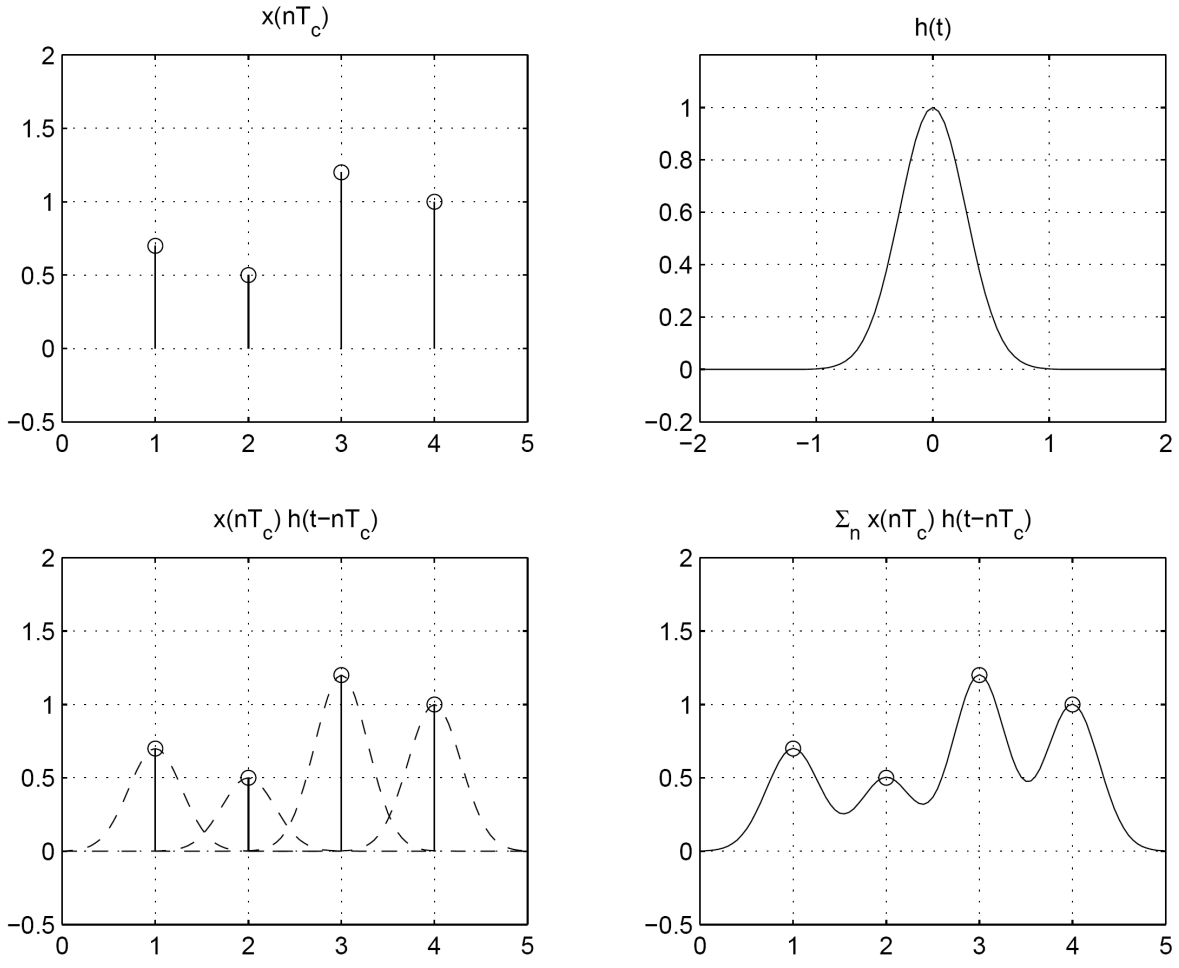
Laboratorio di Fondamenti di Segnali e Trasmissione

Esercitazione n.3 del 24/4/2009 (Alessandro Tomasoni)

Ricostruzione di segnali campionati

Abbiamo visto a lezione che uno dei passi necessari a convertire un segnale analogico in un segnale numerico è il campionamento. Campionare un segnale significa discretizzarlo nel tempo. Formalmente si passa da un segnale continuo nel tempo $x(t)$ ad una sua versione discreta $x(nT_c)$, cioè una sequenza di valori equispaziati da un tempo $T_c = 1/f_c$. Abbiamo anche visto che il teorema del campionamento garantisce che tale operazione è completamente reversibile, e cioè è possibile ricostruire $x(t)$ da $x(nT_c)$, a patto che $f_c > 2B_x$. In tal caso la ricostruzione del segnale è possibile tramite "filtraggio" con una funzione continua $h(t)$, risposta all'impulso di un filtro passa-basso:

$$x(t) = \sum_{n=-\infty}^{+\infty} x(nT_c)\, h(t - nT_c) \qquad (1)$$

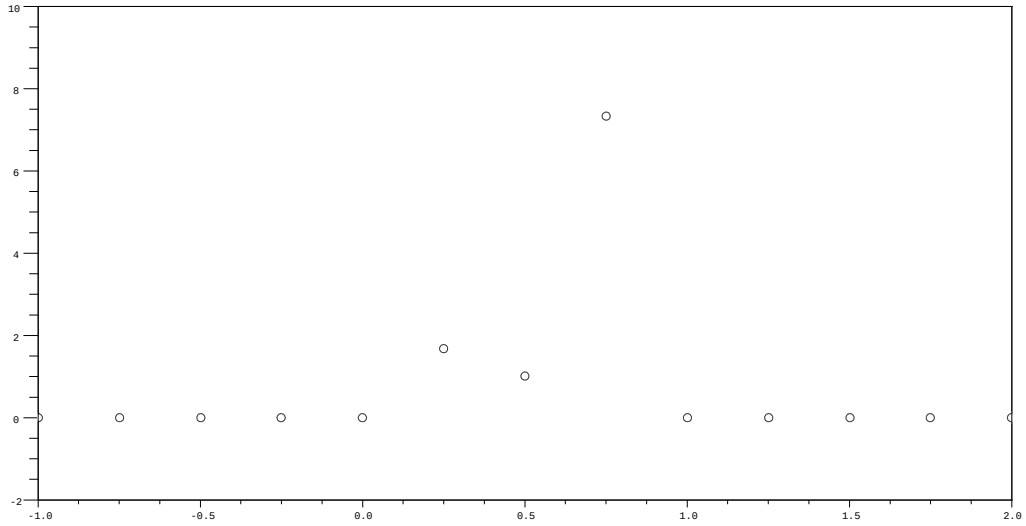


Oggi cercheremo di implementare la ricostruzione in SCILAB. In realtà operando in SCILAB, come sappiamo, anche $x(t)$ e $h(t)$ saranno a loro volta segnali discreti. Tuttavia, avranno in generale un passo di discretizzazione minore di T_c, e quindi il filtraggio ci permetterà di ricostruire una versione di $x(t)$ più precisa, e di verificare quanto studiato in teoria. In SCILAB, l'operazione (1) si può implementare in diversi modi. Il modo più semplice dal punto di vista del codice, utilizza di nuovo la funzione **`convol(x,h)`**. Vediamo come.

Esempio 1:

Come prima cosa supponiamo di avere come dato in ingresso T_c e campioniamo una sequenza *x(t)*: p.e. supponiamo di avere campionato con f_c = 4 Hz (T_c = 250 ms) la nostra $x(t) = (\sin(5\pi t) + 2\sin(2\pi t))^2$, per $0 \le t \le 1$, osservata però su un intervallo più ampio:

```
--> Tc=0.25;
--> tc=-1:Tc:2;
--> xc=(sin(5*%pi*tc)+2*sin(2*%pi*tc)).^2;
--> idx=find(tc<0 | tc>1);
--> xc(idx)=0;
--> plot(tc,xc,'ok')
--> set(gca(),'data_bounds',[-1 -2;2 9]);
```



Sappiamo che non é in linea di principio a banda limitata. Quando abbiamo usato la $x(t)$ per gli esempi di convoluzione, abbiamo assunto per la banda B_x un valore di 5 Hz che corrisponde ad un ampiezza pari al 10% del massimo, circa. Quindi il teorema del campionamento ci dice che questa f_c non basta; verifichiamo che la ricostruzione fallisce anche con un passa-basso ideale.
Scegliamo il passo di discretizzazione del segnale ricostruito $dt < T_c$: abbiamo visto che un buon aspetto della forma d'onda si ha con dt = 5 ms:

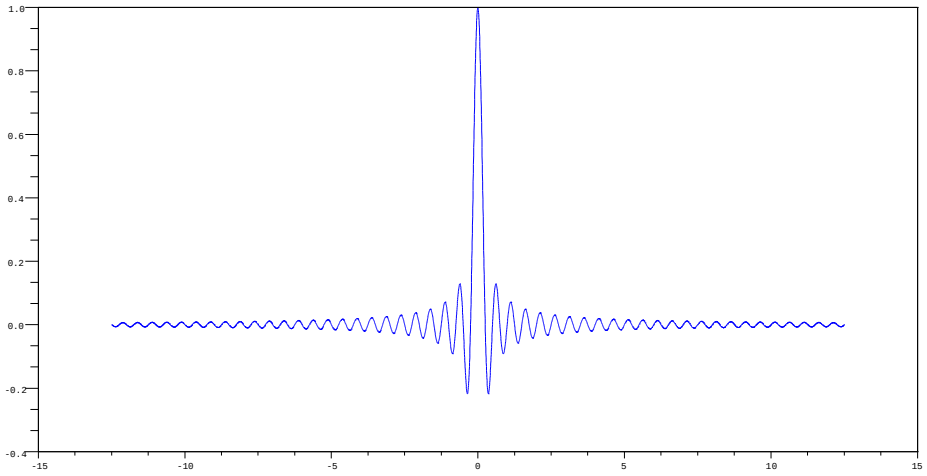
```
--> dt=0.005;
```

Scegliamo la funzione interpolante *h(t)*: un filtro passa-basso ideale con banda $f_c/2$:

$$H(f) = T_c\, rect(fT_c) \quad \xrightarrow{\Im^{-1}} \quad h(t) = sinc(t/T_c)$$

Sappiamo che tale forma d'onda si estende molto a lungo nei tempi. Prendiamola limitata all'intervallo ($-50T_c$, $50T_c$):

```
--> th=-50*Tc:dt:50*Tc;
--> h=sinc(%pi*th/Tc);
```

