UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Elaborato Calcolo Numerico

Anno Accademico 2019/2020

Prof. D'Amore Luisa

Student:

Coppola Vincenzo Matr. M63/1000
Della Torca Salvatore Matr. M63/1011

Indice

INTRODUZIONE FFT

La *Fast Fourier Transform*, nota anche come **FFT**, è un algoritmo ottimizzato per il calcolo della *trasformata discreta di Fourier* (**DFT**).
L'obiettivo della FFT è quello di ridurre la complessità di calcolo della DFT da $O(N^2)$ a $O(N \cdot logN)$. In molti problemi, come i problemi di elaborazione di segnali digitali, è richiesto il calcolo della *trasformata di Fourier*.
Data una funzione *f(t)*, reale o complessa, si definisce Trasformata di Fourier la funzione

$$F(i\omega) = \frac{1}{\sqrt{2\pi}} \int_R f(t) \cdot e^{-i\omega t} dt$$

dove $e^{-i\omega t} = cos(\omega t) - i \cdot sin(\omega t)$ è l'esponenziale complesso e $i = \sqrt{-1}$.
La trasformata di Fourier ci consente di rappresentare una funzione definita nel tempo nel dominio delle *frequenze*.
Per le proprietà dell'esponenziale complesso la funzione *f(t)* può essere ottenuta come somma di funzioni periodiche *(seni e coseni)* le cui frequenze sono multipli della frequenza fondamentale.
Indicata con $\omega_0 = f_0 = \frac{1}{T}$ la *frequenza fondamentale*, il segnale *f(t)* sarà:

$$f(t) = A_0 + \sum_{n=1}^{+\infty} A_n cos(n\omega_0 t) + \sum_{n=1}^{+\infty} B_n sin(n\omega_0 t)$$

Per poter calcolare la **DFT** è necessaria una discretizzazione in quanto l'integrale è un operatore continuo. É necessario quindi campionare la funzione *f(t)* in un numero finito **N** di nodi t_j e sostituire all'integrale una sommatoria finita.
Se indichiamo con $f_j = f(t_j)$ allora il componente *K-esimo* del vettore della DFT sarà

$$F_k = \sum_{j=0}^{N-1} f_j \cdot e^{-i2\pi j \frac{k}{N}}$$

dove $k = 0, \dots, N-1$ e $e^{i2\pi j \frac{k}{N}} = cos(\frac{2\pi}{N} jK) + isen(\frac{2\pi}{N} jK)$.

Il vettore $F = (F_0, F_1, \dots, F_{N-1})$ è detto *Trasformata Discreta di Fourier* ***DFT*** del vettore $f = (f_0, f_1, \dots, f_{N-1})$. Calcolare direttamente questa somma richiede una quantità di operazioni aritmetiche $O(N^2)$, mentre un algoritmo FFT ottiene lo stesso risultato con un numero di operazioni $O(N \cdot logN)$. In generale questi algoritmi si basano sulla *fattorizzazione* di N, ma esistono anche algoritmi applicabili per un qualunque N, anche per numeri primi.

La *Trasformata Discreta Inversa di Fourier* può essere ottenuta a partire dal vettore $\boldsymbol{F}$:

$$f_j = \frac{1}{N} \sum_{k=0}^{N-1} F_k \cdot W_N^{-jk}$$

dove $j = 0, \ldots, N-1$ e $W_N = e^{-i\frac{2\pi}{N}}$.
Se indichiamo con Δ il passo di campionamento e con N il numero di nodi allora $t_j = j\Delta$, il periodo $T = N\Delta$ e la frequenza di campionamento $Fs = \frac{N}{T} = \frac{1}{\Delta}$.
Se consideriamo l'esponenziale complesso possiamo facilmente determinare le frequenze che intervengono nella DFT: $\frac{2\pi}{N}jk = \frac{2\pi\Delta}{T}jk = \frac{2\pi}{T}t_jk$
$\frac{1}{T} = \Delta f$ è la frequenza fondamentale, cioè la più bassa frequenza nella DFT, e tutte le altre frequenze sono multipli interi della frequenza fondamentale.
La più alta frequenza è stabilita dal *teorema di Shannon* che definisce la *frequenza di Nyquist* $F_N = \frac{1}{2\Delta} = \frac{N}{2T} = \frac{Fs}{2}$.
L'algoritmo FFT più diffuso è l'algoritmo di CooleyTukey che si basa sul principio di *divide et impera* e spezza ricorsivamente una DFT di qualsiasi dimensione **N**, con $N = N_1N_2$, in DFT di dimensioni inferiori.
L'uso più conosciuto dell'algoritmo di Cooley-Tukey è di dividere e trasformare la DFT in due pezzi di lunghezza $\frac{N}{2}$ ad ogni passo, ed è quindi ottimizzato solo per dimensioni che siano potenze di due, ma in generale può essere utilizzata qualsiasi fattorizzazione. Si parla rispettivamente di *Radix-2* e *Mixed-Radix*. Anche se l'idea di base è ricorsiva, la gran parte delle implementazioni tradizionali riorganizzano l'algoritmo per evitare la ricorsione esplicita e utilizzano dei metodi iterativi come l'algoritmo di Gentleman-Sand.

