Indice

1 Domini numerici

I domini numerici sono particolari insiemi infiniti, contenenti pertanto elementi infiniti raggruppati in base a determinate caratteristiche comuni.

L'insieme più piccolo è l'insieme dei *numeri naturali*, anche indicato con $\mathbb{N}$. Appartenente a questo insieme sono i numeri interi positivi:

$$\mathbb{N} = \{0, 1, 2, 3, \ldots\}^1$$

A seguire, l'insieme dei *numeri interi*, anche indicato con $\mathbb{Z}$, comprende, oltre che l'insieme dei numeri naturali, anche i numeri negativi:

$$\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$$

L'insieme dei numeri *razionali*, anche indicato con $\mathbb{Q}$, contiente tutti quei numeri che possono essere espressi sottoforma di frazione. Per cui comprende un numero infinito di elementi tra ogni coppia di numeri interi

$$\mathbb{Q} = \{\ldots, -\frac{1}{3}, \ldots, -\frac{1}{2}, \ldots, -1, \ldots, 0, \ldots, 1, \ldots, \frac{1}{2}, \ldots, \frac{1}{3}, \ldots\}$$

Questi insieme non sono sufficienti a descrivere ogni tipo di numero. Esistono pertanto dei numeri che non possono essere espressi sotto forma di frazione (come per esempio $\sqrt{2}$). Allo scopo di rappresentare questo tipo di elementi, che possono essere nominati *numeri irrazionali* ($\mathbb{I}$), viene introdotto l'insieme dei *numeri reali*, anche indicato con $\mathbb{R}$. Questo insieme rappresenta contemporaneamente i numeri razionali e i numeri irrazionali.

Infine esistono ulteriori casi che non possono essere rappresentati nemmeno dall'insieme dei numeri reali. Esemplare è l'equazione di secondo grado $x^2+1 = 0$. Tale equazione non ha una soluzione nell'insieme dei numeri reali poiché $x^2 = -1$ non è un numero reale. Infatti, ogni numero reale, elevato al quadrato, restituisce un numero positivo. Così, allo scopo di risolvere questa equazione viene introdotto l'unità immaginaria i che identifica l'insieme dei numeri complessi, anche indicato con $\mathbb{C}$.

Carl Friedrich Gauss definisce concluso il problema dei numeri con l'insieme dei numeri complessi, poiché con questi si risolve ogni problema matematico.

Lo studio del segnale audio discreto avviene principalmente all'interno del dominio dei numeri complessi. Questo è dovuto ad alcune loro proprietà che si vedranno di seguito.

2 Proprietà fondamentali dei numeri complessi

L'unità immaginaria i ha il merito di risolvere l'equazione di secondo grado $x^2 + 1 = 0$ con il conseguente risultato:

$$x = \pm i$$

Pertanto, elevata al quadrato restituisce -1:

$$i^2 = -1$$

```
octave:#> sqrt(-1)
ans = 0+1i
```

[1] Lo *zero* fu introdotto nella nomenclatura occidentale solo nel XII secolo d.C. Il numero deriva dall'algebra orientale (dall'Arabia) e arrivò in occidente per mano del matematico italiano Leonardo Fibonacci.

la struttura dell'unità immaginaria è data dalla combinazione di una parte reale e una parte immaginaria

$$a + ib$$

dove a è la parte reale e ib la parte immaginaria.

```
octave:#> real(i)
ans = 0
octave:#> imag(i)
and = 1
```

Un'importante proprietà dell'unità immaginaria è la ciclicità dell'elevazione a potenza

$$i^0 = 1$$

$$i^1 = i$$

$$i^2 = -1$$

$$i^3 = -i$$

$$i^4 = 1$$

$$i^5 = i$$

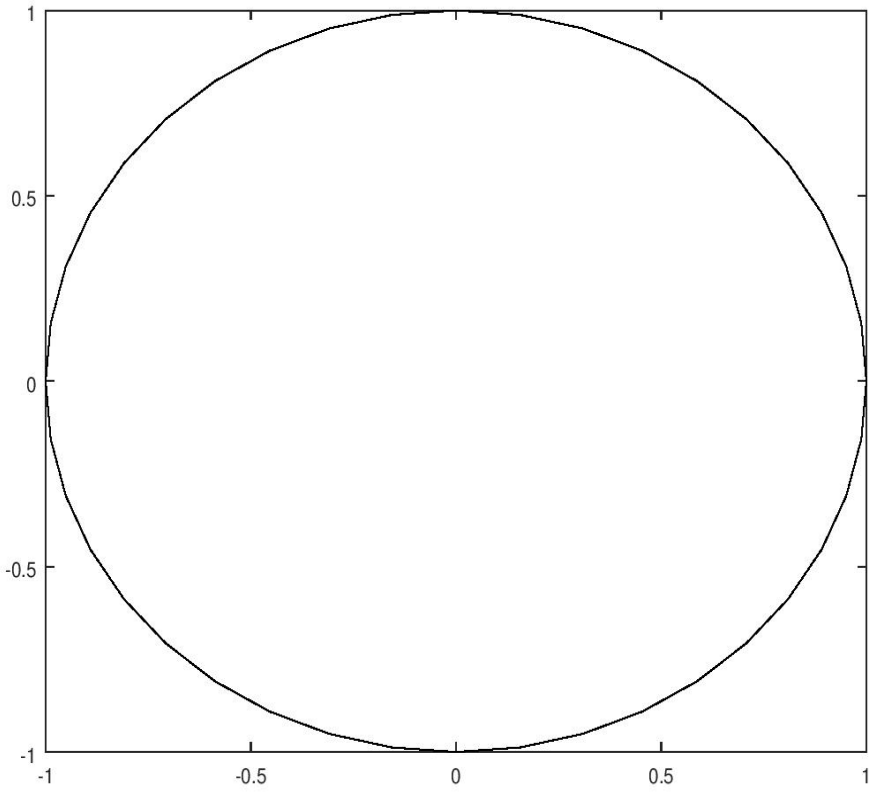
$$\dots$$

```
octave:#> i^0
ans= 1
octave:#> i^1
ans = 6.1232e-17 + 1.0000e+00i # approssimabile a 0 + 1i
octave:#> i^2
ans = -1.0000e+00 + 1.2246e-16i # approssimabile a -1+ 0i
octave:#> i^3
ans = -1.8370e-16 - 1.0000e+00i # approssimabile a 0 -1i
```

È possibile descrivere il numero complesso nel piano cartesiano. In questo caso le ascisse rappresenteranno la parte reale del numero e le ordinate la parte immaginaria.

Sfruttando la proprietà dell'elevazione a potenza del numero complesso è possibile rappresentare una circonferenza sul piano cartesiano

```
%PLOTTING CIRCONFERENZA
octave:#> potenze=0:0.1:4; # potenza da 0 a 4 con passo di 0.1
octave:#> cerchio =i.^potenze;
octave:#> plot(cerchio)
```



3 Formula di Eulero

La proprietà della ciclicità dell'elevazione a potenza rende evidente una relazione dei numeri complessi con la trigonometria. Questa relazione viene espressa dalla *formula di Eulero*. Attraverso questa formula è possibile rappresentare il numero complesso con funzioni trigonometriche.

$$e^{i\alpha} = \cos\alpha + i \cdot \sin\alpha$$

dove e è il *numero irrazionale di Nepero* (2,7183...) e α è un generico angolo in radianti. La dimostrazione più diffusa che verifica tale relazione è basata sullo sviluppo in serie di Taylor della funzione esponenziale (che non verrà trattata in questa sede).

Il risultato riportato da questa formula è un numero immaginario costituito dal coseno dell'angolo α, che rappresenta la parte reale, e dal seno dello stesso angolo che invece rappresenta la parte immaginaria.

```
octave:#> gradi = 30; %angolo in gradi
octave:#> alfa = gradi*pi/180 %angolo in radianti alfa = 0.52360
octave:#> eul_re = real(eul) %parte reale
eul_re = 0.86603
octave:#> eul_im = imag(eul) %parte immaginaria eul_im = 0.50000
octave:#> cos(alfa)
ans = 0.86603
octave:#> sin(alfa)
ans = 0.50000
```

Da questa relazione è possibile ricavare il coseno e il seno dell'angolo α.

Quando si somma al numero complesso $e^{i\alpha}$ il suon coniugato[2] $e^{-i\alpha}$, la parte immaginaria viene annullata, pertanto si estrarrà la parte reale, cioè il coseno:

$$\cos\alpha = \frac{e^{i\alpha} + e^{-i\alpha}}{2}$$

Nel caso inverso, e cioè quando al numero complesso viene sottratto il suo coniugato, verrà annullata la parte reale, estraendo quindi la parte immaginaria $i \cdot \sin\alpha$. Quindi, per ottenere il seno dell'angolo α si dovrà eliminare l'unità immaginaria dividendo per i.

$$\sin\alpha = \frac{e^{i\alpha} - e^{-i\alpha}}{2i}$$

Per rappresentare il segnale audio con i numeri complessi occorre variare l'angolo α in funzione del tempo t (come nel caso delle funzioni trigonometriche quando viene generato il segnale reale[3]). Pertanto, aggiungendo la velocità angolare[4] ω si definisce un segnale sinudoidale come:

$$e^{i\omega t}$$

[2]In matematica, si definisce complesso coniugato di un numero complesso il numero ottenuto dal primo cambiando il segno della parte immaginaria. Pensando il numero complesso come punto del piano complesso, il suo complesso coniugato è il punto riflesso rispetto all'asse reale. Il prodotto di un numero complesso per il suo complesso coniugato è sempre un numero reale positivo, ed è il modulo quadro del numero complesso.