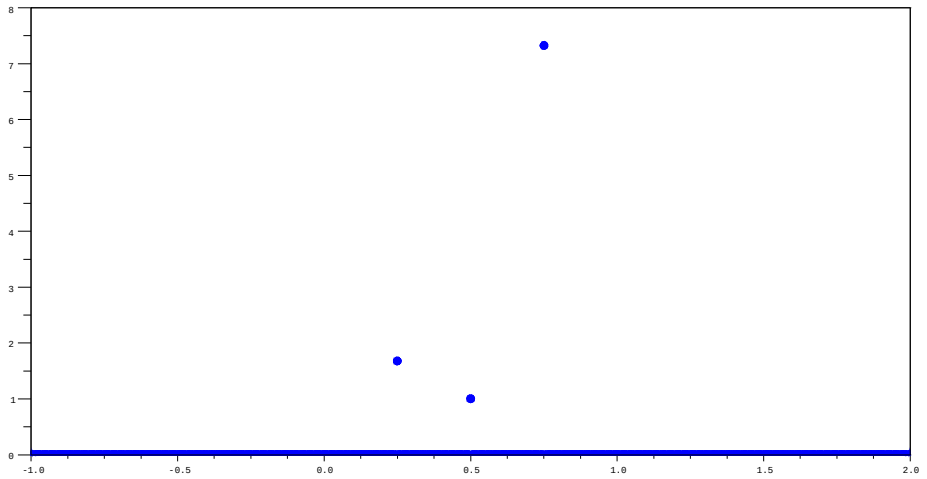


Ora occorre sovracampionare $x(nT_c)$: bisogna portarlo ad avere lo stesso passo di discretizzazione di h, per poterli convolvere tramite la funzione **convol**; negli istanti che aggiungiamo porremmo **xc** a zero (in modo da dare contributo nullo) prima di operare con la ricostruzione. Per fare questo facciamo la seguente sequenza di operazioni:

```
--> Npassi=Tc/dt;
--> tc2=-1:dt:2;
--> xc2=zeros(1,length(tc2));
--> xc2(1:Npassi:length(xc2))=xc;
```

Verifichiamo di aver ottenuto quello che volevamo:

```
--> figure
--> plot(tc2,xc2,'.r')
```



Ok: xc2 è una versione sovracampionata di **xc**, con zeri dove **xc** non era definita. Ora operiamo tramite **convol**:

```
--> xr=convol(h,xc2);
```

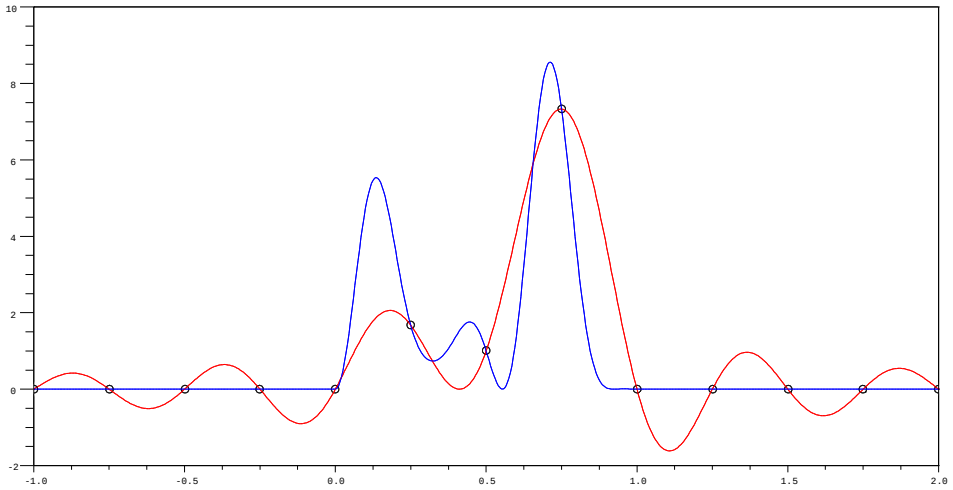
Non rimane che da calcolare il nuovo vettore dei tempi; per fare questo ricordate che la funzione **convol(h,x)** restituisce come primo campione, quello che corrisponde all'istante **th(1)+tx(1)**, quindi nel nostro caso:

```
--> t=th(1)+tc2(1)+(0:length(xr)-1)*dt;
```

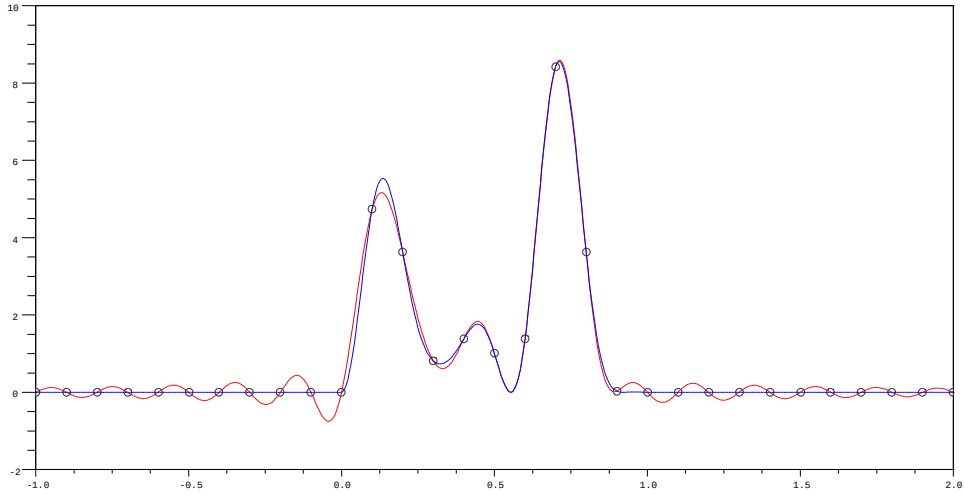
Vediamo il risultato della ricostruzione, confrontandolo anche con la vera x:

```
--> plot(t,xr,'r-')
--> x=(sin(5*%pi*t)+2*sin(2*%pi*t)).^2;
--> x (find(t>1))=0;
```

```
--> x (find(t<0))=0;
--> plot(t,x,'b-')
```

