical harmonic transforms: tools



healpy.sphtfunc.Alm.getlm

static **Alm.getlm**(*lmax, i=None*)

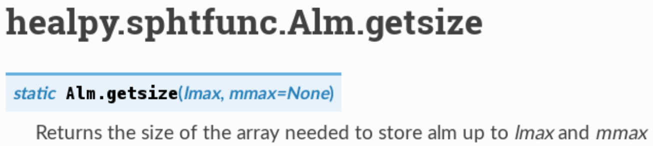
Get the l and m from index and lmax.



healpy.sphtfunc.Alm.getidx

static **Alm.getidx**(*lmax, l, m*)

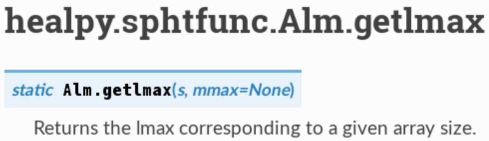
Returns index corresponding to (l,m) in an array describing alm up to lmax.



healpy.sphtfunc.Alm.getsize

static **Alm.getsize**(*lmax, mmax=None*)

Returns the size of the array needed to store alm up to *lmax* and *mmax*



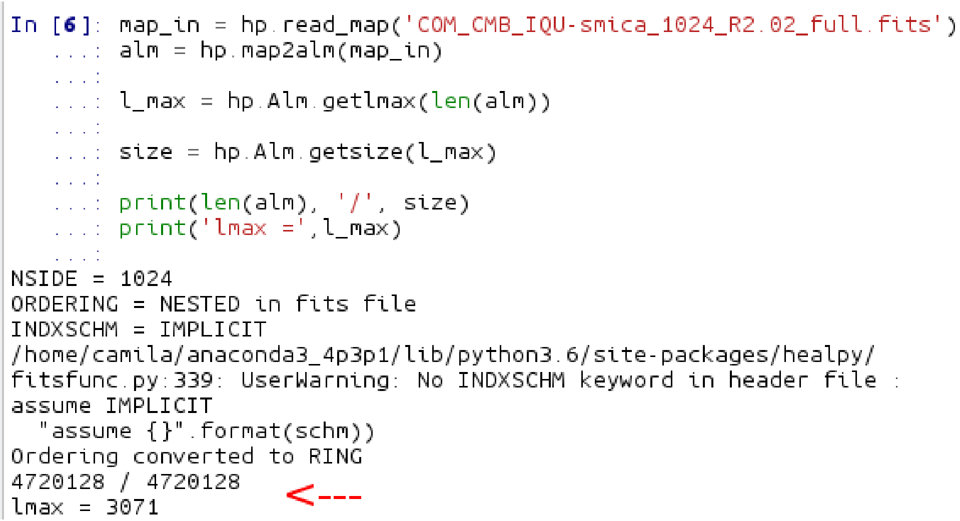
healpy.sphtfunc.Alm.getlmax

static **Alm.getlmax**(*s, mmax=None*)

Returns the lmax corresponding to a given array size.

Spherical harmonic transforms: tools

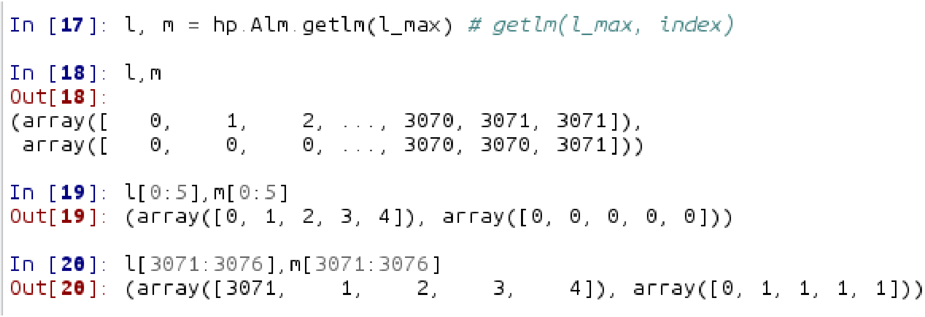
How to use:



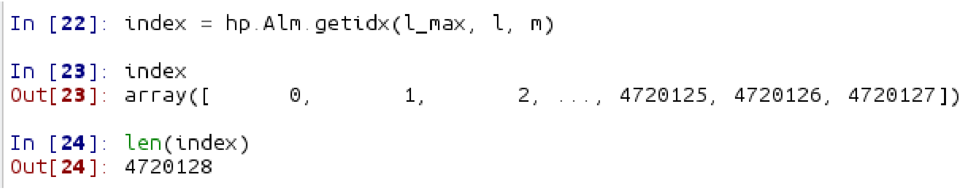
```
In [6]: map_in = hp.read_map('COM_CMB_IQU-smica_1024_R2.02_full.fits')
   ...: alm = hp.map2alm(map_in)
   ...:
   ...: l_max = hp.Alm.getlmax(len(alm))
   ...:
   ...: size = hp.Alm.getsize(l_max)
   ...:
   ...: print(len(alm), '/', size)
   ...: print('lmax =',l_max)
   ...:
NSIDE = 1024
ORDERING = NESTED in fits file
INDXSCHM = IMPLICIT
/home/camila/anaconda3_4p3p1/lib/python3.6/site-packages/healpy/
fitsfunc.py:339: UserWarning: No INDXSCHM keyword in header file :
assume IMPLICIT
  "assume {}".format(schm))
Ordering converted to RING
4720128 / 4720128      <---
lmax = 3071
```

Spherical harmonic transforms: tools

How to use:



```
In [17]: l, m = hp.Alm.getlm(l_max) # getlm(l_max, index)

In [18]: l,m
Out[18]:
(array([   0,     1,     2, ..., 3070, 3071, 3071]),
 array([   0,     0,     0, ..., 3070, 3070, 3071]))

In [19]: l[0:5],m[0:5]
Out[19]: (array([0, 1, 2, 3, 4]), array([0, 0, 0, 0, 0]))

In [20]: l[3071:3076],m[3071:3076]
Out[20]: (array([3071,    1,    2,    3,    4]), array([0, 1, 1, 1, 1]))
```



```
In [22]: index = hp.Alm.getidx(l_max, l, m)

In [23]: index
Out[23]: array([      0,       1,       2, ..., 4720125, 4720126, 4720127])

In [24]: len(index)
Out[24]: 4720128
```

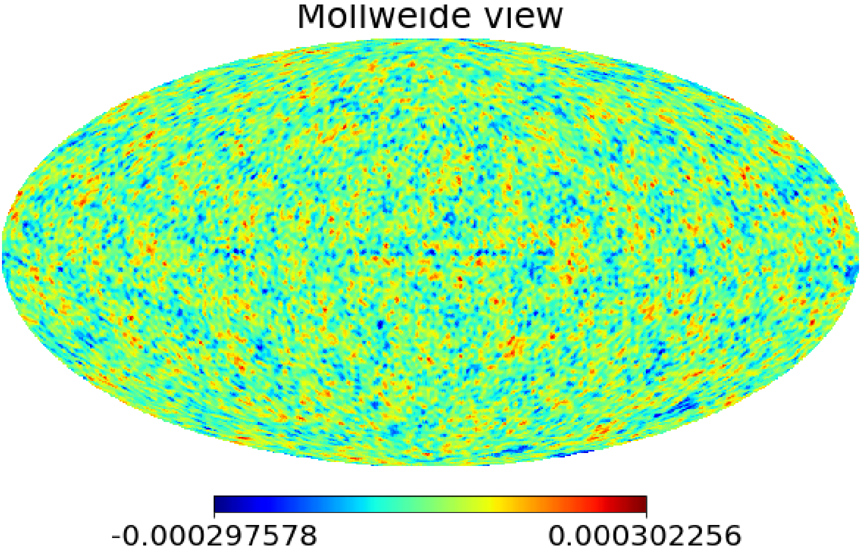
Spherical harmonic transforms: tools

Exercise:

- Reconstruct the CMB map [`'COM_CMB_IQU-smica_1024_R2.02_full.fits'`] corresponding to the range of multipole $50 \leq \ell \leq 200$.
  Steps:
  - Read the map.
  - Calculate the $a_{\ell m}$'s [`map2alm`].
  - Calculate the $\ell_{\max}$ [`Alm.getlmax`].
  - Calculate the ℓ and m for the whole $a_{\ell m}$ array [`Alm.getlm`].
  - Set to zero the $a_{\ell m}$ components for multipoles out of the chosen range [`alm[l < l_min] = (0+0j), alm[l > l_max] = (0+0j)`]
  - Rebuild the map [`alm2map`].
- Visualize it.

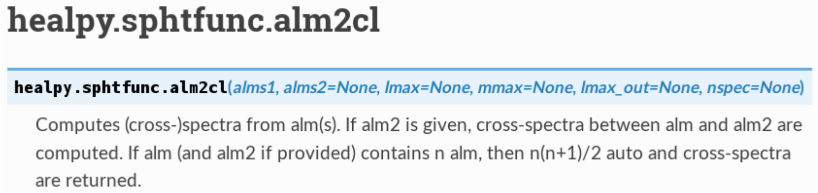
Spherical harmonic transforms: tools

Result:



Mollweide view

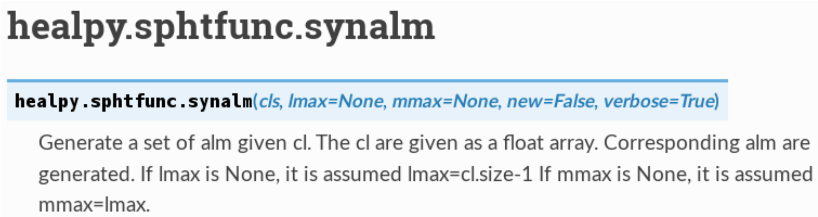
Spherical harmonic transforms: tools



healpy.sphtfunc.alm2cl

`healpy.sphtfunc.alm2cl(alms1, alms2=None, lmax=None, mmax=None, lmax_out=None, nspec=None)`

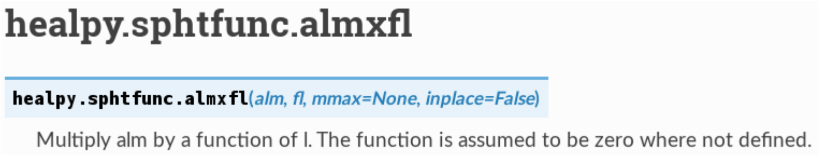
Computes (cross-)spectra from alm(s). If alm2 is given, cross-spectra between alm and alm2 are computed. If alm (and alm2 if provided) contains n alm, then n(n+1)/2 auto and cross-spectra are returned.



healpy.sphtfunc.synalm

`healpy.sphtfunc.synalm(cls, lmax=None, mmax=None, new=False, verbose=True)`

Generate a set of alm given cl. The cl are given as a float array. Corresponding alm are generated. If lmax is None, it is assumed lmax=cl.size-1 If mmax is None, it is assumed mmax=lmax.



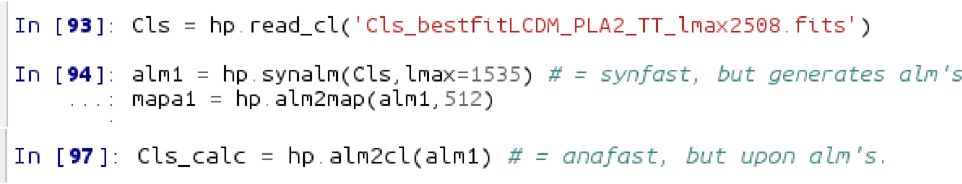
healpy.sphtfunc.almxfl

`healpy.sphtfunc.almxfl(alm, fl, mmax=None, inplace=False)`

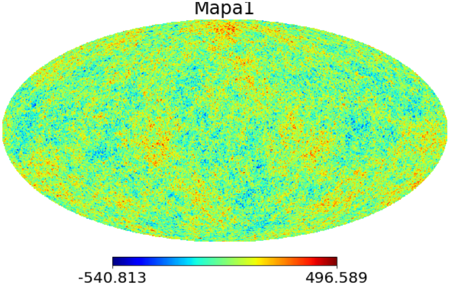
Multiply alm by a function of l. The function is assumed to be zero where not defined.

