HANDS-ON ON PLANCK DATA

Tuhin Ghosh

August 27, 2021

1 Healpy (Python version of HEALPix)

Healpix software is commonly used in the field of Cosmology.
Detail documentation of healpy is available on http://healpy.readthedocs.org/en/latest/tutorial.html.

1.1 Call the Healpy function

```
import matplotlib.pyplot as plt
import healpy as hp
import numpy as np

# To get interactive plotting

plt.ion()
```

1.2 Visualization and handling of full-sky Planck data

In this part, you will learn how to visualize the Planck full sky maps and how to use the Healpy basic tools to change the maps resolution and pixelization.

- The first exercise is to visualize the full sky Planck data using the mollview procedure of Healpy. Here is an example Python script to read and visualize the HFI 100 GHz full sky map in $\mathrm{K_{CMB}}$:

  ```
  dir_maps = ‘/mymaps/directory/’
  m100=hp.read_map(dir_maps+‘HFI_SkyMap_100-field-IQU_2048_R3.00_full.fits’,field=(0,1,2))
  ```

  then do :

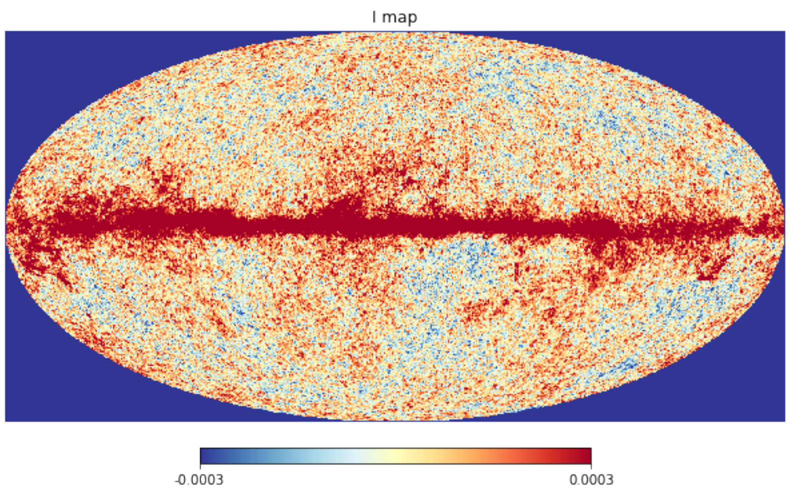
  ```
  np.shape(m100)
  (3, 50331648)
  ```

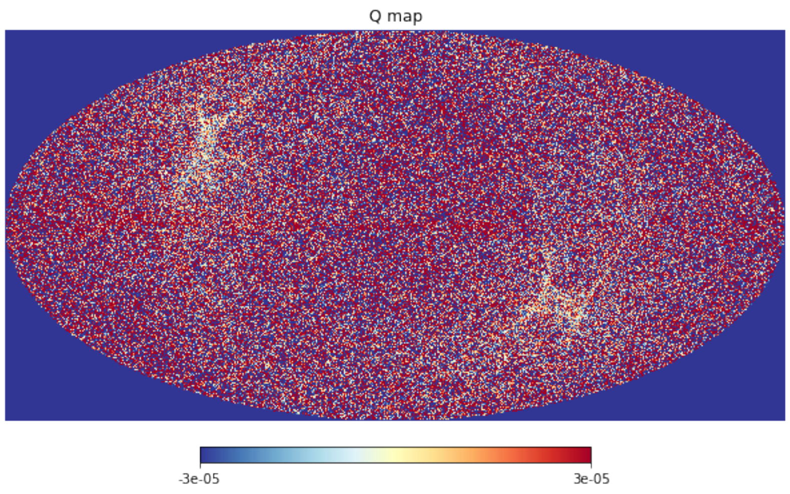
  To read the map header type, type the following