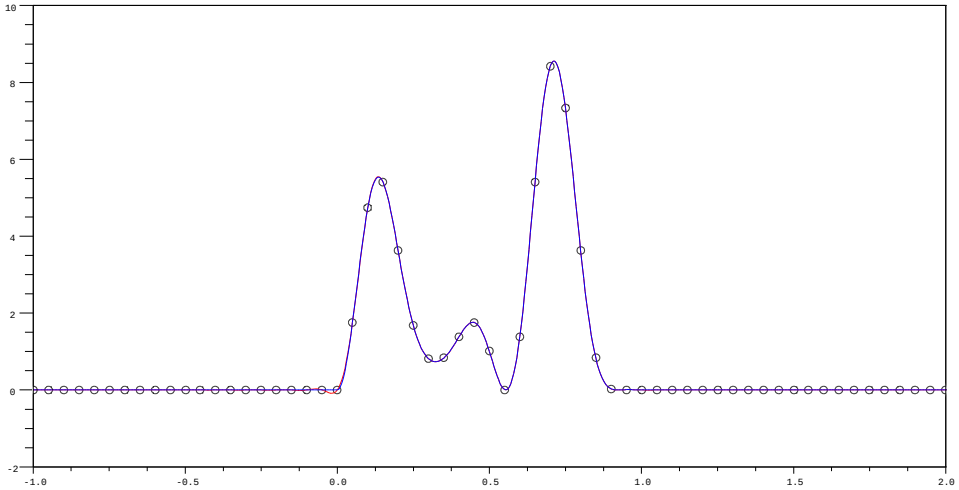


Come vedete la ricostruzione è molto deludente. Se provate con f_c = 10 Hz, che è la minima garantita dal teorema del campionamento, ritrovate ancora parecchie oscillazioni in ricostruzione.



In effetti é vero che il teorema del campionamento vale per segnali a banda rigorosamente limitata. A 5 Hz X(f) abbiamo visto che cade circa al 10% dell'ampiezza e non è proprio zero. Se assumiamo Bx=10 Hz, proviamo con f_c = 20 Hz. Definizione del segnale campionato a 20Hz:

```
--> clear
--> Tc=0.05;
--> …
```



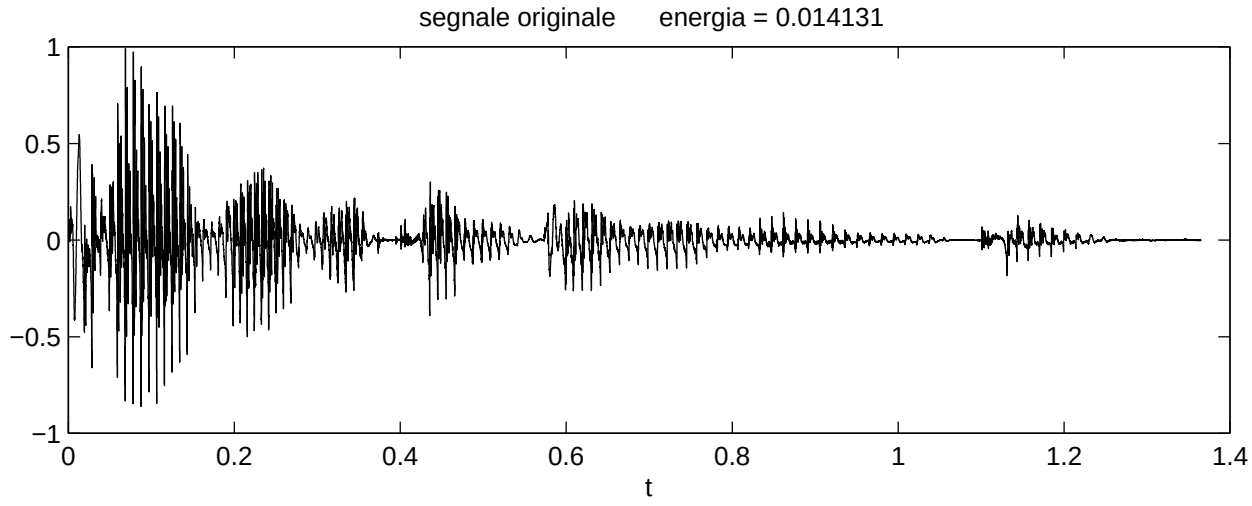
La ricostruzione è buona, anche se zoomando si notano ancora delle piccole oscillazioni, soprattutto intorno a t=0, che é il punto più critico. Si consideri anche un altro problema: la durata del sinc è teoricamente infinita, mentre noi siamo costretti a limitarne la finestra di osservazione.

