

## Tutorial : FFT, PSD & coherence with MATLAB

Version 2 - April 2008 - Ch. Bailly

### 1 Fast Fourier Transform

Definition of the Fourier transform :

$$\hat{u}(f) = \mathcal{F}[u(t)] = \int_{-\infty}^{+\infty} u(t)e^{-i2\pi ft} dt \quad u(t) = \int_{-\infty}^{+\infty} \hat{u}(f)e^{i2\pi ft} df \quad (1)$$

As an example, consider the following function :

$$u(t) = e^{-(t/\tau_0)^2} \cos(2\pi f_0 t) \quad (2)$$

This signal has a Gaussian envelope with a 3-dB bandwidth  $b_w = 2\sqrt{\ln 2}/(\pi\tau_0)$  and an oscillatory part centered around  $f_0$ . The Fourier transform of (2) in the frequency domain is given by :

$$\begin{aligned} \hat{u}(f) &= \int_{-\infty}^{+\infty} e^{-(t/\tau_0)^2} \cos(2\pi f_0 t) \cos(2\pi ft) dt \\ &= \frac{1}{2} \int_{-\infty}^{+\infty} e^{-(t/\tau_0)^2} \{ \cos[2\pi(f-f_0)t] + \cos[2\pi(f+f_0)t] \} dt \\ &= \frac{\tau_0\sqrt{\pi}}{2} \left( e^{-\pi^2\tau_0^2(f-f_0)^2} + e^{-\pi^2\tau_0^2(f+f_0)^2} \right) \end{aligned} \quad (3)$$

The total power of the signal is the same in the time domain or in the frequency domain. This result is known as the Parseval theorem :

$$\int_{-\infty}^{+\infty} |u(t)|^2 dt = \int_{-\infty}^{+\infty} |\hat{u}(f)|^2 df \quad (4)$$

The discrete Fourier transform associated with expression (3) can be computed using the Fast Fourier Transform (FFT) algorithm. Let  $\Delta t$  the time interval for sampling or  $f_s = 1/\Delta t$  the sampling frequency :

$$t_n = n\Delta t \quad n = -N/2, \dots, N/2 \quad u(t_n) = u_n$$

Notice that  $s(t_{-N/2}) = s(t_{N/2})$ . Thus we have  $N$  consecutive independent sampled values where  $N$  is even to make things simpler. The discrete Fourier transform writes :

$$\hat{u}(f_j) = \hat{u}_j = \Delta t \sum_{n=1}^N u_n e^{-i2\pi f_j t_n} \quad f_j = j\Delta f \quad \Delta f = \frac{1}{N\Delta t} \quad j = -N/2, \dots, N/2$$

The two extreme values of  $j$  are not independent,  $\hat{u}_{-N/2} = \hat{u}_{N/2}$ , so that the discrete Fourier transform maps  $N$  complex values ( $u_n$ ) to ( $\hat{u}_j$ ). A complete representation is obtained if the signal is bandlimited and the sampling frequency greater than twice the signal bandwidth : this is the Nyquist-Shannon theorem or the sampling theorem. The critical frequency associated to the bandwidth of the signal is the Nyquist frequency  $f_c = 1/(2\Delta t)$ .

Using MATLAB convention for subscripts, the FFT algorithm provides the following coefficients  $[\hat{u}_l]$  :

$$[\hat{u}_l] = \sum_{n=1}^N u_n e^{i2\pi(n-1)(l-1)/N} \quad 1 \leq l \leq N \quad (5)$$