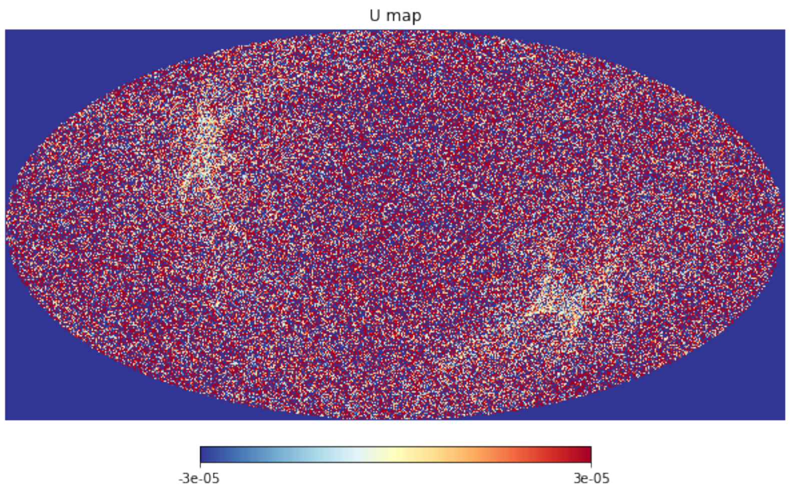
  ```
  m100, header=hp.read_map(dir_maps+‘HFI_SkyMap_100-field-IQU_2048_R3.00_full.fits’,
  field=(0,1,2),h=True)

  print dict(header)
  ```

  You will see that the map `m100` contains in fact 3 maps of 50331648 elements ($N_{\rm pix} = 12N_{\rm side}^2$). The first one is the intensity I map, the second is the Stokes Q map and the the last one is Stokes U map. To visualize the three full sky maps you can do :

  ```
  hp.mollview(m100[0],cbar=True,nest=True,min=-3e-4,max=3e-4,title=‘I map’,cmap=‘RdYlBu_r’)
  hp.mollview(m100[1],cbar=True,nest=True,min=-3e-5,max=3e-5,title=‘Q map’,cmap=‘RdYlBu_r’)
  hp.mollview(m100[2],cbar=True,nest=True,min=-3e-5,max=3e-5,title=‘U map’,cmap=‘RdYlBu_r’)
  hp.graticule()
  ```







As an example, the following window is opened :

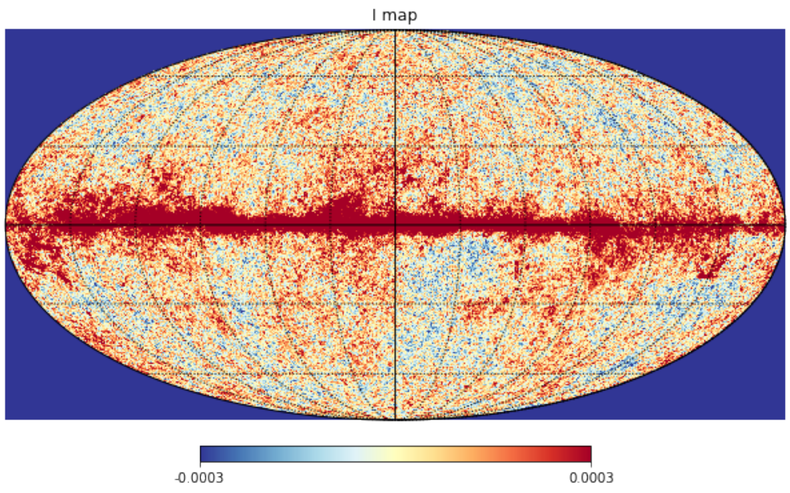
The nest=True keyword has to be set because the Planck maps from the Planck Legacy Archive are in nested HEALPix ordering. You might try mollview with keyword nest=False and see what happens. For the following of the exercise, we will focus only on the intensity maps.

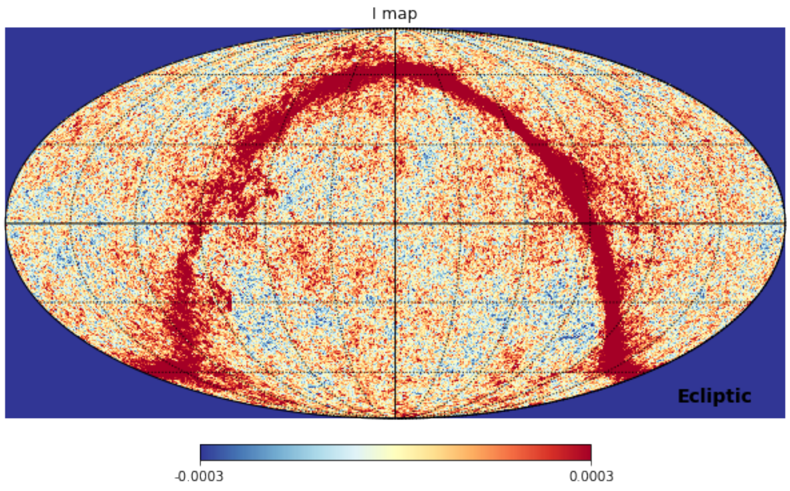
- To learn all the keywords of hp.mollview, type

```
hp.mollview?
or ?hp.mollview
```

