TECNICHE DI ANALISI DEL SEGNALE: IMPLEMENTAZIONE IN AMBIENTE MATLAB

G. D'Elia

SEZIONE 1

Trasformata Discreta di Fourier

La trasformata discreta di Fourier (DFT) é:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j2\pi\frac{k}{N}n} \tag{1}$$

In ambiente Matlab é presente un comando per la valutazione della DFT tramite l'algoritmo FFT. In particolare il comando **fft** valuta la seguente espressione:

$$fft(x) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi\frac{k}{N}n} \tag{2}$$

Per cui, supponendo che x sia un vettore riga contente il segnale formato da N punti, si ha:

```
X = 1/N*fft(x)
```

La trasformata inversa puó quindi essere ottenuta con il comando:

```
x = N*ifft(X)
```

Di seguito il programma Matlab che genera un sengnele sinusoidale e ne calcola la trasformata di Fourier.

```
clear
clc

% Creazione del segnale
T = 4;  %[s]
N = 2^14;         % numero di punti in cui \'e diviso il segnale
fs = N/T;         % frequenza di campionamento [Hz]


t = (0:N-1)/fs; % vettore dei tempi

f = 4;  % frequenza del segnale
x = sin(2*pi*f*t);       % segnale sinusoidale di frequenca f
```

```
% plot del segnale
figure,
% settiamo le dimensioni delle figure in cm
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
% plotta il segnale in nero
plot(t,x,'k')
xlabel('time [s]')
ylabel('amplitude')
```

```
% Valutazione della trasformata di Fourier
X = 1/N * fft(x);
f = (0:N-1)*fs/N;         % vettore delle frequenze
```

```
% plot della trasformata di Fourier
figure,
% settiamo le dimensioni delle figure in cm
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
% plot dell valore assoluto
plot(f,abs(X),'k')
xlabel('frequency [Hz]')
ylabel('amplitude')
xlim([0 10])      % limita il plot da 0 a 10 Hz
```

Il vettore X contenente la trasformata di Fourier di x é double-side. Cioé i primi $N/2$ punti sono riferiti alla frequenze positive dello spettro, gli altri $N/2$ alle frequenze negative.

