DIPARTIMENTO DI
ELETTROTECNICA
ED ELETTRONICA

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

ESAME DI ELABORAZIONE NUMERICA DEI SEGNALI

ANNO ACCADEMICO 1999 / 2000

PROGRAMMI IN MATLAB

Prof. Ing. *Ciro Cafforio*

Ing. *Pietro Guccione*

Eseguito da :

MARINELLI VITO
Matr. 507566 U

Questa relazione potrete trovarla in formato HTML al sito :
http://digilander.iol.it/marinel/vito/elaborazione/elaborazione.htm

Sempre nel sito : http://digilander.iol.it/marinel potrete trovare altro materiale didattico.

INDICE GENERALE

ARGOMENTI SVILUPPATI E RISPETTIVI NOMI DEI PROGRAMMI REALIZZATI pagina :

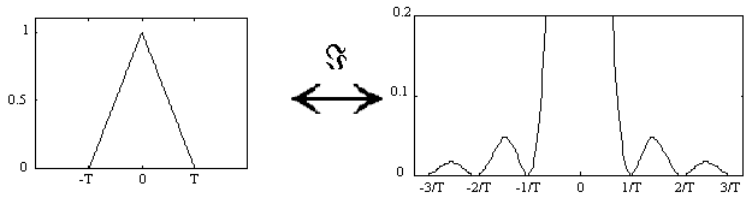
FUNCTION UTILIZZATE NEI VARI PROGRAMMI

Analisi dell'aliasing per segnali sinusoidali campionati senza rispettare il teorema del campionamento.
Lo studio verrà effettuato considerando un segnale sinusoidale di frequenza pari a 1 KHz.
Il programma in matlab la spiegazione teorica e le esecuzioni al variare del periodo di campionamento sono le seguenti:

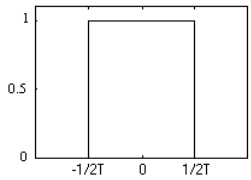
alias.m

```
% RAPPRESENTAZIONE DELL'ALIASING PER SEGNALI CAMPIONATI
%
% Considereremo un segnale sinusoidale [ sin(2*pi*f*k*T) ]
% con frequenza pari a f=1 kHz
% periodo di campionamento fissato dall'esterno

clear all
f=1000;
T=input('inserisci il periodo di campionamento in secondi : ');
t=T*1000;
strt=num2str(t);
K=input('inserisci il numero di periodi della sinusoide da rappresentare : ');
strK=num2str(K);
fase=input('inserisci la fase iniziale della sinusoide in radianti : ');
sf=num2str(fase/pi);
k_=K/(2*f*T) ;
k=(-k_:1:k_);
sen=sin(2*pi*f*k*T+fase);
k=k*t;
figure
plot(k,sen)
hold on
plot(k,sen,'r*')
axis([-k_*t k_*t -1 1]);
title(['f=1kHz, T=',strt,'ms e ',strK,' periodi del segnale di partenza ( ' ,sf,'*\pi rad )']);
xlabel('ms')
hold off
```

I campioni che visualizzeremo saranno interpolati utilizzando una spline lineare che implementa automaticamente la funzione PLOT del MATLAB.
Da questo si deduce che, anche se valgono le ipotesi del teorema del campionamento, il segnale interpolato sarà ugualmente affetto da aliasing tanto più evidente quanto più ci avviciniamo con la frequenza di campionamento alla frequenza di Nyquist e scendiamo addirittura sotto il suo valore, perché l'andamento nel tempo e lo spettro dell'interpolatore lineare sono le seguenti:



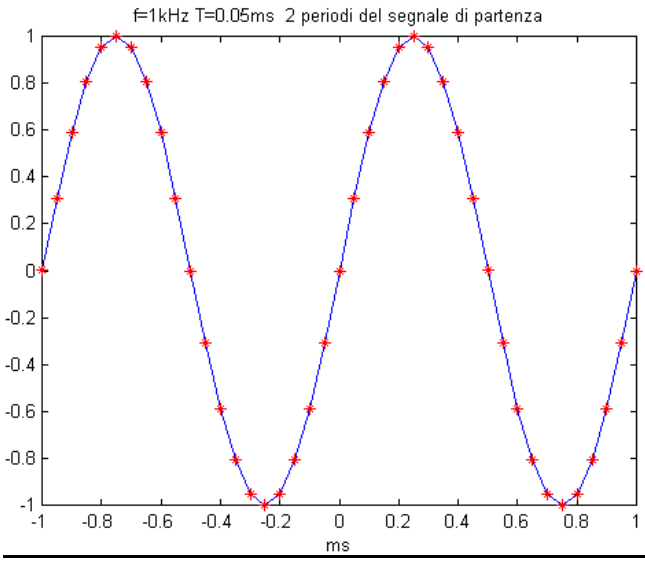
dove : $S_i(f) = T \cdot \sin c^2(f \cdot T)$
invece di utilizzare un normale filtro passa basso (ideale):

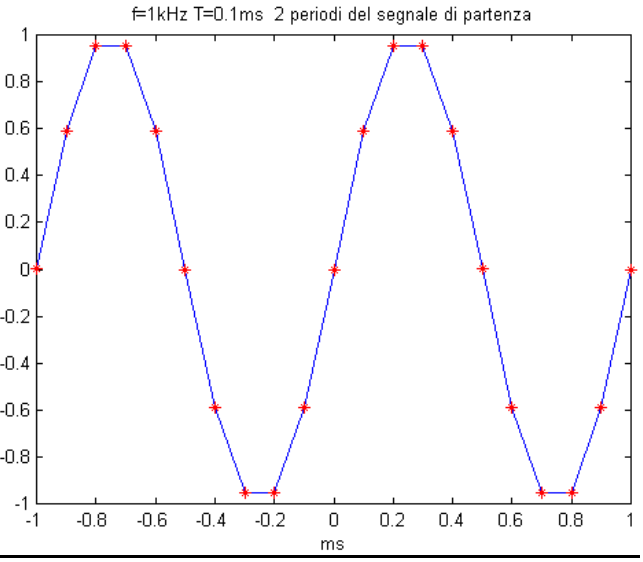


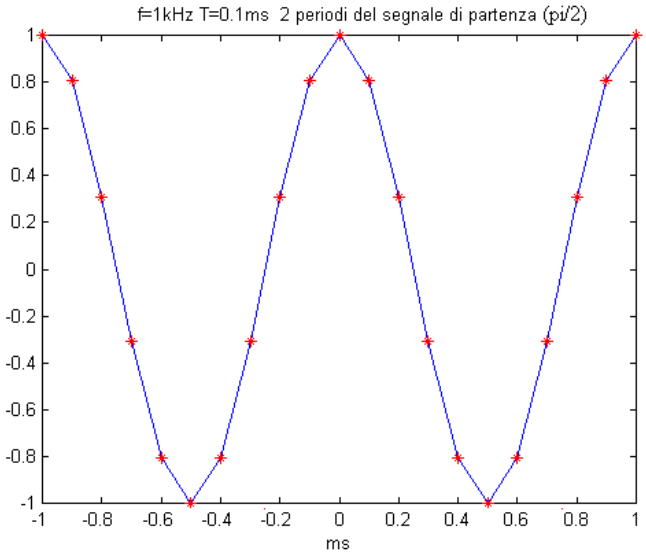
allora il lobo principale dello spettro dell'interpolatore utilizzato ha un'ampiezza in frequenza doppia di quella del filtro passa basso.
In definitiva avremo che il segnale ricostruito e il suo spettro sono i seguenti:

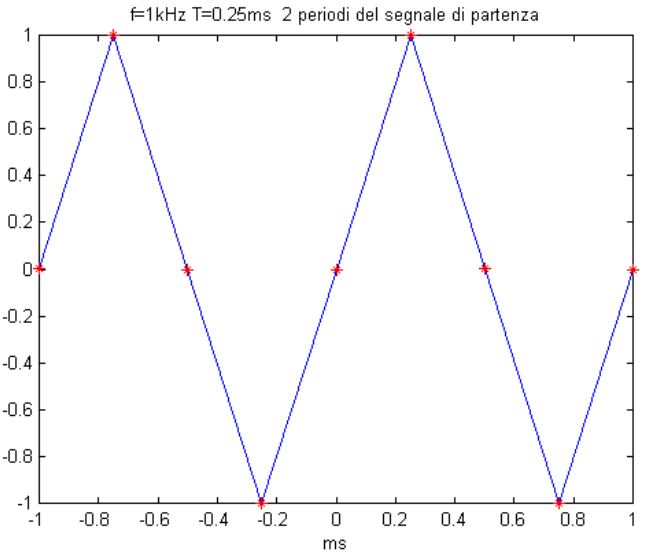
$$\begin{cases} s_r(t) = s(t) * s_i(t) \\ S_r(f) = S(f) \cdot S_i(f) \end{cases} \quad \text{(ottenendo gli andamenti a spezzata che vedremo successivamente)}$$

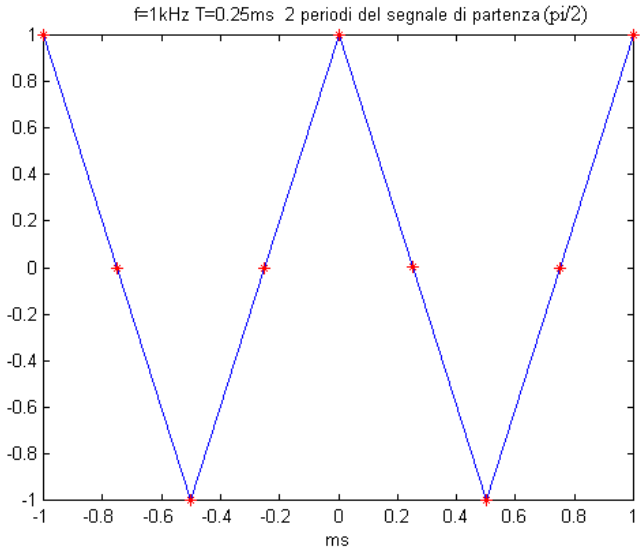
OSS.: Per ogni tabella che verrà visualizzata viene precisato il periodo di campionamento utilizzato e il numero di periodi del segnale di partenza considerati in modo tale da fare un confronto con il segnale che effettivamente vogliamo campionare. Consideriamo innanzitutto il caso in cui andiamo a sovracampionare in maniera abbastanza evidente, prendendo un numero di campioni in ogni periodo pari a 20. Le immagini successive, oltre ad avere un periodo di campionamento sempre più basso, sono realizzate anche sfasando di un certo angolo il segnale di partenza (alcune volte non sarà specificato perché lo sfasamento si vede che è di $\pi/2$.

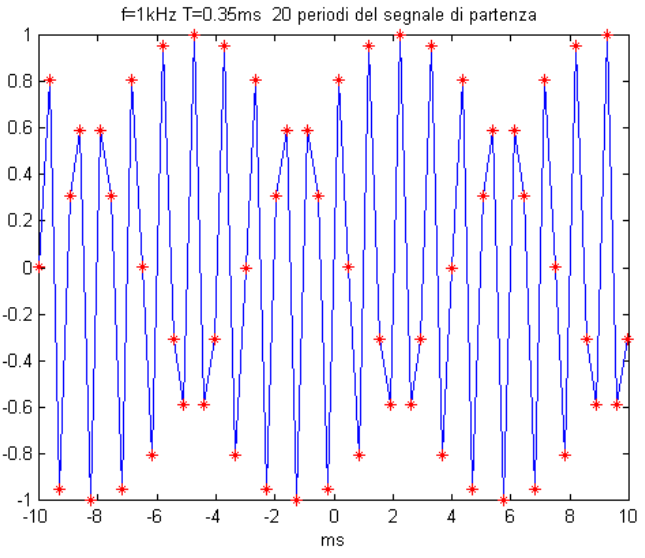

f=1kHz T=0.05ms 2 periodi del segnale di partenza
ms

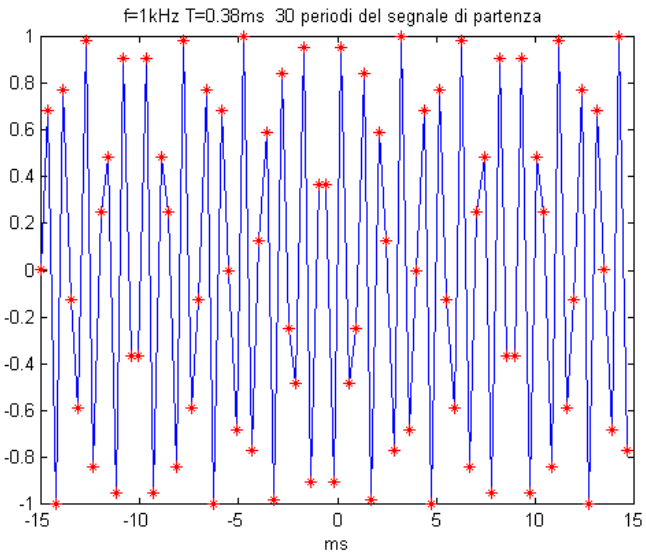

f=1kHz T=0.1ms 2 periodi del segnale di partenza
ms

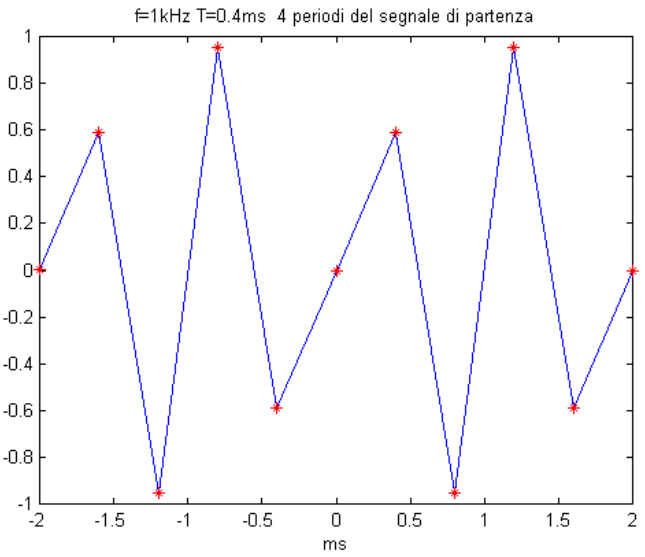

f=1kHz T=0.1ms 2 periodi del segnale di partenza (pi/2)
ms

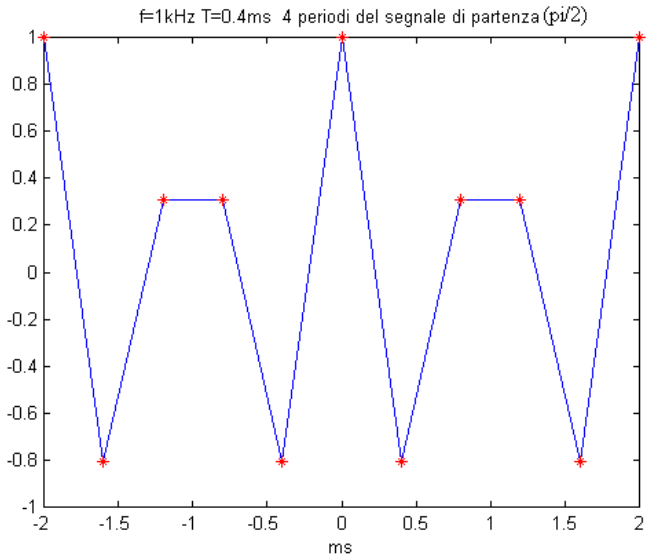

f=1kHz T=0.25ms 2 periodi del segnale di partenza
ms

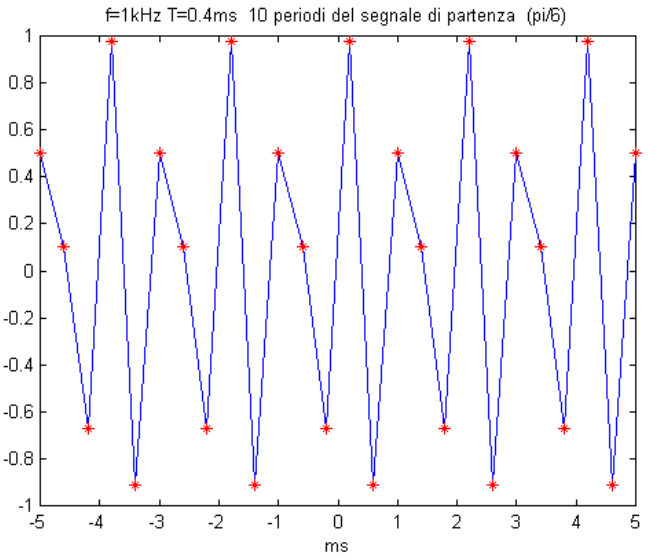

f=1kHz T=0.25ms 2 periodi del segnale di partenza (pi/2)
ms

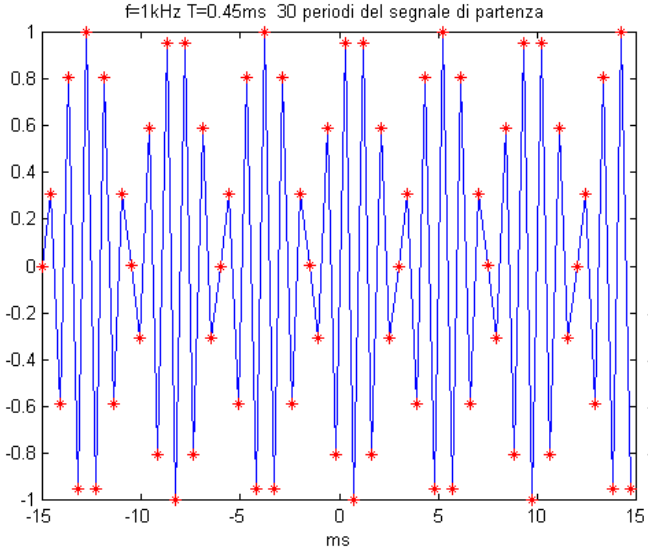

f=1kHz T=0.35ms 20 periodi del segnale di partenza
ms

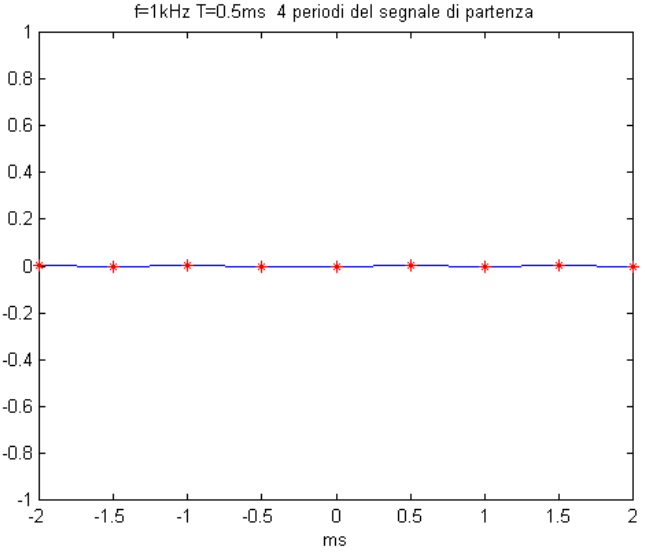

f=1kHz T=0.38ms 30 periodi del segnale di partenza
ms

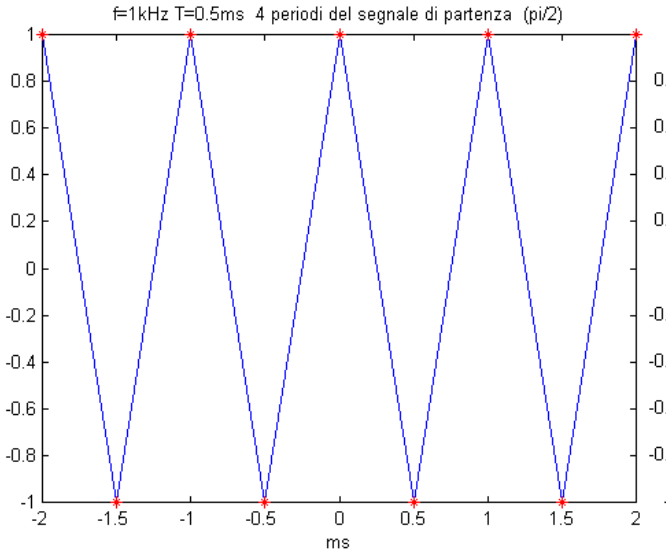

f=1kHz T=0.4ms 4 periodi del segnale di partenza
ms

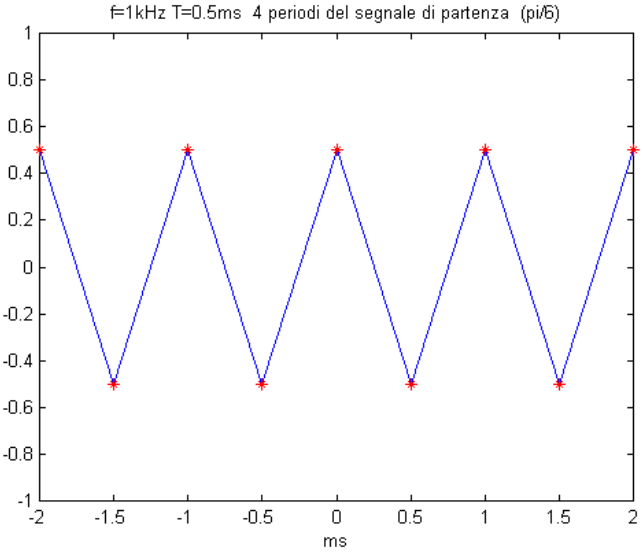

f=1kHz T=0.4ms 4 periodi del segnale di partenza (pi/2)
ms

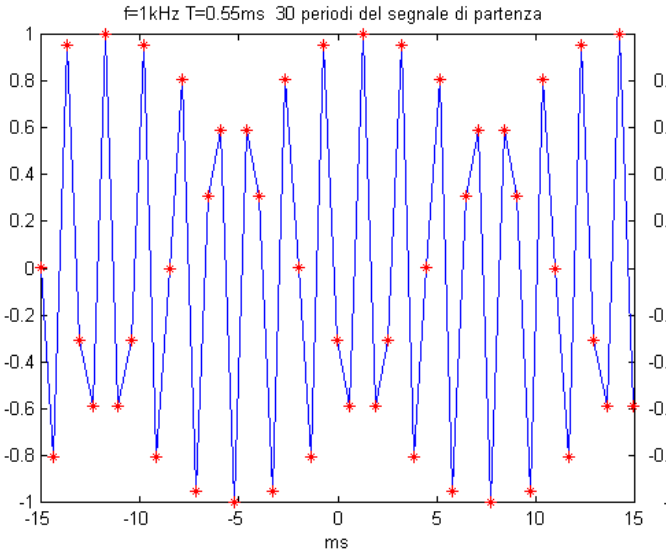

f=1kHz T=0.4ms 10 periodi del segnale di partenza (pi/6)
ms

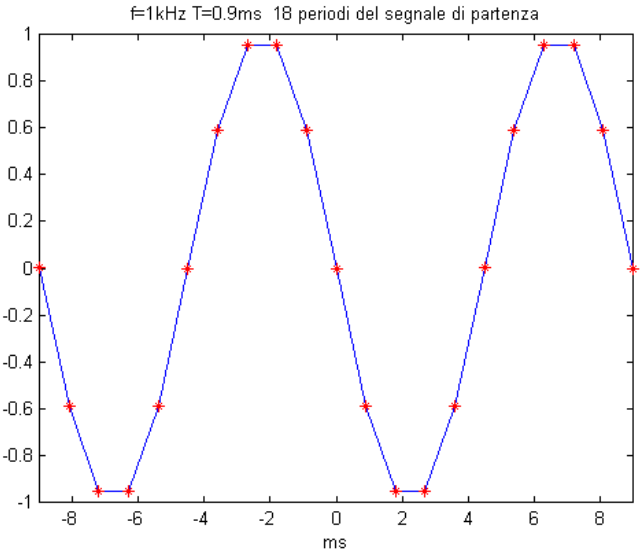

f=1kHz T=0.45ms 30 periodi del segnale di partenza
ms

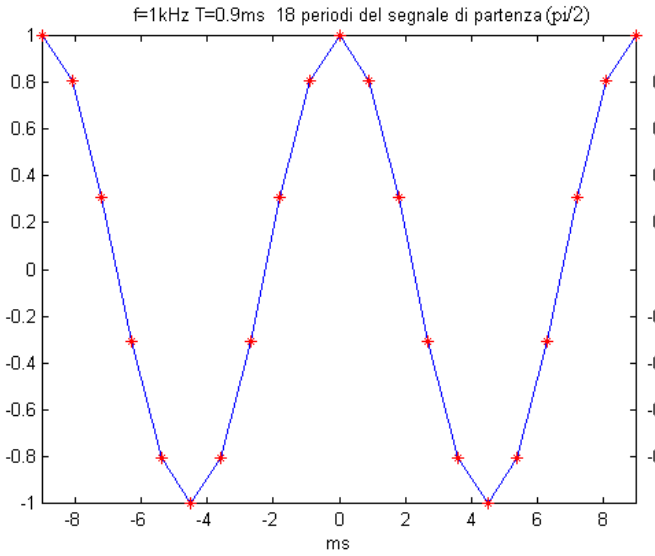

f=1kHz T=0.5ms 4 periodi del segnale di partenza
ms

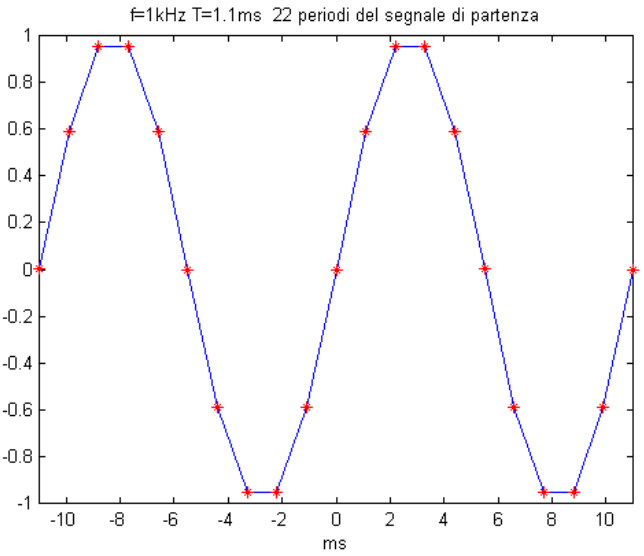

f=1kHz T=0.5ms 4 periodi del segnale di partenza (pi/2)
ms


f=1kHz T=0.5ms 4 periodi del segnale di partenza (pi/6)
ms


f=1kHz T=0.55ms 30 periodi del segnale di partenza
ms

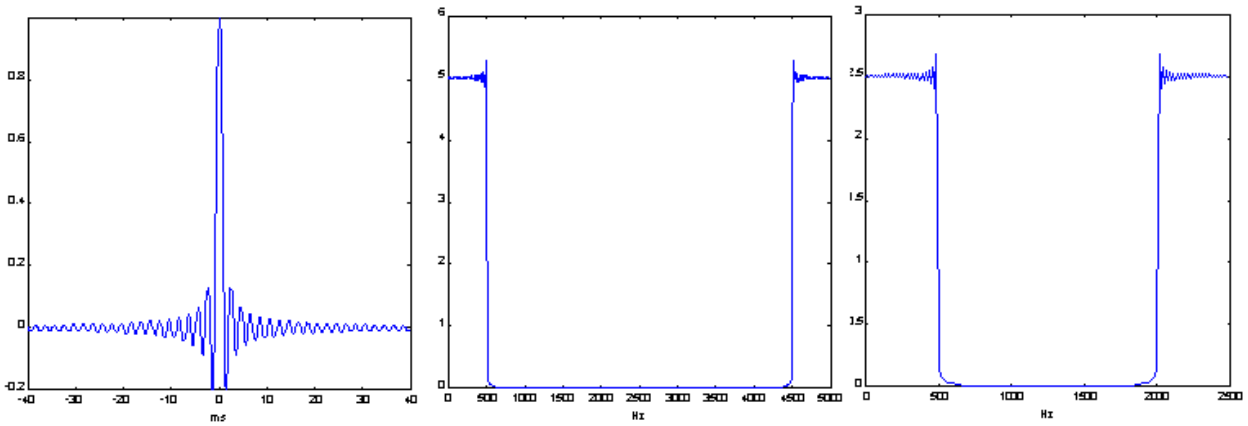

f=1kHz T=0.9ms 18 periodi del segnale di partenza
ms


f=1kHz T=0.9ms 18 periodi del segnale di partenza (pi/2)
ms


f=1kHz T=1.1ms 22 periodi del segnale di partenza
ms

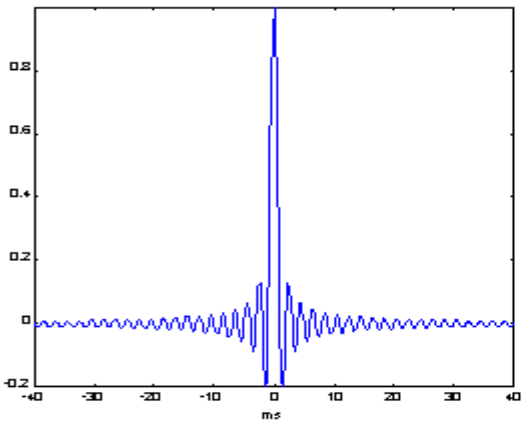
1) Campioniamo un segnale a banda praticamente limitata considerando un numero abbastanza elevato di campioni e calcoliamo la DFT. Dopo di che effettuiamo una decimazione dei campioni eliminando quelli di posto pari nel dominio del tempo ed effettuiamo la DFT nuovamente : lo spettro periodico ora dovrà avere le repliche ad una distanza INFERIORE.
Consideriamo come segnale un sinc traslato e troncato fino al ventesimo lobo:

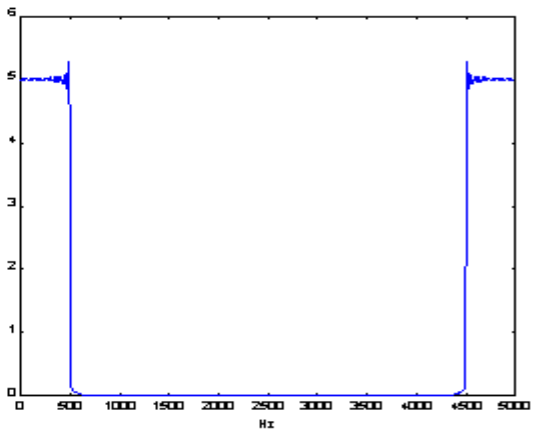
```
k=[-201:1:201];
f0=1000;
fc=5000;
x=k/fc;
y=sin(pi*f0*x)./(pi*f0*x);
y(202)=1;
plot(x*1000,y)
xlabel('ms');
ff=fft(y);
figure
freq=[0:5000/403:5000*(1-1/403)];
plot(freq,abs(ff))
xlabel('Hz');
zoom
j=1;
for i=1:403
   if i==2*j
      yi(j)=y(i);
      j=j+1;
   end
end
ffi=fft(yi);
m=length(yi);
freq=[0:2500/m:2500*(1-1/m)];
figure
plot(freq,abs(ffi))
xlabel('Hz');
```

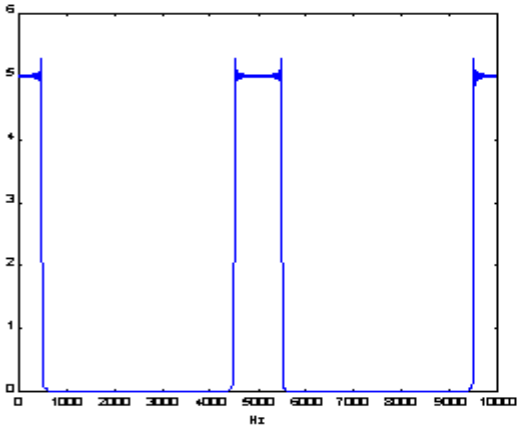


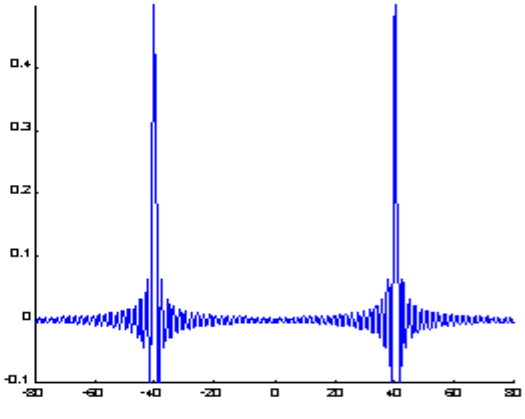
2) Consideriamo lo stesso segnale precedente, calcoliamo la DFT, inseriamo dei campioni nulli nel tempo ricalcoliamo la DFT: quello che dobbiamo ottenere è un ampliamento della finestra di osservazione nel dominio delle frequenze.Viceversa se gli zeri sono inseriti nello spettro e calcoliamo la IDFT.

```
k=[-201:1:201];
f0=1000;
fc=5000;
x=k/fc;
y=sin(pi*f0*x)./(pi*f0*x);
y(202)=1;
plot(x*1000,y)
xlabel('ms');
ff=fft(y);
figure
freq=[0:5000/403:5000*(1-1/403)];
plot(freq,abs(ff))
xlabel('Hz');
zoom
j=1;
for i=1:403
   yi(j)=y(i);
   yi(j+1)=0;
   j=j+2;
end
ffi=fft(yi);
m=length(yi);
freq=[0:10000/m:10000*(1-1/m)];
figure
plot(freq,abs(ffi))
xlabel('Hz');
j=1;
for i=1:403
   ffii(j)=ff(i);
   ffii(j+1)=0;
   j=j+2;
end
yii=ifft(ffii);
n=length(yii);
xi=[2*x(1):(x(403)-x(1))/402:2*x(403)+(x(403)-x(1))/402];
figure
plot(xi*1000,real(yii))
```



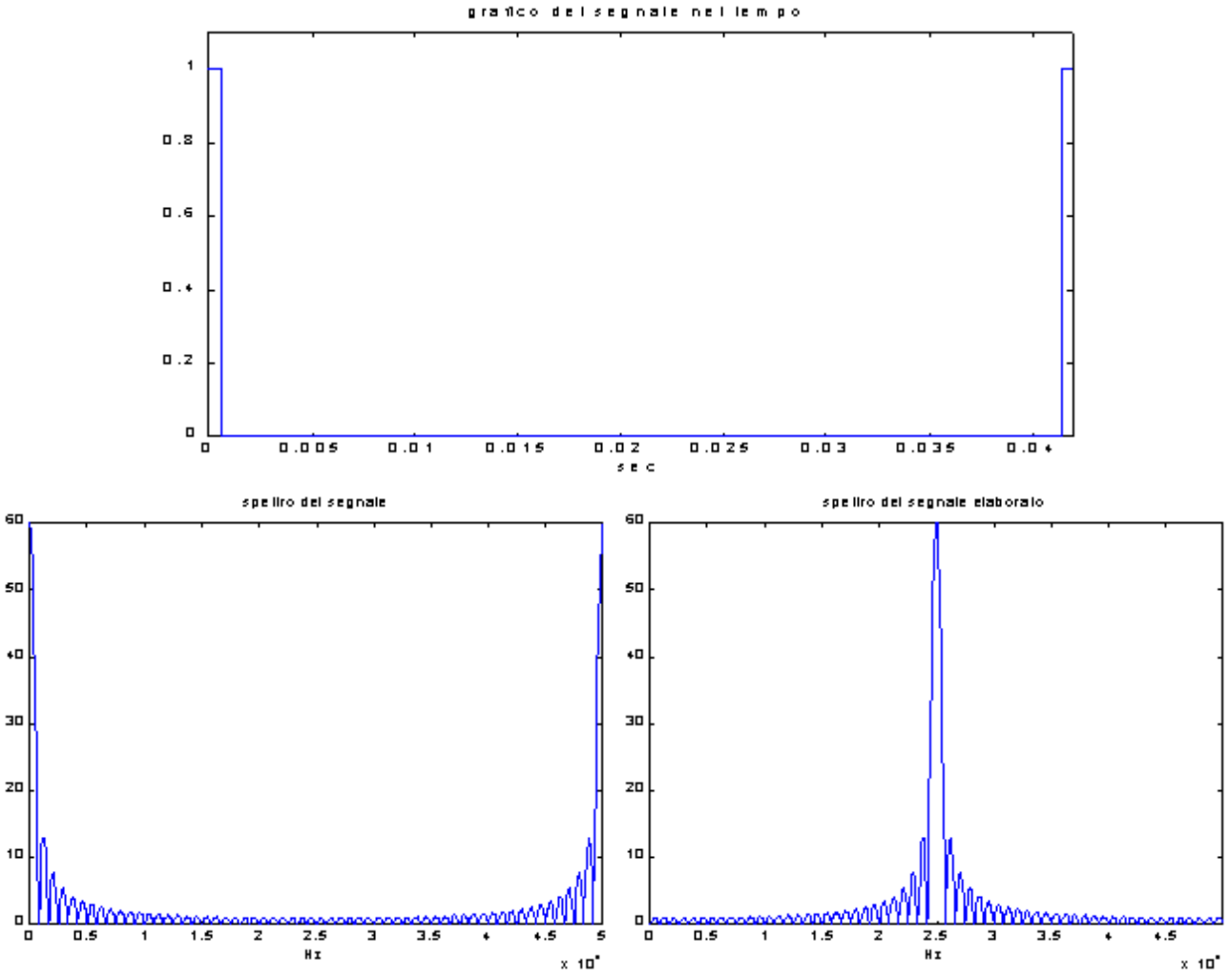






3) Campioniamo un segnale qualsiasi considerando un numero abbastanza elevato di campioni e calcoliamo la DFT. Dopo di che cambiamo alternativamente il segno del vettore dei campioni nel dominio del tempo ed effettuiamo la DFT nuovamente : lo spettro periodico ora dovrà essere traslato in frequenza di una quantità pari alla frequenza di Nyquist.
Consideriamo come segnale un rettangolo periodicizzato:

```
clear all
tempo=42e-3;
T=1/50000;
campioni=round((tempo/T));
k=(1:1:campioni);
xf=(k-1)*T;
yf=[ones(1,campioni/70) zeros(1,(34/35)*campioni) ones(1,campioni/70)];
figure
plot(xf,yf);
xlabel('sec');
title('grafico del segnale nel tempo');
zoom
ff=fft(yf);
figure
freq=[0:1/(T*campioni):(1/T)*(1-1/campioni)];
plot(freq,abs(ff));
xlabel('Hz');
title('spettro del segnale');
zoom
for i=1:length(ff)
   yi(i)=yf(i)*cos(pi*(i-1));
end
ffi=fft(yi);
figure
plot(freq,abs(ffi));
xlabel('Hz');
title('spettro del segnale elaborato');
zoom
```

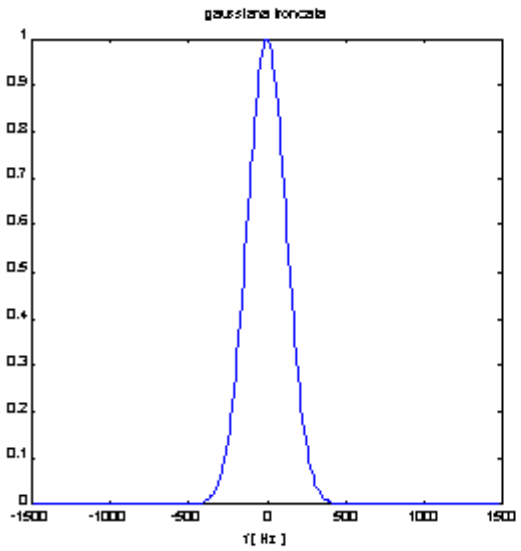


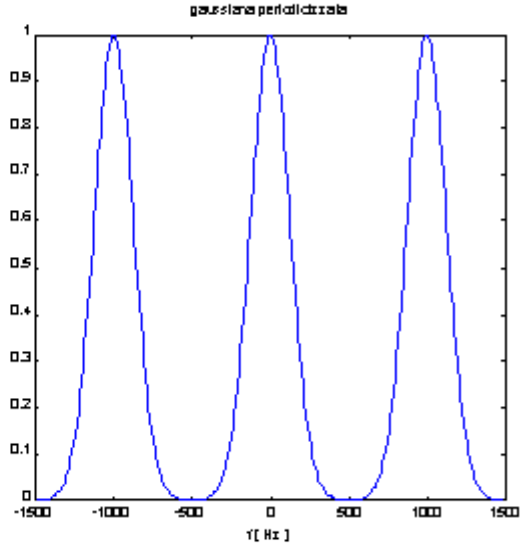
4) Sia G(f) una funzione gaussiana e considero la funzione : $F(f) = \frac{G(f)}{\sum G(f - kf_c)}$ allora si dimostra che F(f) è la trasformata di un sinc (un rettangolo) se G(f) ha un andamento molto dolce nel tempo e limitata in frequenza (deve essere la trasformata di un segnale passa-basso). Ovviamente la G(f) considerata non è limitata in frequenza però basta prendere un'opportuna finestra in frequenza affinché gli errori che commettiamo sono trascurabili. Fisso f_c = 1 kHz e quindi prendo come G(f) la funzione :

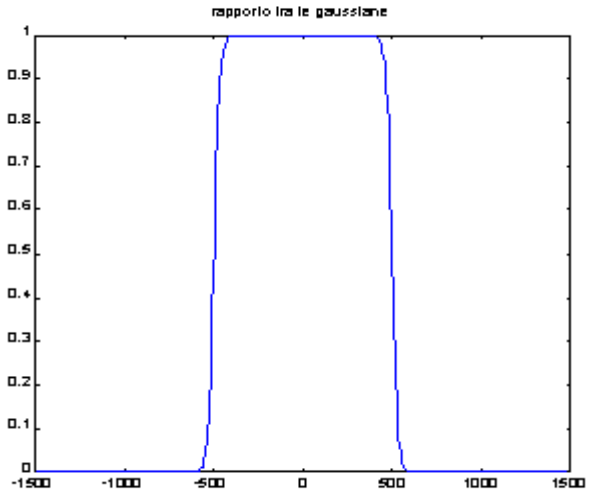
$$G(f) = e^{-\frac{f^2}{31250}}$$

in maniera tale che $G\left(\frac{f_c}{2}\right) = 0.033\%$ del valore massimo (approssimazione molto buona):

```
k=[-150:1:150];
fi=3000;
fc=1000;
f=k*fi/301;
R=exp((-f.^2)/31250);
Ri=exp((-f.^2)/31250)+exp((-(f-fc).^2)/31250)+exp((-(f+fc).^2)/31250);
plot(f,R)
xlabel('f [ Hz ]');
title('gaussiana troncata');
figure
plot(f,Ri)
xlabel('f [ Hz ]');
title('gaussiana periodicizzata');
rap=R./Ri;
figure
plot(f,rap);
title('rapporto tra le gaussiane');
```







Costruiamo adesso un interpolatore lineare mettendo in evidenza l'effetto del ricampionamento di un fattore non intero e dell'interpolazione su dati non sufficientemente sovracampionati

Interlin.m

```
% COSTRUZIONE DI UN INTERPOLATORE LINEARE

% Considereremo un segnale generico da introdurre dall'esterno

% può essere inserito anche un fattore non intero di ricampionamento

clear all

tempo=input('inserisci l''istante finale di rappresentazione del segnale (in sec) : ');
f1=input('inserisci la frequenza di campionamento (in Hz) : ');
T1=1/f1;
err=0;
while err==0
   F=input('inserisci il nome della function contenente il segnale : ','s');
   if exist(F)==0
      disp('LA FUNZIONE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end

T=1/(100*f1);
k_=(tempo/T)+1;
k=(1:1:k_);
x=(k-1)*T;
y=feval(F,x);
x=(k-1)*T*1000;
figure
plot(x,y,'y');
xlabel('ms');
hold on

T=T1;
k_=(tempo/T)+1;
k=(1:1:k_);
x=(k-1)*T;
y=feval(F,x);
x=(k-1)*T*1000;
if length(x)<40
   plot(x,y,'r*');
end
plot(x,y,'g');

disp('OPERAZIONE DI RICAMPIONAMENTO ');

valore=input('introduci il fattore di ricampionamento : ');
n=length(y);
valori=valore*(n-1)+1;
T1=x(n)-x(1);
T2=T1/(valori-1);
xi(1)=x(1);
for i=1:valori-1
   xi(i+1)=xi(i)+T2;
end
m=length(xi);
for k=1:m
   for i=2:n
      if xi(k)>=x(i-1) & xi(k)<=x(i)
         yi(k)=y(i-1)+((y(i)-y(i-1))/(T*1000))*(xi(k)-x(i-1));
      end
   end
end
yi(m)=y(n-1)+((y(n)-y(n-1))/(T*1000))*(xi(m)-x(n-1));
```

```
if m<55
   plot(xi,yi,'b+');
else
   plot(xi,yi,'b');
end
hold off
zoom

ff=fft(yi);
freq=[0:(f1*valore)/m:f1*valore*(1-1/m)];
figure
plot(freq,abs(ff))
xlabel('Hz');
zoom
```

segnale.m

```
function f=segnale(x);

f=sin(2*pi*200*x)+sin(2*pi*350*x+pi/7)+sin(2*pi*420*x-pi/4);      % la frequenza è in Hz
```

impulso.m

```
function i=impulso(x)

n=length(x);
i(n)=0;
i(round(n/2))=1;
```

rect1.m

```
function r=rect1(x)

n=length(x);
r(n)=0;
for i=round(2*n/5) : round(3*n/5)
   r(i)=1;
end
```

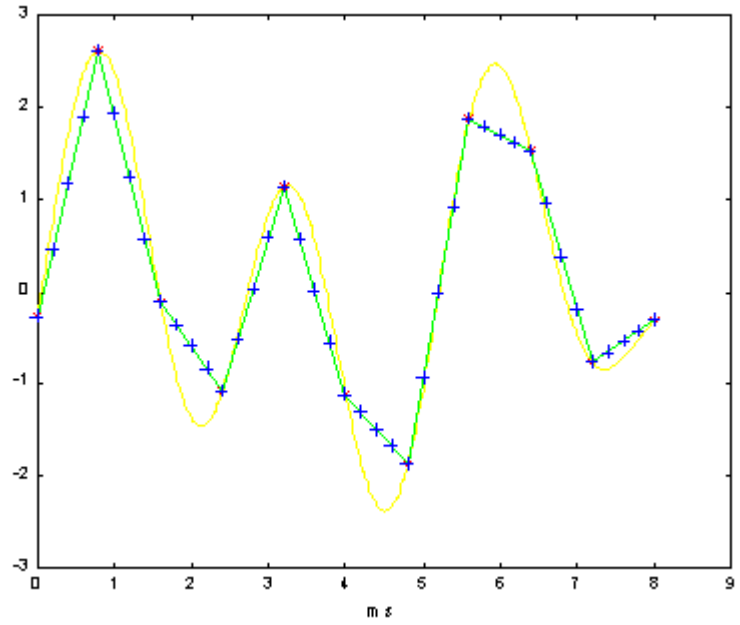
rect2.m

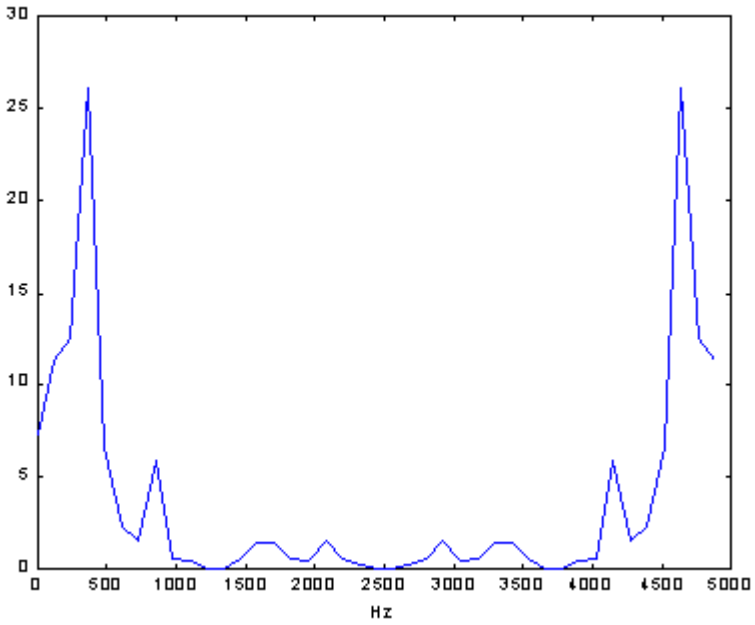
```
function r=rect2(x)

n=length(x);
r(n)=0;
for i=round(49*n/100) : round(51*n/100)
   r(i)=1;
end
```

esecuzione del programma

1) interpolazione di un segnale periodico

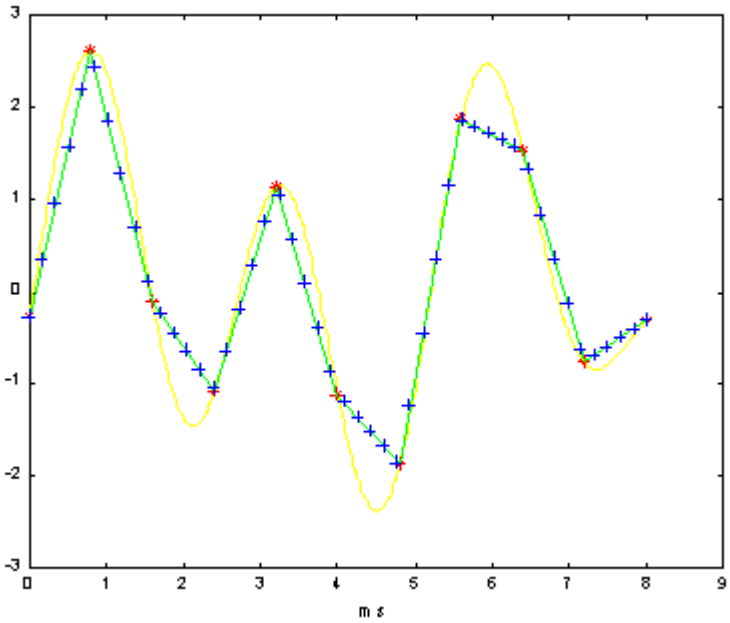
```
» interlin
inserisci l'istante finale di rappresentazione del segnale (in sec) : 8e-3
inserisci la frequenza di campionamento (in Hz) : 1250
inserisci il nome della function contenente il segnale : segnale
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 4
```





OSS.: la funzione rappresentata è la somma di tre sinusoidi, per cui la sua trasformata di Fourier dovrebbe avere 3 impulsi localizzati alle frequenze di 200 Hz, 350 Hz e 420 Hz : come si può notare da quest'ultimo grafico, invece è possibile individuare soltanto l'impulso a 420 Hz perchè il passo di campionamento e di sovracampionamento non sono molto alti per poter trascurare gli effetti di bordo che si hanno durante il calcolo della DFT (infatti ci sono anche impulsi di ampiezza inferiore localizzati a frequenze intermedie che sono indesiderati).
Per quanto riguarda il primo grafico con questo tipo di ricampionamento riusciamo a rispettare i dati : questo accade se introduciamo un fattore di ricampionamento intero. In caso contrario si avrà che il segnale ricampionato (punti in blu) si discosta dai valori effettivi nei punti in cui andiamo a campionare all'inizio (in rosso) come succede ora:

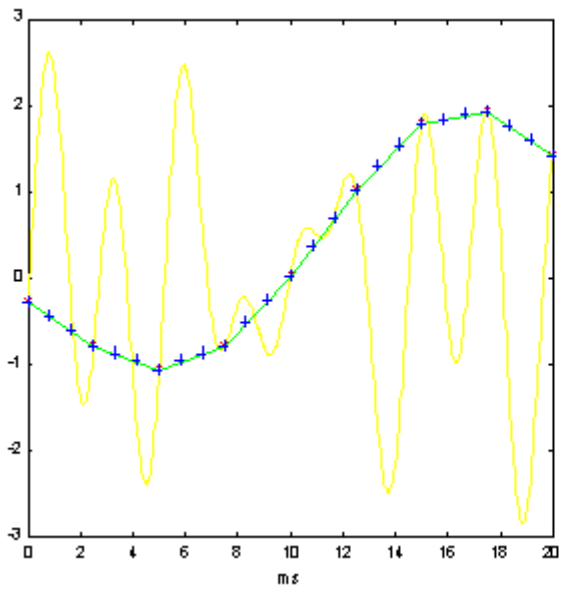
```
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 4.7
```

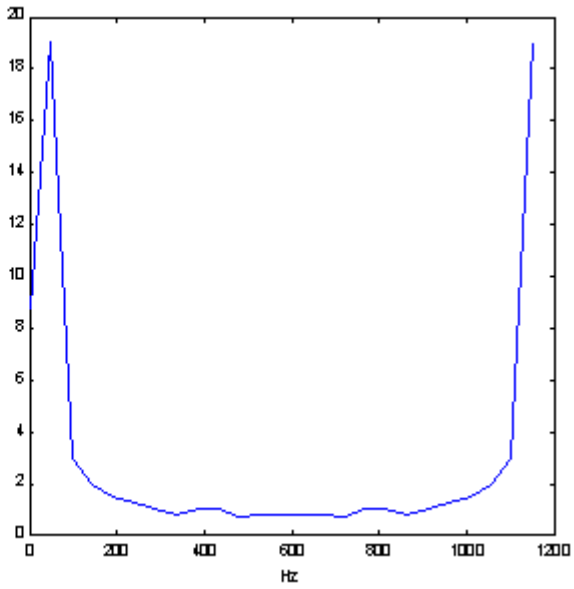


l'altro grafico e praticamente identico al caso precedente di ricampionamento intero

Nel caso in cui consideriamo sempre lo stesso segnale l'effetto dell'interpolazione su dati non sufficientemente sovracampionati è il seguente :

```
» interlin
inserisci l'istante finale di rappresentazione del segnale (in sec) : 20e-3
inserisci la frequenza di campionamento (in Hz) : 400
inserisci il nome della function contenente il segnale : segnale
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 3
```

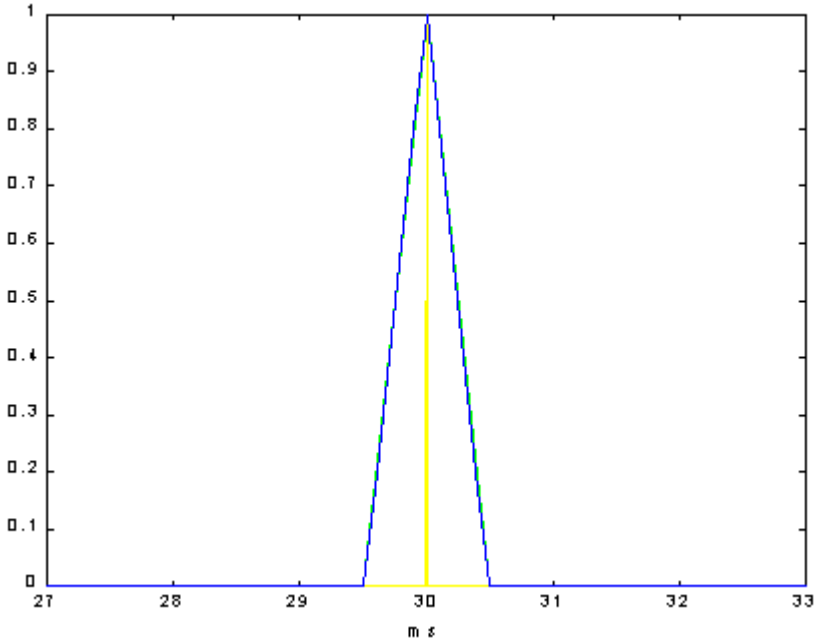


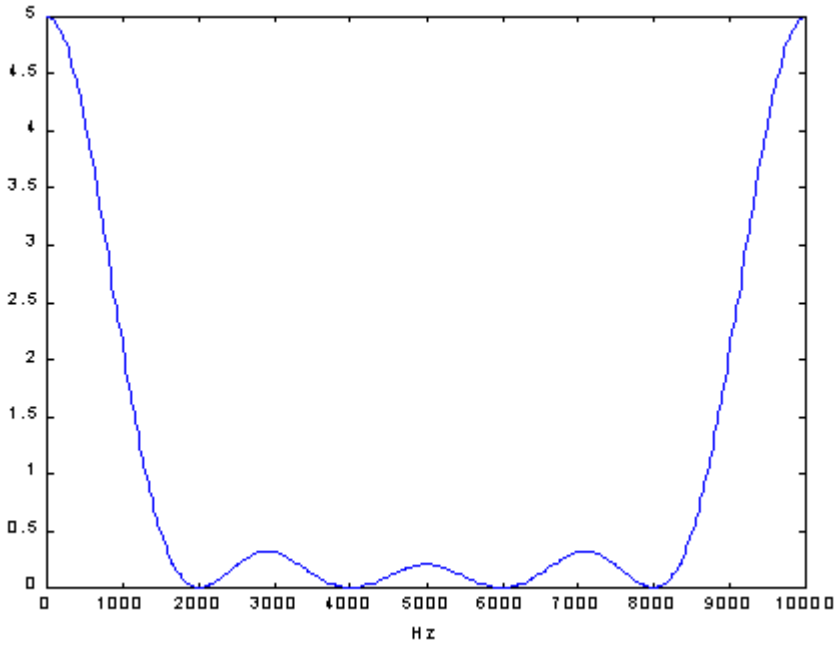


e osserviamo che non riusciamo assolutamente a ricostruire il segnale di partenza : lo spettro è anche completamente differente (non è più presente la riga alla frequenza di 420 Hz).