- You can change the coordinate system of the input Planck map from Galactic to Celestial coordinates using the following commands.

```
hp.mollview(m100[0],cbar=True,nest=True,min=-3e-4,max=3e-4,title='I map',cmap='RdYlBu_r')
hp.graticule()
hp.mollview(m100[0],cbar=True,nest=True,min=-3e-4,max=3e-4,title='I map',coord=['G','E']
,cmap='RdYlBu_r')
hp.graticule()
```





- You will now open and visualize the intensity field of all the nine frequency maps (to start with with the same intensity range as before for the 100 GHz) and see how the different components (CMB, dust, synchrotron, free-free, CO, AME, ...) behave. For high frequency maps, you will see that it saturates and you will have to change the range accordingly. You may also try the norm=hist keyword for mollview to visualize the maps in histogram equalized mode to enhance the contrast.
- Then you can try to look at differences between maps to get rid of the CMB emission and to identify the other components. When doing so, you will see that the maps are not sharing a common resolution.
- For the following and in order to do component separation, the maps have to be reduced to a common resolution. For this, you have to use the ismoothing tool of HEALPix. For smoothing the intensity field of the m100 map at a resolution out_beam to a map named map2 at a 10 arcmin resolution the syntax is the following :

  ```
  sm100=hp.smoothing(m100,fwhm=np.radians(np.sqrt((out_beam/60.)**2-(10./60.)**2)),lmax=2048)
  ```

  You can smooth for example the 100 and 143 GHz maps to a one degree resolution and then compare at high Galactic latitudes the differences of the 1 degree maps and of the original resolutions maps to see how the noise level changes.
- Finally, you may have noticed that the LFI and HFI maps don't have the same number of elements. This is because they don't have the same pixel size ($N_{\rm side}$ = 1024 for LFI, corresponding to a pixel size of ~ 3.4

arcmin and $N_{\rm side} = 2048$ for HFI, corresponding to a pixel size of ~ 1.7 arcmin). To change the HEALPix pixel size of map1 at nside to the one of map2 at nside2, you have to use the hp.ud_grade command :

```
ud_sm100  = hp.ud_grade(sm100, nside_out=1024, order_in='RING')
```

You may try the same exercise than with the smoothing to see how the noise level changes, for example from $N_{\rm side} = 2048$ to $N_{\rm side} = 256$.

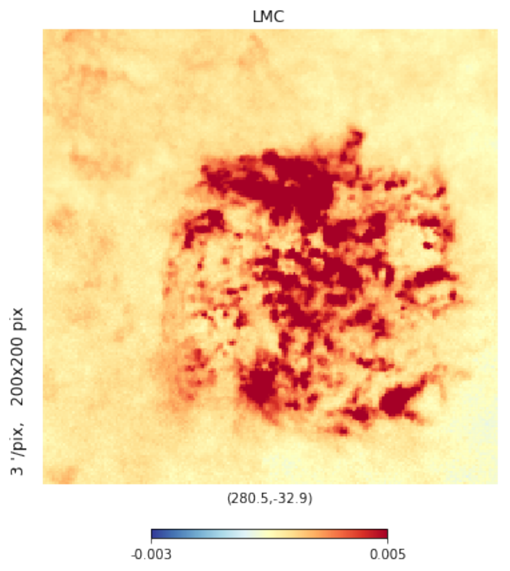
1.3 Visualization and handling of Planck data on patches

In this part you will learn how to visualize the Planck maps on specific patches and how to recover a python array containing the square map.