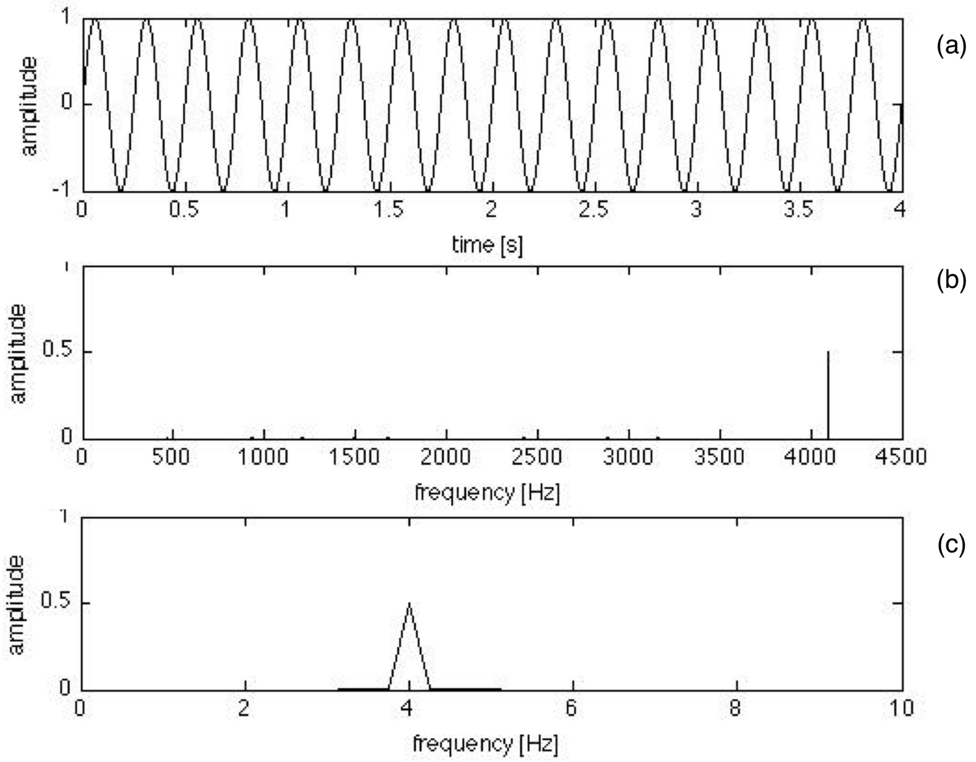

Figura 1: (a) segnale nel tempo, (b) trasformata di Fourier, (c) trasformata di Fourier nel range $0 \div 10$ Hz

SEZIONE 2

Autopower e PSD

Sia X la trasformata di Fourier del vettore x, l'Autospettro é definito come:

$$Autopower = \mid X \mid^2 = XX^*$$

dove X^* é il complesso coniugato di X.

```
Autopower = X.*conj(X); % In matlab l'istruzione conj() calcola il
                         % complesso coniugato di un numero complesso
```

Per determinare la PSD occorrerebbe quindi avere durate di acquisizione dati infinite, ciononostante si puó ricavare una stima della PSD ($\hat{S}(f)$) prendendo un numero di acquisizioni di durata finita e mediando i loro risultati (metodo del Periodogramma). Si divide il segnale totale in M segmenti, ciascuno di durata T, la stima della PSD é quindi data da:

$$\hat{S}(f) = \frac{1}{M} \sum_{m=1}^{M} \frac{|X_m(f)|^2}{T} \quad (3)$$

Considerando quindi la trasformata discreta di Fourier, ottenuta dalla trasformata integrale, si ha:

$$\begin{aligned}\hat{S}(k\Delta f) &= \frac{1}{M}\sum_{m=1}^{M}\frac{|X_m(k\Delta f)|^2}{T} \\ &= \frac{1}{M}\sum_{m=1}^{M}\frac{1}{N\Delta t}\Big|\Delta t\sum_{n=0}^{N-1}x(n)e^{-j2\pi\frac{k}{N}n}\Big|^2 \\ &= \frac{1}{M}\sum_{m=1}^{M}\frac{\Delta t}{N}\Big|\sum_{n=0}^{N-1}x(n)e^{-j2\pi\frac{k}{N}n}\Big|^2 \end{aligned} \tag{4}$$

si puó quindi usare lo strumento **fft** per determinare la PSD.

```
T = 10;      %[s]
N = 2^18;    % numero di punti in cui \'e diviso il segnale

fs = N/T;    % frequenza di campionamento

t = [0:N-1]/fs; % vettore dei tempi

f_sig = 40;          % frequenza del segnale

x = sin(2*pi*f_sig*t) + 10.*randn(1,N);  % segnale sinusoidale di frequenca f

% plot del segnale
figure,
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 6])
plot(t,x,'k')
set(get(gca,'XLabel'),'String','Time [s]');
set(get(gca,'YLabel'),'String','Amplitude');


% Stima della psd
Nw = 2^16;        % window length
Noverlap = 0;  % number of overlaping points
R = Nw - Noverlap;
K = fix((N - Noverlap)/R);      % Number of averages
win = rectwin(Nw)';
psd_stim = zeros(K,Nw);
index = 1:Nw;
for ind = 1:K,
    x_win = x(index).*win;
    X_win = 1/Nw .* fft(x_win);
    psd_stim (ind,:)= (X_win .* conj(X_win))*N/fs;
    index = index + R;
end

PSD = 1/K.*sum(psd_stim)./(1/Nw.*sum(abs(win).^2));              % L'ultimo termine Ë dovuto alla
% normalizzazione dell'energia
f = (0:Nw-1)*fs/Nw;

figure,
% settiamo le dimensioni delle figure in cm
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
% plot dell valore assoluto
plot(f,PSD,'k')
xlabel('frequency [Hz]')
ylabel('amplitude')
xlim([0 100])    % limita il plot da 0 a 100 Hz
```