2) interpolazione di un segnale impulsivo

```
» interlin
inserisci l'istante finale di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2000
inserisci il nome della function contenente il segnale : impulso
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 5
```



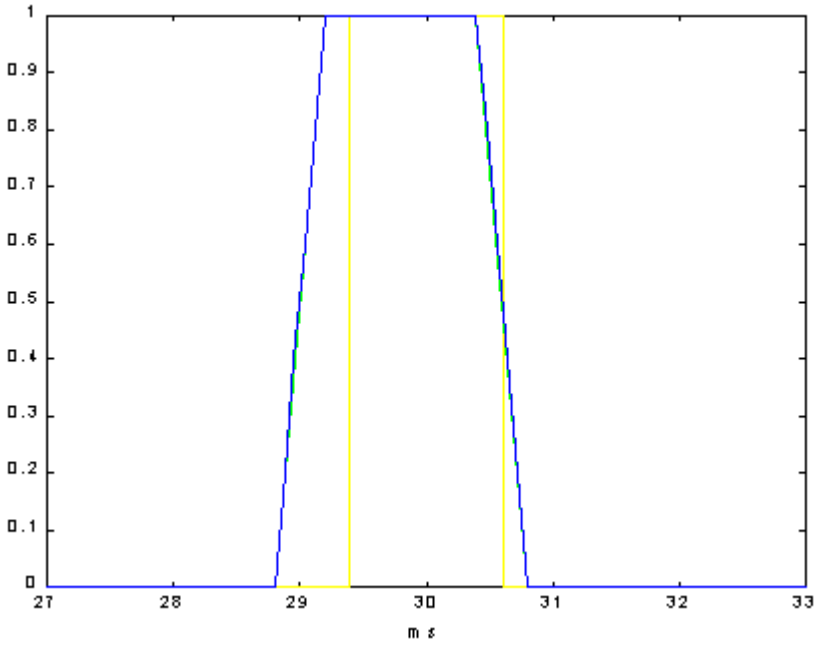


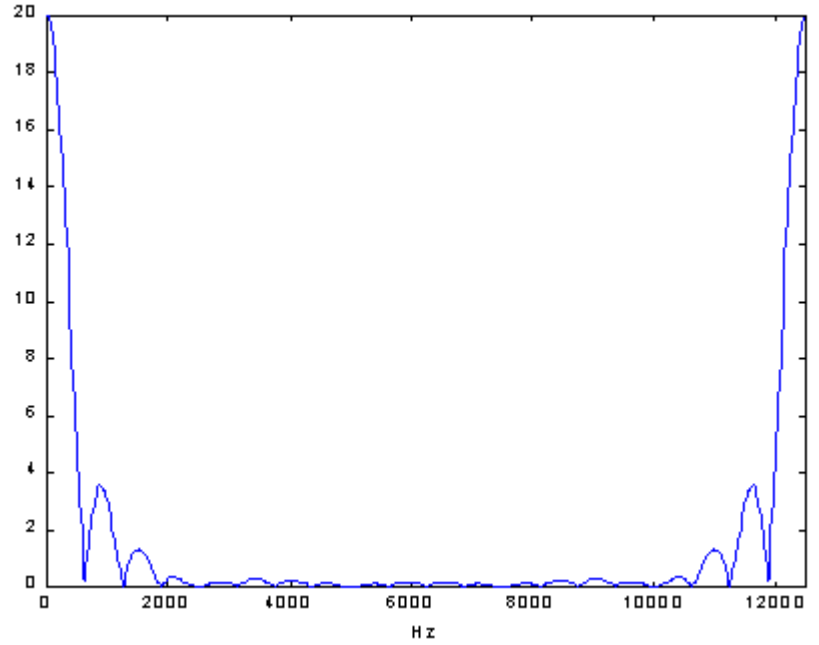
OSS.: il primo grafico è stato ingrandito nella finestra temporale per poter visualizzare meglio l'impulso centrato in un istante di tempo non nullo per evitare la presenza di effetti di bordo nel calcolo della DFT.

3) interpolazione di un segnale rettangolare

Consideriamo solo un rettangolo che ha ampiezza temporale pari a τ =1,6 ms , per poter osservare i lobi secondari (infatti il lobo principale termina alla frequenza $f_0 = 1/\tau = 625$ Hz:

```
» interlin
inserisci l'istante finale di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2500
inserisci il nome della function contenente il segnale : rect2
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 5
```





OSS.: il primo grafico è stato ingrandito nella finestra temporale per poter visualizzare meglio il rettangolo centrato in un istante di tempo non nullo, in modo tale da evitare la presenza di effetti di bordo nel calcolo della DFT. Si può notare che il grafico in giallo, corrispondente ad un passo di campionamento molto più elevato di quello da noi fissato, si discosta dagli altri grafici, proprio a causa dell'elevata differenza del passo di campionamento .

Costruiamo adesso una spline mettendo in evidenza l'effetto del ricampionamento di un fattore non intero e dell'interpolazione su dati non sufficientemente sovracampionati come fatto per l'interpolatore lineare.

spline.m

```
% COSTRUZIONE DI UNA SPLINE

% Considereremo un segnale generico da introdurre dall'esterno

% può essere inserito anche un fattore non intero di ricampionamento
% e il periodo di campionamento può essere generico

clear all
tempo=input('inserisci l''istante finale di rappresentazione del segnale (in sec) : ');
f1=input('inserisci la frequenza di campionamento (in Hz) : ');
T1=1/f1;
err=0;
while err==0
   F=input('inserisci il nome della function contenente il segnale : ','s');
   if exist(F)==0
      disp('LA FUNZIONE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end
T=1/(100*f1);
k_=(tempo/T)+1;
k=(1:1:k_);
x=(k-1)*T;
y=feval(F,x);
x=(k-1)*T*1000;
figure
plot(x,y,'y');
xlabel('ms');
hold on

T=T1;
k_=(tempo/T)+1;
k=(1:1:k_);
x=(k-1)*T;
y=feval(F,x);
x=(k-1)*T*1000;
if length(x)<40
   plot(x,y,'r*');
else
   plot(x,y,'r');
end

disp('OPERAZIONE DI RICAMPIONAMENTO ');
valore=input('introduci il fattore di ricampionamento : ');
n=length(y);
valori=valore*(n-1)+1;
T1=x(n)-x(1);
T2=T1/(valori-1);
xi(1)=x(1);
for i=1:valori-1
   xi(i+1)=xi(i)+T2;
end
m=length(xi);
matr1(1,1)=2;
matr1(1,2)=1;
matr1(n,n-1)=1;
matr1(n,n)=2;
matr2(1,1)=-3;
matr2(1,2)=3;
matr2(n,n-1)=-3;
matr2(n,n)=3;
```

```
for i=2:n-1
   matr1(i,i-1)=1;
   matr1(i,i)=4;
   matr1(i,i+1)=1;
   matr2(i,i-1)=-3;
   matr2(i,i+1)=3;
end
matr=matr1\matr2;
yd=matr*y';
for j=1:m
   for i=1:n-1
      if xi(j)>=x(i) & xi(j)<=x(i+1)
         xin(j)=(xi(j)-x(i))*f1/1000;
         yi(j)=(-2*xin(j)^3+3*xin(j)^2)*y(i+1)+(2*xin(j)^3-3*xin(j)^2+1)*y(i)+...
         (xin(j)^3-2*xin(j)^2+xin(j))*yd(i)+(xin(j)^3-xin(j)^2)*yd(i+1);
      end
   end
end
xin(j)=(xi(j)-x(n-1))*f1/1000;
yi(j)=(-2*xin(j)^3+3*xin(j)^2)*y(n)+(2*xin(j)^3-3*xin(j)^2+1)*y(n-1)+...
   (xin(j)^3-2*xin(j)^2+xin(j))*yd(n-1)+(xin(j)^3-xin(j)^2)*yd(n);
if m<55
   plot(xi,yi,'b+');
else
   plot(xi,yi,'b');
end
hold off
zoom
ff=fft(yi);
freq=[0:(f1*valore)/m:f1*valore*(1-1/m)];
figure
plot(freq,abs(ff))
xlabel('Hz');
zoom
```

segnale.m

```
function f=segnale(x);
f=sin(2*pi*200*x)+sin(2*pi*350*x+pi/7)+sin(2*pi*420*x-pi/4);     % la frequenza è in Hz
```

impulso.m

```
function i=impulso(x)
n=length(x);
i(n)=0;
i(round(n/2))=1;
```

rect1.m

```
function r=rect1(x)
n=length(x);
r(n)=0;
for i=round(2*n/5) : round(3*n/5)
   r(i)=1;
end
```

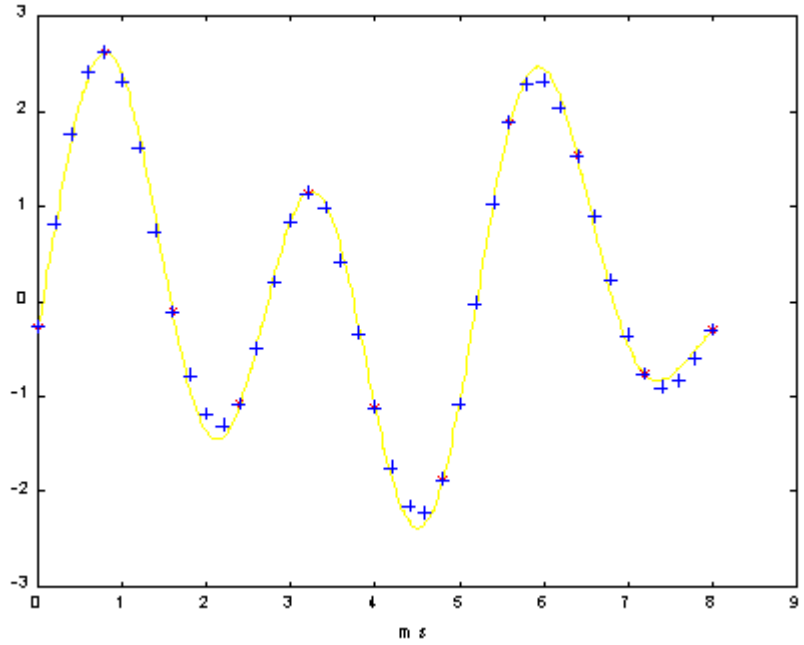
rect2.m

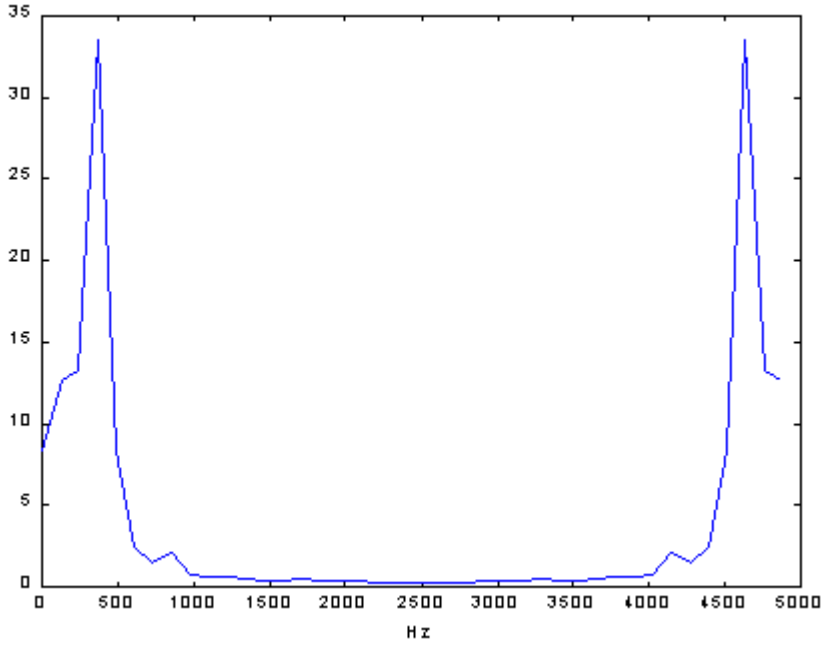
```
function r=rect2(x)
n=length(x);
r(n)=0;
for i=round(49*n/100) : round(51*n/100)
   r(i)=1;
end
```

esecuzione del programma

1) interpolazione di un segnale periodico

```
» spline
inserisci l'istante finale di rappresentazione del segnale (in sec) : 8e-3
inserisci la frequenza di campionamento (in Hz) : 1250
inserisci il nome della function contenente il segnale : segnale
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 4
```

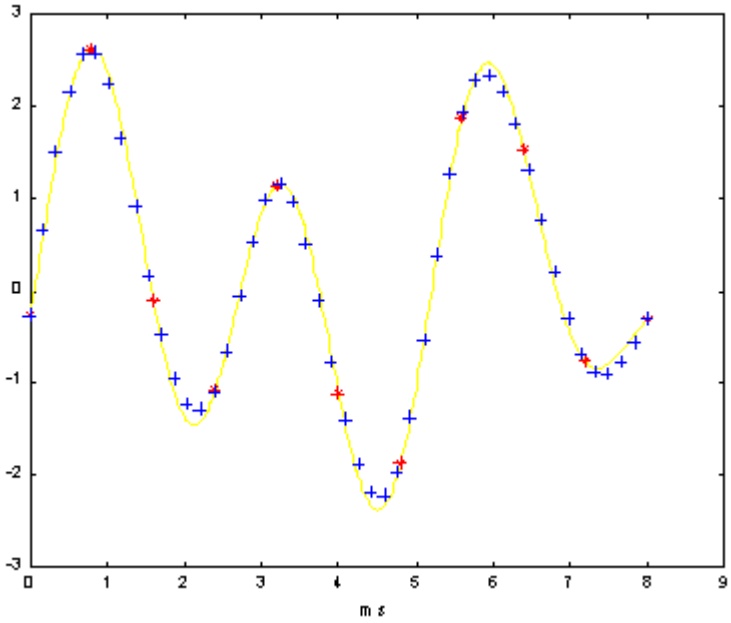




OSS.: la funzione rappresentata è la somma di tre sinusoidi, per cui la sua trasformata di Fourier dovrebbe avere 3 impulsi localizzati alle frequenze di 200 Hz, 350 Hz e 420 Hz : come si può notare da quest'ultimo grafico, invece è possibile individuare soltanto l'impulso a 420 Hz perchè il passo di campionamento e di sovracampionamento non sono molto alti per poter trascurare gli effetti di bordo che si hanno durante il calcolo della DFT.

Consideriamo adesso lo stesso segnale però ricampionato di un fattore non intero:

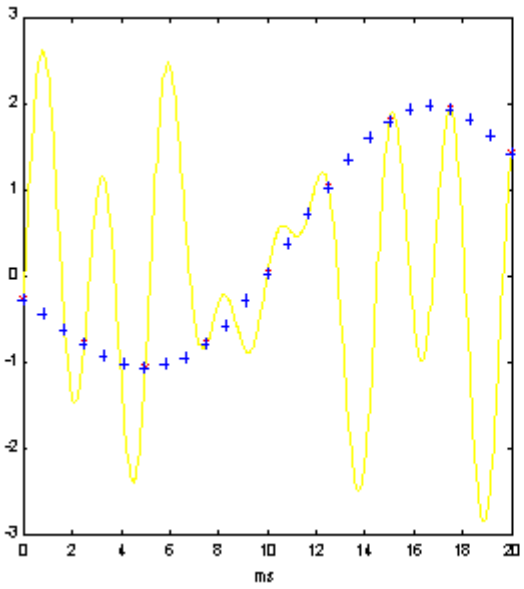
```
OPERAZIONE DI RICAMPIONAMENTO

introduci il fattore di ricampionamento : 4.7
```

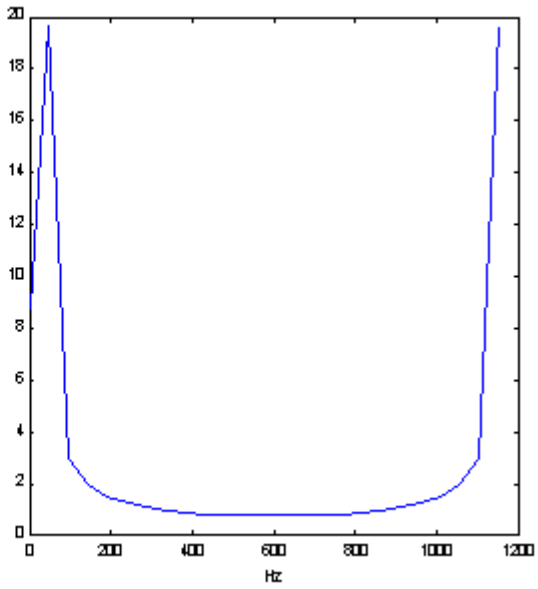


l'altro grafico e praticamente identico al caso precedente di ricampionamento intero

Adesso consideriamo l'effetto dell'interpolazione su dati non sufficientemente sovracampionati :

```
» spline

inserisci l'istante finale di rappresentazione del segnale (in sec) : 20e-3
inserisci la frequenza di campionamento (in Hz) : 400
inserisci il nome della function contenente il segnale : segnale
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 3
```

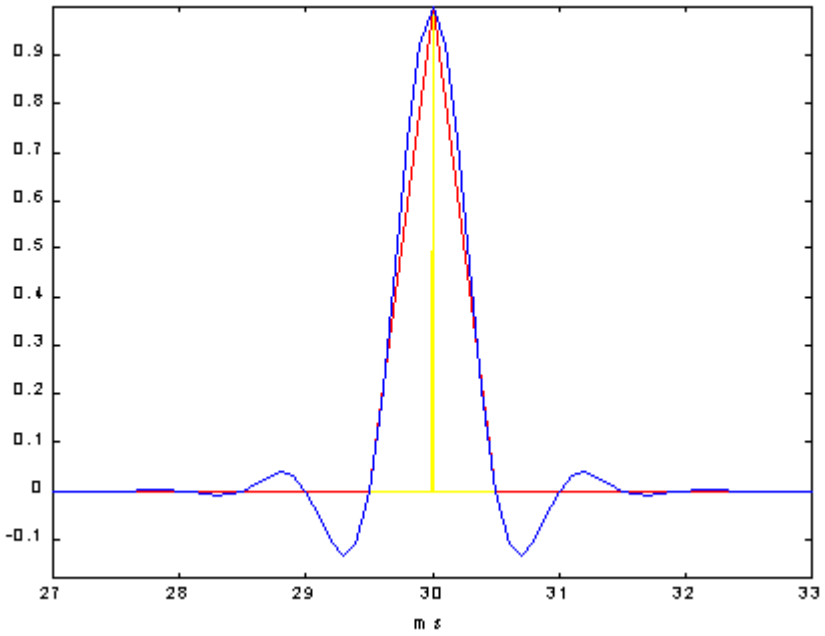


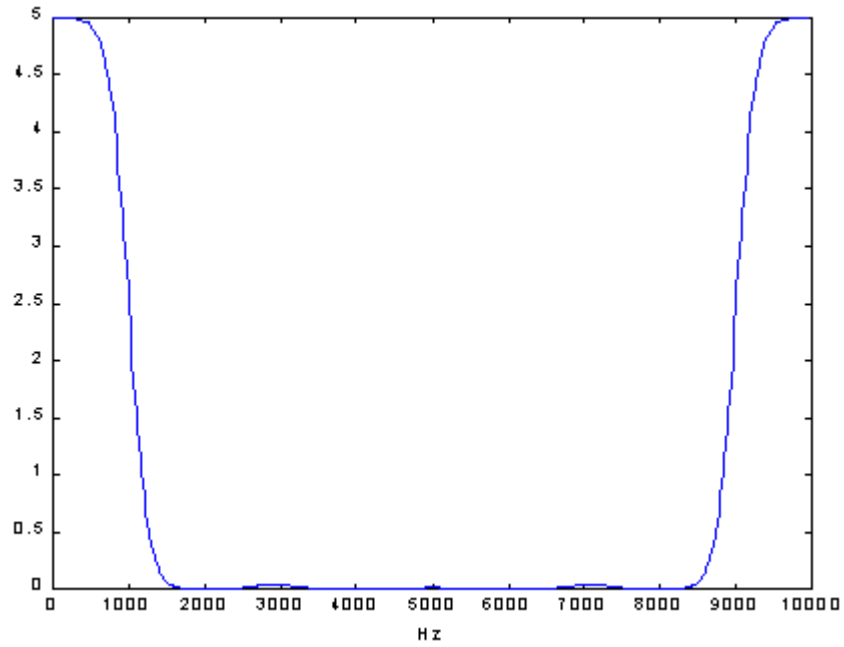


e osserviamo che non riusciamo assolutamente a ricostruire il segnale di partenza e lo spettro è anche completamente differente (non è più presente la riga alla frequenza di 420 Hz).

2) interpolazione di un segnale impulsivo

```
» spline

inserisci l'istante finale di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2000
inserisci il nome della function contenente il segnale : impulso

OPERAZIONE DI RICAMPIONAMENTO

introduci il fattore di ricampionamento : 5
```



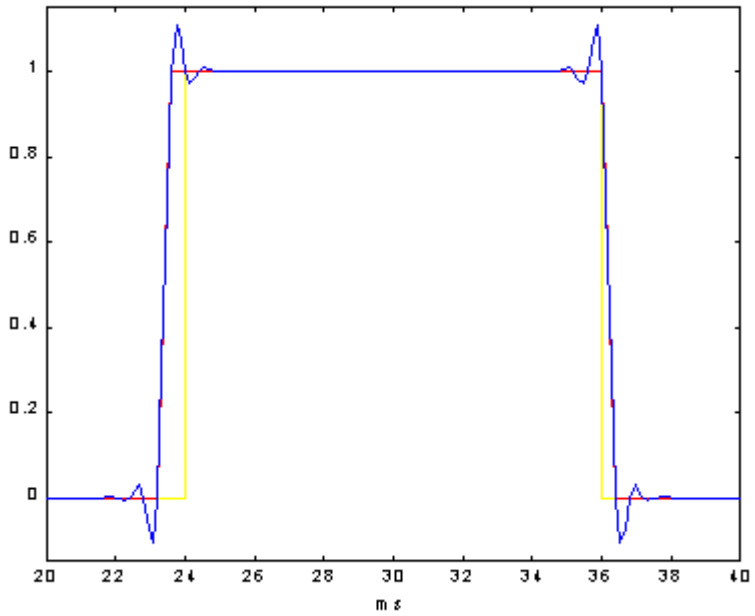


OSS.: il primo grafico è stato ingrandito nella finestra temporale per poter visualizzare meglio l'impulso centrato in un istante di tempo non nullo per evitare la presenza di effetti di bordo nel calcolo della DFT. E' stata disegnata anche l'interpolazione lineare degli stessi campioni.

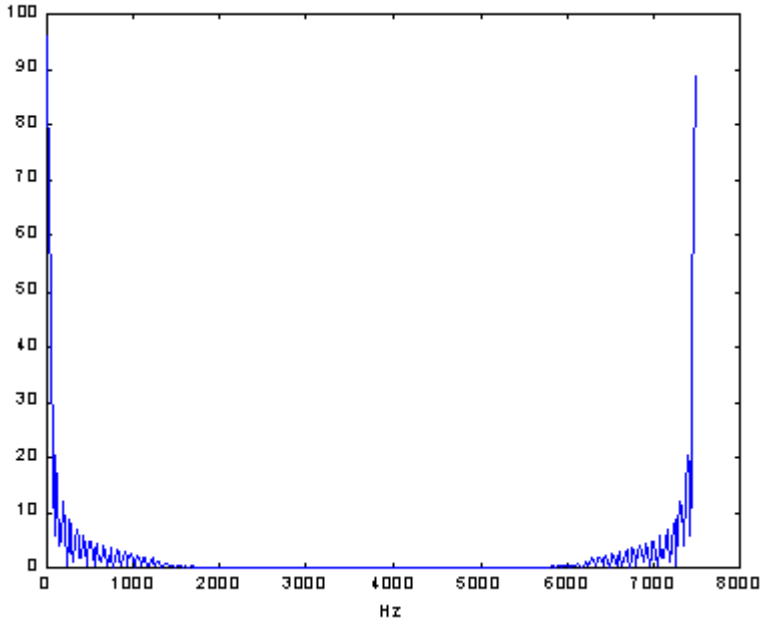
3) interpolazione di un segnale rettangolare

Consideriamo innanzitutto un segnale rettangolare di ampiezza temporale pari a 12,8 ms , considerando una finestra temporale di 60ms:

```
spline
inserisci l'istante finale di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2500
inserisci il nome della function contenente il segnale : rect1
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 3
```



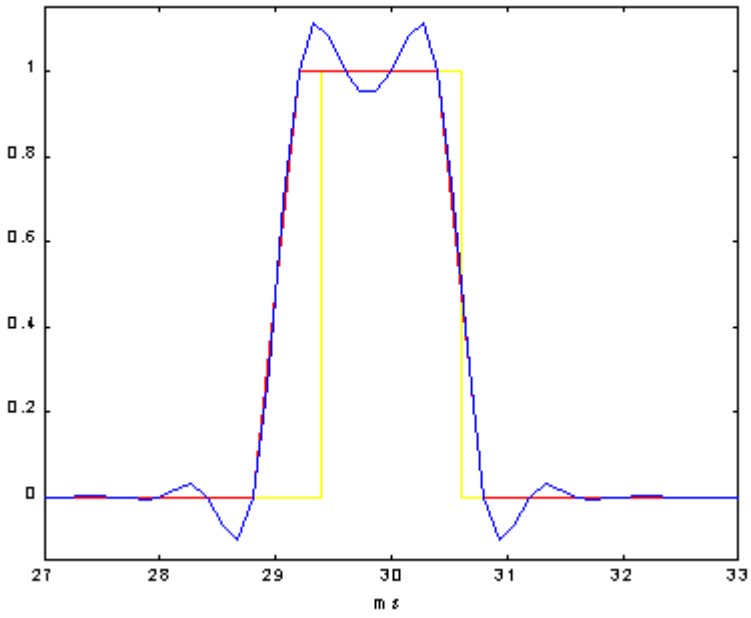
OSS.: anche ora il grafico è stato ingrandito nella finestra temporale per poter visualizzare meglio il rettangolo centrato in un istante di tempo non nullo per evitare la presenza di effetti di bordo nel calcolo della DFT. E' stata disegnata anche l'interpolazione lineare degli stessi campioni in rosso e si può notare che il grafico in giallo, corrispondente ad un passo di campionamento molto più elevato di quello da noi fissato, si discosta dagli altri grafici, proprio a causa dell'elevata differenza del passo di campionamento .

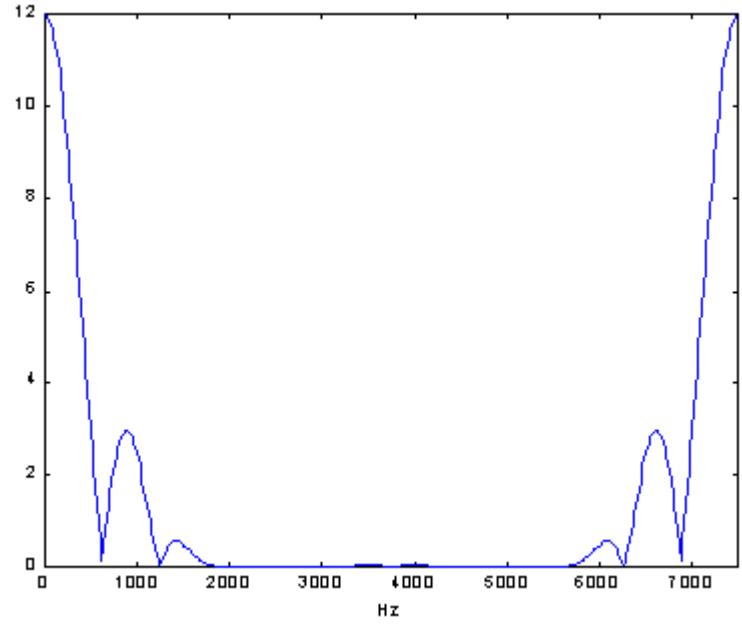


OSS.: il grafico della DFT non è molto comprensibile perché il rettangolo ha ampiezza temporale pari a τ =12,8 ms e sappiamo che il primo lobo termina a $f_0 = 1/\tau$ = 78 Hz ed un valore molto basso rispetto alla frequenza di campionamento per poter essere osservato.

Consideriamo adesso un rettangolo che ha ampiezza temporale molto più piccola e pari a τ =1,6 ms , per poter osservare i lobi secondari (infatti il lobo principale termina alla frequenza $f_0 = 1/\tau$ = 625 Hz:

```
» spline

inserisci l'istante finale di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2500
inserisci il nome della function contenente il segnale : rect2

OPERAZIONE DI RICAMPIONAMENTO

introduci il fattore di ricampionamento : 3
```





Costruiamo adesso una spline bicubica mettendo in evidenza l'effetto del ricampionamento di un fattore non intero e dell'interpolazione su dati non sufficientemente sovracampionati come fatto per l'interpolatore lineare e la spline.

bicubica.m

```
% COSTRUZIONE DI UNA SPLINE BICUBICA

% Considereremo un segnale generico da introdurre dall'esterno

% può essere inserito anche un fattore non intero di ricampionamento
% e il periodo di campionamento può essere generico

clear all
tempo=input('inserisci l''istante finale di rappresentazione del segnale (in sec) : ');
f1=input('inserisci la frequenza di campionamento (in Hz) : ');
T1=1/f1;
err=0;
while err==0
   F=input('inserisci il nome della function contenente il segnale : ','s');
   if exist(F)==0
      disp('LA FUNZIONE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end
T=1/(100*f1);
k_=(tempo/T)+1;
k=(1:1:k_);
x=(k-1)*T;
y=feval(F,x);
x=(k-1)*T*1000;
figure
plot(x,y,'y');
xlabel('ms');
hold on

T=T1;
k_=(tempo/T)+1;
k=(1:1:k_);
x=(k-1)*T;
y=feval(F,x);
x=(k-1)*T*1000;
if length(x)<40
   plot(x,y,'r*');
else
   plot(x,y,'r');
end

disp('OPERAZIONE DI RICAMPIONAMENTO ');
valore=input('introduci il fattore di ricampionamento : ');
a=input('inserire il valore di a : ');
n=length(y);
valori=valore*(n-1)+1;
T1=x(n)-x(1);
T2=T1/(valori-1);
xi(1)=x(1);
for i=1:valori-1
   xi(i+1)=xi(i)+T2;
end
m=length(xi);
for j=1:m
   yi(j)=0;
   for i=1:n-2
      if xi(j)>=x(i) & xi(j)<=x(i+1)
         xin=(xi(j)-x(i))*f1/1000;
         yi(j)=yi(j)+(1-(1+a)*xin^2+a*xin^3)*y(i);
      end
      if xi(j)>=x(i+1) & xi(j)<=x(i+2)
         xin=(xi(j)-x(i))*f1/1000;
```

```
            yi(j)=yi(j)+(a-2)*(xin^3-5*xin^2+8*xin-4)*y(i);
         end
      end
      for i=3:n
         if xi(j)>x(i-1) & xi(j)<x(i)
            xin=-(xi(j)-x(i))*f1/1000;
            yi(j)=yi(j)+(1-(1+a)*xin^2+a*xin^3)*y(i);
         end
         if xi(j)>x(i-2) & xi(j)<x(i-1)
            xin=-(xi(j)-x(i))*f1/1000;
            yi(j)=yi(j)+(a-2)*(xin^3-5*xin^2+8*xin-4)*y(i);
         end
      end
      if xi(j)>x(1) & xi(j)<x(2)
         xin=-(xi(j)-x(2))*f1/1000;
         yi(j)=yi(j)+(1-(1+a)*xin^2+a*xin^3)*y(2);
      end
      if xi(j)>x(n-1) & xi(j)<x(n)
         xin=(xi(j)-x(n-1))*f1/1000;
         yi(j)=yi(j)+(1-(1+a)*xin^2+a*xin^3)*y(n-1);
      end
   end
   yi(m)=y(n);
   if m<55
      plot(xi,yi,'b+');
   else
      plot(xi,yi,'b');
   end
   hold off
   zoom
   ff=fft(yi);
   freq=[0:(f1*valore)/m:f1*valore*(1-1/m)];
   figure
   plot(freq,abs(ff))
   xlabel('Hz');
   zoom
```

segnale.m

```
function f=segnale(x);
f=sin(2*pi*200*x)+sin(2*pi*350*x+pi/7)+sin(2*pi*420*x-pi/4);      % la frequenza è in Hz
```

impulso.m

```
function i=impulso(x)
n=length(x);
i(n)=0;
i(round(n/2))=1;
```

rect1.m

```
function r=rect1(x)
n=length(x);
r(n)=0;
for i=round(2*n/5) : round(3*n/5)
   r(i)=1;
end
```

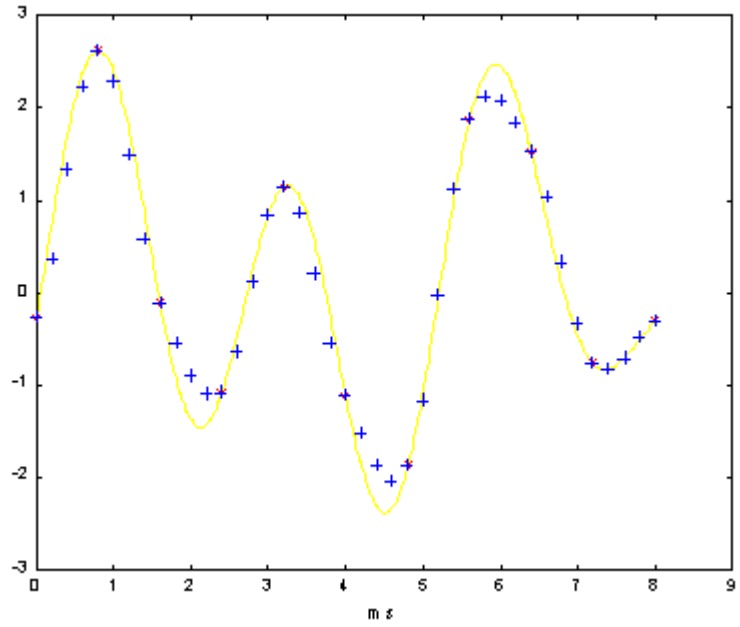
rect2.m

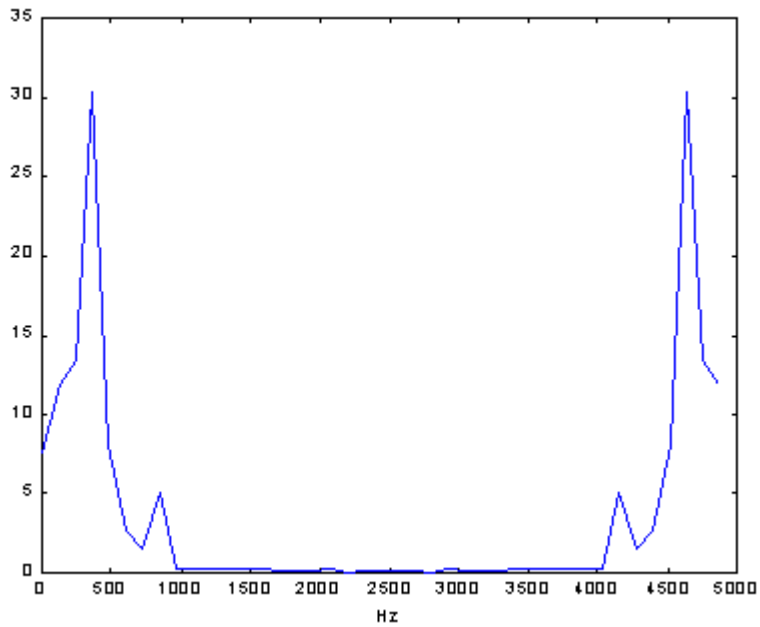
```
function r=rect2(x)
n=length(x);
r(n)=0;
for i=round(49*n/100) : round(51*n/100)
   r(i)=1;
end
```

esecuzione del programma

1) interpolazione di un segnale periodico

```
» bicubica

inserisci l'istante finale di rappresentazione del segnale (in sec) : 8e-3
inserisci la frequenza di campionamento (in Hz) : 1250
inserisci il nome della function contenente il segnale : segnale

OPERAZIONE DI RICAMPIONAMENTO

introduci il fattore di ricampionamento : 4
inserire il valore di a : 3/2
```

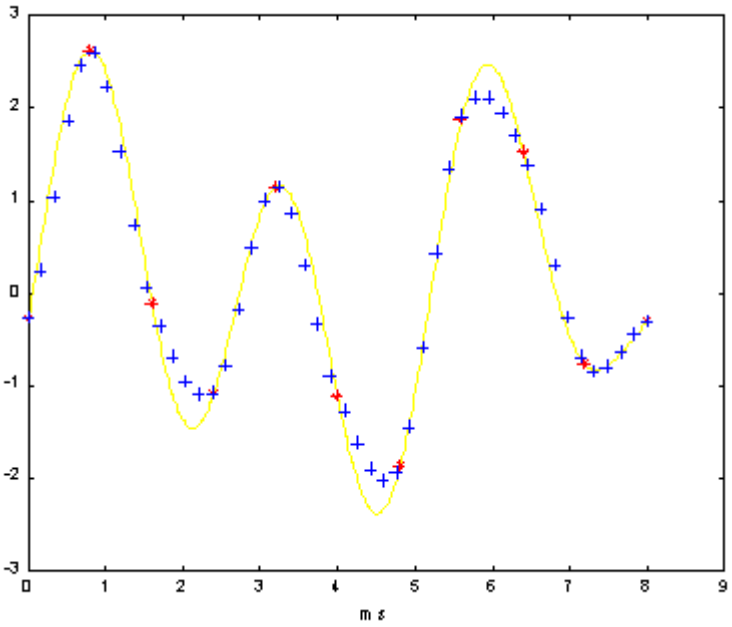




OSS.: la funzione rappresentata è la somma di tre sinusoidi, per cui la sua trasformata di Fourier dovrebbe avere 3 impulsi localizzati alle frequenze di 200 Hz, 350 Hz e 420 Hz : come si può notare da quest'ultimo grafico, invece e possibile individuare soltanto l'impulso a 420 Hz perchè il passo di campionamento e di sovracampionamento non sono molto alti per poter trascurare gli effetti di bordo che si hanno durante il calcolo della DFT.

Consideriamo adesso lo stesso segnale però ricampionato di un fattore non intero:

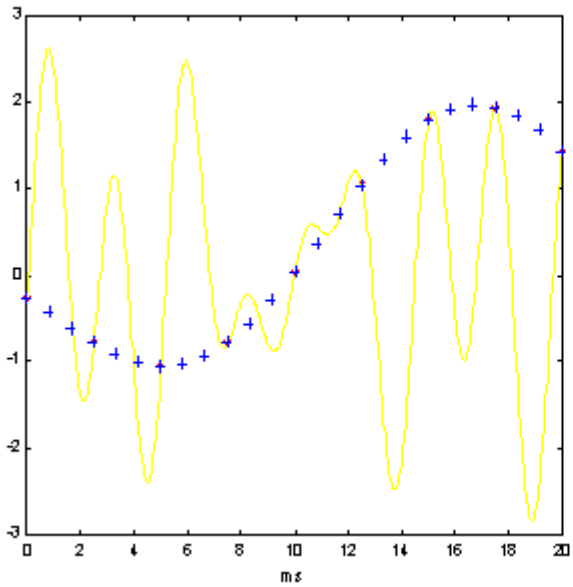
```
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 4.7
inserire il valore di a : 3/2
```

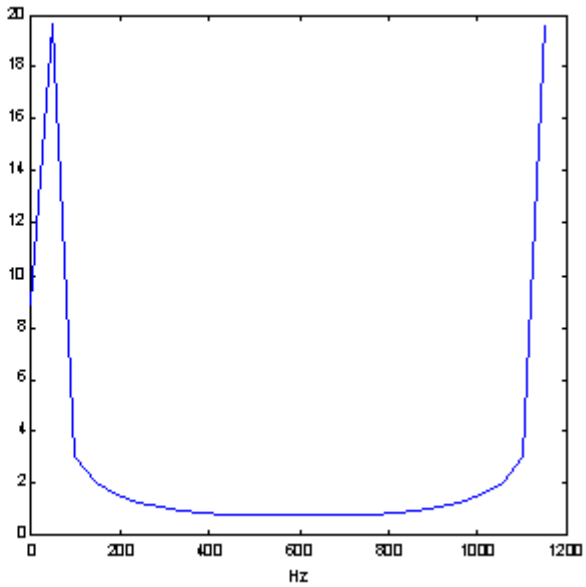


l'altro grafico e praticamente identico al caso precedente di ricampionamento intero

Adesso consideriamo l'effetto dell'interpolazione su dati non sufficientemente sovracampionati :

```
» bicubica
inserisci l'istante finale di rappresentazione del segnale (in sec) : 20e-3
inserisci la frequenza di campionamento (in Hz) : 400
inserisci il nome della function contenente il segnale : segnale

OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 3
inserire il valore di a : 3/2
```

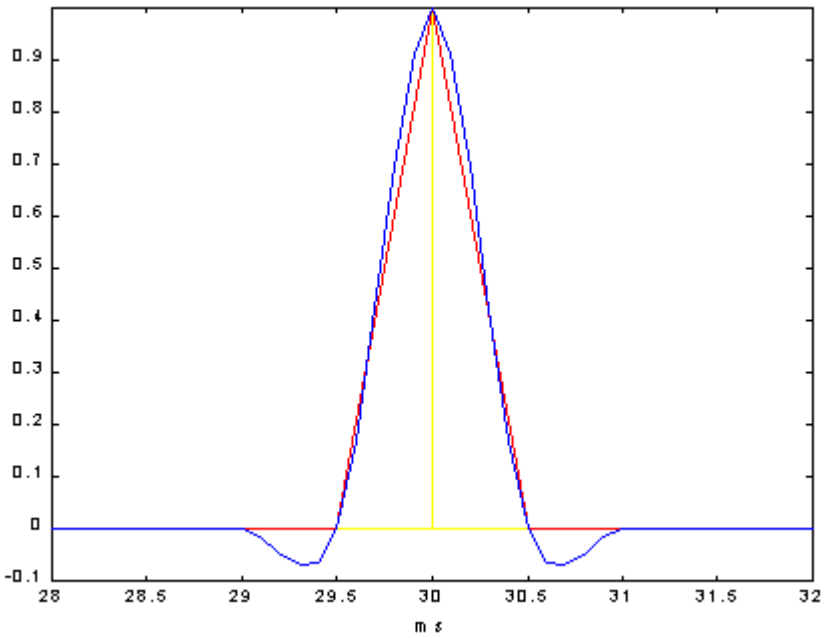


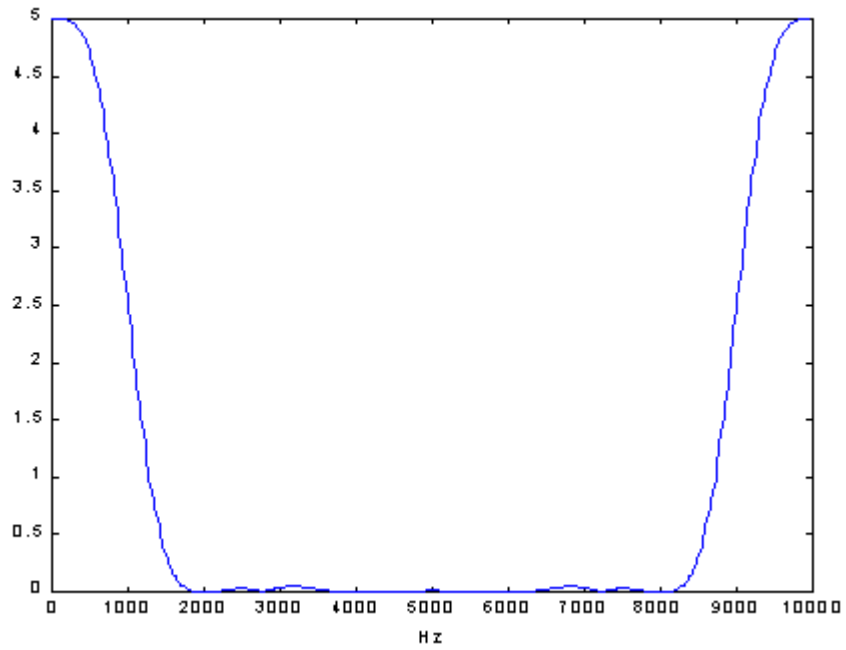


e osserviamo che non riusciamo assolutamente a ricostruire il segnale di partenza e lo spettro è anche completamente differente (non è più presente la riga alla frequenza di 420 Hz).

