Matlab mette a disposizione la funzione fft che fornisce la DFT del vettore X calcolata mediante l’algoritmo “fast Fourier transform (FFT)»,

fft

Fast Fourier transform

collaps

Syntax

```
Y = fft(X)
Y = fft(X,n)
Y = fft(X,n,dim)
```

Description

`Y = fft(X)` computes the discrete Fourier transform (DFT) of `X` using a fast Fourier transform (FFT) algorithm.

`Y = fft(X,n)` returns the `n`-point DFT. If no value is specified, `Y` is the same size as `X`.

`Y = fft(X,n,dim)` returns the Fourier transform along the dimension `dim`. For example, if `X` is a matrix, then `fft(X,n,2)` returns the n-point Fourier transform of each row.

La funzione ifft matlab fornisce la *DFT inversa* del vettore X calcolata mediante l'algoritmo "fast Fourier transform (FFT)".

ifft

R202(

Inverse fast Fourier transform

collapse all in p

Syntax

```
X = ifft(Y)
X = ifft(Y,n)
X = ifft(Y,n,dim)
```

Algorithms

The FFT functions are based on a library called FFTW [1] [2].

References

[1] FFTW (http://www.fftw.org)

[2] Frigo, M., and S. G. Johnson. "FFTW: An Adaptive Software Architecture for the FFT." *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3, 1998, pp. 1381-1384.

Trasformata Discreta di Fourier (DFT)

OSSERVAZIONE:

F_S =1/ Δ =N/T =frequenza di campionamento (numero campioni per unità di tempo)in Hz

1/T = è il minimo valore del passo di discretizzazione con cui si può discretizzare f(t).

in termini di frequenze:

$1/T=F_S/N$ (frequenza fondamentale), è la minima distanza alla quale si devono trovare due frequenze per riuscire a distinguerle nel campionamento di f(t)

Tutte le frequenze sono multipli interi di 1/T: k/T. La massima frequenza di f che può essere calcolata si dimostra che è per k=N/2 :

$$FN=1/(2\Delta)=N/(2T)=$$ frequenza di Nyquist

Applichiamo la function fft e ifft con N =2^m con m=3 (radix-2)

Per la function fft abbiamo bisogno come input un vettore di dimensione n,

Scegliamo il seguente vettore di dimensione 2^3=8

```
f=[4 3 7 -9 10 6 1 9]';
```

Applichiamo la function fft e ifft con N =2^m con m=3 (radix-2)

Per la function fft abbiamo bisogno come input un vettore di dimensione n,

Scegliamo il seguente vettore di dimensione 2^3=8

```
f=[4 3 7 -9 10 6 1 9]';
Richiamiamo la function fft con f dato di
input

y=fft(f)
```
% function Matlab che calcola la DFT

```
>> ese1
y =
 31.0000 + 0.0000i
  4.6066 - 8.8492i
  6.0000 + 9.0000i
-16.6066 -20.8492i
 13.0000 + 0.0000i
-16.6066 +20.8492i
  6.0000 - 9.0000i
  4.6066 + 8.8492i
```

Applichiamo la function fft e ifft con N =2^m con m=3 (radix-2)

Per la function fft abbiamo bisogno come input un vettore di dimensione n,

Scegliamo il seguente vettore di dimensione 2^3=8

```
f=[4 3 7 -9 10 6 1 9]';
Richiamiamo la function fft con f dato di
input

y=fft(f)
```
% function Matlab che calcola la DFT

```
>> ese1
y =
 31.0000 + 0.0000i
  4.6066 - 8.8492i
  6.0000 + 9.0000i
-16.6066 -20.8492i
 13.0000 + 0.0000i
-16.6066 +20.8492i
  6.0000 - 9.0000i
  4.6066 + 8.8492i
```

Nyquist

Simmetria

Applichiamo la function fft e ifft con N =2^m con m=3 (radix-2)

Per la function fft abbiamo bisogno come input un vettore di dimensione n,

Scegliamo il seguente vettore di dimensione 2^3=8

```
f=[4 3 7 -9 10 6 1 9]';
Richiamiamo la function fft con f dato di
input

y=fft(f)
```
% function Matlab che calcola la DFT

```
>> ese1
y =
 31.0000 + 0.0000i
  4.6066 - 8.8492i
  6.0000 + 9.0000i
-16.6066 -20.8492i
 13.0000 + 0.0000i
-16.6066 +20.8492i
  6.0000 - 9.0000i
  4.6066 + 8.8492i
```

Nyquist

simmetria

Richiamiamo la function ifft per ottenere la trasformate inversa

```
iy=ifft(y)
```

```
iy=
 4.0000
 3.0000
 7.0000
-9.0000
10.0000
 6.0000
 1.0000
 9.0000
```

Applichiamo la function fft e ifft con N =2^m con m=3 (radix-2)

Per la function fft abbiamo bisogno come input un vettore di dimensione n,

Scegliamo il seguente vettore di dimensione 2^3=8

```
f=[4 3 7 -9 10 6 1 9]';
Richiamiamo la function fft con f dato di input

y=fft(f)  % function Matlab che calcola la DFT
```

```
>> ese1
y =
 31.0000 + 0.0000i
  4.6066 - 8.8492i
  6.0000 + 9.0000i
-16.6066 -20.8492i
 13.0000 + 0.0000i   Nyquist
-16.6066 +20.8492i
  6.0000 - 9.0000i
  4.6066 + 8.8492i
```

simmetria

Richiamiamo la function ifft per ottenere la trasformate inversa

```
iy=ifft(y)
```

```
iy=
 4.0000
 3.0000
 7.0000
-9.0000
10.0000
 6.0000
 1.0000
 9.0000
```

Applichiamo la function fft e ifft con N =2^m con m=3 (radix-2)

Per la function fft abbiamo bisogno come input un vettore di dimensione n,

Scegliamo il seguente vettore di dimensione 2^3=8

```
f=[4 3 7 -9 10 6 1 9]';
Richiamiamo la function fft con f dato di input
```

`y=fft(f)` % function Matlab che calcola la DFT

Richiamiamo la function ifft per ottenere la trasformate inversa

```
iy=ifft(y)
```

verifichiamo che si riottene il vettore di partenza

```
>> ese1
y =
 31.0000 + 0.0000i
  4.6066 - 8.8492i
  6.0000 + 9.0000i
-16.6066 -20.8492i
 13.0000 + 0.0000i
-16.6066 +20.8492i
  6.0000 - 9.0000i
  4.6066 + 8.8492i
```

Nyquist

simmetria

```
iy=
 4.0000
 3.0000
 7.0000
-9.0000
10.0000
 6.0000
 1.0000
 9.0000
```

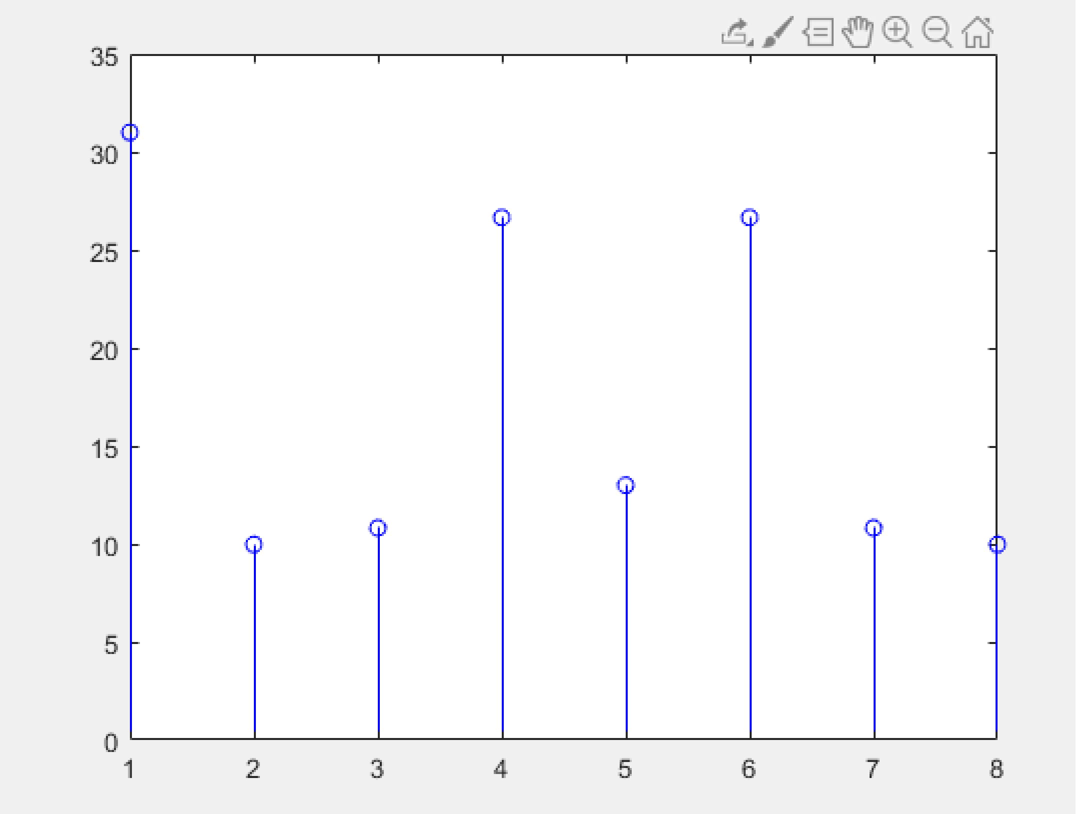
periodogramma

Grafico delle ampiezze rispetto alle frequenze

F_k complesso ⇒ **Si visualizza il modulo = $|F_k|$**

Poiché F_k è complesso si preferisce visualizzarne l'istogramma, che è il grafico dell'*ampiezza* (modulo)= $|F_k|$ rispetto le frequenze

```
stem(1:8,abs(y),'b')
```



Si osserva la simmetria intorno alla frequenza di Nyquist, tranne che per il primo punto.

Applichiamo la function fft e ifft con N =3^m con m=2 (radix-3)

```
Consideriamo il seguente vettore di
Dimensione 3^2=9

f=[4 3 7 -9 1 0 0 0 5]';
Richiamo la function matlab fft
y=fft(f);

Grafico funzione (istogramma)
stem(1:9, abs(y))
```

Applichiamo la function fft e ifft con N =3^m con m=2 (radix-3)

```
>> f=[4 3 7 -9 1 0 0 0 5];  %N=9
>> y=fft(f)
y =
  11.0000
  14.9042 + 1.8441i
   4.0774 - 7.5760i
 -13.0000 + 6.9282i
   6.5184 +13.9626i
   6.5184 -13.9626i
 -13.0000 - 6.9282i
   4.0774 + 7.5760i
  14.9042 - 1.8441i
```

Applichiamo la function fft e ifft con N =3^m con m=2 (radix-3)

```
>> f=[4 3 7 -9 1 0 0 0 5];  %N=9
>> y=fft(f)
y =
  11.0000
  14.9042 + 1.8441i
   4.0774 - 7.5760i
 -13.0000 + 6.9282i
   6.5184 +13.9626i
   6.5184 -13.9626i
 -13.0000 - 6.9282i
   4.0774 + 7.5760i
  14.9042 - 1.8441i
```

Nyquist non c'è
ma c'è la simmetria

Applichiamo la function fft e ifft con N =3^m con m=2 (radix-3)

Consideriamo il seguente vettore di
Dimensione 3^2=9

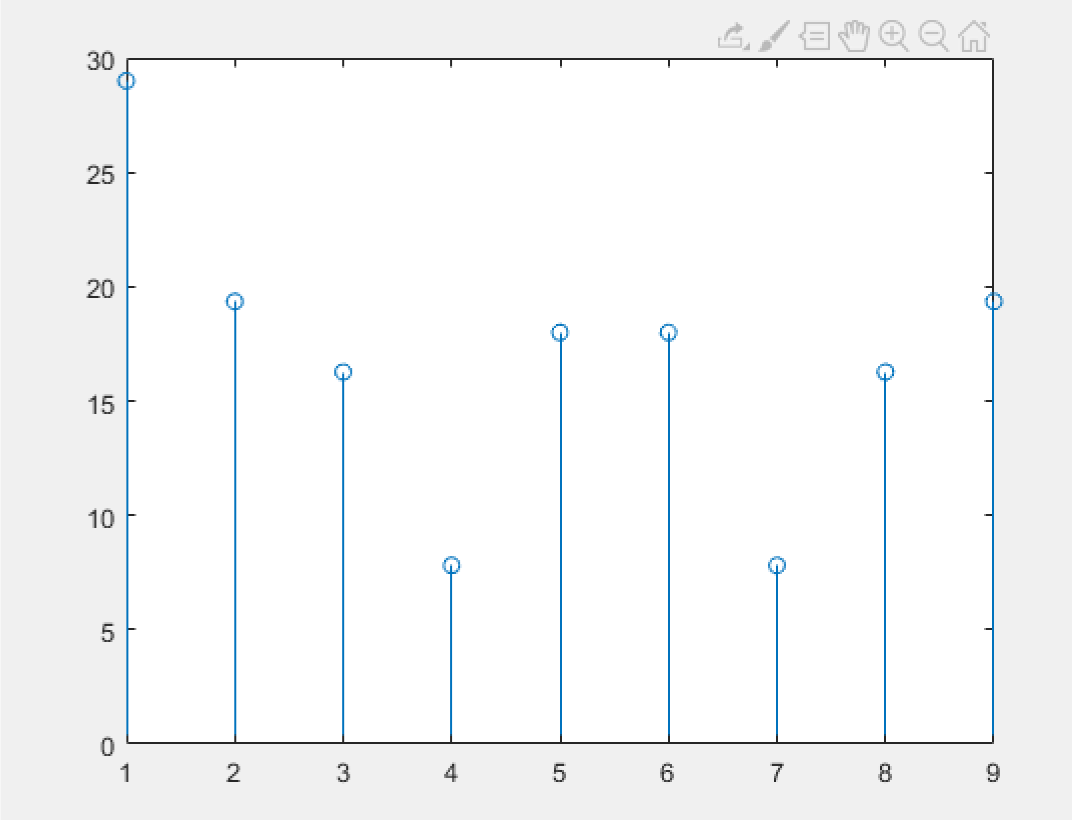
```
f=[4 3 7 -9 1 0 0 0 5]';
```

Richiamo la function matlab fft

```
y=fft(f);
```

Grafico funzione (istogramma)

```
stem(1:9, abs(y))
```



Si osserva la simmetria tranne che per il primo punto.

$f = (f_0, f_1, \ldots, f_{N-1})$ reale

la DFT ha modulo con simmetria pari intorno alla frequenza di Nyquist $\Rightarrow$

$real(F_k) = real(F_{N-k})$

$imag(F_k) = -imag(F_{N-k})$

$k = 1, .., N/2$

$F_{N-k} = \overline{F_k}$ complesso coniugato di F_k

$$F_0 \; F_1 \; F_2 \ldots F_{(N/2)-1} \; F_{N/2} \; \overline{F_{(N/2)-1}} \ldots \overline{F_2} \; \overline{F_1}$$

Consideriamo f(t) periodica

Applichiamo la function fft con N =12=4*3 (mixed-radix)

Consideriamo la funzione f(t)=sin(2*pi*t);

Per richiamare la function fft cosa ci serve?
Un vettore di lunghezza N .
Lo otteniamo a partire dalla funzione f(t).

Supposto di voler ottenere un vettore di lunghezza N=12
Definiamo 12 punti nell'intervallo, ad esempio, [0,1]

```
n=12;
dt=1/(n-1);
t=[0:dt:1];
%segnale
ft=sin(2*pi*t);
```

Richiamo la Function fft

```
fftft=fft(ft);
%grafico del segnale
figure(1)
plot(t,ft)
```

Applichiamo la function fft con N =12=4*3 (mixed-radix)

Consideriamo la funzione f(t)=sin(2*pi*t);

Per richiamare la function fft cosa ci serve?
Un vettore di lunghezza N .
Lo otteniamo a partire dalla funzione f(t).

Supposto di voler ottenere un vettore di lunghezza N=12
Definiamo 12 punti nell'intervallo, ad esempio, [0,1]

```
n=12;
dt=1/(n-1);
t=[0:dt:1];
%segnale
ft=sin(2*pi*t);
```

Richiamo la Function fft

```
fftft=fft(ft);
%grafico del segnale
figure(1)
plot(t,ft)
```

Dobbiamo calcolare le frequenza

Applichiamo la function fft con N =12=4*3 (mixed-radix)

Consideriamo la funzione f(t)=sin(2*pi*t);

Per richiamare la function fft cosa ci serve?
Un vettore di lunghezza N .
Lo otteniamo a partire dalla funzione f(t).

Supposto di voler ottenere un vettore di lunghezza N=12
Definiamo 12 punti nell'intervallo, ad esempio, [0,1]

```
n=12;
dt=1/(n-1);
t=[0:dt:1];
%segnale
ft=sin(2*pi*t);
```

Richiamo la Function fft

```
fftft=fft(ft);
%grafico del segnale
figure(1)
plot(t,ft)
```

Dobbiamo calcolare le frequenza.
Come?
Sono multipli della freq. Fondamentale

Applichiamo la function fft con N =12=4*3 (mixed-radix)

Consideriamo la funzione f(t)=sin(2*pi*t);

Per richiamare la function fft cosa ci serve?
Un vettore di lunghezza N .
Lo otteniamo a partire dalla funzione f(t).

Supposto di voler ottenere un vettore di lunghezza N=12
Definiamo 12 punti nell'intervallo, ad esempio, [0,1]

```
n=12;
dt=1/(n-1);
t=[0:dt:1];
%segnale
ft=sin(2*pi*t);
```

Richiamo la Function fft

```
fftft=fft(ft);
%grafico del segnale
figure(1)
plot(t,ft)
```

Dobbiamo calcolare le frequenza.
Come?
Sono multipli della freq. Fondamentale

```
FS=1/dt; %frequenza di campionamento
freq=(0:n/2)*FS/n; %multipli frequenza fondamentale
```

Applichiamo la function fft con N =12=4*3 (mixed-radix)

Consideriamo la funzione f(t)=sin(2*pi*t);

Per richiamare la function fft cosa ci serve?
Un vettore di lunghezza N .
Lo otteniamo a partire dalla funzione f(t).

Supposto di voler ottenere un vettore di lunghezza N=12
Definiamo 12 punti nell'intervallo, ad esempio, [0,1]

```
n=12;
dt=1/(n-1);
t=[0:dt:1];
%segnale
ft=sin(2*pi*t);
```

Richiamo la Function fft

```
fftft=fft(ft);
%grafico del segnale
figure(1)
plot(t,ft)
```

Dobbiamo calcolare le frequenza.
Come?
Sono multipli della freq. Fondamentale

```
FS=1/dt; %frequenza di campionamento
freq=(0:n/2)*FS/n; %multipli frequenza
fondamentale
```

Grafico

```
figure(2)  %grafico istogramma
stem(freq,abs(fftft(1:n/2+1)),'b')
xlabel('freq')
ylabel('abs(fft)')
```

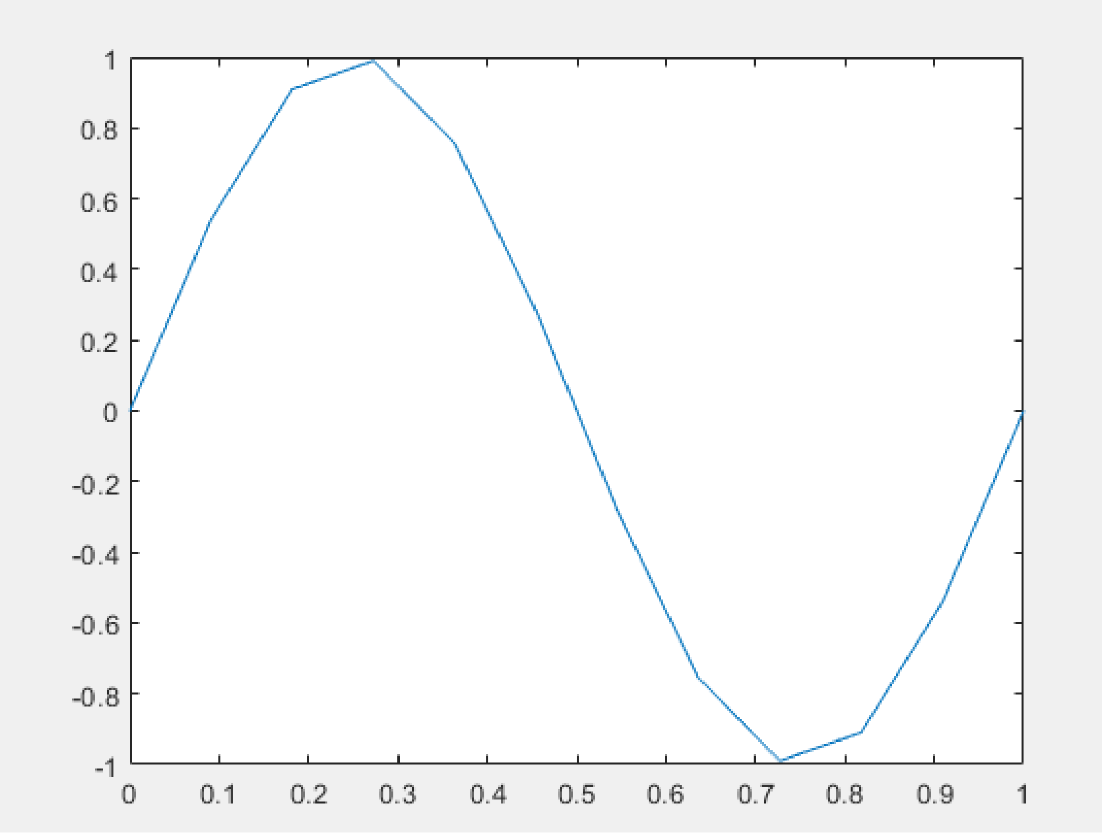
Applichiamo la function fft con N =12=4*3 (mixed-radix)

Output fft

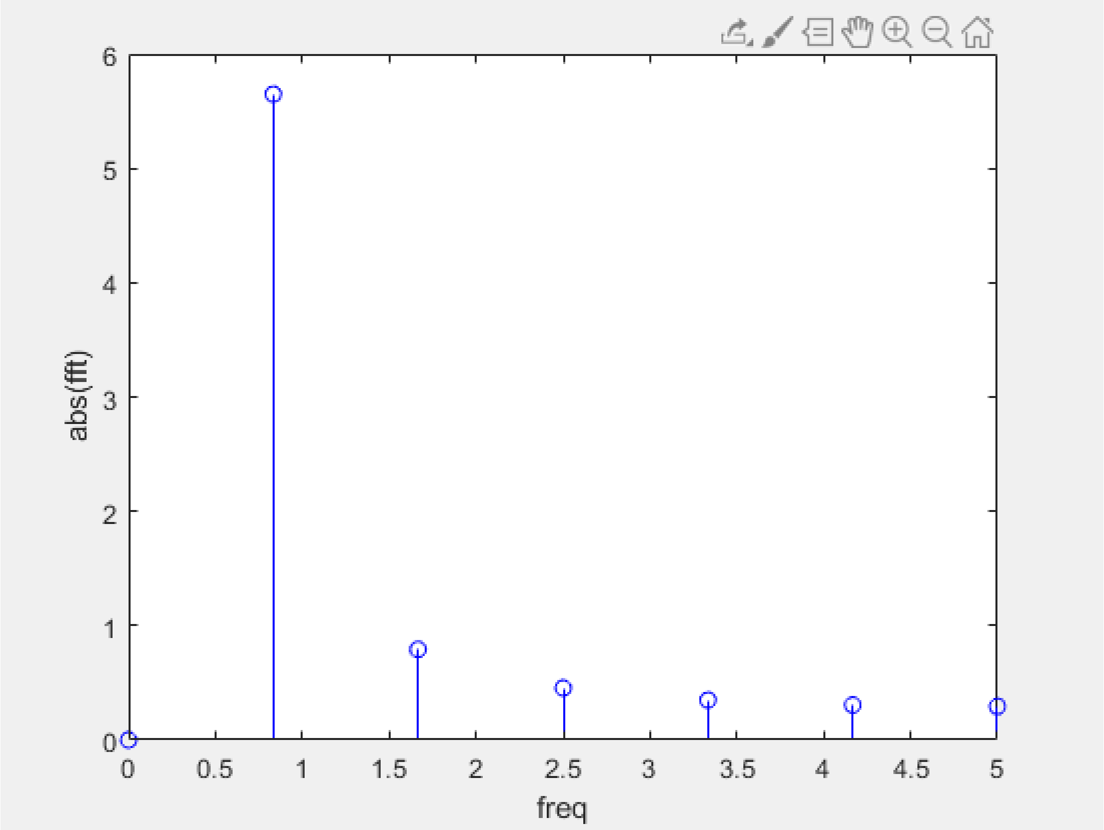
```
>> fftft'

ans =

  -0.0000 + 0.0000i
   1.4620 + 5.4562i
  -0.3961 - 0.6860i
  -0.3213 - 0.3213i
  -0.3023 - 0.1745i
  -0.2955 - 0.0792i
  -0.2936 + 0.0000i
  -0.2955 + 0.0792i
  -0.3023 + 0.1745i
  -0.3213 + 0.3213i
  -0.3961 + 0.6860i
   1.4620 - 5.4562i
```