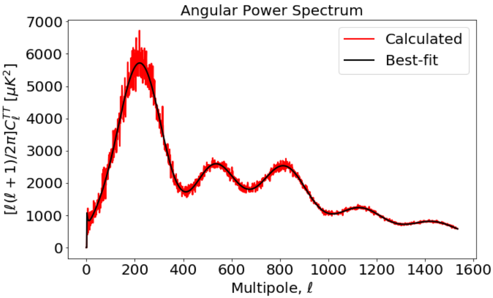
Spherical harmonic transforms: tools

How to use:



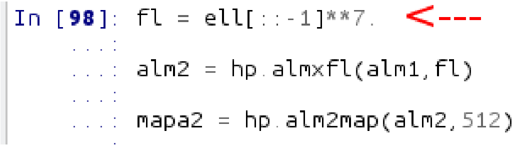
```
In [93]: Cls = hp.read_cl('Cls_bestfitLCDM_PLA2_TT_lmax2508.fits')

In [94]: alm1 = hp.synalm(Cls,lmax=1535) # = synfast, but generates alm's
    ...: mapa1 = hp.alm2map(alm1,512)
    ...:

In [97]: Cls_calc = hp.alm2cl(alm1) # = anafast, but upon alm's.
```



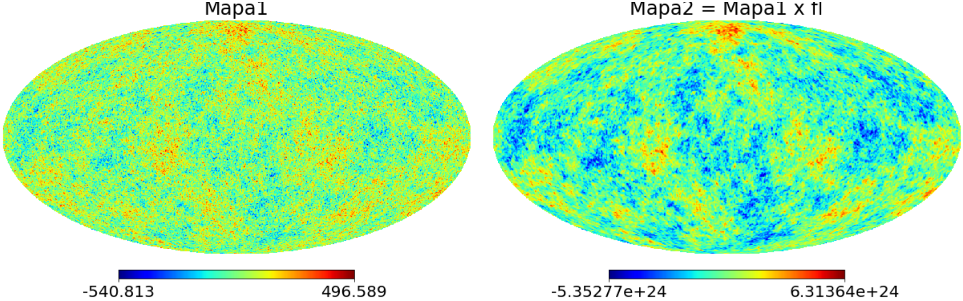


Spherical harmonic transforms: tools

How to use:

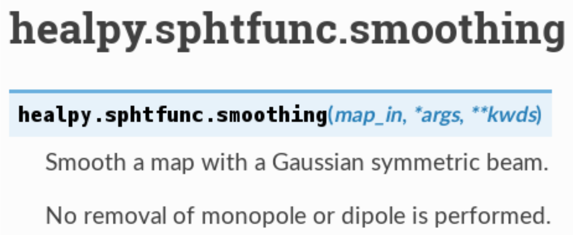


```
In [98]: fl = ell[::-1]**7.
    ....:
    ....: alm2 = hp.almxfl(alm1,fl)
    ....:
    ....: mapa2 = hp.alm2map(alm2,512)
```



⟶ Also to select a multipole range.