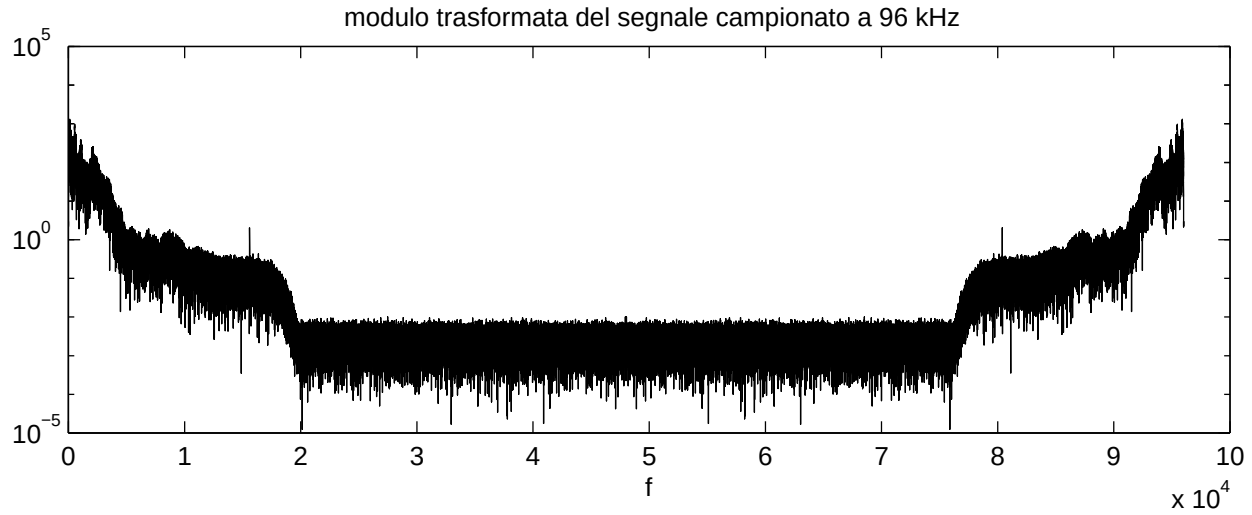
Esempio 2:

Quest'esempio utilizza un vero segnale vocale, che nella sua versione originale è campionato a 96 kHz, e memorizzato su file in formato wav: camp96.wav.
SCILAB dispone di funzioni **wavread wavwrite** per la lettura ed il salvataggio in formato wav di segnali.
In questo esempio leggiamo il segnale originale e lo sottocampioniamo a diverse frequenze, per poi ricostruirlo alla frequenza originale di 96 kHz, tramite filtraggio passa-basso non ideale. Settando il parametro salvataggio=1, ciascuno dei segnali ricostruiti viene anche salvato su file. Al termine di ciascuna ricostruzione, viene calcolato l'errore di ricostruzione e la sua energia.
Le frequenze di sottocampionamento sono: 48, 24, 12, 6 e 3 kHz.

Si noti:

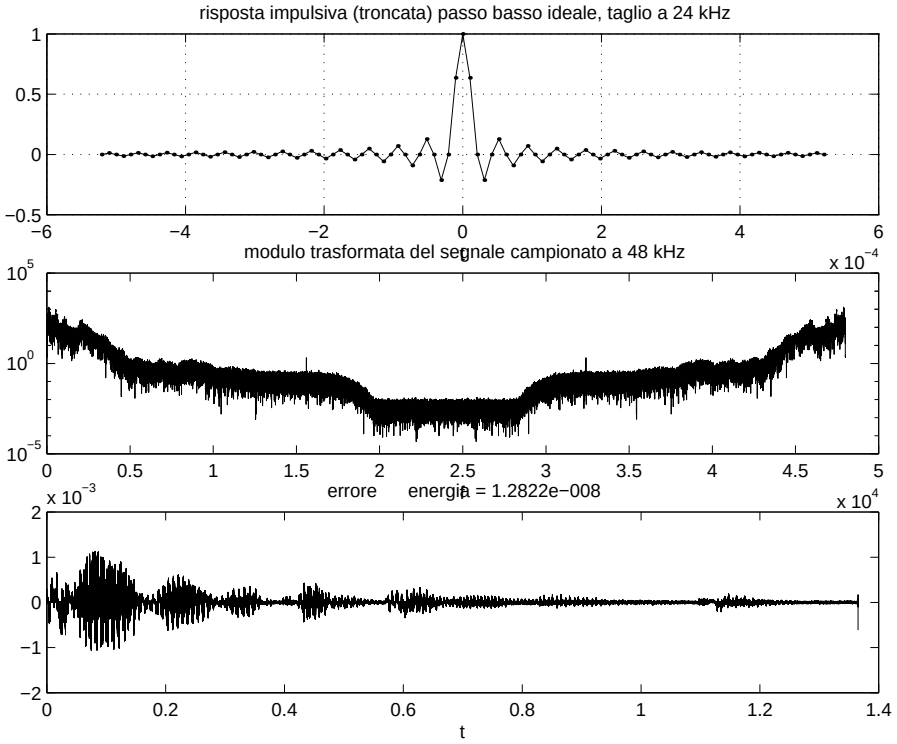
- la frequenza centrale (che sarebbe la massima frequenza osservata) è, al solito, fc/2 = 48 kHz.
- dalla trasformata del segnale originale, si notano tre livelli (grosso modo): un primo livello di ampiezza intorno a 1000-100 (60-40 dB) tra 0 5 kHz, un secondo livello intorno a 1 (0 dB) tra 5 e 20 kHz, ed un terzo "plateu" sui 0.01 (–40 dB), oltre i 20 kHz. Quest'ultimo è essenzialmente rumore.