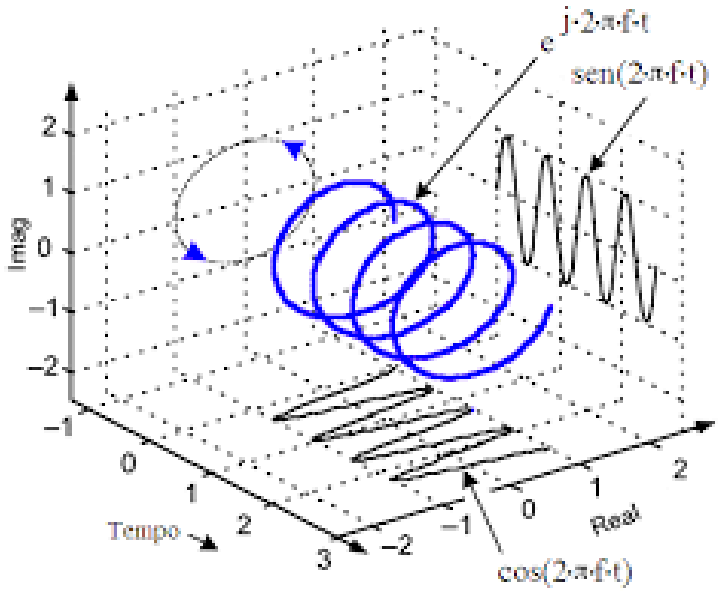
$$(a + ib)(a - ib) = a^2 - (ib)^2 = a^2 + b^2$$

[3]Per la generazione del segnale rale occorre variare l'angolo della funzione coseno

$$\cos(\omega * t)$$

[4]La velocità angolare, indicata normamlmente con ω (omega) è definita come l'arco percorso nel tempo e si misura in radianti/secondi. La relazione tra la frequenza f e la velocità angolare ω è:

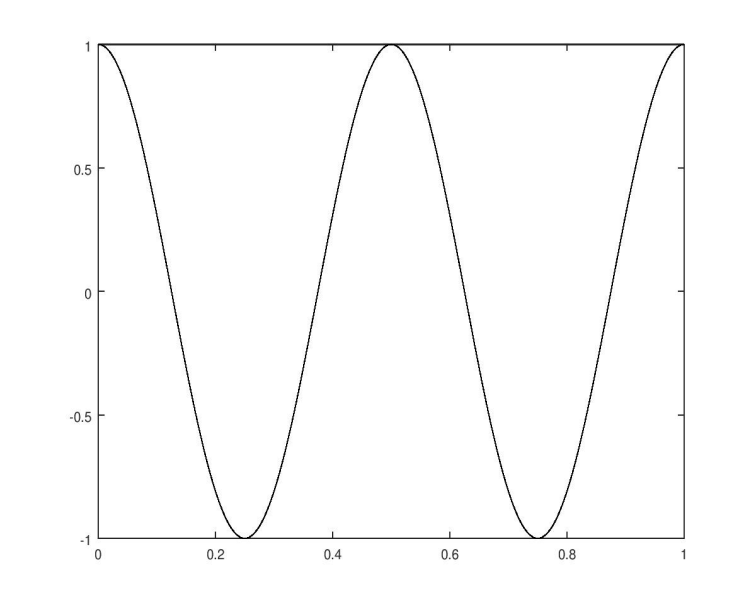
$$\omega = 2\pi f$$



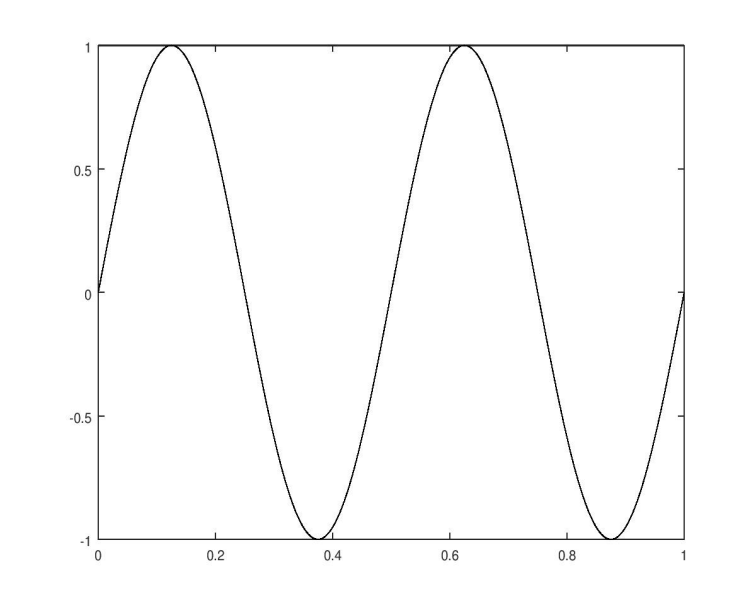

```
%Plotting di un segnale complesso

octave:#> fc = 1000;
octave:#> pc = 1/fc;
octave:#> t = [0:pc:1];
octave:#> f = 2; %Hz
octave:#> w = 2*pi*f; %velocita' angolare
octave:#> z = e.^(i*w*t);
```

```
%il segnale z contenente numeri complessi
z =
Columns 1 through 3:
1.00000 + 0.00000i   0.99992 + 0.01257i   0.99968 + 0.02513i
Columns 4 through 6:
0.99929 + 0.03769i   0.99874 + 0.05024i   0.99803 + 0.06279i
Columns 7 through 9:
0.99716 + 0.07533i   0.99613 + 0.08785i   0.99495 + 0.10036i
Columns 10 through 12:
0.99361 + 0.11286i   0.99211 + 0.12533i   0.99046 + 0.13779i
Columns 13 through 15:
0.98865 + 0.15023i   0.98669 + 0.16264i   0.98456 + 0.17502i
Columns 16 through 18:
..ecc..
```

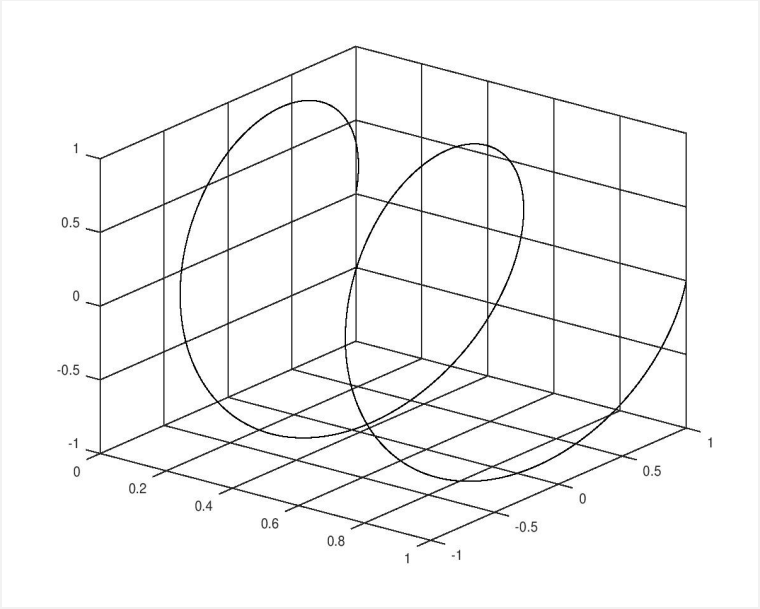
```
octave:#> plot(t, real(z))
```



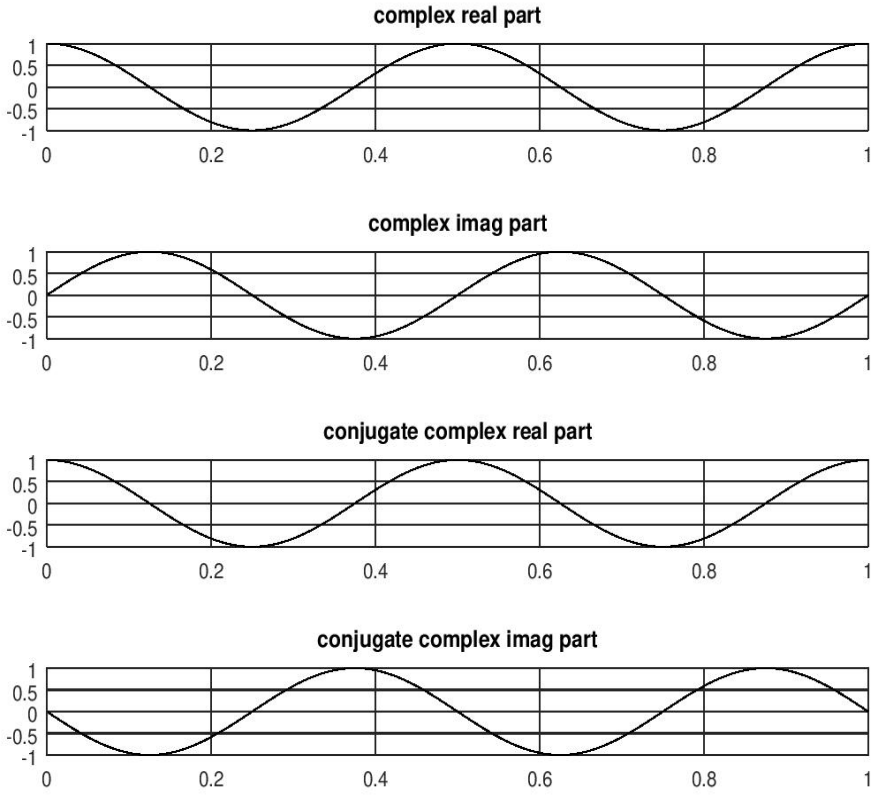

```
octave:#> plot(t, imag(z))
```




```
octave:#> plot3(t, z)
```