2) interpolazione di un segnale impulsivo

```
» bicubica

inserisci l'istante finale di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2000
inserisci il nome della function contenente il segnale : impulso

OPERAZIONE DI RICAMPIONAMENTO

introduci il fattore di ricampionamento : 5
inserire il valore di a : 3/2
```

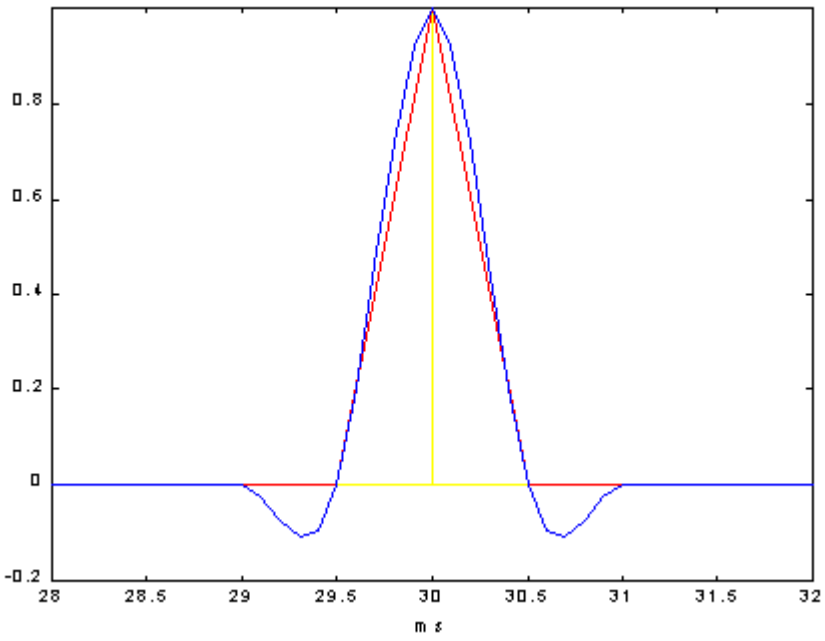


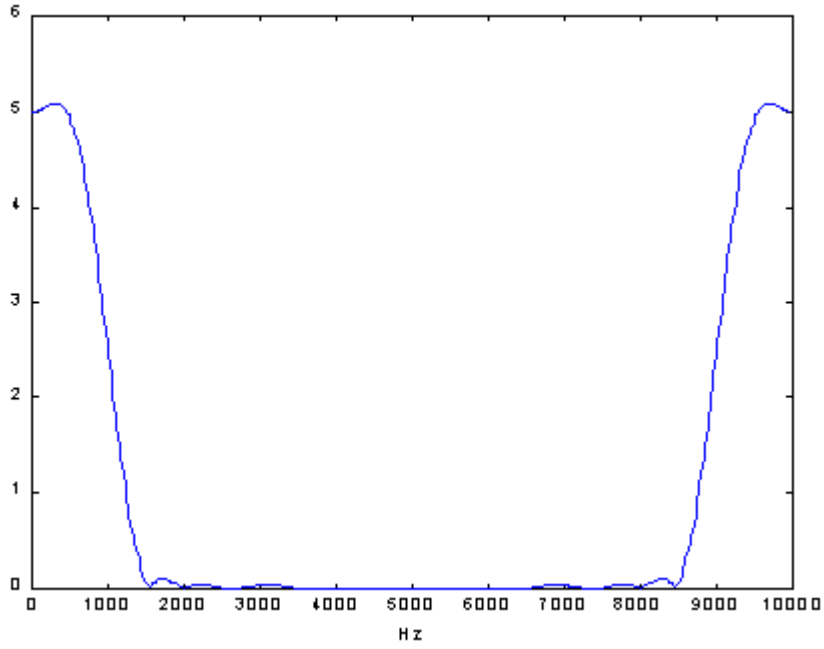


OSS.: il primo grafico è stato ingrandito nella finestra temporale per poter visualizzare meglio l'impulso centrato in un istante di tempo non nullo per evitare la presenza di effetti di bordo nel calcolo della DFT. E' stata disegnata anche l'interpolazione lineare degli stessi campioni.

Adesso eseguiamo l'interpolazione sempre dello stesso segnale impulsivo solo però che inseriamo un coefficiente 'a' pari a 5/4 per mettere in evidenza come la funzione di trasferimento della spline bicubica non è più piatta:

```
» bicubica

inserisci l'istante finale di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2000
inserisci il nome della function contenente il segnale : impulso

OPERAZIONE DI RICAMPIONAMENTO

introduci il fattore di ricampionamento : 5
inserire il valore di a : 5/4
```

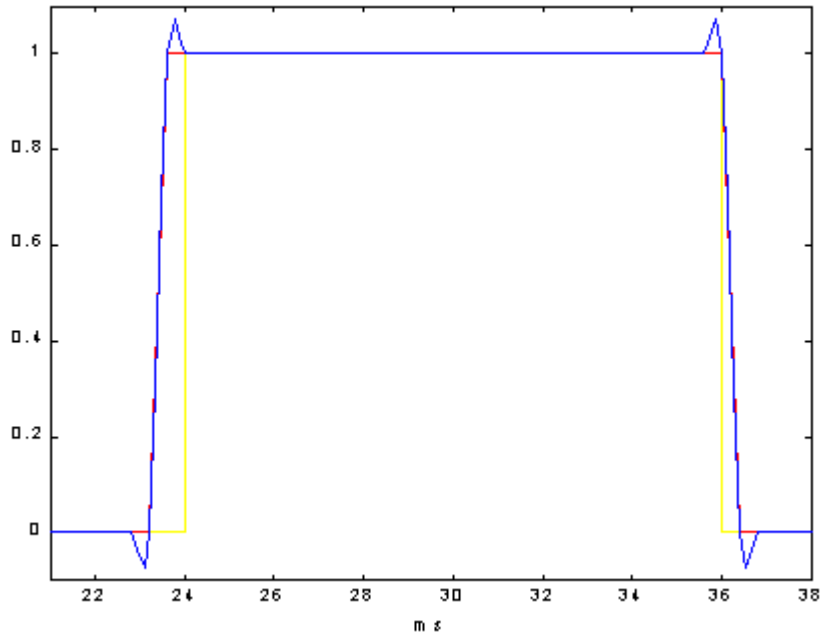




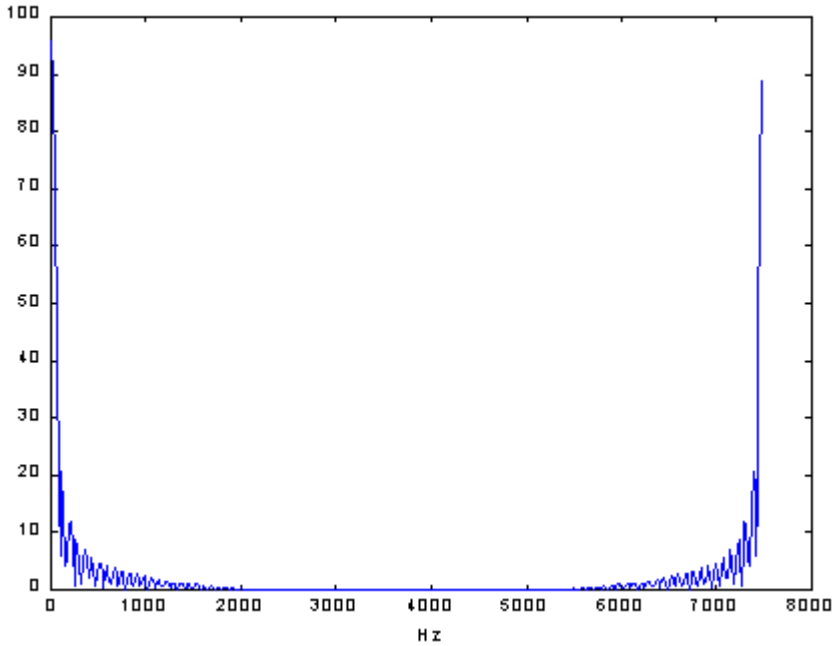
3) interpolazione di un segnale rettangolare

Consideriamo innanzitutto un segnale rettangolare di ampiezza temporale pari a 12,8 ms , considerando una finestra temporale di 60ms:

```
» bicubica
inserisci l'istante finale di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2500
inserisci il nome della function contenente il segnale : rect1
OPERAZIONE DI RICAMPIONAMENTO
introduci il fattore di ricampionamento : 3
inserire il valore di a : 3/2
```



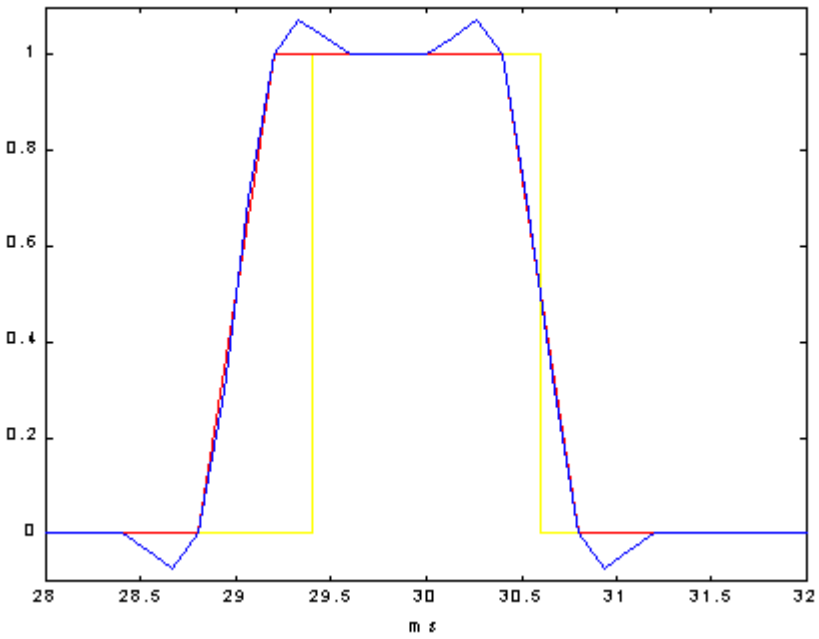
OSS.: anche ora il grafico è stato ingrandito nella finestra temporale per poter visualizzare meglio il rettangolo centrato in un istante di tempo non nullo per evitare la presenza di effetti di bordo nel calcolo della DFT. E' stata disegnata anche l'interpolazione lineare degli stessi campioni in rosso e si può notare che il grafico in giallo, corrispondente ad un passo di campionamento molto più elevato di quello da noi fissato, si discosta dagli altri grafici, proprio a causa dell'elevata differenza del passo di campionamento .

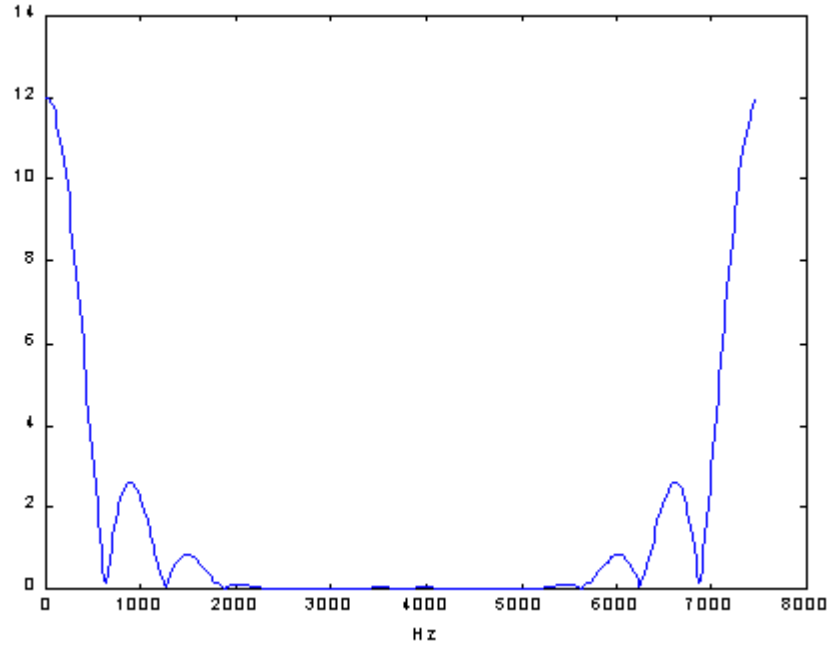


OSS.: il grafico della DFT non è molto comprensibile perché il rettangolo ha ampiezza temporale pari a τ =12,8 ms e sappiamo che il primo lobo termina a $f_0 = 1/\tau = 78$ Hz ed un valore molto basso rispetto alla frequenza di campionamento per poter essere osservato.

Consideriamo adesso un rettangolo che ha ampiezza temporale molto più piccola e pari a τ =1,6 ms , per poter osservare i lobi secondari (infatti il lobo principale termina alla frequenza $f_0 = 1/\tau = 625$ Hz:

```
» bicubica

inserisci l'istante finale di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2500
inserisci il nome della function contenente il segnale : rect2

OPERAZIONE DI RICAMPIONAMENTO

introduci il fattore di ricampionamento : 3
inserire il valore di a : 3/2
```





A questo punto possiamo costruire un interpolatore utilizzando però la DFT, mettendo sempre in evidenza l'effetto del ricampionamento di un fattore non intero e dell'interpolazione su dati non sufficientemente sovracampionati come fatto per gli altri interpolatori.

interdft.m

```
% COSTRUZIONE DI UN CAMPIONATORE CON L'USO DELLA DFT
% Considereremo un segnale generico da introdurre dall'esterno

% può essere inserito anche un fattore non intero di ricampionamento comunque sempre
% maggiore di 1 e il periodo di campionamento può essere generico

clear all
tempo=input('inserisci l''istante finale MINIMO di rappresentazione del segnale (in sec) : ');
f1=input('inserisci la frequenza di campionamento (in Hz) : ');
T1=1/f1;
err=0;
while err==0
   F=input('inserisci il nome della function contenente il segnale : ','s');
   if exist(F)==0
      disp('LA FUNZIONE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end
valore=input('introduci il fattore di ricampionamento : ');
[N,D] = rat(valore);
n=(tempo/T1)+1;
l=fix(n/D);
if n~=l*D
   n=l*D;
   disp('l''effettiva finestra temporale in secondi è : ');
   tempo=(n-1)*T1
end
Nz=(N-D)*l;
T=1/(100*f1);
n=(tempo/T)+1;
k=(1:1:n);
x=(k-1)*T;
y=feval(F,x);
x=(k-1)*T*1000;
figure
plot(x,y,'y');
xlabel('ms');
hold on

T=T1;
n=(tempo/T)+1;
k=(1:1:n);
x=(k-1)*T;
y=feval(F,x);
x=x*1000;
if length(x)<40
   plot(x,y,'r*');
else
   plot(x,y,'r');
end
hold off
figura=gcf;

ff=fft(y);
figure
freq=[0:f1/n:f1*(1-1/n)];
plot(freq,abs(ff))
xlabel('f [ Hz ]')
title('FFT del segnale di partenza')
if n<55
   hold on
```

```
   plot(freq,abs(ff),'r+');
   hold off
end
zoom
if (fix(n/2))*2~=n
   ffi=[ff(1:1:(n+1)/2),zeros(1,Nz),ff((n+3)/2:1:n)];
else
   ffi=[ff(1:1:n/2),zeros(1,Nz),ff((n/2)+1:1:n)];
end
freq=[0:(f1*valore)/(n+Nz):f1*valore*(1-1/(n+Nz))];
figure
plot(freq,abs(ffi));
if n*valore<55
   hold on
   plot(freq,abs(ffi),'r+');
   hold off
end
xlabel('f [ Hz ]')
title('FFT del segnale in cui sono stati aggiunti gli zeri')
zoom
yi=ifft(ffi);
yi=valore*yi;
m=length(yi);
xi=[x(1):(x(n)-x(1))/(n+Nz-valore):x(n)*(1+(valore-1)/(n+Nz-valore))];
figure(figura)
hold on
if m<65
   plot(xi,real(yi),'b+');
else
   plot(xi,real(yi),'b');
end
hold off
zoom
```

segnale.m

```
function f=segnale(x);
f=sin(2*pi*200*x)+sin(2*pi*350*x+pi/7)+sin(2*pi*420*x-pi/4);     % la frequenza è in Hz
```

impulso.m

```
function i=impulso(x)
n=length(x);
i(n)=0;
i(round(n/2))=1;
```

rect1.m

```
function r=rect1(x)
n=length(x);
r(n)=0;
for i=round(2*n/5) : round(3*n/5)
   r(i)=1;
end
```

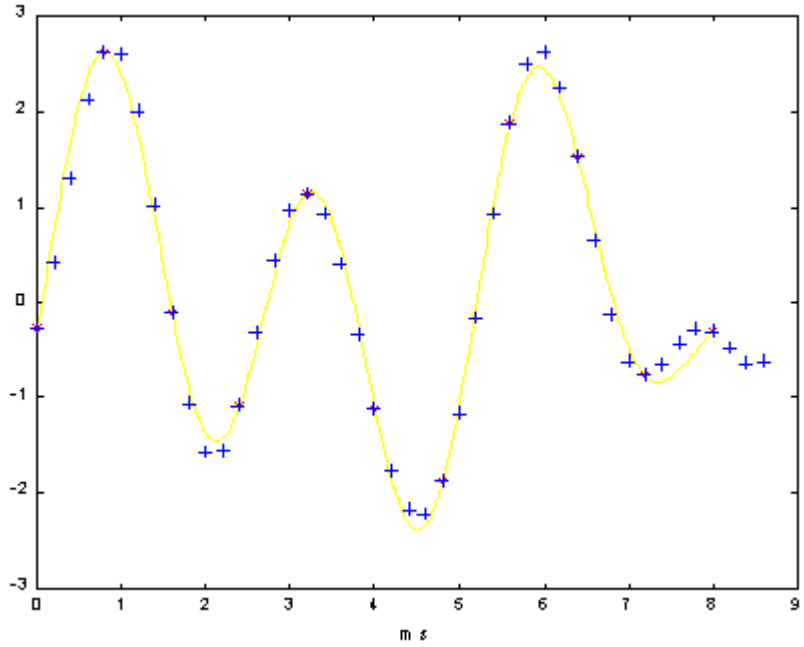
rect2.m

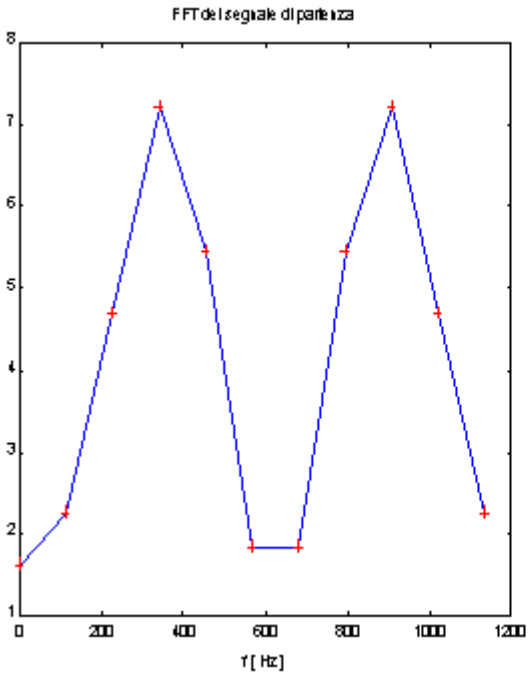
```
function r=rect2(x)
n=length(x);
r(n)=0;
for i=round(49*n/100) : round(51*n/100)
   r(i)=1;
end
```

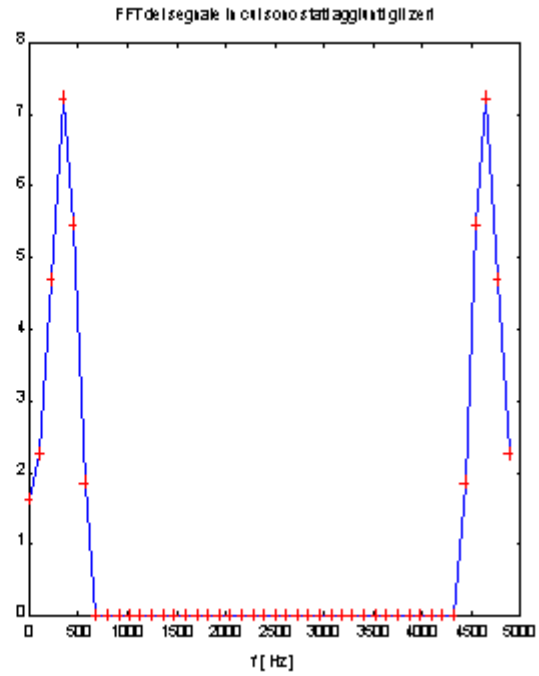
esecuzione del programma

1) interpolazione di un segnale periodico

```
» interdft

inserisci l'istante finale MINIMO di rappresentazione del segnale (in sec) : 8e-3
inserisci la frequenza di campionamento (in Hz) : 1250
inserisci il nome della function contenente il segnale : segnale
introduci il fattore di ricampionamento : 4
```

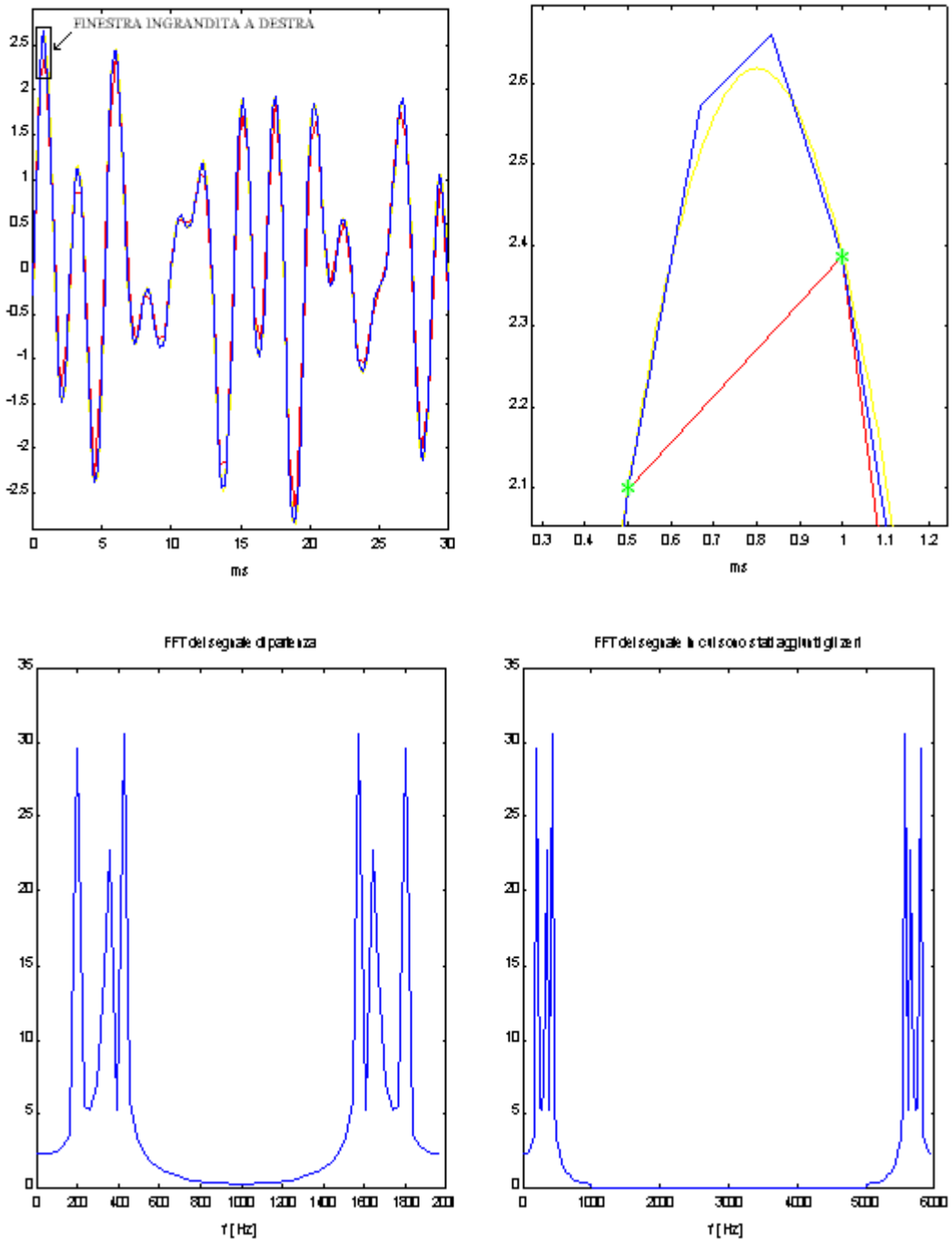






OSS.: la funzione rappresentata è la somma di tre sinusoidi, per cui la sua trasformata di Fourier dovrebbe avere 3 impulsi localizzati alle frequenze di 200 Hz, 350 Hz e 420 Hz : come si può notare da questi ultimi grafici, invece non è possibile individuare nessun l'impulso perchè il passo di campionamento non è molto alto per poter trascurare gli effetti di bordo che si hanno durante il calcolo della DFT e i punti considerati sono troppo pochi. Se si esegue il file considerando una finestra temporale che comprende almeno un periodo del segnale e si considera un passo di campionamento più alto allora nello spettro si individuano i tre impulsi:

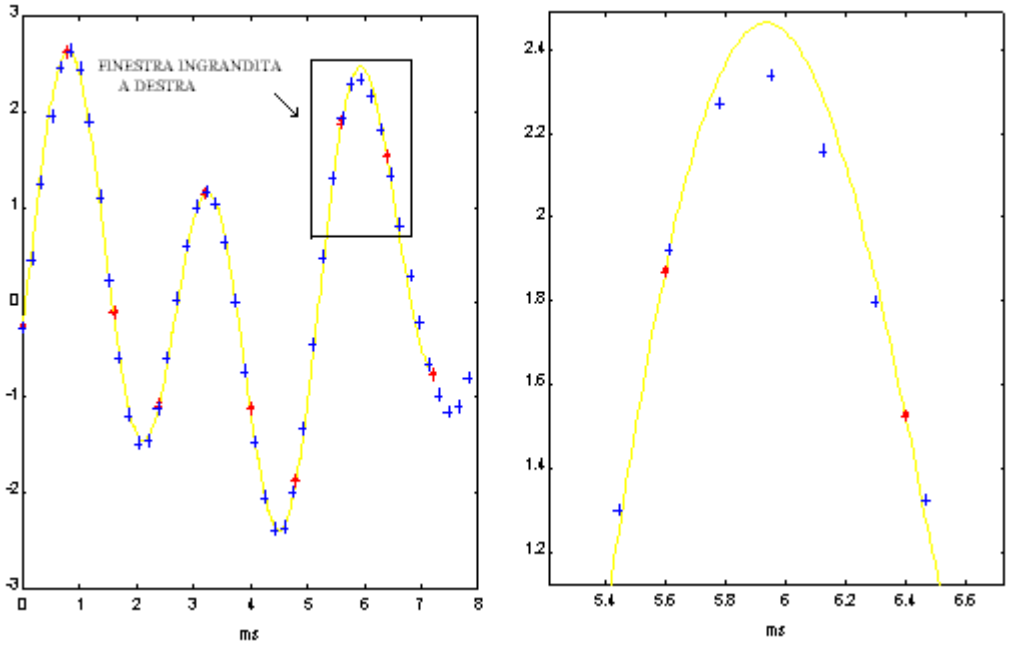
```
» interdft

inserisci l'istante finale MINIMO di rappresentazione del segnale (in sec) : 30e-3

inserisci la frequenza di campionamento (in Hz) : 2000

inserisci il nome della function contenente il segnale : segnale

introduci il fattore di ricampionamento : 3
```



OSS.: la prima figura è stata ingrandita a destra per poter osservare come questo tipo di interpolatore non è assolutamente buono. Infatti quando calcola la IDFT di un segnale che è stato ampliato in frequenza, non facciamo altro che calcolare la nel dominio del tempo la convoluzione CIRCOLARE del segnale di partenza con un sinc sovracampionato che altera il valore dei campioni che andiamo ad interpolare.

Consideriamo adesso lo stesso segnale però ricampionato di un fattore non intero:

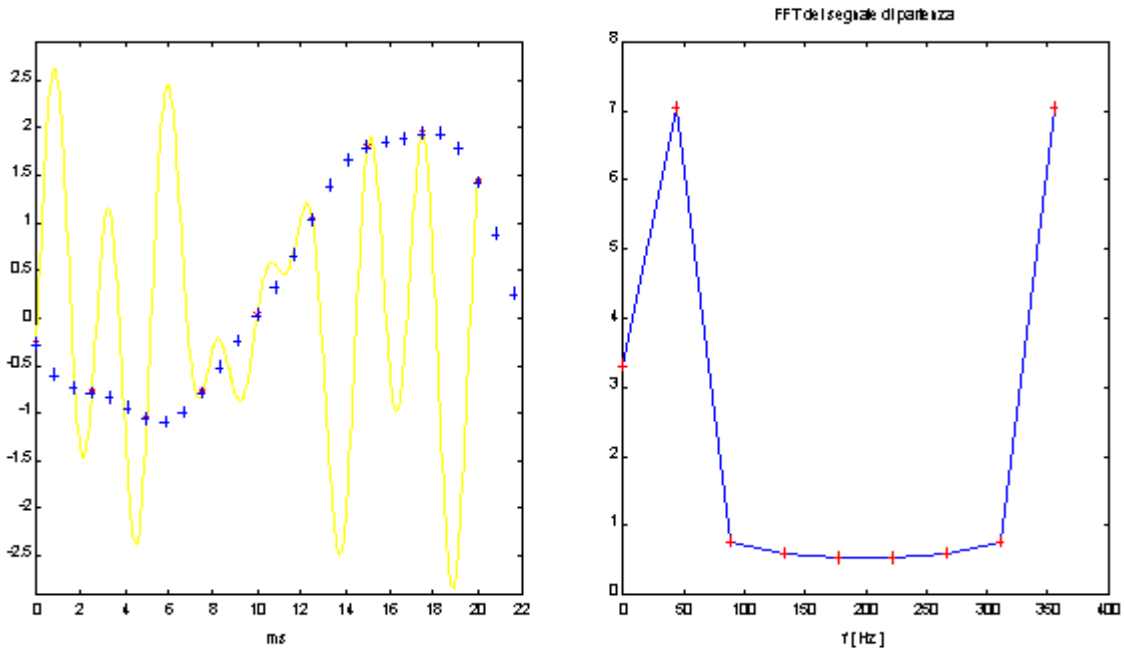
```
» interdft
inserisci l'istante finale MINIMO di rappresentazione del segnale (in sec) : 8e-3
inserisci la frequenza di campionamento (in Hz) : 1250
inserisci il nome della function contenente il segnale : segnale
introduci il fattore di ricampionamento : 4.7
l'effettiva finestra temporale in secondi è : tempo =    0.0072
```



gli altri grafici sono quasi identici al primo caso con ricampionamento intero

Adesso consideriamo l'effetto dell'interpolazione su dati non sufficientemente sovracampionati :

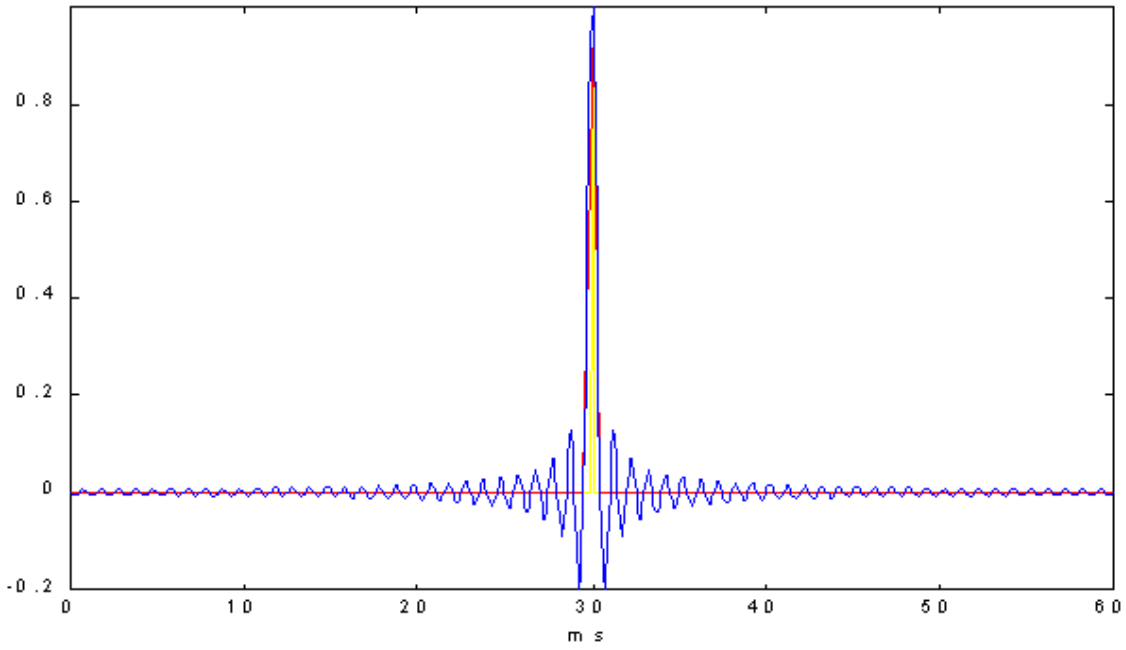
```
» interdft
inserisci l'istante finale MINIMO di rappresentazione del segnale (in sec) : 20e-3
inserisci la frequenza di campionamento (in Hz) : 400
inserisci il nome della function contenente il segnale : segnale
introduci il fattore di ricampionamento : 3
```

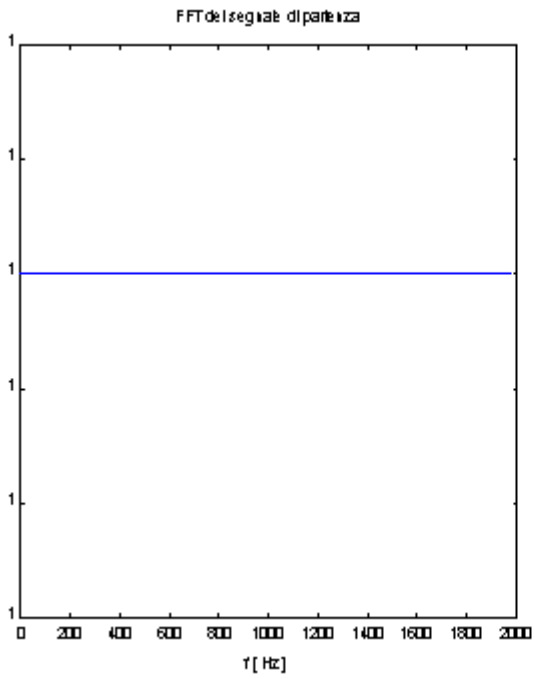


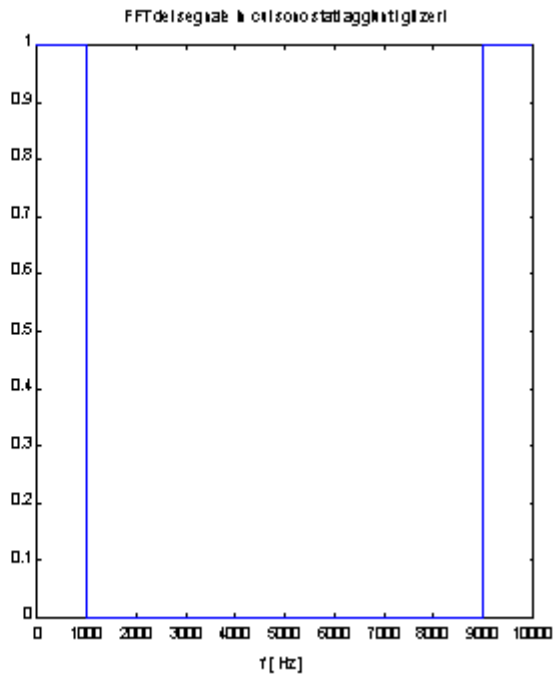
e osserviamo come non riusciamo assolutamente a ricostruire il segnale di partenza e lo spettro è anche completamente differente (non è più presente la riga alla frequenza di 420 Hz). Notare anche come i dati interpolati non seguono con continuità i campioni di partenza.

2) interpolazione di un segnale impulsivo

```
» interdft
inserisci l'istante finale MINIMO di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2000
inserisci il nome della function contenente il segnale : impulso
introduci il fattore di ricampionamento : 5
```



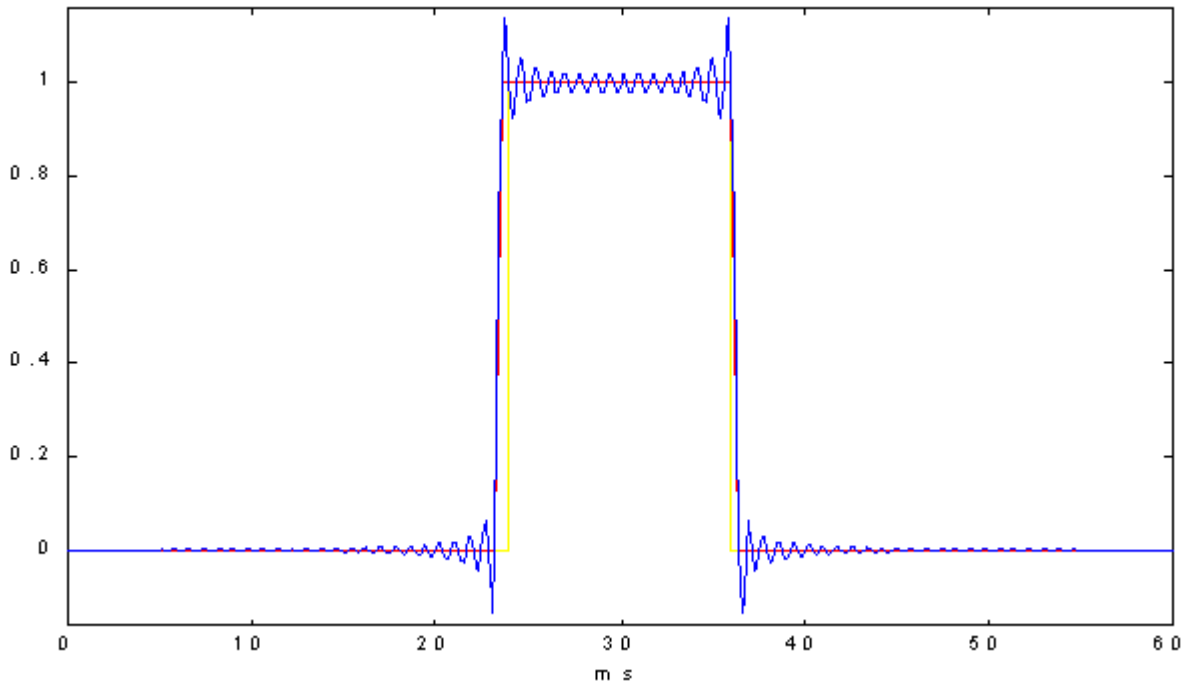




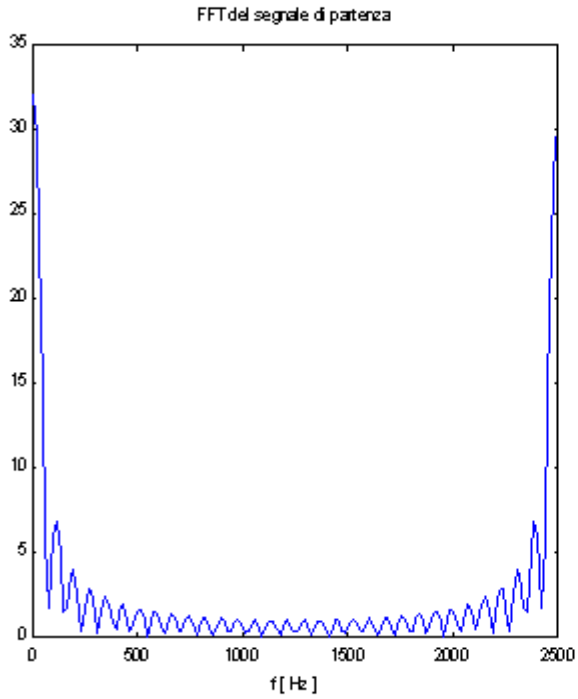
3) interpolazione di un segnale rettangolare

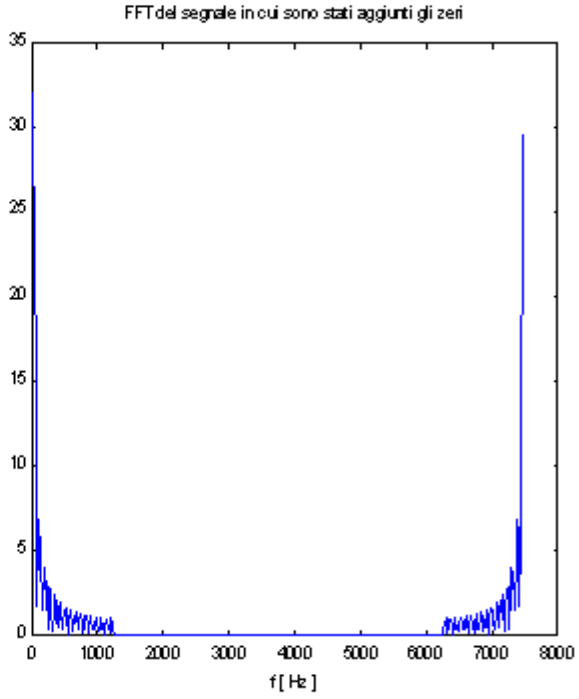
Consideriamo innanzitutto un segnale rettangolare di ampiezza temporale pari a 12,8 ms , considerando una finestra temporale di 60ms:

```
» interdft

inserisci l'istante finale MINIMO di rappresentazione del segnale (in sec) : 60e-3
inserisci la frequenza di campionamento (in Hz) : 2500
inserisci il nome della function contenente il segnale : rect1
introduci il fattore di ricampionamento : 3
```



OSS.: è stata disegnata anche l'interpolazione lineare degli stessi campioni in rosso e si può notare che il grafico in giallo, corrispondente ad un passo di campionamento molto più elevato di quello da noi fissato, si discosta dagli altri grafici, proprio a causa dell'elevata differenza del passo di campionamento .





Realizziamo adesso un programma con il quale è possibile calcolare la convoluzione sia lineare che circolare tra due generici segnali introdotti dall'esterno. E' possibile anche calcolare la convoluzione utilizzando i metodi : overlap & save e overlap & add.

conv.m

```
%                       LA CONVOLUZIONE

% Considereremo un segnale generico da introdurre dall'esterno

% così come la risposta impulsiva del filtro

clear all

err=0;
while err==0
   F=input('inserisci il nome della function contenente il segnale:','s');
   if exist(F)==0
      disp('LA FUNZIONE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end
err=0;
while err==0
   R=input('inserisci il nome della function contenente la risposta impulsiva del filtro:','s');
   if exist(R)==0
      disp('LA FUNZIONE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end
disp('Scegli il problema da risolvere : ');
err=0;
while err==0
   disp('1. Convoluzione lineare');
   disp('2. Convoluzione circolare');
   disp('3. Convoluzione circolare con l''uso della FFT');
   disp('4. Convoluzione utilizzando il metodo overlap & save');
   disp('5. Convoluzione utilizzando il metodo overlap & add');
   risp=input('Introduci il numero corrispondente : ');
   if risp~=1 & risp~=2 & risp~=3 & risp~=4 & risp~=5
      disp('Il valore introdotto é errato.Ricomincia');
   else
      err=1;
   end
end
f=feval(F);
r=feval(R);
N=length(f);
M=length(r);
if M<N
   c1=r;
   m1=f;
else
   c1=f;
   m1=r;
   pass=M;
   M=N;
   N=pass;
end
if risp==1

   % CONVOLUZIONE LINEARE

   c2=[zeros(1,N-1),c1,zeros(1,N-1)];
   for i=1:length(c2)
      c3(i)=c2(length(c2)-i+1);
   end
   for i=1:N+M-1
```

```
      y1(i)=sum(c3([N+M-i:2*N+M-i-1]).*m1);
   end
   figure
   plot(y1);
   title('convoluzione lineare');
   zoom
end

if risp==2

   %CONVOLUZIONE CIRCOLARE

   c2=[c1,zeros(1,N-M)];
   for i=1:length(c2)
      c3(i)=c2(length(c2)-i+1);
   end
   for i=1:N
      y2(i)=0;
      for j=1:N
         k=i-j+1;
         if k<=0
            k=k+N;
         end
         y2(i)=y2(i)+c3(N+1-j)*m1(k);
      end
    end
    figure
    plot(y2);
    title('convoluzione circolare');
    zoom
end

if risp==3

   %CONVOLUZIONE CIRCOLARE CON DFT

   c2=[c1,zeros(1,N-M)];
   ff1=fft(c2);
   ff2=fft(m1);
   ff=ff1.*ff2;
   y3=ifft(ff);
   figure
   plot(real(y3));
   title('convoluzione circolare con DFT');
   zoom
end

if risp==4

   % CONVOLUZIONE UTILIZZANDO IL METODO OVERLAP & SAVE

   a=N/M;
   if a>1
      m=ceil(2*(M-1)*log10(N/M));
      N1=M-1+m;
      cicli=fix(((N-N1)/m)+1);     % manca il calcolo per l'ultimo pezzo del vettore
   else
      m=0;
   end
   if m==0
      disp('Il metodo overlap & save non si può effettuare per questi segnali.');
      disp('Usare la convoluzione normale');
   else
      c2=[c1,zeros(1,N1-M)];
      C2=fft(c2);
      for i=1:cicli
         n=(i-1)*m+1;
         m0=m1(n:n+N1-1);
         M0=fft(m0);
         Y4i=M0.*C2;
```

```
         y4i=ifft(Y4i);
         if n==1
            y4(n:n+N1-1)=y4i(1:length(y4i));
         else
            n=n+M-1;
            y4(n:n+N1-M)=y4i(M:length(y4i));
         end
      end
      disp('Il vettore di dimensione maggiore è stato suddiviso in : ');
      if length(y4)==N
         cicli
      else
         cicli=cicli+1
         N1=N-length(y4)+M-1;
         m0=m1(N-N1+1:N);
         M0=fft(m0);
         c2=[c1,zeros(1,N1-M)];
         C2=fft(c2);
         Y4i=M0.*C2;
         y4i=ifft(Y4i);
         y4(N-N1+M:N)=y4i(M:length(y4i));
      end
      figure
      plot(real(y4));
      title('convoluzione con il metodo overlap & save');
      zoom
   end
end

if risp==5

   % CONVOLUZIONE UTILIZZANDO IL METODO OVERLAP & ADD

   a=N/(M-1);
   if a>=2
      m=fix(2*(M-1)*(log10(N/(M-1))-log10(2)));
      N1=M-1+m;
      cicli=fix(N/N1);      % manca il calcolo per l'ultimo pezzo del vettore
   else
      m=-1;
   end
   if m==-1
      disp('Il metodo overlap & add non si può effettuare per questi segnali.');
      disp('Usare la convoluzione normale');
   else
      c2=[c1,zeros(1,N1-1)];
      C2=fft(c2);
      y5j=zeros(1,M-1);
      for i=1:cicli
         n=(i-1)*N1+1;
         m0=[zeros(1,M-1),m1(n:n+N1-1)];
         M0=fft(m0);
         Y5i=M0.*C2;
         y5i=ifft(Y5i);
         y5(n:n+N1-1)=y5i(M:length(y5i));
         y5(length(y5)-N1+1:length(y5)-N1+M-1)=y5(length(y5)-N1+1:length(y5)-N1+M-1)+y5j;
         y5j=y5i(1:M-1);
      end
      disp('Il vettore di dimensione maggiore è stato suddiviso in : ');
      if length(y5)==N
         cicli
      else
         cicli=cicli+1
         N1=N-length(y5);
         m0=[zeros(1,M-1),m1(N-N1+1:N)];
         M0=fft(m0);
         c2=[c1,zeros(1,N1-1)];
         C2=fft(c2);
         Y5i=M0.*C2;
```

```
            y5i=ifft(Y5i);
            y5(N-N1+1:N)=y5i(M:length(y5i));
            if M-1>N1
               y5(N-N1+1:N)=y5(N-N1+1:N)+y5j(1:N1);
            else
               y5(N-N1+1:N-N1+M-1)=y5(N-N1+1:N-N1+M-1)+y5j;
            end
         end
         figure
         plot(real(y5));
         title('convoluzione con il metodo overlap & add');
         zoom
      end
   end
```

segnale.m

```
function vet=segnale;

tempo=50e-3;
T=1/10000;
f1=200;
f2=420;
campioni=(tempo/T)+1;
k=(1:1:campioni);
xf=(k-1)*T;
yf=sin(2*pi*f1*xf)+sin(2*pi*f2*xf);
vet=yf;
figure
plot(xf,yf);
xlabel('sec');
title('grafico del segnale nel tempo');
ff=fft(yf);
zoom
figure
freq=[0:1/(T*campioni):(1/T)*(1-1/campioni)];
plot(freq,abs(ff));
xlabel('Hz');
title('spettro del segnale');
axis([0 500,min(abs(ff)) max(abs(ff))]);
zoom
```

banda.m

```
function r=banda;

f1=400;          %frequenza di taglio inferiore
f2=440;          %frequenza di taglio superiore
tempo=10e-3;
T=1/10000;
campioni=(tempo/T)+1;
k=(1:campioni);
x=(k-(length(k)+1)/2)*T;
for i=1:length(x)
   if i~=(length(k)+1)/2
      y(i)=(sin(2*pi*f2*x(i))-sin(2*pi*f1*x(i)))./(pi*x(i));
   end
end
y((length(k)+1)/2)=2*f2-2*f1;
r=y;
figure
plot(x*1000,y);
xlabel('ms');
title('grafico del segnale nel tempo');
zoom
ff=fft(r);
figure
freq=[0:1/(T*campioni):(1/T)*(1-1/campioni)];
plot(freq,abs(ff));
xlabel('Hz');
title('spettro del segnale');
axis([0 500,min(abs(ff)) max(abs(ff))]);
zoom
```

rect1.m

```
function vet=rect1;

tempo=20e-3;
T=1/40000;
campioni=(tempo/T)+1;
k=(1:1:campioni);
yf(length(k))=0;
xf=(k-1)*T;
for i=1:round(campioni/2)-1
   yf(i)=1;
end
vet=yf;
figure
plot(xf,yf);
xlabel('sec');
title('grafico del segnale nel tempo');
ff=fft(yf);
zoom
figure
freq=[0:1/(T*campioni):(1/T)*(1-1/campioni)];
plot(freq,abs(ff));
xlabel('Hz');
title('spettro del segnale');
zoom
```

rect2.m

```
function vet=rect2;

tempo=20e-3;
T=1/40000;
campioni=(tempo/T)+1;
k=(1:1:campioni);
yf(length(k))=0;
xf=(k-1)*T;
for i=1:campioni
   yf(i)=1;
end
vet=yf;
figure
plot(xf,yf);
xlabel('sec');
title('grafico del segnale nel tempo');
ff=fft(yf);
zoom
figure
freq=[0:1/(T*campioni):(1/T)*(1-1/campioni)];
plot(freq,abs(ff));
xlabel('Hz');
title('spettro del segnale');
zoom
```

esecuzione del programma

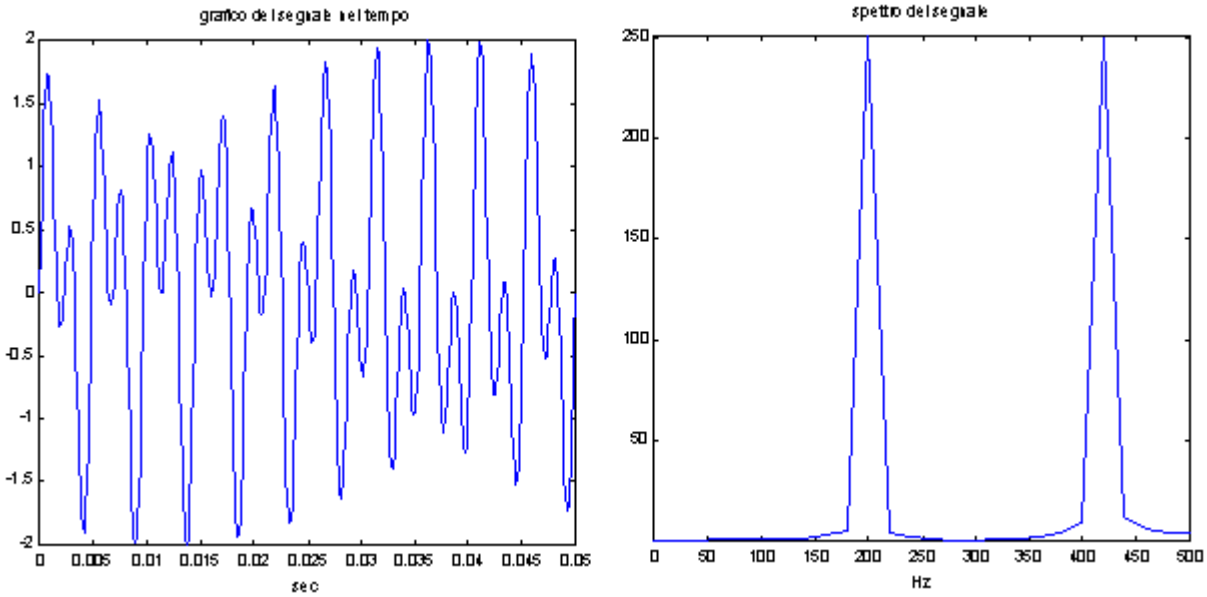
1) convoluzione tra un segnale somma di due sinusoidi a frequenza 200 Hz e 420 Hz e un filtro passa banda che filtra la sinusoide alla frequenza di 200 Hz. Mostreremo quindi le differenze che si hanno quando il filtraggio avviene con metodi differenti:

```
» conv
inserisci il nome della function contenente il segnale:segnale
inserisci il nome della function contenente la risposta impulsiva del filtro:banda

Scegli il problema da risolvere :

1. Convoluzione lineare
2. Convoluzione circolare
3. Convoluzione circolare con l'uso della FFT
4. Convoluzione utilizzando il metodo overlap & save
5. Convoluzione utilizzando il metodo overlap & add

Introduci il numero corrispondente : 1
```

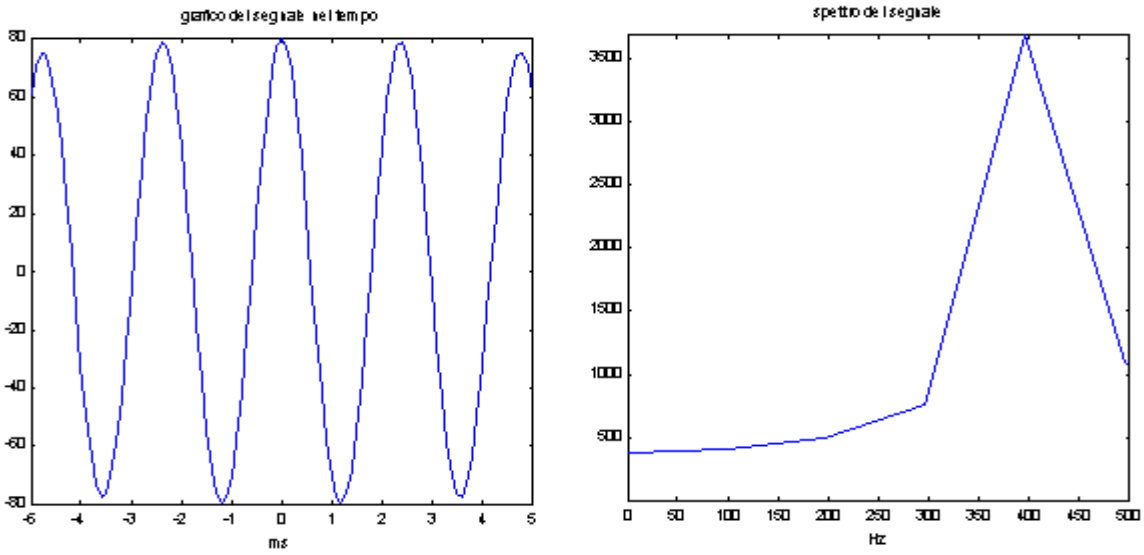
(e successivamente 2 3 4 5)

OSS.: per gli spettri dei 2 segnali considerati, vengono rappresentati soltanto le frequenze positive.