Spherical harmonic transforms: tools

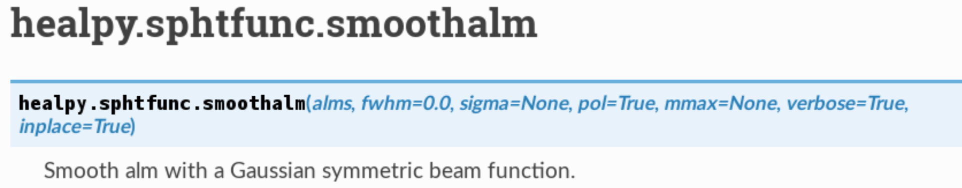


healpy.sphtfunc.smoothing

healpy.sphtfunc.smoothing(*map_in, *args, **kwds*)

Smooth a map with a Gaussian symmetric beam.

No removal of monopole or dipole is performed.



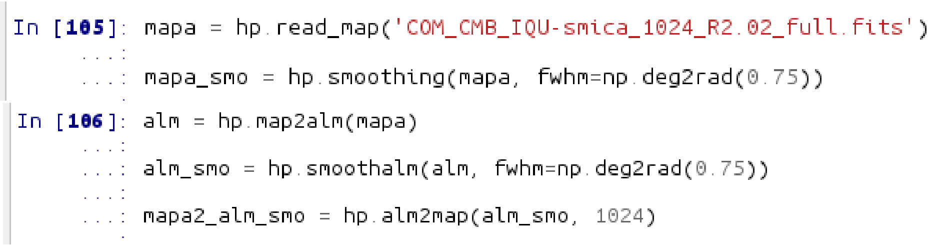
healpy.sphtfunc.smoothalm

healpy.sphtfunc.smoothalm(*alms, fwhm=0.0, sigma=None, pol=True, mmax=None, verbose=True, inplace=True*)

Smooth alm with a Gaussian symmetric beam function.

Spherical harmonic transforms: tools

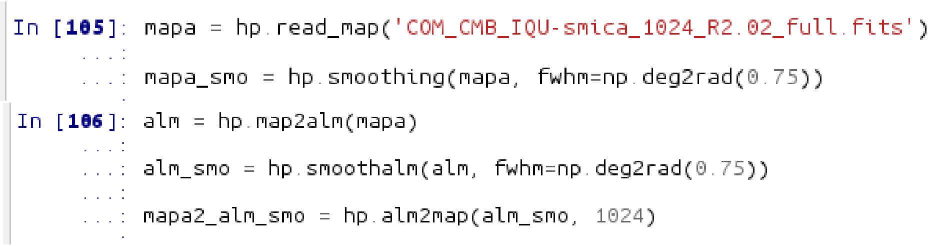
How to use:



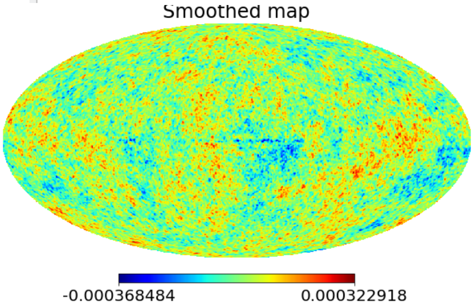
```
In [105]: mapa = hp.read_map('COM_CMB_IQU-smica_1024_R2.02_full.fits')
     ...:
     ...: mapa_smo = hp.smoothing(mapa, fwhm=np.deg2rad(0.75))
     ...:
In [106]: alm = hp.map2alm(mapa)
     ...:
     ...: alm_smo = hp.smoothalm(alm, fwhm=np.deg2rad(0.75))
     ...:
     ...: mapa2_alm_smo = hp.alm2map(alm_smo, 1024)
     ...:
```

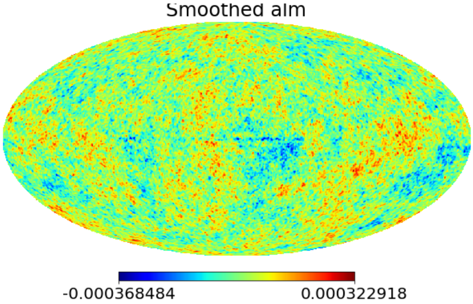
Spherical harmonic transforms: tools

How to use:



```
In [105]: mapa = hp.read_map('COM_CMB_IQU-smica_1024_R2.02_full.fits')
     ...:
     ...: mapa_smo = hp.smoothing(mapa, fwhm=np.deg2rad(0.75))
     ...:
In [106]: alm = hp.map2alm(mapa)
     ...:
     ...: alm_smo = hp.smoothalm(alm, fwhm=np.deg2rad(0.75))
     ...:
     ...: mapa2_alm_smo = hp.alm2map(alm_smo, 1024)
     ...:
```