È interessante inoltre osservare le differenze tra il segnale complesso e il suo coniugato $e^{-i\omega T}$



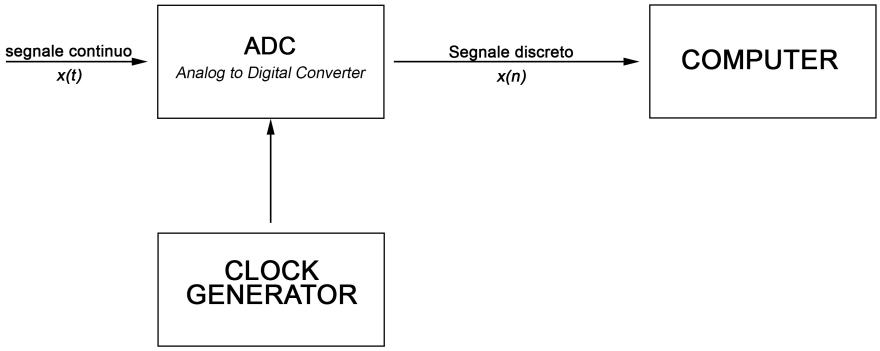
Qui è possibile notare che la parte reale rimane invariata mentre diventa di segno opposto la parte immaginaria.

L'identità di Eulero è di fondamentale importanza quando si svolge l'analisi del segnale audio discreto. Un chiaro esempio è la trasformata discreta di Fourier (*DFT – Discrete Fourier Transform*) che si vedrà di seguito.

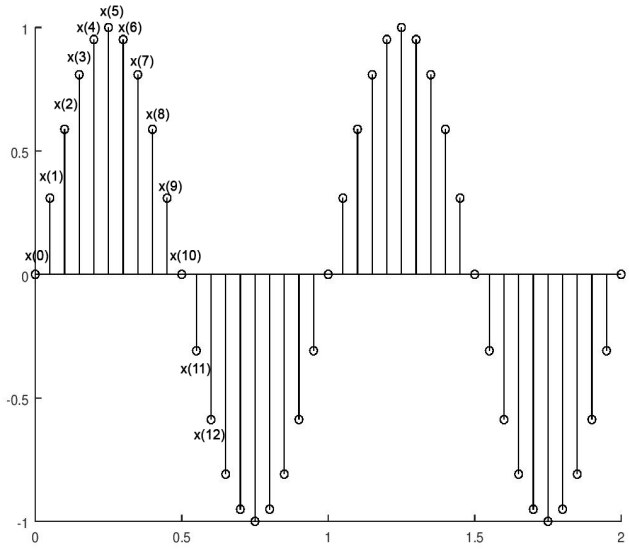
4 Il campionamento e le sue problematiche

Qualora si parli del segnale audio discreto è importante conoscere le problematiche che sorgono nel processo di discretizzazione.

La prima cosa da conoscere è il processo di campionamento



Tale processo trasforma un segnale x continuo nel tempo *(t)* in un numero di campioni *(n)* che lo rappresentano per intervalli discreti con ampiezze discrete. La variabile n è un numero intero che identifica l'n-esimo campione di x nel tempo. In alter parole *x(n)* rappresenta l'ampiezza di *x(t)* per ogni n-esimo istante.

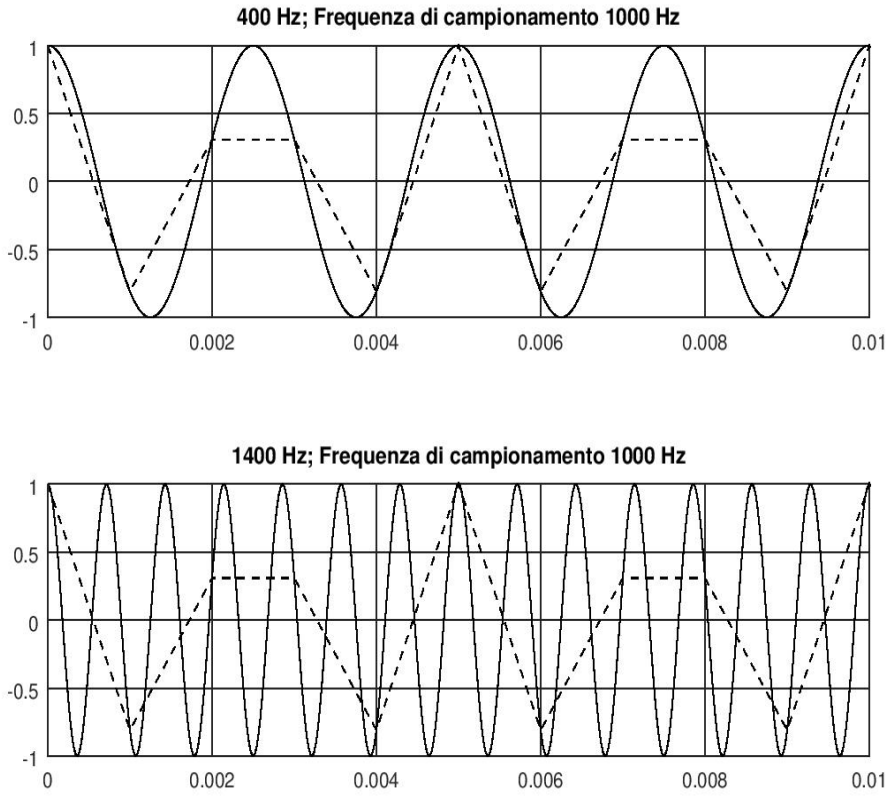


La quantità di valori estratti dall'ADC nell'unità di tempo è la *frequenza* di campionamento che indica in altre parole gli n valori di *x(t)* estratti al secondo. La distanza in unità di tempo

tra un campione ed un altro, cioè il reciproco della frequenza di campionamento, è il *periodo di campionamento*(espresso in secondi).

Le problematiche trattate in questa sede nascono nel momento in cui si studia il segnale campionato nel dominio della frequenza. Una di queste problematiche viene chiamata l'*ambiguità del dominio della frequenza*.

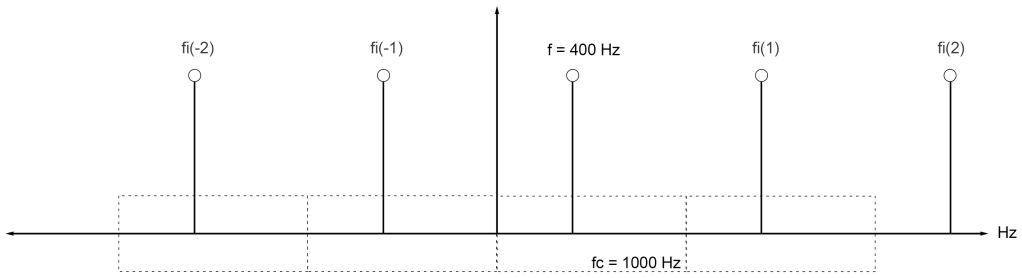
La figura sottostante mostra il campionamento di due diverse sinusoidi entrambe campionate ad una frequenza di campionamento a 1000 Hz. Nella figura si può visualizzare il prodotto della discretizzazione del segnale, rappresentato da una linea tratteggiata. Quello che si nota è che entrambe le sinusoidi sono campionate nello stesso modo, per cui il segnale *x(n)* risulterà identico sia per una sinusoide di 400 Hz (la prima) che per un'altra a 1400 Hz(la seconda).



Il segnale discreto prodotto dal campionamento a 1000 Hz della sinusoide a 400 Hz sarà dato non solo dai due segnali appena visti, ma da tutte quelle frequenze ottenute dalla somma tra la frequenza di 400 Hz con un multiplo della frequenza di campionamento:

$$f_i(k) = f + k \cdot fc$$

Dove f è la frequenza (nell'esempio 400 Hz) per la quale si ottiene un determinato campionamento a frequenza fc (frequenza di campionamento); k è un numero intero che moltiplicherà la fc; Infine $f_i(k)$ è la cosidetta frequenza immaginaria, ossia quella frequenza che restituisce nella discretizzazione lo stesso risultato di f. Pertanto si possono avere infinite frequenze immaginarie:



In base a questo principio, nel dominio della frequenza si riscontreranno infinite repliche di quanto è stato campionato. Tali repliche si succedono ad intervalli stabiliti dalla frequenza di campionamento. Nel caso della sinusoide a 400 Hz si avrà:

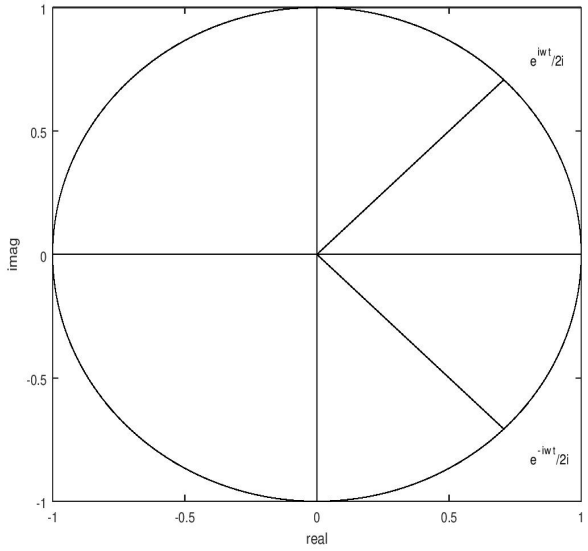
$$f_i(k) = 400 + k \cdot 1000 \begin{Bmatrix} \cdots \\ f_i(2) = 2400Hz \\ f_i(1) = 1400Hz \\ f_i(0) = 400 \\ f_i(-1) = -600 \\ \cdots \end{Bmatrix}$$

Un'altra problematica della rappresentazione nel dominio delle frequenze sorge dalle così dette *frequenze negative*. Per comprendere tale concetto si riporterà in studio l'idtentità di Eulero.

