OBSERVATIONAL COSMOLOGY

Exercise 7 – ILC application
Due: 18.6.2019

Aim: In this exercise you will create an all-sky CMB map using the actual WMAP satellite data, based on the Internal Linear Combination (ILC) method. The channel maps at five different frequencies are provided to you in the form of FITS files. You are also given part of a code, as a jupyter notebook, with `astropy` modules to read and write FITS files as well as examples of how to read and view these specific WMAP data files. Your job is to code the couple of necessary steps for the ILC method to create an all-sky CMB map from these data.

Method: Please refer to the pages 68 – 70 of the 3rd CMB lecture slides for details of the ILC method. In summary, the channel maps represent a linear superposition of all the astrophysical signal and noise. Let $s(p)$ be the astrophysical signal you are interested in (in this case CMB), with the spectrum a_i. Here p is the pixel index and i denotes the i-th frequency channel. Then each channel map can be written as

$$T_i(p) = a_i s(p) + n_i(p), \tag{0.1}$$

where the noise term $n_i(p)$ contains instrumental noise and all other astrophysical signal that we are not interested in (together grouped as the "nuisance term"). The vector $\vec{a}$ is called the mixing vector.

The ILC method returns an estimate of the sought-after signal map, denoted as $s_{\rm ILC}(p)$, by computing specific weights for each frequency channel:

$$\vec{\omega} = \frac{C^{-1}\vec{a}}{\vec{a}^T C^{-1}\vec{a}} \tag{0.2}$$

$$s_{\rm ILC}(p) = \sum_i \omega_i T_i \tag{0.3}$$

(note the vector notation in the first equation). Here C is the channel-to-channel (i.e. frequency-to-frequency) covariance matrix, defined as

$$C_{i,j} = COV(T_i, T_j) = \frac{1}{N_{\rm pix}} \sum_{p=1}^{N_{\rm pix}} (T_i(p) - \bar{T})(T_j(p) - \bar{T}) \tag{0.4}$$

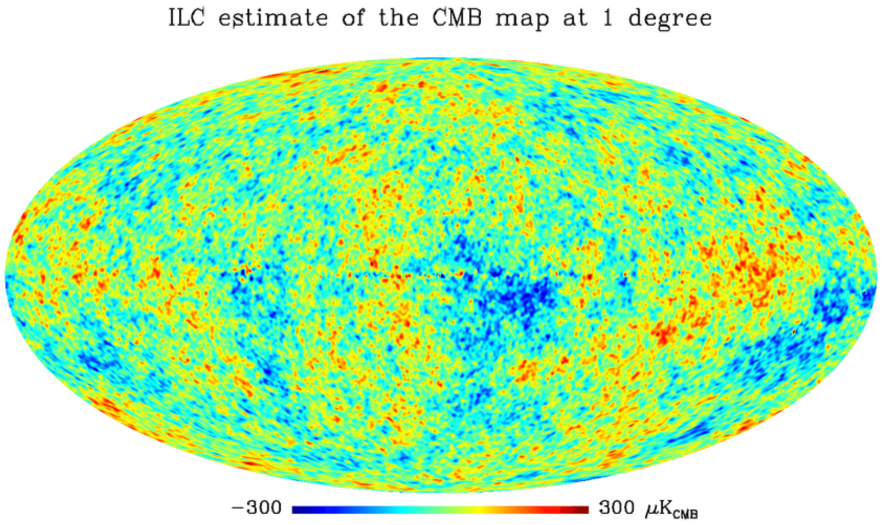
In the ILC method we compute these weights (denoted ω_i) by setting the minimum-variance condition for the nuisance maps *and* requiring unit response for the component of interest to preserve its signal

$$\frac{\partial}{\partial \omega_i}\left(\vec{\omega}^T C \vec{\omega}\right) = 0, \qquad \vec{\omega}^T \vec{a} = 1. \tag{0.5}$$

You can download the data files (in a compressed gzipped format) from this link:
https://www.dropbox.com/s/4p2iyc1e3nb3kr9/wmap_mollweide_channel_maps.tar.gz?dl=0

They are originally from this website (we need only the mollweide projected data files):
https://lambda.gsfc.nasa.gov/product/map/dr3/maps_band_smth_r9_i_5yr_get.cfm

Result: Your aim is to make the map $s_{\mathrm{ILC}}(p)$ by computing the channel weights. You will be surprised, after coding only these few simple steps, how similar your map will look as compared to this "official" result shown below from the WMAP team, created from their 9-year full-mission data (with some minor additional improvements to the ILC technique)!



Hints:

- The maps are all in temperature units (mK) and smoothed to the same angular resolution of 1°. The central frequencies for WMAP channels are 22.0, 25.9, 32.5, 49.1, and 90.1 GHz (called K, Ka, Q, V, and W bands, respectively).
- Think carefully about the spectral shape of the CMB signal that will form the mixing vector! If the maps were in the units of specific intensity (MJy/steradian), then you would have needed to put in the shape of the CMB blackbody spectrum as function of frequency. But here the maps are all in CMB temperature units. More specifically, these maps show the temperature *fluctuations* in units of mK, where the mean CMB temperature is $T_0 = 2.7255$ K.
- For computing the covariance matrix, it is easiest to use the `numpy` function `cov`. But you will need to format the input data matrix appropriately.

ILC references:

1. J. Delabrouille and J.-F. Cardoso, *Diffuse source separation in CMB observations*, arXiv:astro-ph/0702198
2. M. Remazeilles, J. Delabrouille and J.-F. Cardoso, *CMB and SZ effect separation with Constrained Internal Linear Combinations*, arXiv:1006.5599

In [1]:

```
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
def readfits(file_name):
    '''Reads fits files and returns data array and header.
       Note: we have specified to read only the second extension of the fits
    Parameters
   ----------
   file_name: string
        Name of the fits file    Returns
   -------
    data: float array
        Data array extracted from the fits file
    header: fits header
        Header of the fits file
    '''
    hdulist=fits.open(file_name)
    #hdulist.info()
    data = hdulist[1].data
    header = hdulist[1].header
    hdulist.close()
    return(data, header)

def writefits(file_name, data, header=None):
    '''Writes a 2D array (usually an image) to a fits file.
    Parameters
   ----------
   file_name: string
        Name of the fits file
   data: float array
        2D array to be written
   header: fits header
        Optional fits header. A minimal header is created
       automatically if none is provided. Default: None
   -------
    '''
    hdu = fits.PrimaryHDU()
    hdu.data = np.array(data, dtype=np.float32)
    if header is not None:
        hdu.header = header
    hdu.writeto(file_name, overwrite=True)
    return(None)
```

In [ ]:

```
## Read all the channel maps and make a data array (map units mK)

freq_ghz = [22.0,25.9,32.5,49.1,90.1]
bandname = ['K','Ka','Q','V','W']
data = []
map_path = '/Location-of-files-in-your-computer/'

fig = plt.figure(figsize=(9,11), dpi=150)
for i in np.arange(5):
    plt.subplot(3,2,i+1)
    image, header = readfits(map_path+'wmap_mollweide_band_smth_imap_r9_5yr_{0}
_v3.fits'.format(bandname[i]))
    plt.imshow(image, cmap="rainbow", vmin=-0.4, vmax=0.4, origin="lower")
    plt.colorbar(label=r'mK$_\mathrm{CMB}$', orientation='horizontal', aspec
t=40)
    plt.title(bandname[i] + "-band")

    data.append(image)

data = np.array(data)
data.shape # shape of the data array
```