Esercizio 1

Utilizzo della FFT

In questo primo esercizio si vuole mostrare come lavora la funzione messa a disposizione da *Matlab* per il calcolo della DFT.
Matlab fornisce la funzione ***fft*** che prende come parametro di input un vettore X e restituisce come output la DFT calcolata mediante l'algoritmo *Fast Fourier Transform.*
La DFT ottenuta in output avrà la stessa dimensione di X, tranne nel caso in cui come parametro di input venga fornito non solo il vettore X ma anche un parametro n; in tal caso la funzione *fft* restituirà *n punti* della DFT.
La funzione ***ifft*** fornisce invece la DFT inversa del vettore calcolato mediante l'algoritmo FFT.
Tutte le funzioni relative alla FFT che Matlab mette a disposizione sono basate su una libreria chiamata *FFTw*, abbreviazione di *Fastest Fourier Transform in the West*, che è una libreria *C* per il calcolo della Fast Fourier Transform. Le funzioni appartenenti a questa libreria operano con estrema velocità grazie ad un'accurata ottimizzazione degli algoritmi utilizzati e all'utilizzo di istruzioni specifiche per i vari processori.
In questo primo esercizio si va a calcolare la trasformata di Fourier discreta di un segnale *f(t)* dallo dalla somma di due sinusoidi a frequenza diversa.
L'utente deve inserire il *periodo T* e la *frequenza di campionamento Fs* che sono gli input di una funzione il cui compito è quello di calcolare la DFT.
La funzione realizza anche il grafico dello spettro della DFT e poi calcola la DFT inversa per compararla al segnale di partenza.

Esempio di applicazione per T=1 e Fs=500

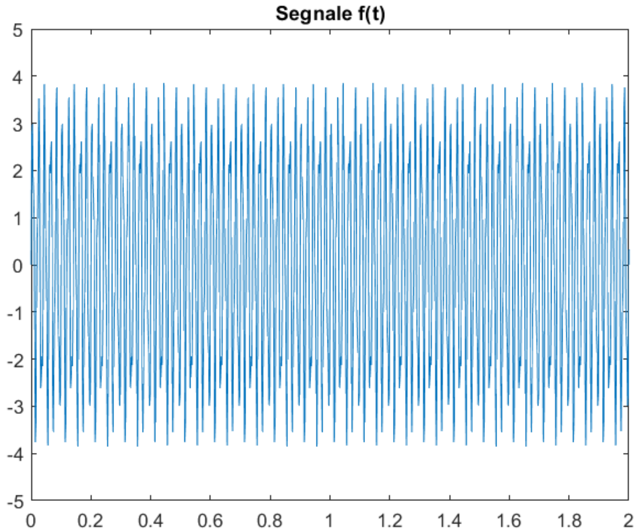

Fig. 1.1: $f(t) = 3sin(2\pi 50t) + sin(2\pi 120t)$

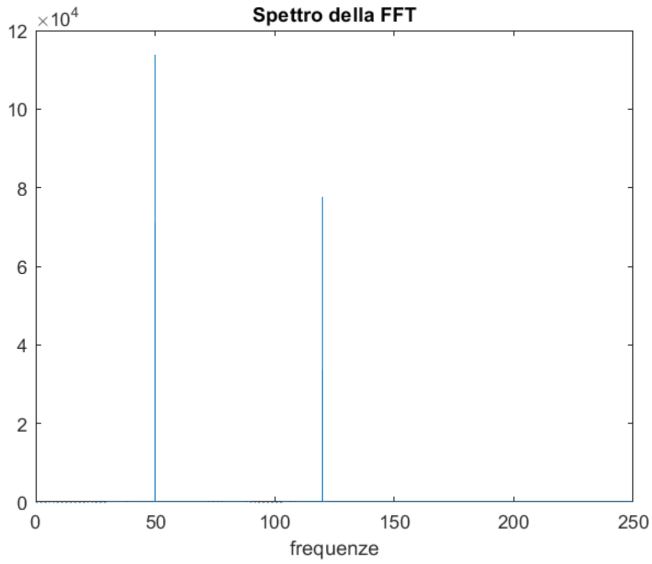

Fig. 1.2: Spettro della DFT del segnale f(t)

Poichè il vettore F_K è un vettore complesso, negli esercizi successivi è stato utilizzato l'*istogramma* per poterlo rappresentare. Un istogramma è il grafico del modulo di F_K rispetto alle frequenze.
Per realizzare l'istogramma è stata utilizzata la *function* ***stem*** che prende come parametri di input X e Y e traccia la sequenza di dati Y ai valori specificati in X.