Applichiamo la function fft con N =12=4*3 (mixed-radix)

Segnale



Abs(fft)



Esempio ffft per N =1000=(5)^3*3^3

Consideriamo la funzione f(t)=sin(2*pi*50*t)+ sin(2*pi*120*t);

Anziché definire N a priori possiamo definire
Il periodo T e la frequenza di campionamento, ovvero:

```
%consideriamo un segnale in [0,1]
T=1;
Fs=1000;     %freq.campionamento
```

Ricavare N come segue

Esempio ffft per N =1000=(5)^3*3^3

Consideriamo la funzione f(t)=sin(2*pi*50*t)+ sin(2*pi*120*t);

Anziché definire N a priori possiamo definire
Il periodo T e la frequenza di campionamento, ovvero:

```
%consideriamo un segnale in [0,1]
T=1;
Fs=1000;     %freq.campionamento
```

Ricavare N come segue

N=1000

```
N=Fs*T;    % lunghezza del segnale
```

Punti in cui valutare il segnale

```
t =(0:1/Fs:N-1);      % punti di discretizzazione
%segnale campionato : somma di due armoniche a 50 Hz e
120 Hz
x = 3*sin(2*pi*50*t) + sin(2*pi*120*t);


figure(1)
plot(t,x),title('segnale nel tempo')
```

Esempio ffft per N =1000=(5)^3*3^3

Consideriamo la funzione f(t)=sin(2*pi*50*t)+ sin(2*pi*120*t);

Anziché definire N a priori possiamo definire
Il periodo T e la frequenza di campionamento, ovvero:

```
%consideriamo un segnale in [0,1]
T=1;
Fs=1000;     %freq.campionamento
```

Ricavare N come segue

```
N=Fs*T;     % lunghezza del segnale

t =(0:1/Fs:N-1);      % punti di discretizzazione
%segnale campionato : somma di due armoniche a 50 Hz e 120 Hz
x = 3*sin(2*pi*50*t) + sin(2*pi*120*t);

figure(1)
plot(t,x),title('segnale nel tempo')
```

```
Y=fft(x);
% per simmetria  si considerano le frequenze da 0 a N/2
freq=(0:floor(N/2))*Fs/N;
```

```
%l’ampiezza che si ottiene con fft è A*(N/2) , dove A è l’ampiezza
figure(2)
amp=2*abs(Y(1:floor(N/2+1))/N);
plot(freq,amp),
title('spettro del segnale’);
xlabel('frequenze')
```

Esempio ffft per N =1000=(5)^3*3^3

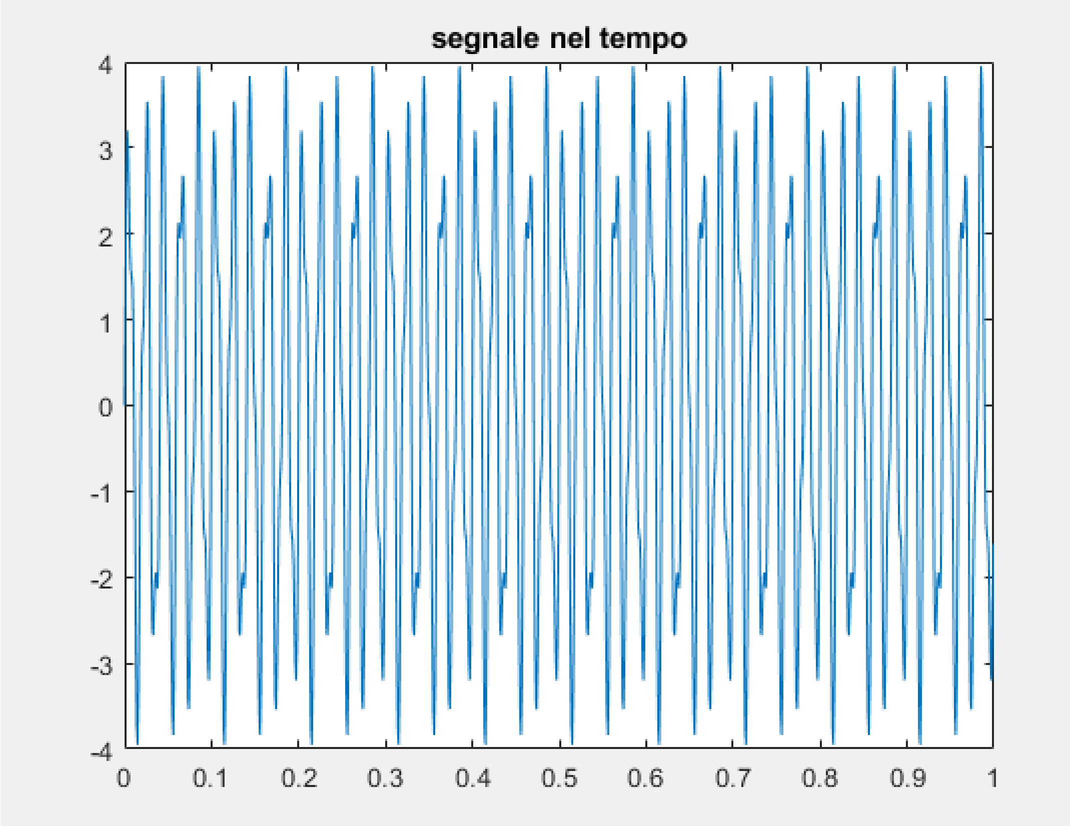
Consideriamo la funzione f(t)=sin(2*pi*50*t)+ sin(2*pi*120*t);

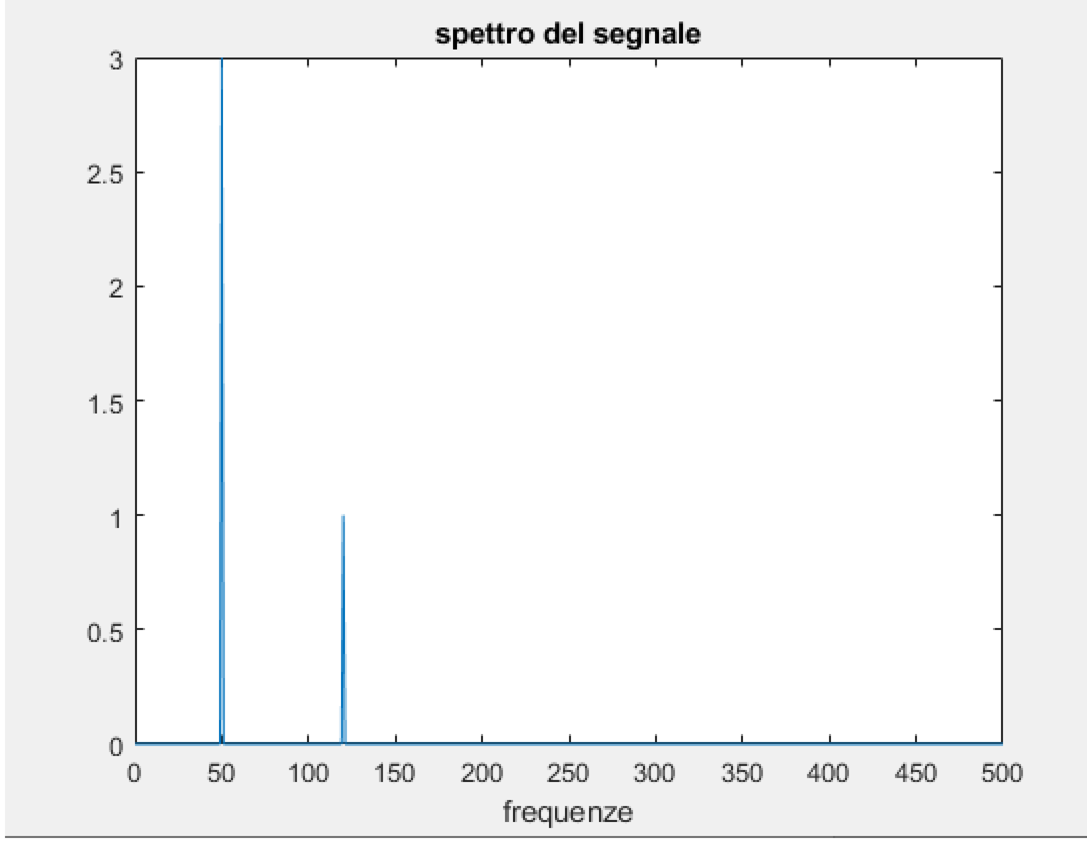
Anziché definire n a priori possiamo definire
Il periodo T e la frequenza di campionamento, ovvero:

```
%consideriamo un segnale in [0,1]
T=1;
Fs=1000;     %freq.campionamento
N=Fs*T;      % lunghezza del segnale

t =(0:1/Fs:N-1);      % punti di discretizzazione
%segnale campionato : somma di due armoniche a 50 Hz e 120 Hz
x = 3*sin(2*pi*50*t) + sin(2*pi*120*t);

figure(1)
plot(t,x),title('segnale nel tempo')
```

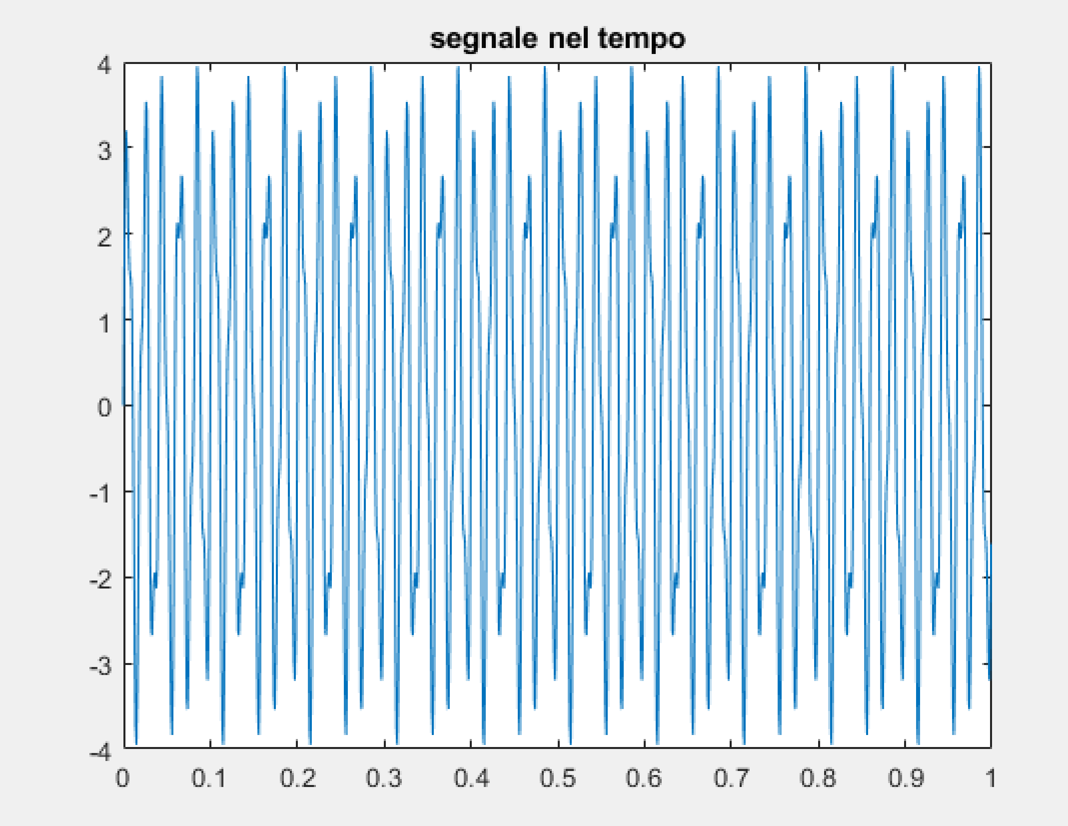
```
figure(2)
Y=fft(x);
% per simmetria  si considerano le frequenze da 0 a N/2
freq=(0:floor(N/2))*Fs/N;

%l'ampiezza che si ottiene con fft è A*(N/2) , dove A è l'ampiezza
amp=2*abs(Y(1:floor(N/2+1))/N);
plot(freq,amp),title('spettro del segnale');xlabel('frequenze')
```

segnale


segnale nel tempo
4
3
2
1
0
-1
-2
-3
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


spettro del segnale
3
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250
300
350
400
450
500
frequenze

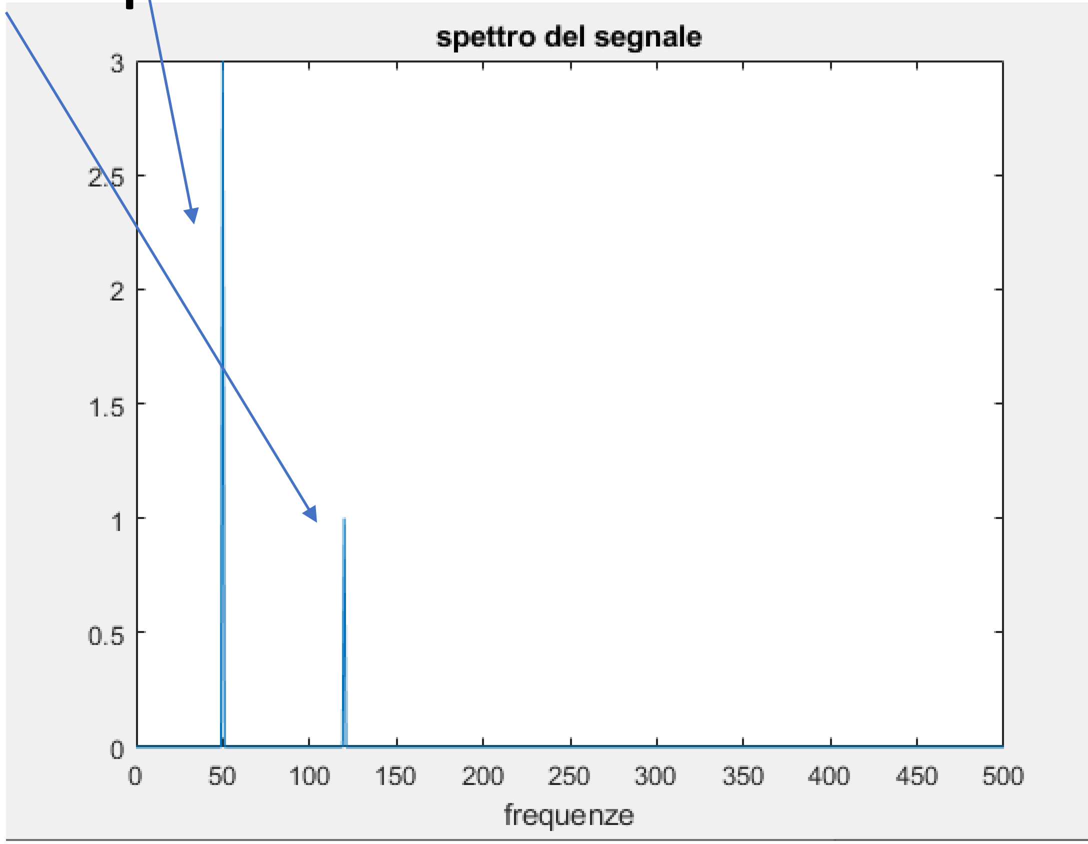
segnale



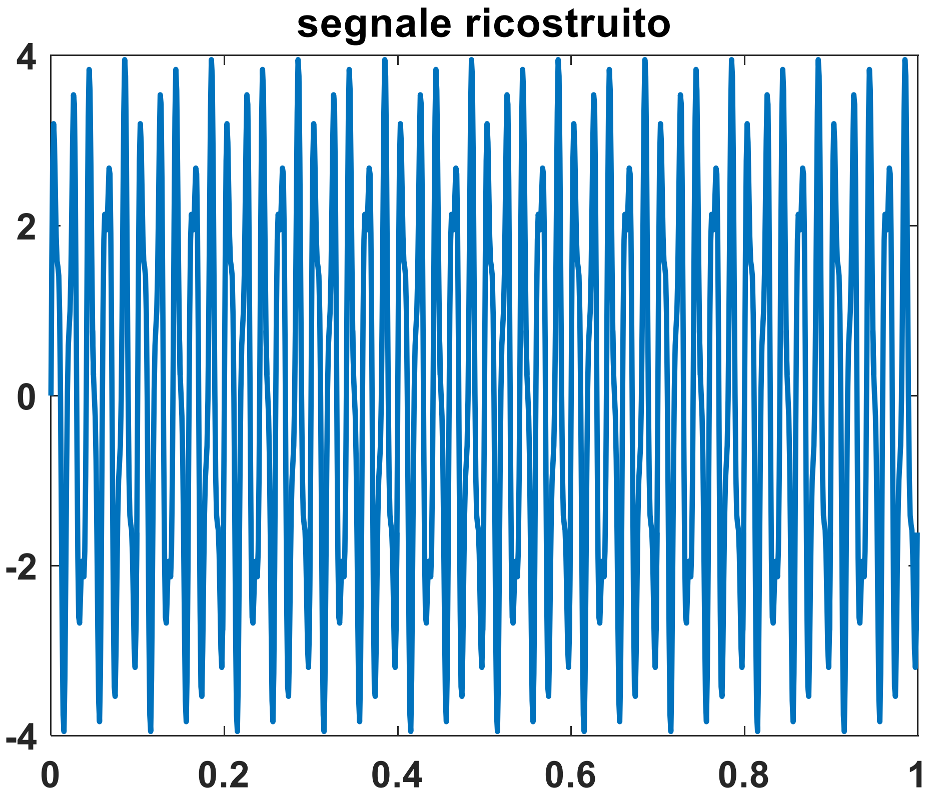
Periodogramma: consente di visualizzare le componenti in frequenza:

Freq=50H amp=3

Freq=120H amp=1



```
%ricostruzione del segnale con la trasformata inversa :
xn=ifft(Y);
figure(3)
plot(t,xn),axis([0 1 -2 2]),title('segnale ricostruito')
```



Esempio ffft per N =997 (primo)

Consideriamo la funzione f(t)=sin(2*pi*50*t)+ sin(2*pi*120*t);

Anziché definire n a priori possiamo definire
Il periodo T e la frequenza campionaria, ovvero:

```
%consideriamo un segnale in [0,1]
T=1;
Fs=997;     %freq.campionamento
N=Fs*T;     % lunghezza del segnale

t =(0:1/Fs:N-1);       % punti di discretizzazione
%segnale campionato : somma di due armoniche a 50 Hz e 120 Hz
x = 3*sin(2*pi*50*t) + sin(2*pi*120*t);

figure(1)
plot(t,x),title('segnale nel tempo')
```

```
figure(2)
Y=fft(x);
% per simmetria  si considerano le frequenze da 0 a N/2
freq=(0:floor(N/2))*Fs/N;

%l'ampiezza che si ottiene con fft è A*(N/2) , dove A è l'ampiezza
amp=2*abs(Y(1:floor(N/2+1))/N);
plot(freq,amp),title('spettro del segnale');xlabel('frequenze')
```

Esempio ffft per N =997 (primo)

Consideriamo la funzione f(t)=sin(2*pi*50*t)+ sin(2*pi*120*t);

Anziché definire n a priori possiamo definire
Il periodo T e la frequenza di campionamento, ovvero:

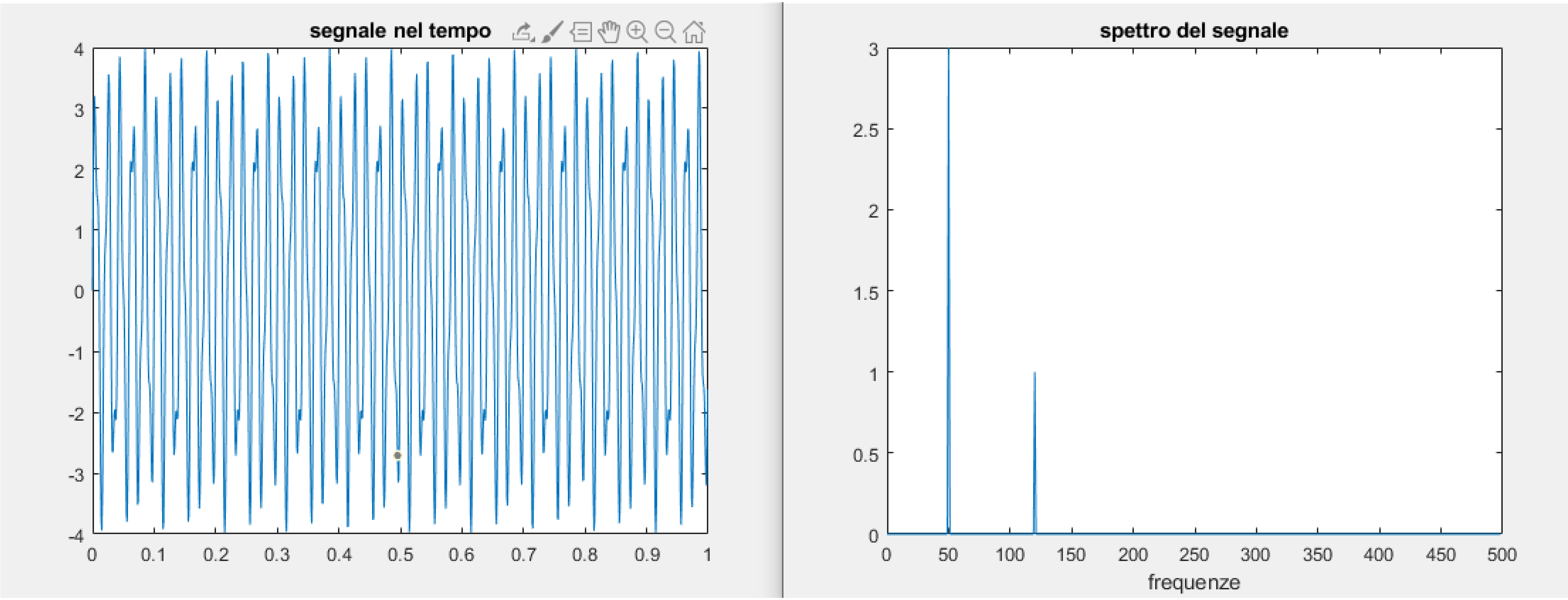
```
%consideriamo un segnale in [0,1]
T=1;
Fs=997;    %freq.campionamento
N=Fs*T;    % lunghezza del segnale
```

997

```
t =(0:1/Fs:N-1);      % punti di discretizzazione
%segnale campionato : somma di due armoniche a 50 Hz e 120 Hz
x = 3*sin(2*pi*50*t) + sin(2*pi*120*t);

figure(1)
plot(t,x),title('segnale nel tempo')
```

```
figure(2)
Y=fft(x);
% per simmetria si considerano le frequenze da 0 a N/2
freq=(0:floor(N/2))*Fs/N;

%l'ampiezza che si ottiene con fft è A*(N/2) , dove A è l'ampiezza
amp=2*abs(Y(1:floor(N/2+1))/N);
plot(freq,amp),title('spettro del segnale');xlabel('frequenze')
```

Grafici


segnale nel tempo
4
3
2
1
0
-1
-2
-3
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
spettro del segnale
3
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250
300
350
400
450
500
frequenze

Esempio fft

Consideriamo un segnale periodico e calcoliamo la dft tramite fft e visualizziamo i grafici del segnale e del periodogramma.