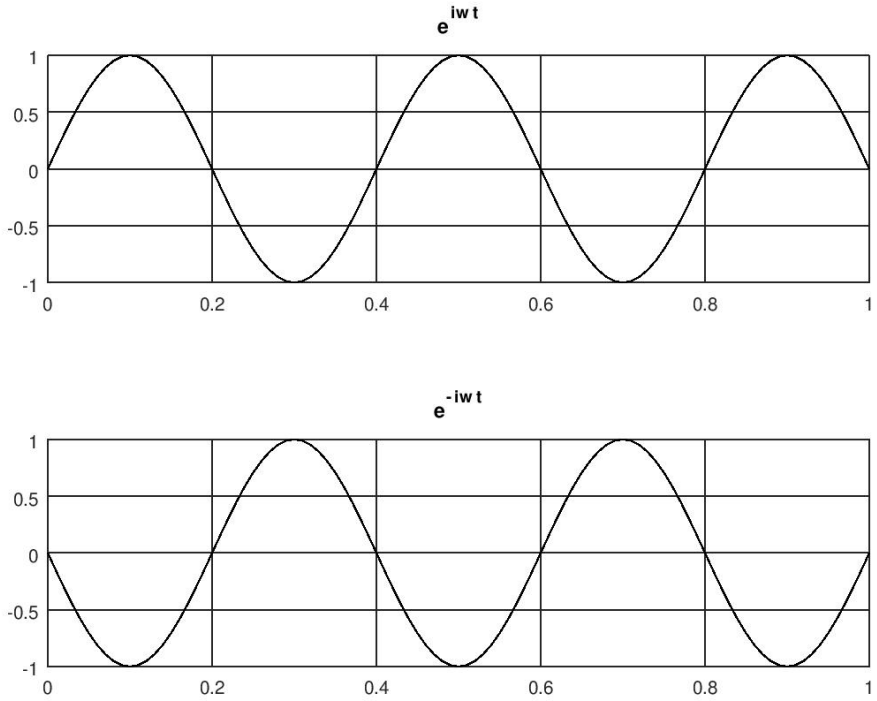
Come si è visto in precedenza, la funzione trigonometrica reale della sinusoide può essere descritta attraverso i numeri complessi secondo la terza identità di Eulero:

$$\sin(\omega T) = \frac{e^{i\omega T} - e^{-i\omega T}}{2i}$$

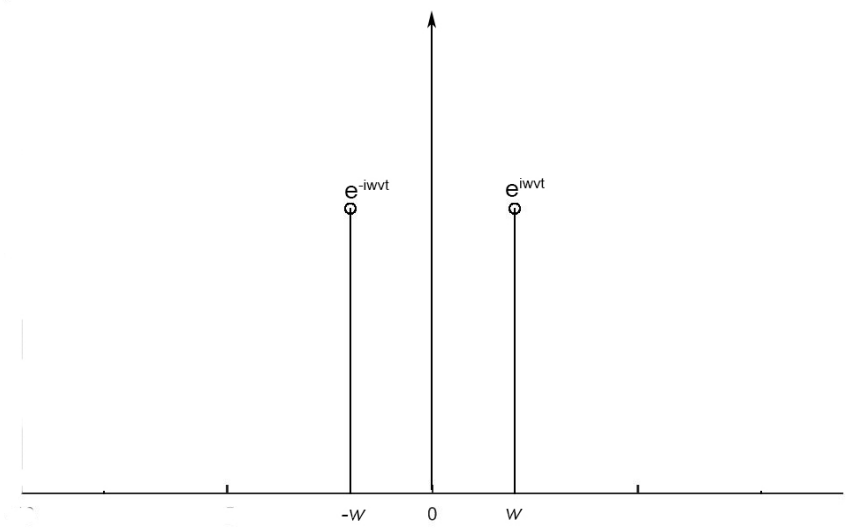
Nella rappresentazione grafica tale identità può essere visualizzata con due punti che ruotano intorno ad una circonferenza, uno in senso antiorario($e^{i\omega T}$) e uno i senso orario ($e^{-i\omega T}$).



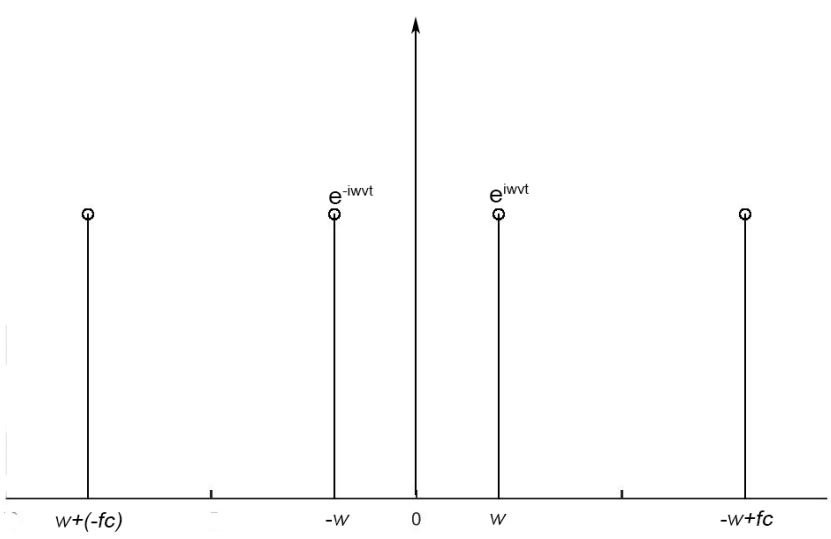
Pertanto variando la velocità angolare ω nel tempo t, in entrambi i punti, si otterrà una rappresentazione nel dominio del tempo. Il risultato sono due diverse sinusoidi.



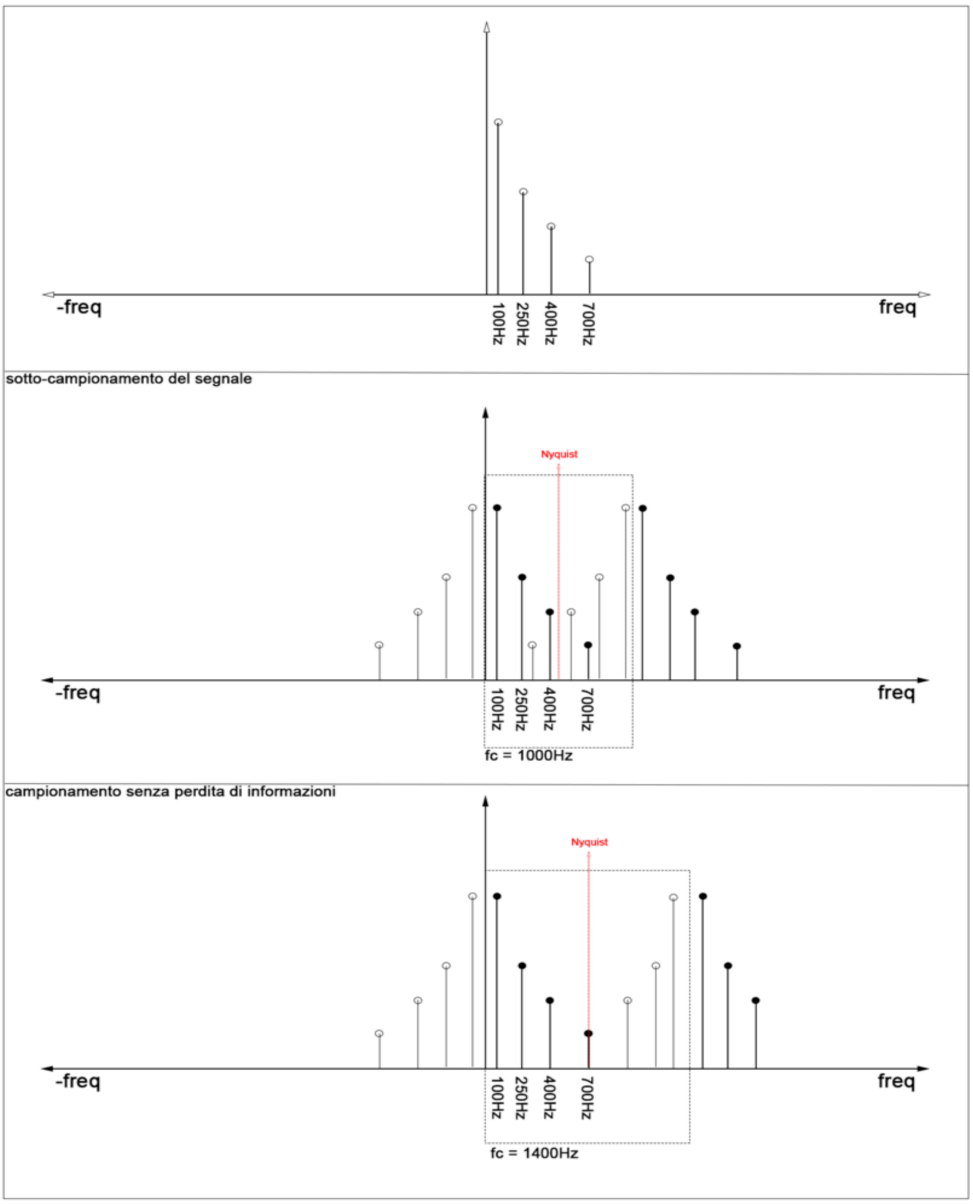
Il punto $e^{-i\omega T}$, ruotando nel senso opposto (quindi in senso orario), genera nel tempo t una frequenza negativa. Pertanto nel dominio della frequenza si avrà:



Ritornando ora all'esempio della sinusoide a 400 Hz campionata a 1000 Hz e aggiungendo alla prima problematica introdotta (l'ambiguità del dominio della frequenza) quest'ultima studiata (la frequenze negative), si otterrà uno spettro del segnale come segue.