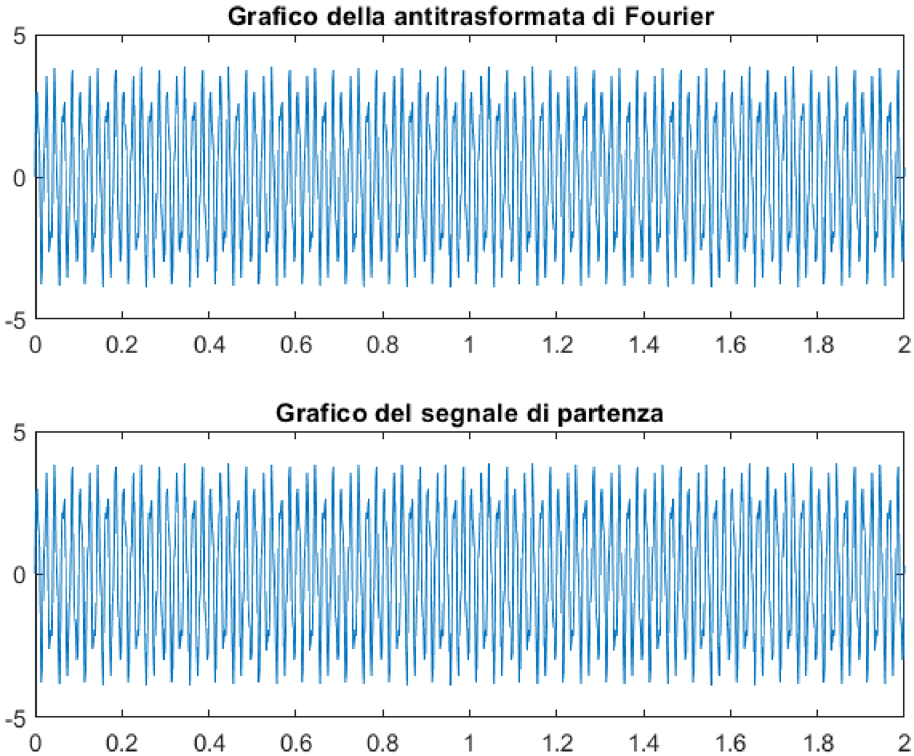

Fig. 1.3: Confronto tra il risultato della IFFT e il segnale di partenza

Esercizio 2

Confronto tra trasformata continua e discreta

Lo scopo di questo esercizio è confrontare la trasformata di Fourier continua con quella discreta per quattro diverse sequenze di 32 campioni. Di seguito verrà mostrata la prima sequenza x, che ha solo 3 campioni diversi da 0: $x(1) = \frac{1}{2}$, $x(2) = \frac{1}{4}$, $x(32) = \frac{1}{4}$.

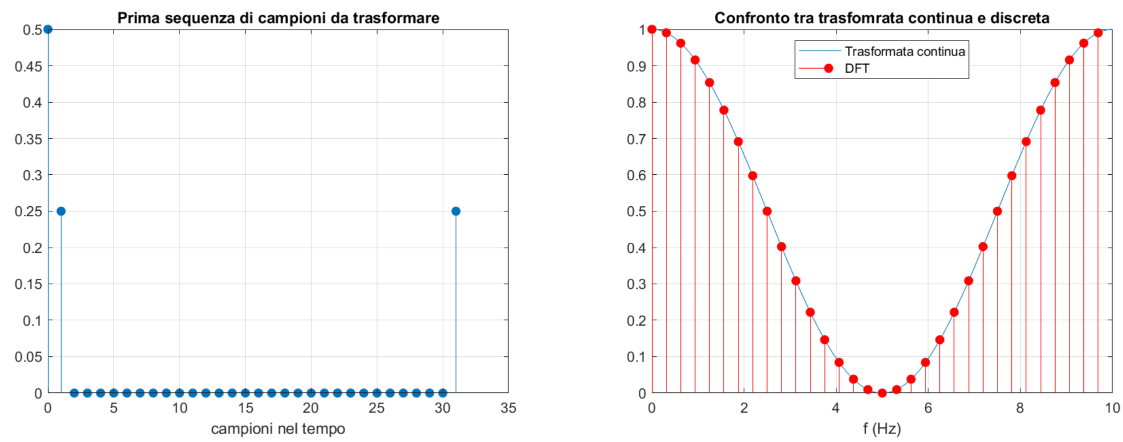

Fig. 2.1: Prima sequenza e confronto tra le trasformate

Di seguito invece verrà mostrata la seconda sequenza di 32 campioni, anch'essa con 3 campioni diversi da 0, ovvero: $x(1) = \frac{1}{2}$, $x(2) = -\frac{1}{4}$, $x(32) = -\frac{1}{4}$.

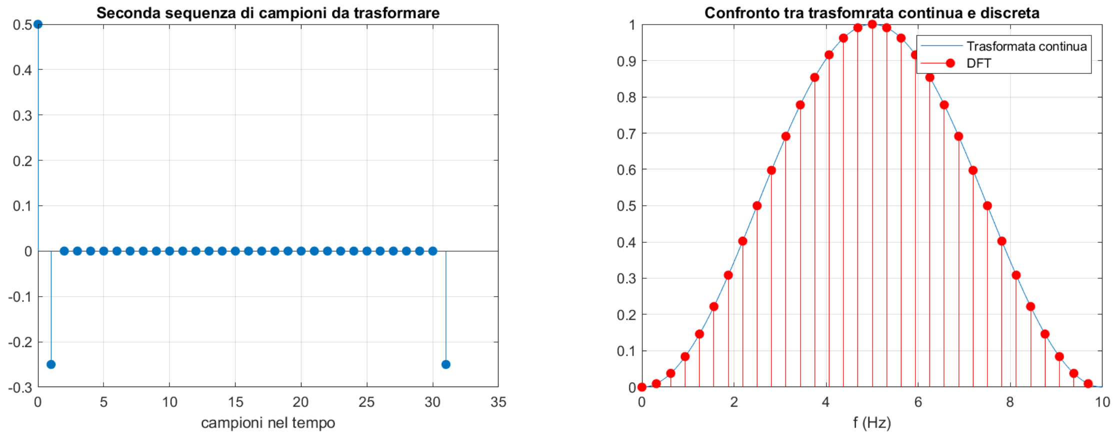

Fig. 2.2: Seconda sequenza e confronto tra le trasformate

La terza sequenza invece è un coseno descritto sempre da 32 campioni: $\cos(2\pi N \frac{1}{8})$, con N=32. In particolare il grafico continuo è stato realizzato campionando il segnale a frequenza più alta, in modo quindi da sembrare continuo anche se non lo è.