- To visualize patches of the sky, you will use the gnomview HEALPix command. An example to look at the Large Magellanic Cloud(LMC) in the HFI 353 GHz map is given below :

  ```
  gLMC=hp.gnomview(m353,rot=[280.5,-32.9],reso=3,min=-3e-3,max=5e-3,title='LMC',cmap='RdYlBu_r')
  ```

  The keyword rot contains the Galactic longitude and latitude of the center of the patch, the keyword reso the pixel size in arcmin of the projected map and xsize is the number of pixels in the x-axis (if no ysize keyword, pxsize = pysize=200 by default). The map size is thus (reso x pxsize)^2 = 10deg $\times$ 10deg = 100 deg^2. Finally, the keyword gLMC returns a python array with the 200×200 pixel map. The following window opens :



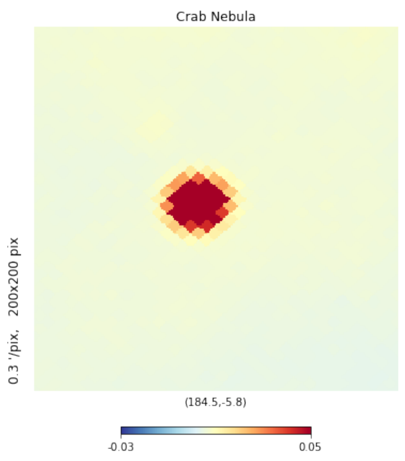
- Now have a look to several peculiar sources of the sky at the different frequencies, to see how the source itself varies with respect to the other components. You can do that for the LMC satellite Galaxy (G_{lon} = 280.5, $G_{lat} = -32.9$), the Crab nebula ($G_{lon} = 184.5$, $G_{lat} = -5.8$), the Taurus molecular clouds (rich in CO) ($G_{lon} = 173.0$, $G_{lat} = -15.2$), the Andromeda Galaxy (M31) ($G_{lon} = 121.2$, $G_{lat} = -21.6$), etc. . .

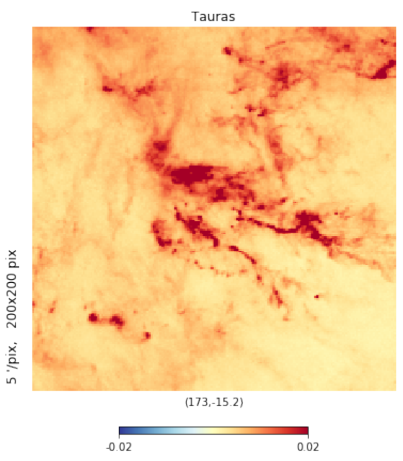
- For some of these sources (e.g. M31), the object is very hard to see because of the CMB at low frequencies. Try to look at M31 by launching gnomview for map differences to get rid of the CMB component.

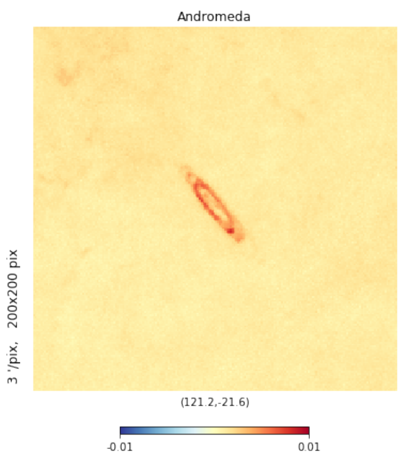
1.4 Component separation to isolate the CMB emission

Now we will implement a very simple component separation method to extract the CMB component out of the Planck data. It is called Internal Linear Combination (ILC) and finds the CMB by co-adding the data at several frequencies with a set of weights that minimizes the final variance of the map, assuming that the CMB has a flat frequency dependence.

- If you feel comfortable with coding you can try to code in python for the ILC following the [Eriksen et al. 2004] paper. There is basically just one equation to code, which is equation (7). If you prefer or if you don't manage, you can ask me.