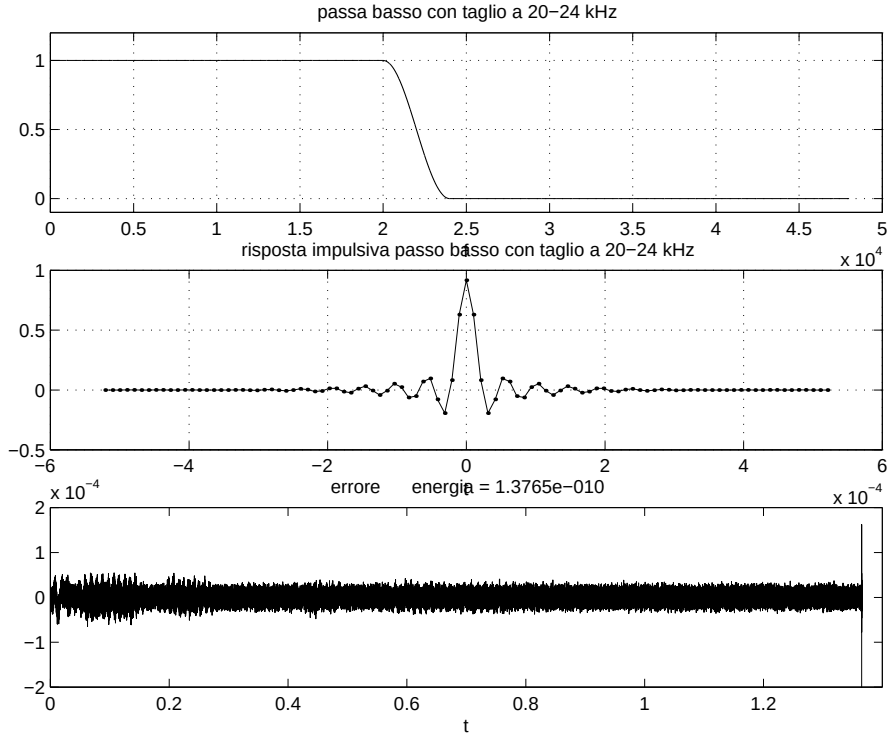


- la h(t) che interpola dal segnale a 48 kHz a quello a 96 kHz è ideale (ampiezza in 0 = a 1, zeri a multipli di 1/48 ms = 0.2 10^{-4} s), anche se è male campionata, perchè ha solo 2 campioni per ogni lobo (96/48)
- la frequenza centrale ora diventa fc/2 = 24 kHz, ma tutto quello che abbiamo trascurato è rumore.
- l'energia dell'errore è 10^{-8} su 10^{-2} => abbiamo perso una frazione ridicola di segnale, e tuttavia si può far di meglio: l'errore dipende dal segnale , perchè abbiamo dovuto troncare la h(t)

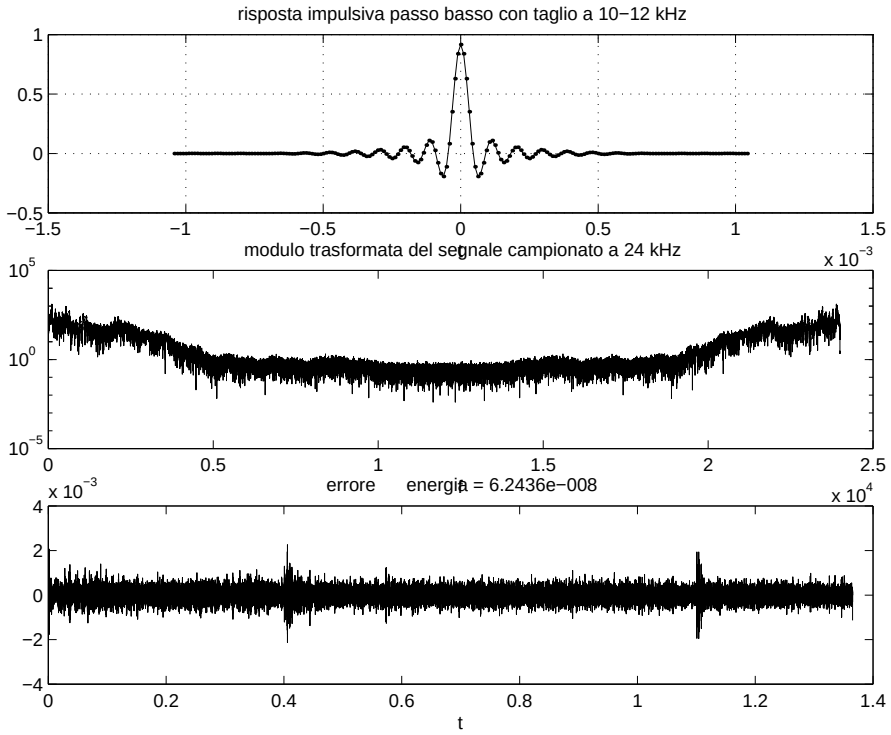
.