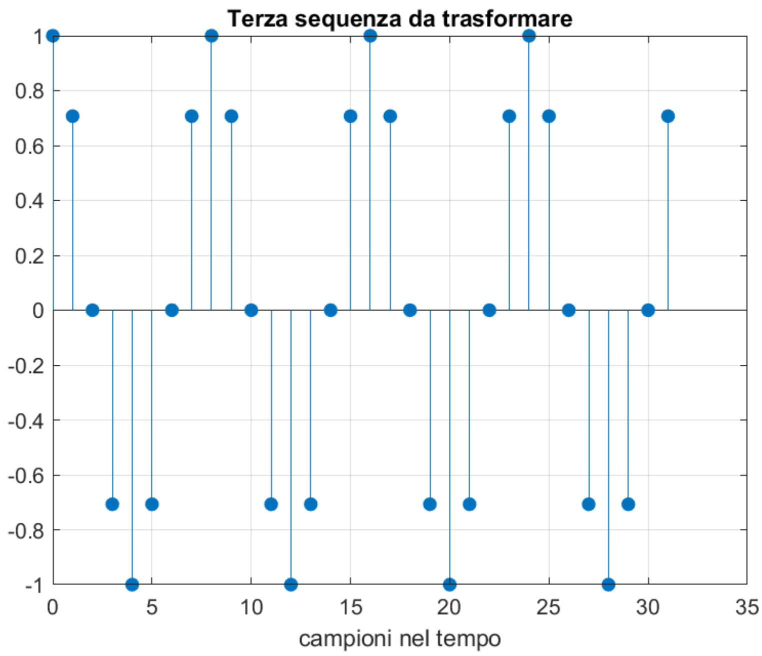

Fig. 2.3: Terza sequenza di campioni

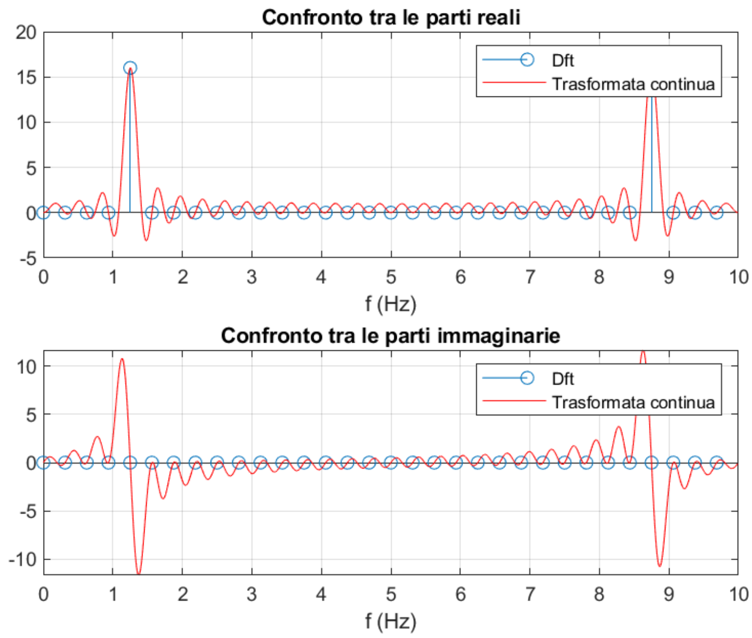

Fig. 2.4: Confronto tra le trasformate

Infine l'ultima sequenza è sempre un coseno di 32 campioni $\cos(2\pi N \frac{1}{7})$, con N=32. La differenza però è che il numero di cicli non è intero, cioè i multipli della frequenza fondamentale non sono interi. Da ciò ne consegue quindi una trasformata *anomala*, che non presenta la coppia di impulsi tipici della trasformata del coseno, a differenza della corretta trasformata della sequenza precedente.

