MC Data Analysis - Plasma astrophysics
Spectral analysis of in-situ time series measurements

Olga Alexandrova (olga.alexandrova@obspm.fr)
Astronomer, Observatoire de Paris/LESIA

1 Introduction

When analyzing an experimental signal $u(t)$ measured as a function of time t by a spatial probe, it is important to know the representation of this signal in the frequency space. The Fourier Transform (FT) gives us this information. Usually a spectral power (or power spectral density, PSD, i.e. the squares of the Fourier coefficient amplitudes) is used to characterize the distribution of the energy of the signal in frequency f. The rest of the information is contained in the Fourier phases (like the time shift of the different modes). The phases are not easy to handle. To retrieve this information in the simplest form, one needs a method that does spectral analysis in windows located in time.

In addition, in reality, a signal of length T can be very inhomogeneous, corresponding to the crossing of very different regions (for example, when the satellite crosses the boundaries of the magnetosphere, energy events in the solar wind as an ejection of coronal mass, or in the case of intermittent turbulence). So it is important to know how the frequency spectrum varies over time. For this purpose, two classes of transformations are usually used : (i) the Fourier transform with sliding window and (ii) the wavelet transforms.

It is clear that the frequency resolution Δf (or time scale $\Delta\tau = 1/\Delta f$) and the temporal resolution Δt can not be small simultaneously but are linked by the uncertainty principle :

$$\Delta f \Delta T \sim const. \tag{1}$$

In wavelet transforms the analysis windows are specially adapted to the scales. In the Fourier transform with sliding window the temporal resolution decreases as the frequency increases (see introductory course).

1.1 Discrete Fourier Transform

Real data are discrete time series $u[j] = u(t_j) = u(t_0 + j\Delta t)$ of N measurements during the recording time T with a resolution $\Delta t = T/N$, and $t_j = j\Delta t$, $j = 0, 1, 2, ..., N-1$.

The discrete Fourier transform (FT) is defined as the convolution of the signal with a base of orthonormal functions $\sim e^{i\omega t}$ (with $\omega = 2\pi f$). There are different notations for the FT, we will use here the notations of [Welch, 1967] (also used in *IDL*) [1] :

$$\hat{u}[n] = \frac{1}{N}\sum_{j=0}^{N-1} u[j] e^{-2\pi i \frac{nj}{N}} \tag{2}$$

where $n = 0, 1, 2, ..., N-1$ is an index of the frequencies and j is an index of the time measurements points.

— Write the inverse FT expression.

1. In Python, the FT is defined without factor $1/N$, thus, this factor appears in the inverse FT.

For a real signal, equation (2) tells us that negative frequencies do not give additional information.

— Show that if $u(t)$ is real,

$$\hat{u}[N-n] = (\hat{u}[n])^* \tag{3}$$

— This transformation retains energy. Demonstrate Parseval's Theorem :

$$\frac{1}{N}\sum_{j=0}^{N-1} |u[j]|^2 = \sum_{n=0}^{N-1} |\hat{u}[n]|^2 \tag{4}$$

The energy distribution in the frequency signal is given by the spectral energy density (or power spectra density PSD), defined as :

$$S[n] = 2T|\hat{u}[n]|^2, \;\; n = 0, 1, 2, ..., N/2 \tag{5}$$
$$f_n = n/T, \;\; n = 0, 1, 2, ..., N/2 \tag{6}$$

Note, that with python definition of the FT, the PSD is

$$S[n] = \frac{2T}{N^2}|\hat{u}[n]|^2, \;\; n = 0, 1, 2, ..., N/2 \tag{7}$$

for N equal to an even number.

— Give the units of $S[n]$.

— Write the Parseval theorem using $S[n]$.

— Show that for $n = 0$, $S[n]$ gives the mean energy of the signal.

1.2 Morlet wavelet transform

Morlet's wavelet [Farge, 1992 ; Torrence & Compo 1998] is a wave of frequency ω_0 modulated by a Gaussian :

$$\psi_0(t) = \pi^{-1/4} e^{-i\omega_0 t} e^{-t^2/2} \tag{8}$$

where t is the time.

The wavelet transform of the signal $u(t_j)$ is the convolution of this signal with the dilated and translated function $\psi_0(t_j)$

$$\mathcal{W}(\tau, t) = \sum_{j=0}^{N-1} u(t_j)\psi^*[(t_j - t)/\tau] \tag{9}$$

where the asterisk * indicates the conjugated complex, t_j is the position (a translation parameter) around which convolution is performed in a window Δt ; and τ is a dilation parameter or time scale. The choice of scales for non-orthogonal wavelet analysis, such as Morlet's, is arbitrary. It is practical to define scales as a fractional power of two :

$$\tau_m = \tau_0 2^{m\cdot\delta m}, \;\; m = 0, 1, ..., M \tag{10}$$
$$M = \frac{1}{\delta m}\log_2(N\delta t/\tau_0) \tag{11}$$

where τ_0 is the smallest scale and M determines the largest scale, τ_M. In total, there are $M + 1$ time scales. It is convenient to choose $\tau_0 = 2\delta t$ so that the time scale τ can be easily transformed into a physical (or Fourier) frequency f. Here δt is the temporal resolution of the measurements of $u(t)$. The resolution in scales of the transformation is determined by δm : the smaller the δm, the better the scales are resolved. Here we use $\delta m = 1/8$. For $\omega_0 = 6$ in (8), there is a simple relationship between scales τ and Fourier frequencies, see [Torrence &Compo 1998] : $\frac{1}{f} = 1.03\tau \simeq \tau$.

Convolution (9) transforms a real function of a variable (time) into a complex function of two variables, time and time scale τ (frequency f). In fact, the temporal resolution of the wavelet transformation ΔT and frequency resolution Δf are linked by the Heisenberg relationship, see equation (1). The Morlet wavelet with $\omega_0 = 6$ is a good compromise for both time and frequency resolution.