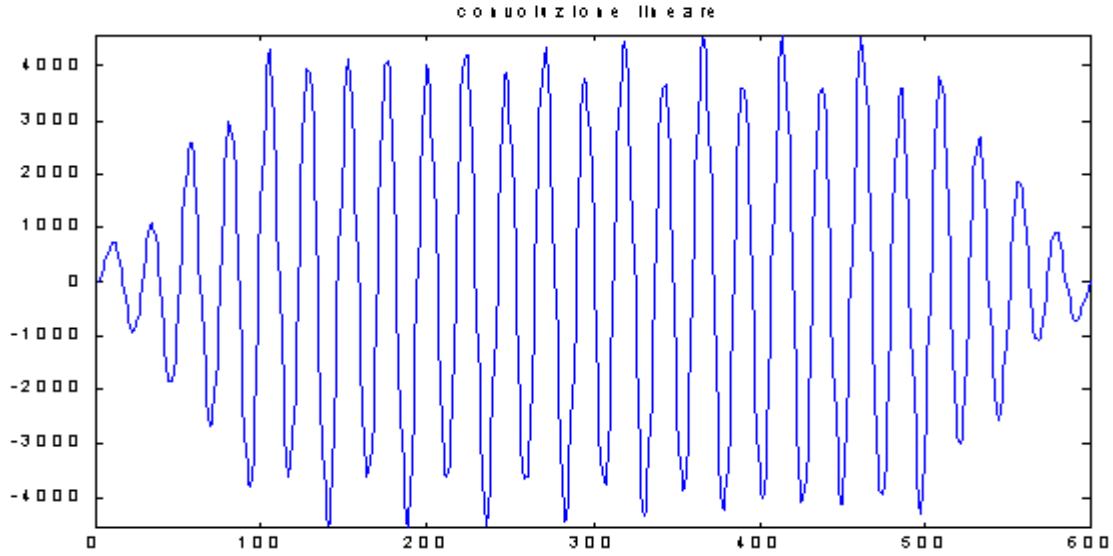
segnale da filtrare rappresentato nel dominio del tempo e in frequenza :

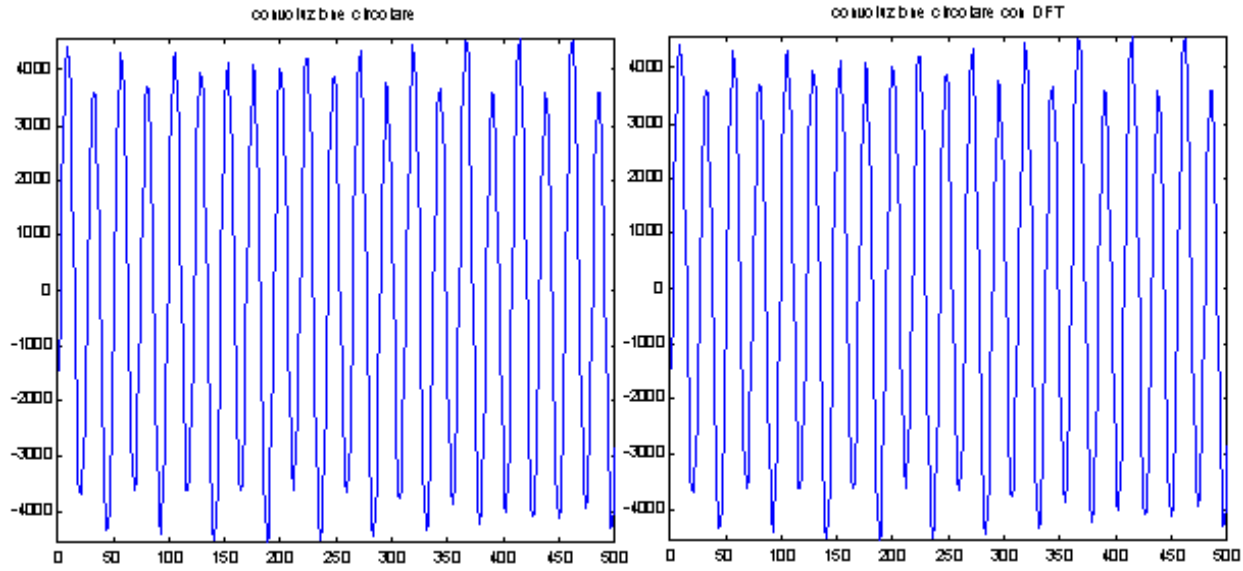


risposta all'impulso e spettro del filtro

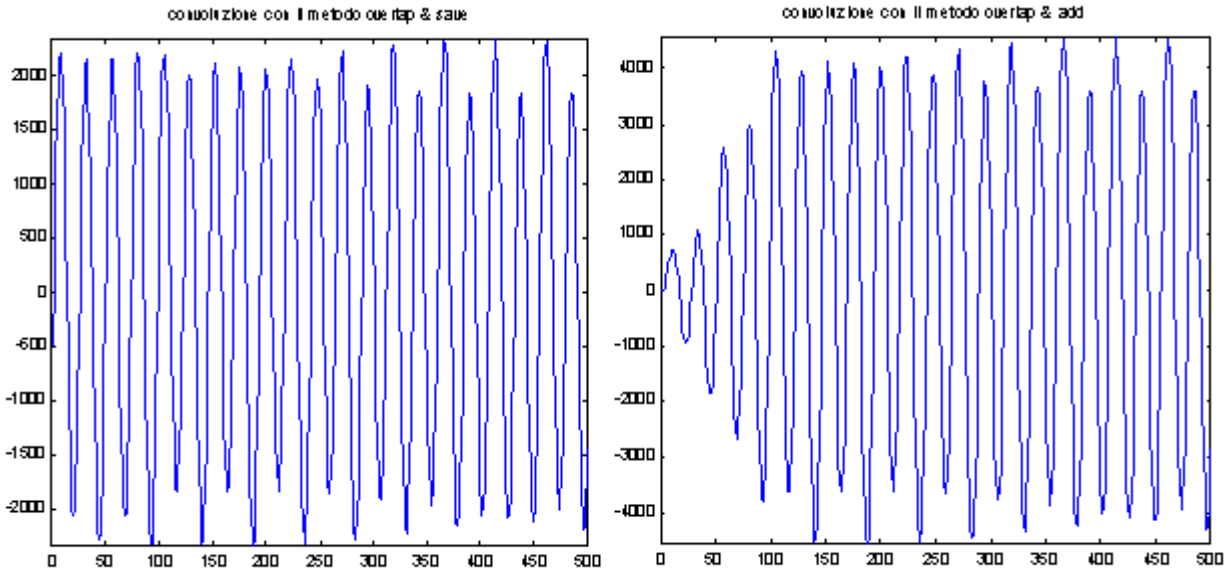


La convoluzione dei due segnali di partenza in funzione del numero di campioni, ottenuta con i vari metodi visti è la seguente:



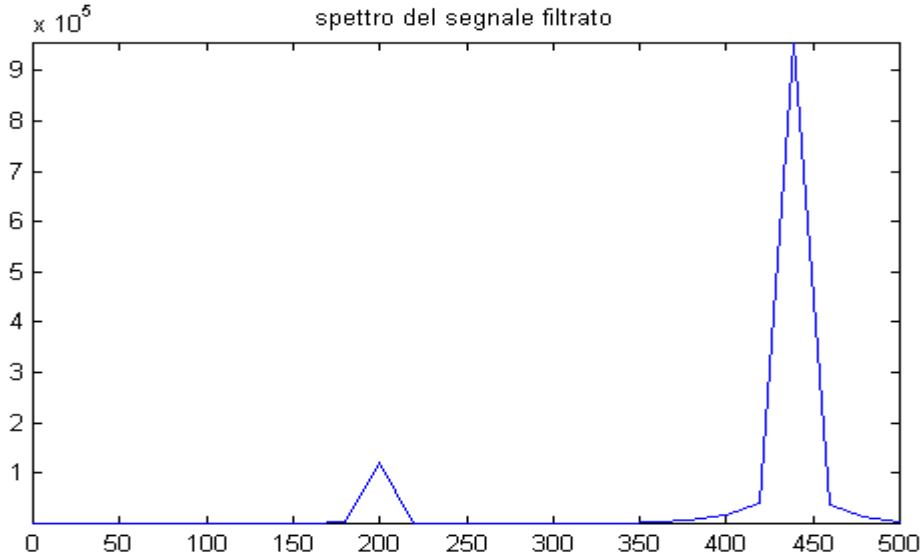


`Il vettore di dimensione maggiore è stato suddiviso in : cicli = 3` (in entrambi i due metodi dell'overlap)



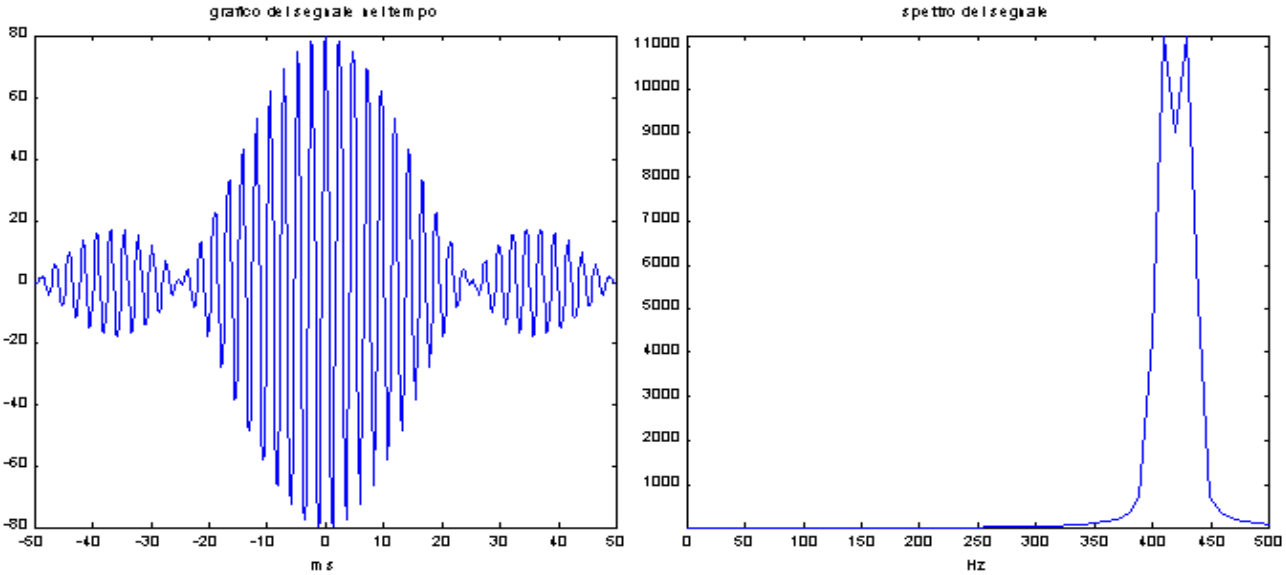
OSS.: come si può notare, il numero di campioni ottenuti con la convoluzione lineare sono maggiori di quelli ottenuti con gli altri metodi, perché solo in questo caso vengono calcolati altri M-1 punti, sempre non corretti, che sono gli ultimi del vettore (come è facile intuire confrontando i vari grafici).
Inoltre i primi M-1 campioni ottenuti con la convoluzione lineare e utilizzando il metodo overlap & add sono diversi da quelli ottenuti negli altri casi: questo è dovuto al fatto che nei primi due casi detti si considera, nel calcolo della convoluzione, il vettore del segnale da filtrare con aggiunti M-1 zeri all'inizio, mentre questo non avviene nei restanti metodi. In particolare la differenza tra il metodo dell'overlap & save e dell'overlap & add sta nel fatto che il primo sottovettore del segnale considerato per calcolare la convoluzione nel primo caso è esattamente la prima parte del segnale di partenza mentre nel secondo caso è quest'ultima con aggiunti M-1 zeri all'inizio (ovviamente questi due vettori hanno FFT differente).
Nel complesso i campioni utili ottenuti con i vari metodi sono praticamente gli stessi e addirittura i due metodi che usano la convoluzione circolare hanno anche uguali i primi M-1 campioni.

OSS.: Il filtraggio del segnale sulla sinusoide a frequenza 200 Hz è avvenuto abbastanza bene, infatti in tutti i casi effettuando la DFT del segnale di uscita otteniamo sempre il tono modulante alla frequenza di 420 Hz mentre quello a 200 Hz è fortemente attenuato (anche se non completamente):



OSS.: la finestra temporale considerata in questi esempi è molto piccola rispetto a quella della risposta all'impulso del filtro in grado di poter apprezzare almeno un lobo secondario. Questo è stato necessario per poter visualizzare il segnale ottenuto dalla convoluzione. Infatti aumentando la finestra temporale della risposta all'impulso il grafico nel tempo e in frequenza diventano :

```
banda
```



dove si può apprezzare meglio la risposta all'impulso e lo spettro del filtro passabanda.
La finestra temporale utilizzata è di 100ms come si può notare dal grafico.

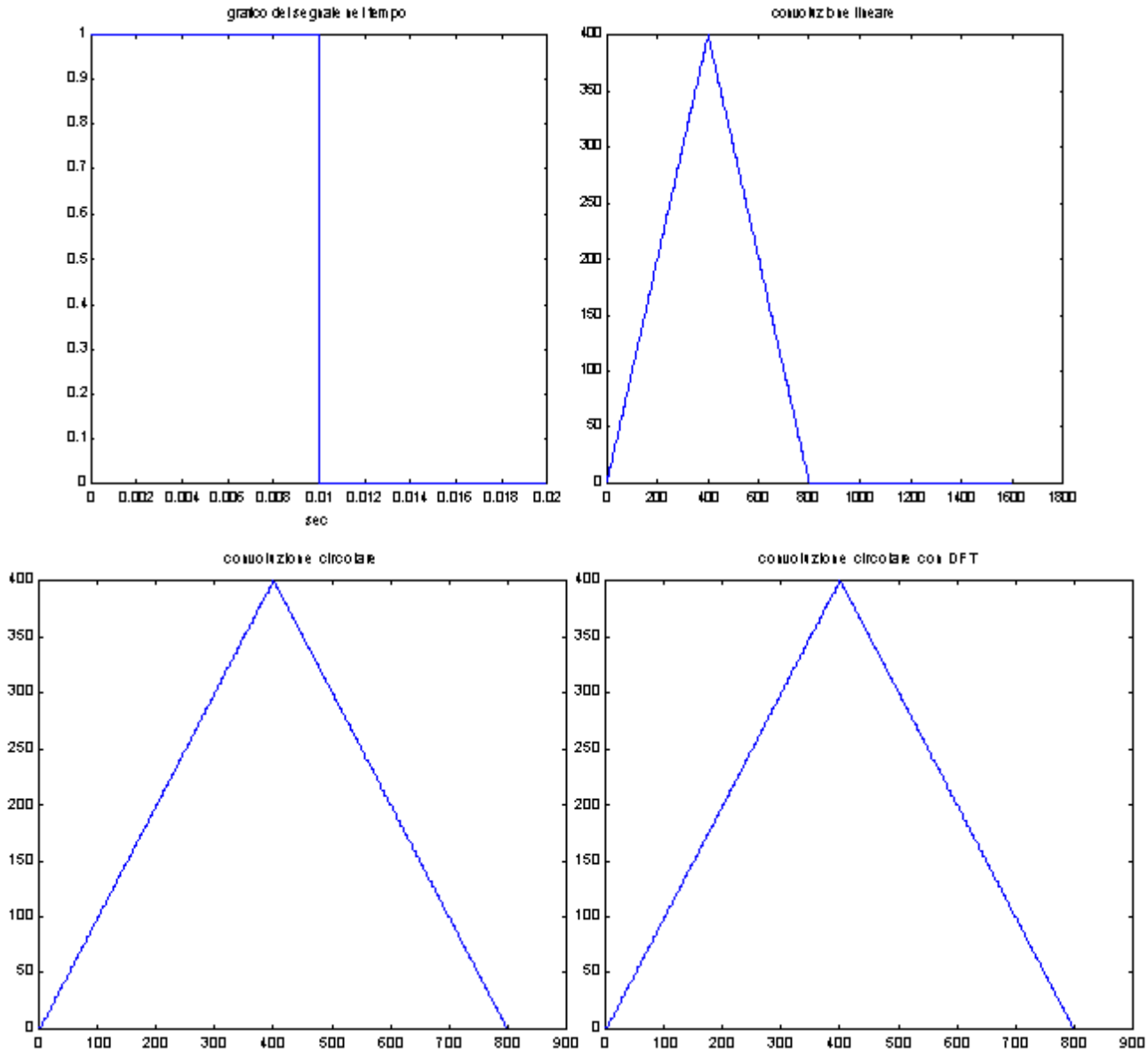
2) convoluzione tra un due segnali rettangolari anche di ampiezza temporale differente. Mostreremo quindi le differenze che si hanno quando il filtraggio avviene con metodi differenti:

OSS.: la convoluzione con il metodo overlap & save e overlap & add non si può effettuare perché i vettori da considerare devono avere lunghezza differente.
Inoltre osserviamo che nel caso in cui un rect assume valore unitario in tutta la finestra temporale la convoluzione circolare valutata in un qualsiasi modo assume anch'essa sempre valore unitario, come si può notare nella seconda esecuzione del programma.

» conv

```
inserisci il nome della function contenente il segnale:rect1
inserisci il nome della function contenente la risposta impulsiva del filtro:rect1
Scegli il problema da risolvere :
1. Convoluzione lineare
2. Convoluzione circolare
3. Convoluzione circolare con l'uso della FFT
4. Convoluzione utilizzando il metodo overlap & save
5. Convoluzione utilizzando il metodo overlap & add
Introduci il numero corrispondente : 1
```
(e successivamente 2 3 4 5)

I grafici del segnale di partenza e la convoluzione con i vari metodi visti sono i seguenti:

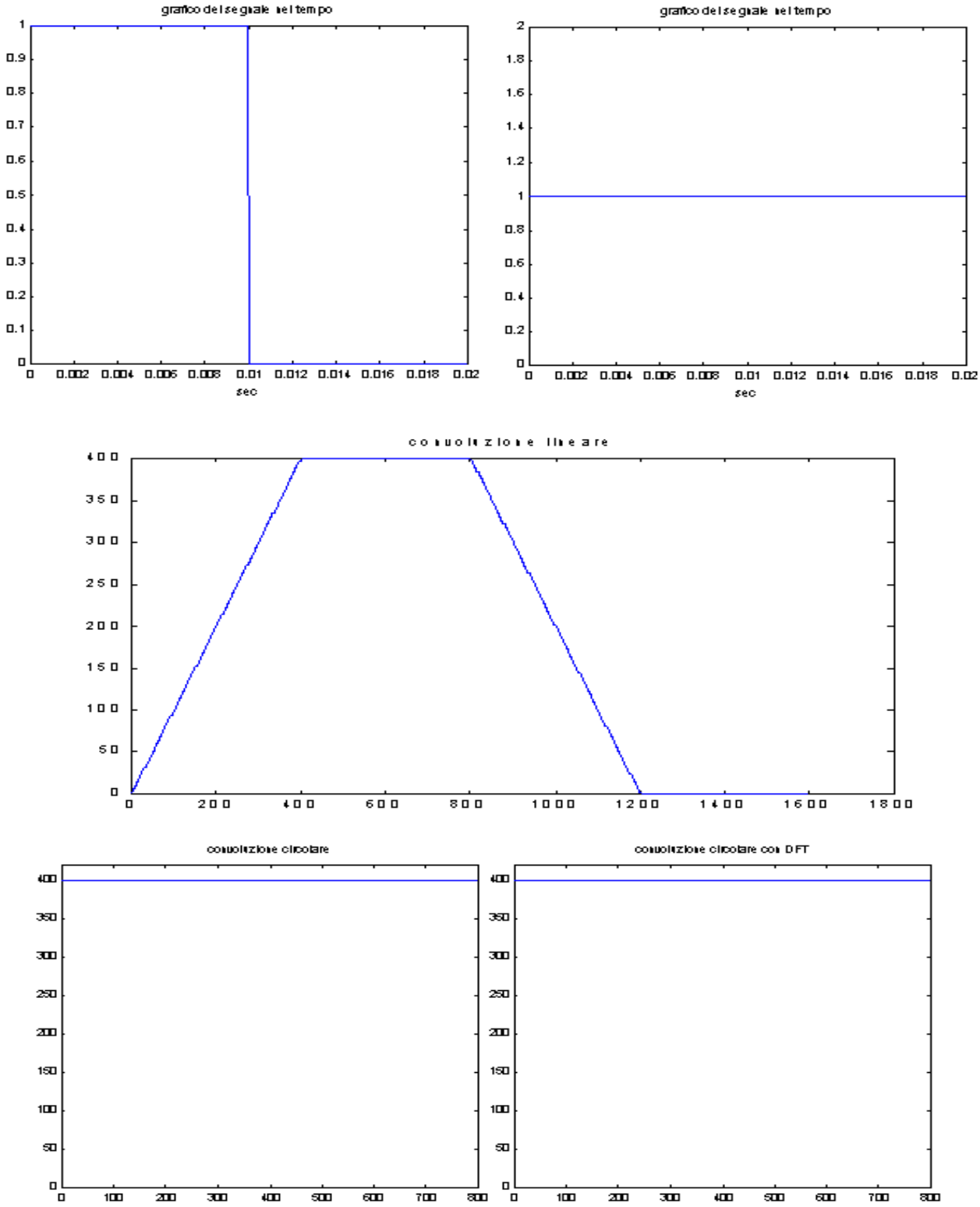


» conv

```
inserisci il nome della function contenente il segnale:rect1
inserisci il nome della function contenente la risposta impulsiva del filtro:rect2
Scegli il problema da risolvere :
1. Convoluzione lineare
2. Convoluzione circolare
3. Convoluzione circolare con l'uso della FFT
4. Convoluzione utilizzando il metodo overlap & save
5. Convoluzione utilizzando il metodo overlap & add
Introduci il numero corrispondente : 1
```

(e successivamente 2 3 4 5)

I grafici del segnale di partenza e la convoluzione con i vari metodi visti sono i seguenti:



Con questo programma verranno realizzati dei filtri FIR passabasso utilizzando le finestre rettangolari, triangolari, cosinusoidali e di Hanning, Hamming e Blackman.

Finestre.m

```
%    PROGETTO DI FILTRI FIR UTILIZZANDO LE FINESTRE

% Verrà realizzato un filtro a partire da quello ideale

% per il quale conosciamo il suo spettro introdotto dall'esterno

% i metodi utilizzati sono differenti

clear all

err=0;
while err==0
   F=input('inserisci il nome della function contenente i punti del filtro ideale : ','s');
   if exist(F)==0
      disp('LA FUNZIONE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end

disp('Scegli il tipo di finestra da utilizzare:');
err=0;
while err==0
   disp('1.Finestra rettangolare');
   disp('2.Finestra triangolare o di Bartlett');
   disp('3.Finestra cosinusoidale');
   disp('4.Finestra coseno rialzato o di Hanning');
   disp('5.Finestra di Hamming');
   disp('6.Finestra di Blackman');
   risp=input('Introduci il numero corrispondente:');
   if risp~=1 & risp~=2 & risp~=3 & risp~=4 & risp~=5 & risp~=6
      disp('Il valore introdotto è errato.Ricomincia');
   else
      err=1;
   end
end

filtro=feval(F);
M=length(filtro);
strM=int2str(M);
disp(['Il numero di campioni del filtro da realizzare sono: ',strM])

for i=1:length(filtro)-1
   filtroinv(i)=filtro(length(filtro)-i+1);
end
filtro=[filtro filtroinv];
N=length(filtro);
strN=int2str(N);
disp(['Il numero di campioni del filtro ideale considerando anche le frequenze '...
      'negative sono: ',strN])
if risp==1

   % FINESTRA RETTANGOLARE

   fin=ones(1,N);
   strfinestra='rettangolare';
end

if risp==2

   % FINESTRA TRIANGOLARE O DI BARTLETT

   for i=2:(N+1)/2
      fin(i)=(2/(1-N))*(i-(N+1)/2);
```

```
      fin(N-i+2)=(2/(1-N))*(i-(N+1)/2);
   end
   fin(1)=1;
   strfinestra='triangolare o di Bartlett';
end

if risp==3

   % FINESTRA COSINUSOIDALE

   for i=2:(N+1)/2
      fin(i)=cos(pi*(i-1)/N);
      fin(N-i+2)=cos(pi*(i-1)/N);
   end
   fin(1)=1;
   strfinestra='cosinusoidale';
end

if risp==4

   % FINESTRA COSENO RIALZATO O DI HANNING

   for i=2:(N+1)/2
      fin(i)=0.5*(1+cos(2*pi*(i-1)/N));
      fin(N-i+2)=0.5*(1+cos(2*pi*(i-1)/N));
   end
   fin(1)=1;
   strfinestra='coseno rialzato o di Hanning';
end

if risp==5

   % FINESTRA DI HAMMING

   for i=2:(N+1)/2
      fin(i)=0.54+0.46*cos(2*pi*(i-1)/N);
      fin(N-i+2)=0.54+0.46*cos(2*pi*(i-1)/N);
   end
   fin(1)=1;
   strfinestra='di Hamming';
end

if risp==6

   % FINESTRA DI BLACKMAN

   for i=2:(N+1)/2
      fin(i)=0.42+0.5*cos(2*pi*(i-1)/N)+0.08*cos(4*pi*(i-1)/N);
      fin(N-i+2)=0.42+0.5*cos(2*pi*(i-1)/N)+0.08*cos(4*pi*(i-1)/N);
   end
   fin(1)=1;
   strfinestra='di Blackman';
end

%CALCOLO DELLA RISPOSTA ALL'IMPULSO
%DISEGNO DELLA RISPOSTA IN FREQUENZA DELLE FINESTRE E DEL FILTRO IDEALE E REALE

x1=(-0.5:1/(N-1):0.5);
graft=[fin((N+3)/2:N) fin(1:(N+1)/2)];
figure
plot(x1,graft);
xlabel('t/NT');
title(['forma della funzione della finestra ',strfinestra]);
zoom

finn=[fin(1:(N+1)/2) zeros(1,39*N) fin((N+3)/2:N)];              % DFT della finestra
ffinn=abs(fft(finn));
graff=[ffinn(40*N-199:40*N) ffinn(1:201)];
```

```
x2=(-5:1/40:5);
figure
plot(x2,graff);
xlabel('fNT');
title(['modulo della risposta in frequenza della finestra ',strfinestra]);
zoom

for i=1:length(finn)
   if ffinn(i)<1e-8
      ffinn(i)=1e-8;
   end
end
dBfinn=20*log10(ffinn);
dBgraff=[dBfinn(40*N-199:40*N) dBfinn(1:201)];
figure
plot(x2,dBgraff);
xlabel('fNT');
ylabel('dB');
title(['modulo della risposta in frequenza della finestra ',strfinestra,' in dB']);
zoom

x3=(-0.5:1/(N-1):0.5);
figure
hold on
grafico=plot(x3,[filtro((N+3)/2:N) filtro(1:(N+1)/2)],'.r');
set(grafico,'linestyle','.','markersize',10)
plot(x3,[filtro((N+3)/2:N) filtro(1:(N+1)/2)]);
title('modulo della risposta in frequenza del filtro ideale');
xlabel('fT');
zoom

risposta=real(ifft(filtro));                    % operazione di finestratura
rispf=risposta.*fin;
filtrof=abs(fft([rispf(1:(N+1)/2) zeros(1,39*N) rispf((N+3)/2:N)]));

x4=(-0.5:1/(40*N-40):0.5);
figure
hold on
grafico=plot(x3,[filtro((N+3)/2:N) filtro(1:(N+1)/2)],'.r');
set(grafico,'linestyle','.','markersize',10);
plot(x4,[filtrof(20*N+21:40*N) filtrof(1:20*N-19)]);
title(['modulo della risposta in frequenza del filtro reale realizzato '...
      'con la finestra ',strfinestra]);
xlabel('fT');
zoom

for i=1:length(filtrof)
   if filtrof(i)<1e-8
      filtrof(i)=1e-8;
   end
end
dBfiltrof=20*log10(filtrof);
figure
plot(x4,[dBfiltrof(20*N+21:40*N) dBfiltrof(1:20*N-19)]);
xlabel('fT');
ylabel('dB');
title(['modulo della risposta in frequenza del filtro reale realizzato '...
      'con la finestra ',strfinestra,' in dB']);
zoom
```

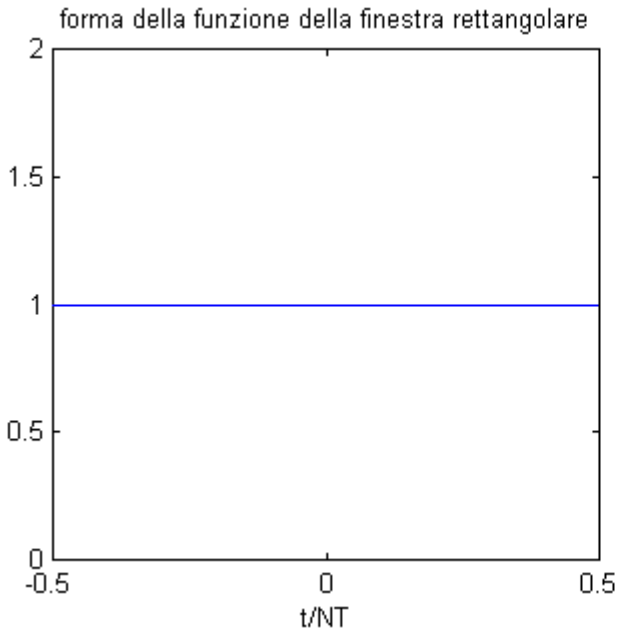
basso.m

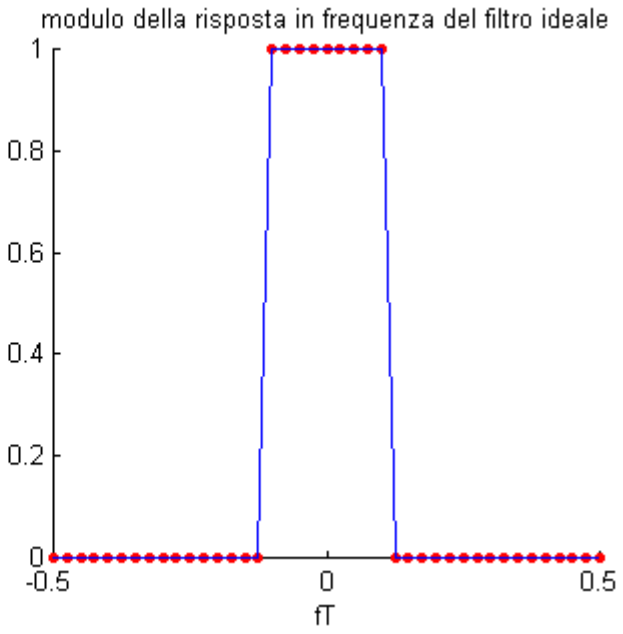
```
function vet=basso

N=21;
vet=[ones(1,(N-1)/5+1) zeros(1,0.8*(N-1))];
```

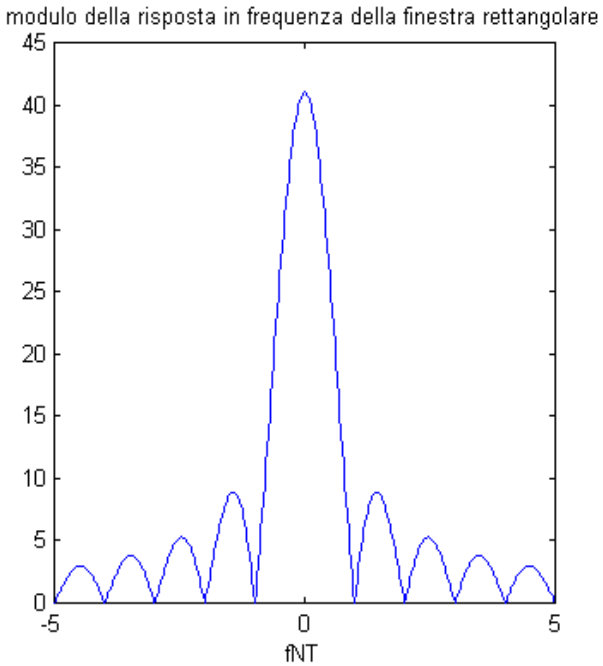
esecuzione del programma

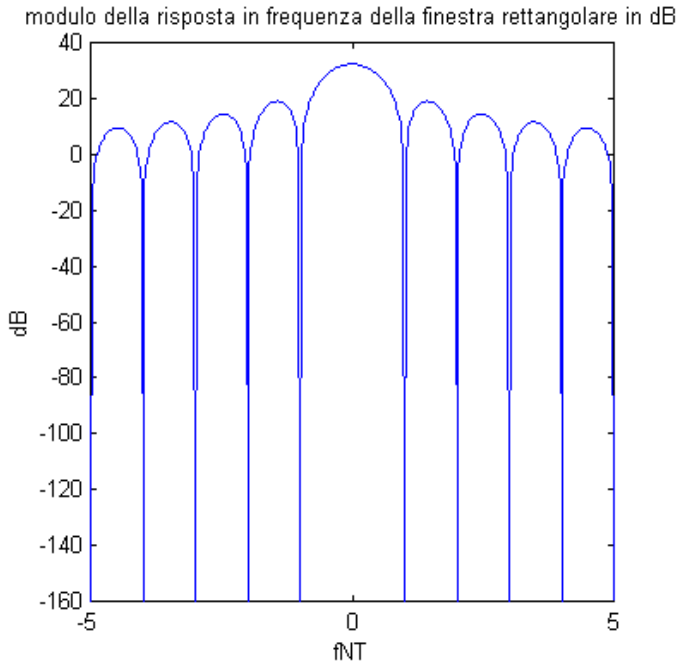
```
» finestre

inserisci il nome della function contenente i punti del filtro ideale : basso
Scegli il tipo di finestra da utilizzare:
1.Finestra rettangolare
2.Finestra triangolare o di Bartlett
3.Finestra cosinusoidale
4.Finestra coseno rialzato o di Hanning
5.Finestra di Hamming
6.Finestra di Blackman

Introduci il numero corrispondente:1              ( e successivamente 2,3,4,5,6)

Il numero di campioni del filtro da realizzare sono: 21
Il numero di campioni del filtro ideale considerando anche le frequenze negative sono: 41
```

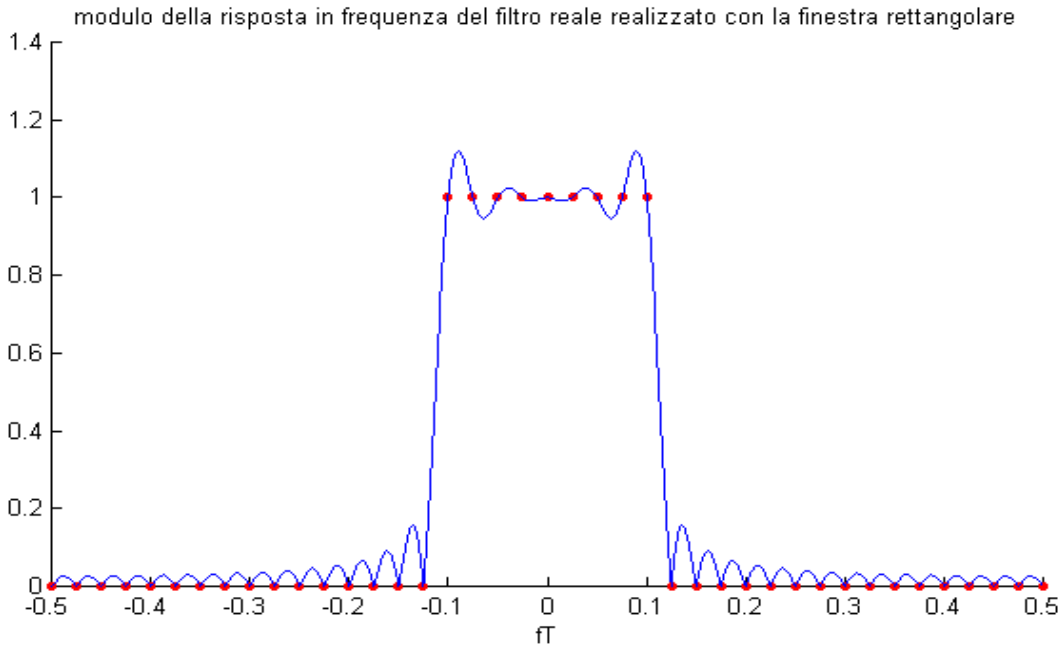
Complessivamente si hanno i seguenti grafici:

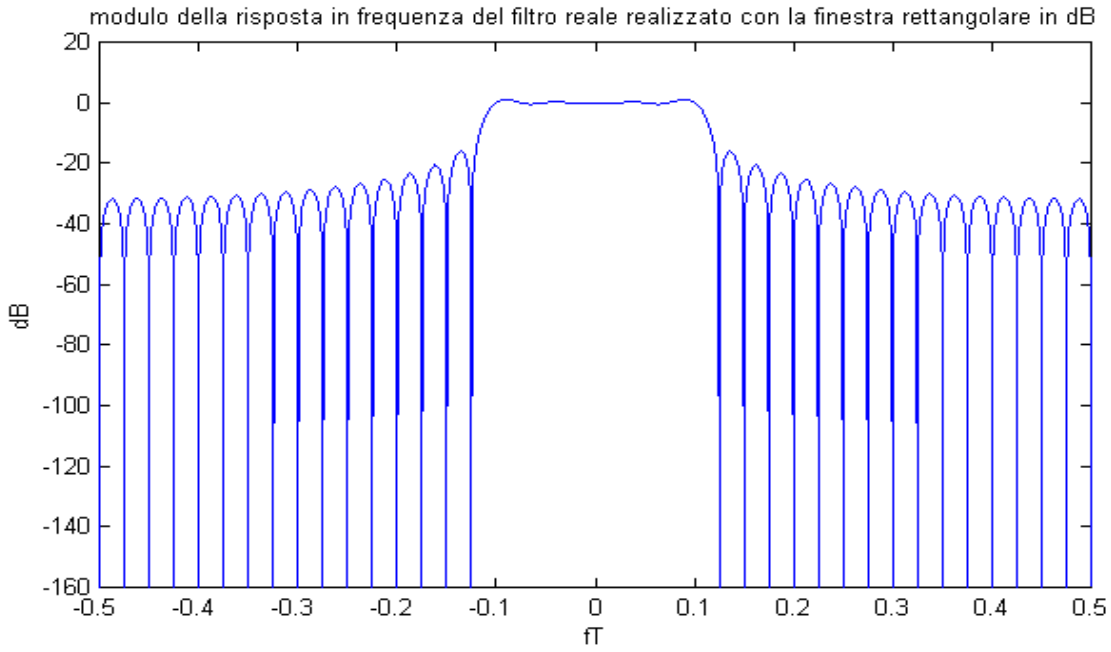


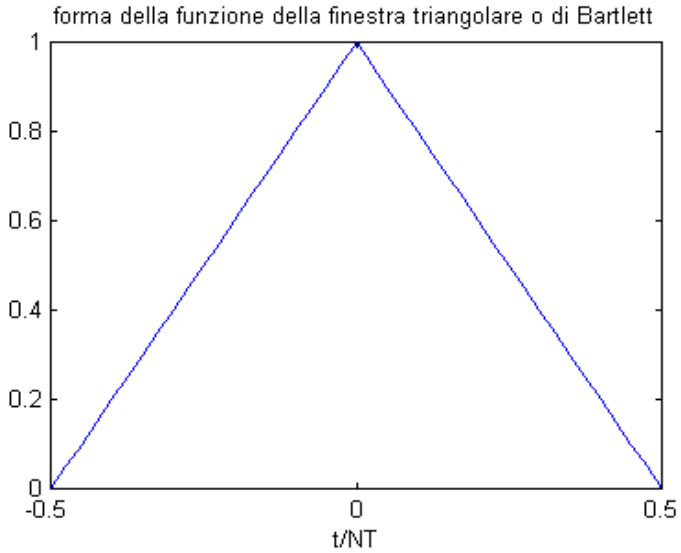


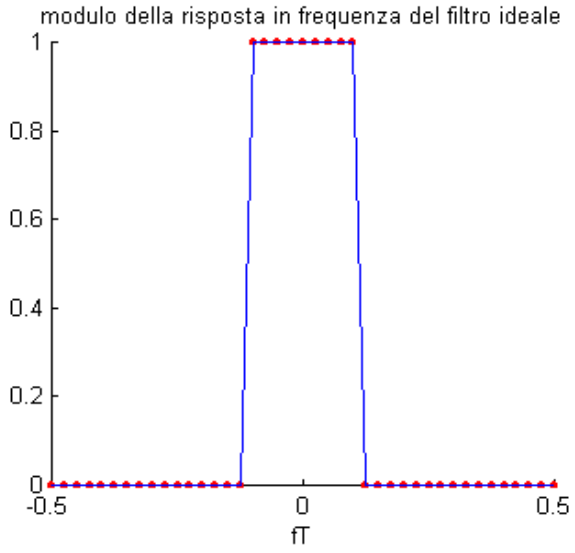


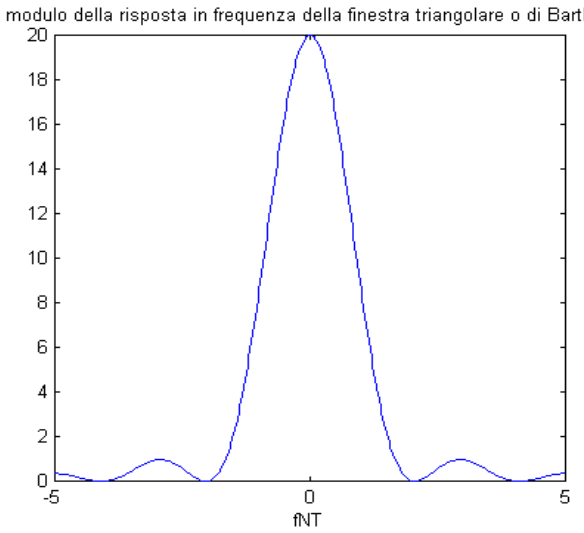


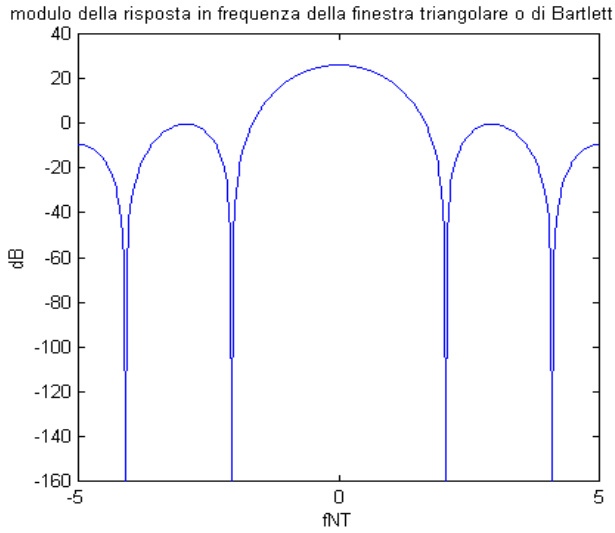

modulo della risposta in frequenza del filtro reale realizzato con la finestra rettangolare
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
fT

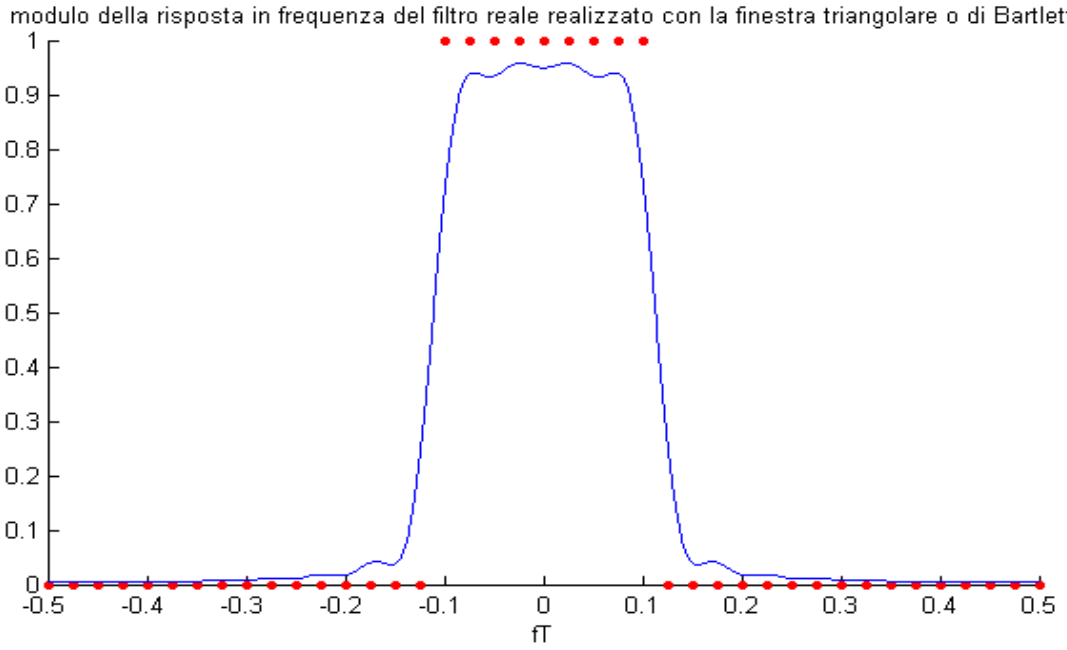

modulo della risposta in frequenza del filtro reale realizzato con la finestra rettangolare in dB
20
0
-20
-40
-60
-80
-100
-120
-140
-160
dB
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
fT

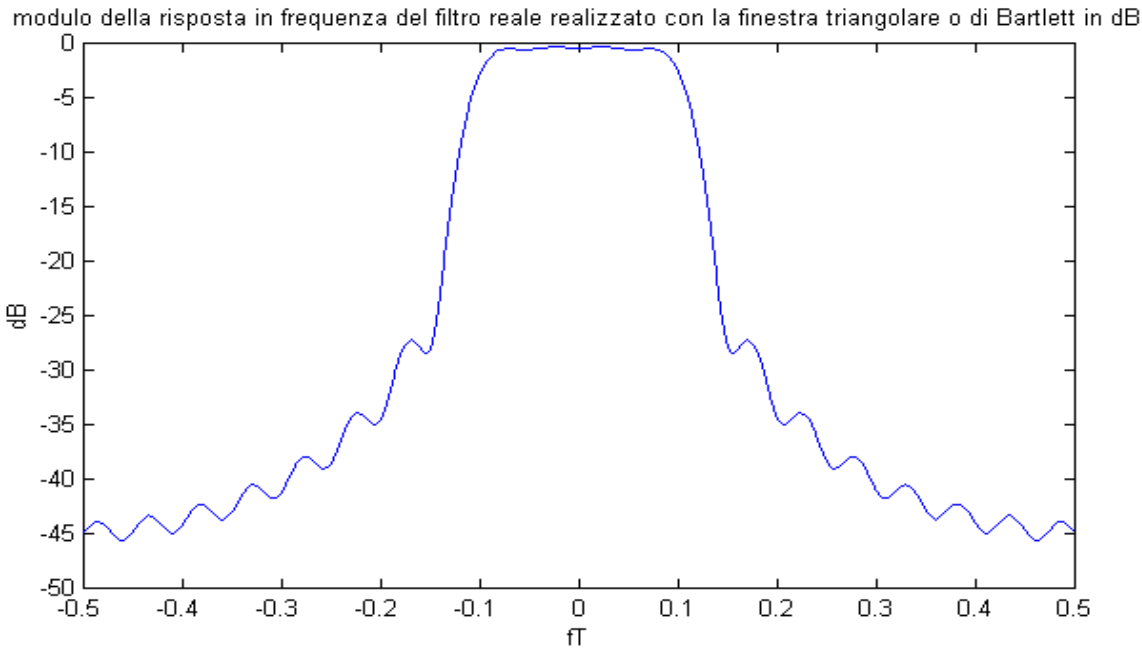

forma della funzione della finestra triangolare o di Bartlett
1
0.8
0.6
0.4
0.2
0
-0.5
0
0.5
t/NT