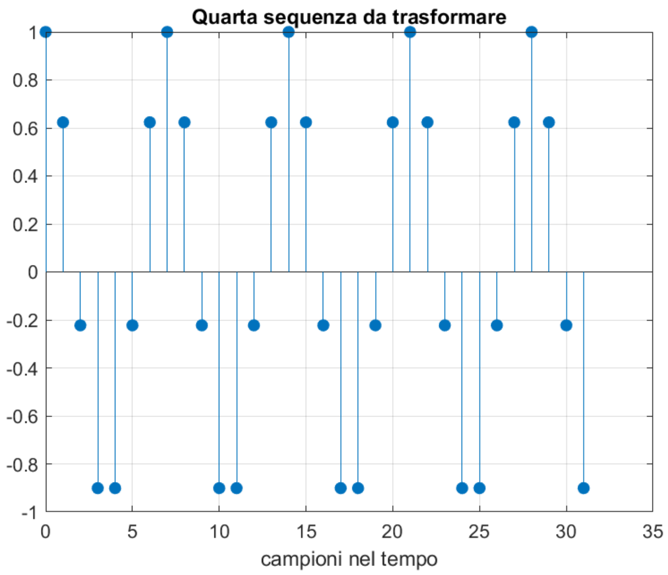

Fig. 2.5: Quarta sequenza di campioni

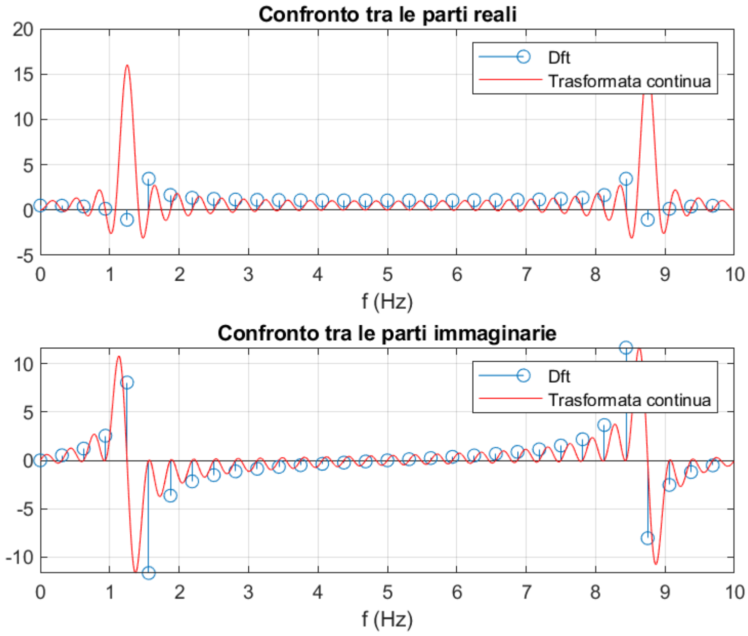

Fig. 2.6: Confronto tra le trasformate

Esercizio 3

Filtraggio di un segnale

Questo esercizio vuole mostrare come è possibile realizzare il filtraggio in maniera molto semplice passando dal dominio del tempo al dominio della frequenza attraverso l'utilizzo dell'algoritmo FFT.
Nello studio e nell'elaborazione dei segnali, al giorno d'oggi, si fa uso di tecniche di ***D.S.P.*** *(Digital Signal Processor)*, attraverso le quali si effettua l'analisi in frequenza. Se consideriamo ad esempio un suono e ne rappresentiamo lo spettro, si potrà osservare che questo è costituito da una sinusoide perfetta *(tono puro)*, alla quale si aggiunge una singola componente ad un certo livello e ad una frequenza pari a quella della sinusoide. Questa componente è dovuta alla presenza del *rumore*.
Tutti i suoni sono generalmente affetti da rumore e quindi si ottengono segnali che sono sovrapposizioni di tantissime sinusoidi ad ampiezze continuamente variabili e che quindi sono difficili da rappresentare. Questo contrasta il *teorema di Fourier* che, come avevamo visto in precedenza, afferma che un qualsiasi segnale è rappresentabile come somma di sinusoidi; questo è vero solo per segnali stazionari e non tempo varianti.
Questo esercizio parte quindi da una funzione *x(t)* periodica, definita come combinazione di due sinusoidi, e a questa funzione viene sommata una sinusoide *y(t)* al alta frequenza che rappresenta un rumore.
Il segnale $x(t) = sin(2\pi 5t) + sin(2\pi 10t)$ viene sommato al rumore sinusoidale $y(t) = 5sin(2\pi 30t)$ per ottenere il segnale rumoroso z(t).

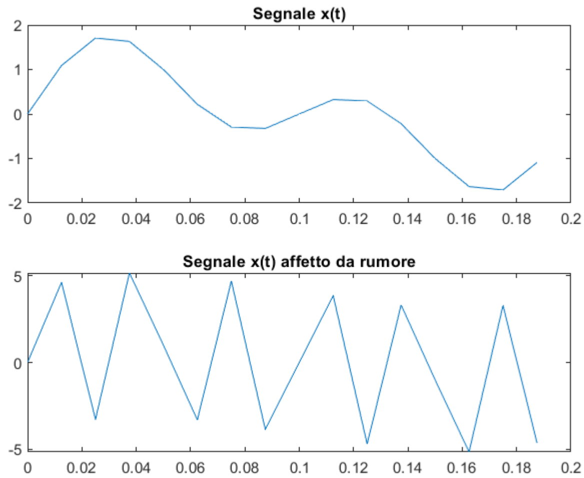

Fig. 3.1: Grafico del segnale x(t) e del segnale z(t)

Il segnale rumoroso z(t) viene poi trasformato nel dominio della frequenza attraverso l'algoritmo FFT. Per rappresentare l'istogramma è stato necessario considerare solamente metà dei multipli della frequenza fondamentale perchè la DFT gode di simmetria.

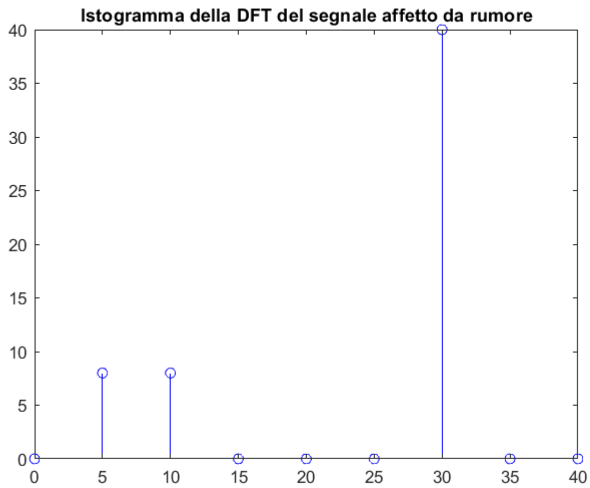

Fig. 3.2: Istogramma della DFT del segnale rumoroso

Dall'istogramma si può osservare che alla frequenza *30 Hz* è presente una componente dovuta al rumore sinusoidale y(t); si vuole filtrare il segnale cosi da eliminare tale componente.