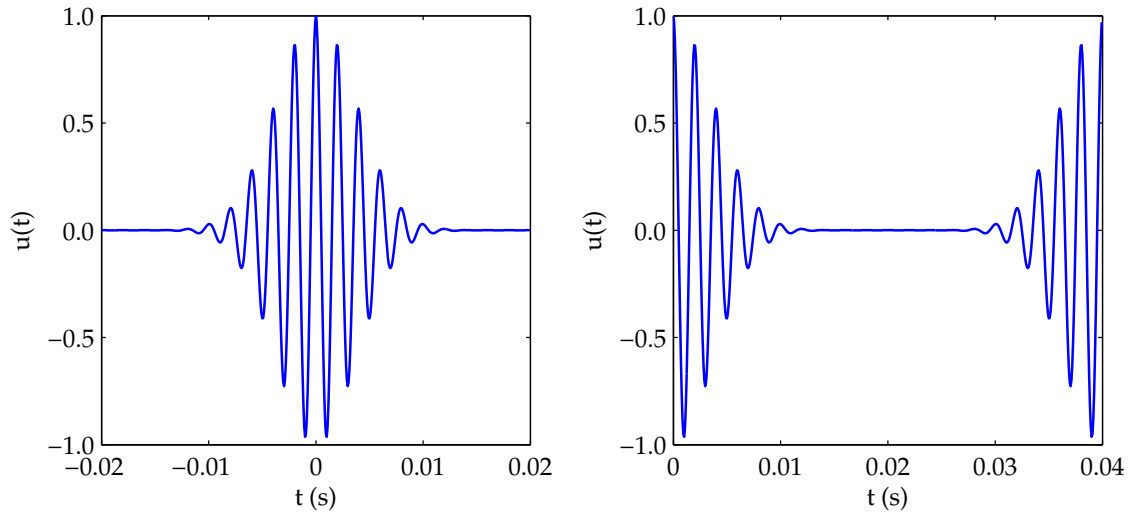


FIG. 1 – At left, signal  $u(t)$  defined by expression (2) with  $f_0 = 500$  Hz and a bandwidth  $b_w = 100$  Hz. At right, the same rearranged signal in the time domain.

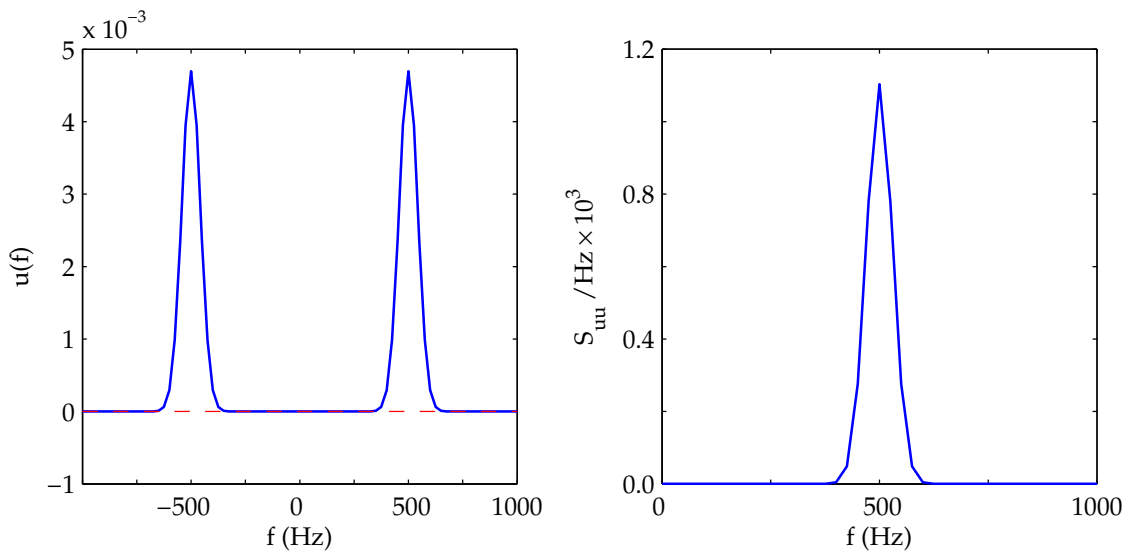


FIG. 2 – At left, Fourier transform of  $u(t)$  with  $N = 512$  points : real part in solid blue line and imaginary part in dashed red line. At right, Power Spectral Density of  $\hat{u}(k)$ .

A specific storage arrangement is used with this algorithm : the first part of the vector  $[\hat{u}_l]$  is associated with positive frequencies, *i.e.*  $l = 1$  to  $N/2 + 1$  corresponds to  $f_j$  for  $j = 0, \dots, N/2$ , and the second part corresponding to negative frequencies, *i.e.*  $l = N/2 + 1$  to  $N$  corresponds to  $f_j$  for  $j = -N/2, \dots, -1$ .

As illustration, figure 1 displays the signal  $u(t)$  and the same rearranged signal to avoid the phase shift using the `fft` algorithm. The Fourier transform is reported in figure 2 and the `MATLAB` script is given page 5.