Come si vede dalla figura soprastante anche le frequenze di segno negativo generano ulteriori frequenze immaginarie. Questo spiega perché la frequenza di campionamento deve essere maggiore o tuttalpiù uguale al doppio della componente frequenziale più alta del segnale *x(t)* (questo concetto prende il nome di *teorema di Nyquist*). La frequenza che sta a metà della frequenza di campionamento è chiamata *frequenza di Nyquist*. Nelle figure sottostanti si vede il campionamento di un segnale con quattro componenti. La componente frequenziale più alta è 700 Hz. Nella seconda figura è illustrato un sotto-campionamento, con perdita di informazione, a 1000 Hz (questo effetto è anche chiamato *aliasing*). Nella terza immagine è possibile visualizzare invece un campionamento senza perdita di informazione, a 1400 Hz (700 Hz x 2).



5 Trasformata di Fourier

5.1 Trasformata di Fourier (*FT - Fourier Transform*)

Prima di iniziare a studiare la *DFT* è importante conoscere cosa si intende per *Trasformata di Fourier* (*FT*). Questa trasformata permette di passare dalla rappresentazione del segnale nel dominio del tempo, alla rappresentazione nel dominio della frequenza. Pertanto, con questa, è possibile scomporre una segnale nelle sue componenti. Nel segnale audio permette di estrarre le componenti pure (sinusoidi) che compongono un suono complesso.

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) \cdot \mathrm{e}^{-i\omega t} \mathrm{d}t, \omega \in (-\infty, +\infty)$$

Quindi, un segnale continuo nel tempo *x(t)* può essere scomposto in infinite componenti frequenziali *X(ω)*.

$$x(t) \rightarrow X(\omega)$$

Tuttavia tale trasformata è possibile solamente ad un livello teorico, poiché non è possibile scomporre in infinite sinusoidi, $\omega \in (-\infty, +\infty)$, un segnale infinito nel tempo, $t \in (-\infty, +\infty)$. Pertanto la *trasformata di Fourier* è applicabile solamente per segnali non casuali, che non hanno origine né fine, e periodici, che si ripetono sempre identici.

5.2 Trasformata Discreta di Fourier (*DFT - Discrete Fourier Transform*)

La *trasformata discreta di Fourier* (*DFT*) consente di applicare ad un livello pratico la *FT*. La *DFT* applica lo stesso concetto visto sopra, ma questa volta impiegato nei segnali discreti (quindi non più infiniti nel tempo bensì finiti). Tale trasformata sostituisce l'integrale infinito della *FT* con una sommatoria finita. Disponendo di una sequenza di lunghezza finita composta da N valori $x(n), n = 0, 1, \ldots, N-1$, corrispondenti a campioni di un segnale *x(t)* prelevati con ritmo $fc = 1/tc$, si indica come *Discrete Fourier Transform - DFT* la nuova sequenza:

$$X(k) = \sum_{n=0}^{N-1} x(n) \mathrm{e}^{-i\omega_k n}, k = 0, 1, 2, \ldots, N-1$$

dove *x(n)* è l'n-esimo campione del segnale discreto x; ω_k è la frequenza analizzata e quindi $\mathrm{e}^{-i\omega_k n}$ è l'n-esimo campione del segnale complesso ricercato in x.

In questo caso un segnale finito nel tempo $n \in (0, N-1)$ è scomposto in finte frequenze $k \in (0, N-1)$

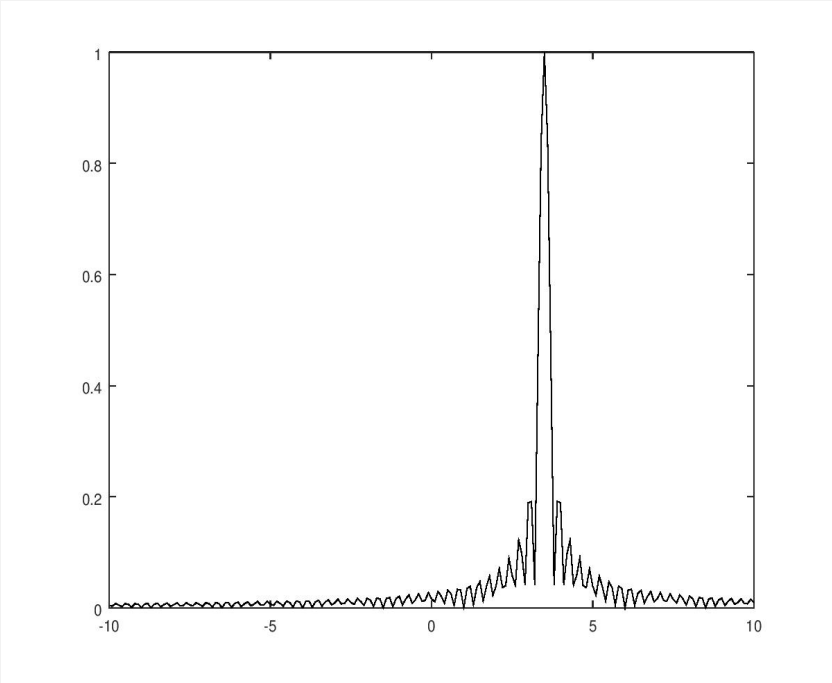
La trasformata discreta di Fourier non è altro che una serie di prodotti scalari tra il segnale e una serie si sinusoidi a frequenza k. Il prodotto scalare, nel calcolo vettoriale, associa ad una coppia di vettori uno scalare che identifica la similarità tra i due.

La formula del prodotto scalare è come segue:

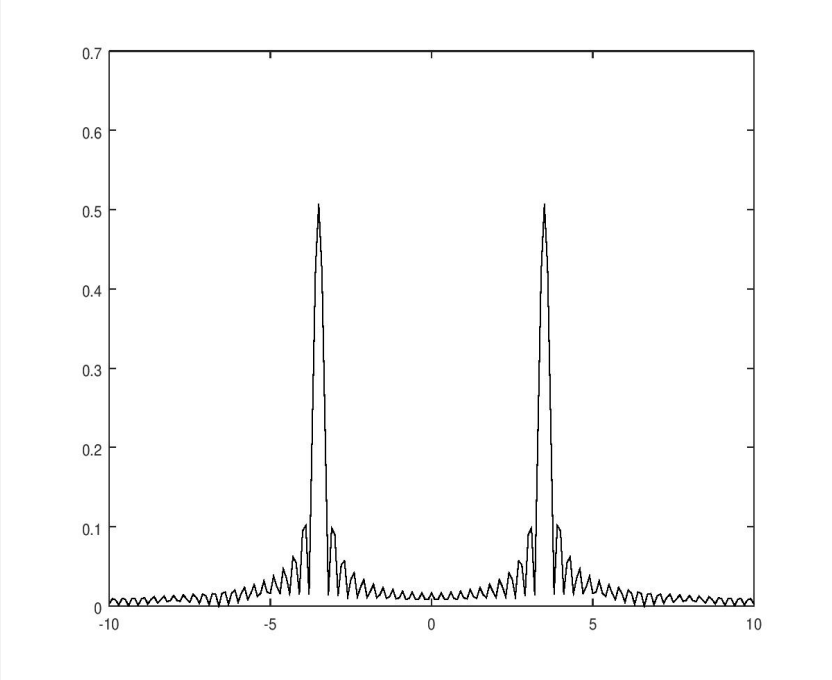
$$\langle v, w \rangle = \sum_{n=0}^{N-1} v_n \overline{w_n}$$

Con il trattino sopra il vettore w si indica il coniugato. Pertanto il prodotto scalare indicherà quanto w sarà presente in v. Sostituendo il vettore v con il segnale x, e, di conseguenza, il vettore w con la frequenza $\mathrm{e}^{-i\omega_k n}$ ricercata nel segnale analizzato, si può notare l'analogia con la *DFT*.

```
%DFT con segnale complesso non casuale

%frequenza in radianti della sinusoide
octave:#> w = 3.5;
%tempo in cui  e' definita la funzione (sinusoide)
octave:#> t = [-10:0.1:10];
%funzione che costruisce la sinusoide
octave:#> z = e.^(i*w*t);
%definisce il dominio della frequenza
octave:#> F = [-10:0.1:10];
%costituisce un array di soli zero di grandezza uguale all'array F
octave:#> dft = zeros(size(F));
%inizializza un ciclo for da 1 alla grandezza dell'array F
octave:#> for k = 1:size(F,2)
> fa = F(k); %analisi frequenza per frequenza all'interno del ciclo
> za = z.*e.^(-i*fa*t); %riempie l'array dft
> dft(k) = abs(sum(za))/size(F,2); %chiude il ciclo for
> endfor
octave:#> plot(F, dft);
```



```
%DFT con segnale reale non casuale

octave:#> w = 3.5;
octave:#> t = [-10:0.1:10];
octave:#> z = cos(w*t);
octave:#> F = [-10:0.1:10];
octave:#> dft = zeros(size(F));
octave:#> for k = 1:size(F,2)
> fa = F(k);
> za = z.*e.^(-i*fa*t);
> dft(k) = abs(sum(za))/size(t,2);
> endfor
octave:#> plot(F, dft);
```



Nei precedenti codici si sono viste due diverse applicazioni della *DFT*, la prima con segnale complesso (e^{i*w*t}), mentre la seconda con segnale reale($\cos(w*t)$). Nella seconda si è riscontrata un'anomalia, che è stata già affrontata nei capitoli precedenti, ossia il problema delle frequenze negative.