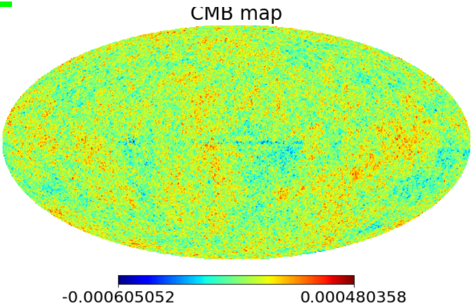


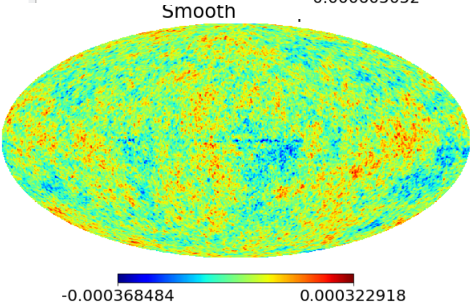


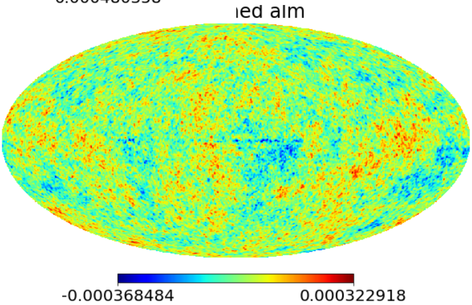
Spherical harmonic transforms: tools

How to use

```
In [105]: mapa                                                    _R2.02_full.fits')
     ...:
     ...: mapa_                                                   rad(0.75))
     ...
In [106]: alm = 
     ...:
     ...: alm_sı                                                  0.75))
     ...:
     ...: mapa2_
     ...
```

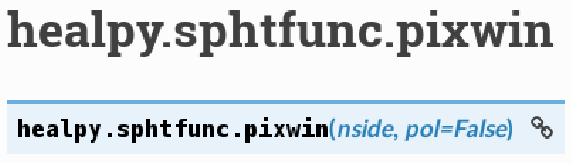






Spherical harmonic transforms: tools

$$C_\ell^{\mathrm{obs}} = B_\ell^2 C_\ell$$

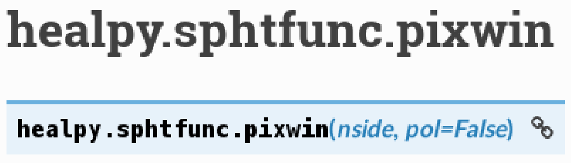


healpy.sphtfunc.pixwin

`healpy.sphtfunc.pixwin(nside, pol=False)`

Spherical harmonic transforms: tools

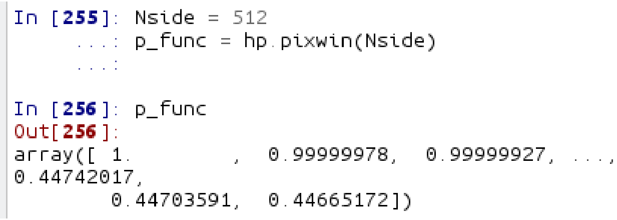
$$C_\ell^{\rm obs} = B_\ell^2 C_\ell$$



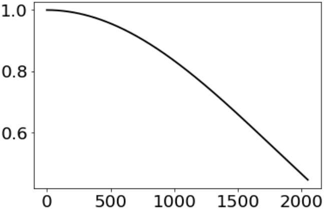
healpy.sphtfunc.pixwin

healpy.sphtfunc.pixwin(*nside*, *pol=False*)

