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.

SEZIONE 2

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.

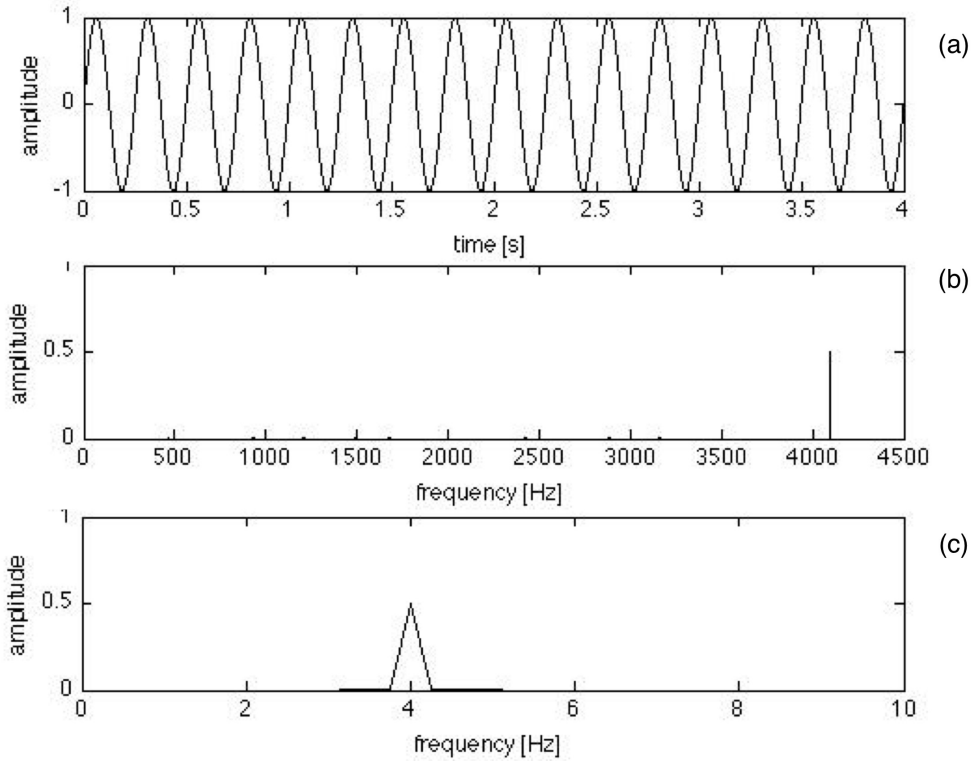

Figura 1: (a) segnale nel tempo, (b) trasformata di Fourier, (c) trasformata di Fourier nel range $0 \div 10$ Hz

```
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. 2.

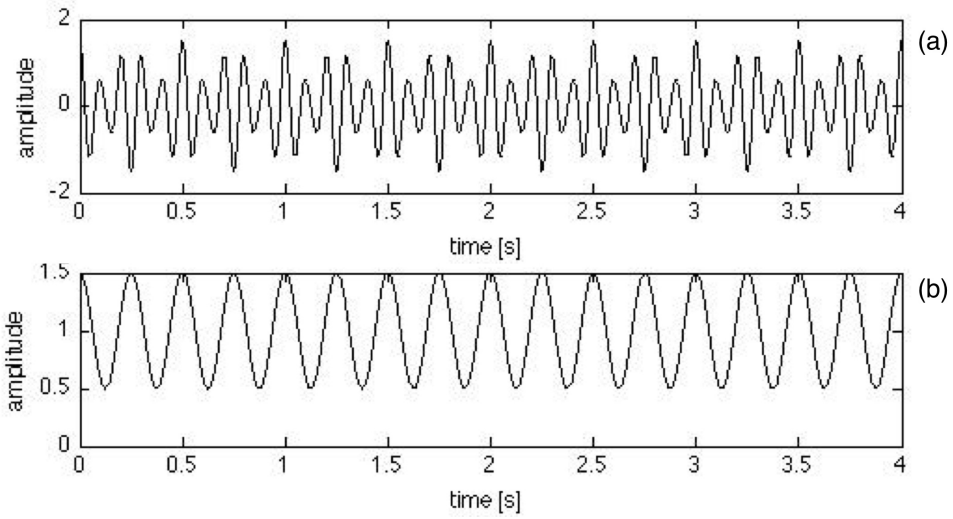

Figura 2: (a) segnale modulato in ampiezza nel tempo, (b) funzione modulante nel tempo

SEZIONE 3

Short-Time Fourier transform

La Short-Time Fourier transform (STFT) é definita nel modo seguente:

$$X(\tau, f) = \int_{-\infty}^{+\infty} x(t)w(t-\tau)e^{-\jmath 2\pi ft}dt \tag{3}$$