Discretizzazione intervallo temporale

Segnale
Valutato nei punti t

```
%ESEMPIO FFT
%vettore con punti equidistanti con h=0.1
passo di discretizzazione
t=[0:0.1:1]
%segnale
ft=sin(2*pi*t);
```

Discretizzazione intervallo temporale

Segnale
Valutato nei punti t

Richiamo function fft

```
%ESEMPIO FFT
%vettore con punti equidistanti con h=0.1
passo di discretizzazione
t=[0:0.1:1]
%segnale
ft=sin(2*pi*t);
%applico la fft
fftft=fft(ft);
%grafico del segnale


figure(1)
plot(t,ft)
```

Discretizzazione intervallo temporale

Segnale
Valutato nei punti t

Richiamo function fft

Grafico segnale

```
%ESEMPIO FFT
%vettore con punti equidistanti con h=0.1
passo di discretizzazione
t=[0:0.1:1]
%segnale
ft=sin(2*pi*t);
%applico la fft
fftft=fft(ft);
%grafico del segnale


figure(1)
plot(t,ft)
```

Discretizzazione intervallo temporale

Segnale
Valutato nei punti t

Richiamo function fft

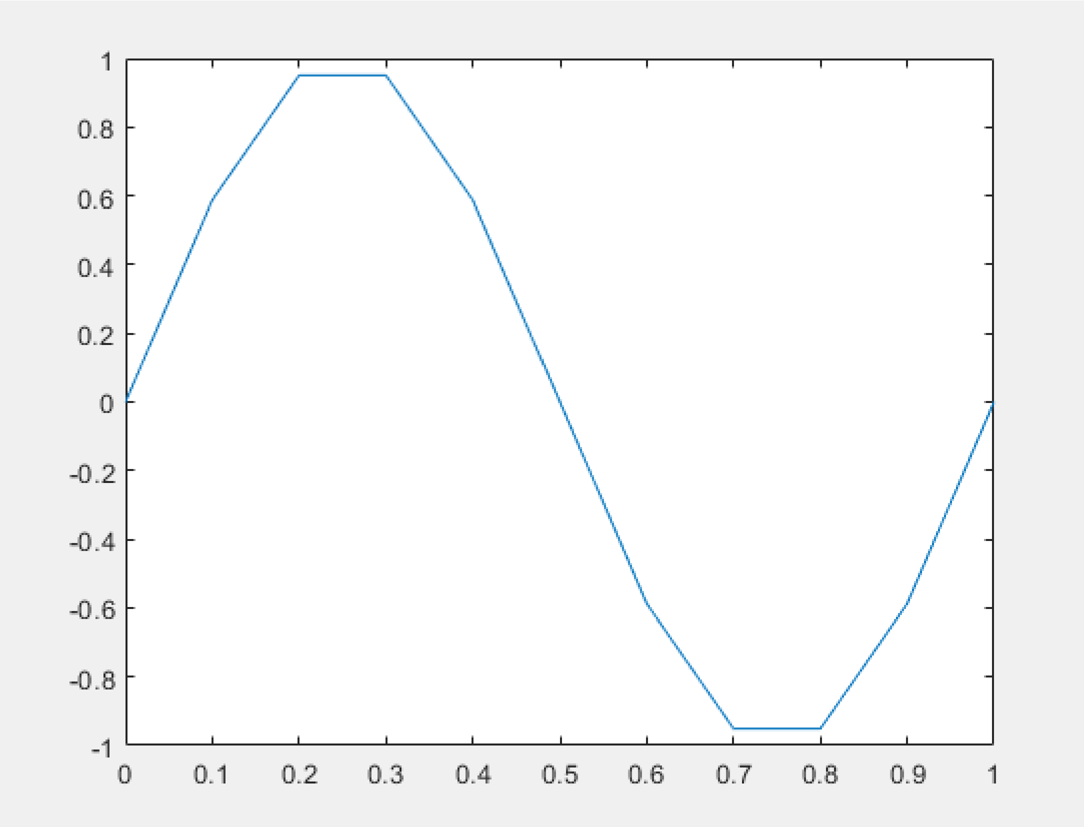
Grafico segnale

```
%ESEMPIO FFT
%vettore con punti equidistanti con h=0.1
passo di discretizzazione
t=[0:0.1:1]
%segnale
ft=sin(2*pi*t);
%applico la fft
fftft=fft(ft);
%grafico del segnale


figure(1)
plot(t,ft)

n=length(t);

dt=0.1; %passo di discretizzazione
FS=1/dt; %frequenza di campionamento
freq=(0:n/2)*FS/n; %multipli frequenza
fondamentale

figure(2)  %grafico ad istogramma
stem(freq,abs(fftft(1:n/2+1)),'b')
xlabel('freq')
ylabel('abs(fft)')
```

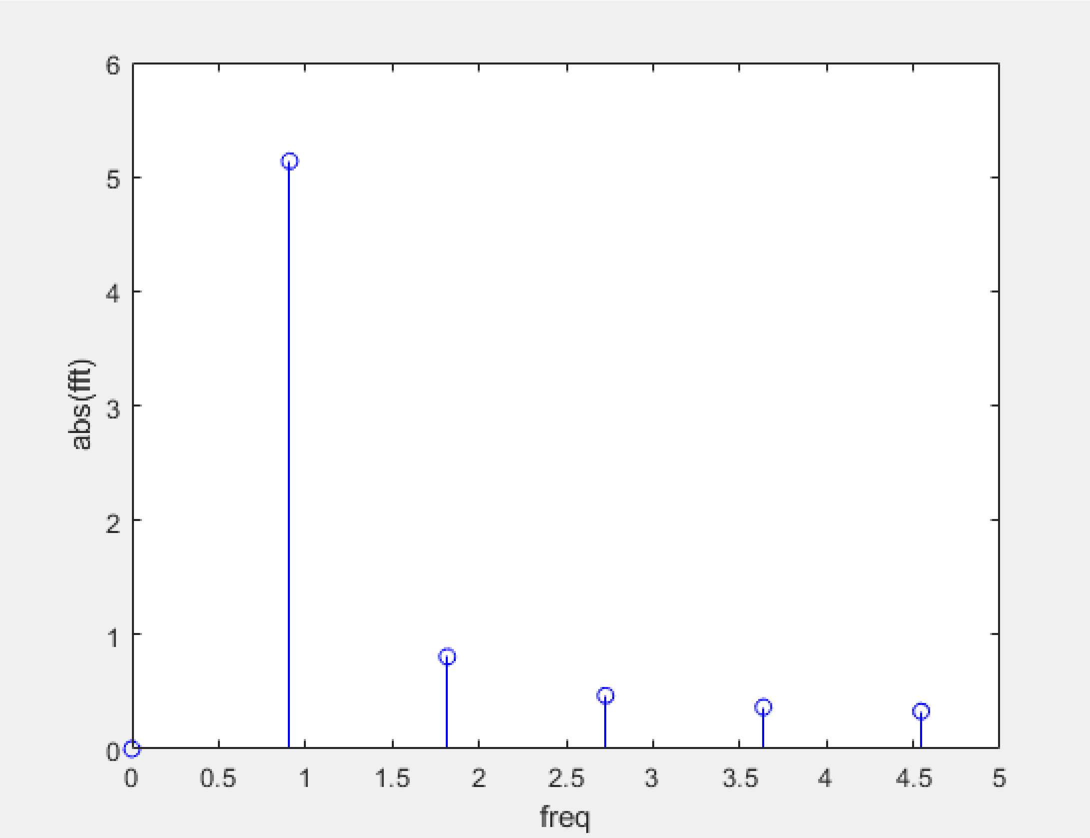
frequenze

Grafico istogramma

segnale



Abs(fft)



Esempio

Si consideri una funzione periodica x(t), combinazione lineare di due o più
funzioni sinusoidali.
Si sommi ad essa una funzione y(t) , anch’essa sinusoidale (rumore) .
Si discretizzi la funzione z(t) = x(t) + y(t)
Si applichi la DFT alla funzione z(t) per individuare la frequenze caratteristica del rumore e si fornisca la funzione x(t) .

```
%script per il filtraggio di un segnale


t=[0:0.0125:(0.2-0.0125)]
%segnale affetto da rumore
fnoised=sin(2*pi*5*t)+sin(2*pi*10*t)+5*sin(
2*pi*30*t);
```

f(t) y(t)

```
%grafio segnale
figure(1)
plot(t,fnoised)
%applico fft
fnoisedfft=fft(fnoised)
%%format long
n=length(t);
dt=0.0125; %passo di discretizzazione
FS=1/dt;  %frequenza di campionamento
freq=(0:n/2)*FS/n; %multipli della
frequenza fondamentale(metà per simmetria)


figure(2)
%istogramma
stem(freq,abs(fnoisedfft(1:n/2+1)),'b')
```

A questo punto, come avviene nella realtà, l’individuazione del rumore dipende dalle informazioni sul segnale e dagli scopi del filtraggio. Supponiamo di sapere che il segnale originale non può avere frequenze superiori a 10 Hz.

```
%script per il filtraggio di un segnale


t=[0:0.0125:(0.2-0.0125)]
%segnale affetto da rumore
fnoised=sin(2*pi*5*t)+sin(2*pi*10*t)+5*sin(
2*pi*30*t);

%g
fi
pl
%applico fft
fnoisedfft=fft(fnoised)
%%format long
n=length(t);
dt=0.0125; %passo di discretizzazione
FS=1/dt;  %frequenza di campionamento
freq=(0:n/2)*FS/n; %multipli della
frequenza fondamentale(metà per simmetria)


figure(2)
%istogramma
stem(freq,abs(fnoisedfft(1:n/2+1)),'b')
```

eliminare le frequenza più grandi di 10 Hz, si costruisce un filtro

```
%%rimuovere il rumore e ritornare al
segnale pulito non più affetto da rumore
%%, nell'istogramma al punto 30 si deve
togliere il rumore.

%eliminare le frequenza più grandi di 10
Hz, si costruisce un filtro

freqq=(0:n-1)*FS/n
%definizione del filtro
H=ones(1,length(freq));
H(freqq>10)=0; %elimina le frequenze
relative al rumore
H(n/2+1:n)=fliplr(H(1:n/2)) %copia la prima
metà del vettore H nella seconda metà
                            % la dft e il
filtro sono simmetrici rispetto alla
frequenza di Nyquist

%applicazione del filtro e ritorno nel
dominio del tempo
filtred=fnoisedfft.*H;
ify=ifft(filtred,n);
```

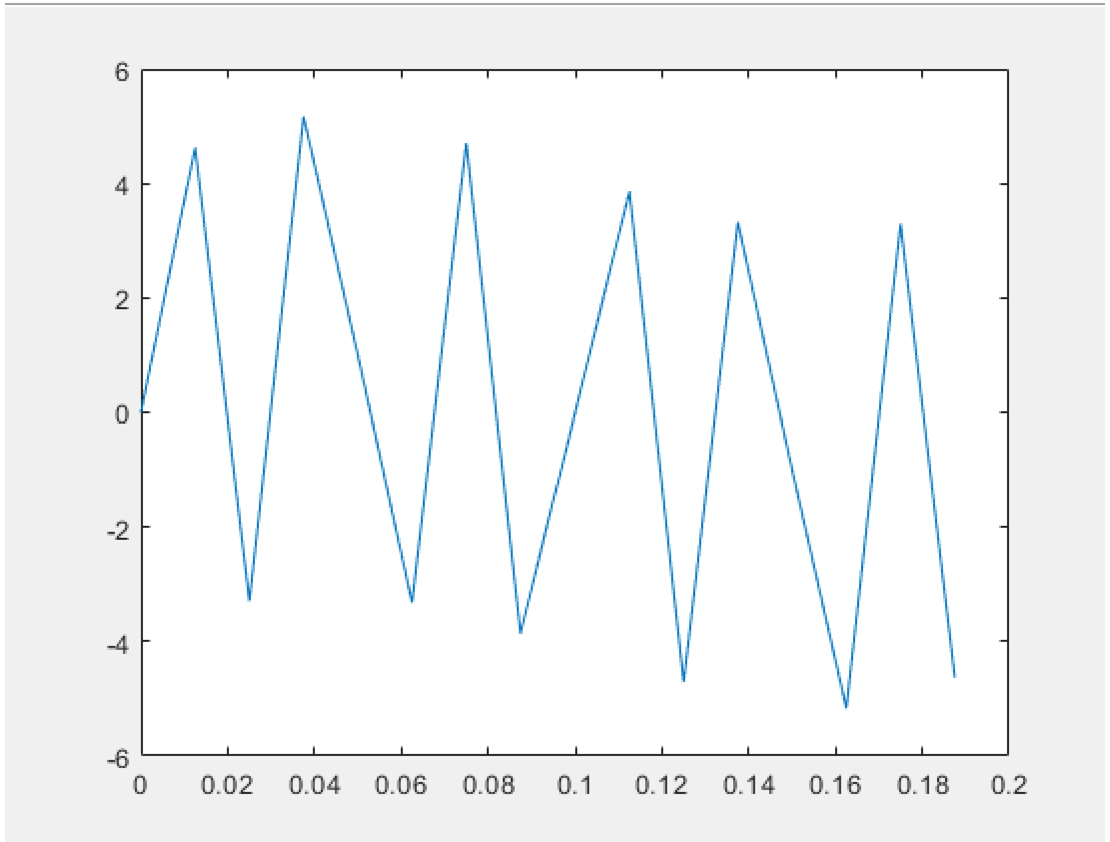
```
%script per il filtraggio di un segnale

t=[0:0.0125:(0.2-0.0125)]
%segnale affetto da rumore
fnoised=sin(2*pi*5*t)+sin(2*pi*10*t)+5*sin(
2*pi*30*t);

%g
fi
pl
%applico fft
fnoisedfft=fft(fnoised)
%%format long
n=length(t);
d
F
f
frequenza fondamentale(metà per simmetria)

figure(2)
%istogramma
stem(freq,abs(fnoisedfft(1:n/2+1)),'b')
```

eliminare le frequenza più grandi di 10 Hz, si costruisce un filtro

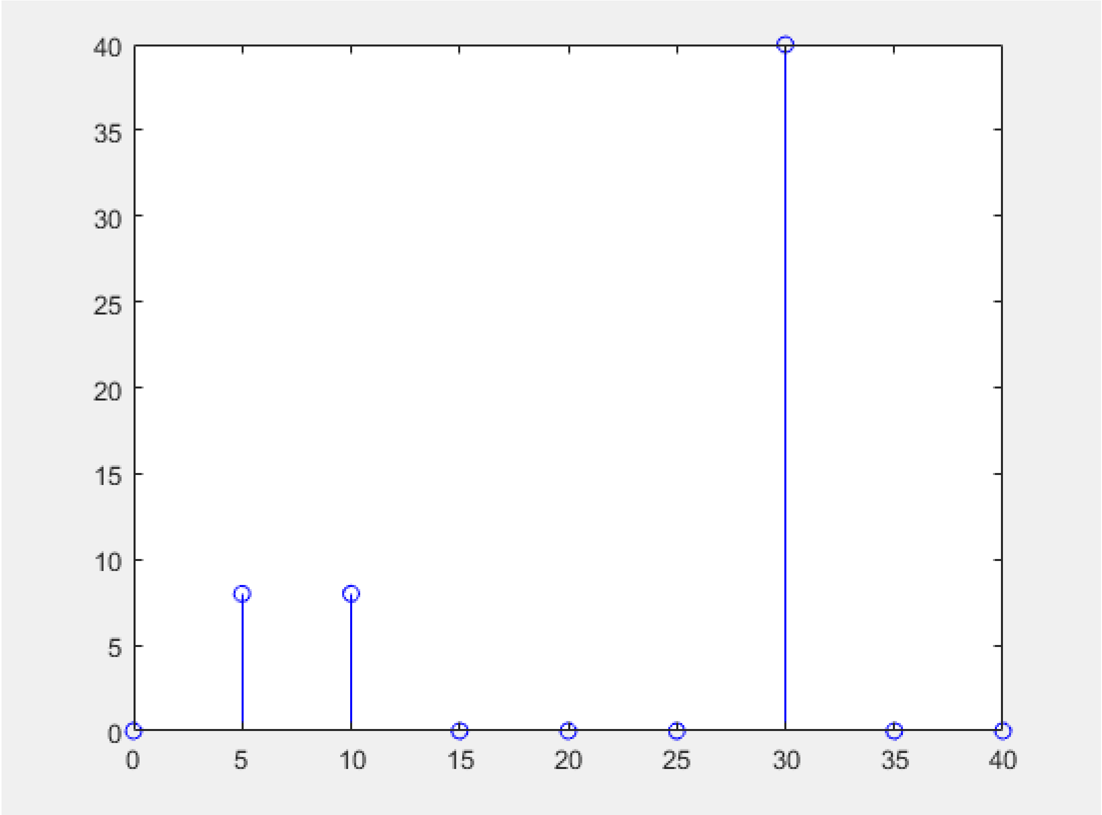
calcolo dell'inversa della sequenza trasformata *per tornare al dominio originale*

```
%%rimuovere il rumore e ritornare al
segnale pulito non più affetto da rumore
%%, nell'istogramma al punto 30 si deve
togliere il rumore.

%eliminare le frequenza più grandi di 10
Hz, si costruisce un filtro

freqq=(0:n-1)*FS/n
%definizione del filtro
H=ones(1,length(freq));
H(freqq>10)=0; %elimina le frequenze
relative al rumore
H(n/2+1:n)=fliplr(H(1:n/2)) %copia la prima
metà del vettore H nella seconda metà
                               % la dft e il
iltro sono simmetrici rispetto alla
requenza di Nyquist

%applicazione del filtro e ritorno nel
dominio del tempo
filtred=fnoisedfft.*H;
ify=ifft(filtred,n);
```

segnale



Abs(fft)



Segnale filtrato

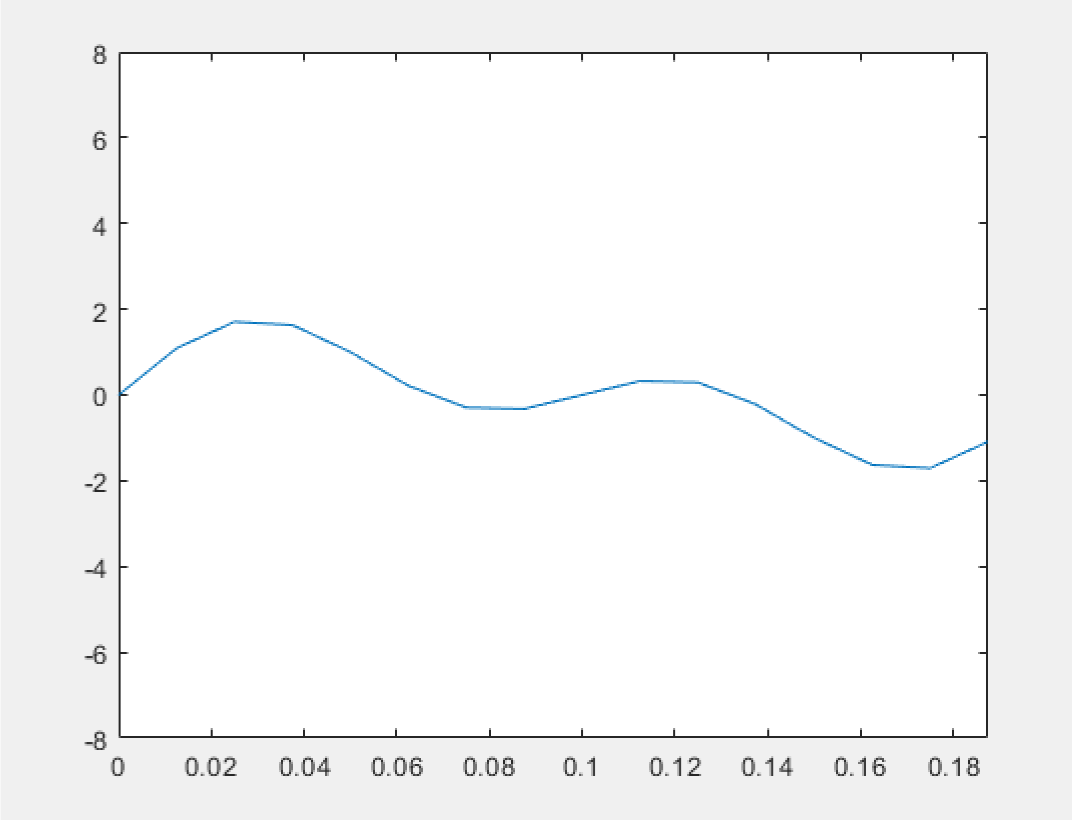
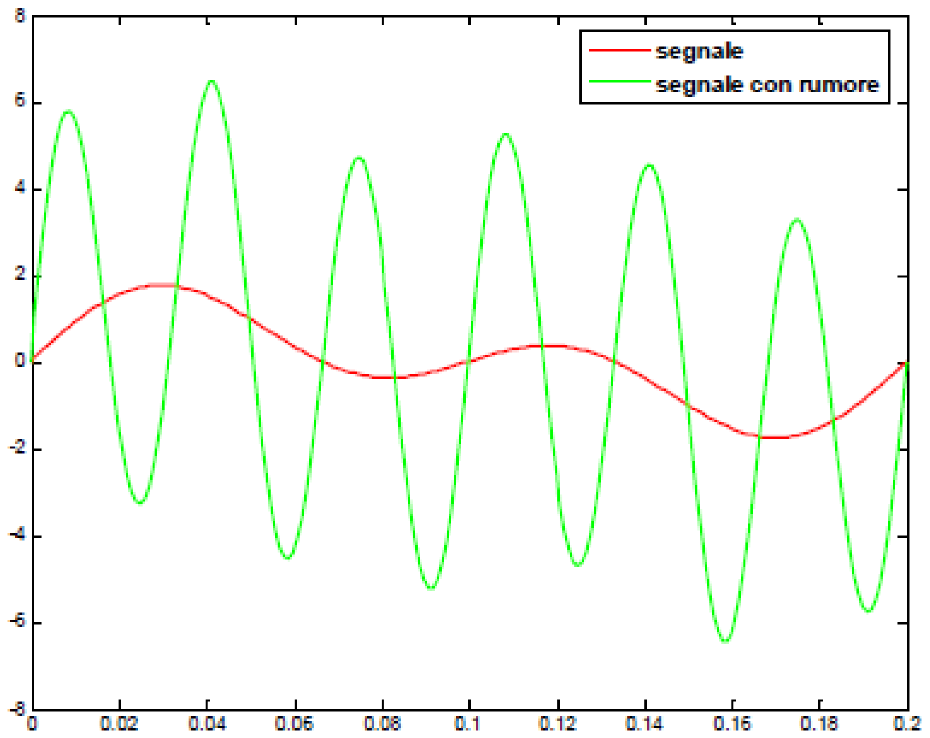


Grafico segnale con rumore e Segnale filtrato



Applicazione dft

Il DTMF(Dual Tone Multi-Frequency System) è un sistema di codifica usato per codificare codici numerici sotto forma di segnali sonori (telefonia, sistemi di integrazione computer-telefono, codici di carte di credito,…). Il telefono a tastiera è un esempio dell’uso quotidiano della DFT, usata per la codifica e decodifica di segnali che contengono le informazioni sul numero di telefono digitato.