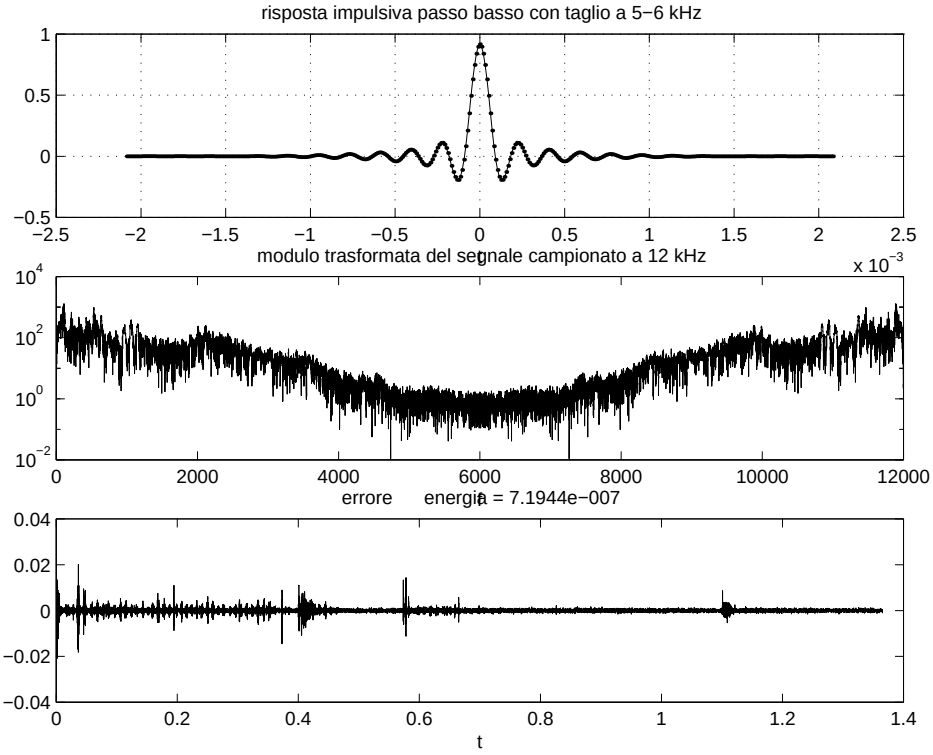
- sempre a 48 kHz: ricostruzione con una funzione di trasferimento del filtro passa basso a 24 kHz, che non è ideale...
- la risposta all'impulso non vale 1 nell'origine, e non ha zeri in corrispondenza dei multipli di 1/fc = 0.2 10^{-4} s
- ora l'energia dell'errore è 10^{-10} su 10^{-2} => grazie al fatto che non abbiamo dovuto troncare artificialmente la risposta del filtro: l'errore non dipende più dal segnale => abbiamo troncato solo rumore



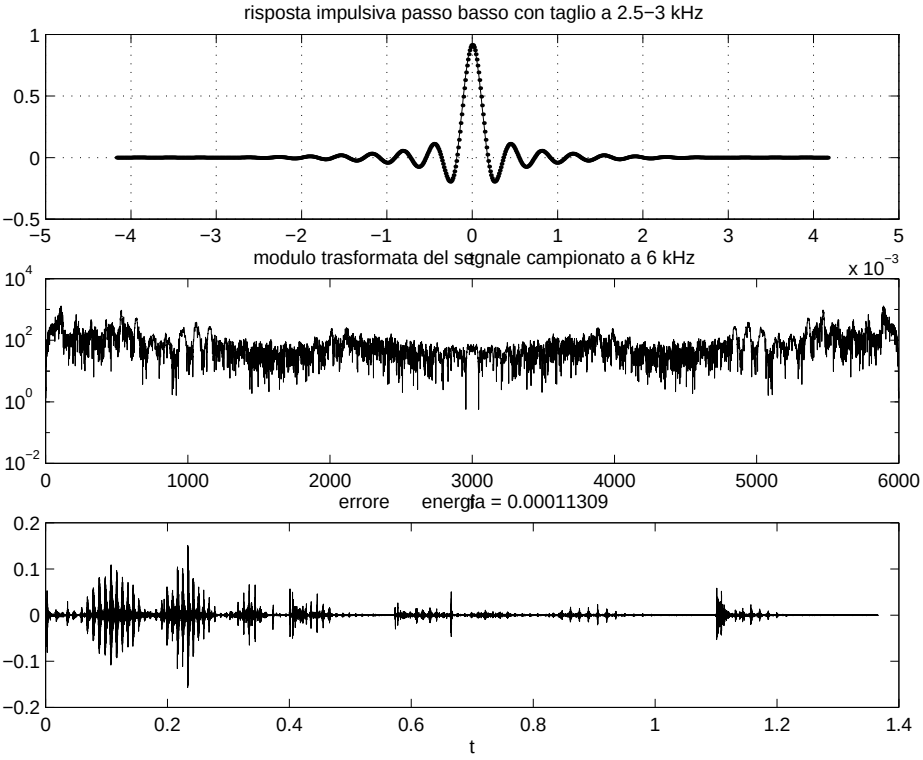
- di nuovo filtro non ideale per ricostruire da un segnale sottocampionato a 24 kHz, (notare 4 campioni nel semilobo principale 96/24=4)
- è scomparso il plateu di rumore, fmax=12kHz
- l'energia dell'errore (6 10^{-8}) cresce di 2 ordini di grandezza rispetto a prima.