Nel segnale complesso, il suo coniugato presenta la stessa parte reale (quindi il coseno) mentre la parte immaginaria (seno) di segno opposto. Pertanto nella moltiplicazione del numero complesso con il suo coniugato la parte immaginaria verrà annullata:

$$e^{i\alpha} \cdot e^{-i\alpha} = (\cos\alpha + i \cdot \sin\alpha) \cdot (\cos\alpha - i \cdot \sin\alpha) =$$

$$= (\cos\alpha)^2 - i\cos\alpha\sin\alpha + i\cos\alpha\sin\alpha - i^2(\sin\alpha)^2 =$$

$$= (\cos\alpha)^2 - i^2(\sin\alpha)^2 = (\cos\alpha)^2 - (-1) \cdot (\sin\alpha)^2 =$$

$$= (\cos\alpha)^2 + (\sin\alpha)^2 = 1$$

Per cui, eliminata l'unità immaginaria (poiché $i^2 = -1$) si otterrà, secondo le identità della trigonometria (per la quale $\cos\alpha^2 + \sin\alpha^2 = 1$), un numero reale uguale ad 1.

Questo non accade con un segnale reale. La cosinusoide reale non avrà infatti la parte immaginaria. Tale mancanza non permetterà di eliminare la parte negativa e immaginaria del coniugato $e^{-i\alpha}$, che si aggiungerà al risultato della *DFT*.

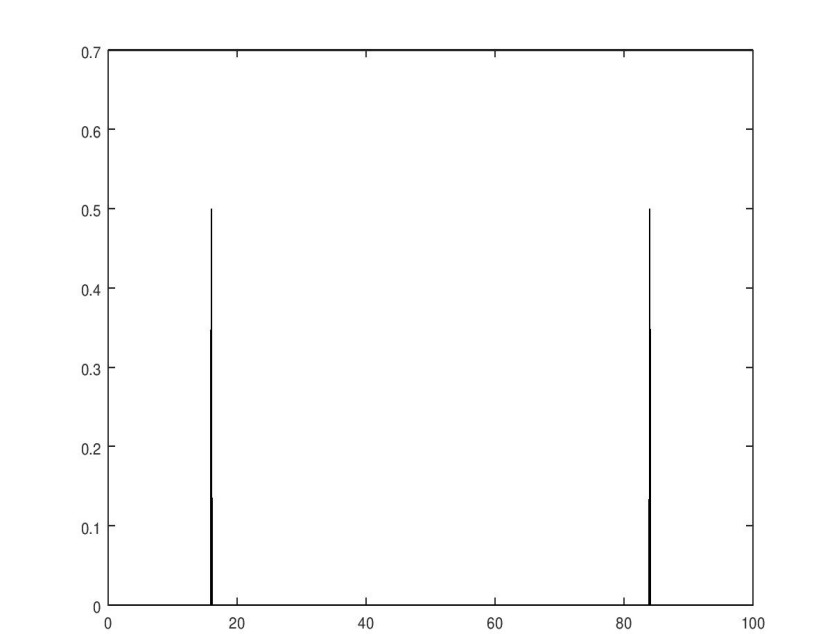
Secondo una delle già studiate identità di Eulero è possibile vedere il comportamento del coseno nel dominio dei numeri complessi che presenta due componenti, $e^{i\alpha}$ e il suo coniugato $e^{-i\alpha}$, ad ampiezza dimezzata.

$$\cos\alpha = \frac{e^{i\alpha} + e^{-i\alpha}}{2}$$

Inoltre, nei codici sono stati trattati dei segnali di tipo *non casuale*. Un segnale di questo genere è un segnale ideale che viene preso ancor prima dello zero nel dominio del tempo e va da $-\infty$ a $+\infty$. In questo modo, anche la *DFT* sarà a sua volta intorno allo zero. Per cui consente di vedere anche le frequenze negative.

Di seguito si vedranno segnali di tipo *casuale*, e che quindi iniziano in un punto preciso. Questo genere di segnali sono dipendenti dal tempo e si esprimono attraverso una funzione $f > 0$. Rappresentano segnali realistici e sono quelli su cui si applica generalmente la *DFT*.

```
%DFT di un segnale reale casuale

octave:#> fc = 100; %frequenza di campionamento in Hz
octave:#> pc = 1/fc;     %periodo di campionamento
octave:#> f = 16; % in Hz (100 radianti)
octave:#> w = f*2*pi;
octave:#> t = [0:SampInc:100]; % ci sono 10.000 campioni
octave:#> z = cos(w*t);
octave:#> BinSize = 0.1;%analizziamo le frequenze ad 1\10 di distanza
octave:#> F = [0:BinSize:fc-BinSize];
octave:#> dft = zeros(size(F));
octave:#> for k = 1:size(F, 2)
> wa = F(k)*2*pi;
> za = z.*e.^(-i*wa*t);
> dft(k) = abs(sum(za))/size(t,2);
> endfor

plot(F, dft);
```

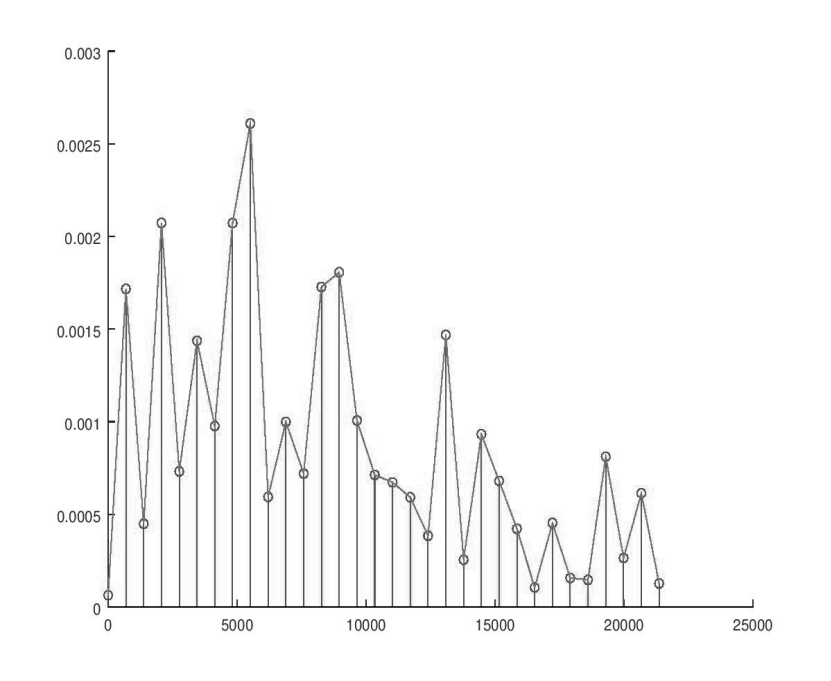


Il codice riportato sopra mostra anche le problematiche affrontate nel capitolo precedente. Ora la frequenza negativa non è più mostrata, poiché la *DFT* è applicata solo per le frequenze positive. Tuttavia ritroviamo la sua replica a 84 Hz, oltre la frequenza di Niquist a 50 Hz (la metà della frequenza di campionamento a 100 Hz).

Nel codice successivo si presenterà una situazione più realistica, in cui si applicherà la *DFT* in un campione audio.

```
%DFT di un campione audio

octave:#> clear all
octave:#> close all
octave:#> [y, fs] = audioread('../audio/campione_mono.wav');
octave:#> pc = 1/fs;
octave:#> nSamp = size(y, 1);
octave:#> dur = pc*nSamp;
octave:#> t = [0:pc:dur-pc]';

%normalizzazione del segnale y
octave:#> y = y*(1/max(y))*(10^(-(1/20)));

octave:#> winSize = 64;
octave:#> binSize = fs/winSize

%Rotazione del vettore (con l'apice finale)
%da vettore riga a vettore colonna
octave:#> F = [0:binSize:fs-binSize]';
```

```
octave:#> for k = 1:size(F, 1)/2
> fa = F(k)*2*pi;
> za = y.*e.^(-i*fa*t);
> dft(k, 1) = (abs(sum(za))/size(t,1))*2;
> endfor

octave:#> stem(F(1:size(dft, 1),1), dft);
octave:#> hold on
octave:#> plot(F(1:size(dft, 1),1), dft);
octave:#> hold off
```



Nel codice precedente si sono dichiarate per la prima volta due importanti variabili della *DFT*, *winSize* e *binSize*. Queste variabili definiscono molto semplicemente la risoluzione frequenziale della trasformata discreta di Fourier. La grandezza della finestra definisce la risoluzione mentre la grandezza del *bin* definisce la distanza tra le frequenze analizzate. Come si vede nel codice, queste due variabili sono legate alla frequenza di campionamento.

Si può anche notare che la figura riportata dal codice presenta solamente 32 valori, distanziati tra loro di 689.06 Hz (fs/winSize). Infatti, nel ciclo *for* si è applicata la *DFT* solamente ad una metà del segnale, tralasciando quindi le frequenze oltre Nyquist.