Hz	1209	1336	1477
697	1	2	3
770	4	5	6
852	7	8	9
941	*	0	#

Tramite la DFT sono individuate le frequenze del segnale e quindi il numero a cui sono univocamente associate.

```
function [tone]=codifica(numero)
FS=8000; %frequenza di campionamento
dt=1/FS; %passo di discretizzazione
t=0:dt:0.5-dt; %asse dei tempi per la discretizzazione della funzione f(t) che descrive il suono corrispondente ad una singola cifra
tsil=0:dt:0.2-dt; %asse dei tempi per un silenzio di 0.2 sec
%frequenze base per la costruzione dei toni:
fr=[697 770 852 941]; fc=[1209 1336 1477 1633];
for i=1:length(numero)
 s=numero(i);
 switch s
 case 1
     r=1; c=1;
 case 2
     r=1; c=2;
 case 3
     r=1; c=3;
 case 4
     r=2; c=1;
 case 5
     r=2; c=2;
 case 6
     r=2; c=3;
 case 7
     r=3; c=1;
 case 8
     r=3; c=2;
 case 9
     r=3; c=3;
 case '*'
     r=4; c=1;
 case 0
     r=4; c=2;
 case '#'
     r=4; c=3;
 end
```

Discretizzazione degli intervalli

```
function [tone]=codifica(numero)
FS=8000; %frequenza di campionamento
dt=1/FS; %passo di discretizzazione
t=0:dt:0.5-dt; %asse dei tempi per la discretizzazione della funzione f(t) che descrive il suono corrispondente ad una singola cifra
tsil=0:dt:0.2-dt; %asse dei tempi per un silenzio di 0.2 sec
%frequenze base per la costruzione dei toni:
fr=[697 770 852 941]; fc=[1209 1336 1477 1633];
for i=1:length(numero)
 s=numero(i);
 switch s
 case 1
     r=1; c=1;
 case 2
     r=1; c=2;
 case 3
     r=1; c=3;
 case 4
     r=2; c=1;
 case 5
     r=2; c=2;
 case 6
     r=2; c=3;
```

```
    case 7
         r=3; c=1;
     case 8
         r=3; c=2;
     case 9
         r=3; c=3;
     case '*'
         r=4; c=1;
     case 0
         r=4; c=2;
     case '#'
         r=4; c=3;
     end
```

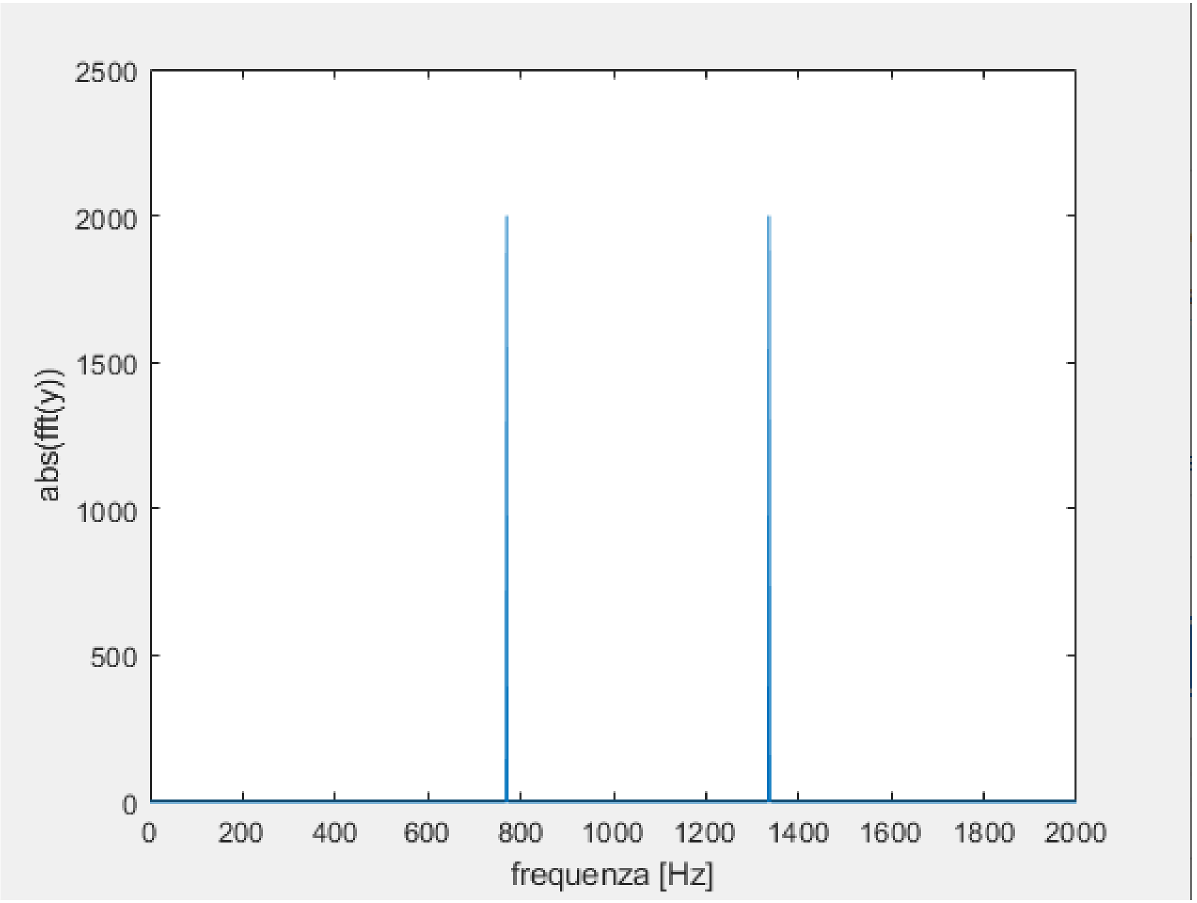
Identificazione delle frequenze

```
%per ogni cifra vengono individuate le frequenze
base,
%viene generato il vettore tone attraverso la
discretizzazione di due funzioni,
% la funzione identicamente nulla che descrive
il silenzio ...
tone=[];
silenzio=zeros(1,length(tsil));
% ... e la funzione f(t) che descrive il suono
associato alla cifra digitata
suono=sin(2*pi*fr(r)*t)+sin(2*pi*fc(c)*t)
tone=[tone suono silenzio]
%visualizzazione dello spettro delle frequenze
del suono determinato

n=length(t);
y=fft(suono,n);
figure(1)
subplot(3,3,i);
plot((0:n/2)*(FS/n), abs(y(1:n/2+1)));
axis([0 2000 0 2500])
xlabel('frequenza [Hz]')
ylabel('abs(fft(y))')
end
%ascolto del segnale in uscita
soundsc(tone)
end
```

Discretizzazione funzione

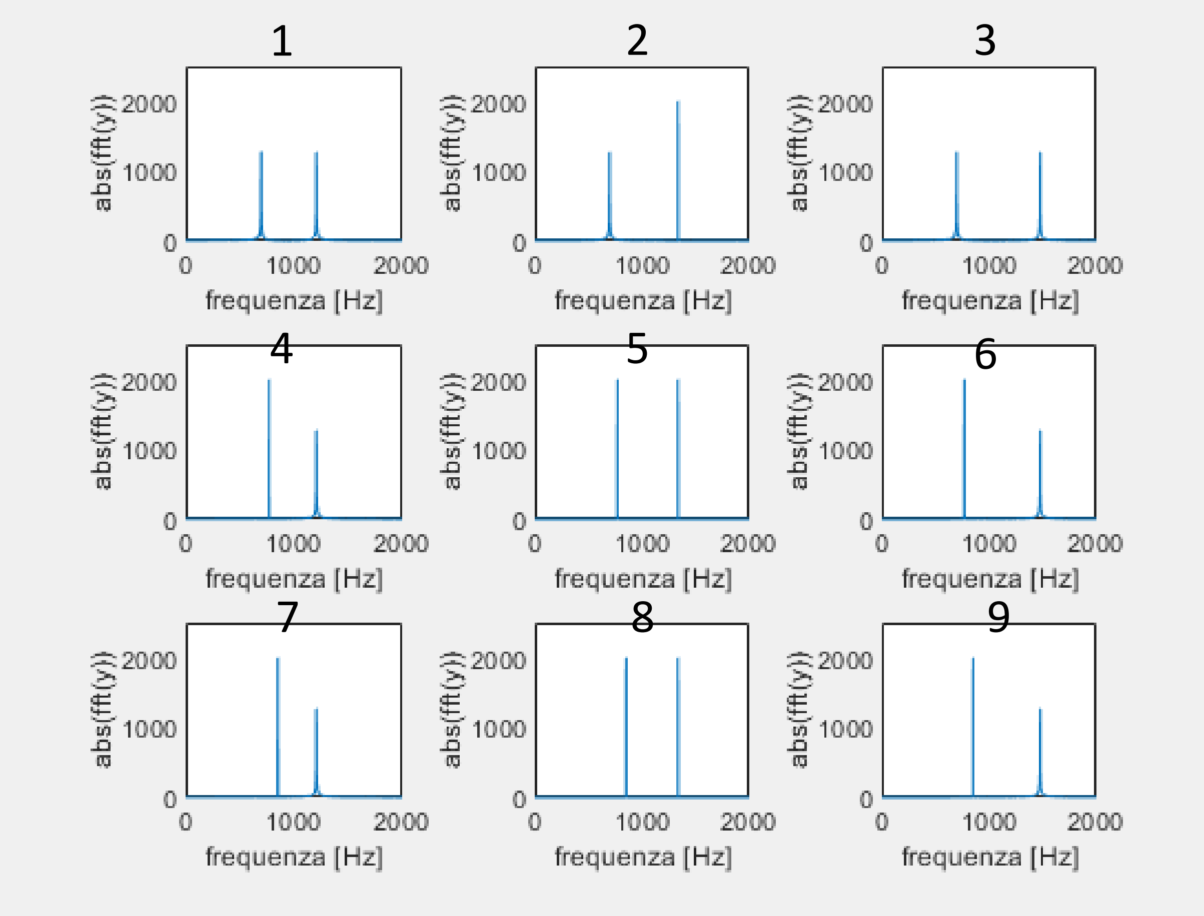
Calcolo dft e grafico

Tasto 5



2500
2000
1500
1000
500
0
abs(fft(y))
0
200
400
600
800
1000
1200
1400
1600
1800
2000
frequenza [Hz]

Codifica con input [1 2 3 4 5 6 7 8 9]



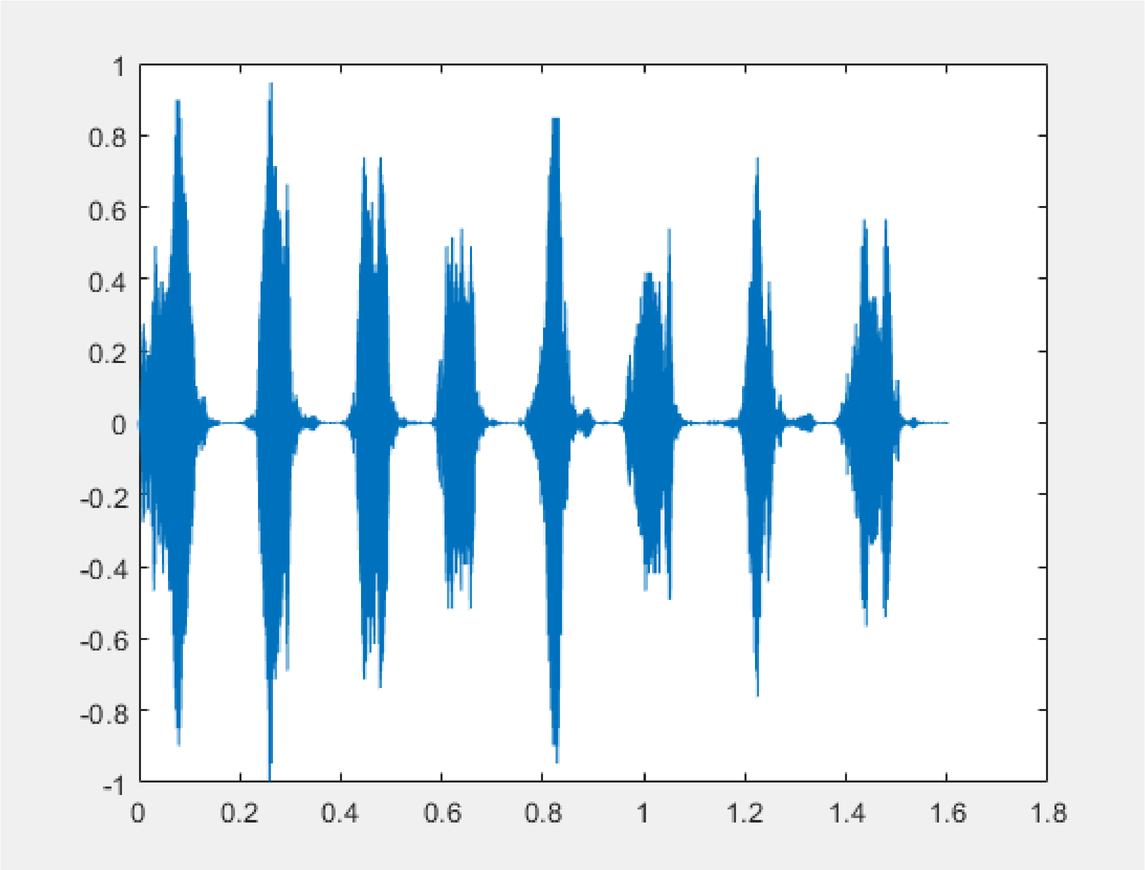
Il suono in Matlab

In questo esempio utilizziamo un file audio Matlab (help audiovideo dà un elenco dei file audio disponibili). Carico il file:

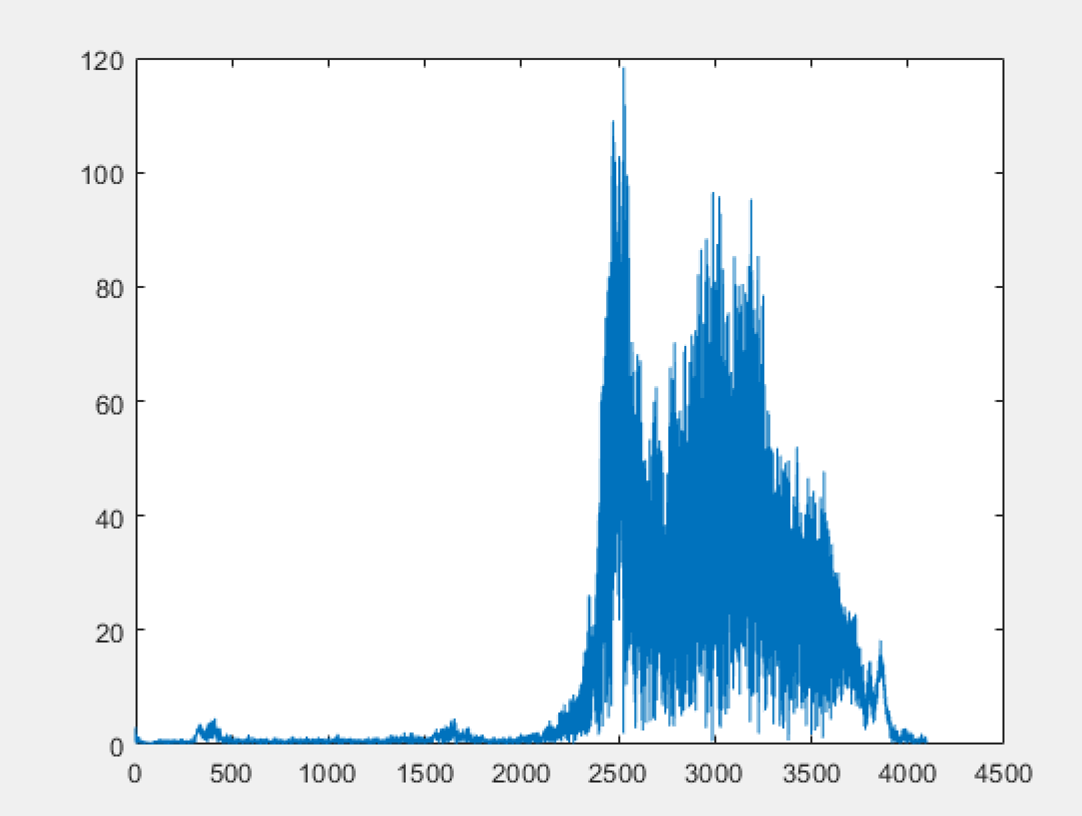
```
load chirp
n1=length(y);
t=0:1/Fs:(n1-1)/Fs;
plot(t,y)
```

Il file contiene un vettore y (segnale campionato) e la relativa frequenza di campionamento.

Segnale campionato



Effettuiamo l’analisi dello spettro del segnale facendo il grafico dell’ampiezza (la metà per simmetria intorno alla frequenza di Nyquist) rispetto le frequenze:



Esercizio da fare dal prompt

Il comando `soundsc(y,Fs)` converte un vettore in suono:

```
soundsc(y,Fs)
iiy=ifft(Y);
```

iiy	13129x1 double
n1	13129
t	1x13129 double
y	13129x1 double
Y	13129x1 complex dou...

Osserviamo che Y (DFT) è un vettore complesso, mentre iiy, come è da aspettarsi, è un vettore reale.

Grafico parte reale e immaginaria

```
subplot(2,2,1),plot(real(Y))
subplot(2,2,2),plot(imag(Y))
```

reale immaginaria

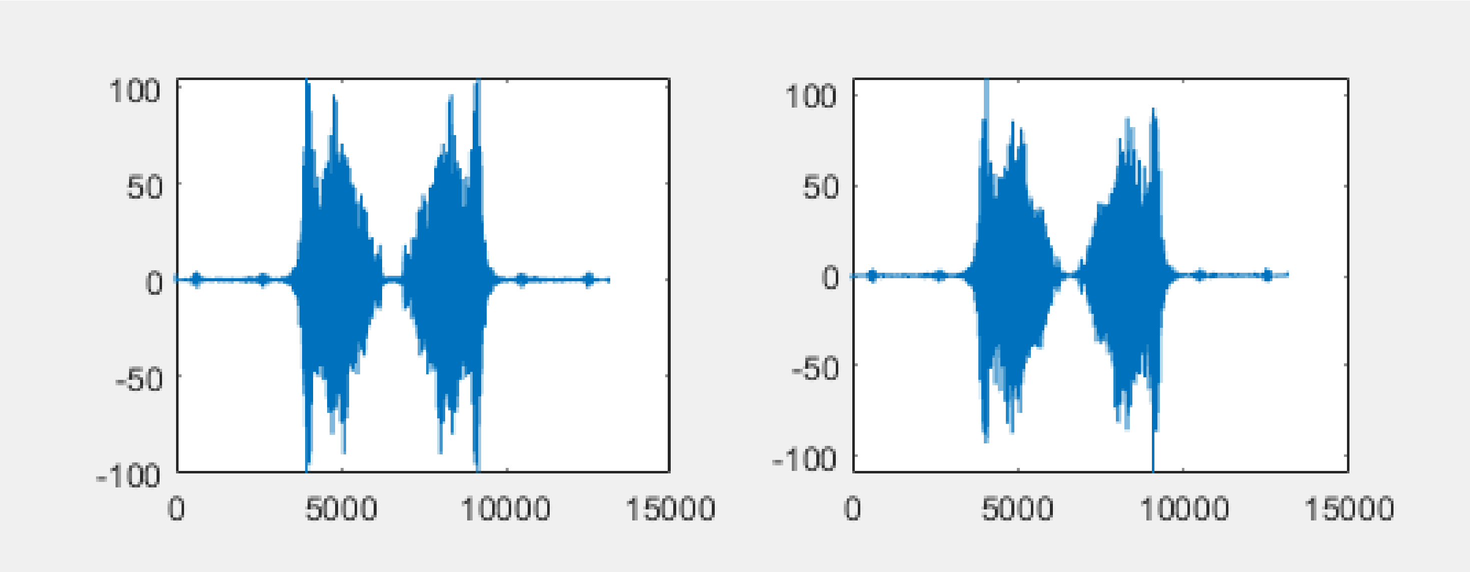
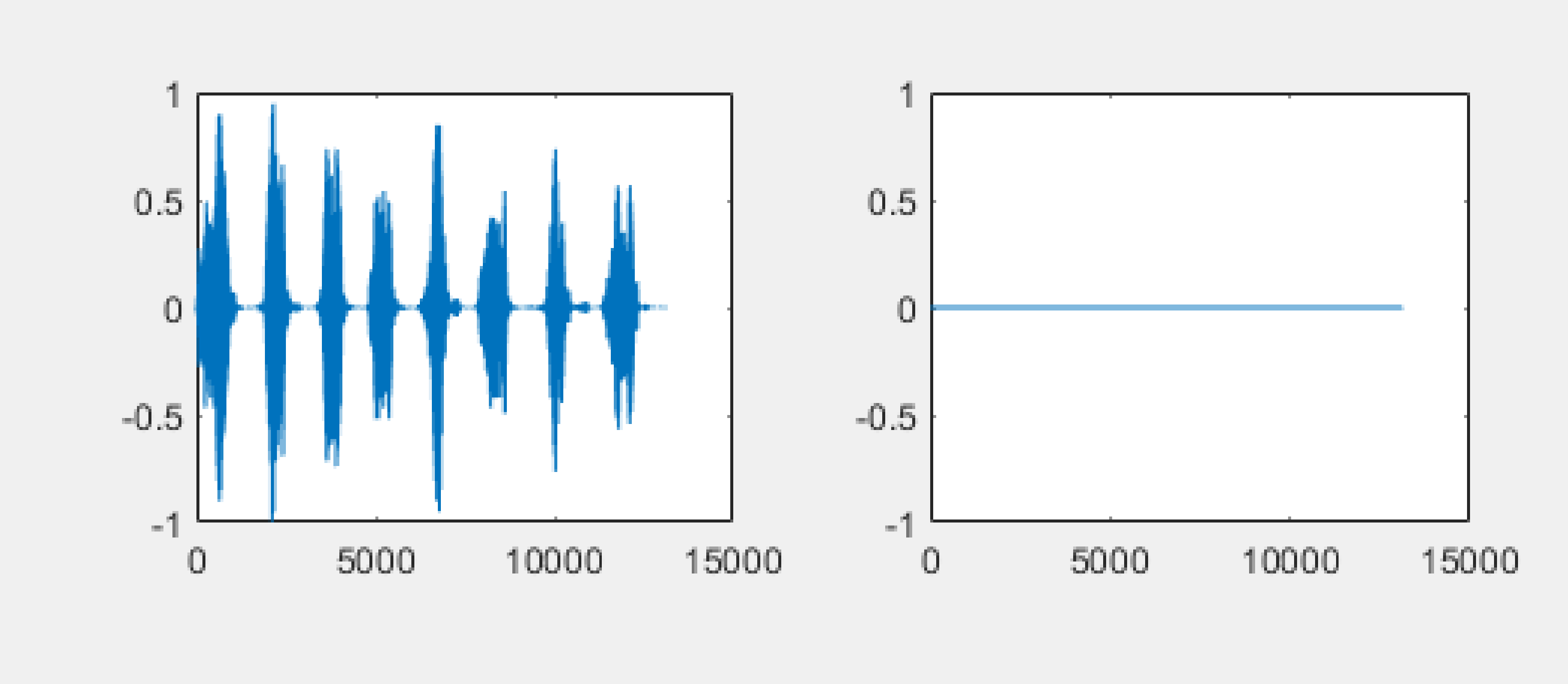


Grafico parte reale e immaginaria

```
subplot(2,2,1),plot(real(iiy))
subplot(2,2,2),plot(imag(iiy))
```

reale

immaginaria



soundsc(iiy) si risente il suono iniziale.