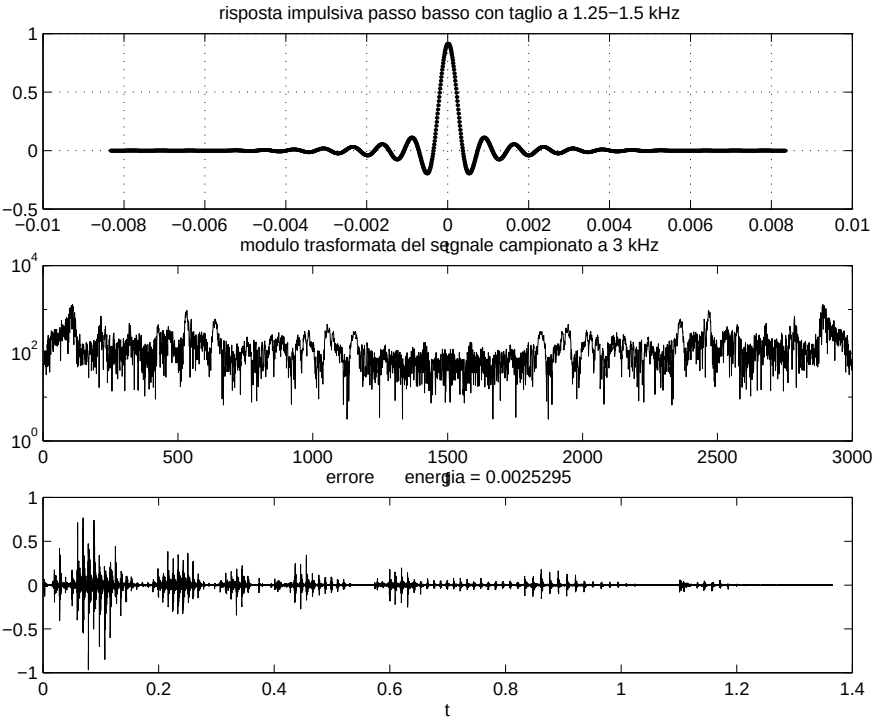
- di nuovo filtro non ideale per ricostruire da un segnale sottocampionato a 12 kHz, (notare 8 campioni nel semilobo principale 96/12=8)
- fmax = 6 kHz
- l'energia dell'errore (10^{-6}) cresce ancora di 2 ordini di grandezza rispetto a prima, ma rimane lo 0.1 per mille del segnale originale. Notare che cambia la forma dell'errore... inizia a seguire il segnale: distorsione



- filtro non ideale per ricostruire da un segnale sottocampionato a 6 kHz, (16 campioni nel semilobo principale)
- fmax = 3 kHz: ormai anche il secondo plateu che era segnale è stato tagliato
- l'energia dell'errore (10^{-4}) cresce ancora di 2 ordini di grandezza rispetto a prima, ma rimane l'1% del segnale originale. Ormai l'errore segue nettamente il segnale: distorsione



- filtro non ideale per ricostruire da un segnale sottocampionato a 3 kHz
- fmax = 1.5 kHz: la trasformato non va più neanche lontanamente a zero
- l'energia dell'errore ($2.5\ 10^{-3}$) è circa 1/4 di quella del segnale originale. Ormai l'errore segue nettamente il segnale: distorsione



ESERCIZI

1. Antitrasformare la seguente H(f):

$$H_1(f)=\begin{cases} T_c & |f|<f_c/6 \\ T_c\left(1-3T_c\left(|f|-f_c/6\right)\right) & f_c/6<|f|<f_c/2 \\ 0 & |f|>f_c/2 \end{cases}$$

   Ripetere la ricostruzione del segnale x(t) dell'esempio 1, campionato con Tc=40 ms utilizzando come forme interpolanti la h(t) ottenuta. Per ottenere un passo T = 5ms per h(t), ed un intervallo di troncamento sicuramente sufficiente, si consiglia di definire H(f) con passo v=0.1 Hz e 2000 punti (quindi tra -100 e 99.9 Hz).

   Rispondere a priori:
   - h(t) interpola correttamente x(t)? Verificare con SCILAB.
   - Se no, a quale Tc inizia a funzionare? Verificare con SCILAB.

2. Antitrasformare la seguente H(f), e procedere come nell'esercizio 1:

$$H_2(f)=\begin{cases} T_c & |f|<f_c/3 \\ T_c\left(1-6T_c\left(|f|-f_c/3\right)\right) & f_c/3<|f|<f_c/2 \\ 0 & |f|>f_c/2 \end{cases}$$

3. Antitrasformare la seguente H(f), e procedere come nell'esercizio 1:

$$H_3(f)=\begin{cases} T_c & |f|<f_c/6 \\ T_c\cos^2\left(\frac{3}{2}\pi T_c\left(|f|-\frac{f_c}{6}\right)\right) & f_c/6<|f|<f_c/2 \\ 0 & |f|>f_c/2 \end{cases}$$