Si può osservare che le componenti del segnale x(t) non presentano frequenze maggiori a *10 Hz*, per cui si può pensare di applicare un filtro che vada a tagliare tutte le frequenze maggiori di tale valore.

Per costruire il filtro passa basso è stato realizzato un vettore di tutti termini pari a 1 e lunghezza pari alla lunghezza del vettore delle frequenze, dopo di chè tutti i valori del vettore a frequenza maggiore di 10 Hz sono stati azzerati.
Poichè la DFT e il filtro sono simmetrici rispetto alla frequenza è stata copiata la prima metà del vettore filtro nella seconda metà e, a questo punto, è stato possibile filtrare il segnale semplicemente moltiplicando *(componente per componente* il segnale rumoroso e il vettore filtro.
É stata poi calcolata la DFT inversa del segnale filtrato per osservare se il segnale coincidesse con quello di partenza.

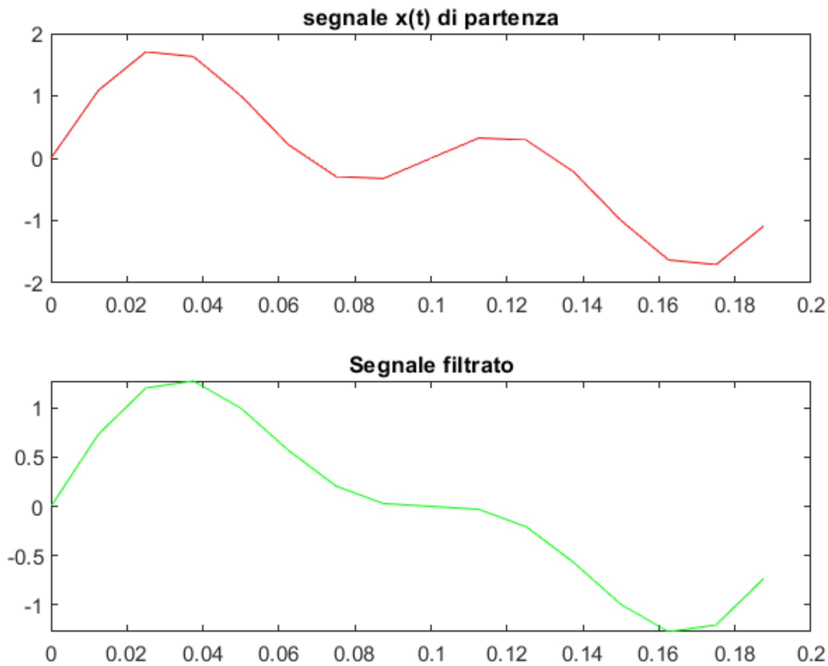

Fig. 3.3: Confronto fra segnale filtrato e segnale di partenza

Esercizio 4

Analisi errore della FFT

L'obiettivo di questo esercizio è mettere in evidenza l'errore introdotto dall'algoritmo FFT quando il numero di campioni non è sufficientemente elevato.
Per semplicità è stata utilizzata come funzione di esempio la *finestra rettangolare*, la cui trasformata di Fourier è nota ed è una *sinc*.
Matlab però mette a disposizione solamente una *function* per la realizzazione della finestra rettangolare continua e per questo motivo è stata realizzata una function *(rett.m)* che permette di creare una finestra rettangolare discreta, cioè un segnale *rect(t) campionato*. Questa funzione prende come parametri di input l'intervallo di campionamento, il punto iniziale e il punto finale della finestra rettangolare che si desidera rappresentare, e fornisce come output la rect campionata.
Per quanto riguarda la *sinc*, Matlab mette a disposizione la *function sinc* che però può essere utilizzata solamente se è installato il *"Signal Processing Toolbox"*. Per poter eseguire il programma anche se il tool non è installato è stata realizzata la funzione *sinc.m* che prende come parametro di ingresso il vettore dei valori in cui calcolare la sinc e fornisce come output i valori della sinc nei punti del vettore di ingresso.
N.B: In questa funzione è stata inserita un'istruzione che sostituisce il valore della sinc in 0 col valore 1, perchè altrimenti Matlab avrebbe interpretato tale valore come ***NaN*** *essendo dato dal rapporto* $\frac{0}{0}$.

Lo script permette all'utente di inserire il passo di campionamento, estremo inferiore e superiore dell'intervallo di campionamento, il punto iniziale e il punto finale della finestra rettangolare, e a partire da tali parametri calcola il vettore dei punti in cui campionare la finestra rettangolare.
Per applicare l'algoritmo FFT è necessario rendere il segnale periodico, per cui bisogna individuare il campione che corrisponde a t = 0: se il campione non esiste l'algoritmo termina restituendo un messaggio di errore.

Se il campione esiste viene generato il segnale periodico e viene calcolata la FFT di questo segnale; moltiplicando poi la DFT per il passo di discretizzazione dt si ottiene la sequenza campionata e periodizzata nella finestra da 0 a $\frac{1}{dt}Hz$.
Tramite la funzione ***"fftshift"*** viene spostata la finestra da $-\frac{1}{2dt}$ a $\frac{1}{2dt}$, e a questo punto viene calcolato il passo di campionamento in frequenza necessario per determinare il vettore delle frequenze.
ATTENZIONE: a tal proposito, la *"fftshift"* utilizza una particolare convenzione, cioè in caso di lunghezza **pari** della sequenza assegna un campione in più alle frequenze negative, mentre in caso di lunghezza **dispari** della sequenza assegna un numero uguale di campioni alle frequenze positive e negative.
Per questo motivo è stato realizzato un controllo sul numero di campioni utilizzando la funzione ***rem*** che restituisce il resto della divisione tra i parametri di ingresso.