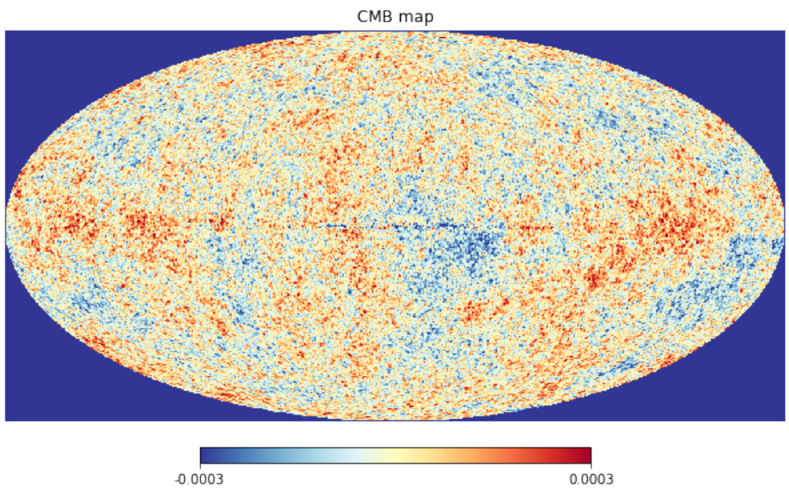


- Then, to apply the ILC method to the data, remember that the ILC linearly combines maps and as a consequence they must share the same resolution and same pixelization. To perform the ILC, we will use the Planck maps from 70 to 353 GHz, so you will have to smooth all the maps to the resolution of the 70 GHz channel which has the largest beam. You can find each channel resolution in [The Planck Collaboration, Planck results I, 2018]. Be careful in using only the intensity field of the maps. Then you have to degrade the HFI maps pixelization to the HEALPix N_{side} of LFI. You have now five maps sharing the same resolution and pixelization that you will be able to combine.

- Then apply the ILC code to an array containing the five intensity maps to get the CMB emission. The input array must be of the form $[N_{pix}, N_{freq}]$. To form that array from your five maps, you can do :

  ```
  maps[:,0] = m70, maps[:,1] = m100, maps[:,2] = m143, maps[:,3] = m217, maps[:,4] = m353
  ```

  Then, launch the ILC. You should see the following map :

  You can see that even if the extraction is not perfect in the Galactic plane, most of the Galactic emission has been removed.

- The ILC weights were computed on the full sky. This might not be the the optimal way to compute them, since the weights might be driven by few very bright pixels in the Galactic center that contribute very significantly to the map variance. In order to test this, you can try to use the ILC algorithm with the mask keyword. For building masks, as Planck maps most significant foreground from 70 to 353 GHz is the thermal dust emission, you can use the 353 GHz map as a tracer. On this map, you can choose a threshold called cut above which the sky will be masked. Once chosen, you can build a mask map like this :

  ```
    mask = m353 * 0.
  ```

```
ind=np.where(ud353 < 3.e-3)
nind=np.size(ind)
mask[ind] = 1.
```

Try several threshold to mask the Galactic plane and use the resulting mask for the ILC. See how the extracted CMB component changes with respect to the masking.

- Use one of the CMB maps you have produced to produce CMB-removed Planck maps and have again a look at the sources (especially the ones that were difficult to see because of the CMB, like M31) at the different frequencies.

1.5 CMB angular power spectrum estimation

- Finally, you have to compute the angular power spectrum of the CMB intensity anisotropies. To compute the power spectrum from a map, you can use the hp.anafast Healpy routine. The syntax is the following :

```
cl= hp.anafast(map_ilc, lmax=2048)
```

cl will then contain the angular power spectrum C_ℓ^{TT} of the map, as a function of the multipole ℓ. To plot it, type the following :

```
ell=np.arange(2049)
p1 =plt.loglog(ell,ell*(ell+1)*cl*1.e12/(2.*np.pi),linestyle='solid',color='b',label='ILC map')
plt.xlim(xmin=3,xmax=2000.)
plt.ylim(ymin=200,ymax=7000)
plt.xlabel(r'Multipole, $\ell$')
plt.ylabel(r'$\ell (\ell+1) C_{\ell}/2\pi$ [in $\mu K^2$]')
plt.legend()
plt.show()
```

This is just an example plot, but you can try to plot it in linear scale instead of logarithmic.