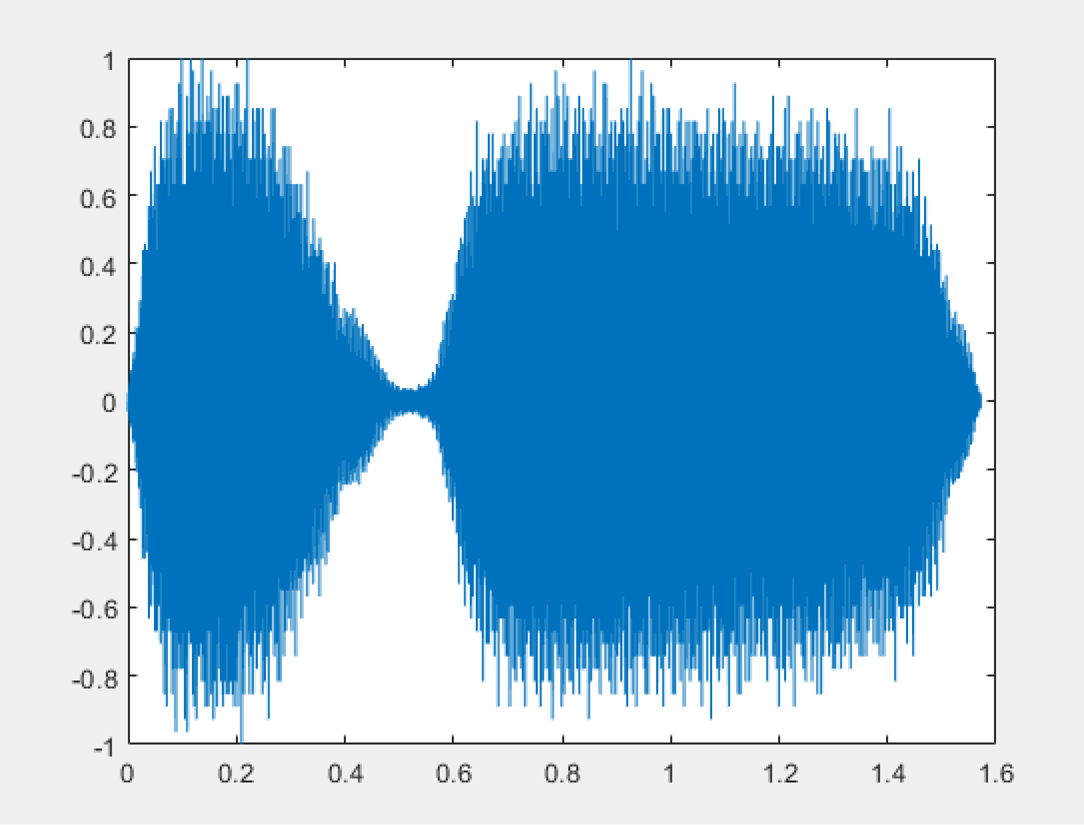
Esempio suono in Matlab

Consideriamo il suono chirp usato precedentemente e prendiamo dalla libreria Matlab un altro suono, train, abbastanza separato in frequenza:

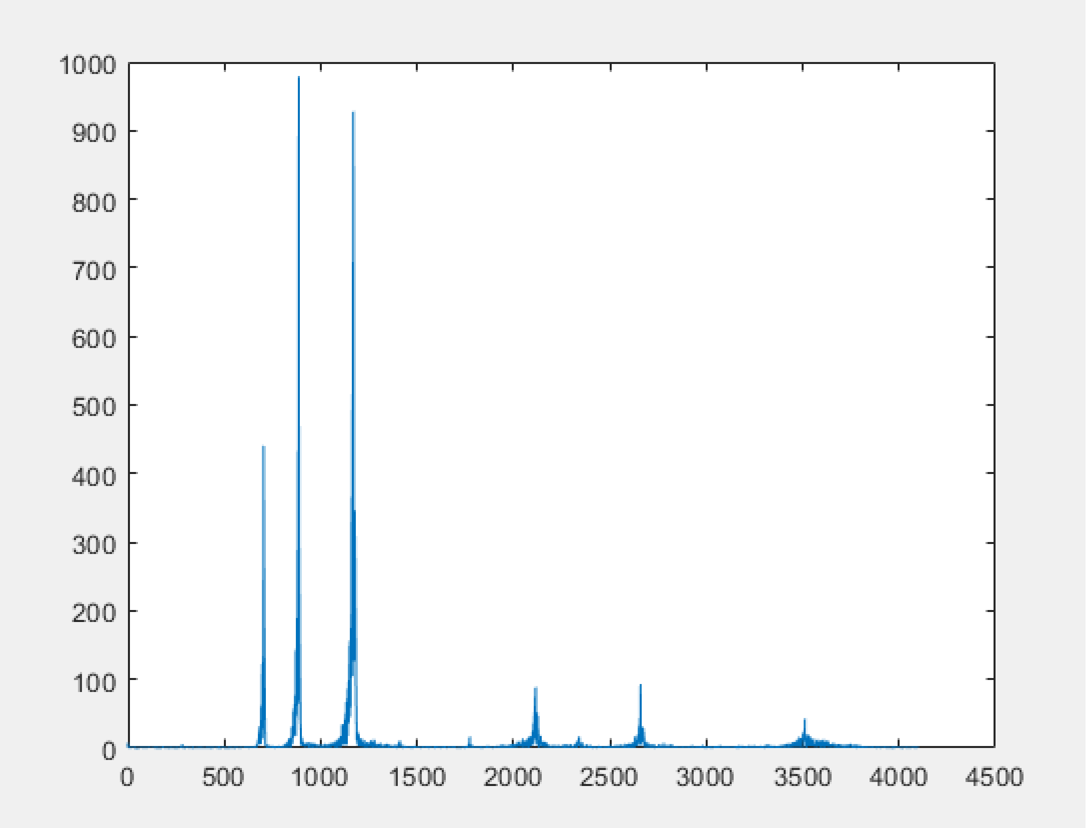
```
load train
n2=length(y);
t=0:1/Fs:(n2-1)/Fs;
y2=y;
YT=fft(y2);
amptreno=abs(YT(1:floor(n2/2)+1));
%ampiezze
ftreno=(0:n2/2)*Fs/n2; %frequenze
figure(1)
plot(t,y)
```

Grafici

segnale



Abs(fft)



Sovrapponiamo i suoni chirp e train ottenendo, così, un nuovo segnale:

```
%SOVRAPPOSIZIONE SUONO
```

```
load train
n2=length(y);
t=0:1/Fs:(n2-1)/Fs;
y2=y;
%--------------------Figura 3--------------
------
%figure(3)
%plot(t,y2)
%------------------------------------------
YT=fft(y2);
amptreno=abs(YT(1:floor(n2/2)+1));
%ampiezze
ftreno=(0:n2/2)*Fs/n2; %frequenze
soundsc(y2)
f=(0:n2/2)*Fs/n2;
```

Suono treno

```
%--------------------Figura 4--------
------
%figure(4)
%plot(f,amptreno)
%------------------------------------
%Sovrapponiamo i suoni ottenendo un n
segnale:

d=n1-n2; %uguaglio la lunghezza dei
campioni
y2=[y2;zeros(d,1)];
y=y1+y2;
n=length(y);
Y=fft(y);
amp=abs(Y(1:floor(n/2)+1));
f=(0:n/2)*Fs/n; %frequenze
```

Sovrapposizione 2 suoni

Sovrapponiamo i suoni chirp e train ottenendo, così, un nuovo segnale:

```
%SOVRAPPOSIZIONE SUONO
```

```
load train
n2=length(y);
t=0:1/Fs:(n2-1)/Fs;
y2=y;
%---------------------Figura 3---------------
------
%figure(3)
%plot(t,y2)
%-----------------------------------------
YT=fft(y2);
amptreno=abs(YT(1:floor(n2/2)+1));
%ampiezze
ftreno=(0:n2/2)*Fs/n2; %frequenze
soundsc(y2)
f=(0:n2/2)*Fs/n2;
```

Suono treno

```
%---------------------Figura 4---------
------
%figure(4)
%plot(f,amptreno)
%-------------------------------------
%Sovrapponiamo i suoni ottenendo un n
segnale:

d=n1-n2; %uguaglio la lunghezza dei
campioni
y2=[y2;zeros(d,1)];
y=y1+y2;
n=length(y);
Y=fft(y);
amp=abs(Y(1:floor(n/2)+1));
f=(0:n/2)*Fs/n; %frequenze
```

Calcolo fft del suono sovrapposto

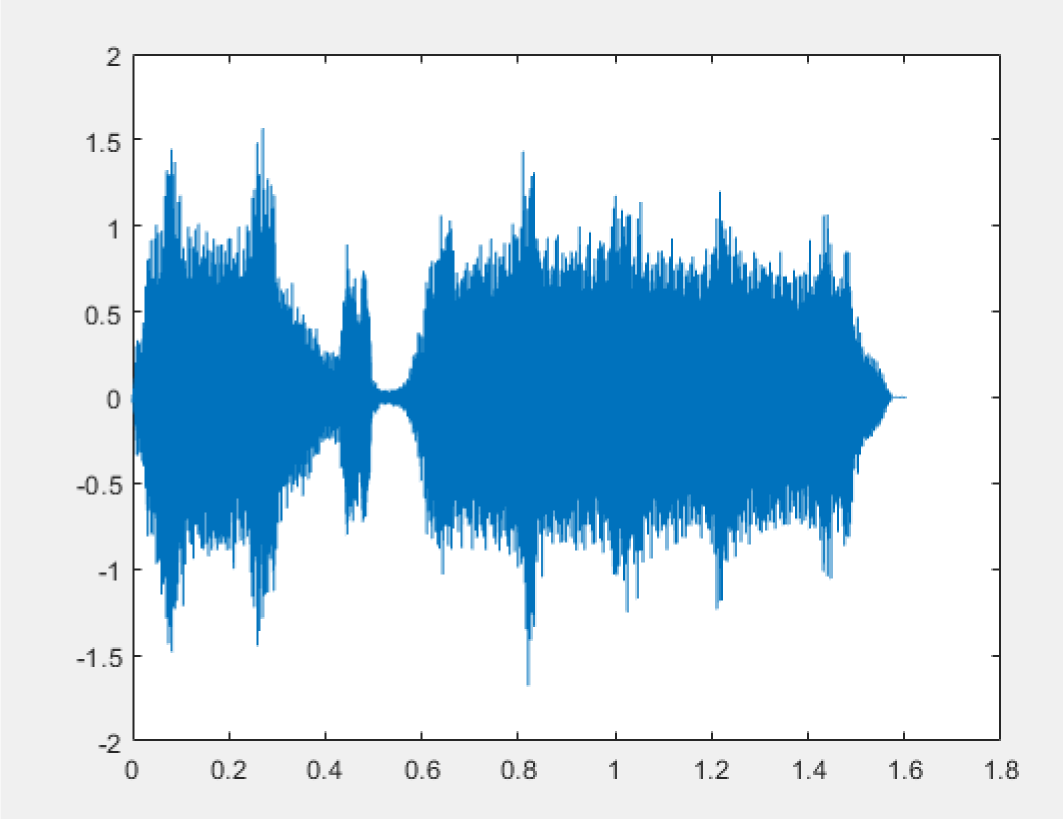
Sovrapponiamo i suoni chirp e train ottenendo, così, un nuovo segnale:

Istruzioni per genere i grafici del suono sovrapposto

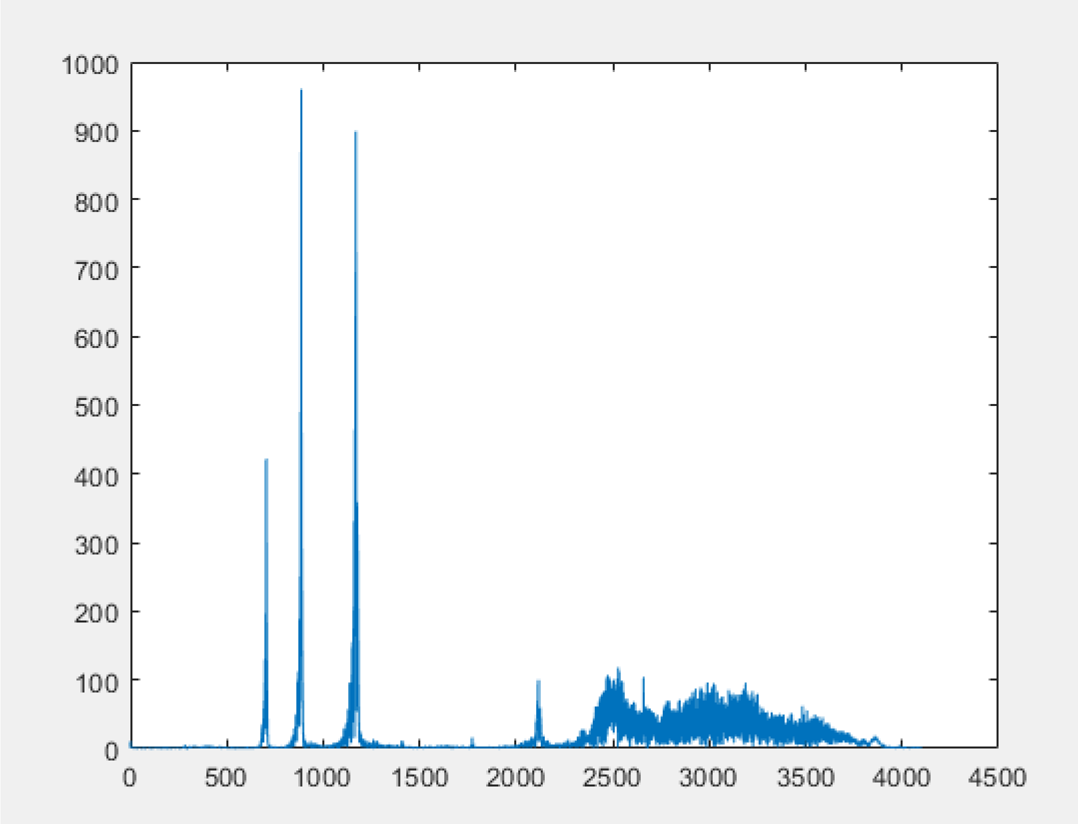
```
%--------------------Figura 5-6------------
---------

%figure(5)
%plot(t1,y)

%figure(6)
%plot(f,amp)

%suono sovrapposto

%soundsc(y)
```

Grafici

Segnale (treno+chirp)



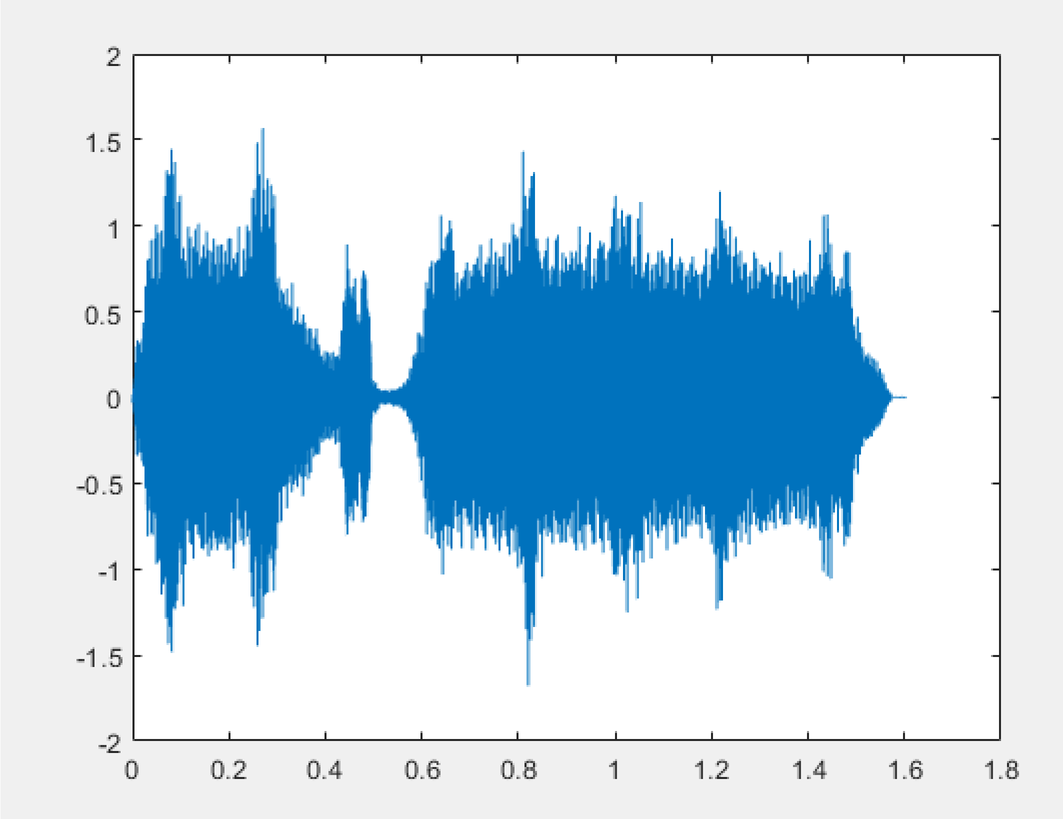
Abs(fft)



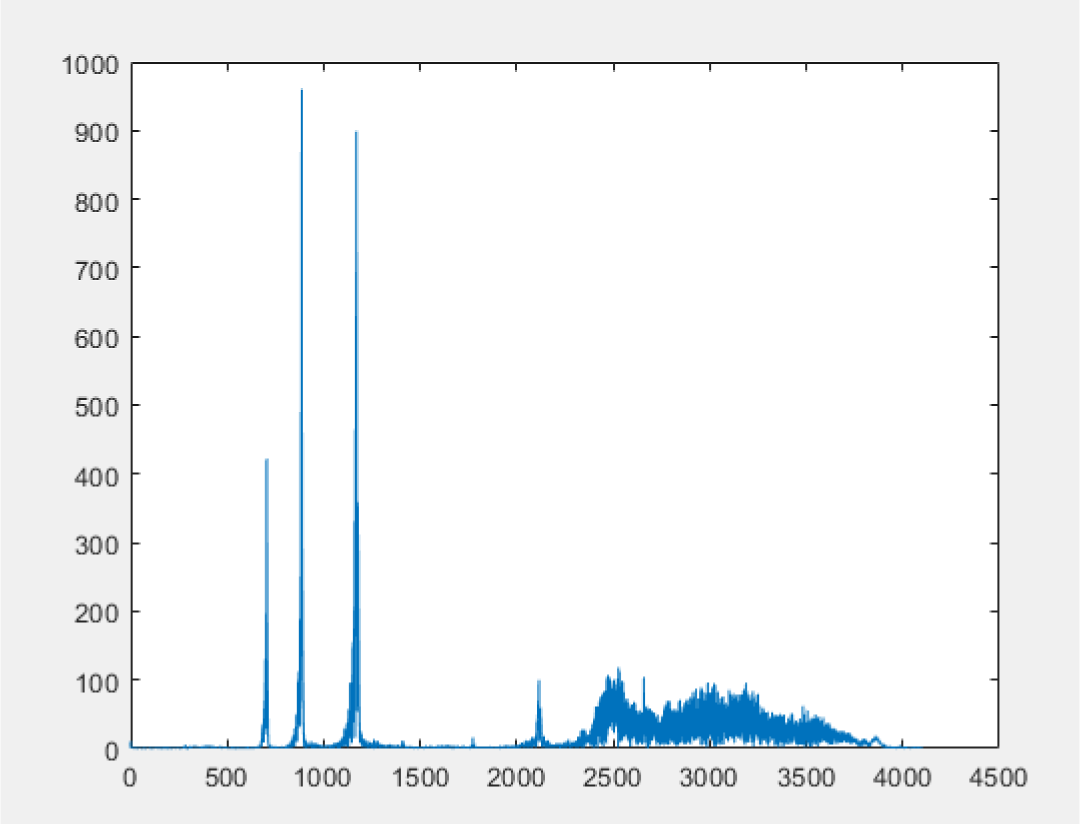
soundsc(y)

Grafici

Segnale (treno+chirp)



Abs(fft)



soundsc(y)

I due suoni che compongono il segnale sono indistinguibili nel tempo, mentre nel dominio delle frequenze risultano separati, circa intorno a 2200Hz.