Esempio di applicazione per:
- dt = 0.1;
- intervallo di campionamento [-2 2];
- finestra rettangolare nell'intervallo [-0.5 0.5].

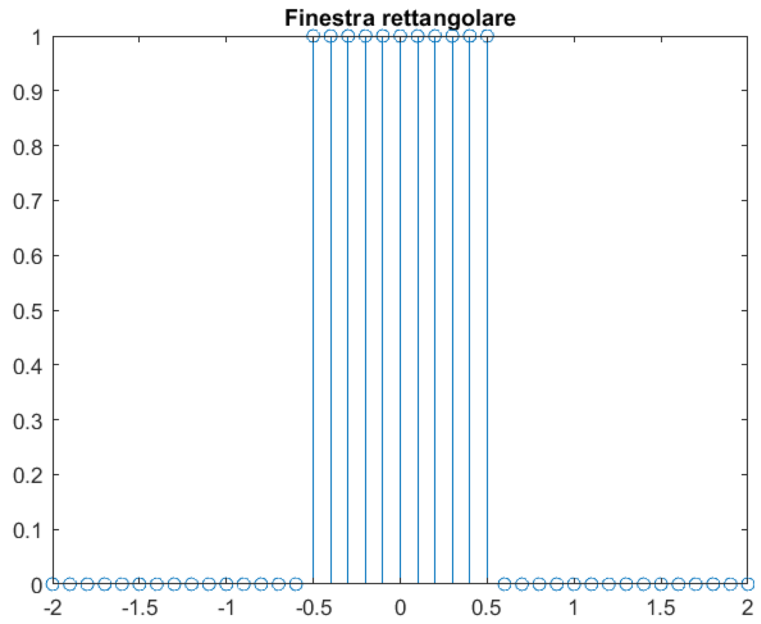

Fig. 4.1: Rappresentazione campionata della rect(t)

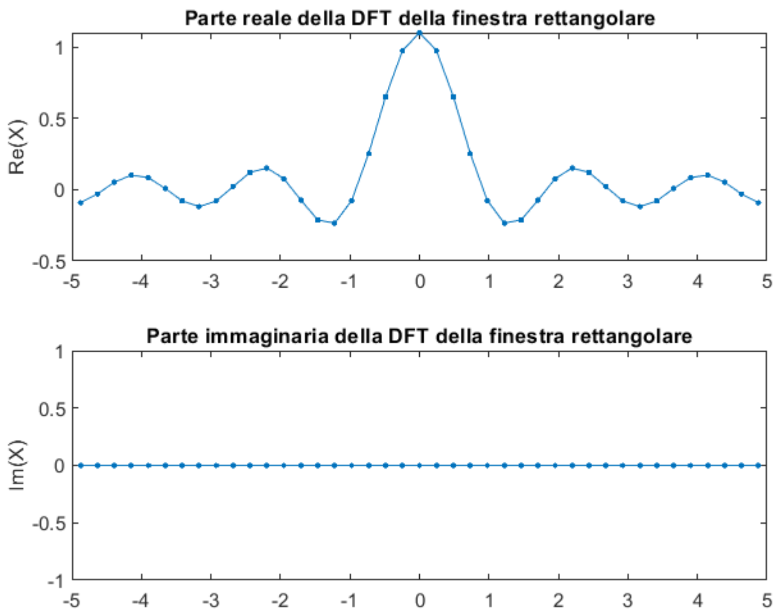

Fig. 4.2: FFT della funzione rect (parte reale ed immaginaria)

Anche se in questo grafico non si nota, l'algoritmo FFT esegue delle approssimazioni nel calcolo della trasformata di Fourier che generano una piccola distorsione, dell'ordine di 10^{-16}, sulla parte immaginaria.
Per poter osservare questo fenomeno è stato confrontato il risultato ottenuto attraverso la FFT con il risultato ottenuto con il calcolo analitico della trasformata, ossia la funzione *sinc*.