```
clear
clc

%% Signal and constants
T = 4;          % signal length
N = 2^14;       % Number of points
fs = N/T;       % sample frequency

f1 = 10;        % frequency of the first sinusoid
f2 = 40;        % frequency of the second sinusoid

t = (1:N)/fs;   %Timev ector used to plot


% Construct a step change in frequency
tn = (1:N/4)/fs;        % Time vector used to create sinusoids
x = [zeros(1,N/4) sin(2*pi*f1*tn) sin(2*pi*f2*tn) zeros(1,N/4)];


%% Computation of the STFT
Nw = 1024;      % window length
Noverlap = fix(2*Nw/3);  % number of overlaping points
R = Nw - Noverlap;
K = fix((N - Noverlap)/R);     % Number of averages
Window = hanning(Nw);  % Window function (column vector)

% memory pre-allocation
x_win = zeros(Nw,1);
STFT = zeros(Nw,K);

index = 1:Nw;
for ind = 1:K,
x_win = x(index)' .* Window;
STFT(:,ind) = 1/Nw .* fft(x_win);
index = index + (Nw - Noverlap);
end;

STFT = STFT(1:Nw/2-1,:);     % Positive frequencies only
f_stft = (0:Nw/2-2)*fs/Nw;     % Frequency vector
t_stft = (1:K)*R/fs;
```

```
%% Plot
figure,
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
plot(t,x,'k'),
xlabel('Time [s]'),
ylabel('Amplitude')

% STFT plot
f_range = [0 200];
If = find(f_stft >= f_range(1) & f_stft < f_range(2));

figure, mesh(t_stft,f_stft(If),abs(STFT(If,:)))
colormap(gray)
view(-18,54)
xlabel(['Time [s] ; \Delta t=',num2str(diff(t_stft(1:2))),' s']),
ylabel(['Frequency [Hz] ; \Delta f=',num2str(diff(f_stft(1:2))),' Hz']),
zlabel('|STFT|')
title(['STFT; Nw=',num2str(Nw),'; Noverlap=',num2str(Noverlap),'; K = ',num2str(K)])

figure, imagesc(t_stft,f_stft(If),abs(STFT(If,:)))
axis xy
colormap(gray)
xlabel(['Time [s] ; \Delta t = ',num2str(diff(t_stft(1:2))),' s']),
ylabel(['Frequency [Hz] ; \Delta f = ',num2str(diff(f_stft(1:2))),' Hz']),
title(['STFT; Nw=',num2str(Nw),'; Noverlap=',num2str(Noverlap),'; K = ',num2str(K)])
```

L'output di questo script é riportato in Fig. 3.

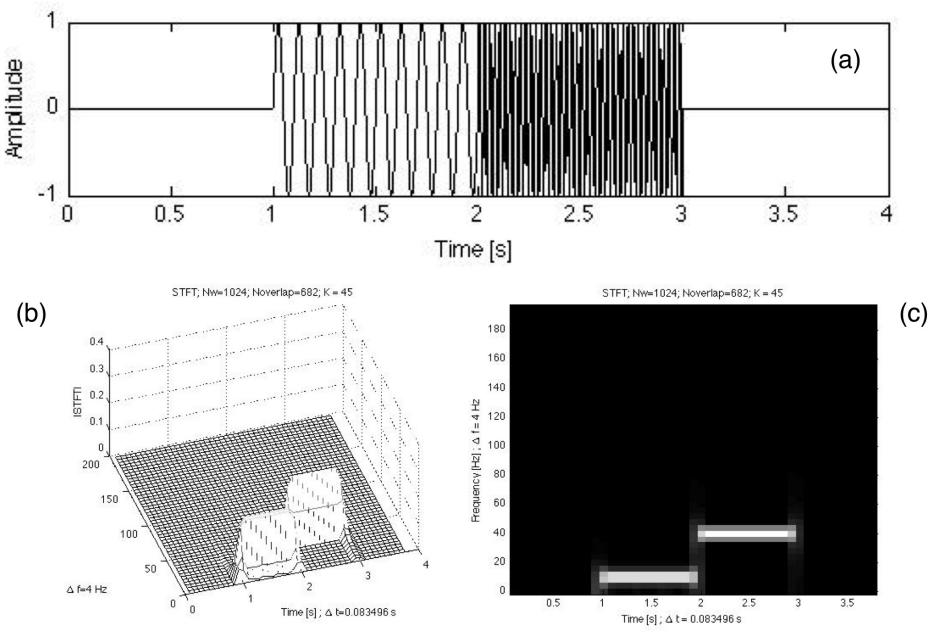

Figura 3: (a) segnale nel tempo, (b) (c) STFT

La STFT puó anche essere determinata con una function presente nel signal processing toolbox di Matlab:

```
[STFT,t,f] = spectrogram(x,window,noverlap,nfft,fs);
```

SEZIONE 4

Media sincrona

Di seguito lo script Matlab che calcola la media sincrona di un segnale data una tachimetrica.

```
clear
clc

%% Creazione del segnale
N = 2^18;                   % Numero di punti in cui ? diviso il segnale
T = 60;             % Periodo del segnale [s]
fs = N/T;                   % Freqeunza di campionamento [Hz]
t = [0:N-1]/fs;             % Vettore dei tempi [s]

fr1 = 1;            % Frequenza della prima componente
fr2 = 37.5;                 % Frequenza della seconda componente

% Segnale con due componenti pi\'u rumore
x = sin(2*pi*fr1*t) + sin(2*pi*fr2*t) + randn(1,N);

%% Generazione della tachimetrica alla frequenza fr1
tacho = 1 + square(2*pi*fr1*t);

%% Filtraggio del segnale per evitare l'aliasing
M = 512;                    % Numero di punti per giro dell'albero
fcut = M * fr1;
B = fir1(100,fcut/fs);
xf = filtfilt(B,1,x);

%% Calcolo della media sincrona a fr1
% Calcolo dell'indice di superamento della soglia
triglevel = 1; % livello del triggher
test = tacho>triglevel; % restituisce 1 o 0 in caso sia vera o falsa
test = diff(test); % restituisce 1 per il superamento in salita e -1 per
il superamento in discesa
test=(test > 0.5); % tiene solo il superamento in salita
ind = find(test); % trova gli indici del superamento in salita della soglia
indup = ind+1; %indice subito dopo il superamento della soglia

% Calcolo della media sincrona con un delta O di 1
deltaangolo = 360/M;
angoli = [0:M-1]*deltaangolo;

endk = length(indup) - 1;
```

```
cicli_x = zeros(endk,length(angoli));

for k = 1:endk,
    ciclo = x(1,indup(k):indup(k+1)-1);
    tciclo = [0:length(ciclo)-1]*(360/length(ciclo));
    cicli_x(k,:) = interp1(tciclo,ciclo,angoli,'spline');
end
% vettore che contiene la media sincrona fatta su un giro
media_sinc_x = mean(cicli_x);

%% Plot dei risultati
% Segnale iniziale
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')
xlim([0 2.5])
ylim([-6 6])

% Tacho
indici_tacho_plot = zeros(1,N);
indici_tacho_plot(indup) = 1;
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,tacho,'k')
xlabel('time [s]')
ylabel('amplitude')
hold on
plot(t,indici_tacho_plot,'or')
xlim([0 2.5])
ylim([0 3])