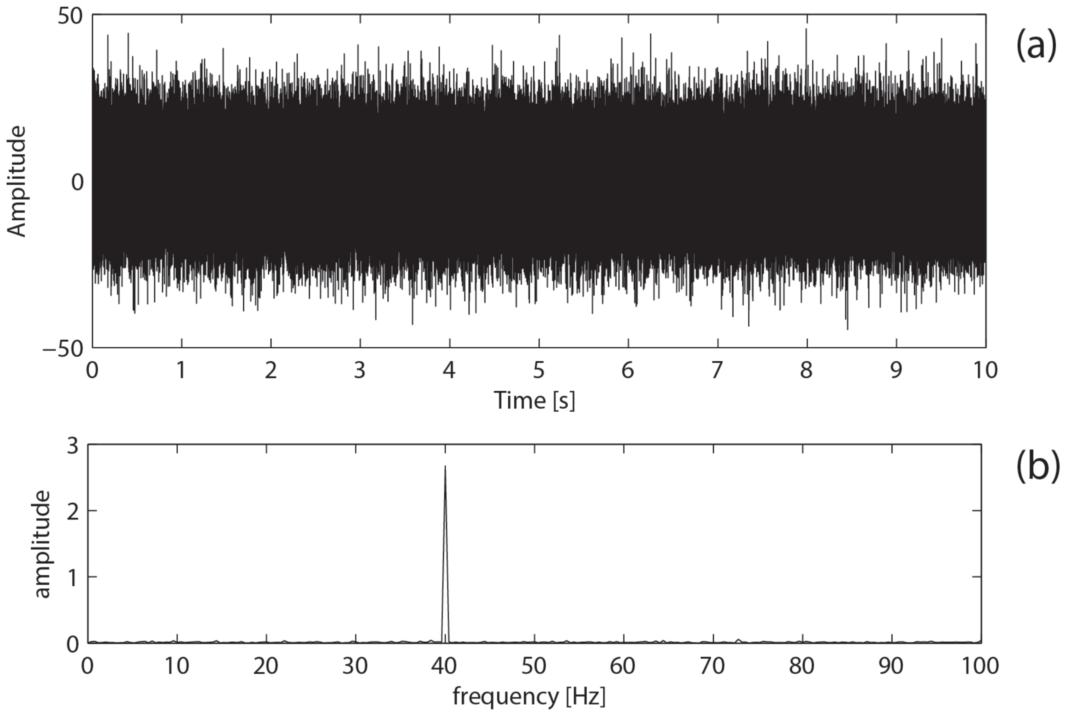

Figura 2: (a) segnale nel tempo, (b) PSD ottenuta con 4 medie e $\Delta f = 0.4Hz$

SEZIONE 3

Trasformata di Hilbert

Il comando Matlab:

```
hilbert(x)
```

non valuta la trasformata di Hilbert del segnale x, ma il segnale analitico x_a. Di conseguenza la trasformata di Hilber di x viene otttenuta dalla parte immaginaria di x_a.

Per fini diagnostici il segnale analitico ha una maggiore utilitá rispetto alla trasformata di Hilbert. Infatti dal segnale analiti possono essere estratte le modulanti di ampiezza e fase del segnale.

Di seguito lo script Matlab che genera un segnale sinusoidale modulato in ampiezza e ne estrae la modulazione tramite la trasformata di Hilbert.

```
clear
clc


%% Creazione del segnale
T = 4;                          % Durata segnale
N = 2^14;                       % Numero di punti
fs = N/T;                                   % Frequenza di campionamento


t = (0:N-1)/fs;                 % Vettore tempi
```

```
f = (0:N-1)*fs/N;        % Vettore delle frequenze


% segnale modulato
Xm = 1;                  % ampiezza del segnale
fs = 10;                 % frequenza del segnale [Hz]
Am = 0.5;        % ampiezza della modulante
fm = 4;                  % frequenza della modulante [Hz]

x = Xm*(1 + (Am*cos(2*pi*fm.*t))).*cos(2*pi*fs.*t);


%% Calcolo del segnale analitico
xa = hilbert(x);

%% Estrazione della modulazione di ampiezza
ma = abs(xa);


%% Plot dei risulatati
% Plot del segnale
figure,
% settiamo le dimensioni delle figure in cm
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
% plotta il segnale in nero
plot(t,x,'k')
xlabel('time [s]')
ylabel('amplitude')


% Plot della modulante
figure,
% settiamo le dimensioni delle figure in cm
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
% plotta il segnale in nero
plot(t,ma,'k')
xlabel('time [s]')
ylabel('amplitude')
```