- You can also overplot the Planck best fit C_ℓ^{TT}. It is a file that was provided to you and that you can read and overplot like that :

```
ell_best, dl_best = np.loadtxt('COM_PowerSpect_CMB-TT-full_R3.01.txt',usecols=(0,1),
unpack=True,skiprows=1)
p2 =plt.loglog(ell_best, dl_best, linestyle='solid', color='k', label = 'Planck best fit')
```

You obtain the following figure :

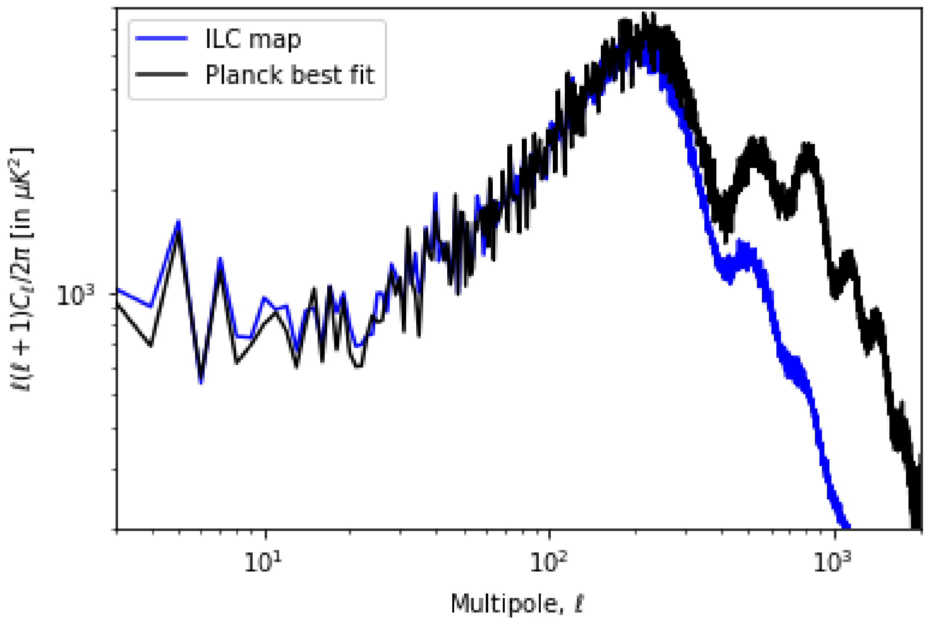
You can see that at low ℓ (large angular scales) the ILC estimate of the CMB is very close to the Planck best fit C_ℓ. It means that to the first order, the ILC removes most of the foreground emission that should dominate at large scales. At higher ℓ, you can observe a departure from the best fit model. This is due to the beam transfer function that reduces the power towards the smallest scales. The rise observed at very high ℓ is due to the noise power in the ILC map. Both effect can be accounted for produce the agreement in figures you might have seen in the Planck papers and press releases.

- You can finally try to compute the power spectra from the raw Planck maps (e.g. 100, 143 and 217 GHz) with different maskings, using the mask generated at 353 GHz. See how the foregrounds impact the CMB power spectrum at low ℓ.

1.6 Masked angular power spectrum estimation using Polspice

In this section, we will use publicly available package called Polspice to compute the power spectrum over the masked sky. The computation of cutsky power spectrum to fullsky power spectrum was discussed in Hivon et al. 2001 paper. The main result of the paper is equation number A20 and A21.

- To initiate spice command within python, you need to type the following command.