% Media sincrona
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(angoli,media_sinc_x,'k')
xlabel('angle [deg]')
ylabel('amplitude')
xlim([0 360])
```

L'output dello script é mostrato in Fig. 4.

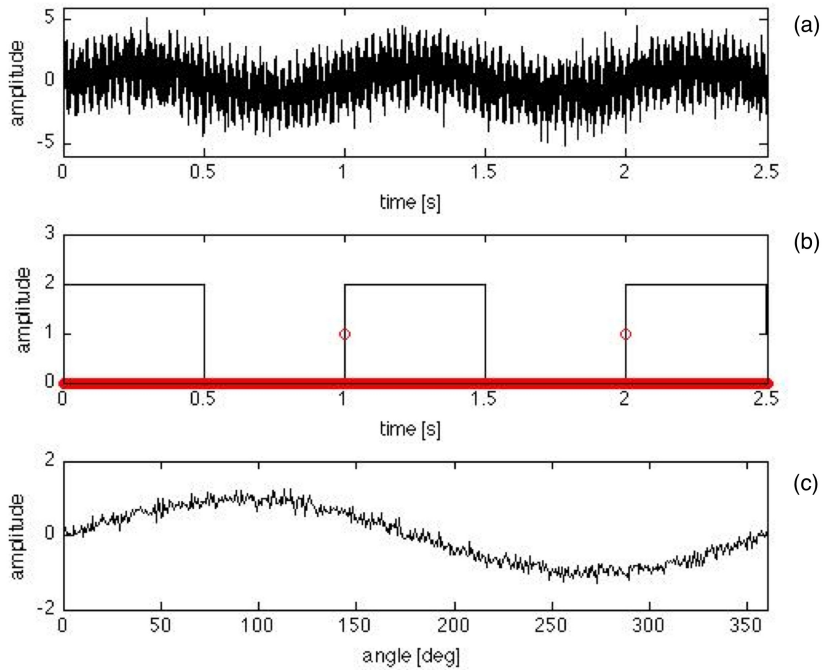

Figura 4: (a) segnale nel tempo, (b) tachimetrica con indici di superamento della soglia, (c) media sincrona

SEZIONE 5

Ciclostazionarietá

```
clear
clc

%% Generazione di un segnale del secondo ordine ciclostazionario
T = 6;
N = 2^15;
fs = N/T;

fciclica = 3;
t = (0:N-1)/fs;

A=1;                                % Ampiezza portante
ee=0.3;                             % Rapporto tra Ampiezza modulante e Ampiezza portante

modul=ee*A*sin(2*pi*fciclica*t);    % Modulante
x = (A+modul).*randn(1,N);

% x = randn(1,N);
% x = x.*sin(pi*fciclica*t);

figure,
```

```
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
plot(t,x,'k'),
xlabel('Time [s]'),
ylabel('Amplitude'),

%% Evaluatoin of the Spectrum
X = 1/N*fft(x);
f = (0:N-1)*fs/N;

figure,
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
plot(f,abs(X),'k')
xlim([0 f(end)])
xlabel('Frequenza [Hz]')
ylabel('Ampiezza')

%% Mean instantaneous power
Spectrum = 1/N.*fft(x.^2);
alpha = (0:N-1)*fs/N;

figure,
set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
plot(alpha,abs(Spectrum),'k')
xlabel('Frequenza Ciclica [Hz]')
ylabel('Ampiezza')
xlim([0 10])
ylim([0 0.35])


Ialpha = find(abs(Spectrum)>0.1);

Spectrum_alpha = zeros(1,N);
Spectrum_alpha(Ialpha) = Spectrum(Ialpha);

Px = N*ifft(Spectrum_alpha);

figure,
plot(t,x,'k')
hold on
plot(t,Px,'r')

set(gcf,'Units','centimeters')
set(gcf,'Position',[2 10 16 4])
xlabel('Time [s]'),
ylabel('Amplitude'),
```

```
%% Instantaneous Power spectrum
fracovlp = .75;      % fraction of overlap (should be greater than or equal to .75)
Nw = 2^5;
Nfft = Nw;

Noverlap = fix(fracovlp*Nw);
R = Nw - Noverlap;
Win = hanning(Nw);
[Px_alpha,f,t] = specgram(x,Nfft,fs,Win,Noverlap);
Px_alpha = abs(Px_alpha).^2;

Ti = find(t>0 & t<2);
figure,
mesh(t(Ti),f,Px_alpha(:,Ti)), colormap(gray)
view(-30,86)
xlabel(['Time [s] ; \Delta t=',num2str(diff(t(1:2))),' s']),
ylabel(['f [Hz] ; \Delta f=',num2str(diff(f(1:2))),' Hz']),


%% Cyclic modulation Spectrum
fracovlp = .75;      % fraction of overlap (should be greater than or equal to .75)
Nw = 2^5;
Nfft = Nw;

Noverlap = fix(fracovlp*Nw);
R = Nw - Noverlap;
Win = hanning(Nw);
[Stf,f,t] = specgram(x,Nfft,fs,Win,Noverlap);
Lt = length(t);

Stf = abs(Stf).^2;
Saf = (fft(Stf',Lt))';
Saf = Saf(1:Nfft/2,1:fix(Lt/2));