L'output di questo programma é rappresentato in Fig. 3.

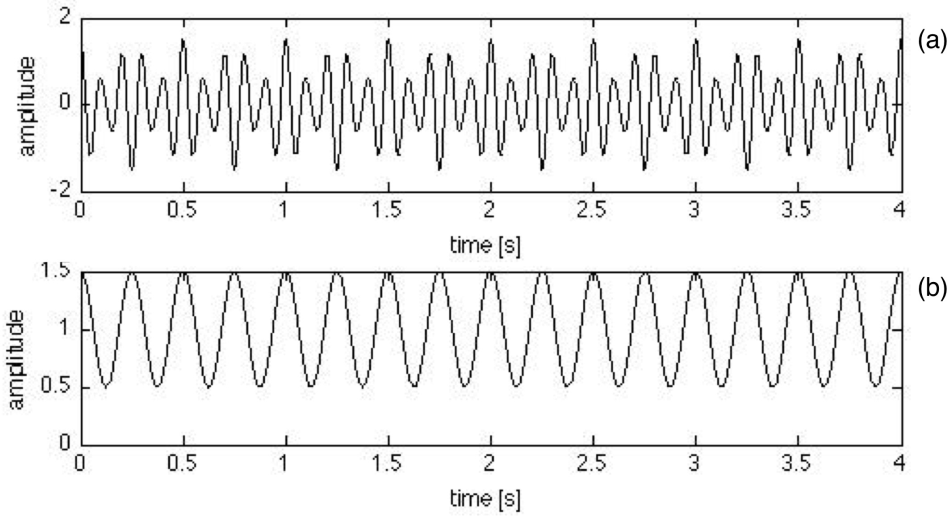

Figura 3: (a) segnale modulato in ampiezza nel tempo, (b) funzione modulante nel tempo

SEZIONE 4

Media Sincrona

```
%% Ricampionamento in base angolo
triglevel = 1;            %livello del triggher
test = tacho>triglevel;
test = diff(test);
test=(test > 0.5);
ind_low = find(test);

t_trig = zeros(1,length(ind_low));
for ind = 1:length(ind_low),
    t_trig(ind) = interp1(tacho(ind_low(ind):ind_low(ind)+1),t(ind_low(ind):ind_low(i
end

Mpitch = 24;
M = Mpitch*z;
k = M/2:3*M/2-1;
deltatheta = 2*pi/M;
theta = k*deltatheta;
index = 1:length(k);
tk = zeros(1,length(k)*(length(t_trig) - 2));
for ind = 1:length(t_trig) - 2;
    tMatrix = [1 t_trig(ind) t_trig(ind)^2
        1 t_trig(ind+1) t_trig(ind+1)^2
        1 t_trig(ind+2) t_trig(ind+2)^2];
    DeltaPhi = [0
        2*pi
        2*2*pi];
    b = tMatrix\DeltaPhi;
    tk(1,index) = 1/(2*b(3)) .* (sqrt(b(2)^2 + 4*b(3)*(theta-b(1))) - b(2));
```

```
    index = index + length(k);
end
xAngle = interp1(t,x,tk,'spline');

figure,
plot(1./diff(t_trig))

%% Media sincrona
ciclix = zeros(length(xAngle)/M,M);
indexMean = 1:M;
for ind = 1:length(xAngle)/M,
    ciclix(ind,:) = xAngle(indexMean);
    indexMean = indexMean + M;
end

mediaSinc = mean(ciclix);
```