How to use:



```
In [255]: Nside = 512
     ...: p_func = hp.pixwin(Nside)
     ...:

In [256]: p_func
Out[256]:
array([ 1.        ,  0.99999978,  0.99999927, ...,
0.44742017,
        0.44703591,  0.44665172])
```



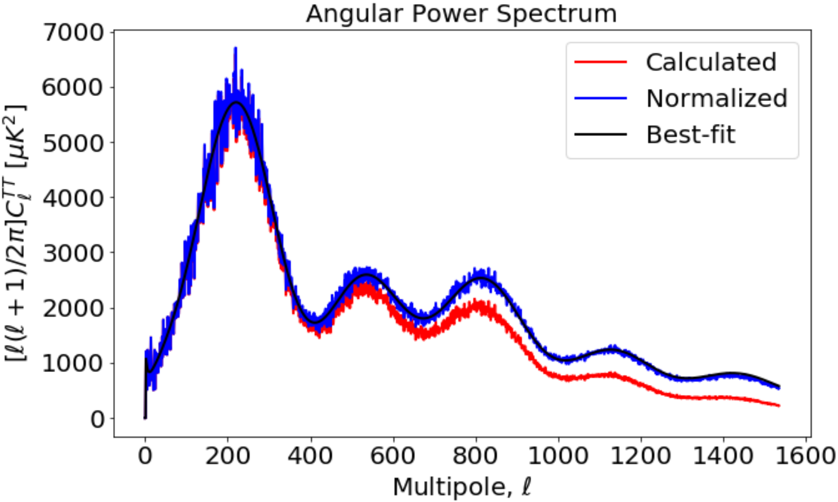
Spherical harmonic transforms: tools

Exercise:

- Use the pixwin function to correct the angular power spectrum calculated from a map generated considering the pixel window function.
  Steps:
  - Read the Planck best-fit C_ℓ's.
  - Generate a map (Nside = 512) including the pixel window effect [pixwin = True].
  - Calculate its C_ℓ's [anafast].
  - Calculate the pixel window function [pixwin].
  - Use the result to correct the calculated C_ℓ's [$C_\ell^{\mathrm{obs}} = B_\ell^2 C_\ell$].
  - Compare them in a Plot.

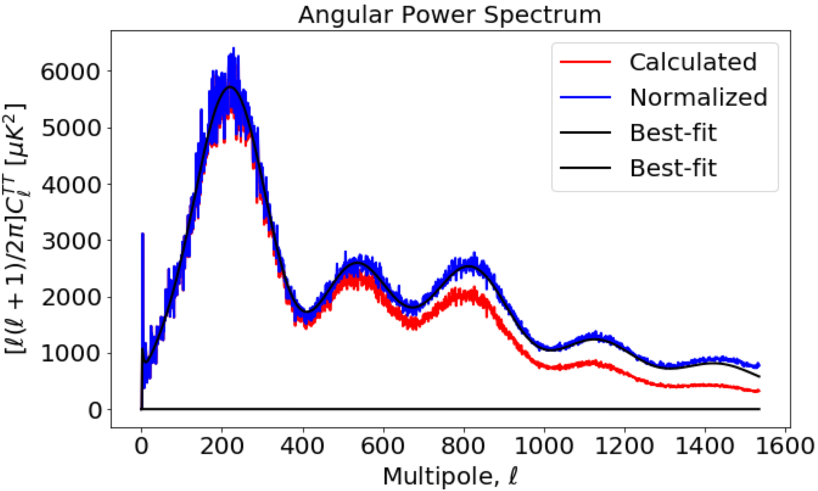
Spherical harmonic transforms: tools

Result:



Spherical harmonic transforms: tools

Result: We can also correct the effect of degrading a map. The example shows the C_ℓ's of a map degraded from `Nside = 2048` to 512.


Angular Power Spectrum
Calculated
Normalized
Best-fit
Best-fit
$[\ell(\ell+1)/2\pi]C_\ell^{TT}$ [μK^2]
Multipole, ℓ