% Plot cms
f = f(1:Nfft/2);
a = (0:fix(Lt/2)-1)*fs/(Lt*Nw*(1-fracovlp));
la = find(a>0 & a<20);

figure, mesh(a(la),f(2:end),abs(Saf(2:end,la))), colormap(gray)
view(-26,38)
xlabel(['\alpha [Hz] ; \Delta \alpha = ',num2str(diff(a(1:2))),' Hz'])
ylabel(['f [Hz] ; \Delta f = ',num2str(f(2)),' Hz'])
title('Cyclic Modulation Spectrum'),
```

L'output é mostrato in Fig. 5.

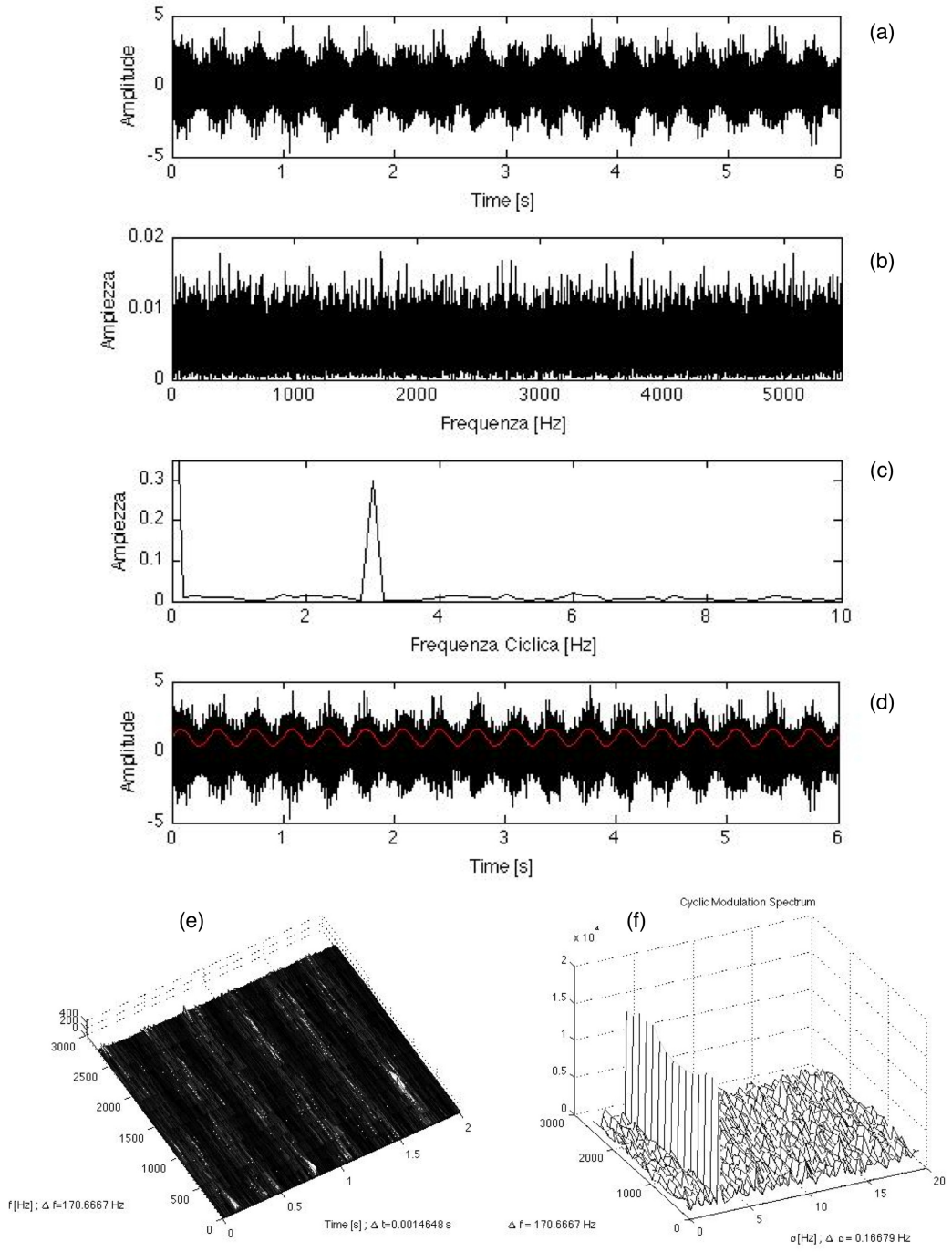

Figura 5: (a) segnale nel tempo, (b) spettro, (c) cyclic spectrum, (d) mean instantaneous power, (e) instantaneous power spectrum, (f) cyclic modulation spectrum