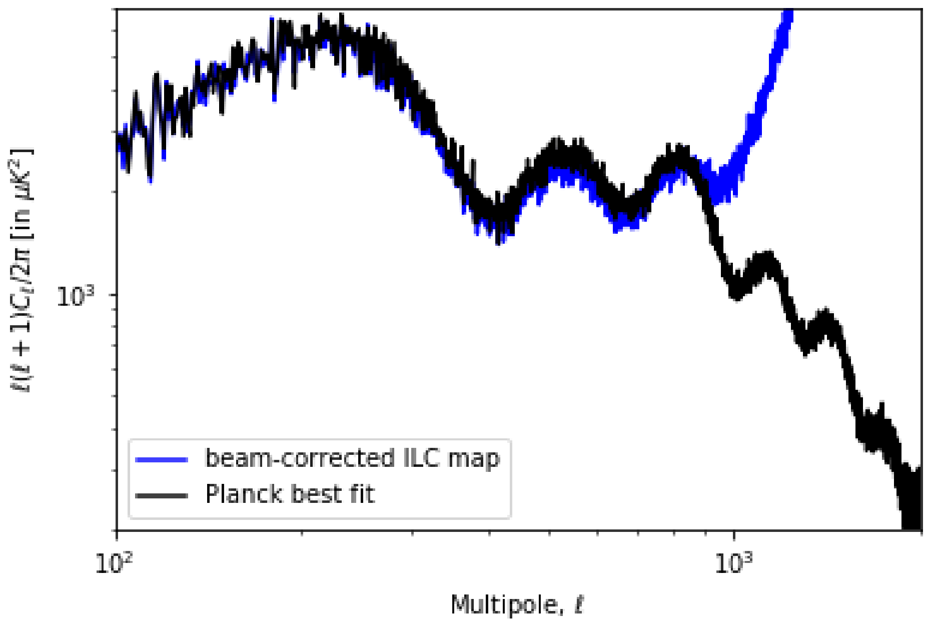


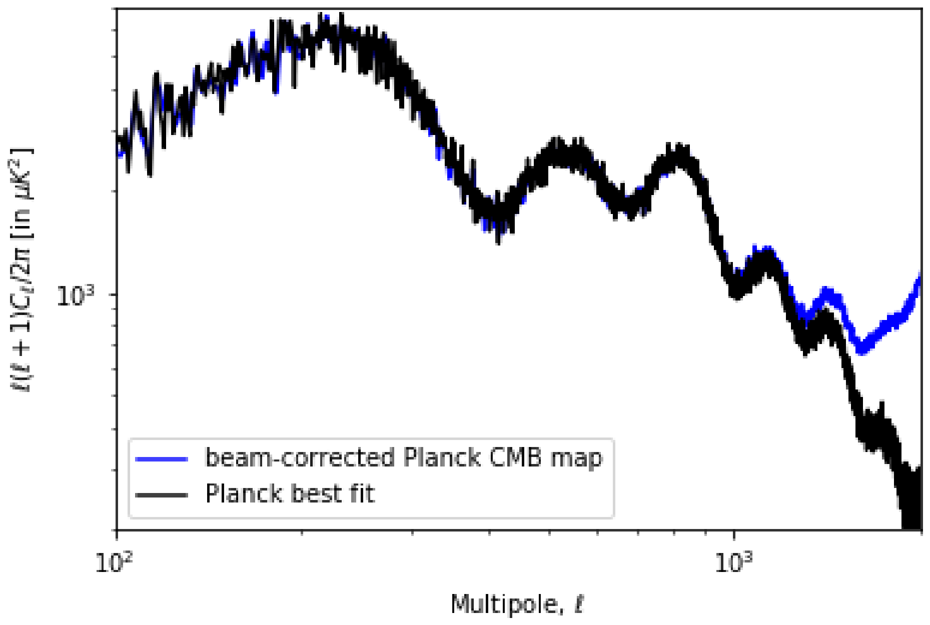
```
import sys
sys.path.append('mydir/PolSpice_v03-01-06/src/')
import ispice
```

- The syntax to call Polspice to compute the full-sky angular power spectra corrected from the galactic mask and pixel window is the following

```
ispice.ispice('ilc_map.fits', 'cls_tmp.fits', nlmax=2048, maskfile1='mask.fits',beam1=15.,
label=''spice", polarization='NO')

tmp=hp.read_cl('cls_tmp.fits')
dl= ell*(ell+1)*tmp/(2.*np.pi)
```

- Next plot the power spectrum computed from ILC map and see how close it is to the Planck best-fit C_ℓ. Alternatively, you can download the CMB data from the Planck legacy achieve and the mask used by the Planck collaboration in their 2013 analysis. Run Polspice command and see how the power spectrum computed from the CMB map matches with the Planck best-fit C_ℓ. If you do things correctly, here is the plot that you expect to see.


$\ell(\ell+1)C_\ell/2\pi$ [in μK^2]
10^3
10^2
10^3
Multipole, ℓ
beam-corrected ILC map
Planck best fit


$\ell(\ell+1)C_\ell/2\pi$ [in μK^2]
10^3
10^2
10^3
Multipole, ℓ
beam-corrected Planck CMB map
Planck best fit