Esempio fft con n non una potenza di 2

Convertiamo un impulso gaussiano dal dominio del tempo al dominio della frequenza.
Definire i parametri del segnale e un impulso gaussiano, X

```
Fs = 100;             % frequenza di campionamento
t = -0.5:1/Fs:0.5;  % discretizzazione intervallo temporale
L = length(t);        %lunghezza segnale
```

Esempio fft con n non una potenza di 2

Convertiamo un impulso gaussiano dal dominio del tempo al dominio della frequenza.
Definire i parametri del segnale e un impulso gaussiano, X

```
Fs = 100;               % frequenza di campionamento
t = -0.5:1/Fs:0.5;      % discretizzazione intervallo temporale
L = length(t);          %lunghezza segnale

y=1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01))); %impulso gaussiano

Y = fft(y);             %richiamiamo la function fft
```

Definiamo i parametri

Esempio fft con n non una potenza di 2

Convertiamo un impulso gaussiano dal dominio del tempo al dominio della frequenza.
Definire i parametri del segnale e un impulso gaussiano, X

```
Fs = 100;                % frequenza di campionamento
t = -0.5:1/Fs:0.5;       % discretizzazione intervallo temporale
L = length(t);           %lunghezza segnale

y=1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01)));
Y = fft(y);               %richiamiamo la function fft
```

Definiamo i parametri

Impulso gaussiano

Esempio fft con n non una potenza di 2

Convertiamo un impulso gaussiano dal dominio del tempo al dominio della frequenza.
Definire i parametri del segnale e un impulso gaussiano, X

```
Fs = 100;              % frequenza di campionamento
t = -0.5:1/Fs:0.5;     % discretizzazione intervallo temporale
L = length(t);         %lunghezza segnale
```

Definiamo i parametri

```
y=1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01)));
```

Impulso gaussiano

```
Y = fft(y);     %richiamiamo la function fft
```

Funzione fft per calolare la dft

Esempio fft con n non una potenza di 2

Convertiamo un impulso gaussiano dal dominio del tempo al dominio della frequenza.
Definire i parametri del segnale e un impulso gaussiano, X

```
Fs = 100;              % frequenza di campionamento
t = -0.5:1/Fs:0.5;     % discretizzazione intervallo temporale
L = length(t);         %lunghezza segnale
```

Definiamo i parametri

```
y=1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01)));
```

Impulso gaussiano

```
Y = fft(y);      %richiamiamo la function fft
```

Funzione fft per calolare la dft

```
t1=timeit(@() fft(y)) %function per la
media dei tempi per eseguire la fft
```

Media dei tempi

Esempio fft con n non una potenza di 2

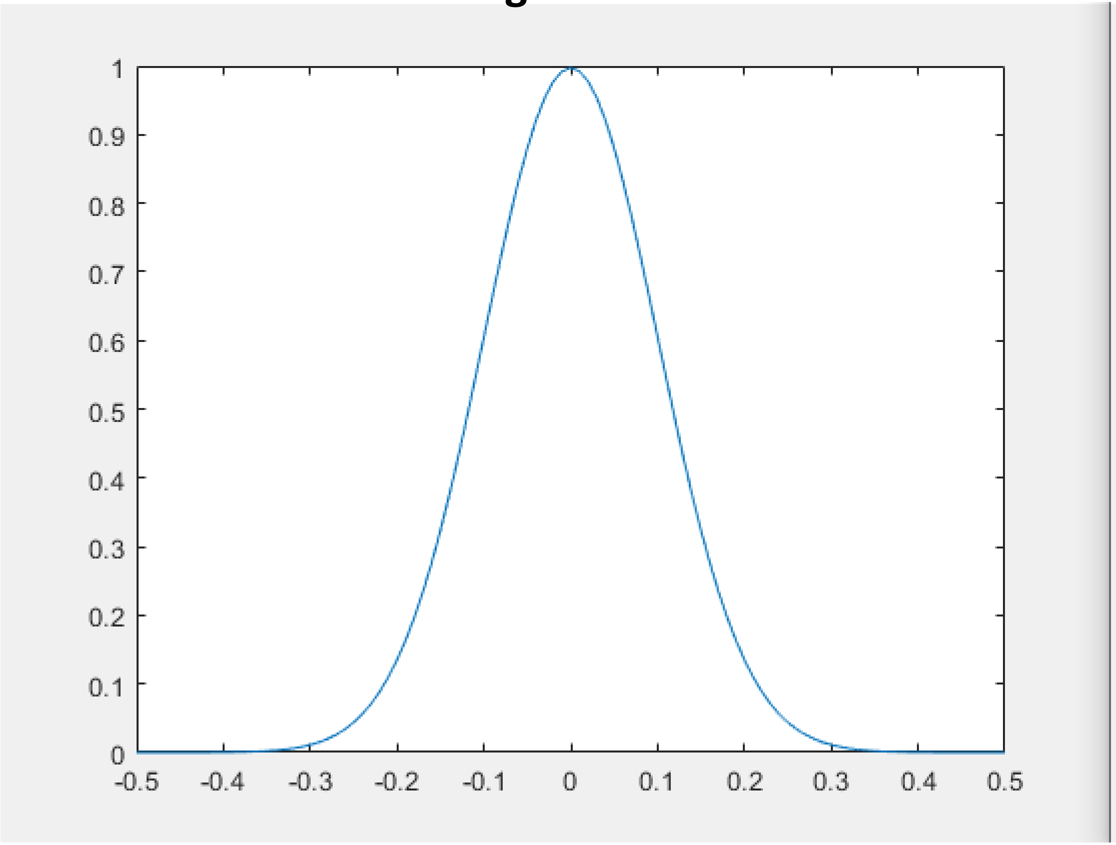
Convertiamo un impulso gaussiano dal dominio del tempo al dominio della frequenza.
Definire i parametri del segnale e un impulso gaussiano, X

```
Fs = 100;             % frequenza di campionamento
t = -0.5:1/Fs:0.5;    % discretizzazione intervallo temporale
L = length(t);        %lunghezza segnale
y=1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01))); %impulso gaussiano

Y = fft(y);
t1=timeit(@() fft(y)) %function per la media dei tempi per eseguire la
fft
f=(Fs/L)*(0:L/2);%frequenza


figure(1)          %grafici dft e impulso
plot(f,abs(Y(1:L/2+1))); %Plot single-sided amplitude spectrum
xlabel('Frequenza (Hz)')
ylabel('|Y(f)|')

figure(2)
plot(t,y)
```

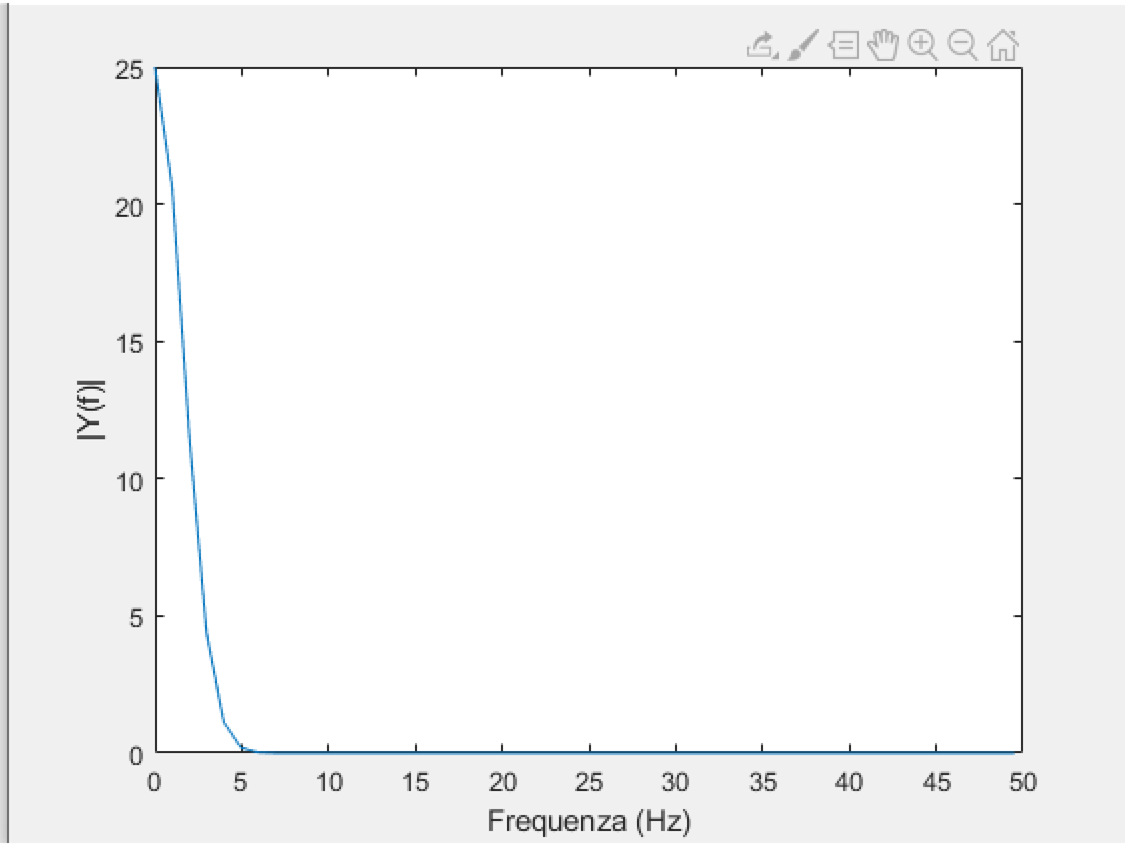
Notiamo che la dimensione del vettore y (input fft) è 121

Grafici

segnale



dft



Esempio fft con input una potenza di due «più vicina» ad n

Identificare una nuova lunghezza di ingresso che è la potenza successiva di 2 dalla lunghezza del segnale originale. Questo riempirà il segnale X di zeri finali per migliorare le prestazioni di fft.

```
Fs = 100;           % frequenza
t = -0.5:1/Fs:0.5;  % Time
L = length(t);      %lunghezza segnale
```

Parametri di ingresso

Esempio fft con input una potenza di due «più vicina» ad n

Identificare una nuova lunghezza di ingresso che è la potenza successiva di 2 dalla lunghezza del segnale originale. Questo riempirà il segnale X di zeri finali per migliorare le prestazioni di fft.

```
Fs = 100;              % frequenza
t = -0.5:1/Fs:0.5;     % Time
L = length(t);         %lunghezza segnale
```

Parametri di ingresso

```
y=1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01))); %funzione
```

Impulso gaussiano

Esempio fft con input una potenza di due «più vicina» ad n

Identificare una nuova lunghezza di ingresso che è la potenza successiva di 2 dalla lunghezza del segnale originale. Questo riempirà il segnale X di zeri finali per migliorare le prestazioni di fft.

```
Fs = 100;            % frequenza
t = -0.5:1/Fs:0.5;   % Time
L = length(t);       %lunghezza segnale
```

Parametri di ingresso

```
y=1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01))); %funzione
```

Impulso gaussiano

```
NFFT = 2^nextpow2(L);
```

`nextpow2 fornisce`
il primo p tale che 2^p > |L|

L=121< NFFT=128=2^7

Esempio fft con input una potenza di due «più vicina» ad n

Identificare una nuova lunghezza di ingresso che è la potenza successiva di 2 dalla lunghezza del segnale originale. Questo riempirà il segnale X di zeri finali per migliorare le prestazioni di fft.

```
Fs = 100;             % frequenza
t = -0.5:1/Fs:0.5;    % Time
L = length(t);        %lunghezza segnale
```

Parametri di ingresso

```
y=1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01))); %funzione
```

Impulso gaussiano

```
NFFT = 2^nextpow2(L);
```

`nextpow2 fornisce`
il primo p tale che 2^p > |L|

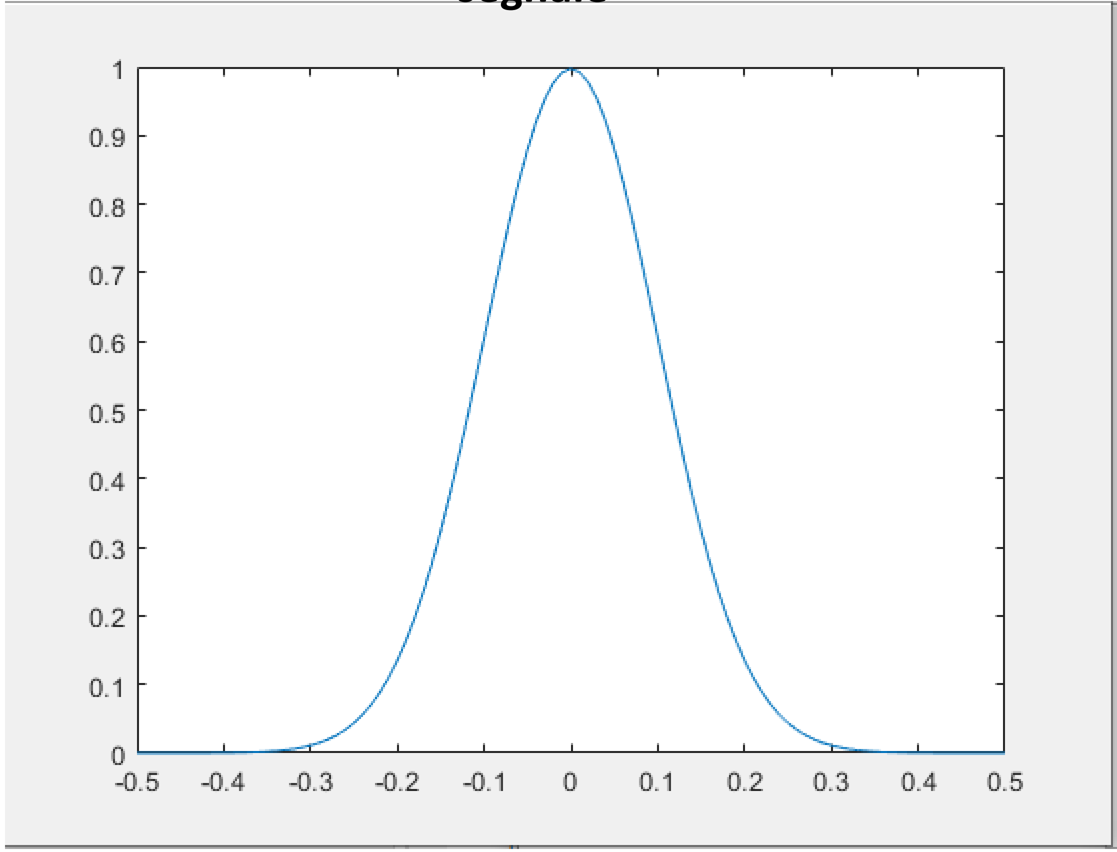
L=121< NFFT=128=2^7

```
Y = fft(y,NFFT);

timeit(@() fft(y,NFFT)) %function per la media dei tempi per eseguire la fft

f=(Fs/L)*(0:NFFT/2);%costruzione di un vettore di punti equispaziati
```

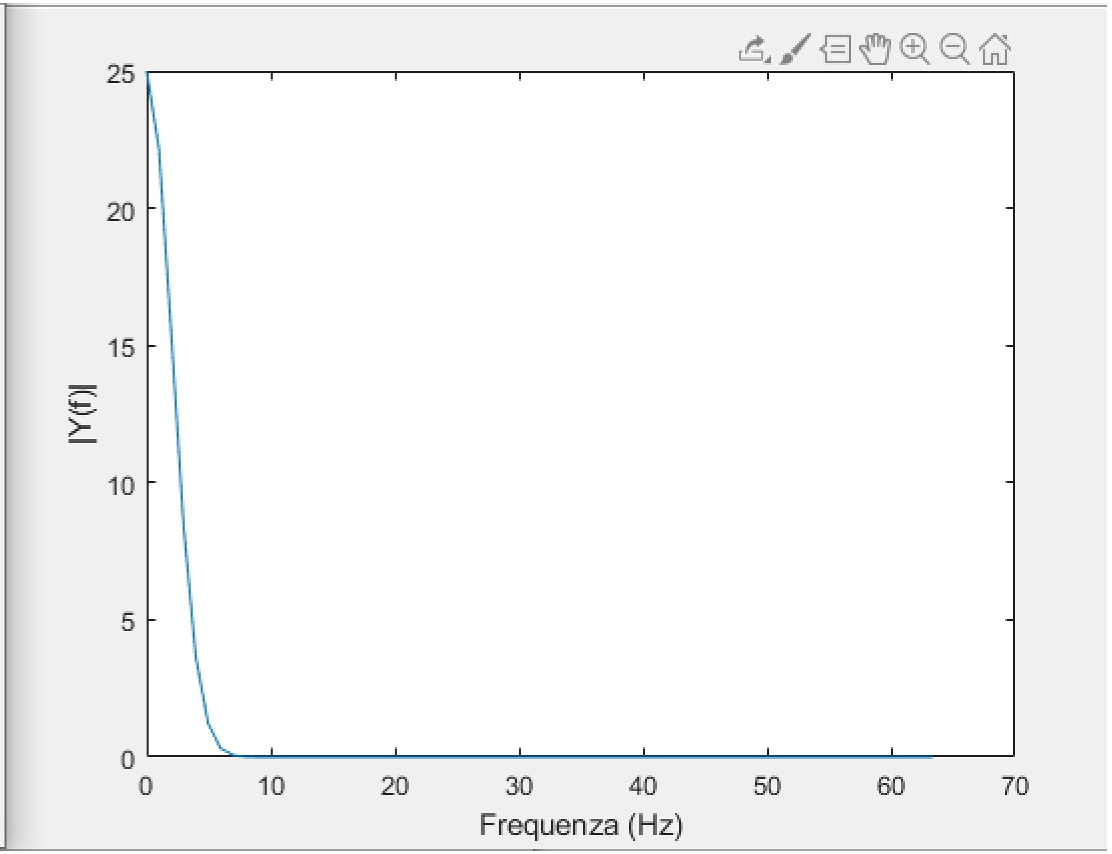
Aggiungendo in input NFFT, effettua
Il padding di 0 fino ad arrivare
alla potenza 2^7(=NFFT)

Grafici

segnale



dft



Confronto dei tempi per l'impulso gaussiano

Per L=121 (dimesione vettore input della fft)

t1= 1.7343e-05

Confronto dei tempi per l'impulso gaussiano

Per L=121 (dimesione vettore input della fft)

t1= 1.7343e-05

Passiamo alla potenza di 2 più vicina e L

Confronto dei tempi per l'impulso gaussiano

Per L=121 (dimesione vettore input della fft)

Per NFFT=2^7=128 (dimesione vettore input della fft)

t1= 1.7343e-05

Passiamo alla potenza di 2 più vicina e L

Confronto dei tempi per l'impulso gaussiano

Per L=121 (dimesione vettore input della fft)

Per NFFT=2^7=128 (dimesione vettore input della fft)

t1= 1.7343e-05

Passiamo alla potenza di 2 più vicina e L

t2=1.2293e-05

Confronto dei tempi per l'impulso gaussiano

Per L=121 (dimesione vettore input della fft)

Per NFFT=2^7=128 (dimesione vettore input della fft)

t1= 1.7343e-05

Passiamo alla potenza di 2 più vicina e L

t2=1.2293e-05

Si ha che il calcolo è più veloce se N è potenza di due, anche se la sequenza è più lunga.

Esempio fft(X,n)

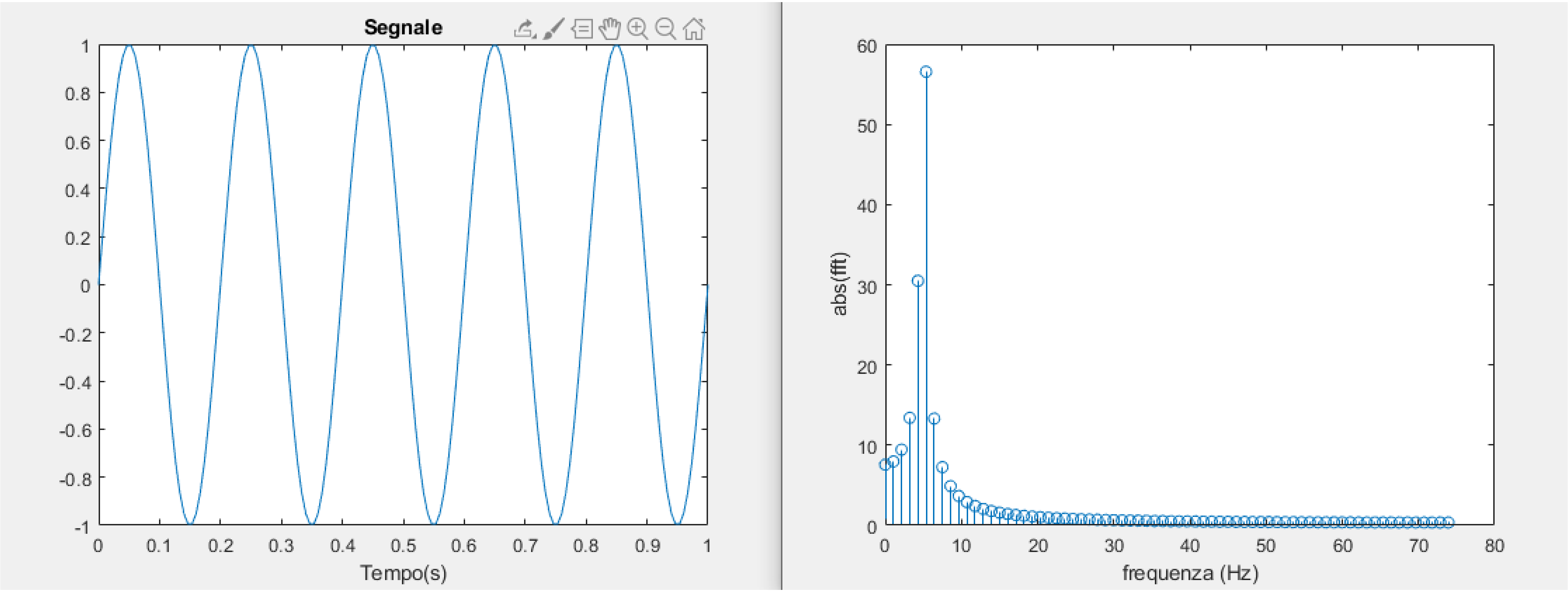
```
Fs = 150; % freq. campionaria
dt=1/Fs;
t = 0:dt:1; % vettore tempo
f = 5;
x = sin(2* pi*t*f);
n=size(x);
nfft = 140; % lunghezza fft
X = fft(x,nfft);
X = X(1:nfft/2);
mx = abs(X);
% vettore frequenza
f = (0:nfft/2-1)*Fs/nfft;
% genero i grafici
figure(1);
plot(t,x);
title('Signale');
xlabel('Tempo(s)');
ylabel('Amplitude');
figure(2);
stem(f,mx);
xlabel('frequenza (Hz)');
ylabel('abs(fft)');
```

```
>> n

n =

   1  151
```

```
>> nfft

ans =
      140
```

Essendo nfft<n allora la fft
Calcola la dft del vettore x fino alla
140-esima componente (x(1:nfft))

Esempio fft(X,n)



Esempio fft con impulso rettangolare

Esempio fft nel caso di un impulso rettangolare
Generato tramite la funzione rectpuls di matlab

Esempio fft con impulso rettangolare

Esempio fft nel caso di un impulso rettangolare
Generato tramite la funzione rectpuls di matlab

Passo (dt)

```
Fs = 150; % freq.campionaria
t = -0.5:1/Fs:0.5; % tempo
w = .2; % larghezza del rettangolo
x=rectpuls( t, w); %Genera impulso
rettangolare
```

Esempio fft con impulso rettangolare

Passo (dt)

```
Fs = 150; % freq.campionaria
t = -0.5:1/Fs:0.5; % tempo
w = .2; % larghezza del rettangolo
x=rectpuls( t, w); %Genera impulso rettangolare
nfft = 512; %lunghezza  FFT
X = fft(x,nfft);
```

Esempio fft nel caso di un impulso rettangolare
Generato tramite la funzione rectpuls di matlab

Richiamo la function
fft

Esempio fft con impulso rettangolare

Passo (dt)

```
Fs = 150; % freq.campionaria
t = -0.5:1/Fs:0.5; % tempo
w = .2; % larghezza del rettangolo
x=rectpuls( t, w); %Genera impulso rettangolare
nfft = 512; %lunghezza  FFT
X = fft(x,nfft);
% per la simmetria della fft
X = X(1:nfft/2);
mx = abs(X);
% frequenze
f = (0:nfft/2-1)*Fs/nfft;
```

Esempio fft nel caso di un impulso rettangolare
Generato tramite la funzione rectpuls di matlab

Richiamo la function fft

Frequenze

Esempio fft con impulso rettangolare

Passo (dt)

Esempio fft nel caso di un impulso rettangolare
Generato tramite la funzione rectpuls di matlab