## 2 Power Spectrum Estimation

An estimation of the power spectrum  $S_u$  is usually defined from the mean squared amplitude of the signal for a stationary random process :

$$\overline{u^2} = \frac{1}{T} \int_{-\infty}^{+\infty} u^2(t) dt = \frac{1}{T} \int_0^T |\hat{u}(f)|^2 df = \int_{-\infty}^{+\infty} S_{uu}(f) df$$

where  $T = N\Delta t$  is the observation interval. It follows that :

$$\int_{-\infty}^{+\infty} S_u(f) df \simeq \frac{\Delta f}{T} \sum_{j=-N/2}^{N/2} |\hat{u}_j|^2$$

which provides :

$$S_{uu}(f_j) = \frac{1}{T} |\hat{u}_j|^2$$

Usually, a one-sided spectrum is defined for positive frequencies  $0 \leq f_j \leq f_c$ , and the Power Spectral Density (PSD) is estimated by the expression :

$$S_{uu}(f_j) = \frac{2}{T} |\hat{u}_j|^2 = \frac{2\Delta t}{N} |[\hat{u}_l]|^2 \quad j = 0, \dots, N/2 \quad \text{or} \quad l = 1, \dots, N/2 + 1. \quad (6)$$

where the coefficients  $[\hat{u}_l]$  are directly provided by the FFT, see equation (5). This function is plotted in figure 2 for the signal (2) and several scripts are given as illustration page 5. Note that no windowing is used in the present example.

## 3 Coherence function

The cross-correlation of two functions  $u$  and  $v$  can be estimated by :

$$R_{uv}(\tau) = E [u(t)v^*(t - \tau)] \simeq \frac{1}{T} \int_0^T u(t)v^*(t - \tau) dt \quad (7)$$

The cross-correlation theorem states that  $\mathcal{F}[R_{uv}(\tau)] = S_{uv}(f)$ , and this result is known as the Wiener-Khinchin theorem for  $u = v$ . As a consequence, the previous expression (6) is recovered for the one-side spectrum :

$$S_{uu}(f_i) = \frac{2}{T} \hat{u}(f_j) \hat{u}^*(f_j) = \frac{2}{T} |\hat{u}_j|^2 = \frac{2\Delta t}{N} |[\hat{u}_l]|^2$$

with  $j = 0, \dots, N/2$  or  $l = 1, \dots, N/2 + 1$ . The coherence function may be defined as :

$$\gamma_{uv}(f) = \frac{S_{uv}}{\sqrt{S_{uu}}\sqrt{S_{vv}}} \quad \text{or} \quad \gamma_{uv}^2(f) = \frac{|S_{uv}|^2}{S_{uu} S_{vv}} \quad (8)$$

An example of coherence function is displayed in figure (3), and the corresponding script is given in page 6.