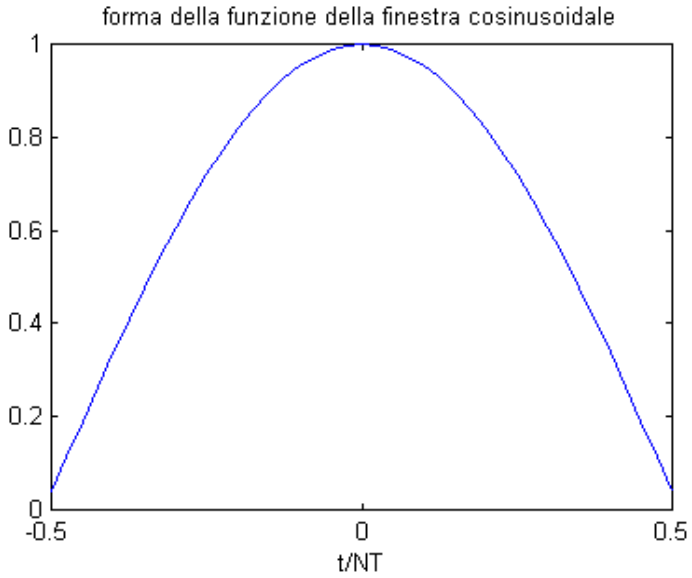

modulo della risposta in frequenza del filtro ideale
1
0.8
0.6
0.4
0.2
0
-0.5
0
0.5
fT

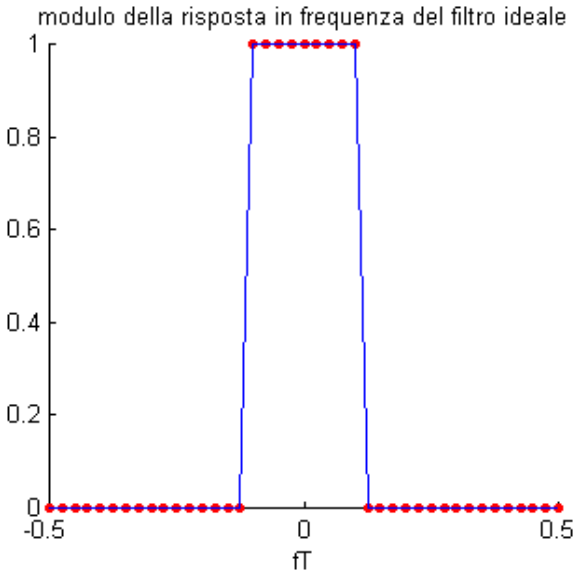

modulo della risposta in frequenza della finestra triangolare o di Bartlett
fNT

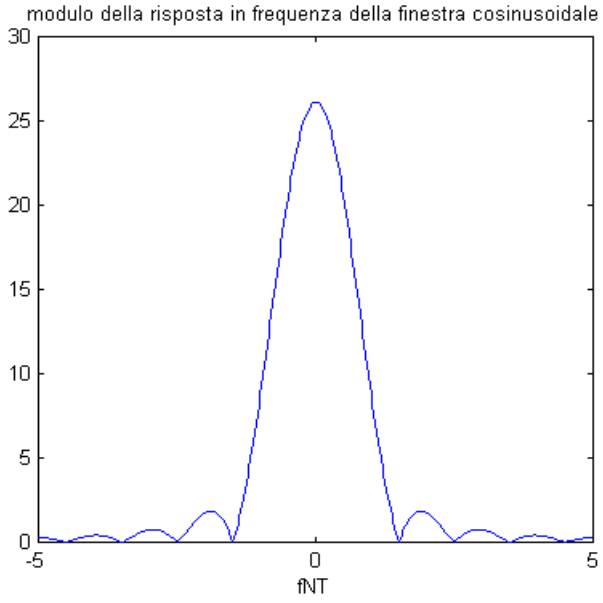

modulo della risposta in frequenza della finestra triangolare o di Bartlett in dB
dB
fNT

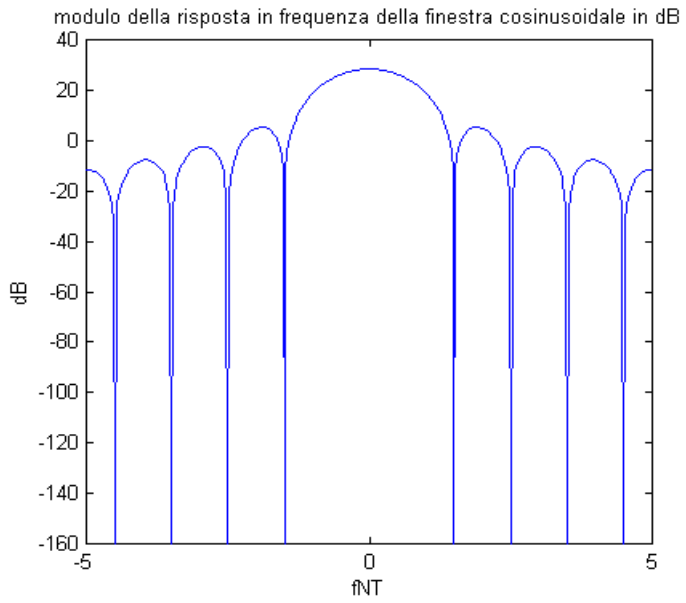

modulo della risposta in frequenza del filtro reale realizzato con la finestra triangolare o di Bartlett
fT

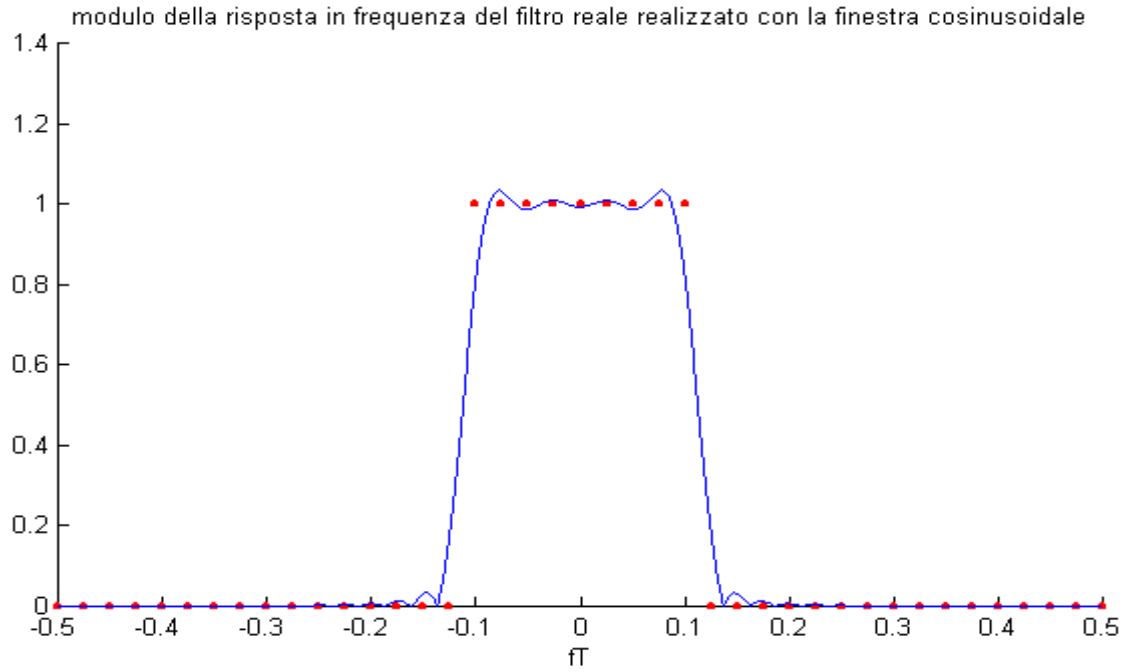

modulo della risposta in frequenza del filtro reale realizzato con la finestra triangolare o di Bartlett in dB
dB
fT

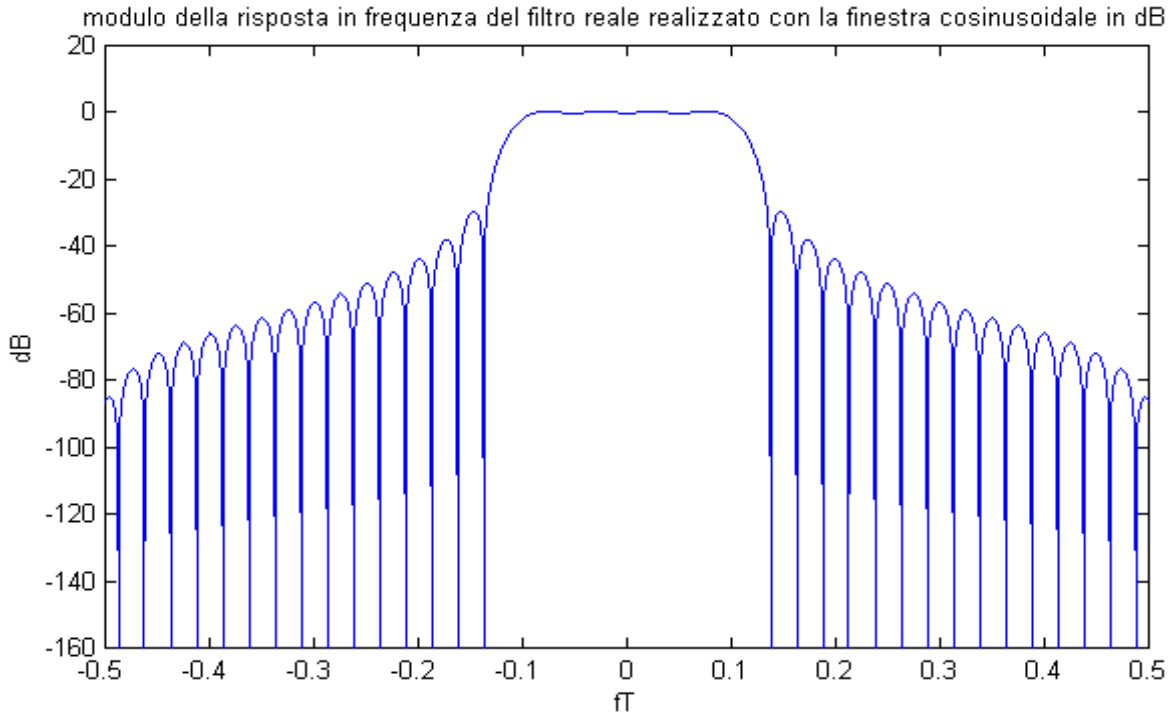

forma della funzione della finestra cosinusoidale
1
0.8
0.6
0.4
0.2
0
-0.5
0
0.5
t/NT

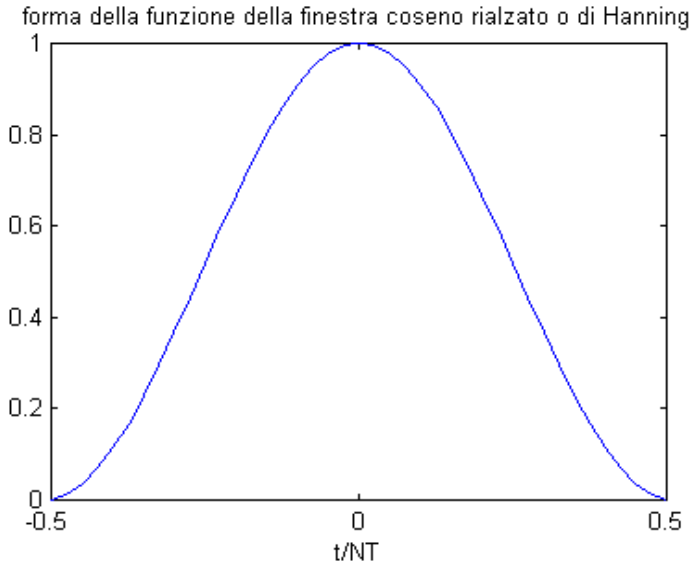

modulo della risposta in frequenza del filtro ideale
1
0.8
0.6
0.4
0.2
0
-0.5
0
0.5
fT

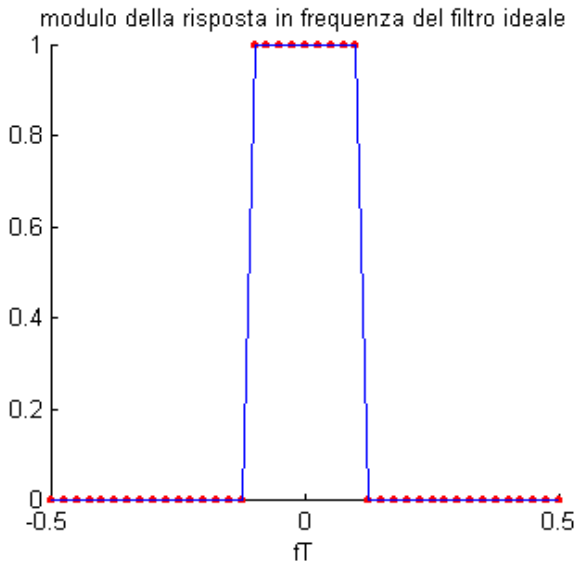

modulo della risposta in frequenza della finestra cosinusoidale
30
25
20
15
10
5
0
-5
0
5
fNT

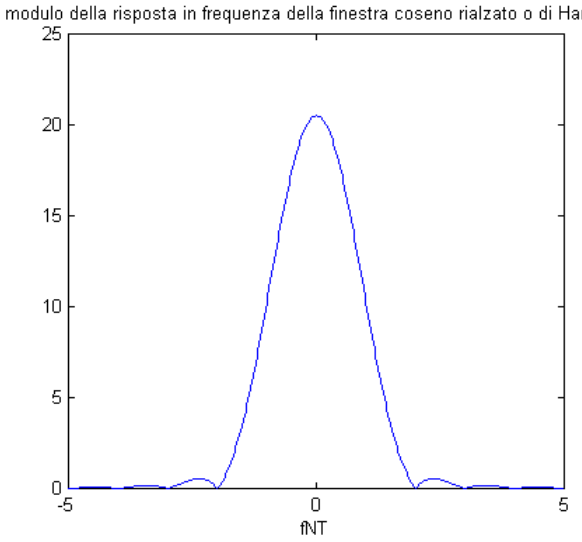

modulo della risposta in frequenza della finestra cosinusoidale in dB
dB
40
20
0
-20
-40
-60
-80
-100
-120
-140
-160
-5
0
5
fNT

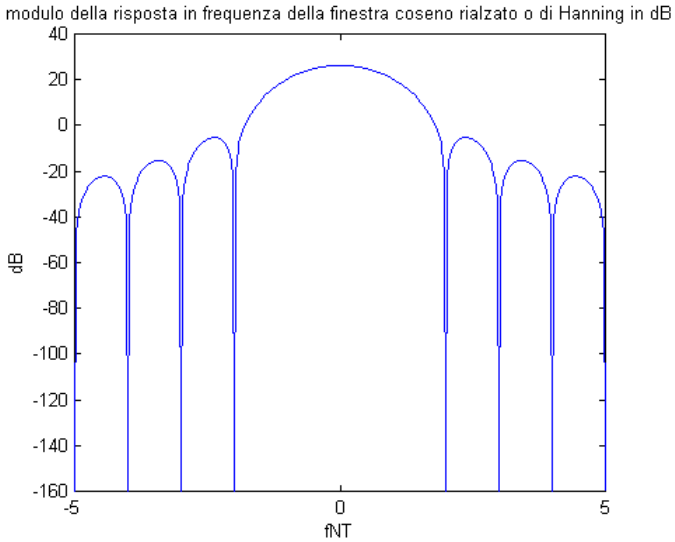

modulo della risposta in frequenza del filtro reale realizzato con la finestra cosinusoidale
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
fT

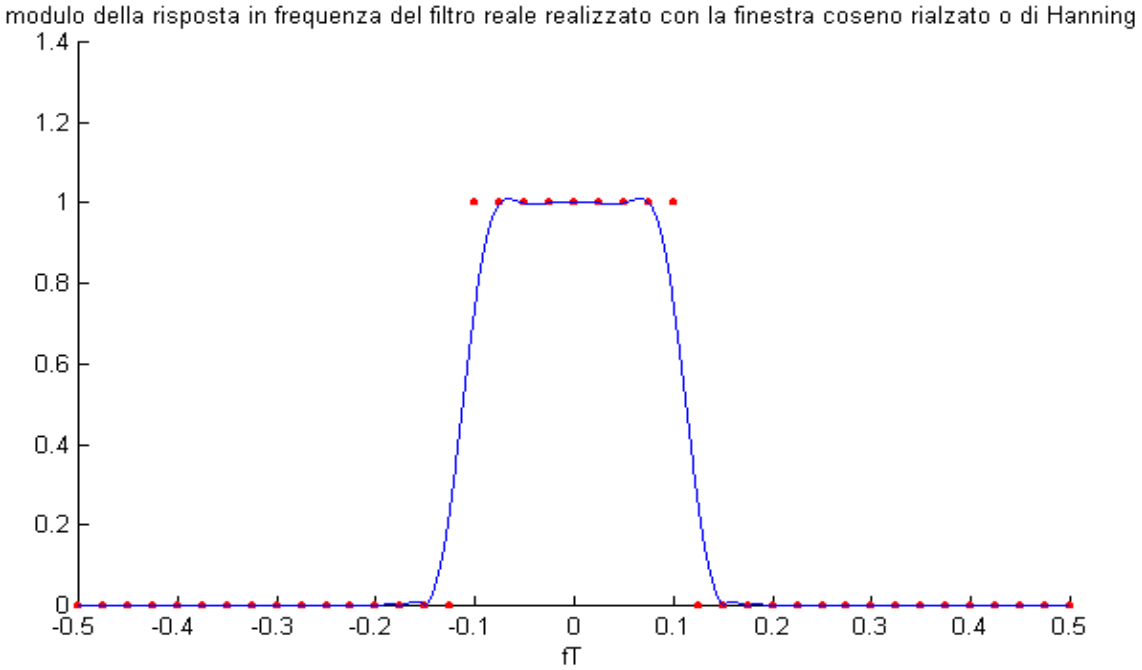

modulo della risposta in frequenza del filtro reale realizzato con la finestra cosinusoidale in dB
dB
fT

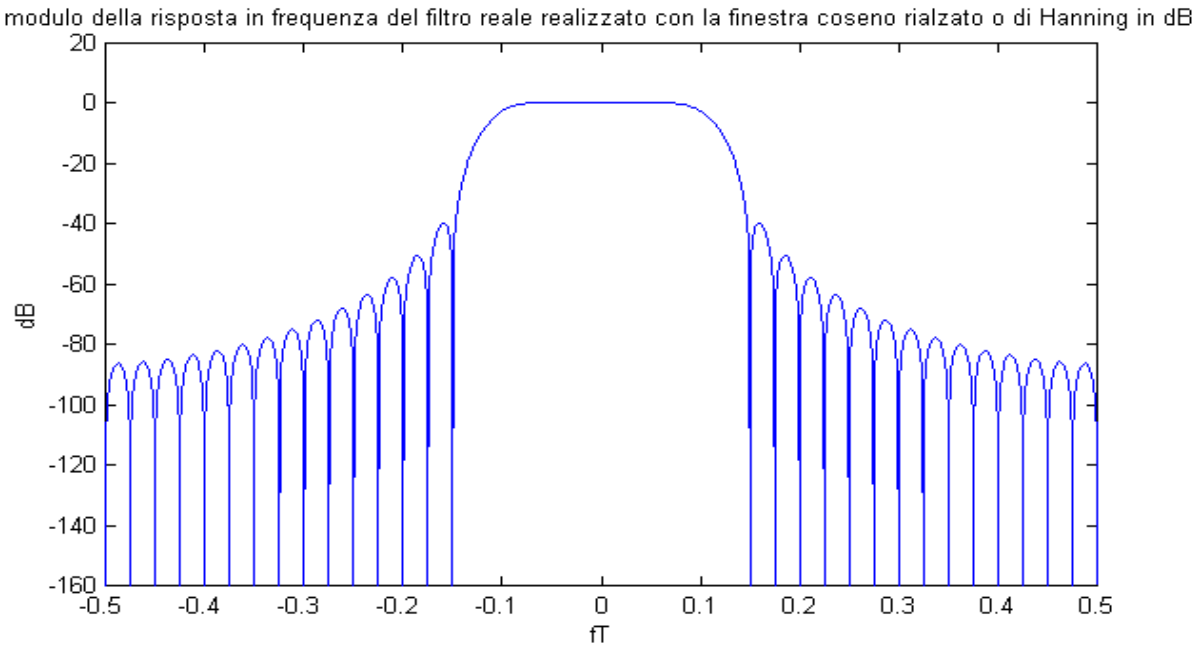

forma della funzione della finestra coseno rialzato o di Hanning
t/NT

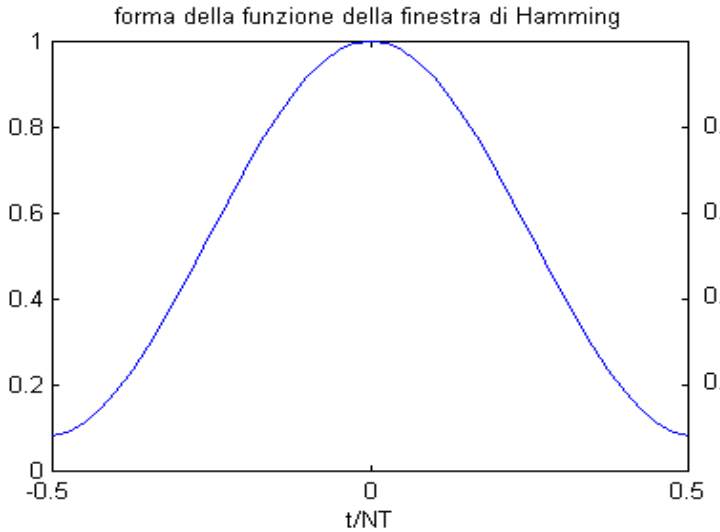

modulo della risposta in frequenza del filtro ideale
fT

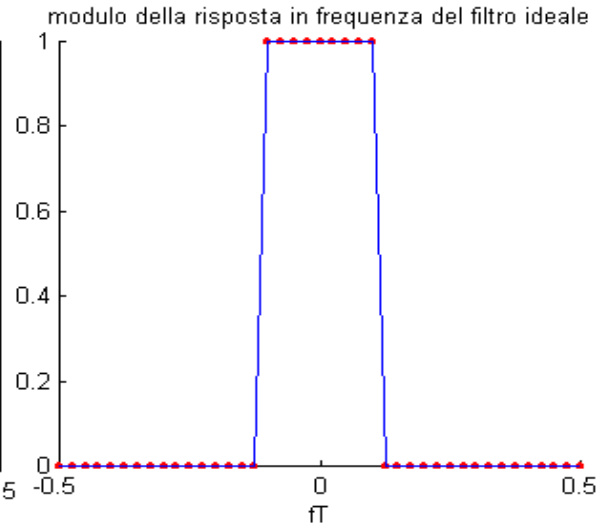

modulo della risposta in frequenza della finestra coseno rialzato o di Hanning
fNT

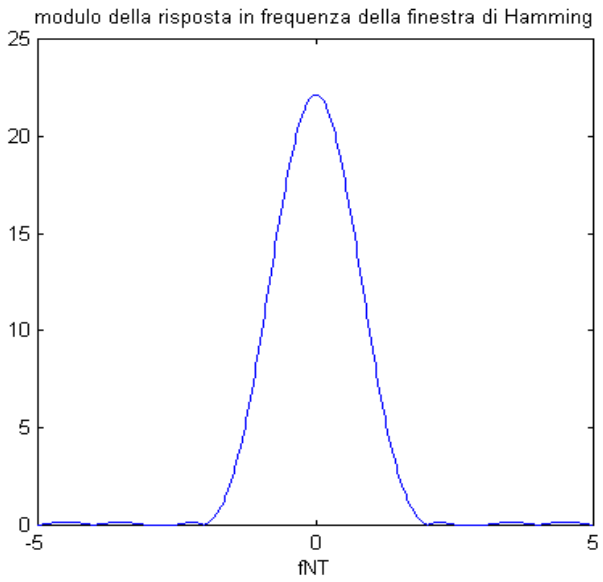

modulo della risposta in frequenza della finestra coseno rialzato o di Hanning in dB
dB
fNT

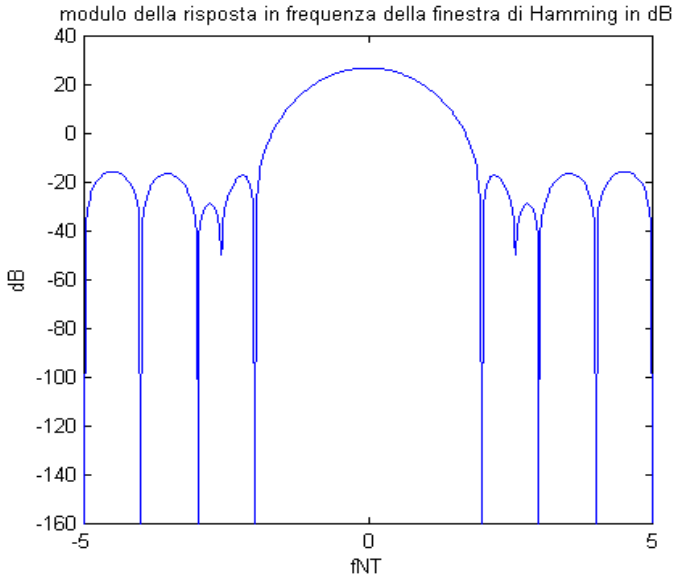

modulo della risposta in frequenza del filtro reale realizzato con la finestra coseno rialzato o di Hanning
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
fT

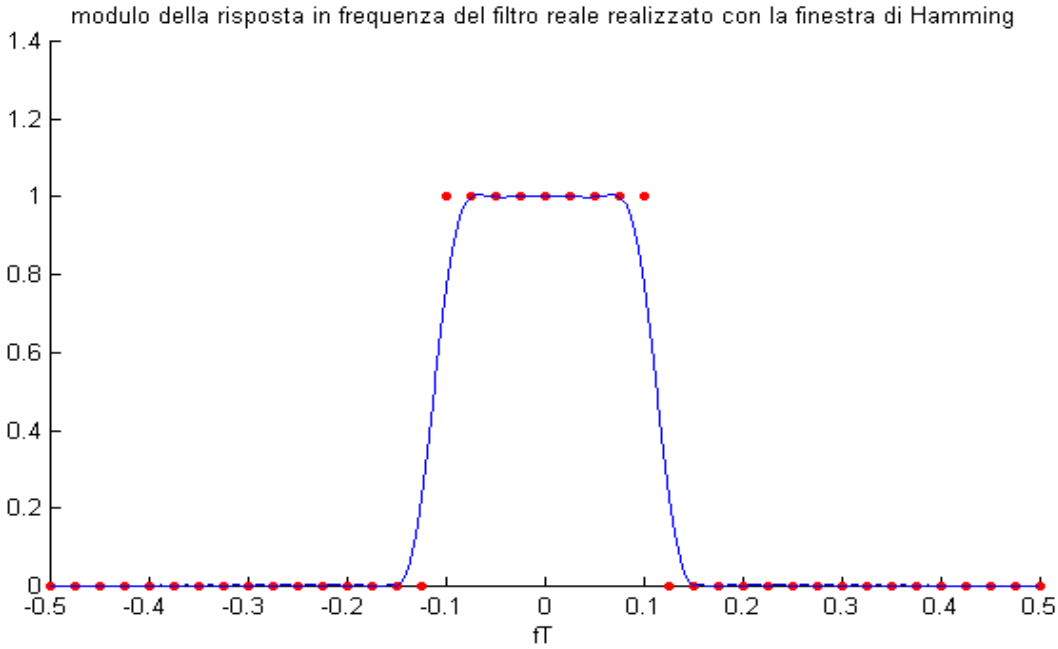

modulo della risposta in frequenza del filtro reale realizzato con la finestra coseno rialzato o di Hanning in dB
20
0
-20
-40
-60
-80
-100
-120
-140
-160
dB
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
fT

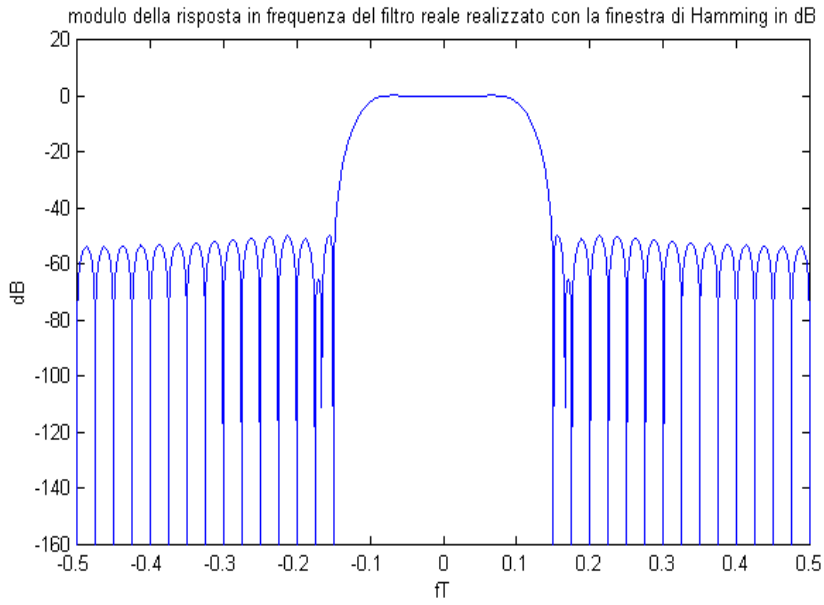

forma della funzione della finestra di Hamming
1
0.8
0.6
0.4
0.2
0
-0.5
0
0.5
t/NT

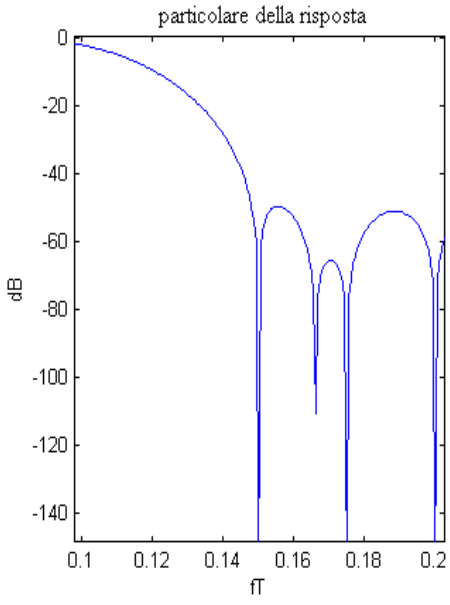

modulo della risposta in frequenza del filtro ideale
1
0.8
0.6
0.4
0.2
0
-0.5
0
0.5
fT

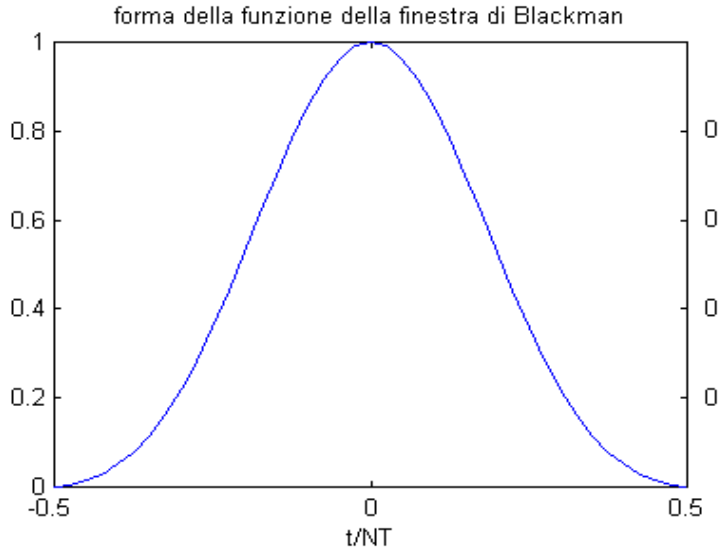

modulo della risposta in frequenza della finestra di Hamming
25
20
15
10
5
0
-5
0
5
fNT

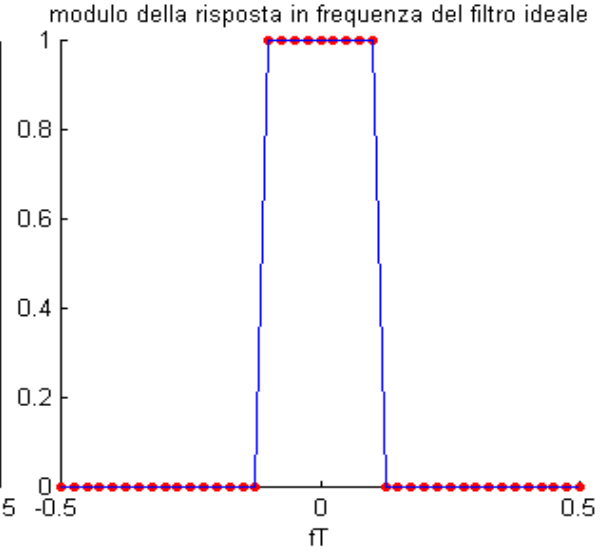

modulo della risposta in frequenza della finestra di Hamming in dB
40
20
0
-20
-40
-60
-80
-100
-120
-140
-160
dB
-5
0
5
fNT

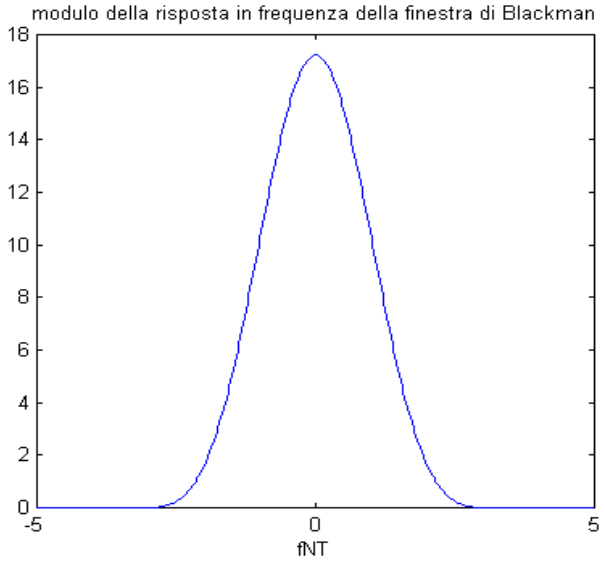

modulo della risposta in frequenza del filtro reale realizzato con la finestra di Hamming
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
fT

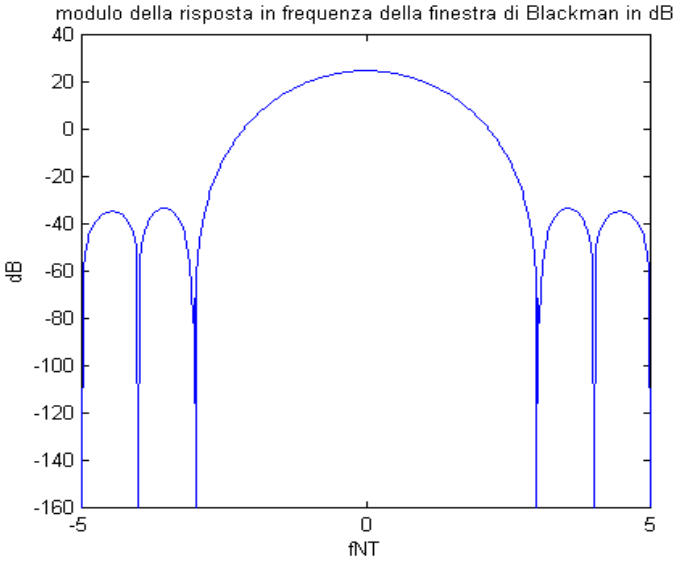

modulo della risposta in frequenza del filtro reale realizzato con la finestra di Hamming in dB
20
0
-20
-40
-60
-80
-100
-120
-140
-160
dB
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
fT

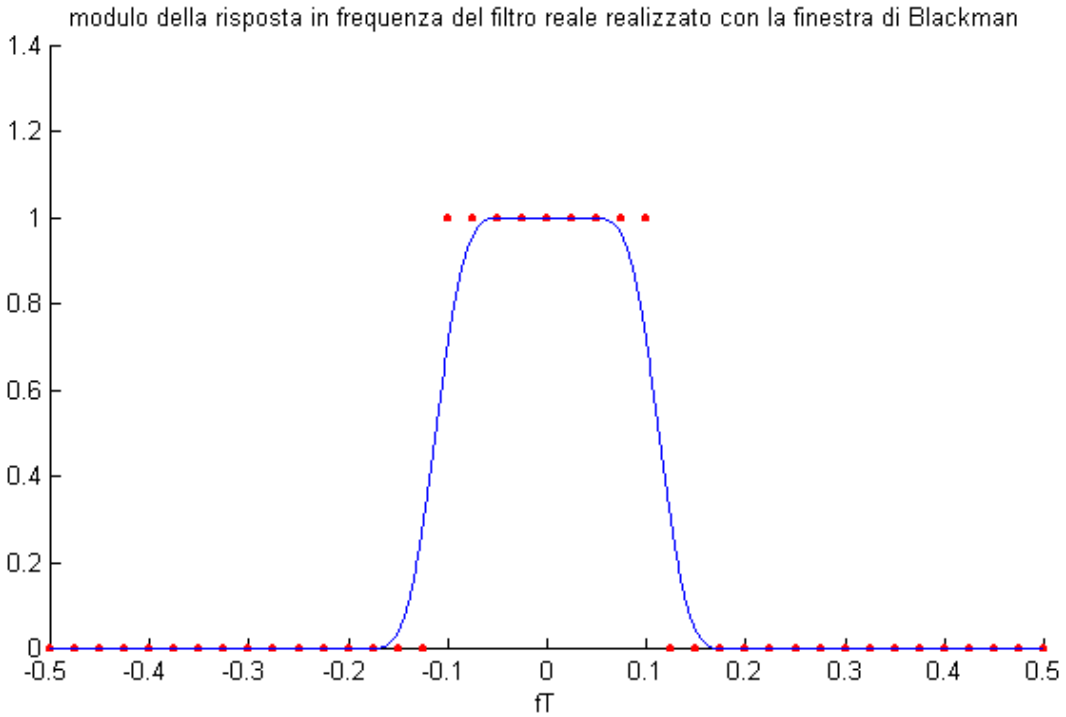

particolare della risposta
0
-20
-40
-60
-80
-100
-120
-140
dB
0.1
0.12
0.14
0.16
0.18
0.2
fT

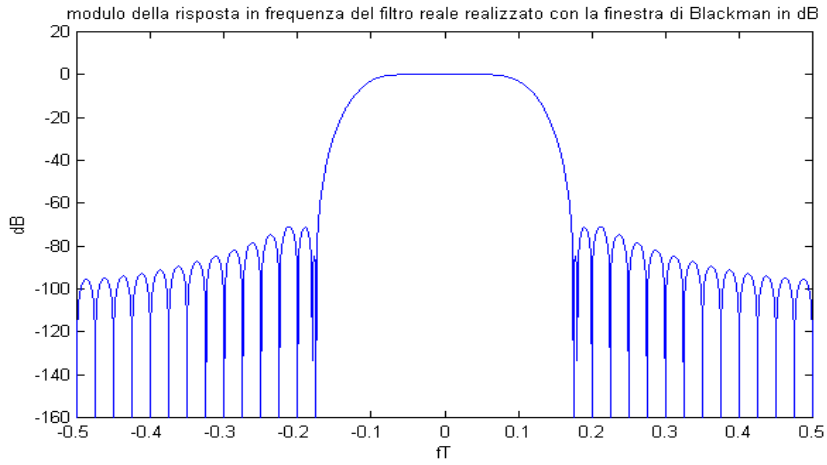

forma della funzione della finestra di Blackman
1
0.8
0.6
0.4
0.2
0
-0.5
0
0.5
t/NT

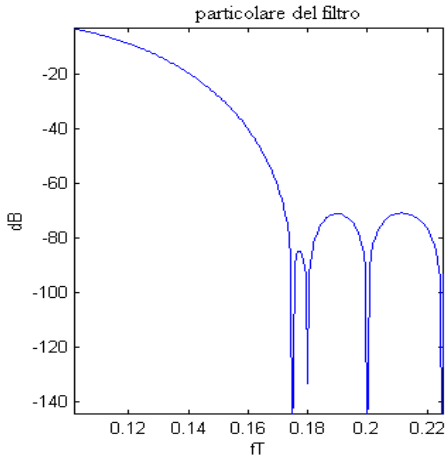

modulo della risposta in frequenza del filtro ideale
1
0.8
0.6
0.4
0.2
0
-0.5
0
0.5
fT


modulo della risposta in frequenza della finestra di Blackman
18
16
14
12
10
8
6
4
2
0
-5
0
5
fNT

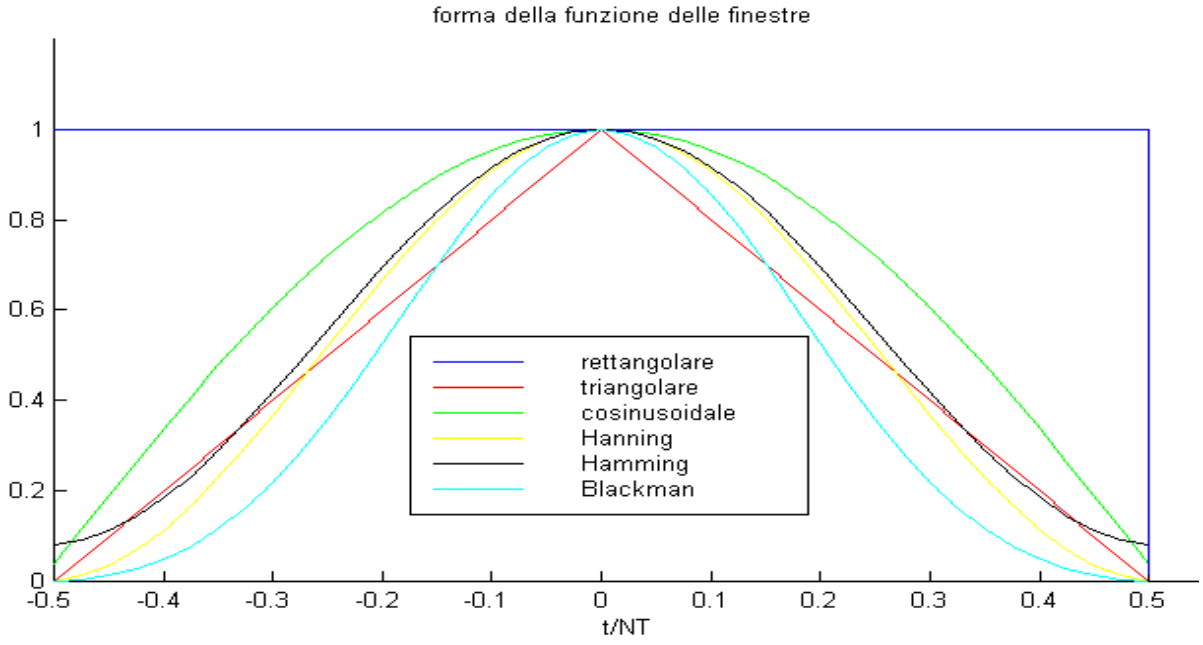

modulo della risposta in frequenza della finestra di Blackman in dB
40
20
0
-20
-40
-60
-80
-100
-120
-140
-160
dB
-5
0
5
fNT

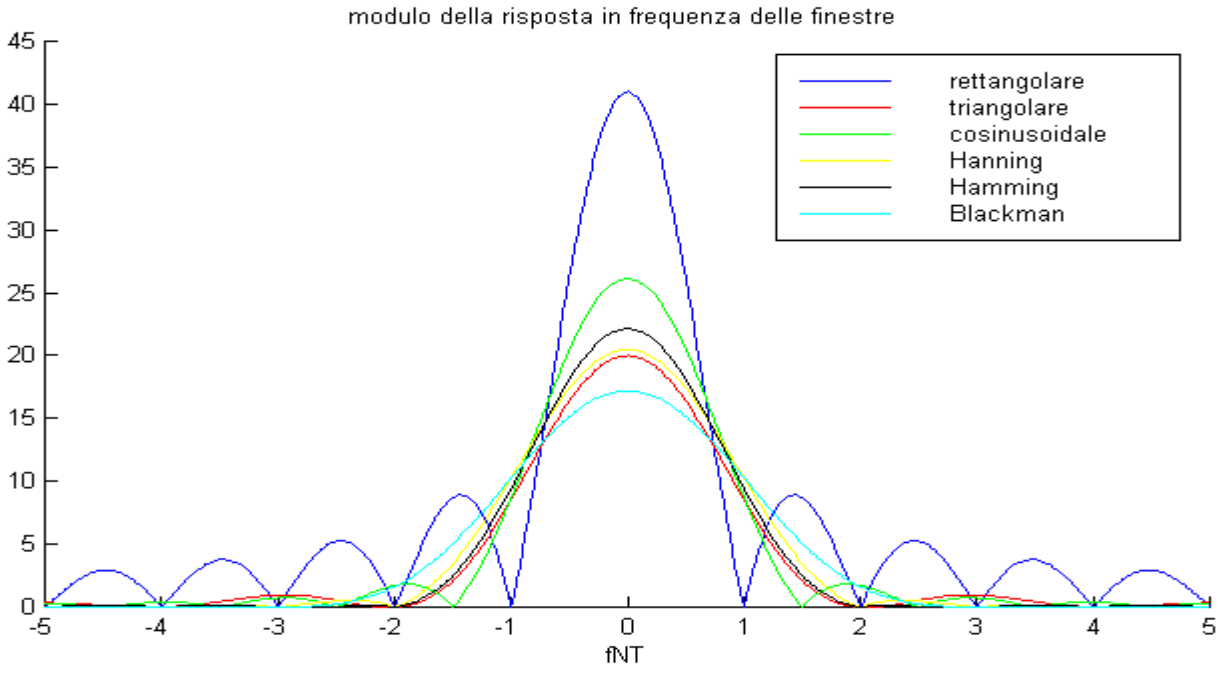

modulo della risposta in frequenza del filtro reale realizzato con la finestra di Blackman
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
fT





Nel complesso le finestre utilizzate sono le seguenti:





In questo programma verrà sviluppata una tecnica di progetto di filtri FIR a fase lineare (in questo caso particolare passabasso e passabanda) utilizzando i metodi di realizzazione dei filtri ottimi che sfruttano l'approssimazione di Chebyshev e quindi il **teorema dell'alternanza**:
Teorema:
se P(jw) è una combinazione lineare di r funzioni coseno , cioè:

$$P(j\omega) = \sum_{n=0}^{r-1} a(n)\cos(\omega n)$$

allora condizione necessaria e sufficiente perché P(jω) sia l'unica migliore approssimazione di una funzione continua D(jω) su A , sottoinsieme compatto di [0,π] , è che l'errore pesato :

$$E(j\omega) = W(j\omega)[\,D(j\omega) - P(j\omega)\,]$$

presenti almeno r+1 frequenze di massimo o minimo in A , ovvero devono esistere in A (r+1) punti ω_i :

tali che: $\omega_1 < \omega_2 < .. < \omega_{r+1}$ e tali che: $E(j\omega_i) = -E(j\omega_{i+1})_{\,i=1..r}$ e $|E(j\omega_i)| = \max_{\omega \in A} |E(j\omega)|$

OSS.: Il teorema permette di trovare la soluzione ottima secondo Chebyshev.

I filtri realizzati sono i seguenti:

- **basso.m** : è un filtro passabasso di lunghezza pari a 13 punti e frequenze di transizione normalizzate rispetto alla frequenza di Nyquist pari a : f_p =0.333 e f_s =0.5 . La funzione peso invece moltiplica di un fattore 10 l'errore in banda passante;
- **banda.m** : è un filtro passabanda di lunghezza pari a 21 punti e con bande attenuate comprese tra le frequenze normalizzate rispetto alla frequenza di Nyquist di : f_1 =0 , f_2 =0.2 e f_5 =0.8 , f_6 =1 e banda passante compresa tra f_3 =0.3 e f_4 =0.7 . La funzione peso invece moltiplica di un fattore 10 l'errore in banda passante;
- **bassopk.m** : è un filtro passabasso di 61 punti e frequenze di transizione normalizzate pari a : f_p =0.2 e f_s =0.3. La funzione peso invece è unitaria in tutto il campo di frequenze.