5.3 Finestratura

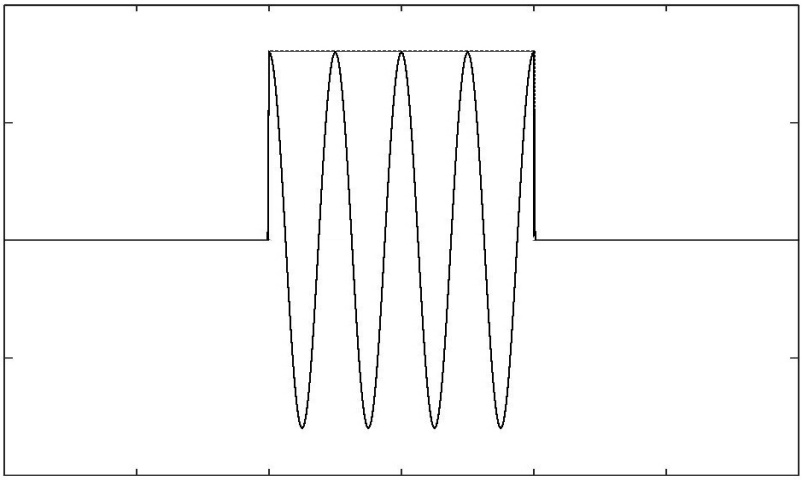
Passare dalla *FT* alla *DFT* porta tuttavia a delle problematiche. Se la *FT* si occupa di segnali continui ed infiniti nel tempo, la *DFT* ci consente di analizzare segnali discretizzati e limitati. Pertanto la sequenza dei campioni di un segnale discreto avrà necessariamente un inizio e una fine, e il numero dei campioni a disposizione sarà un numero finito. A questo punto bisogna considerare questo segnale discreto e limitato come una porzione del segnale continuo ed infinito della *FT*, prelevato attraverso un'ulteriore funzione. Questa, chiamata finestra di troncamento, più generalmente conosciuta come finestra di analisi, entra a far parte dal prodotto della *DFT*.

Il risultato della *DFT* è quindi una convoluzione tra gli spettri del segnale *x(t)* e la finestra di analisi *w(t)*:

$$X_w(k) = X(k) * W(k)$$

Questa operazione di troncamento del segnale restituisce ovviamente delle distorsioni sul risultato dell'analisi.

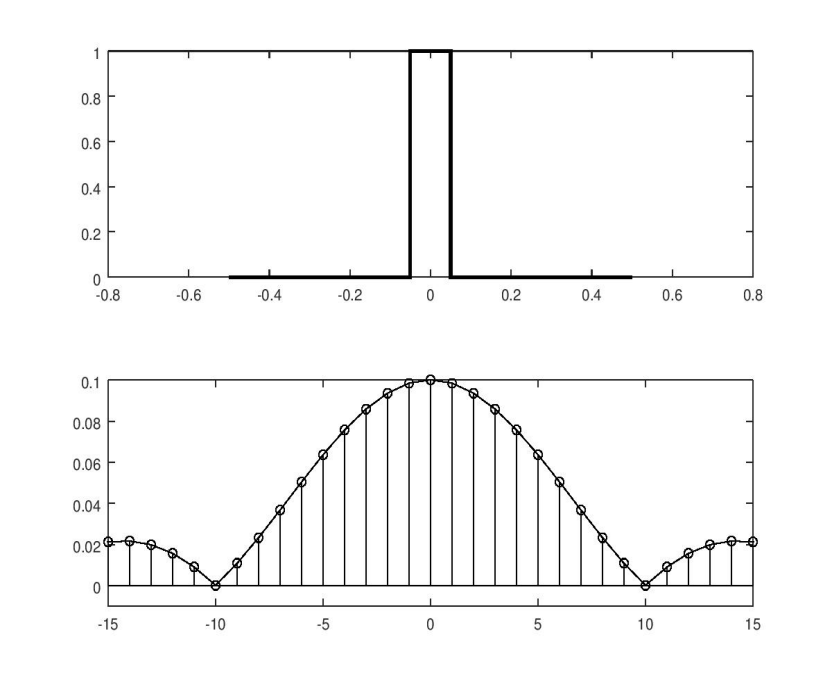
Nei codici precedenti si è utilizzata una finestra rettangolare.



Nei codici successivi si porterà in studio l'analisi spettrale della finestra rettangolare di analisi.

```
sr = 10000; %Sample Rate
pc = 1/sr;

durBig = 1; %sec
durSmall = 0.1; % durata della finestra
SmallOffSet = 0.45; % inizio della finestra

%duarata in campioni della finstra (dur Small Sample)
dSS = floor(durSmall*sr);
%OffSet in campioni della finestra
dSO = floor(SmallOffSet*sr);

t = [-durBig/2:pc:(durBig/2)-pc]; %array tempo

% generazione della finestra y
y = zeros(1, size(t, 2));
y(dSO:dSO+dSS) = 1;

winSize = floor(durBig*sr);
binSize = sr/winSize;

F = [-winSize/2:binSize:(winSize/2)-binSize];
```

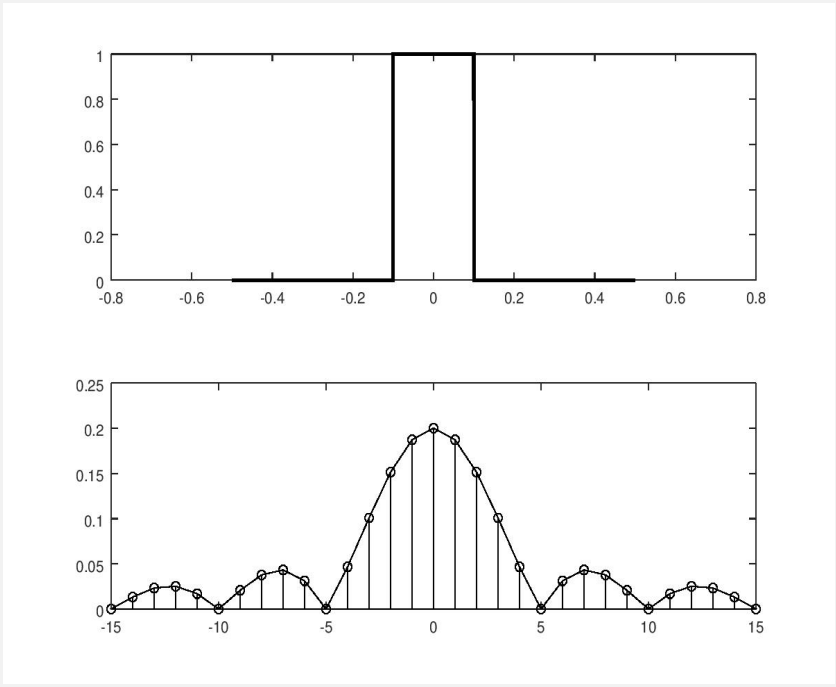
```
% DFT della finestra
for k = 1:size(F, 2);
fa = F(k)*2*pi;
za = e.^(-i*fa*t);
ya = y.*za;
dft(k) = abs(sum(ya))/size(ya, 2);
end

subplot(2, 1, 1)
plot(t, y, 'r')
subplot(2, 1, 2)
plot(F, dft, 'k')
hold on
stem(F, dft, 'b')
axis([-15 15 -0.01 0.1])
hold off
```



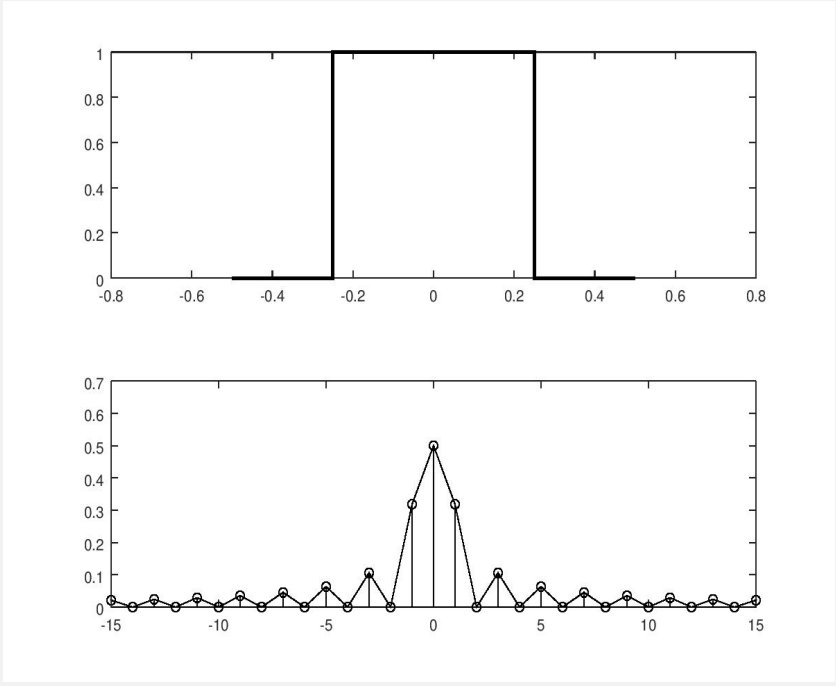
Dal grafico precedente si può notare che il periodo della nostra forma d'onda è 0,1 e la frequenza è 10. L'ampiezza è nulla ogni 10 bin.

