Documentation

Anne Hutter*

Kapteyn Astronomical Institute, University of Groningen
a.k.hutter@rug.nl

Abstract

1. Fast Fourier Transformation Conventions

1.1. FFTW

The software package FFTW computes the Fourier transformation as follows: the forward transformation (*FFT*), i.e. real to k-space, is defined as

$$FFT[r] = F_m = \sum_{n=0}^{N_{\rm side}-1} r_n \, e^{-2\pi\sqrt{-1}nm/N_{\rm side}}, \quad (1)$$

while the backward transformation (*IFFT*), i.e. k- to real space, is defined as

$$IFFT[F] = r_n = \sum_{m=0}^{N_{\rm side}-1} F_m \, e^{2\pi\sqrt{-1}nm/N_{\rm side}}. \quad (2)$$

We note that this definition leads to $IFFT[FFT[r]] = N_{\rm side} r$ and not r.

1.2. NUMPY

For the FFTW defintion to be consistent with the *FFT* implementation in numpy, the backward Fourier transformation *IFFT* needs to be multiplied with $1/N_{\rm side}$, yielding

$$r_n = \frac{1}{N_{\rm side}} \sum_{m=0}^{N_{\rm side}-1} F_m \, e^{2\pi\sqrt{-1}nm/N_{\rm side}}. \quad (3)$$

2. Computing polyspectra

In the following we adopt the FFTW definition and call the forward and backward Fourier transformation *FFT* and *IFFT* respectively.

The polyspectrum is given as

$$\mathscr{P}(\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_p) \approx \frac{1}{V}\left(\frac{V}{N_{\rm pix}}\right)^p \frac{\sum_n^{N_{\rm pix}} \prod_{i=1}^{p} \mathscr{D}(\mathbf{n}, \mathbf{k}_i)}{\sum_n^{N_{\rm pix}} \prod_{i=1}^{p} \mathscr{I}(\mathbf{n}, \mathbf{k}_i)} \quad (4)$$

whereas the functions $\mathscr{F}$ and $\mathscr{I}$ are given as

$$\mathscr{D}(\mathbf{n}, \mathbf{k}_i) = \sum_{\mathbf{l}_i \pm s/2} FFT[\delta](\mathbf{m}_i) \, e^{i2\pi \mathbf{n}\mathbf{m}_i/N_{\rm side}} \quad (5)$$

$$\mathscr{I}(\mathbf{n}, \mathbf{k}_i) = \sum_{\mathbf{l}_i \pm s/2} e^{i2\pi \mathbf{n}\mathbf{m}_i/N_{\rm side}}. \quad (6)$$

$N_{\rm pix}$ are all cells, i.e. for a 3D grid this would be $N_{\rm side}^3$. $\mathbf{l}_i = |(\mathbf{k}_i/k_F) - \mathbf{m}_i|$ represent the deviation between the exact value of $\mathbf{k}_i$ and the binned value $k_F \mathbf{m}_i$. s represents the binwidth and ideally should be kept to the width of a pixel.

Computing $\mathscr{I}$: In practise we can compute $\mathscr{D}$ and $\mathscr{I}$ as follows: We identify all pixels that fulfill the criterion $\mathbf{k}_i/k_f \simeq \mathbf{m}_i$, and generate a filter $\mathscr{F}(\mathbf{m}_i)$ that is 1 when the criterion is fulfilled and 0 otherwise. Since we want all triangles that fulfill $\sum_i^p \mathbf{m}_i = 0$, the actual criterion is $|\mathbf{k}_i|/k_f = |\mathbf{m}_i|$.

$$\begin{aligned} \mathscr{I}(\mathbf{n}, \mathbf{k}_i) &= \sum_{\mathbf{l}_i \pm s/2} e^{i2\pi \mathbf{n}\mathbf{m}_i/N_{\rm side}} \\ &= \sum_n^{N_{\rm pix}} \mathscr{F}(\mathbf{m}_i) e^{i2\pi \mathbf{n}\mathbf{m}_i/N_{\rm side}} \\ &= IFFT[\mathscr{F}(\mathbf{m}_i)] \end{aligned} \quad (7)$$

In the first step over all cells that fulfill the criterion is looped, while in the second step we loop over all cells but set all cells that do not fulfill the criterion to zero.

Computing $\mathscr{D}$: $\mathscr{D}$ can be calculated analogously to $\mathscr{I}$. Where the filter function $\mathscr{F}(\mathbf{m}_i)$ has values of 1, the corresponding filter function $\mathscr{W}(\mathbf{m}_i)$ has values of $FFT[\delta](\mathbf{m}_i)$.

Finally we derive the polyspectrum as

$$\mathscr{P}(\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_p) \approx \frac{1}{V}\left(\frac{V}{N_{\rm pix}}\right)^p \frac{\sum_n^{N_{\rm pix}} \prod_{i=1}^{p} \mathscr{D}(\mathbf{n}, \mathbf{k}_i)}{\sum_n^{N_{\rm pix}} \prod_{i=1}^{p} \mathscr{I}(\mathbf{n}, \mathbf{k}_i)} \quad (8)$$

*A thank you or further information

and with the Fourier definition according to NUMPY

$$\mathscr{P}(\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_p) \approx V^{p-1} \frac{\sum_{n}^{N_{\text{pix}}} \prod_{i=1}^{p} \mathscr{D}(\mathbf{n}, \mathbf{k}_i)}{\sum_{n}^{N_{\text{pix}}} \prod_{i=1}^{p} \mathscr{I}(\mathbf{n}, \mathbf{k}_i)}. \quad (9)$$

Note: The resulting powerspectrum relates to the dimension free power spectrum as $\Delta^2(k) = \frac{1}{2\pi^2} k^3 P(k)$.

REFERENCES