La soluzione del problema secondo l'approssimazione di Chebyshev può essere ottenuta impostando l'algoritmo di Remez secondo Parks e McClellan come verrà proposto in questo in questo programma, stando attenti che non sempre questo algoritmo converge. Infine verrà visualizzata la risposta del filtro modificando la risoluzione in bit dei coefficienti del polinomio trigonometrico calcolati.

remez.m

```
% PROGETTO DI FILTRI FIR CON LA TECNICA MINMAX

% Verrà realizzato un filtro FIR a partire dalla
% sua funzione di trasferimento ideale introdotta
% dall'esterno considerando le SOLE frequenze positive

% il progetto tieni conto dell'algoritmo di Remez
% ed in particolare fa uso del solo caso 1 visto nella teoria ,in cui
% si considerano un numero di campioni pari ad NFILT=2r-1 (sempre dispari)
% ed il filtro è sempre a simmetria pari e deve essere composto da rettangoli
% in cui verranno specificati gli estremi di banda (filtro costante a pezzi)
% oppure deve essere un filtro lineare a pezzi

% r-1 sarà quindi l'ordine del polinomio trigonometrico  mentre
% r+1 sono i massimi e minimi successivi dell'errore massimo di
% modulo uguale e segno alterno che individueremo (frequenze estremali)

% indico con EDGE il vettore contenente gli estremi di banda del filtro
% normalizzati alla frequenza di Nyquist (vettore con elementi compresi tra 0 e 1)

% WTX è il vettore contenente i pesi dell'errore normalizzati
% nelle bande specificate precedentemente in EDGE

% f è il vettore contenente le frequenze in cui conosciamo
% la funzione di trasferimento ideale di partenza

% EDGE,WTX,f devono essere introdotte dall'esterno tutte insieme con una function

% con la function 'peso' verrà individuato il valore della funzione peso,
% che si trova nel vettore WTX, in corrispondenza della banda del filtro
% in cui si trova l'elemento della griglia di frequenze preso in considerazione

% il polinomio trigonometrico nei punti della griglia si ottiene risolvendo
% un sistema di equazioni lineari che valuta i vari coefficienti del polinomio stesso
```

```
clear all

err=0;
while err==0

   % i punti del filtro da introdurre devono corrispondere SOLO alle frequenze positive

   F=input('inserisci il nome della function contenente il filtro ideale da realizzare: ','s');
   if exist(F)==0
      disp('LA FUNZIONE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end

LGRID=input('introduci il valore della densità della griglia per calcolare E(w) : ');
grafici=input('Per disegnare i grafici ad ogni passo di interazione digitare "1" : ');
if grafici==1
   DB=input('Per disegnare i grafici in dB digitare "1" : ');
else
   DB=0;
end

[f,filtro,EDGE,WTX]=feval(F);% introduzione del filtro e delle sue caratteristiche
r=length(filtro);
strr=int2str(r);
disp(['Il numero di campioni del filtro da realizzare sono: ',strr]);

figure
hold on
plot(f,filtro,'r');
grafico=plot(f,filtro,'.b');
set(grafico,'markersize',14);
set(gcf,'color',[1 1 1]);
title('filtro ideale di partenza');
xlabel('f/(fc/2)');
zoom

NBANDS=length(WTX);

j=1;                              % costruzione della griglia di frequenze
for t=0:1/(LGRID*r):1
   for i=1:2:NBANDS*2-1
      if t>=EDGE(i) & t<=EDGE(i+1)
         s(j)=t;
         j=j+1;
      end
   end
end
L=length(s);
if s(1)>EDGE(1);
   g(1)=EDGE(1);
   g(2)=s(1);
   k=3;
else
   g(1)=s(1);
   k=2;
end
indice1(1)=1;
kin=2;
for j=2:L-1                       % aggiungiamo gli estremi di banda nella griglia
   agg=0;
   for i=1:2:NBANDS*2-1
      if s(j)>EDGE(i) & s(j-1)<EDGE(i)
         g(k)=EDGE(i);
         indice1(kin)=k;
         g(k+1)=s(j);
         k=k+2;
         kin=kin+1;
```

```
            agg=1;
        end
        if s(j)<EDGE(i+1) & s(j+1)>EDGE(i+1)
            g(k)=s(j);
            g(k+1)=EDGE(i+1);
            indice1(kin)=k+1;
            k=k+2;
            kin=kin+1;
            agg=1;
        end
        if (s(j)==EDGE(i))|(s(j)==EDGE(i+1))
            indice1(kin)=k;
            kin=kin+1;
        end
    end
    if agg==0
        g(k)=s(j);
        k=k+1;
    end
end
g(k)=s(L);
indice1(kin)=k;
clear s
L=length(g);

indice2(1)=1;                    % costruzione della griglia nelle bande di transizione
kin=2;
if EDGE(1)>0
    bt=(0:1/(LGRID*r):EDGE(1));
    if bt(length(bt))<EDGE(1)
        bt(length(bt)+1)=EDGE(1);
    end
    indice2(kin)=length(bt);
    indice2(kin+1)=length(bt)+1;
    kin=kin+2;
else
    bt=[];
end
for j=1:NBANDS-1
    bt=[bt,(EDGE(2*j):1/(LGRID*r):EDGE(2*j+1))];
    if bt(length(bt))<EDGE(2*j+1)
        bt(length(bt)+1)=EDGE(2*j+1);
    end
    indice2(kin)=length(bt);
    indice2(kin+1)=length(bt)+1;
    kin=kin+2;
end
if EDGE(2*NBANDS)<1
    bt=[bt,(EDGE(2*NBANDS):1/(LGRID*r):1)];
    if bt(length(bt))<1
        bt(length(bt)+1)=1;
    end
    indice2(kin)=length(bt);
else
    indice2=indice2(1:kin-2);
end

j=fix(L/r);                      % fissiamo a caso r+1 frequenze
wi(1)=g(1);                      % di partenza considerando la continua
for i=2:r+1
    wi(i)=g((i-1)*j);
end

for j=2:r                        % inseriamo nelle frequenze estremali gli estremi di banda
    for i=1:NBANDS
        if wi(j)>EDGE(2*i-1) & wi(j-1)<EDGE(2*i-1)
            wi(j)=EDGE(2*i-1);
        elseif wi(j)<EDGE(2*i) & wi(j+1)>EDGE(2*i)
            wi(j)=EDGE(2*i);
        end
```

```
   end
end
wi(r+1)=1;

passi=0;
cambiamenti=1;                  % comincia il ciclo di interazioni
while cambiamenti~=0            % che termina quando non ci sono più
   cambiamenti=0;               % cambiamenti nelle r+1 frequenze
   passi=passi+1;

   D=interlin(f,filtro,wi,r,r+1);    % valori del filtro ideale nelle frequenze estremali

   for j=1:r
      COS(:,j)=(cos(wi*pi*(j-1)))';
   end
   for i=1:r+1
      COS(i,r+1)=((-1)^(i-1))/peso(wi(i),EDGE,WTX,NBANDS);
   end

   % i valori che il filtro ideale assume nei punti della grilia si
   % ottengono interpolando linearmente gli r punti di partenza :
   % per far questo usiamo la function interlin.m

   Y=COS\D;                     % risoluzione del sistema lineare
   X=Y(1:r);

   for i=1:L                    % valori del polinomio trigonometrico
      p1(i)=X(1);               % valutati nella griglia a partire
      for j=2:r                 % dalle r+1 frequenze estremali
         p1(i)=p1(i)+X(j)*cos(g(i)*pi*(j-1));
      end
   end

   for i=1:r+1                  % valori del polinomio trigonometrico
      p2(i)=X(1);               % nelle frequenze estremali
      for j=2:r
         p2(i)=p2(i)+X(j)*cos(wi(i)*pi*(j-1));
      end
   end

   for i=1:length(bt)           % valori del polinomio trigoniometrico
      p3(i)=X(1);               % nelle bande di transizione
      for j=2:r
         p3(i)=p3(i)+X(j)*cos(bt(i)*pi*(j-1));
      end
   end

   % disegno della risposta in frequenza del filtro del filtro ad ogni passo se richiesto

   strpassi=int2str(passi);
   if grafici==1
      figure
      w1=p1;
      w2=p2;
      w3=p3;
      if DB==1
         for i=1:length(w1)
            if abs(w1(i))<1e-5
               w1(i)=1e-5;
            end
         end
         for i=1:length(w2)
            if abs(w2(i))<1e-5
               w2(i)=1e-5;
            end
         end
         for i=1:length(w3)
            if abs(w3(i))<1e-5
               w3(i)=1e-5;
            end
```

```
        end
        w1=20*log10(abs(w1));
        w2=20*log10(abs(w2));
        w3=20*log10(abs(w3));
        ylabel('dB');
    end
    hold on
    for i=1:NBANDS
        plot(g(indice1(2*i-1):indice1(2*i)),w1(indice1(2*i-1):indice1(2*i)),'k');
    end
    for i=1:length(indice2)/2
        plot(bt(indice2(2*i-1):indice2(2*i)),w3(indice2(2*i-1):indice2(2*i)),'r');
    end
    grafico=plot(wi,w2,'.b');
    set(grafico,'markersize',10);
    set(gcf,'color',[1 1 1]);
    xlabel('f/(fc/2)');
    title(['risposta in frequenza del filtro al passo ',strpassi,' di interazione'])
    axis([0 1 min([w1 w3]) max([w1 w3])])
    zoom
    hold off
end

for i=1:L                                   % aggiornamento delle r+1 frequenze 'candidate'
                                            % ad essere le r+1 frequenze estremali
    w=peso(g(i),EDGE,WTX,NBANDS);
    d=interlin(f,filtro,g(i),r,1);
    E2=w*(d-p1(i));

    w=peso(wi(1),EDGE,WTX,NBANDS);
    d=interlin(f,filtro,wi(1),r,1);
    E1=w*(d-p2(1));

    w=peso(wi(r+1),EDGE,WTX,NBANDS);
    d=interlin(f,filtro,wi(r+1),r,1);
    E5=w*(d-p2(r+1));

    if g(i)<wi(1)
        if E2*E1>=0
            if abs(E2)>abs(E1)
                wi(1)=g(i);
                p2(1)=p1(i);
                cambiamenti=cambiamenti+1;
            end
        elseif abs(E2)>abs(E5)
            for k=r:-1:1
                wi(k+1)=wi(k);
                p2(k+1)=p2(k);
            end
            wi(1)=g(i);
            p2(1)=p1(i);
            cambiamenti=cambiamenti+1;
        end
    end

    j=1;
    sc=0;
    while j<=r & sc==0
        if g(i)>=wi(j) & g(i)<wi(j+1)

            w=peso(wi(j),EDGE,WTX,NBANDS);
            d=interlin(f,filtro,wi(j),r,1);
            E3=w*(d-p2(j));

            w=peso(wi(j+1),EDGE,WTX,NBANDS);
            d=interlin(f,filtro,wi(j+1),r,1);
            E4=w*(d-p2(j+1));

            if E2*E3>=0
                if abs(E2)>abs(E3)
```

```
                    wi(j)=g(i);
                    p2(j)=p1(i);
                    sc=1;
                    cambiamenti=cambiamenti+1;
                 else
                    break
                 end
              else
                 if abs(E2)>abs(E4)
                    wi(j+1)=g(i);
                    p2(j+1)=p1(i);
                    sc=1;
                    cambiamenti=cambiamenti+1;
                 else
                    break
                 end
              end
           else
              j=j+1;
           end
        end

        if g(i)>=wi(r+1)
           if E2*E5>=0
              if abs(E2)>abs(E5)
                 wi(r+1)=g(i);
                 p2(r+1)=p1(i);
                 cambiamenti=cambiamenti+1;
              end
           elseif abs(E2)>abs(E1)
              for k=1:r
                 wi(k)=wi(k+1);
                 p2(k)=p2(k+1);
              end
              wi(r+1)=g(i);
              p2(r+1)=p1(i);
              cambiamenti=cambiamenti+1;
           end
        end
     end      % fine dell'aggiornamento delle r+1 frequenze estremali 'candidate'

     strcambi=int2str(cambiamenti);
     disp(['il numero di cambi effettuati al passo ',strpassi,' sono : ',strcambi]);

end         % fine della ricerca delle frequenze estremali


%                          COSTRUZIONE DEI GRAFICI FINALI

if grafici==1 & DB==1
   title('risposta in frequenza del filtro all''ultimo passo di iterazione in dB')
elseif grafici==1 & DB~=1
   title('risposta in frequenza del filtro all''ultimo passo di iterazione')
end
w1=p1;
w2=p2;
w3=p3;
for i=1:length(w1)
   if abs(w1(i))<1e-5
      w1(i)=1e-5;
   end
end
for i=1:length(w2)
   if abs(w2(i))<1e-5
      w2(i)=1e-5;
   end
end
for i=1:length(w3)
   if abs(w3(i))<1e-5
      w3(i)=1e-5;
```

```
   end
end
w1=20*log10(abs(w1));
w2=20*log10(abs(w2));
w3=20*log10(abs(w3));
if (grafici~=1) | (grafici==1 & DB==1)
   figure
   hold on
   for i=1:NBANDS
      plot(g(indice1(2*i-1):indice1(2*i)),p1(indice1(2*i-1):indice1(2*i)),'k');
   end
   for i=1:length(indice2)/2
      plot(bt(indice2(2*i-1):indice2(2*i)),p3(indice2(2*i-1):indice2(2*i)),'r');
   end
   grafico=plot(wi,p2,'.b');
   set(grafico,'markersize',10);
   set(gcf,'color',[1 1 1]);
   xlabel('f/(fc/2)');
   title('risposta in frequenza del filtro all''ultimo passo di iterazione')
   axis([0 1 min([p1 p3]) max([p1 p3])])
   zoom
   hold off
end
if (grafici~=1) | (grafici==1 & DB~=1)
   figure
   hold on
   for i=1:NBANDS
      plot(g(indice1(2*i-1):indice1(2*i)),w1(indice1(2*i-1):indice1(2*i)),'k');
   end
   for i=1:length(indice2)/2
      plot(bt(indice2(2*i-1):indice2(2*i)),w3(indice2(2*i-1):indice2(2*i)),'r');
   end
   grafico=plot(wi,w2,'.b');
   set(grafico,'markersize',10);
   set(gcf,'color',[1 1 1]);
   xlabel('f/(fc/2)');
   ylabel('dB');
   title('risposta in frequenza del filtro all''ultimo passo di iterazione in dB')
   axis([0 1 min([w1 w3]) max([w1 w3])])
   zoom
   hold off
end

disp(['In totale i passi di iterazione sono: ',strpassi]);
strdelta=num2str(abs(Y(r+1)));
disp(['l''ampiezza massima delle oscillazioni normalizzata è : ',strdelta]);

% GRAFICI DELLA RISPOSTA IN FREQUENZA DEL FILTRO MODIFICANDO
% LA RISOLUZIONE DEI COEFFICIENTI DEL POLINOMIO TRIGONIOMETRICO

disp('se si desidera modificare la risoluzione dei coefficienti del polinomio');
risp=input('trigoniometrico digitare ''1'' : ');
if risp==1
   figure
   hold on
   bit=4;
   colori=['k','r','b'];
   for ris=1:3
      for i=1:r
         xi(i)=floor((X(i)/max(X))*(2^bit-1))*(max(X)/(2^bit-1));
      end
      for i=1:L                % valori del polinomio trigonometrico
         p1(i)=xi(1);           % valutati nella griglia a partire
         for j=2:r              % dalle r+1 frequenze estremali
            p1(i)=p1(i)+xi(j)*cos(g(i)*pi*(j-1));
         end
      end
      for i=1:length(bt)       % valori del polinomio trigoniometrico
         p3(i)=xi(1);           % nelle bande di transizione
         for j=2:r
```

```
            p3(i)=p3(i)+xi(j)*cos(bt(i)*pi*(j-1));
         end
      end
      for i=1:length(p1)
         if abs(p1(i))<1e-5
            p1(i)=1e-5;
         end
      end
      for i=1:length(p3)
         if abs(p3(i))<1e-5
            p3(i)=1e-5;
         end
      end
      p1=20*log10(abs(p1));
      p3=20*log10(abs(p3));
      if EDGE(1)>0
         gr=bt(indice2(1):indice2(2));
         w=p3(indice2(1):indice2(2));
         j=3;
      else
         gr=[];
         w=[];
         j=1;
      end
      for i=0:NBANDS-2
         gr=[gr,g(indice1(2*i+1):indice1(2*i+2)),bt(indice2(2*i+j):indice2(2*i+j+1))];
         w=[w,p1(indice1(2*i+1):indice1(2*i+2)),p3(indice2(2*i+j):indice2(2*i+j+1))];
      end
      gr=[gr,g(indice1(2*NBANDS-1):indice1(2*NBANDS))];
      w=[w,p1(indice1(2*NBANDS-1):indice1(2*NBANDS))];
      if EDGE(2*NBANDS)<1
         gr=[gr,bt(length(indice2)-1:length(indice2))];
         w=[w,p3(indice2(length(indice2)-1):indice2(length(indice2)))];
      end
      plot(gr,w,colori(ris));
      set(gcf,'color',[1 1 1]);
      xlabel('f/(fc/2)');
      ylabel('dB');
      title(['risposta in frequenza del filtro con le seguenti risoluzioni'])
      zoom
      bit=bit*2;
   end
   legend('4 bit','8 bit','16 bit');
end
h(r)=X(1);
for i=2:r
   h(r-i+1)=X(i)/2;
end
for j=r+1:2*r-1
   h(j)=h(2*r-j);
end
disp('La risposta all''impulso del filtro è contenuta nel vettore "h" ');
h=h';
```

peso.m

```
function sol=peso(x,EDGE,WTX,NBANDS);

%in sol verrà individuato il valore della funzione peso, che si
%trova nel vettore WTX, in corrispondenza della banda in cui si trova x
%x quindi rappresenta un elemento della griglia

for j=1:NBANDS
   if x>=EDGE(2*j-1) & x<=EDGE(2*j)
      sol=WTX(j);
   end
end
```

interlin.m

```
% INTERPOLAZIONE LINEARE

function s=interlin(x,y,xi,n,m);

% s   vettore soluzione dei punti da interpolare
% x   vettore dei nodi interpolanti
% y   valori della funzione nei nodi x
% xi  vettore dei punti in cui calcolare la funzione interpolante
% n   dimensione di x , y
% m   dimensione di xi

for k=1:m
    s(k)=0;
    if xi(k)<=x(2) & xi(k)>=x(1)
       s(k)=s(k)+((xi(k)-x(2))/(x(1)-x(2)))*y(1);
    end
    for i=2:n-1
        if xi(k)>=x(i-1) & xi(k)<x(i)
           s(k)=s(k)+((xi(k)-x(i-1))/(x(i)-x(i-1)))*y(i);
        end
        if xi(k)>=x(i) & xi(k)<=x(i+1)
           s(k)=s(k)+((xi(k)-x(i+1))/(x(i)-x(i+1)))*y(i);
        end
    end
    if xi(k)>=x(n-1) & xi(k)<=x(n)
       s(k)=s(k)+((xi(k)-x(n-1))/(x(n)-x(n-1)))*y(n);
    end
end
s=s';
```

basso.m

```
function [f,filtro,EDGE,WTX]=basso

f=[0:1/6:1];
filtro=[ones(1,3) zeros(1,4)];
EDGE=[0,1/3,0.5,1];
WTX=[10,1];
```

banda.m

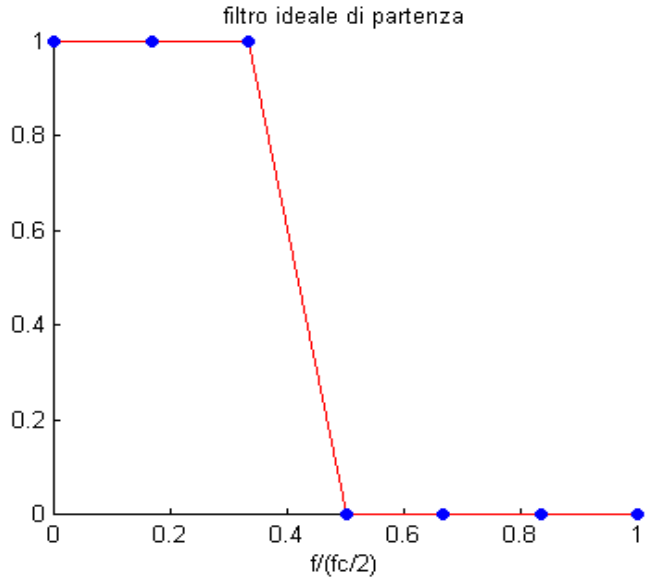
```
function [f,filtro,EDGE,WTX]=banda

f=[0:1/10:1];
filtro=[zeros(1,3) ones(1,5) zeros(1,3)];
EDGE=[0,0.2,0.3,0.7,0.8,1];
WTX=[1,10,1];
```

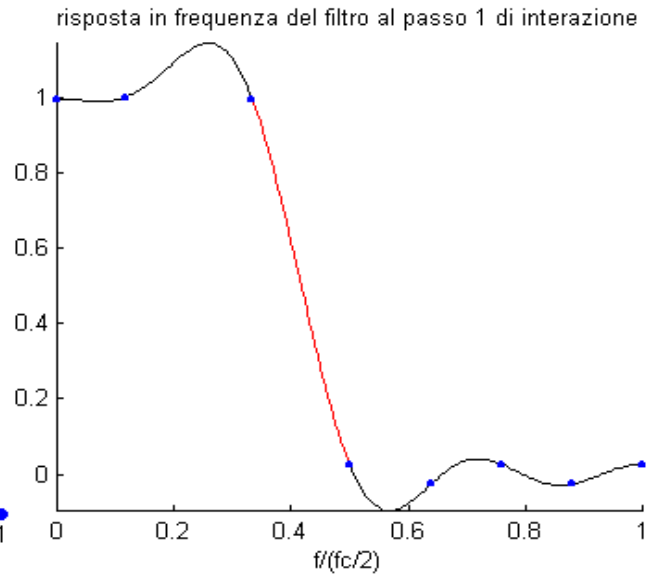
bassopk.m

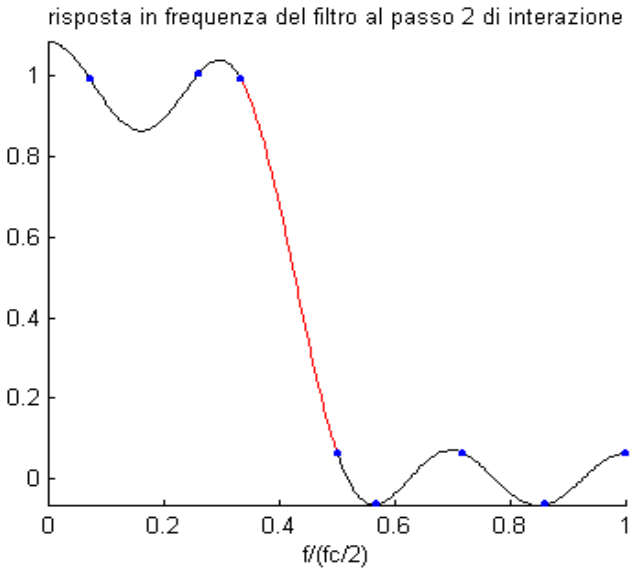
```
function [f,filtro,EDGE,WTX]=bassopk

f=[(0:1/30:0.2) (0.3:7/230:1)];
filtro=[ones(1,7) zeros(1,24)];
EDGE=[0,0.2,0.3,1];
WTX=[1,1];
```

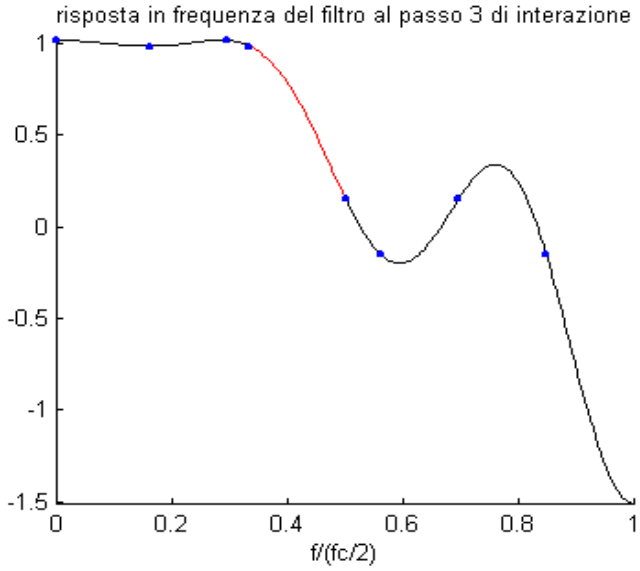
esecuzione del programma

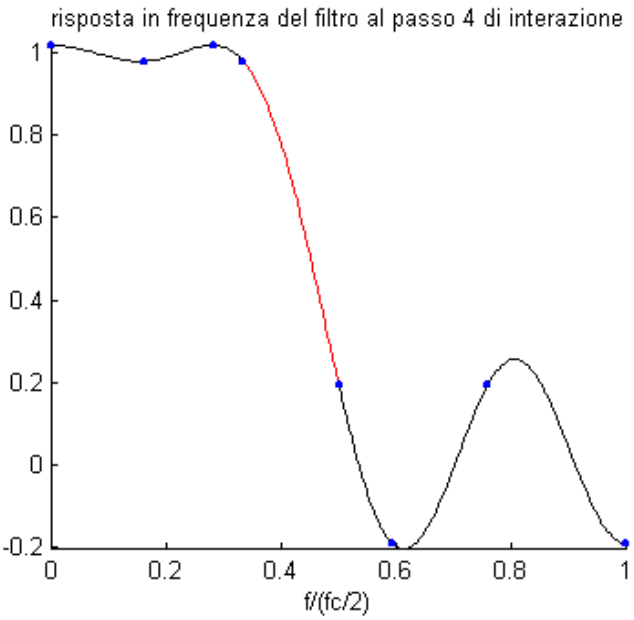
```
» remez
inserisci il nome della function contenente il filtro ideale da realizzare: basso
introduci il valore della densità della griglia per calcolare E(w) : 32
Per disegnare i grafici ad ogni passo di interazione digitare "1" : 1
Per disegnare i grafici in dB digitare "1" : 5
Il numero di campioni del filtro da realizzare sono: 7
il numero di cambi effettuati al passo 1 sono : 74
il numero di cambi effettuati al passo 2 sono : 40
il numero di cambi effettuati al passo 3 sono : 58
il numero di cambi effettuati al passo 4 sono : 16
il numero di cambi effettuati al passo 5 sono : 2
il numero di cambi effettuati al passo 6 sono : 0
In totale i passi di iterazione sono: 6
l'ampiezza massima delle oscillazioni normalizzata è : 0.19985
se si desidera modificare la risoluzione dei coefficienti del polinomio
trigoniometrico digitare '1' : 1
La risposta all'impulso del filtro è contenuta nel vettore "h"
» h(7:13)
ans =
    0.4368
    0.3098
    0.0610
   -0.0834
   -0.0659
    0.0786
   -0.0084
```

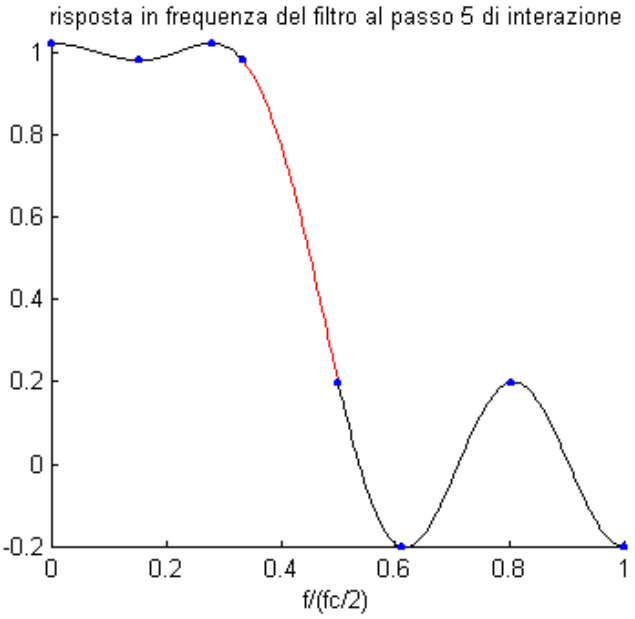


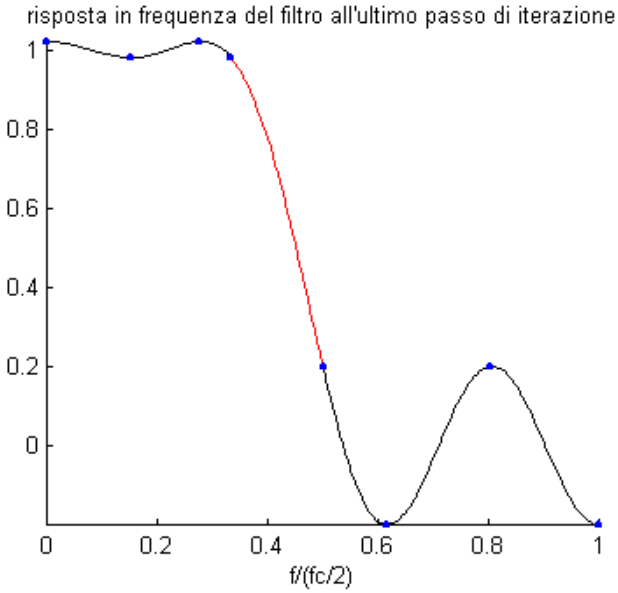


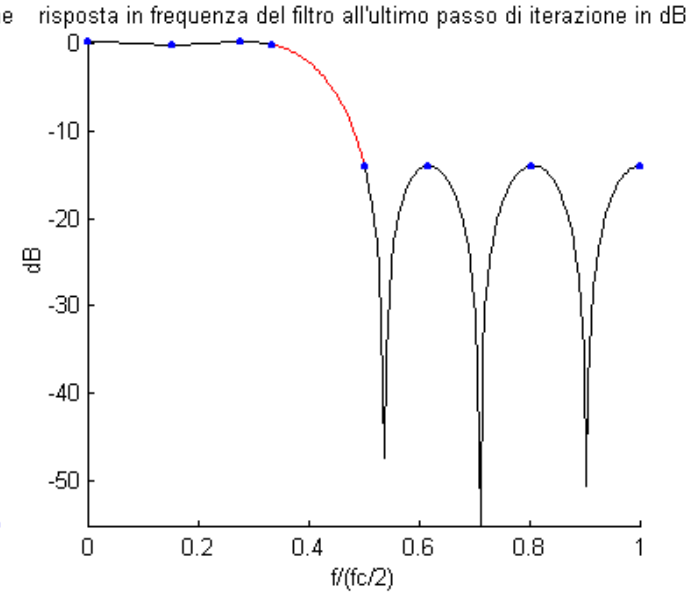


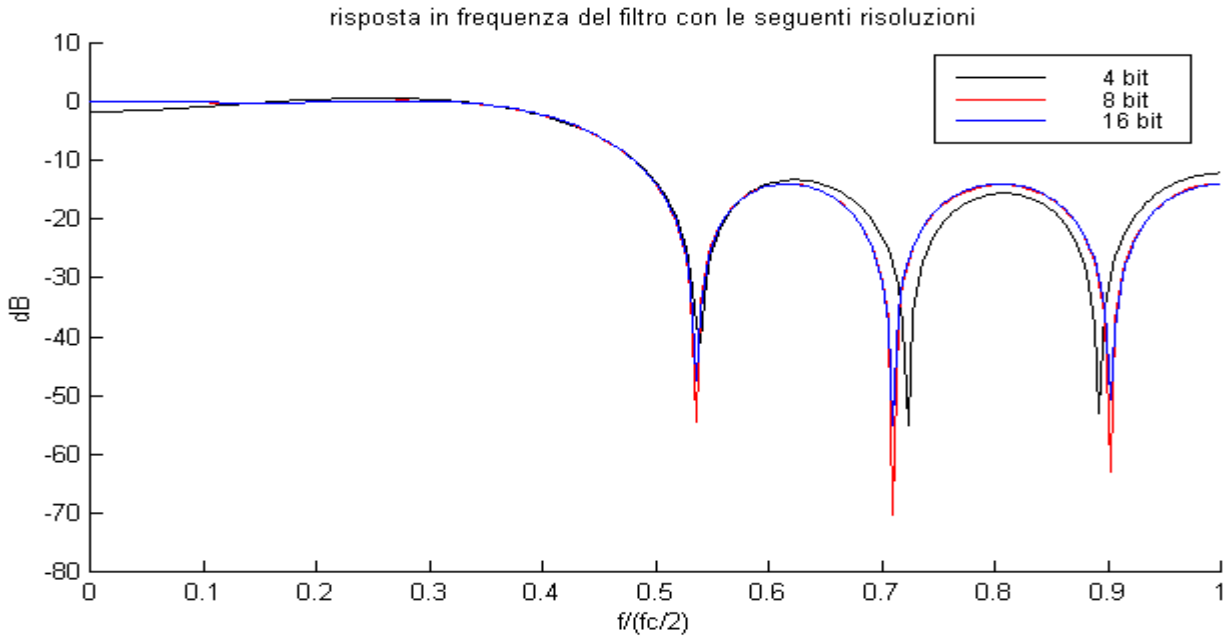



risposta in frequenza del filtro al passo 4 di interazione
f/(fc/2)

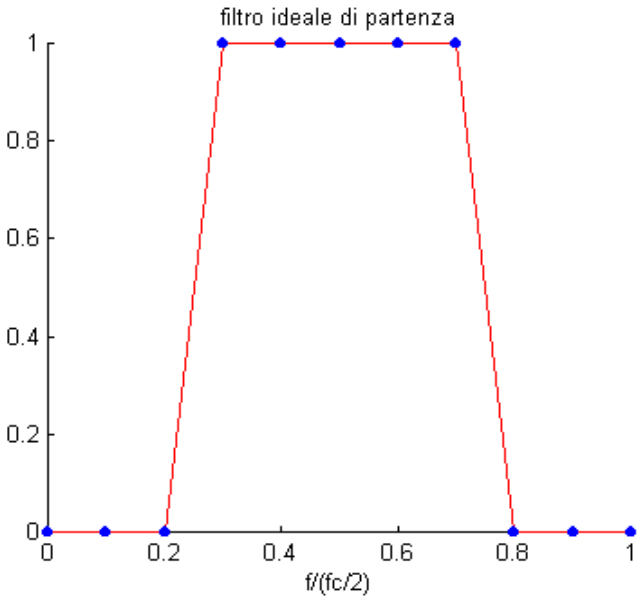

risposta in frequenza del filtro al passo 5 di interazione
f/(fc/2)

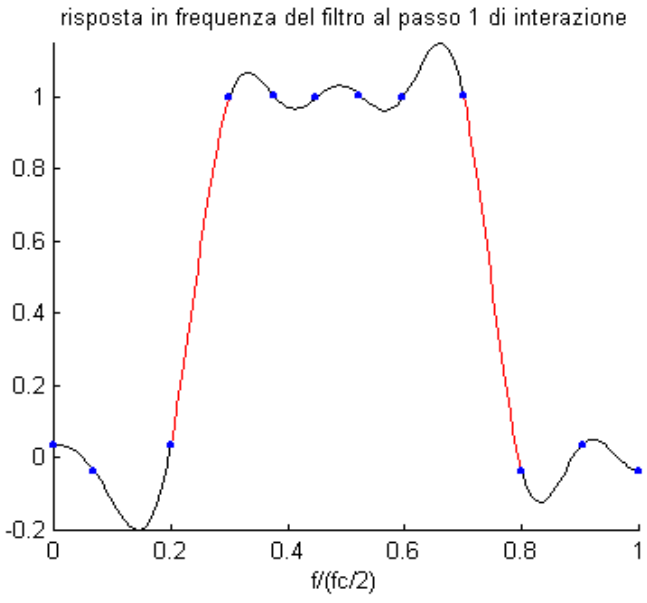

risposta in frequenza del filtro all'ultimo passo di iterazione
f/(fc/2)

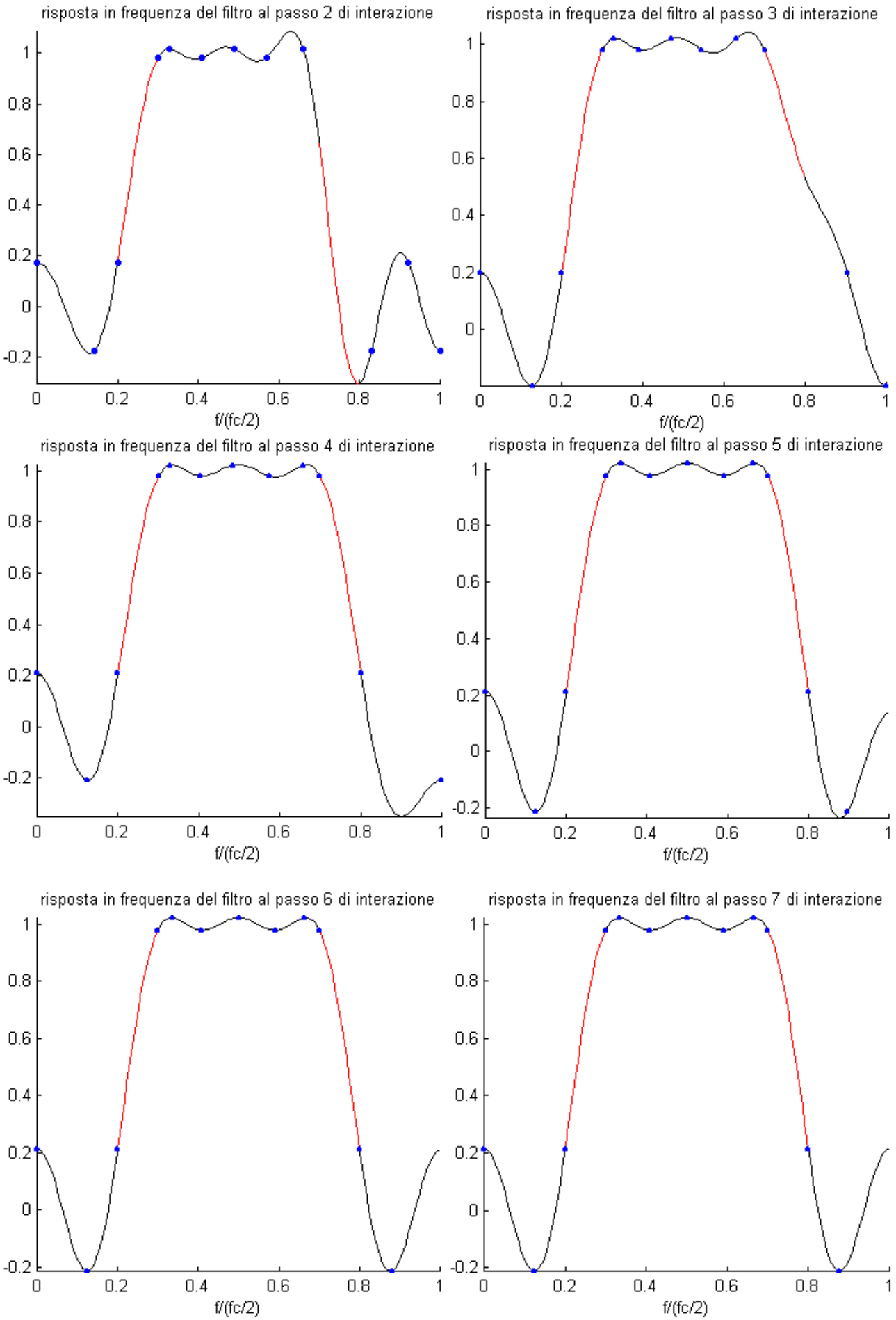

risposta in frequenza del filtro all'ultimo passo di iterazione in dB
dB
f/(fc/2)

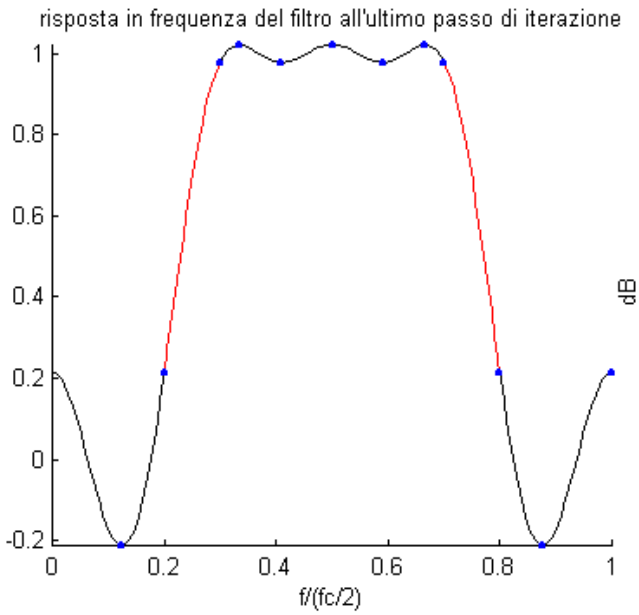

risposta in frequenza del filtro con le seguenti risoluzioni
4 bit
8 bit
16 bit
dB
f/(fc/2)

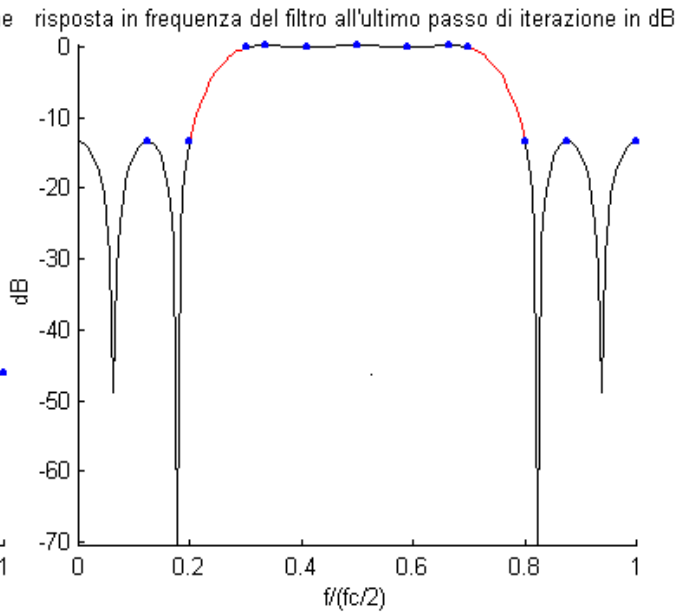
```
» remez

inserisci il nome della function contenente il filtro ideale da realizzare: banda
introduci il valore della densità della griglia per calcolare E(w) : 16
Per disegnare i grafici ad ogni passo di interazione digitare "1" : 1
Per disegnare i grafici in dB digitare "1" : 5
Il numero di campioni del filtro da realizzare sono: 11
il numero di cambi effettuati al passo 1 sono : 59
il numero di cambi effettuati al passo 2 sono : 34
il numero di cambi effettuati al passo 3 sono : 16
il numero di cambi effettuati al passo 4 sono : 19
il numero di cambi effettuati al passo 5 sono : 5
il numero di cambi effettuati al passo 6 sono : 2
il numero di cambi effettuati al passo 7 sono : 1
il numero di cambi effettuati al passo 8 sono : 0
In totale i passi di iterazione sono: 8
l'ampiezza massima delle oscillazioni normalizzata è : 0.21304
se si desidera modificare la risoluzione dei coefficienti del polinomio
trigoniometrico digitare '1' : 1
La risposta all'impulso del filtro è contenuta nel vettore "h"

» h

ans =

-1.7924e-003
 -9.0585e-017
  6.6311e-002
 -3.3682e-016
  1.1425e-001
 -2.9353e-016
 -1.9440e-002
 -3.3627e-016
 -3.1452e-001
 -3.7510e-016
  5.2343e-001
 -3.7510e-016
 -3.1452e-001
 -3.3627e-016
 -1.9440e-002
 -2.9353e-016
  1.1425e-001
 -3.3682e-016
  6.6311e-002
 -9.0585e-017
 -1.7924e-003
```

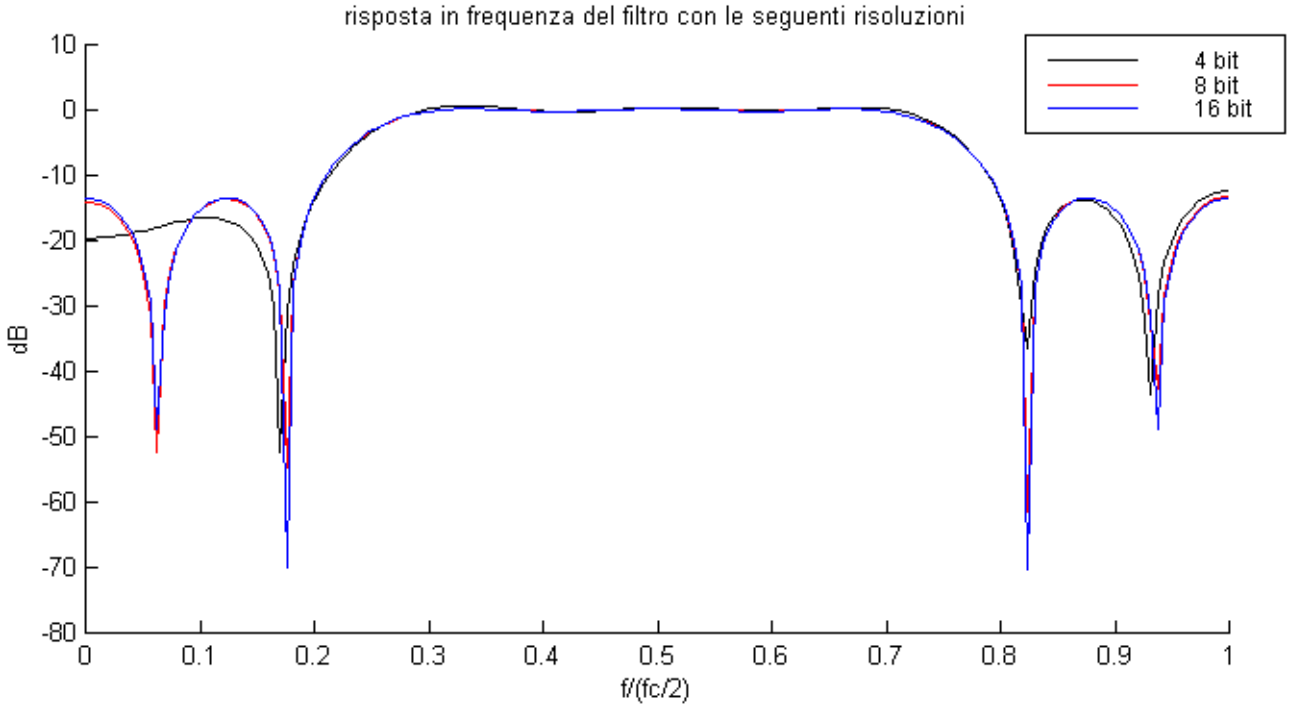




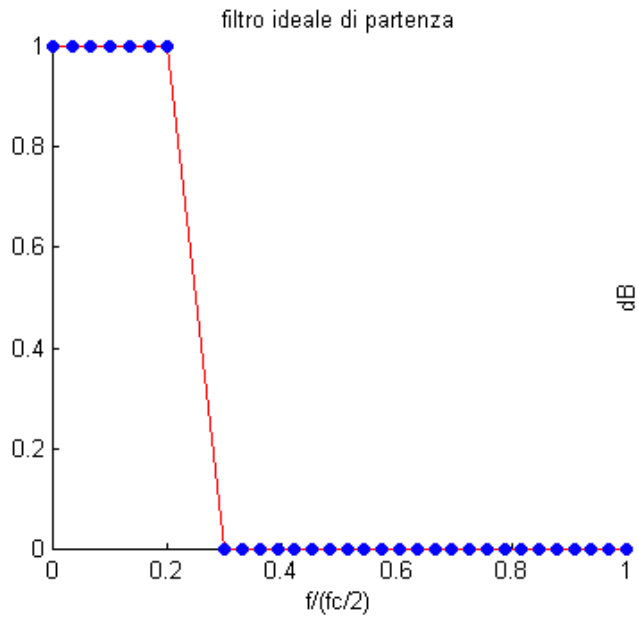

risposta in frequenza del filtro al passo 2 di interazione
risposta in frequenza del filtro al passo 3 di interazione
risposta in frequenza del filtro al passo 4 di interazione
risposta in frequenza del filtro al passo 5 di interazione
risposta in frequenza del filtro al passo 6 di interazione
risposta in frequenza del filtro al passo 7 di interazione
f/(fc/2)

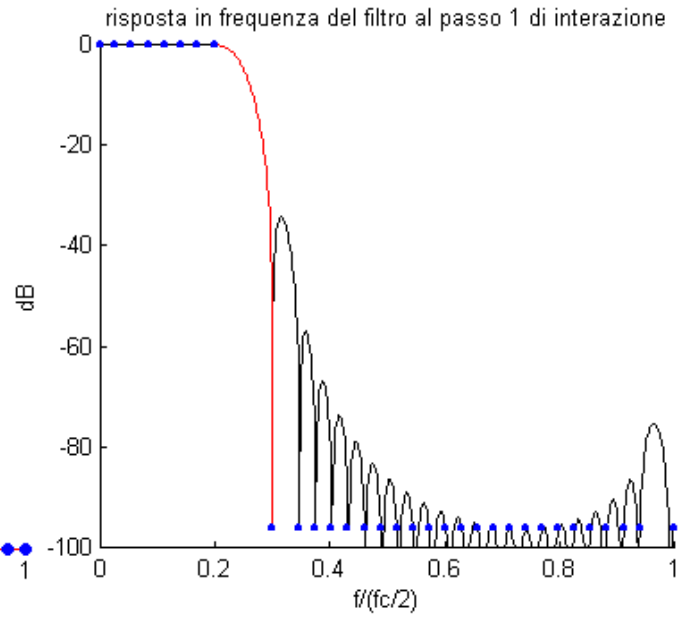

risposta in frequenza del filtro all'ultimo passo di iterazione
f/(fc/2)

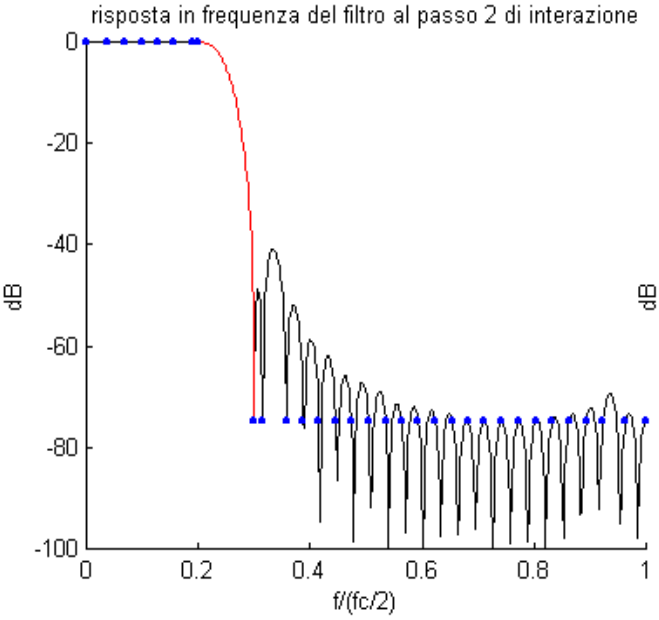

risposta in frequenza del filtro all'ultimo passo di iterazione in dB
dB
f/(fc/2)

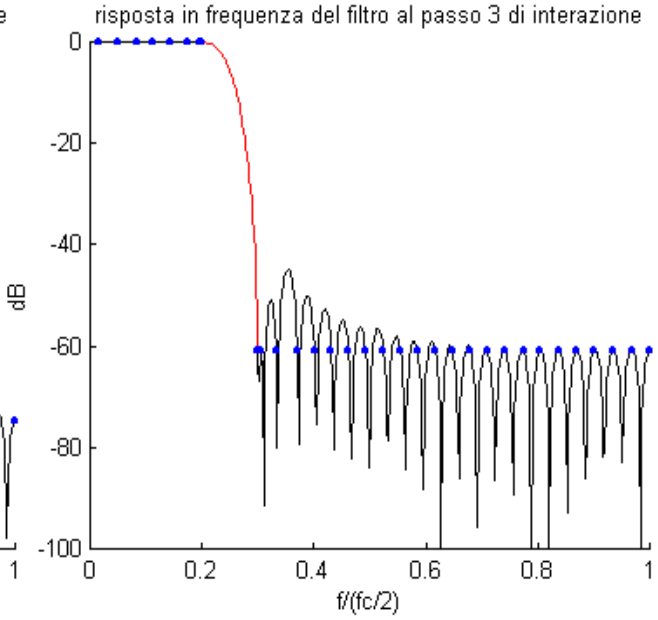

risposta in frequenza del filtro con le seguenti risoluzioni
4 bit
8 bit
16 bit
dB
f/(fc/2)

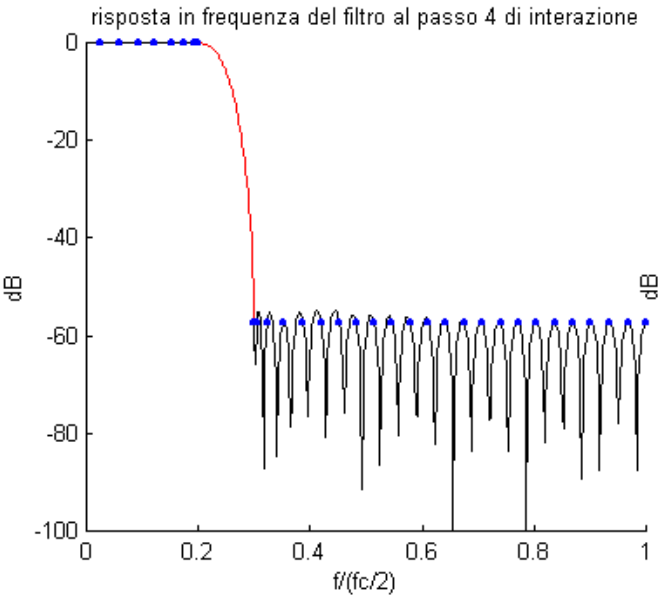
```
» remez

inserisci il nome della function contenente il filtro ideale da realizzare: bassopk

introduci il valore della densità della griglia per calcolare E(w) : 16

Per disegnare i grafici ad ogni passo di interazione digitare "1" : 1

Per disegnare i grafici in dB digitare "1" : 1

Il numero di campioni del filtro da realizzare sono: 31

il numero di cambi effettuati al passo 1 sono : 151
il numero di cambi effettuati al passo 2 sono : 136
il numero di cambi effettuati al passo 3 sono : 95
il numero di cambi effettuati al passo 4 sono : 59
il numero di cambi effettuati al passo 5 sono : 17
il numero di cambi effettuati al passo 6 sono : 3
il numero di cambi effettuati al passo 7 sono : 0

In totale i passi di iterazione sono: 7

l'ampiezza massima delle oscillazioni normalizzata è : 0.0015511

se si desidera modificare la risoluzione dei coefficienti del polinomio
trigoniometrico digitare '1' : 1

La risposta all'impulso del filtro è contenuta nel vettore "h"

» h(31:61)

ans =

  2.5008e-001
  2.2466e-001
  1.5782e-001
  7.3567e-002
 -5.9753e-005
 -4.2721e-002
 -4.9118e-002
 -2.8896e-002
  7.8330e-005
  2.1064e-002
  2.5657e-002
  1.5707e-002
 -3.8482e-005
 -1.1964e-002
 -1.4722e-002
 -9.0350e-003
  7.7435e-005
  6.9573e-003
  8.4873e-003
  5.1750e-003
  8.3595e-006
 -3.7732e-003
 -4.5128e-003
 -2.6433e-003
  1.0050e-004
  1.9972e-003
  2.2953e-003
  1.3515e-003
  9.6359e-005
 -6.7383e-004
 -1.2102e-003
```