Cambiando solamente il *duty cycle* della finestra di analisi si otterranno i diversi risultati spettrali.

```
%cambiando le variabili seguenti nel codice precedente

durBig = 1; %sec
durSmall = 0.2; % durata della finestra
SmallOffSet = 0.4; % inizio della finestra
```



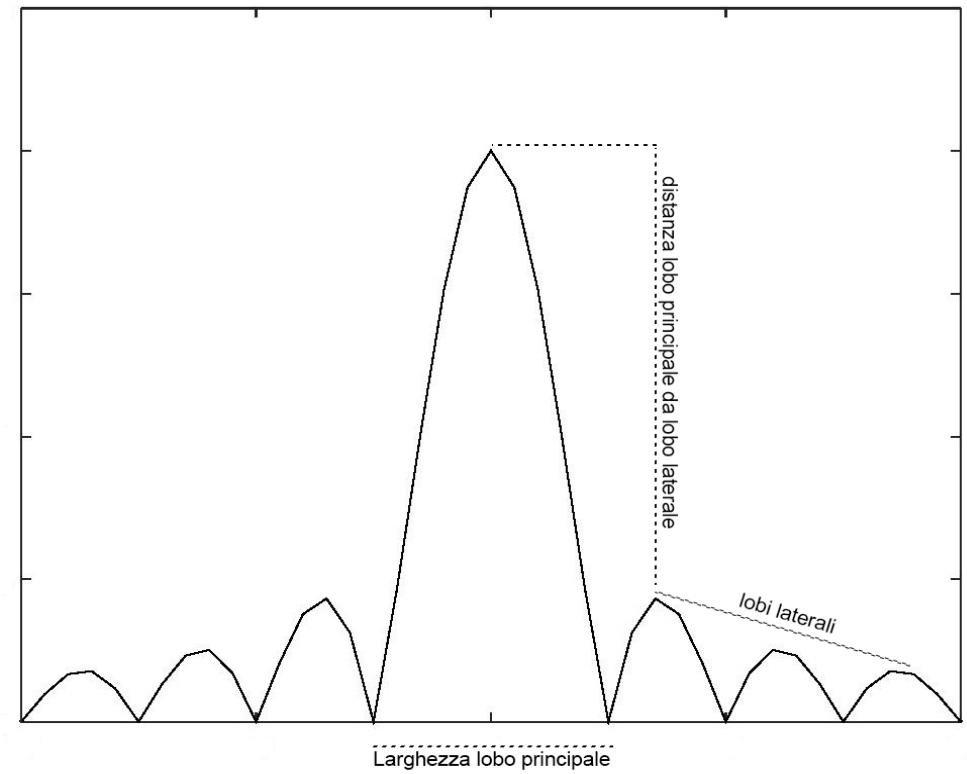

```
%cambiando le variabili seguenti nel codice precedente

durBig = 1; %sec
durSmall = 0.5; % durata della finestra
SmallOffSet = 0.25; % inizio della finestra
```



Nell'ultima figura è possibile notare meglio lo spettro di un'onda quadra (con *duty cycle* al 50%) per la quale si ha uno spettro composto da armoniche di ordine dispari con ampiezza inversamente proporzionale alla frequenza a fase nulla.

Si può capire dalla *DFT* che l'onda quadra analizzata è unipolare, tutta positiva, perché è presente la frequenza 0. Infatti la cosinusoide a 0 Hz è sempre 1.

Nella visualizzazione della finestra di analisi nel dominio della frequenza ci sono, in particolare, due parametri di cui bisogna tenere conto: la larghezza del lobo principale e l'ampiezza dei lobi laterali.



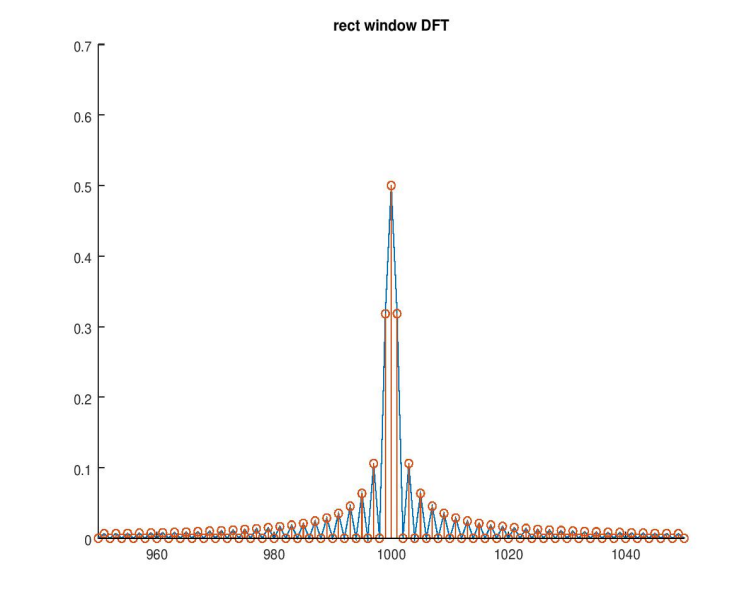
Come si è visto dai codici precedenti, più si aumenta il *duty cycle* più si restringe il lobo centrale. Lo spettro del segnale finestrato, cioè moltiplicato per la finestra rettangolare, per la proprietà del prodotto, è costituito dalla convoluzione dello spettro del segnale con lo spettro della finestra rettangolare. Con la convoluzione tra la finestra di analisi e il segnale analizzato avremo che il lobo principale sarà presente in corrispondenza di ogni componente frequenziale. La differenza tra il lobo centrale e il primo laterale è il *rapporto Segnale/Rumore*, *SNR* (*Signal to Noise Ratio*). La larghezza del lobo dà la definizione sequenziale, cioè aumenta la definizione frequenziale. Più il lobo è stretto più è definito ma maggiore è il *SNR*. Per questo la scelta della finestratura è sempre un compromesso tra accuratezza dell'ampiezza e risoluzione spettrale. La finestra può essere rettangolare ma questa forma altera significativamente lo spettro del segnale. Ad esempio un segnale sinusoidale finestrato dà uno spettro alterato rispetto al valore teorico di FT che restituisce una singola riga: in pratica la singola riga spettrale è diventata una intera banda intorno alla frequenza; inoltre sono prodotte delle repliche di questa banda in tutto lo spettro, con attenuazione decrescente.

Pertanto nella finestratura del segnale da analizzare si dovranno principalmente determinare due caratteristiche della finestra: la larghezza del lobo principale e la distanza tra il lobo laterale più alto e il lobo laterale principale. Idealmente è preferibile un lobo principale stretto (che determina,

come già detto una buona risoluzione) e un lobo laterale molto basso (che diminuisce il cross-talk tra i canali FFT). La scelta della finestra si determina su questo compomesso. Ad esempio, la finestra rettangolare ha il lobo principale più stretto, ma il primo lobo laterale è molto alto rispetto al picco del lobo principale.

Nel codice successivo è possibile vedere una DFT di una funzione finestrata con finestra rettangolare. Si vedranno solo le componenti dispari. Intorno alla frequenza portante si vedranno solo le componenti della modulante

```
fc = 10000;
pc = 1/fc;
durBig = 1;
t = [-durBig/2:pc:durBig/2-pc];
f = 1000;
p = cos(2*pi*f*t);% frequenza portante
m = zeros(1, length(t));% frequenza modulante: l'onda quadra
nSamp = durBig*fc;
m(round(nSamp/4):round(nSamp/4*3)) = 1; %l'onda quadra va
ad 1 da 1/4 a 3/4

BinSize = 1;
F = [-fc/2:BinSize:fc/2-BinSize];
out = m.*p;%la modulazione d'ampiezza moltiplica portante e modulante
for k = 1:length(F)
fa = F(k)*2*pi;
za = out.*e.^(-i*fa*t);
dft(k, 1) = (abs(sum(za))/length(t))*2;% essendo una funzione reale
l'energia si dimezza. Inoltre si dimezza ulteriormente perche' la
finestra si mangia un'altra meta' di energia. Moltiplicando x 2
avremo la portante a 0,5

end
hold on
plot(F, dft);
stem (F, dft);
axis([950 1050]);
title ('rect window DFT');
hold off
```



Esistono pertanto diverse tipologie di finestre. Per esempio la finestra di *Hamming* che ha un lobo principale più largo della rettangolare ma un lobo laterale molto più basso.

La finestra con il lobo laterale più basso è la *Blackman*

Una finestra molto diversa, la *Kaiser*, consente il controllo del *trade-off* tra la larghezza del lobo principale e il livello del lobo laterale più alto. Se vogliamo una minore larghezza del lobo principale otterremo un maggiore livello di lobo laterale e viceversa. Poiché il controllo di questo trade-off è prezioso, la finestra di Kaiser è una buona scelta di uso generale.

```
clear all
close all

[y, fs] = audioread('../audio/ring.wav');

pc = 1/fs;
nsamp = size(y, 1);
dur = pc*nsamp;

t = [0:pc:dur-pc]';

tWinSize = 2000;;
tWin = [floor(size(y, 1)/4) floor(size(y, 1)/4)+tWinSize-1];
y = y*(1/max(y))*(10^(-(1/20)));

%creazione finestra di Hann
Hann = -0.5*cos((2*pi/tWinSize)*(0:tWinSize-1))'+0.5;

yW = y(tWin(1):tWin(2),1).*Hann;
tW = t(tWin(1):tWin(2),1);
```

```
winSize = 250;
binSize = fs/winSize

F = [0:binSize:fs-binSize]';

for k = 1:size(F, 1)/2
fa = F(k)*2*pi;
za = yW.*e.^(-i*fa*tW);
dft(k, 1) = (abs(sum(za))/size(tW,1))*4;
end

dBdft = 20*log10(dft);

subplot(2, 1, 1)
plot(tW, yW, tW, Hann);
subplot(2, 1, 2)
plot(F(1:size(dft, 1), 1), dBdft);
```