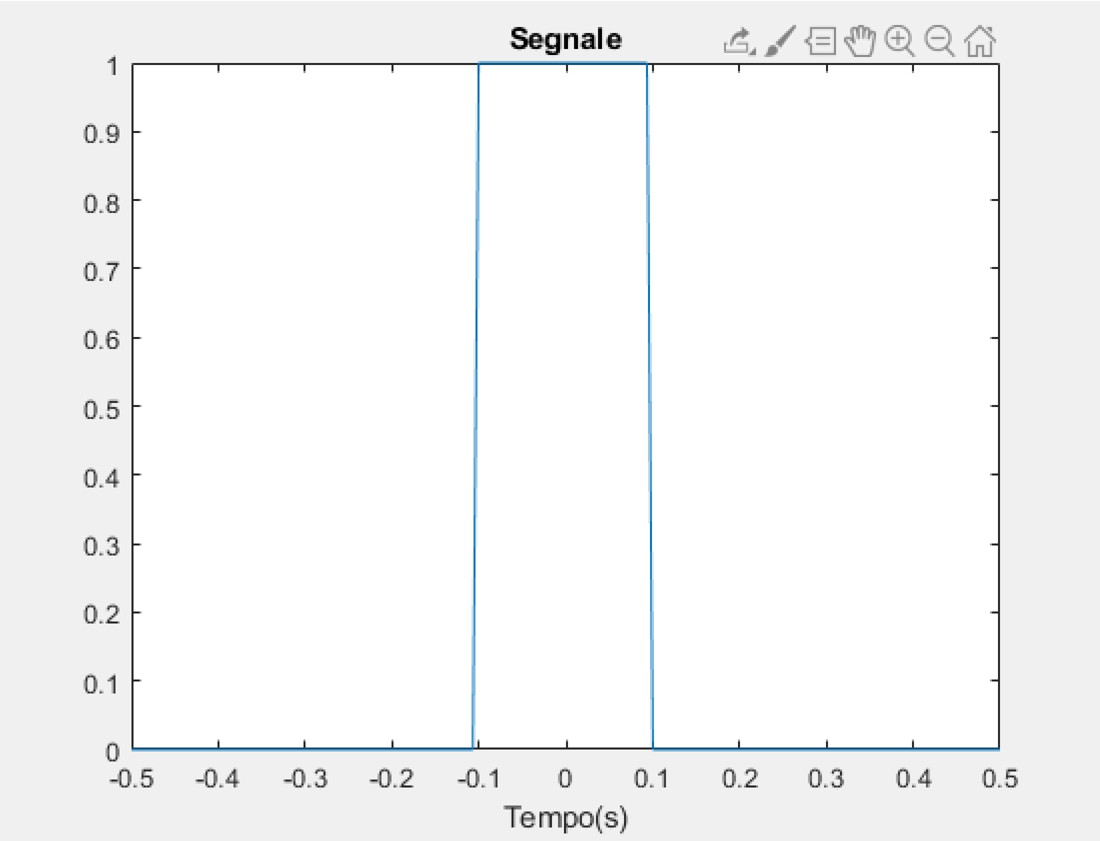
Richiamo la function fft

Frequenze

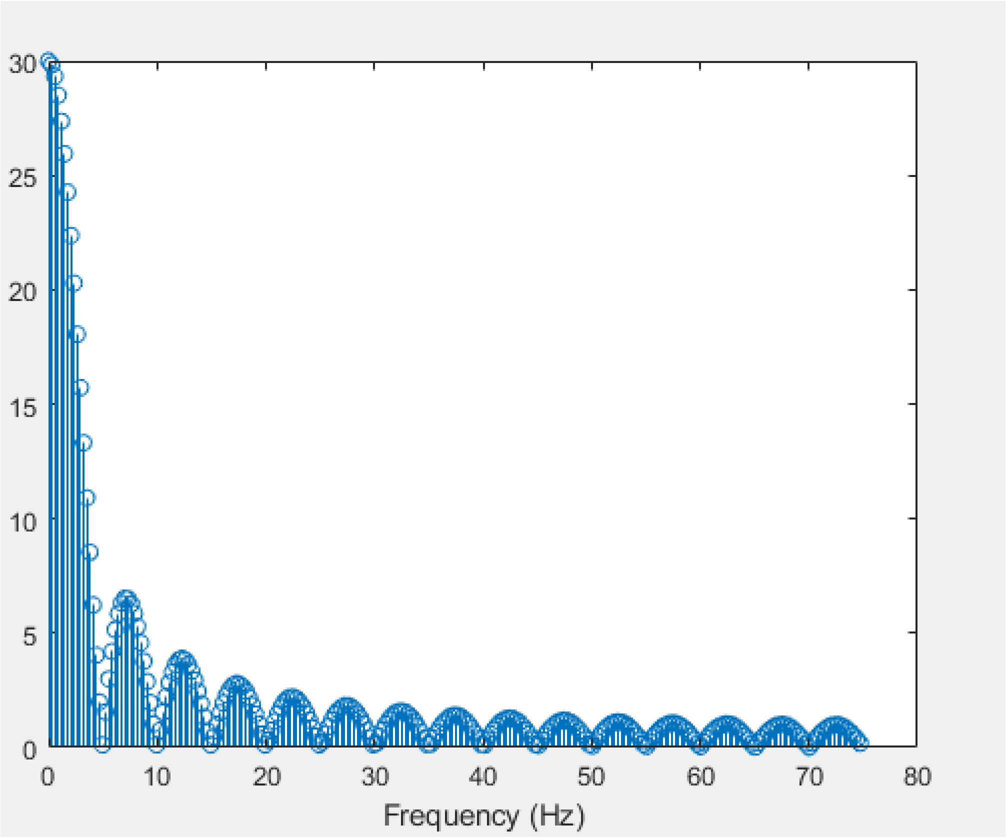
Grafici

```
Fs = 150; % freq.campionaria
t = -0.5:1/Fs:0.5; % tempo
w = .2; % larghezza del rettangolo
x=rectpuls( t, w); %Genera impulso
rettangolare
nfft = 512; %lunghezza  FFT
X = fft(x,nfft);
% per la simmetria della fft
X = X(1:nfft/2);
mx = abs(X);
% frequenze
f = (0:nfft/2-1)*Fs/nfft;
% grafici
figure(1);
plot(t,x);
title('Segnale');
xlabel('Tempo(s)');
figure(2);
stem(f,mx);
xlabel('Frequency (Hz)');
```

segnale



Abs(fft)



Esempio

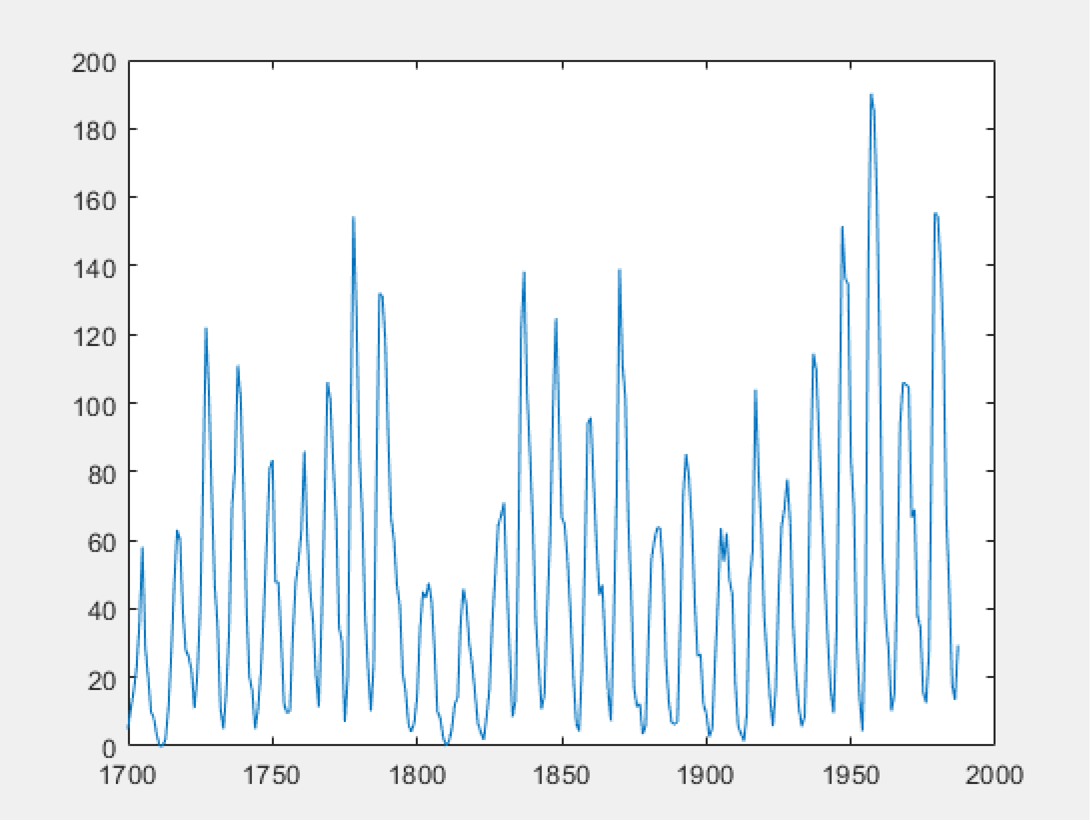
Un esempio è lo studio del fenomeno delle macchie solari (*sunspot*) attraverso la dft.

Indice di Wolfer:accoppia il numero e l'ampiezza delle macchie in un singolo indice

```
load sunspot.dat %carica i dati
t = sunspot(:,1)';
wolfer = sunspot(:,2)';
n = length(wolfer);
Y = fft(wolfer);
n=length(Y);
figure(1)
plot(t,wolfer)
```

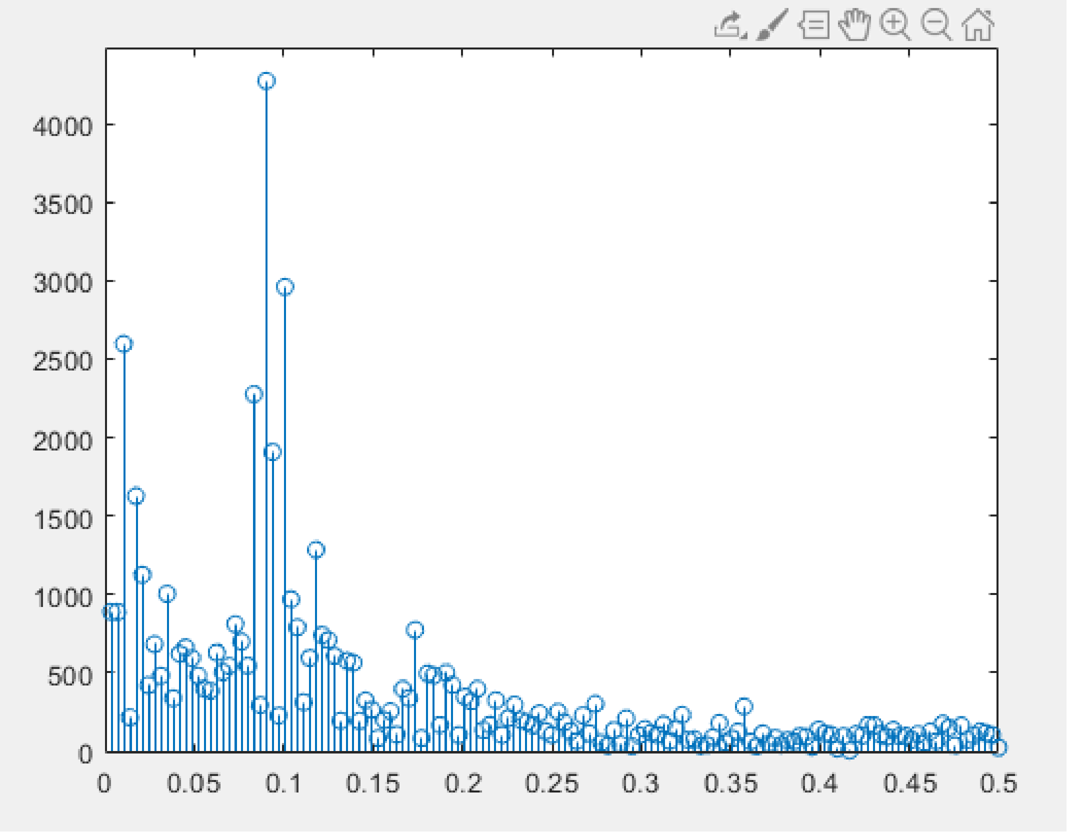
Plot degli indici

Grafico del fenomeno



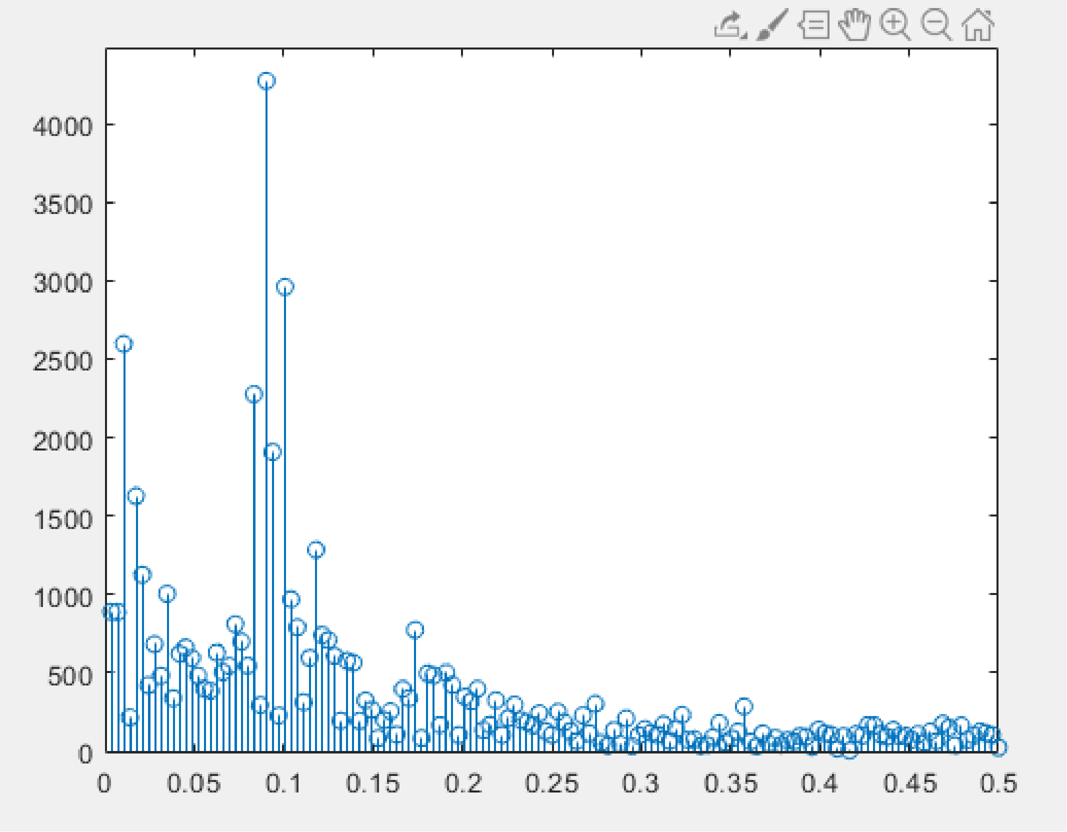
Grafico

Abs(fft)



Grafico

Abs(fft)



Massima ampiezza in corrispondenza di circa 0.09 Hz, periodo è il reciproco della frequenza, quindi circa 11 anni

Esempio:

Consideriamo le seguenti due funzione e calcoliamo la dft per discretizzazioni diverse dell'intervallo [0 1]

f1(x)= sin(2*pi*t)+sin(2*pi*5*t), e f2(x)= sin(2*pi*t)-sin(2*pi*3*t)

Esempio: f1(x)= sin(2*pi*t)+sin(2*pi*5*t), e f2(x)= sin(2*pi*t)-sin(2*pi*3*t)

Discretizzazione con N=9 campioni in [0,1]
$\Delta=1/(N-1)=1/(9-1)=1/8$
$t_j=j\,\Delta=j/8$, j=0,..., 8

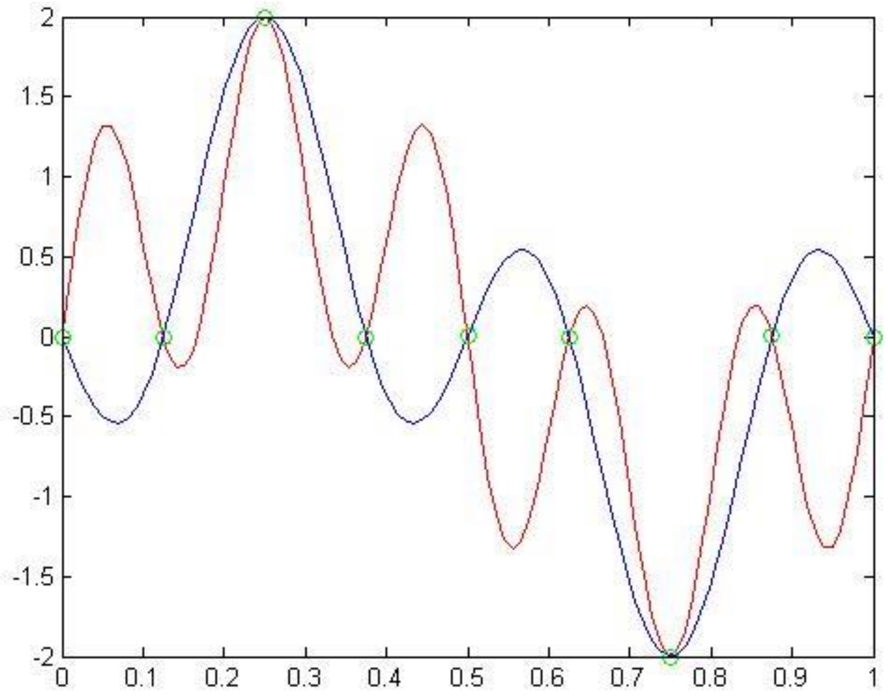
```
N=9;
dt=1/8;
t=0:dt:1;
f_1=@(x) sin(2*pi*x)+sin(2*pi*5*x);
f_2=@(x) sin(2*pi*x)-sin(2*pi*3*x);
f1= sin(2*pi*t)+sin(2*pi*5*t);
f2= sin(2*pi*t)-sin(2*pi*3*t);
figure(1)
plot(t,f1,'o')
hold on
plot(t,f2,'o')
hold on
fplot(f_1)
hold on
fplot(f_2)
axis([ 0 1 -2 2])
```

Esempio: f1(x)= sin(2*pi*t)+sin(2*pi*5*t), e f2(x)= sin(2*pi*t)-sin(2*pi*3*t)

Discretizzazione con N=9 campioni in [0,1]

$\Delta=1/(N-1)=1/(9-1)=1/8$

$t_j=j\,\Delta=j/8$, $j=0,\dots, 8$

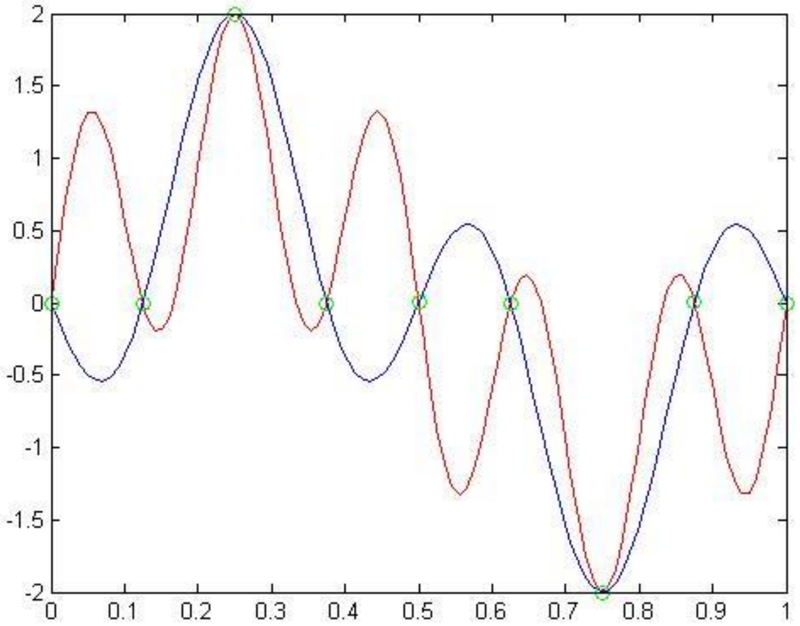


Nei nodi f1 non si distingue da f2 con frequenza minore

Infatti $FN=1/(2\Delta)=1/(2*(1/8))=4$.

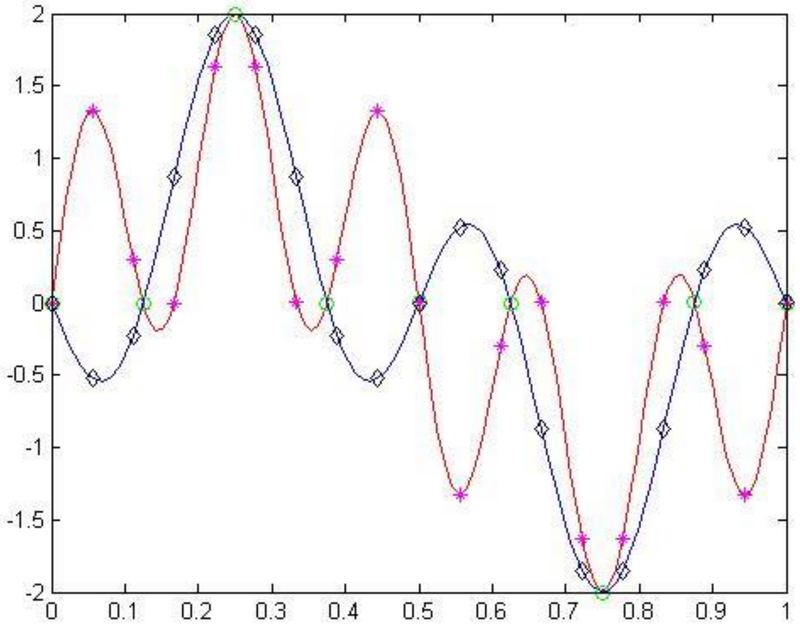
Esempio: f1(x)= sin(2*pi*t)+sin(2*pi*5*t), e f2(x)= sin(2*pi*t)-sin(2*pi*3*t)

Discretizzazione con $\boxed{N=9}$ campioni in [0,1]
$\Delta=1/(N-1)=1/(9-1)=1/8$
$t_j=j\,\Delta=j/8,\ j=0,\ldots,\ 8$



Nei nodi f1 non si distingue da f2 con frequenza minore
Infatti $FN=1/(2\Delta)=1/(2*(1/8))=4$.

Discretizzazione con $\boxed{N=19}$ campioni in [0,1]
$\Delta=1/(N-1)=1/(19-1)=1/18$
$t_j=j\,\Delta=j/18,\ j=0,\ldots,\ 18$

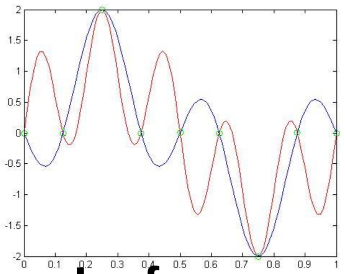


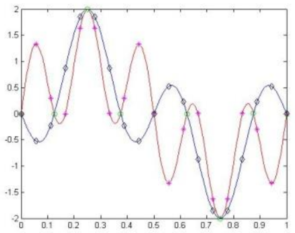
Aumentando il numero di nodi, ovvero riducendo Δ, *si ottiene correttamente la funzione a frequenza maggiore*.