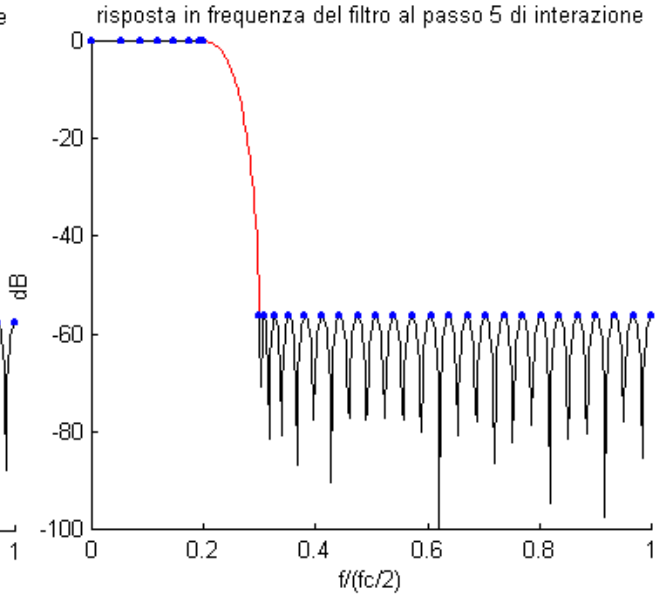

filtro ideale di partenza
f/(fc/2)

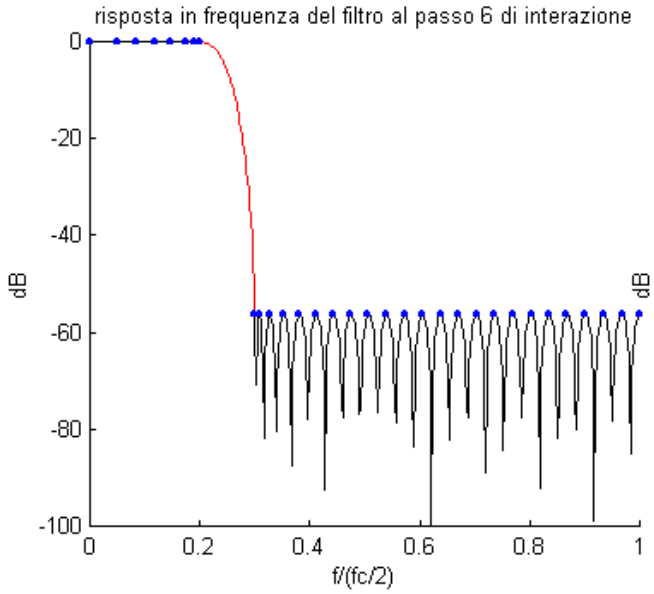

risposta in frequenza del filtro al passo 1 di interazione
dB
f/(fc/2)

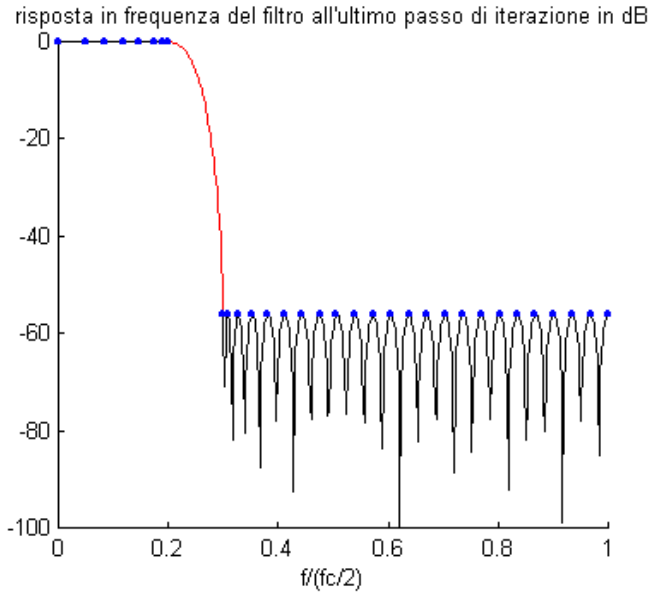

risposta in frequenza del filtro al passo 2 di interazione
dB
f/(fc/2)

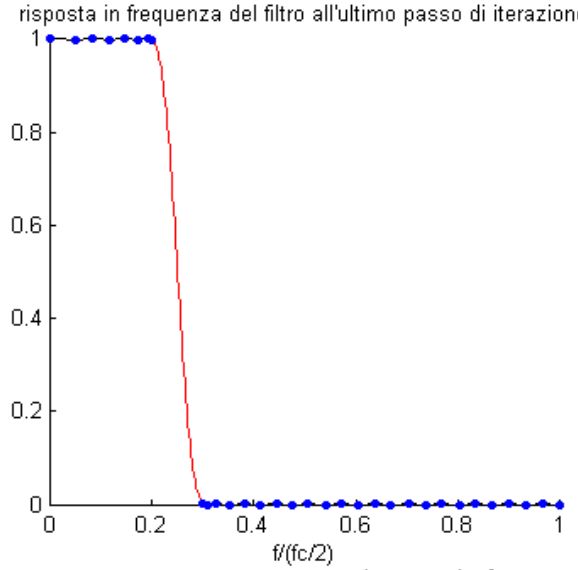

risposta in frequenza del filtro al passo 3 di interazione
dB
f/(fc/2)

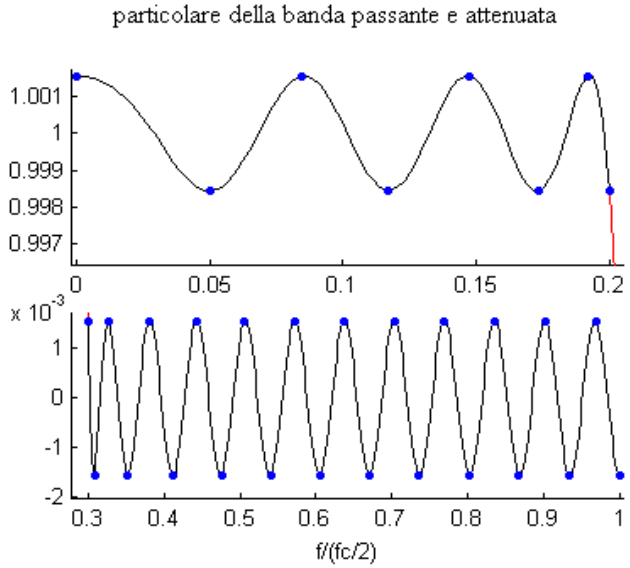

risposta in frequenza del filtro al passo 4 di interazione
dB
f/(fc/2)

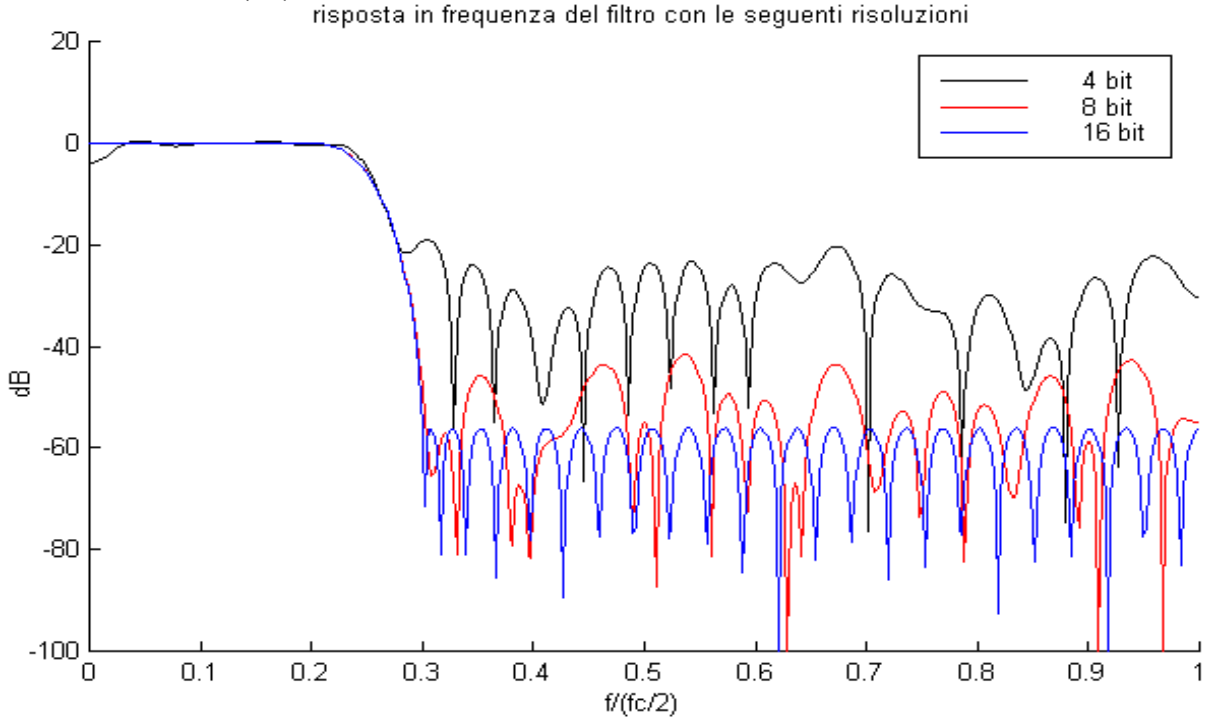

risposta in frequenza del filtro al passo 5 di interazione
dB
f/(fc/2)


risposta in frequenza del filtro al passo 6 di interazione
dB
f/(fc/2)


risposta in frequenza del filtro all'ultimo passo di iterazione in dB
dB
f/(fc/2)


risposta in frequenza del filtro all'ultimo passo di iterazione
f/(fc/2)


particolare della banda passante e attenuata
x 10^-3
f/(fc/2)


risposta in frequenza del filtro con le seguenti risoluzioni
4 bit
8 bit
16 bit
dB
f/(fc/2)

Con questo programma verrà stimata la densità spettrale di potenza di un generico processo stocastico utilizzando l'analisi non parametrica ed in particolare valuteremo la densità spettrale di potenza del segnale vocale.
Il procedimento seguito consiste nel memorizzare dei file audio (in formato wav) in corrispondenza di particolari parole che ogni volta nell'esecuzione del programma saranno specificate e successivamente saranno convertite ed elaborate in MATLAB (con il file analisis.m).
La stima della densità spettrale sarà ottenuta con il metodo del periodogramma e di Blackman-Tukey.

Analisis.m

```
% STIMA DELLA DENSITA' SPETTRALE DI POTENZA DI UN PROCESSO STOCASTICO

% come processo verrà considerato un generico segnale vocale introdotto dall'esterno

clear all

err=0;
while err==0
   F=input('inserisci il nome del file contenente il segnale audio (formato wav): ','s');
   F=[F,'.wav'];
   if exist(F)==0
      disp('non digitare l''estensione del file oppure il file non esiste ');
   else
      err=1;
   end
end
[y,fs]=wavread(F);
x=((1:length(y))-1)/fs;
N=length(y);
strN=int2str(N);
strfs=int2str(fs);
disp(['Il numero di campioni del file audio considerato sono : ',strN]);
disp(['La frequenza di campionamento del file audio è di ',strfs,' Hz']);
err=0;
while err==0
   deltaf=input('introdurre la massima risoluzione spettrale desiderata (in Hz): ');
   M=floor(fs/deltaf);
   if M>=N
      disp('La risoluzione spettrale richiesta è troppo elevata! ');
   else
      err=1;
   end
end
strd=int2str(deltaf);
sound(y,fs)
figure
plot(x,y)
xlabel('sec')
set(gcf,'color',[1 1 1]);
title('andamento del processo nel tempo ')
K=floor(N/M);

% METODO DEL PERIODOGRAMMA

sp1=0;
for i=1:K
   sp1=sp1+(abs(fft(y(1+(i-1)*M:i*M)))).^2;
end
sp1=sp1/(K*M);
strmetodo1='del Periodogramma';

% METODO DI BLACKMAN-TUKEY
sp2=0;
for i=1:K
   autcorr=xcorr(y(1+(i-1)*M:i*M),(y(1+(i-1)*M:i*M)));
   autcorr(length(autcorr)+1)=0;
```

```
    for j=1:M
        fin(j)=(j-1)/M;
        fin(2*M-j)=(j-1)/M;
    end
    fin(2*M)=0;
    autcorr=autcorr.*fin';
    sp2=sp2+abs(fft(autcorr));
end
sp2=sp2/(K*M);
strmetodo2='di Blackman-Tukey';
M2=2*M;

freq1=(0:fs/M:fs/2);
figure
plot(freq1,sp1(1:length(freq1)))
xlabel('f [Hz]')
title(['Densità spettrale di potenza (risoluzione max di ',strd,' Hz) con il metodo
',strmetodo1])
freq2=(0:fs/M2:fs/2);
set(gcf,'color',[1 1 1]);
figure
plot(freq2,sp2(1:length(freq2)))
xlabel('f [Hz]')
title(['Densità spettrale di potenza (risoluzione max di ',strd,' Hz) con il metodo
',strmetodo2])
set(gcf,'color',[1 1 1]);
figure
plot(freq1,sp1(1:length(freq1)),'k',freq2,sp2(1:length(freq2)),'r')
xlabel('f [Hz]')
title(['Densità spettrale di potenza (risoluzione max di ',strd,' Hz) con il metodo :'])
legend(strmetodo1,strmetodo2)
set(gcf,'color',[1 1 1]);

strK=int2str(K);
disp(['Il vettore audio è stato suddiviso in ',strK,' parti'])
```

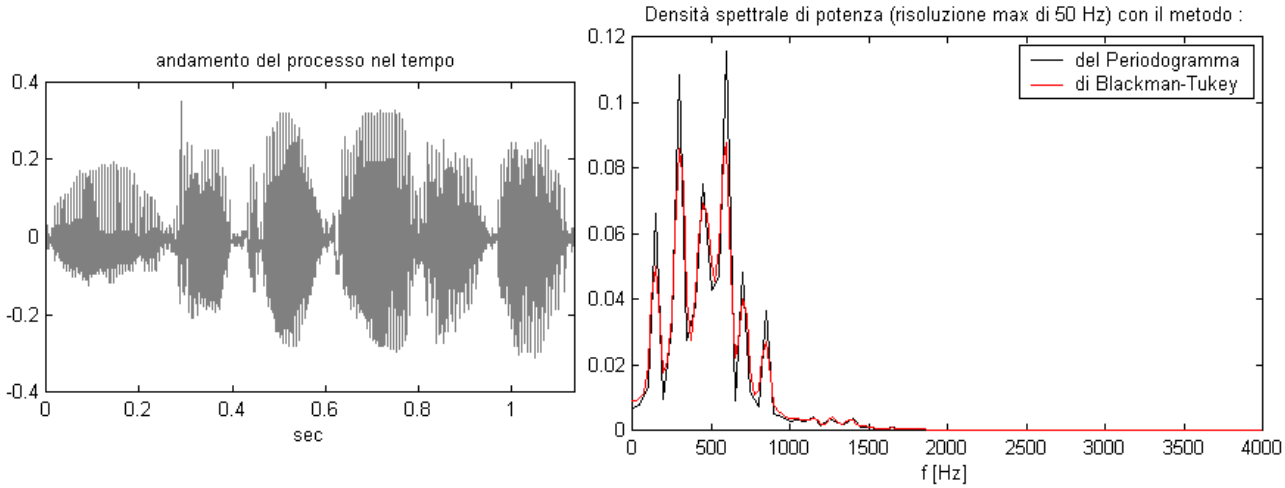
File sonori utilizzati

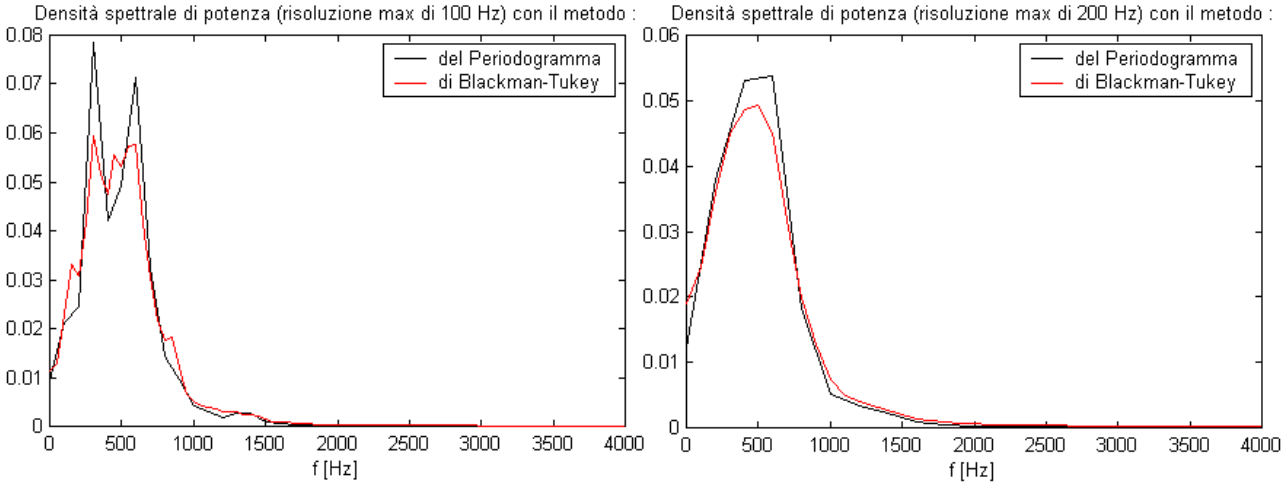
1) wave.wav : “ 1 2 3 PROVA SSSA ”
2) a.wav : “ A “
3) aiuola.wav : “ AIUOLA “
4) area.wav : “ AREA “

esecuzione del programma

1) Suono vocale corrispondente alle parole : " **UNO DUE TRE PROVA SSSA** " (sono stati eliminati gli spazi tra le parole).

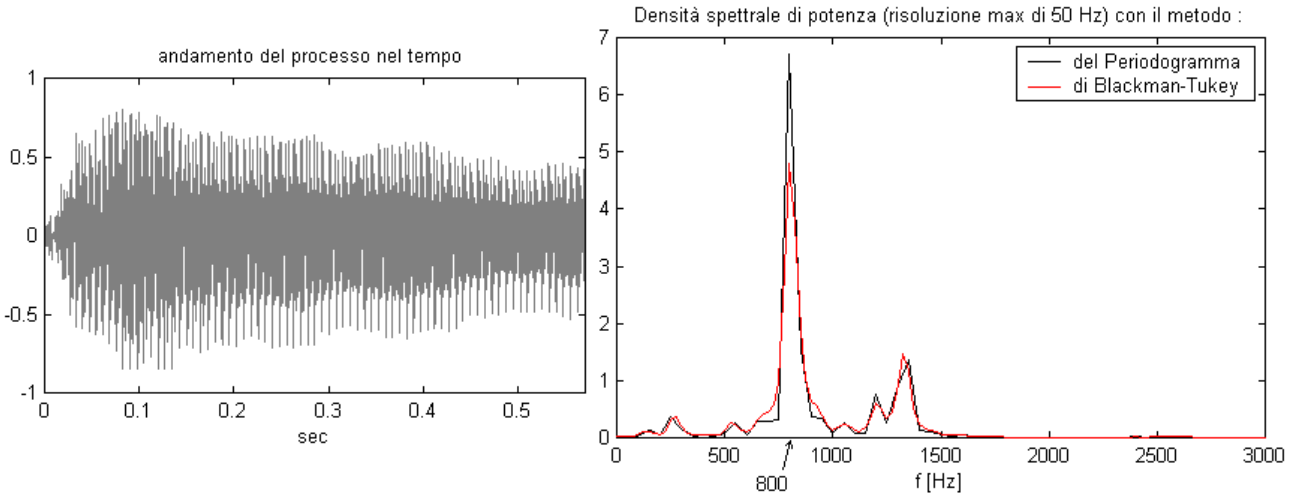
```
» analisis

inserisci il nome del file contenente il segnale audio (formato wav): wave
Il numero di campioni del file audio considerato sono : 9077
La frequenza di campionamento del file audio è di 8000 Hz

introdurre la massima risoluzione spettrale desiderata (in Hz): 50   (e successivamente : 100, 200)

Il vettore audio è stato suddiviso in 56 parti   ( 113 parti per 100 Hz ; 226 parti per 200 Hz )
```

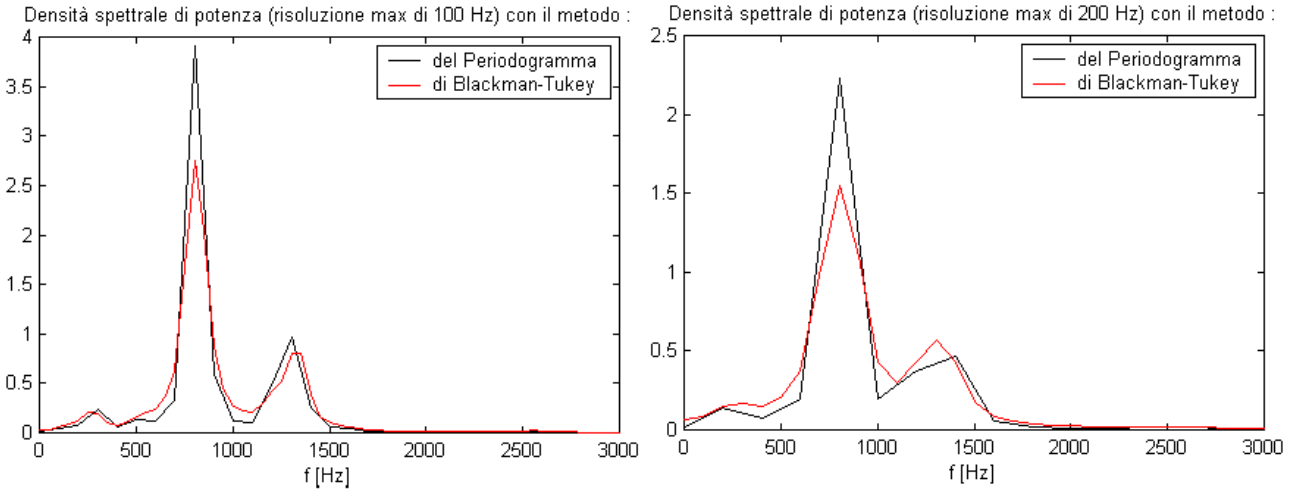




2) Suono vocale corrispondente alla lettera : " **A** " .

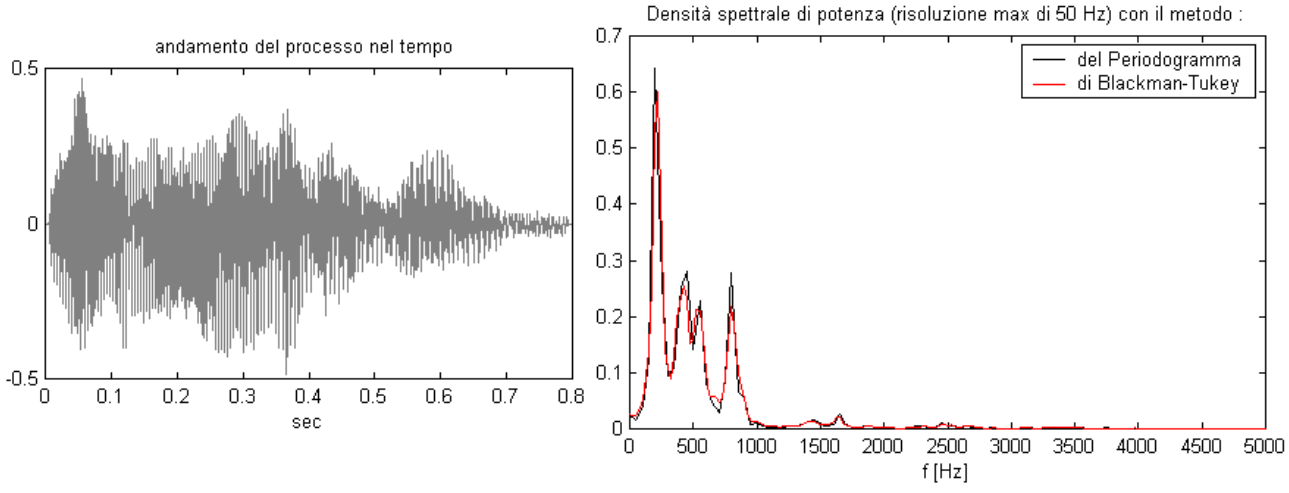
```
» analisis
inserisci il nome del file contenente il segnale audio (formato wav): a
Il numero di campioni del file audio considerato sono : 12601
La frequenza di campionamento del file audio è di 22050 Hz

introdurre la massima risoluzione spettrale desiderata (in Hz): 50   (e successivamente : 100, 200)

Il vettore audio è stato suddiviso in 28 parti   ( 57 parti per 100 Hz ; 114 parti per 200 Hz )
```

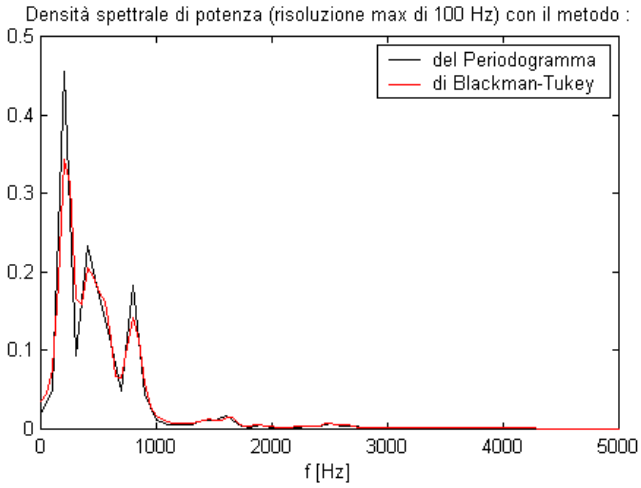


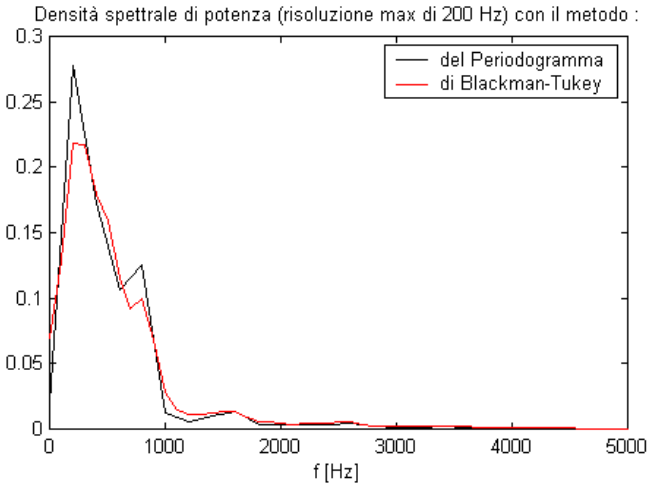


3) Suono vocale corrispondente alla parola : " **AIUOLA** " .

```
» analisis

inserisci il nome del file contenente il segnale audio (formato wav): aiuola
Il numero di campioni del file audio considerato sono : 17484
La frequenza di campionamento del file audio è di 22050 Hz

introdurre la massima risoluzione spettrale desiderata (in Hz): 50   (e successivamente : 100, 200)

Il vettore audio è stato suddiviso in 39 parti   ( 79 parti per 100 Hz ; 158 parti per 200 Hz )
```

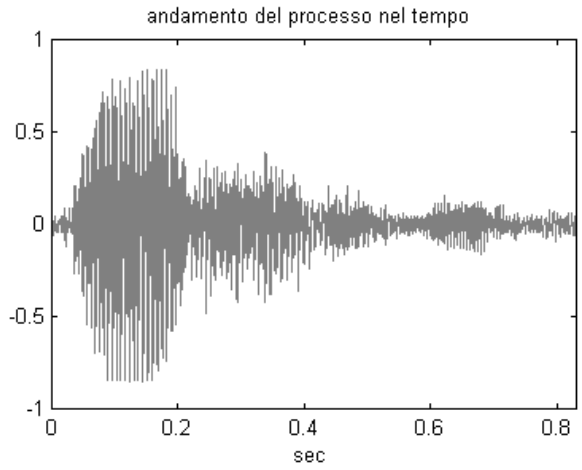


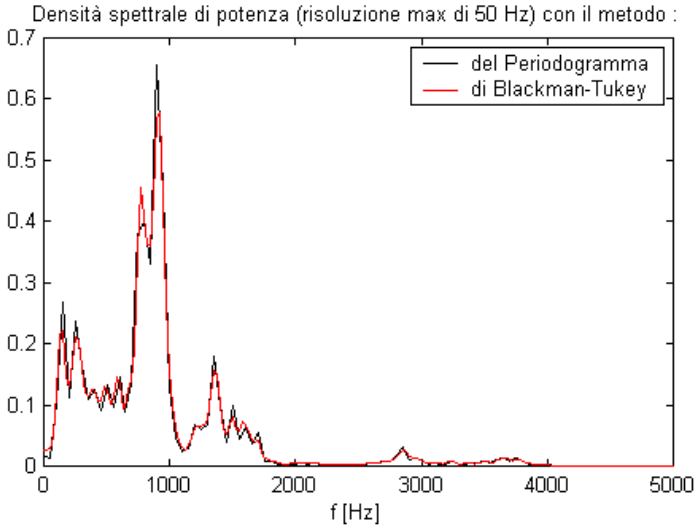


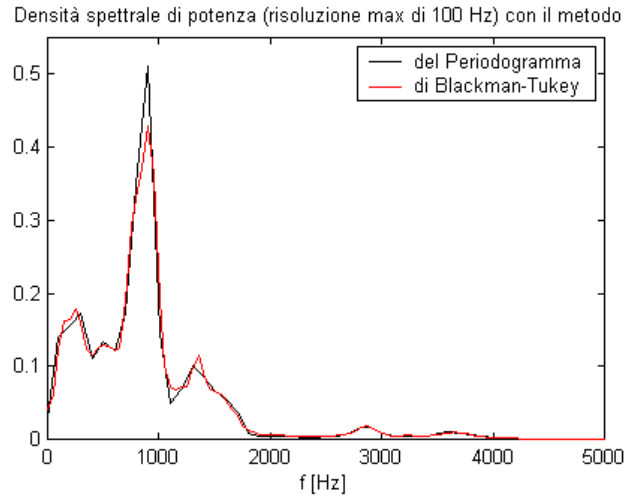


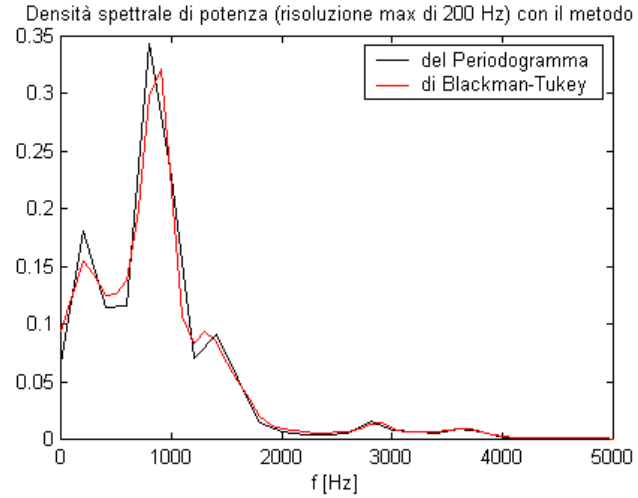
4) Suono vocale corrispondente alla parola : " **AREA** " .

```
» analisis
inserisci il nome del file contenente il segnale audio (formato wav): area
Il numero di campioni del file audio considerato sono : 18309
La frequenza di campionamento del file audio è di 22050 Hz

introdurre la massima risoluzione spettrale desiderata (in Hz): 50   (e successivamente : 100, 200)

Il vettore audio è stato suddiviso in 41 parti   ( 79 parti per 100 Hz ; 158 parti per 200 Hz )
```









Con questo programma verrà dimensionato un banco di filtri come combinazione di un filtro polifase e FFT.
Il procedimento seguito consiste nello scomporre lo spettro di un generico segnale in un numero di canali da noi fissato a priori, utilizzando un opportuno filtro passabasso e sottocampionando sia la risposta all'impulso del segnale che del filtro per il numero di canali considerati.
La risposta all'impulso del filtro è introdotta utilizzando una workspace ossia un file di dati di MATLAB (hanno estensione 'mat').

banco.m

```
% PROGETTO DI UN BANCO DI FILTRI ( decomposizione multicanale del segnale )

% La risposta all'impulso del filtro passabasso verrà introdotta dall'esterno

% Lo spettro del segnale introdotto quindi sarà suddiviso in un numero di blocchi pari
% a quelli richiesti dalle nostre specifiche per cui non ci sarà alcuna elaborazione
% del segnale in queste bande

clear all

err=0;
while err==0
   disp('inserisci il nome del file (.mat) contenente i campioni ');
   H=input('della risposta all''impulso del filtro passabasso : ','s');
   H=[H,'.mat'];
   if exist(H)==0
      disp('LA WORKSPACE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end
load(H);
N=length(h);
strN=int2str(N);
disp(['la risposta all''impulso del filtro è composta da ',strN,' campioni']);
err=0;
while err==0
   M=input('introduci il numero di canali : ');
   if M>N
      disp('Il numero di canali richiesto è troppo elevato! ');
   else
      err=1;
   end
end
if floor(N/M)~=ceil(N/M)
   h=[h',zeros(1,M-mod(N,M))];
   disp('ATTENZIONE : il rapporto tra la lunghezza della risposta all''impulso ');
   disp('             del filtro e il numero di canali non è un numero intero! ');
   N=length(h);
end
K=N/M;
err=0;
while err==0
   F=input('inserisci il nome della function contenente il segnale : ','s');
   if exist(F)==0
      disp('LA FUNZIONE INTRODOTTA NON ESISTE');
   else
      err=1;
   end
end
f=feval(F);
L=length(f);
[H,w]=freqz(h,1,L);

figure
plot(w/(2*pi),abs(H))
xlabel('f/fc')
set(gcf,'color',[1,1,1]);
title('spettro del filtro utilizzato')
axis([min(w/(2*pi)) max(w/(2*pi)) 0 max(abs(H))])
for m=1:M
```

```
   for k=0:K-1
      hh(k+1)=h(k*M+m);
   end
   pass1=[zeros(1,m-1) f];
   for k=0:floor((L-1)/M)
      ff(k+1)=pass1(k*M+1);
   end
   pass2=conv(ff,hh);
   lung=length(ff);
   convol(:,m)=pass2(1:lung)';
   clear ff hh
end
for i=1:lung
   dft(i,:)=fft(convol(i,:));
end
for i=1:M
   spettro(:,i)=abs(fft(dft(:,i)))*M;
end

mas=max(max(spettro));

freq=(-1/(2*M):1/(M*lung):(1/(2*M))*(1-1/(M*lung)))+1/M;
for i=1:M
   figure
   freq=freq-1/M;
   plot(freq,[spettro(ceil(lung/2)+1:lung,i)' spettro(1:floor(length(spettro)/2)+1,i)'])
   xlabel('f/fc')
   set(gcf,'color',[1,1,1]);
   stri=int2str(i);
   title(['spettro del segnale nella banda ',stri])
   axis([min(freq) max(freq) 0 mas])
   fine=[spettro(ceil(lung/2)+1:lung,i)' spettro(1:floor(length(spettro)/2)+1,i)' fine];
end
figure
freq=((1/(2*M))-1:1/(M*lung):(1/(2*M))*(1-1/(M*lung)));
plot(freq,fine)
xlabel('f/fc')
set(gcf,'color',[1,1,1]);
title('spettro del segnale rimettendo insieme tutte le bande')
```

sinus.m

```
function vet=sinus1;

tempo=500e-3;
T=5/50010;
f1=500;
f2=1500;
f3=2500;
f4=3500;
f5=4500;
campioni=tempo/T;
k=(1:1:campioni);
xf=(k-1)*T;
yf=(sin(2*pi*f1*xf)+0.8*sin(2*pi*f2*xf)+0.6*sin(2*pi*f3*xf)+...
   0.4*sin(2*pi*f4*xf)+0.2*sin(2*pi*f5*xf));
vet=yf;
figure
plot(xf,yf);
xlabel('sec');
set(gcf,'color',[1,1,1]);
title('grafico del segnale nel tempo');
zoom
ff=abs(fft(yf));
figure
freq=[-1/2:1/campioni:(1/2)*(1-1/campioni)];
plot(freq,[ff(ceil(campioni/2):campioni) ff(1:floor(campioni/2))]);
xlabel('f/fc');
set(gcf,'color',[1,1,1]);
title('spettro del segnale');
```

triang.m

```
function r=triang;

f=5000;
tempo=500e-3;
T=1/10000;
campioni=(tempo/T)+1;
k=(1:campioni);
x=(k-(length(k)+1)/2)*T;
for i=1:length(x)
   if i~=(length(k)+1)/2
      y(i)=2*(sin(pi*f*x(i))./(pi*(T^-1)*x(i))).^2;
   end
end
y((length(k)+1)/2)=2*(f*T)^2;
r=y;
figure
plot(x*1000,y);
xlabel('ms');
set(gcf,'color',[1,1,1]);
title('grafico del segnale nel tempo');
ff=abs(fft(r));
figure
freq=[-1/2:1/campioni:(1/2)*(1-1/campioni)];
plot(freq,[ff(ceil(campioni/2):campioni) ff(1:floor(campioni/2))]);
xlabel('f/fc');
set(gcf,'color',[1,1,1]);
title('spettro del segnale');
```

workspace utilizzate (filtri passabasso)

1) **Basso1_4.mat** : banda passante compresa tra $f_1 = 0$ Hz e $f_2 = 0{,}125 \cdot f_c$ Hz
   Il filtro viene utilizzato per la scomposizione in 4 canali dello spettro di un generico segnale.
   All'interno della workspace è presente un vettore di 204 elementi.

2) **Basso1_5.mat** : banda passante compresa tra $f_1 = 0$ Hz e $f_2 = 0{,}1 \cdot f_c$ Hz
   Il filtro viene utilizzato per la scomposizione in 5 canali dello spettro di un generico segnale.
   All'interno della workspace è presente un vettore di 205 elementi.

Entrambi realizzati con un filtro FIR in cui è usata una **finestra rettangolare**.

3) **Basso2_4.mat** : banda passante compresa tra $f_1 = 0$ Hz e $f_2 = 0{,}125 \cdot f_c$ Hz

   Il filtro viene utilizzato per la scomposizione in 4 canali dello spettro di un generico segnale.
   All'interno della workspace è presente un vettore di 204 elementi.

4) **Basso2_5.mat** : banda passante compresa tra $f_1 = 0$ Hz e $f_2 = 0{,}1 \cdot f_c$ Hz
   Il filtro viene utilizzato per la scomposizione in 5 canali dello spettro di un generico segnale.
   All'interno della workspace è presente un vettore di 205 elementi.

Entrambi realizzati con un filtro FIR in cui è usata una **finestra di Hamming**.

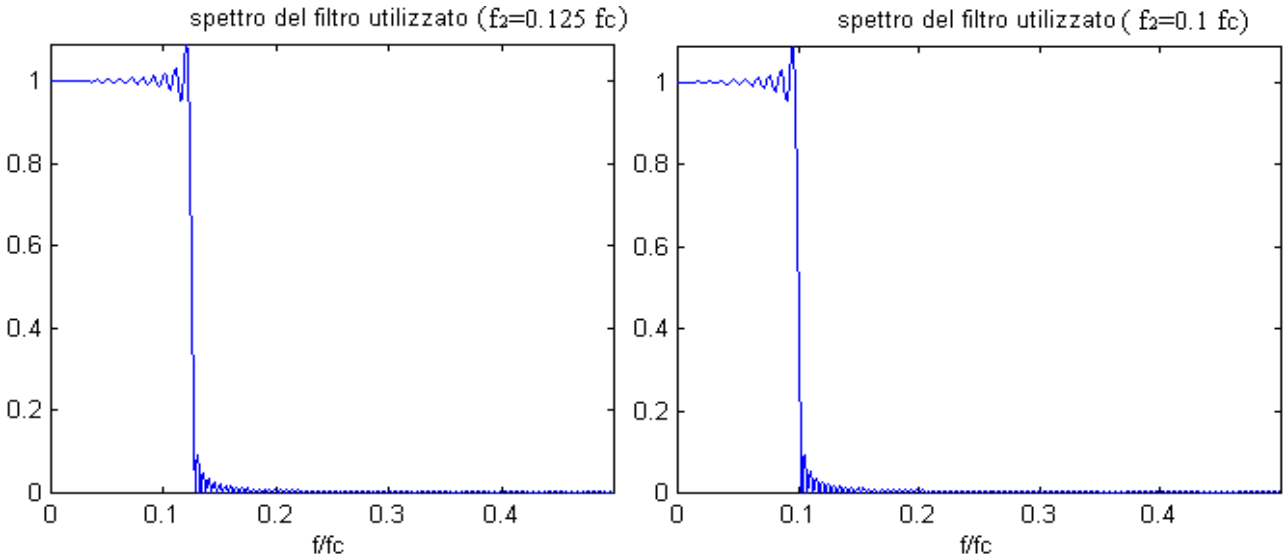
5) **Basso3_5.mat** : banda passante compresa tra $f_1 = 0$ Hz e $f_2 = 0{,}1 \cdot f_c$ Hz
   Il filtro viene utilizzato per la scomposizione in 5 canali dello spettro di un generico segnale.
   All'interno della workspace è presente un vettore di 205 elementi.

Realizzato con un filtro FIR a fase lineare secondo l'approssimazione di **Chebyshev,** utilizzando l'algoritmo di **Remez**: la banda di transizione fissata è compresa tra $f_{t1} = 0{,}095 \cdot f_c$ Hz e $f_{t2} = 0{,}105 \cdot f_c$ Hz ed inoltre il valore del ripple fissato nella banda passante e arrestata è lo stesso.

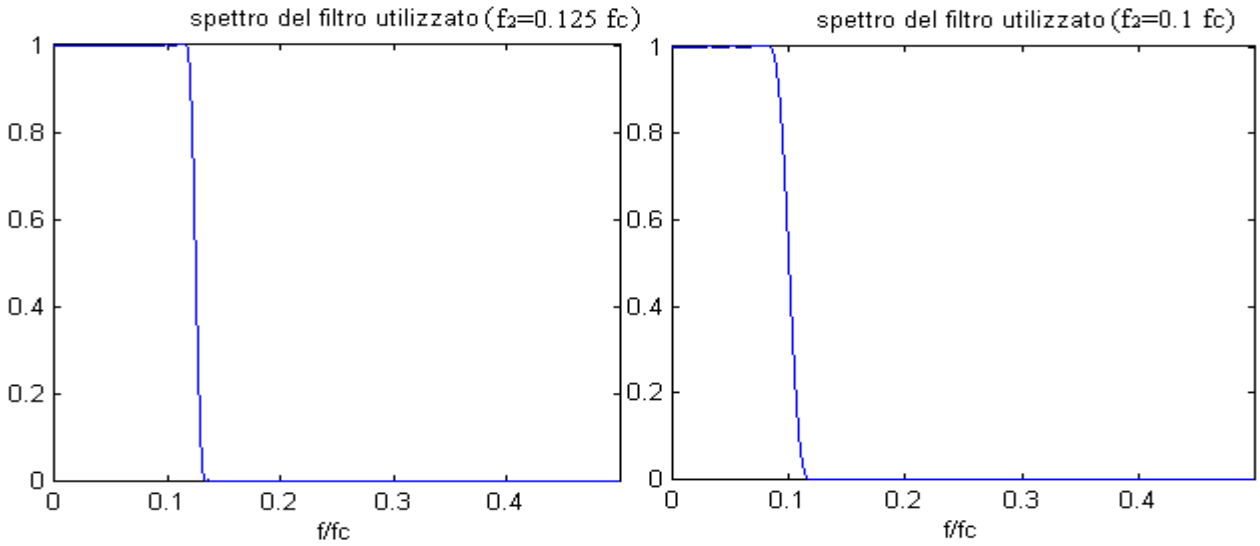
esecuzione del programma

I filtri utilizzati hanno i seguenti spettri :

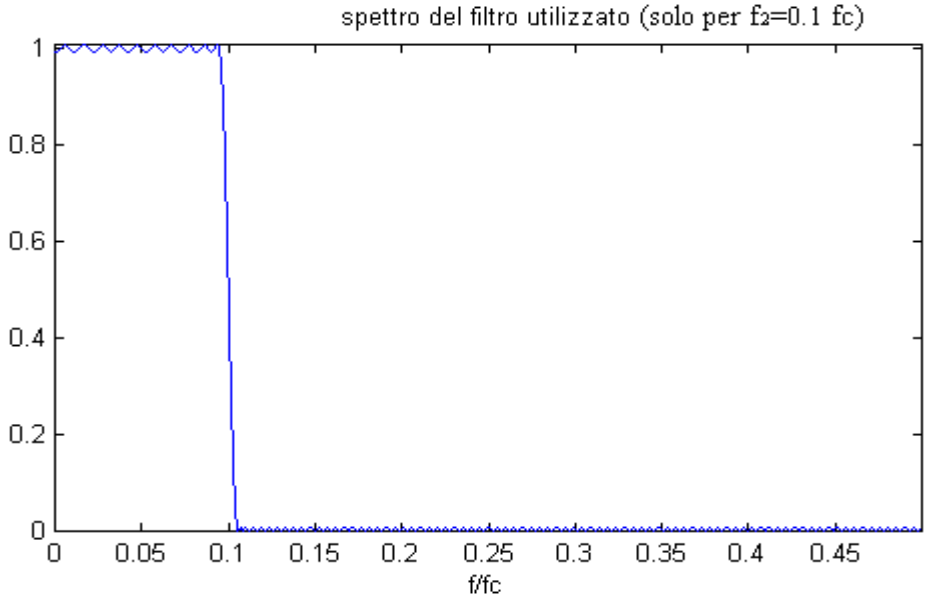
basso1_4 e **basso1_5**



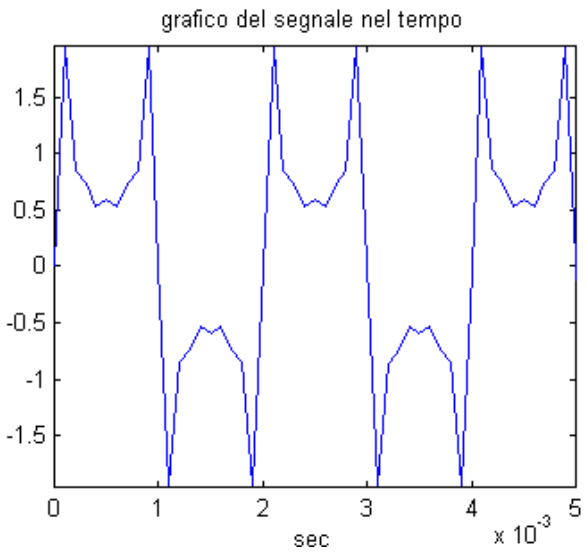
basso2_4 e **basso2_5**

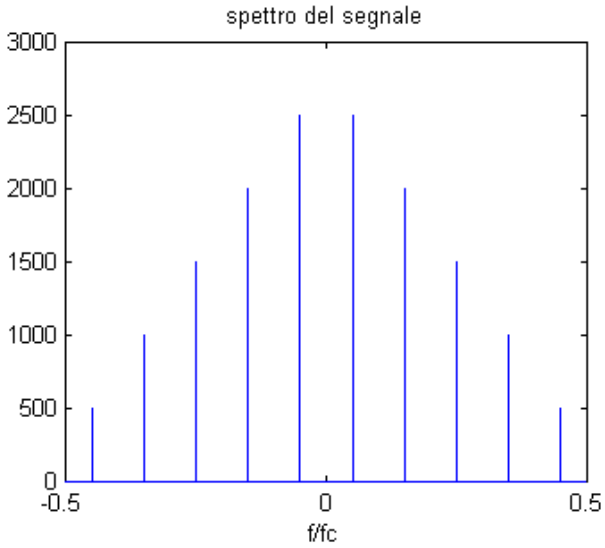


basso1_5

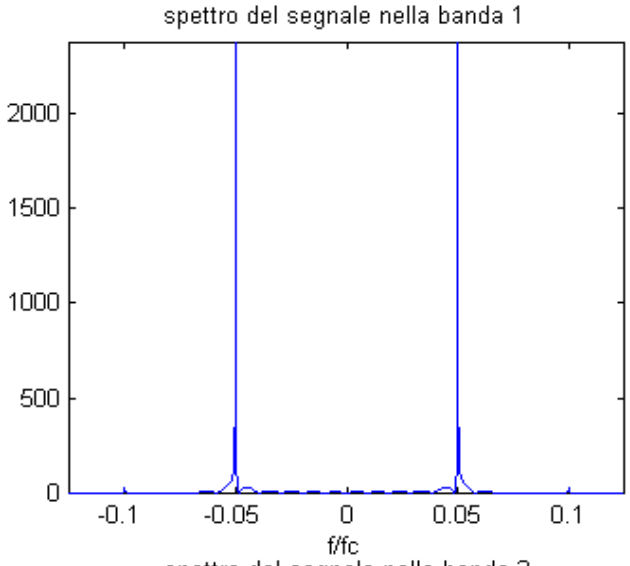


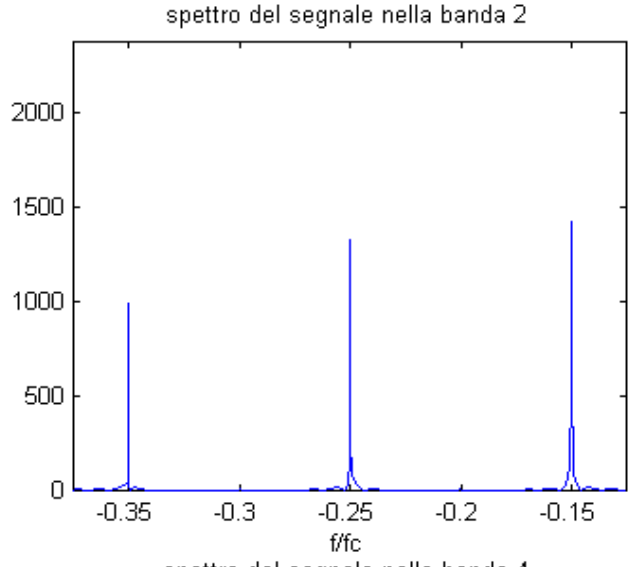
1) Consideriamo la function sinus.m il cui grafico e spettro di questa funzione sono:

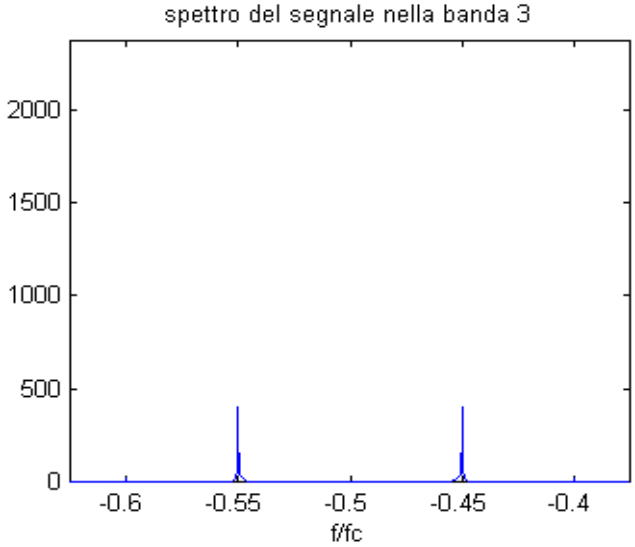


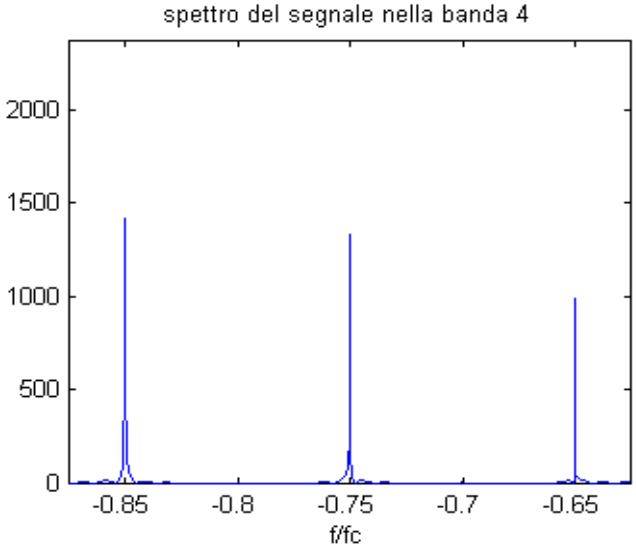


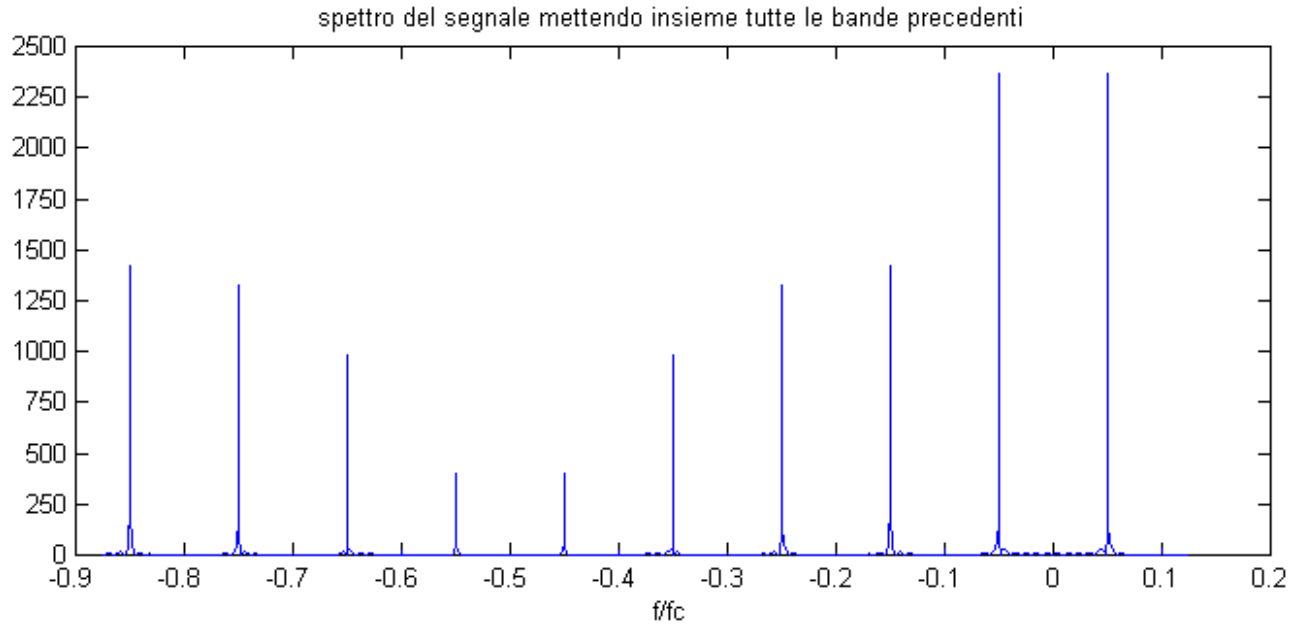

```
» banco
inserisci il nome del file (.mat) contenente i campioni
della risposta all'impulso del filtro passabasso : basso1_4
la risposta all'impulso del filtro è composta da 204 campioni
introduci il numero di canali : 4
inserisci il nome della function contenente il segnale : sinus
```



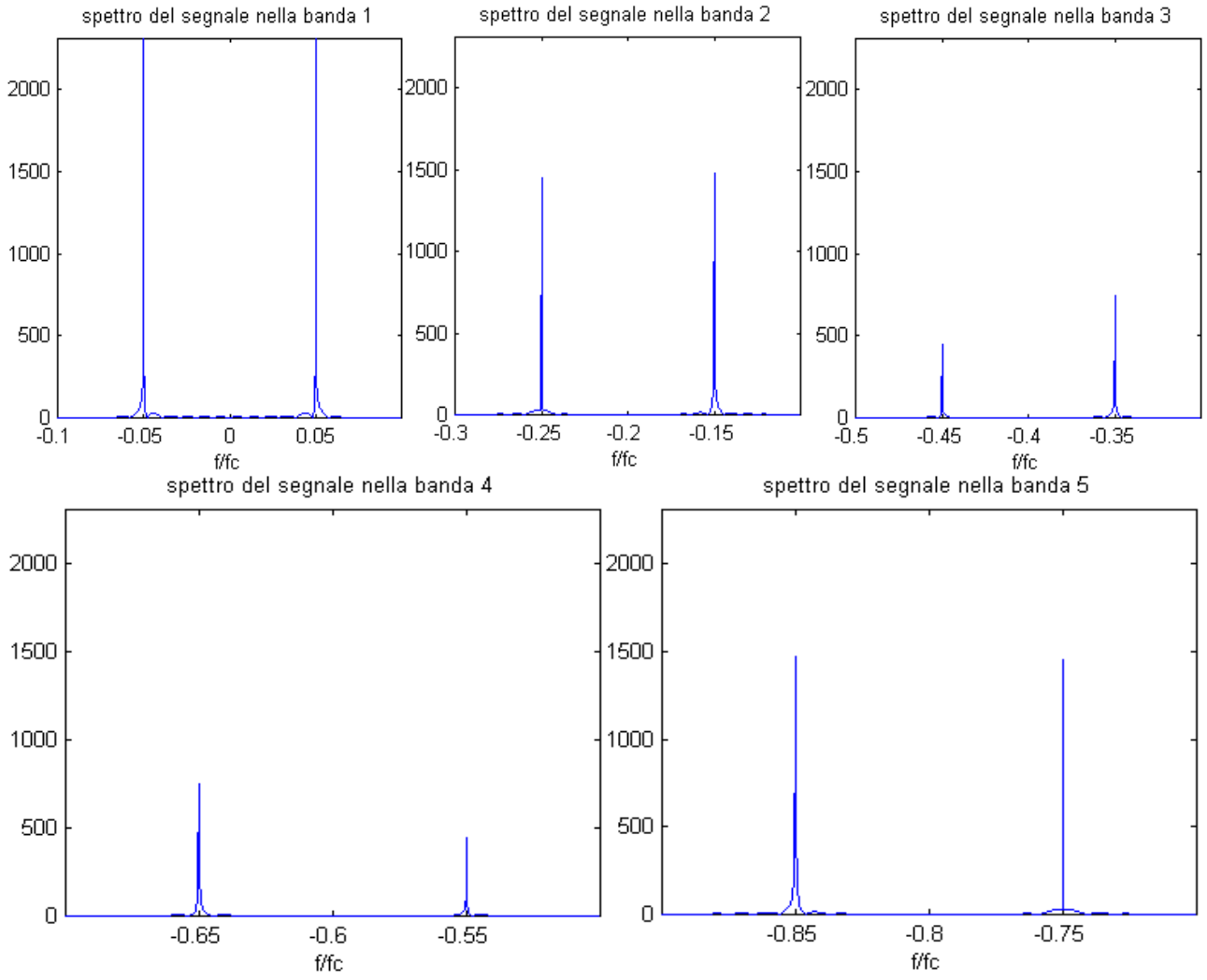


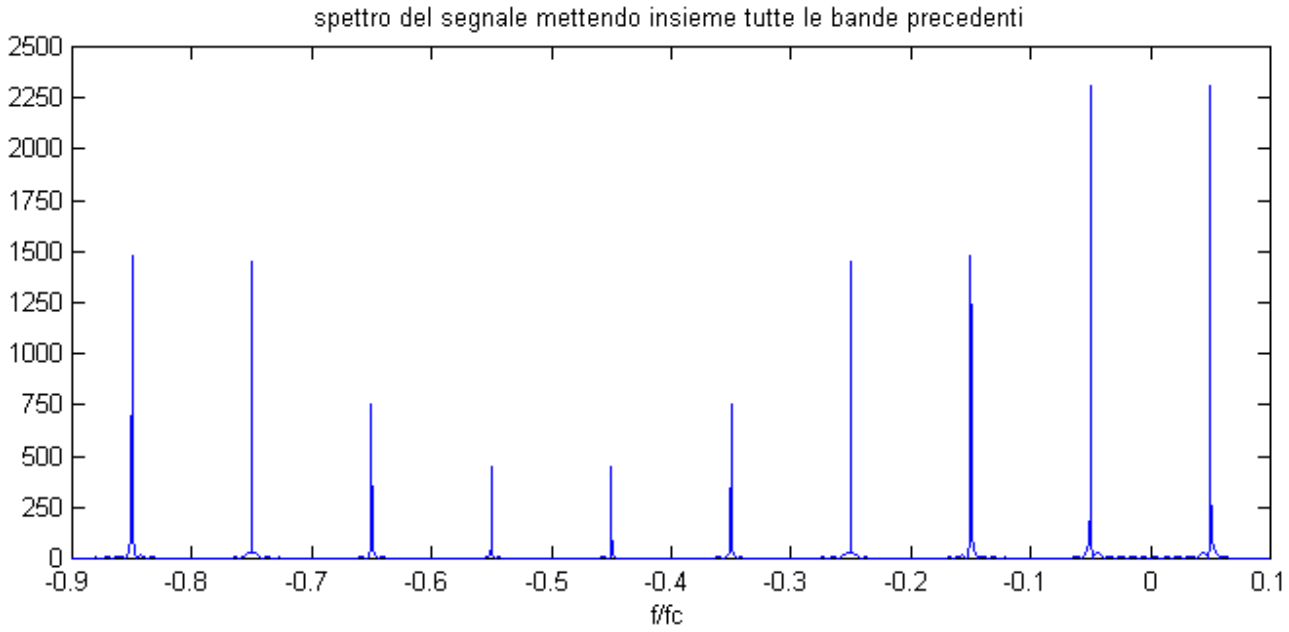






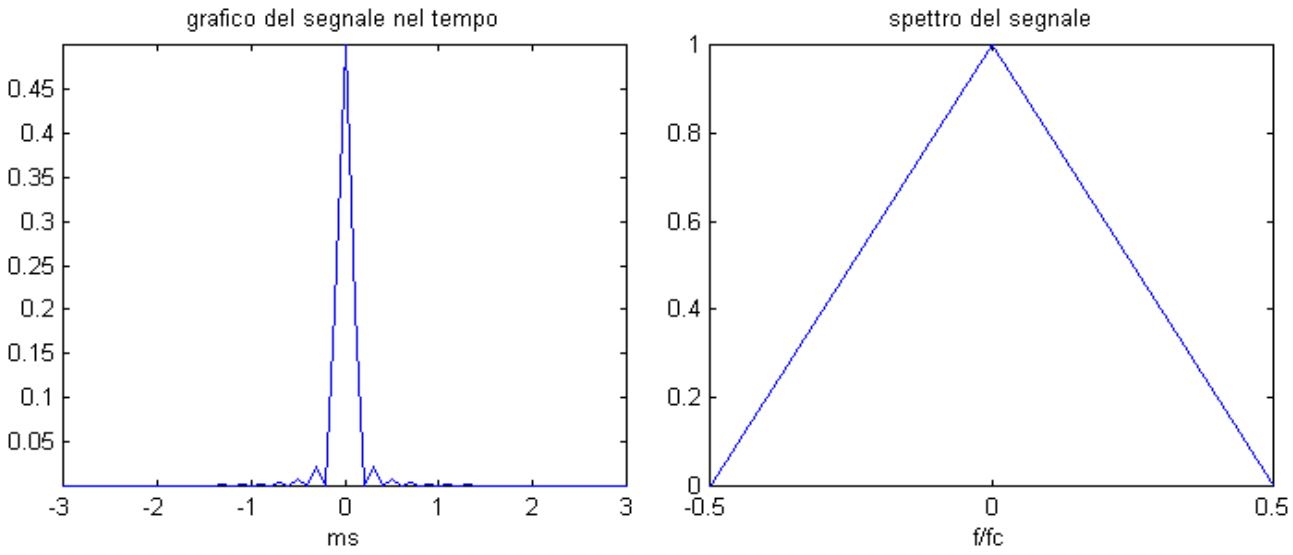

```
» banco
inserisci il nome del file (.mat) contenente i campioni
della risposta all'impulso del filtro passabasso : basso1_5
la risposta all'impulso del filtro è composta da 205 campioni
introduci il numero di canali : 5
inserisci il nome della function contenente il segnale : sinus
```





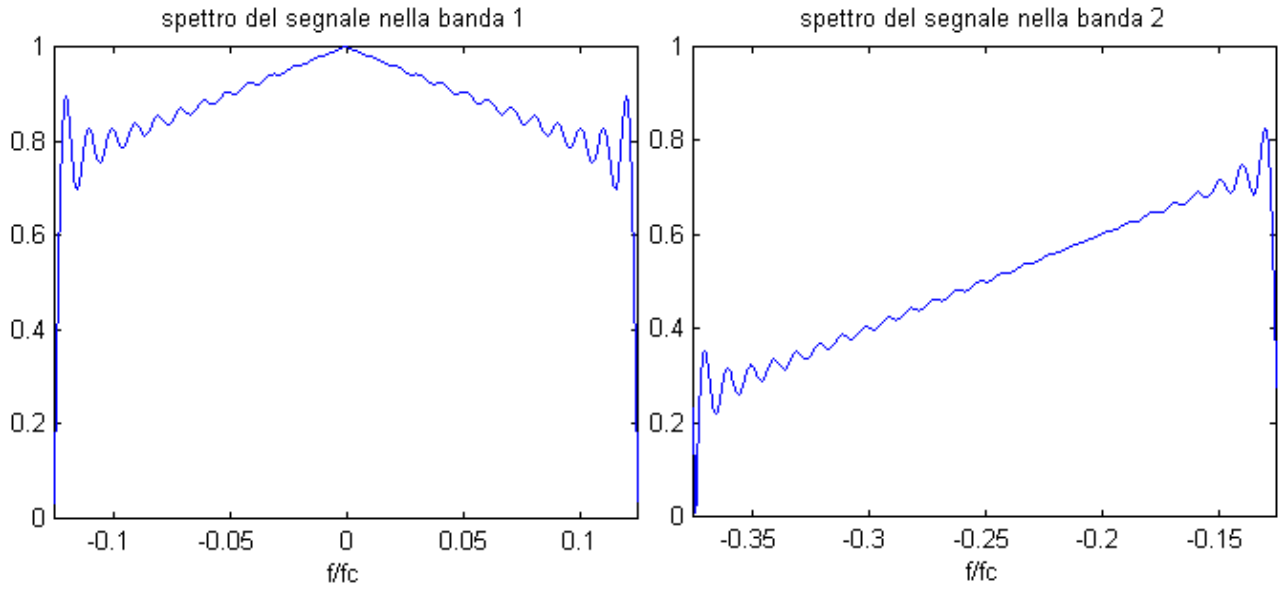
OSS: i grafici sono praticamente gli stessi se considero gli altri due tipi di filtri sia nella scomposizione in 4 che in 5 canali.

2) Consideriamo la function triang.m il cui grafico e spettro di questa funzione sono:

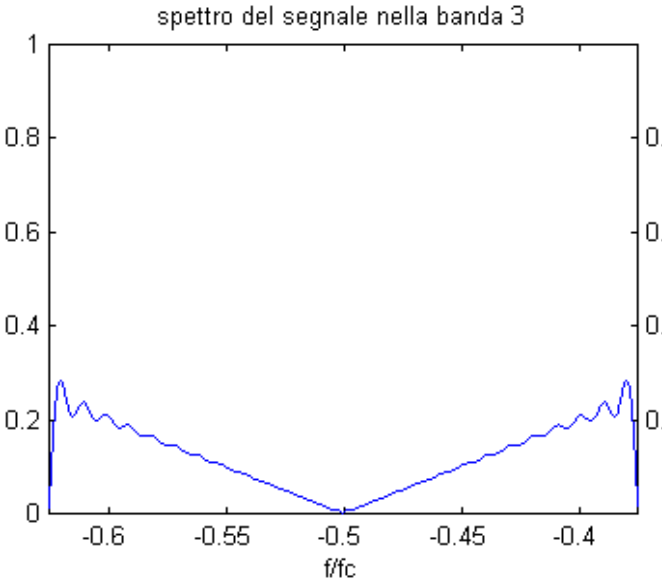


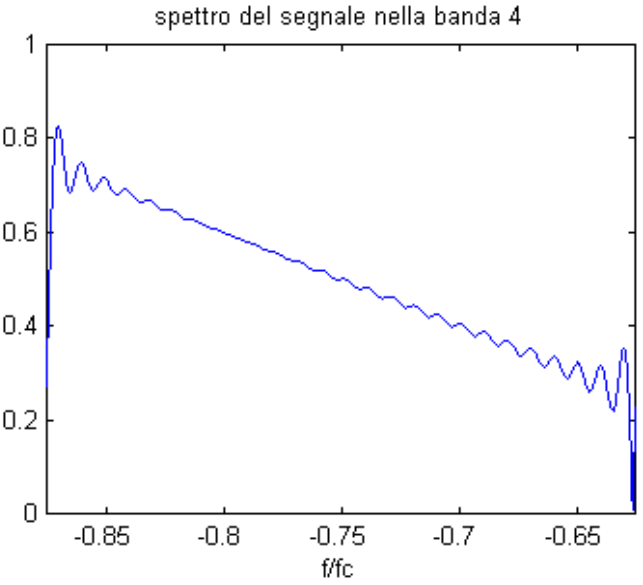

```
» banco
inserisci il nome del file (.mat) contenente i campioni
della risposta all'impulso del filtro passabasso : basso1_4  ( e dopo basso2_4 )
la risposta all'impulso del filtro è composta da 204 campioni
introduci il numero di canali : 4
inserisci il nome della function contenente il segnale : triang
```

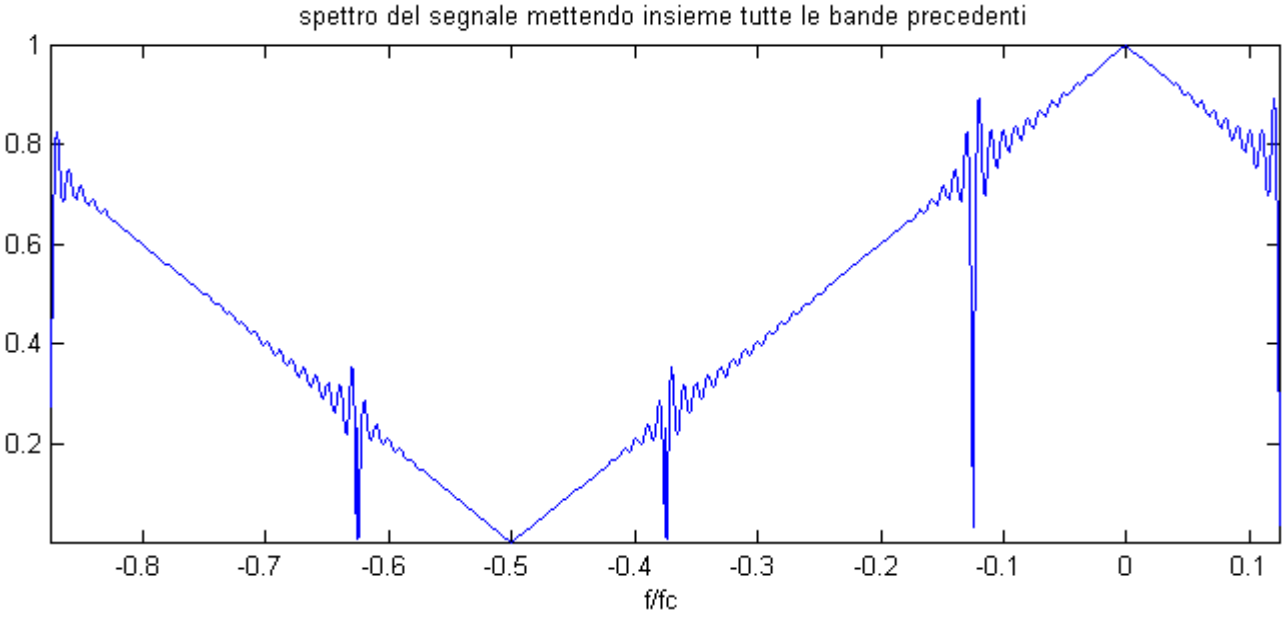
- **Con basso1_4 :**


spettro del segnale nella banda 1
1
0.8
0.6
0.4
0.2
0
-0.1
-0.05
0
0.05
0.1
f/fc

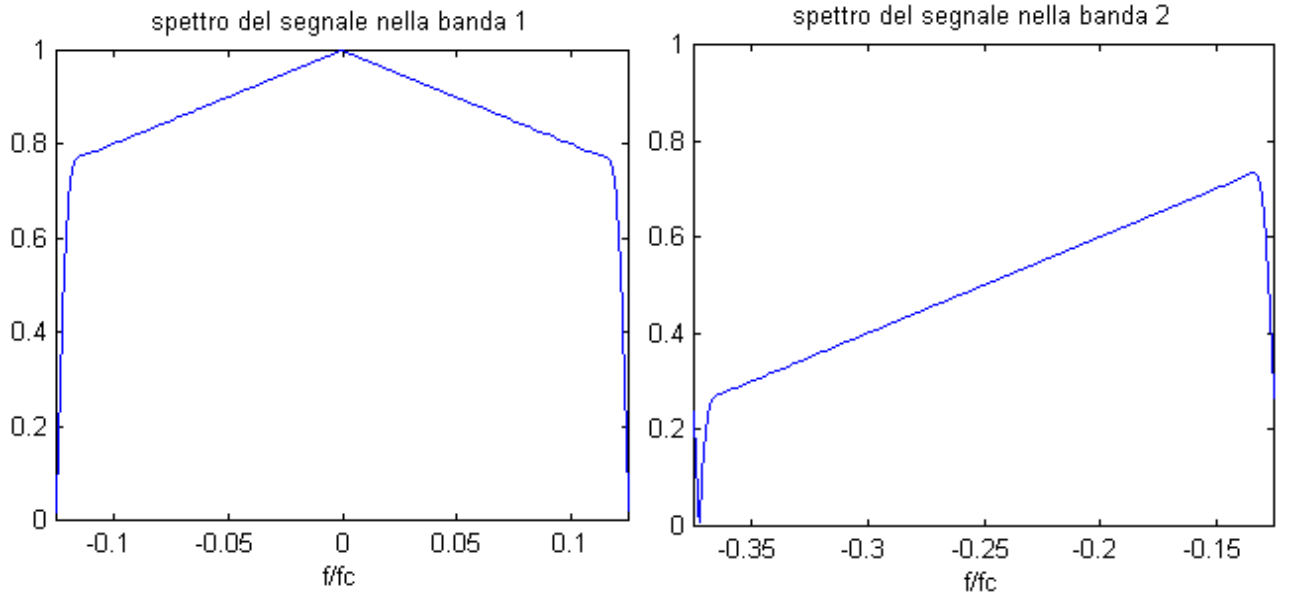
spettro del segnale nella banda 2
1
0.8
0.6
0.4
0.2
0
-0.35
-0.3
-0.25
-0.2
-0.15
f/fc


spettro del segnale nella banda 3
1
0.8
0.6
0.4
0.2
0
-0.6
-0.55
-0.5
-0.45
-0.4
f/fc

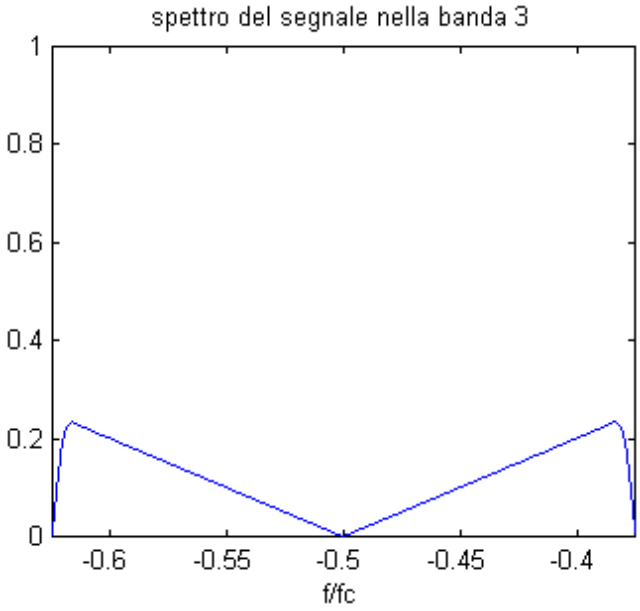

spettro del segnale nella banda 4
1
0.8
0.6
0.4
0.2
0
-0.85
-0.8
-0.75
-0.7
-0.65
f/fc

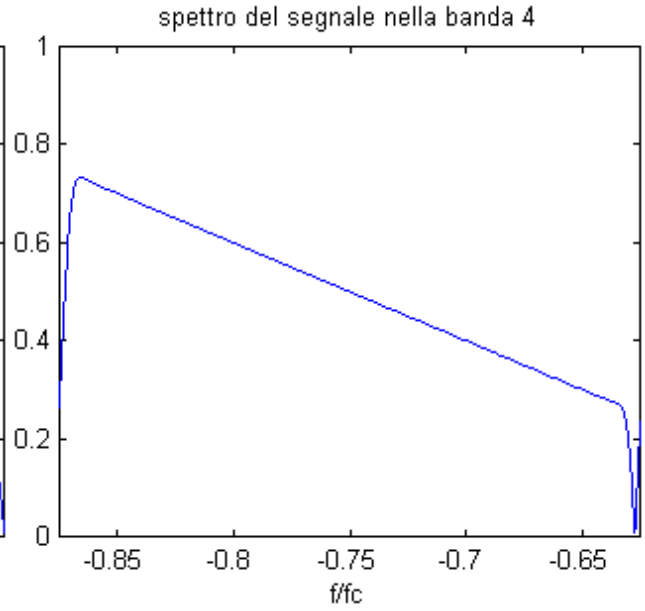

spettro del segnale mettendo insieme tutte le bande precedenti
1
0.8
0.6
0.4
0.2
-0.8
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
f/fc

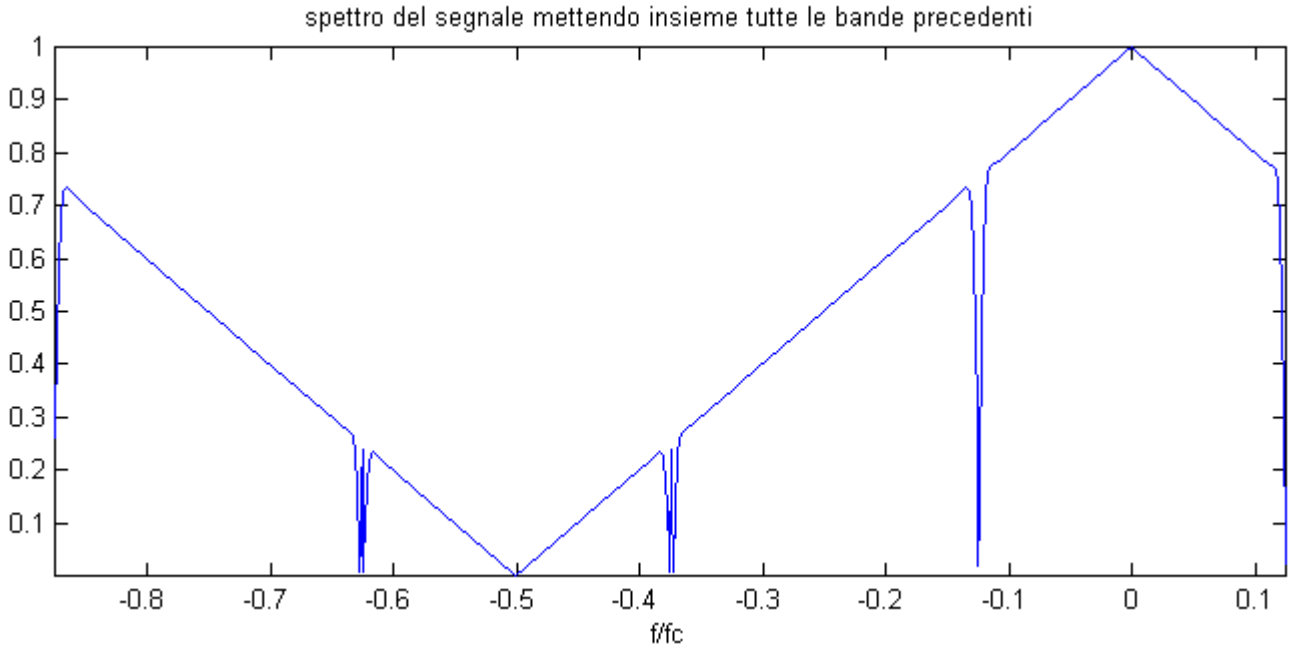
- **Con basso2_4 :**


spettro del segnale nella banda 1
f/fc

spettro del segnale nella banda 2
f/fc

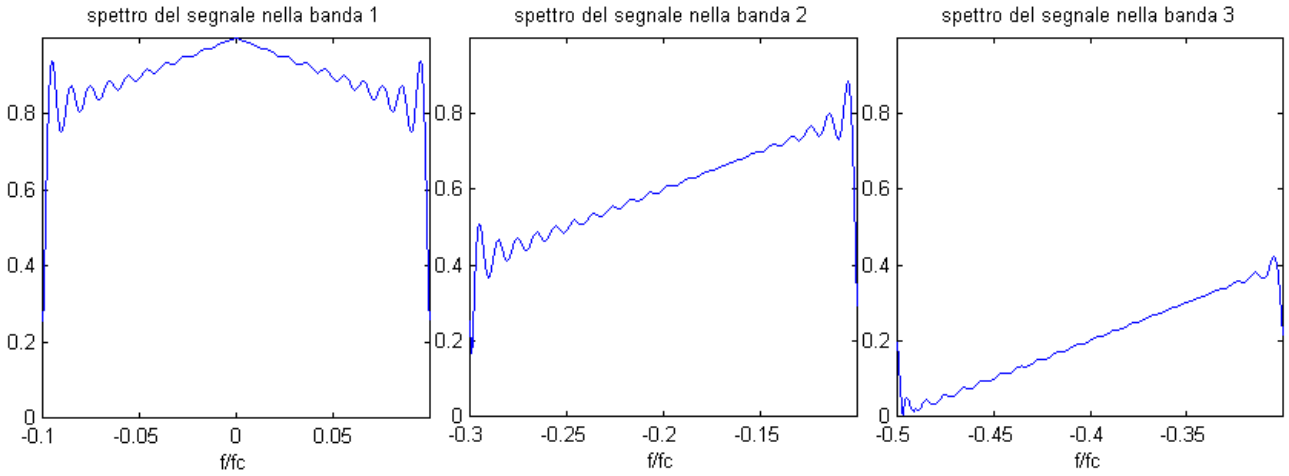

spettro del segnale nella banda 3
f/fc

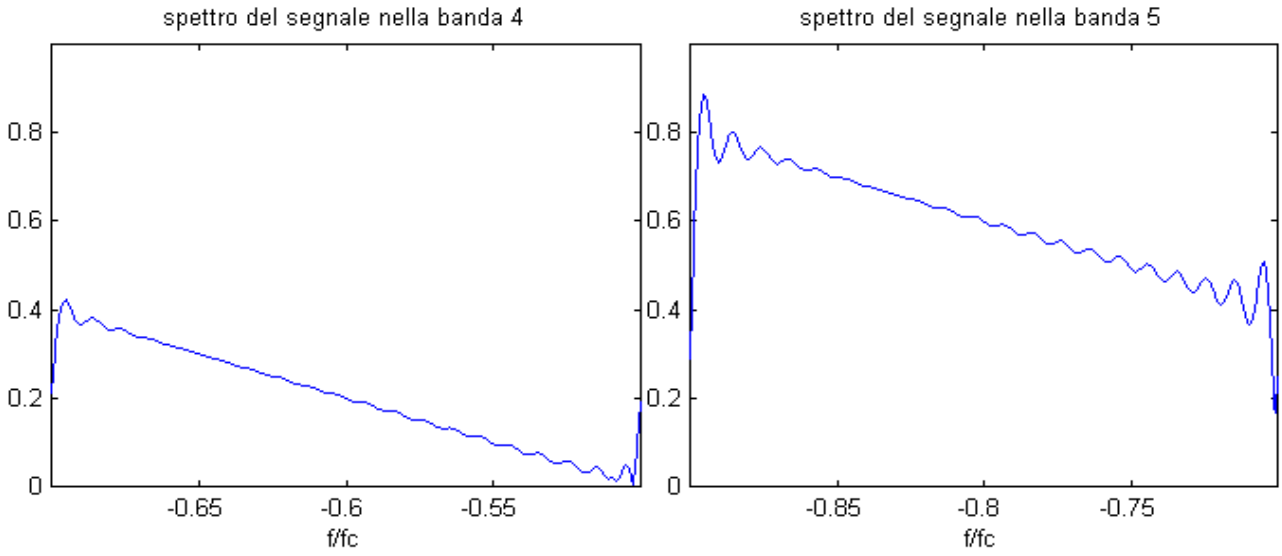

spettro del segnale nella banda 4
f/fc

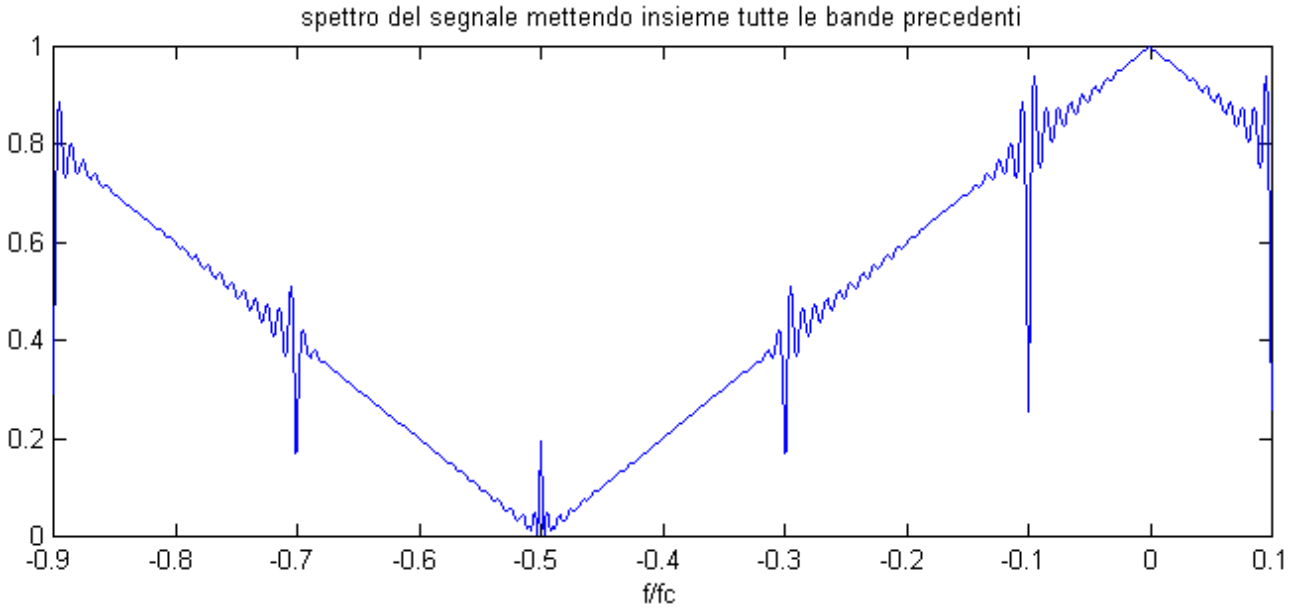

spettro del segnale mettendo insieme tutte le bande precedenti
f/fc

```
» banco
inserisci il nome del file (.mat) contenente i campioni
della risposta all'impulso del filtro passabasso : basso1_5  ( e dopo basso2_5 e basso3_5 )
la risposta all'impulso del filtro è composta da 205 campioni
introduci il numero di canali : 5
inserisci il nome della function contenente il segnale : triang
```

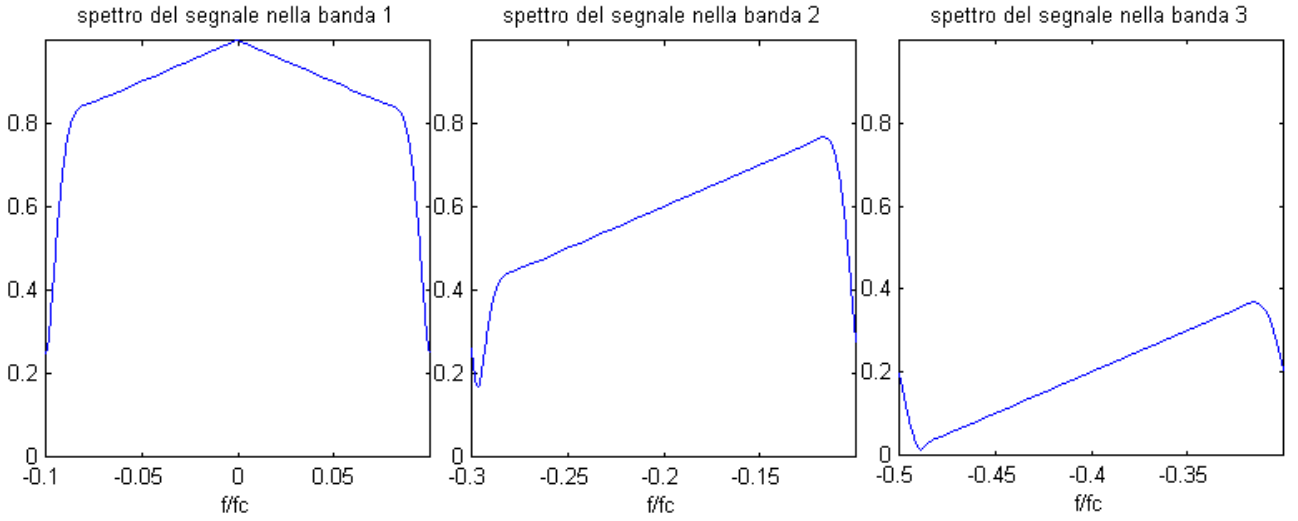
- **Con basso1_5 :**

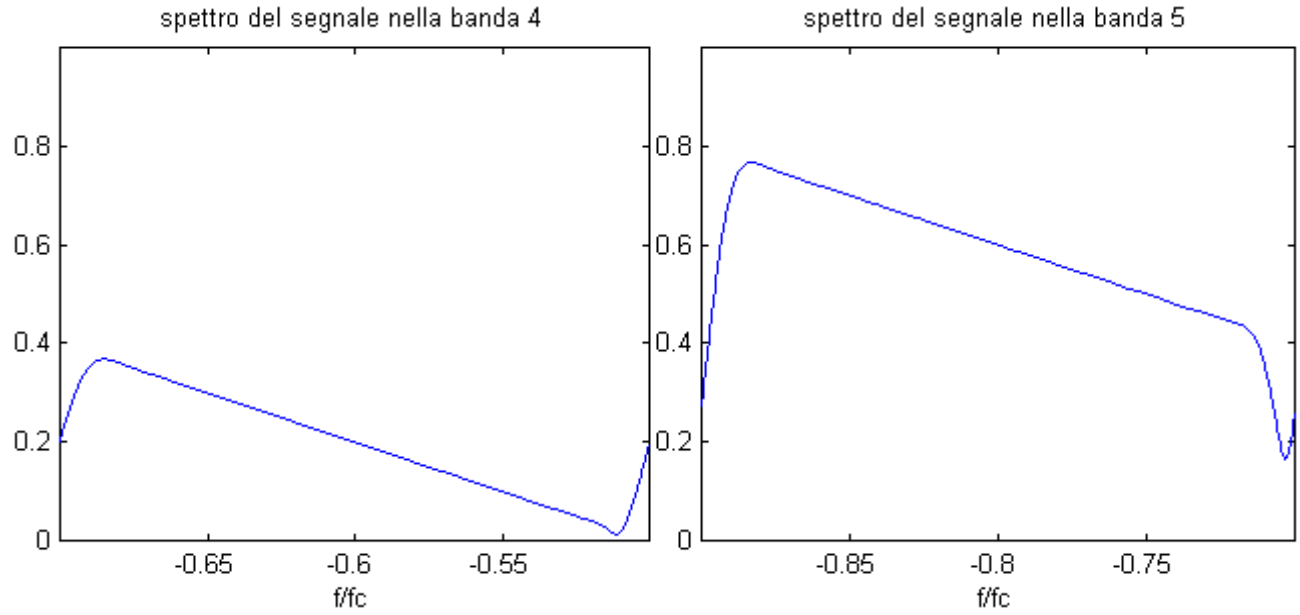


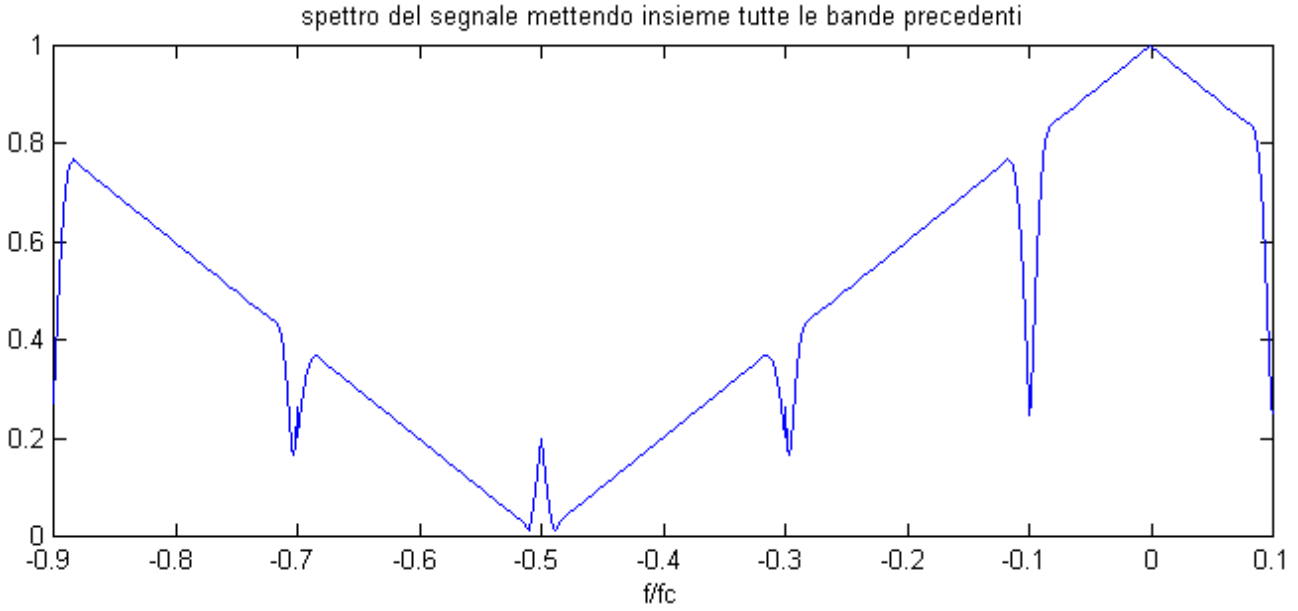




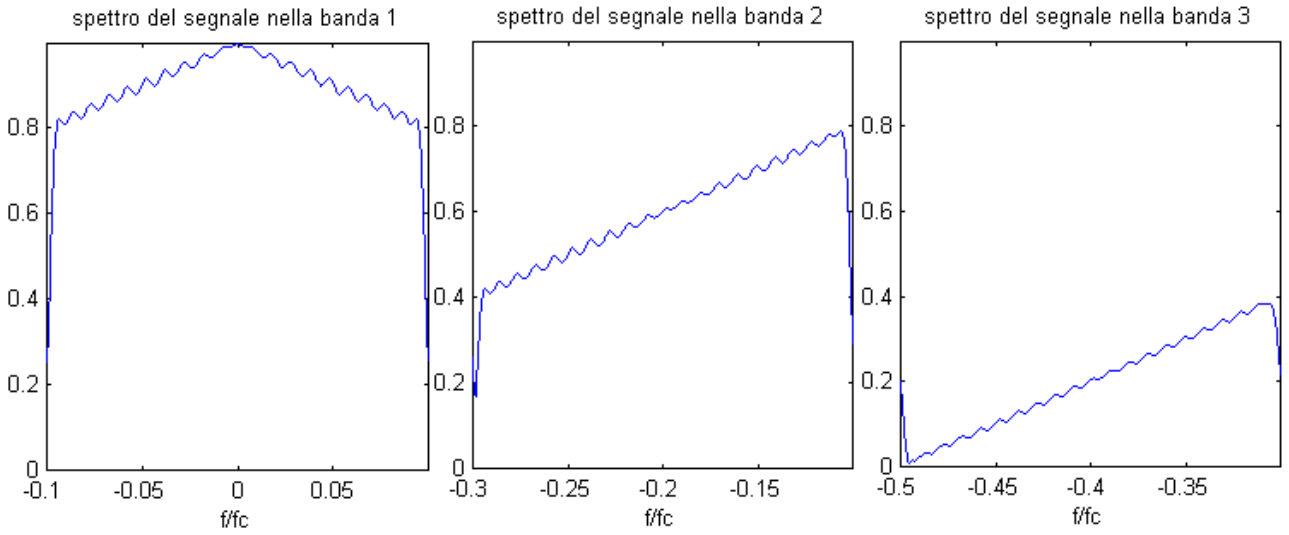
- **Con basso2_5 :**

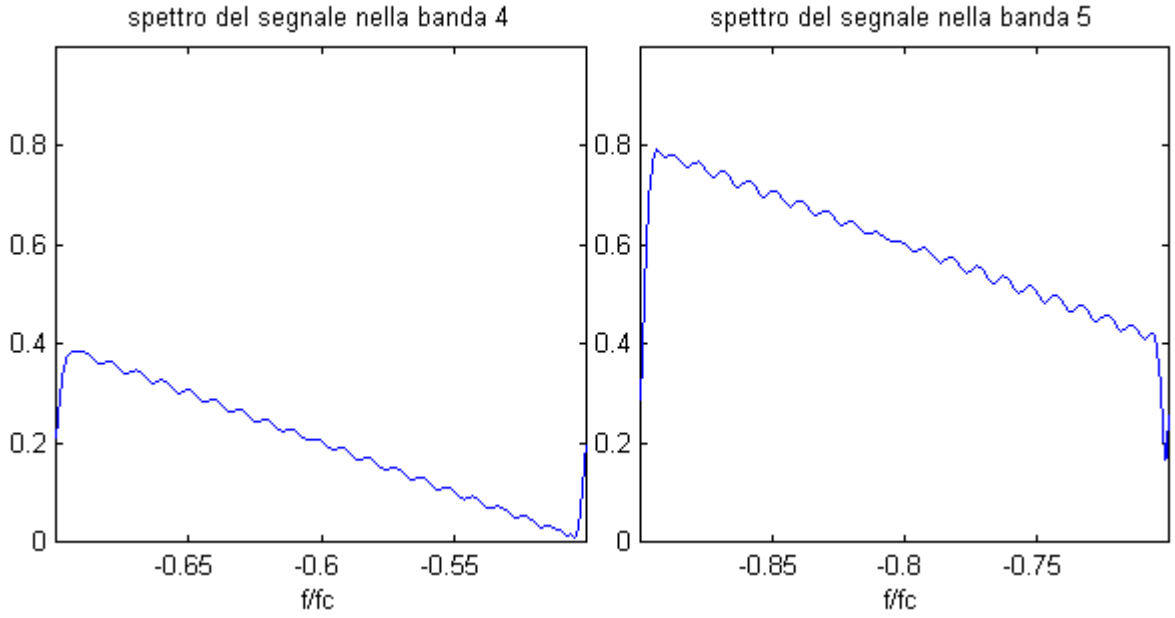

spettro del segnale nella banda 1
spettro del segnale nella banda 2
spettro del segnale nella banda 3
f/fc
f/fc
f/fc

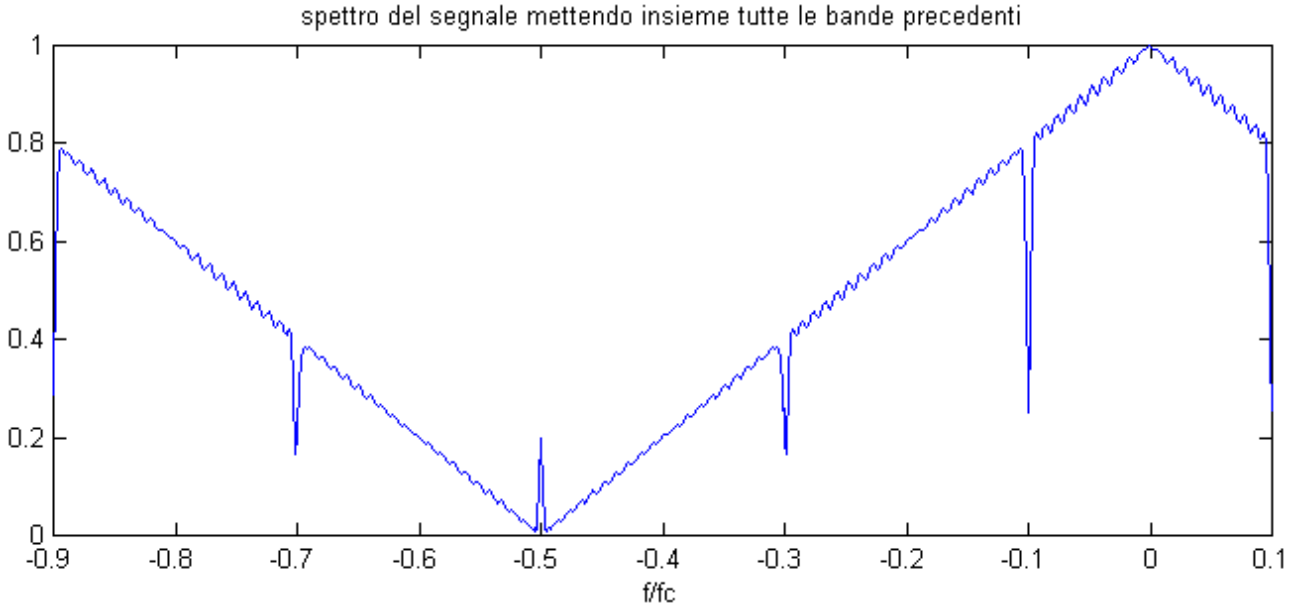

spettro del segnale nella banda 4
spettro del segnale nella banda 5
f/fc
f/fc


spettro del segnale mettendo insieme tutte le bande precedenti
f/fc

- **Con basso3_5 :**


spettro del segnale nella banda 1
spettro del segnale nella banda 2
spettro del segnale nella banda 3
f/fc
f/fc
f/fc


spettro del segnale nella banda 4
spettro del segnale nella banda 5
f/fc
f/fc


spettro del segnale mettendo insieme tutte le bande precedenti
f/fc