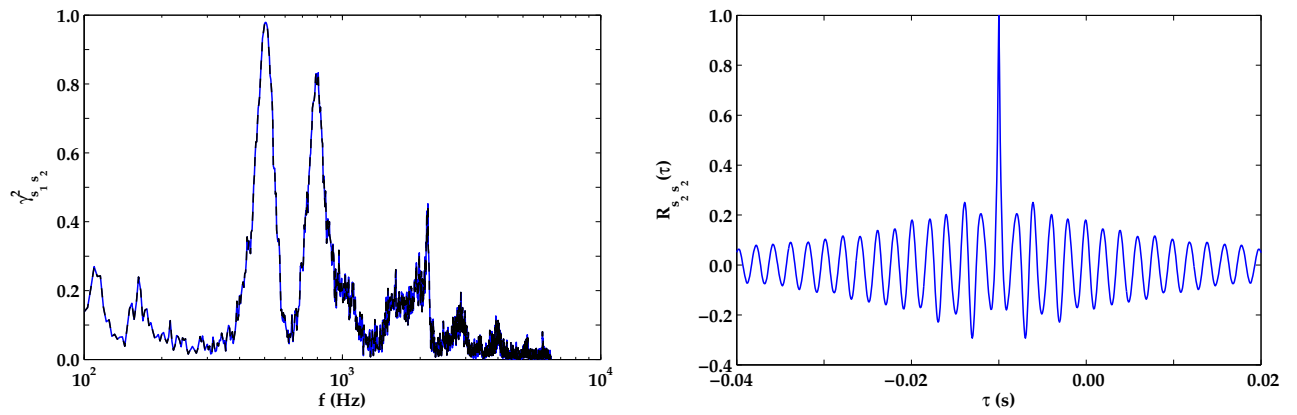


FIG. 3 – Left : mean squared coherence function  $\gamma^2$ ; right : autocorrelation function. See Matlab script page 6.

```

%.. number of points for the Fourier transform
nfft = 512;
nf = nfft/2;

%.. signal
tm = linspace(-0.02,0.02,nfft+1);
t = tm(1:nfft);

dt = t(2)-t(1);
fs = 1./dt;

f0 = 500.;
Bw = 100;
tau = 2.*sqrt(log(2.)) /(pi*Bw);

s = exp(-(t/tau).^2) .* cos(2*pi*f0*t);

%.. Analytical Fourier transform
df = 1/(nfft*dt);
f = -nf*df:df:nf*df;
tfs_ref = tau*sqrt(pi)/2. * (exp(-pi^2*tau^2*(f-f0).^2) + exp(-pi^2*tau^2*(f+f0).^2));

%.. Total power of the signal (must be the same value)
Es = trapz(t,s.^2);
Ets_ref = trapz(f,tfs_ref.^2);
Lt = nfft*dt;
disp(['Mean squared amplitude (Parseval) -- Es = ',num2str(Es/Lt),' Ets = ',num2str(Ets_ref/Lt)]);

%.. Fourier transform (through fftshift here to avoid any phase shift)
t1 = 0:dt:(nfft-1)*dt;
s1 = fftshift(s);

tfs1_coef = fft(s1);
tfs1 = tfs1_coef * dt;

fs1 = 0:df:nf*df;

%.. Power Spectral Density (I)
PSDs = 2.*dt/nfft * abs(tfs1_coef(1:nf+1)).^2;
PSDs_int = trapz(fs1,PSDs);

disp(['PSD - home made E = ',num2str(PSDs_int)]);

%.. Power spectral density (II)
w = ones(nfft,1);
[PSD1 f1] = periodogram(s,w,nfft,1./dt);

PSD1_int = trapz(f1,PSD1);
disp(['PSD - periodogram E = ',num2str(PSD1_int)]);

%.. Power spectral density (III)
w = ones(nfft,1);
noverlap = 0;
[PSD2 f2] = pwelch(s,w,noverlap,nfft,1./dt);

PSD2_int = trapz(f2,PSD2);
disp(['PSD - Welch E = ',num2str(PSD2_int)]);

```

```

%.. coherence function

nfft = 4096;
df = 1/(nfft*dt);

w = hanning(nfft);
noverlap = 0;

ipmax = floor(np/nfft);
ip = ipmax;

tp = (0:ip-1)*dt;

Sxx = zeros(nfft/2+1,1);
Syy = zeros(nfft/2+1,1);
Sxy = zeros(nfft/2+1,1);

for i=1:ip
    [PSD1 f1] = periodogram(ps1((i-1)*nfft+1:i*nfft),w,nfft,1./dt);
    Sxx(:) = Sxx(:) + PSD1;

    [PSD1 f1] = periodogram(ps2((i-1)*nfft+1:i*nfft),w,nfft,1./dt);
    Syy(:) = Syy(:) + PSD1;

    [PSD1,f1] = cpsd(ps1((i-1)*nfft+1:i*nfft),ps2((i-1)*nfft+1:i*nfft),w,noverlap,nfft,1./dt);
    Sxy(:) = Sxy(:) + PSD1;
end
Sxx = Sxx/ip;
Syy = Syy/ip;
Sxy = Sxy/ip;

gxy = Sxy./(sqrt(Sxx).*sqrt(Syy));
g2xy = Sxy.*conj(Sxy)./(Sxx.*Syy);

%.. g2xy = C2xy directly computed by magnitude squared coherence estimate

noverlap = 0;
[C2xy,f2] = mscohere(ps1,ps2,w,noverlap,nfft,1./dt);

%.. cross-correlation function rpp(tau)

nc = 512;

ipmax = floor(np/nc);
ip = ipmax;

u2rms = sqrt( sum(ps2(1:ip*nc).^2)/(ip*nc));

tc = (-nc+1:1:nc-1)*dt;

rpp = zeros(2*nc-1,1);
for i=1:ip
    [rtmp tc1] = xcorr(ps2((i-1)*nc+1:i*nc),ps2((i-1)*nc+1:i*nc),'unbiased');
    rpp(:) = rpp(:) + rtmp;
end
rpp2 = rpp / ip;
rpp2 = rpp2/u2rms^2;

```