The *scalogram* is energy distribution of the signal in time and scales :

$$Power(\tau, t) = |\mathcal{W}(\tau, t)|^2. \tag{12}$$

The square of the module of a wavelet coefficient represents the "quantum" of energy of the fluctuations of $u(t)$ on the surface $\Delta T \times \Delta\tau$ centred around the moment t_j and the scale τ_m (or the frequency f_m). When using $f = 1/\tau$, energy distribution in time and frequency, $|\mathcal{W}(f,t)|^2$, is called *spectrogram.*

To obtain the spectrum of fluctuations, simply integrate on the time variable :

$$\mathcal{W}^2(f) = \frac{1}{N}\sum_{i=0}^{N-1} |\mathcal{W}(f, t_j)|^2. \tag{13}$$

The spectral power equivalent to FT's PSD is, see [Torrence & Compo 1998] :

$$S_w = 2\delta t \mathcal{W}^2(f). \tag{14}$$

2 Application of Fourier and wavelet transformations

2.1 Analysis of synthetic signals

1. Build a synthetic signal
$$y_0(t) = A_0 \sin(2\pi f_0 t),$$
with $t_j = j\Delta t$ and $j = 0, 1, 2, ..., N-1$, we'll take $\Delta t = 1$, $N = 10^3$.
In Python, the numpy library allows you to manage tables. *numpy.arange(N)* allows to build an array from 0 to N-1.
2. Determine frequency range for f_0 : what are minimal, f_{min}, and maximal, f_{max}, frequencies possible ?
3. Write a function that calculates the discrete FT and PSD of a real signal using equations (2) and (7).
4. Apply the discrete FT to the signal y_0. Determine the PSD and frequencies f. Make a plot PSD(f).
5. Build a synthetic signal
$$y_1(t) = A_0 \sin(2\pi f_0 t) + A_1 \sin(2\pi f_1 t).$$
6. Build a synthetic signal
$$y_2(t) = \begin{cases} A_0 \sin(2\pi f_0 t), & \text{pour } t < T/2 \\ A_1 \sin(2\pi f_1 t), & \text{pour } t \geq T/2 \end{cases}$$

7. Apply the discrete FT to $y_1(t)$ and $y_2(t)$. Compare the results for the two signals. Verify your results with FFT of Python (*nampy.fft.fft*).
8. Now, we will apply wavelet transform to $y_1(t)$ and $y_2(t)$. Torrence & Compo, the authors of 'A Practical Guide to Wavelet Analysis' [1998] produced the corresponding program for python, waveletFunctions.py :
   `https://github.com/chris-torrence/wavelets/tree/master/wave_python`.
   Copy it in your folder.
9. Apply the Morlet wavelet transformation to $y_1(t)$ and $y_2(t)$.
   Syntaxe : from waveletFunctions import wavelet
   wave, period, scale, coi = wavelet(y, dt)
   where y is the analysed signal and dt is the time resolution of the signal.

   Plot scalograms for $y_1(t)$ and $y_2(t)$. Compare.

   What is the range of time scales resolved by the transformation ? Compare the range of frequencies f determined above.
10. Conclude on the efficiency of both transformations.
11. Build a synthetic signal which contains a sharp discontinuity (like a shock), a wave-packet localised in time and a soliton-like event much shorter in time than a wave-packet. The discontinuity can be modeled with

$$y_d(t) = A_d \tanh\left(\frac{t - t_d}{\Delta t_d}\right). \qquad (15)$$

   The soliton-like event can be models as

$$y_s(t) = A_s \exp\left(-\left(\frac{t - t_s}{\Delta t_s}\right)^2\right). \qquad (16)$$

   The wave packet is

$$y_w(t) = A \exp\left(-\left(\frac{t - t_0}{\Delta t_w}\right)^2\right) \cos(\omega t). \qquad (17)$$

   Apply the Morlet wavelet transformation to this signal. Comment the result.

2.2 In-situ data analysis

1. Read the data
   syntaxe : `from scipy.io.idl import readsav`
   `data=readsav("filename.sav")`

   To see the saved variables within data dictionary write
   `data.keys()`

   Plot of the components of the magnetic field as a function of time $B_j(t)$, with $j = x, y, z$.
2. Make a zoom for a shorter time interval
```
z=(time >= t_initial) & (time <= t_final)
t=time[z]
Bx=B_x[z]
```

3. Apply the WT to `Bx(t)`. Plot the scalogram together with `coi(t)`. To represent the scalogram use `plt.contourf(time,period,power)`.

   Determine minimal and maximal time scale τ.
4. Using the scalogram, choose a homogeneous time interval and then apply FT to it using FFT of Python (*nampy.fft.fft*). Calculate the PSD of B_x. Give minimal f_{min} and maximal f_{max} frequencies.
5. Calculate PSD using WT, compare with PSD determined via FT.
6. Make a figure resuming the analysis of $B_x(t)$: (a) Signal $B_x(t)$; (b) scalogram $W_x^2(\tau, t)$; (c) Fourier spectrum $S_x(f)$ with wavelet spectrum $S_{w,x}(f)$.

7. Conclude on the limits of the application of the two methods, effectiveness, simplicity of interpretation of the results, quantity of information.
8. Within the time available, determine the total PSD, i.e. the sum of the PSD of the components. If the PSD follows a power law, make a linear adjustment between $\log_{10}(PSD)$ and $\log_{10}(f)$, using the *scipy* library. Report the value of the slope found on the graph.

"Quicklook" of Cluster measurements (at 1 AU) can be found here :
`http://www.cluster.rl.ac.uk/csdsweb-cgi/csdsweb_pick`