Esempio: f1(x)= sin(2*pi*t)+sin(2*pi*5*t), e f2(x)= sin(2*pi*t)-sin(2*pi*3*t)

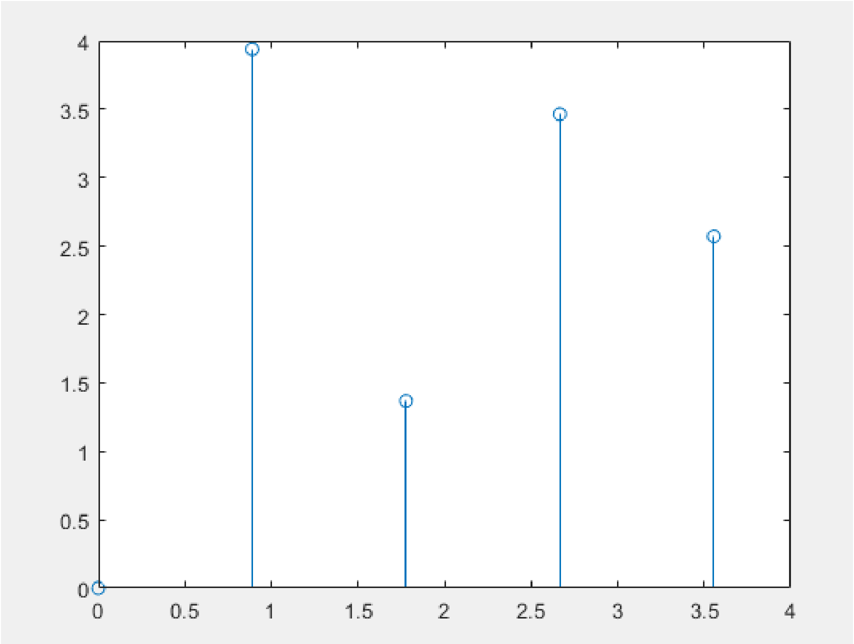
Discretizzazione con N=9 campioni in [0,1]
Δ=1/(N-1)=1/(9-1)=1/8
t_j=j Δ=j/8, j=0,..., 8

Discretizzazione con N=19 campioni in [0,1]
Δ=1/(N-1)=1/(19-1)=1/18
t_j=j Δ=j/18, j=0,..., 18



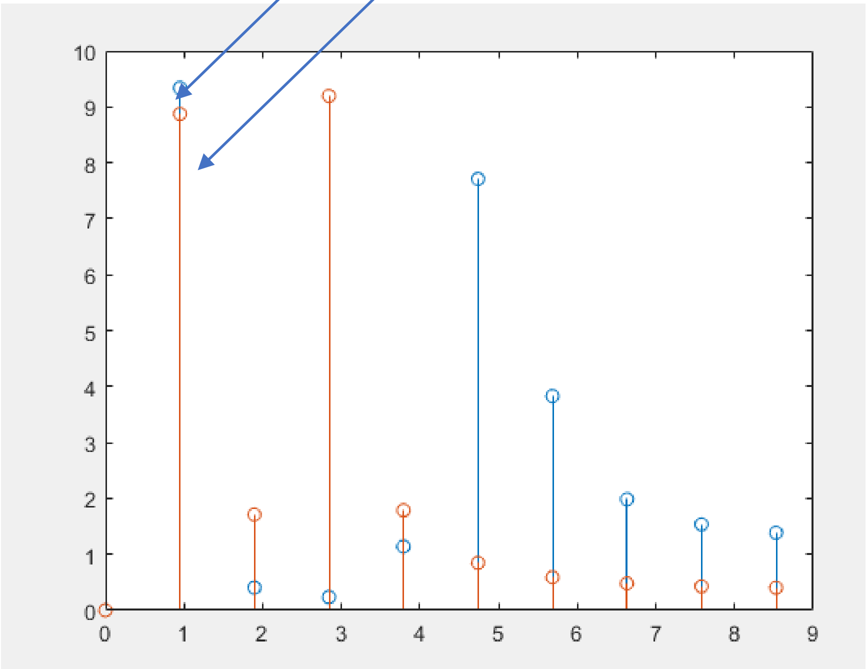


Le frequenze di f1 e f2 sono Sovrapposte!



f1 (blu)

f2 (arancione)

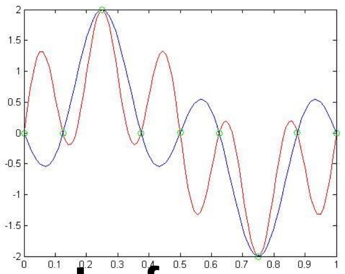


Le frequenze più alte sono confuse con quelle più basse

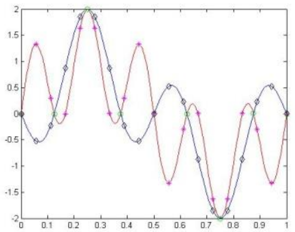
Aumentando il numero di nodi *si ottiene correttamente la funzione a frequenza maggiore.*

Esempio: f1(x)= sin(2*pi*t)+sin(2*pi*5*t), e f2(x)= sin(2*pi*t)-sin(2*pi*3*t)

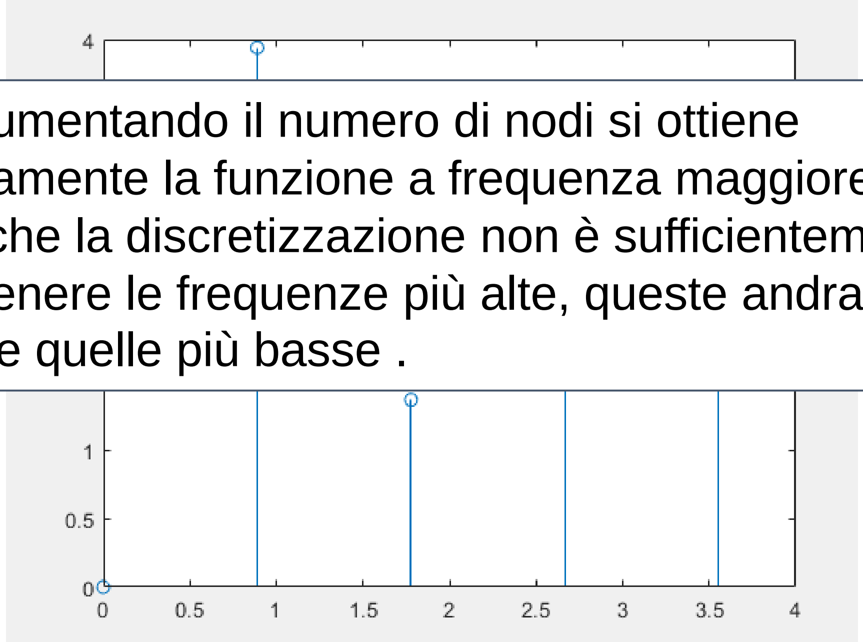
Discretizzazione con N=9 campioni in [0,1]
Δ=1/(N-1)=1/(9-1)=1/8
t_j=j Δ=j/8, j=0,..., 8



Discretizzazione con N=19 campioni in [0,1]
Δ=1/(N-1)=1/(19-1)=1/18
t_j=j Δ=j/18, j=0,..., 18



Le frequenze di f1 e f2 sono Sovrapposte!

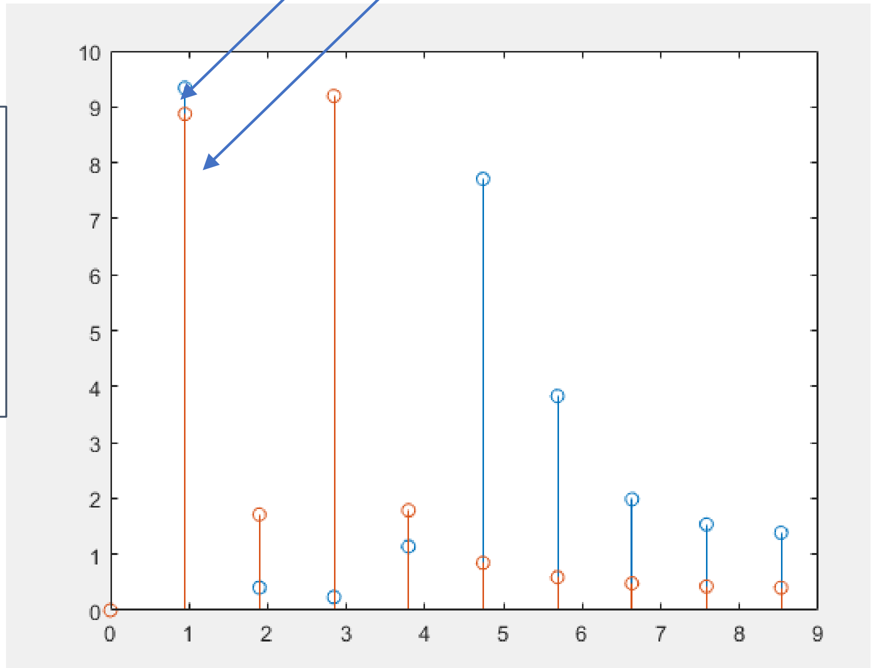


Solo aumentando il numero di nodi si ottiene correttamente la funzione a frequenza maggiore, ossia, fino a che la discretizzazione non è sufficientemente fitta per ottenere le frequenze più alte, queste andranno confuse quelle più basse .

Le frequenze più alte sono confuse con quelle più basse

f1 (blu)

f2 (arancione)

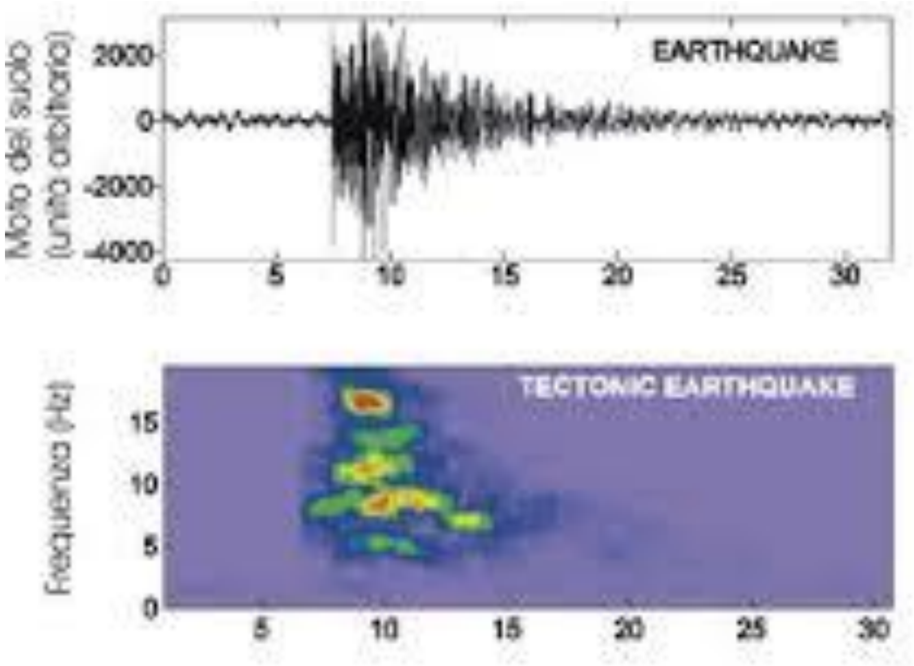


Aumentando il numero di nodi *si ottiene correttamente la funzione a frequenza maggiore.*

spettrogramma

Un segnale non periodico ha frequenze diverse in tempi diversi (musica, voce, evento sismico..)

La DFT dà informazioni sul peso delle diverse frequenze di un segnale
Manca però qualsiasi informazione sull'evoluzione spettrale del segnale nel tempo

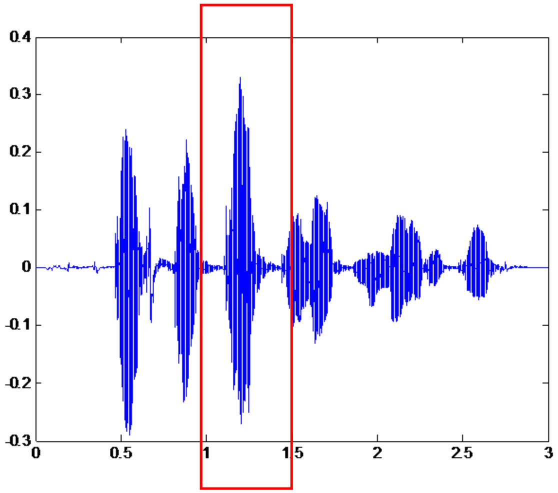


Il legame tra la DFT e la FT per funzioni NON periodiche

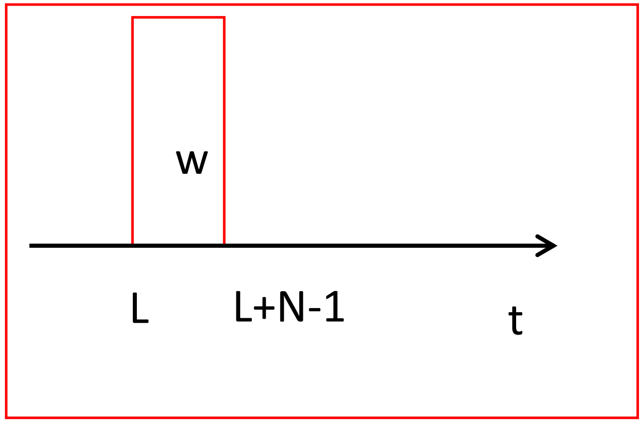
si può calcolare localmente lo spettro e si ha l'idea del contenuto spettrale del segnale in quell'intervallo di tempo

Segnale di 3 min. con F_S =1024Hz
si divide in "pezzi" lunghi N=128, si calcola la DFT in ogni pezzo, si ottengono informazioni in ogni intervallo di 128/1024=1/8sec.

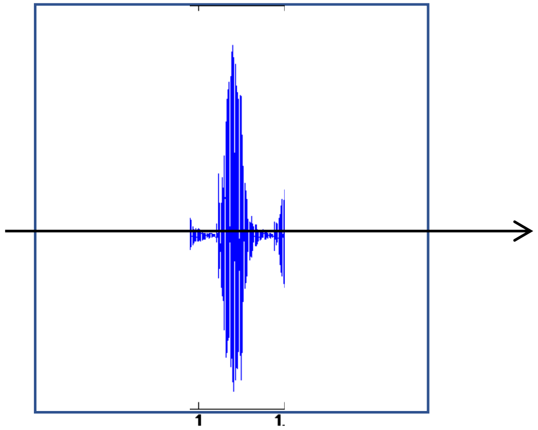
Per fare ciò si «moltiplica» per una “finestra” w di lunghezza N e si calcola la DFT



X



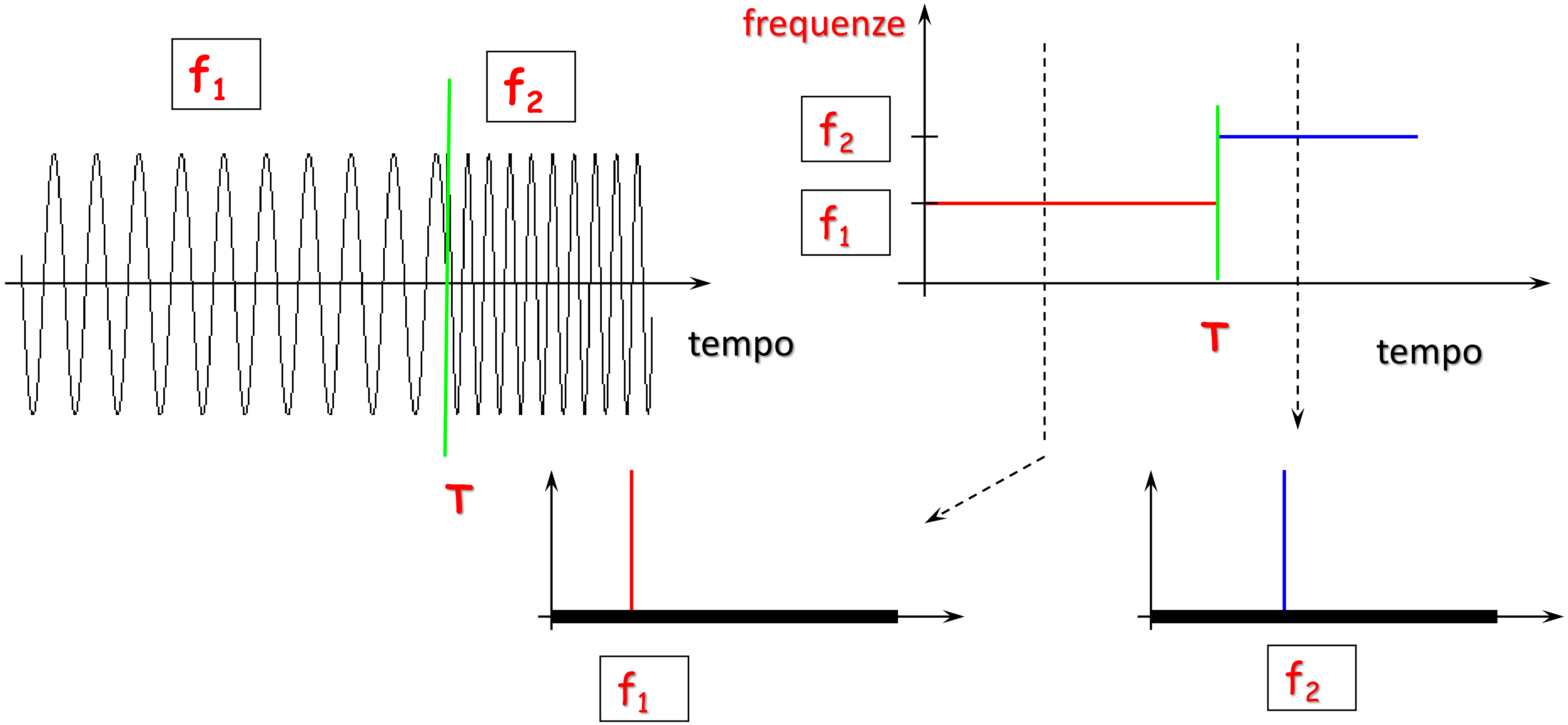
=



FFT

Short Time Fourier Transform

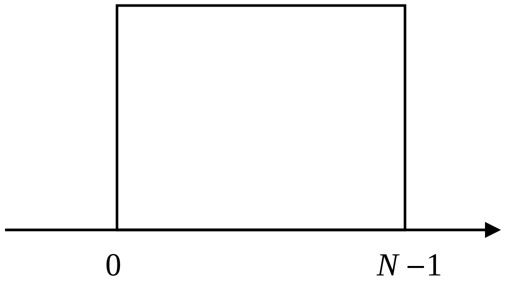
l'asse verticale rappresenta la frequenza, l'asse orizzontale il tempo



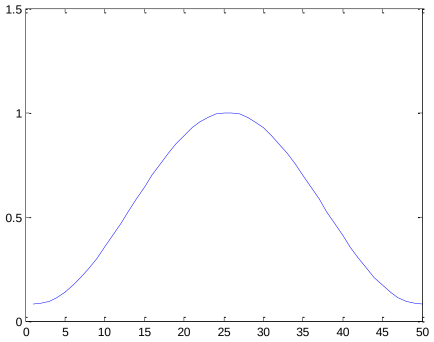
Questo è ciò che si vorrebbe ottenere

le finestre più usate

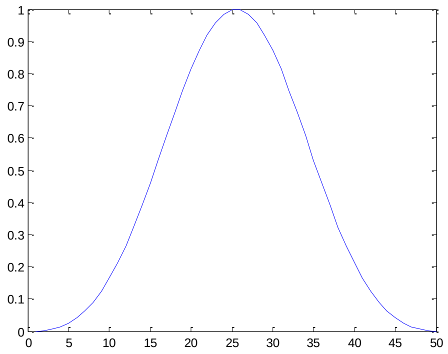
Rettangolare



Hamming



Blackman



Le finestre più usate sono quella di Hamming e di Blakmann (riducono l’errore di troncamento)

spettrogramma

spettrogramma (o sonogramma per segnali sonori)
l'asse verticale rappresenta la frequenza
l'asse orizzontale il tempo(o viceversa)
la scala cromatica l’ampiezza (in dB)
ogni componente è rappresentata da una linea che si muove nel tempo

spettrogramma in Matlab

spectrogram(y, W(N), overlap, NF,Fs,'yaxis')

y = segnale
W(N) = tipo di finestra e lunghezza(di default è la finestra di Hamming)
overlap =numero di campioni di overlap di ogni finestra (di default il 50%)
NF = dimensione della FFT relativa alla finestra(se >N effettua lo zero padding)
F_S = frequenza di campionamento
yaxis = tempo sull'asse y

Osservazione:sull'asse delle frequenze la scala è in Hz o KHz (dipende dell'ordine di grandezza delle frequenze)

N (lunghezza della finestra) influenza la risoluzione in frequenza

F_s /N (errore di discretizzazione nella FT)

e quella nel tempo N/Fs (errore di troncamento nella FT)

(in maniera inversamente proporzionale)

Fs=44100

N=2048

N/Fs = 0.04 sec di segnale per la finestra

Fs/N= 1/T= frequenza fondamentale=

risoluzione in frequenza=21.53Hz

N=4096

N/Fs = T= 0.09 sec di segnale per la finestra

Fs/N=risoluzione in frequenza=10.7Hz

Costruiamo un segnale con frequenze che variano nel tempo

```
x=linspace(0,1,1024); %Fs=1024
 s1=sin(2*pi*80*x);
 s2=sin(2*pi*160*x);
s=[s1 zeros(1, 128) s2 zeros(1, 128) s1+s2];
```

s1,poi una pausa di 128/1024=0.125sec, s2, altra pausa, segnali sommati.Si ha che:

N= 3072+(128*2)= 3328, Tempo= N/Fs=3328/1024=3.25sec

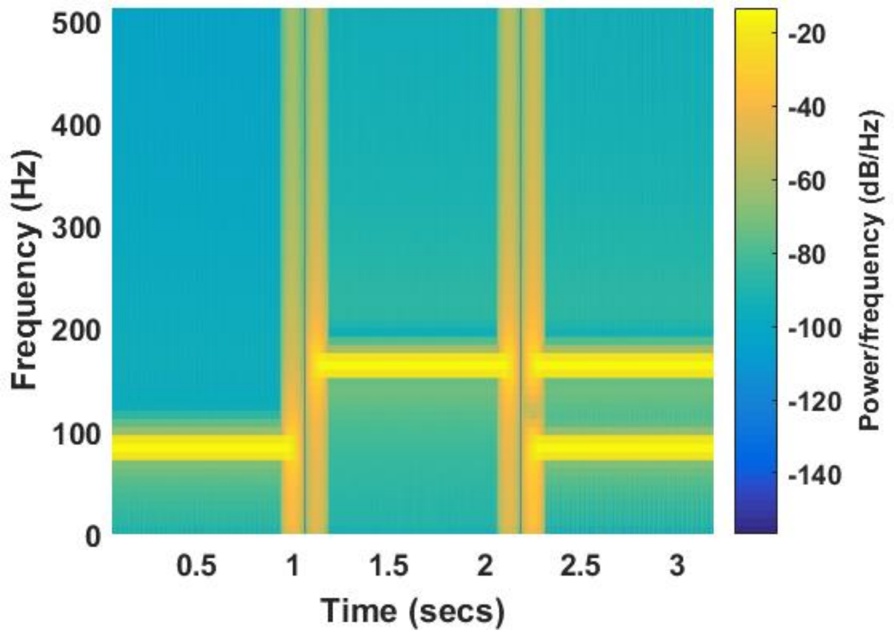
Usiamo la finestra di Hamming

N=128

risoluzione temporale

128/1024 = 0.125sec.

spectrogram(s, hamming(128),...
127, 128,1024,'yaxis')



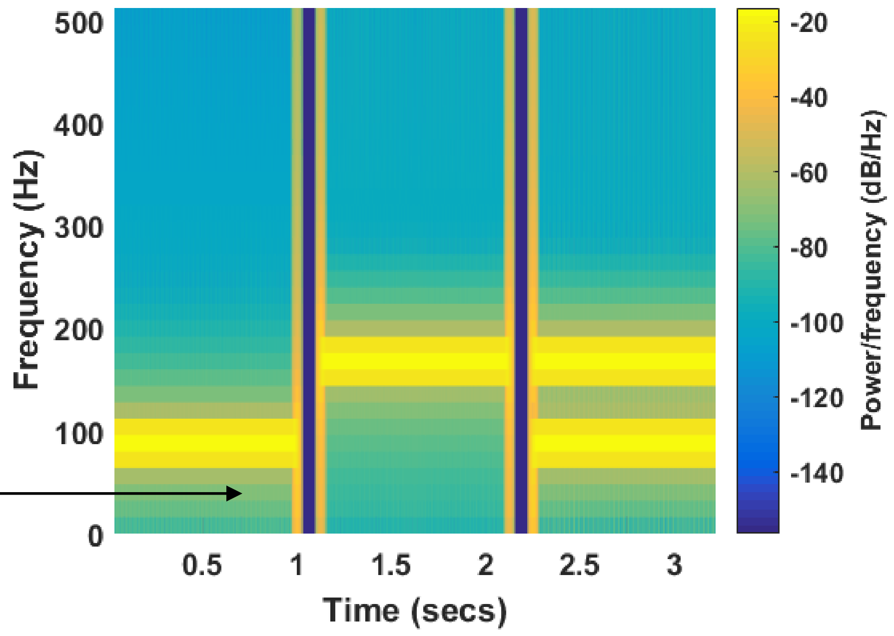
Diminuiamo la finestra,

N=64

risoluzione temporale

N=64/1024=0.0625sec.

spectrogram(s, hamming(64), 63,...
64,1024,'yaxis')



Migliora la risoluzione nel tempo
ma peggiora in frequenza