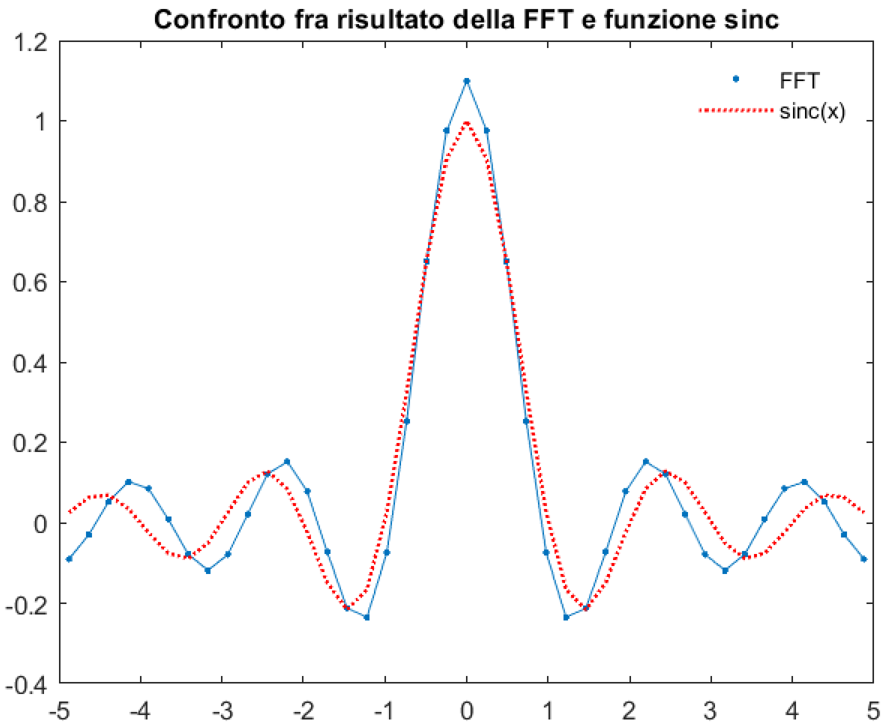

Fig. 4.3: Confronto fra FFT e sinc

Dal grafico si può osservare che l'errore è maggiore sulle frequenze più alte e questo è dovuto alla periodicità della FFT.
Per ridurre l'errore si può intervenire sul numero di campioni, e quindi anche sul passo di discretizzazione: più il numero di campioni è elevato, più il risultato ottenuto sarà accurato.

Esercizio 5

Modulazione mediante portanti in quadratura

Lo scopo di questo esercizio è analizzare la trasmissione di 2 segnali gaussiani in un canale mediante due portanti in quadratura, un seno e un coseno.
In trasmissione si modulano i segnali di input lungo le portanti moltiplicandoli per la sinusoide che rappresenta la portante (nell'esempio abbiamo moltiplicato il primo segnale per un coseno e il secondo segnale per un seno); i due segnali modulati vanno poi sommati così da ottenere il segnale da trasmettere.
In ricezione, ricevuta la somma dei due segnali modulati, tale somma viene moltiplicata per lo stesso seno e coseno usati per modulare i segnali originari così da riottenere i segnali modulati sommati in trasmissione. Quindi il prodotto del segnale somma per il coseno e per il seno permette di annullare la somma e ottenere i segnali demodulati, ai quali va poi applicato un filtraggio passa basso così da eliminare le frequenze relative al seno e al coseno e riottenere i due segnali originari.
Da notare che le sinusoidi usate per modulare devono essere non solo alla stessa frequenza, ma anche ortogonali tra loro per evitare di mescolare i segnali in ricezione.

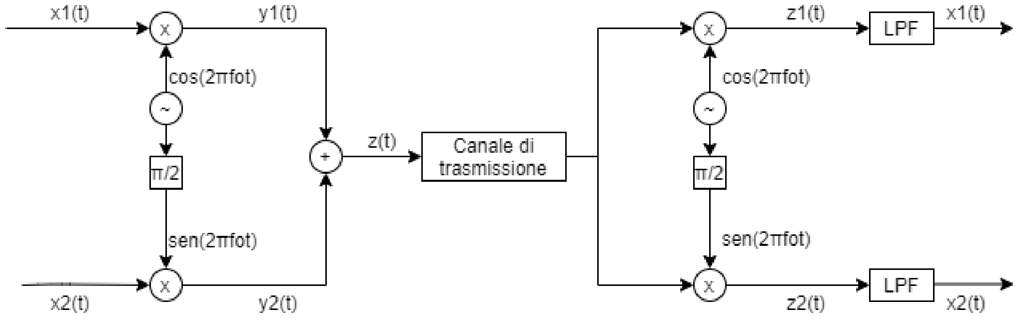

Fig. 5.1: Schema riassuntivo della modulazione

Di seguito vengono mostrati i due segnali gaussiani da modulare, chiamati $x_1(t)$ e $x_2(t)$, e le relative trasformate di Fourier.

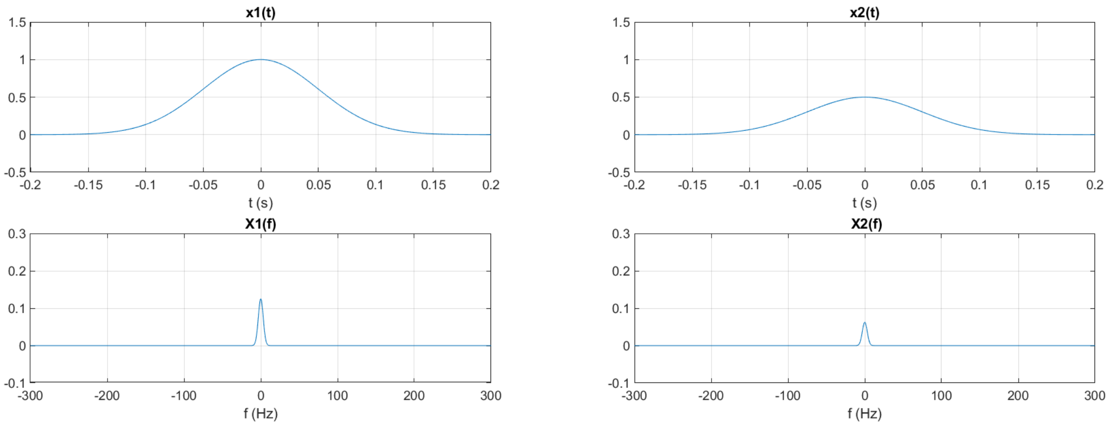

Fig. 5.2: Segnali $x_1(t)$ e $x_2(t)$ da modulare

Di seguito vengono mostrati invece i segnali $y_1(t)$ e $y_2(t)$ e le relative trasformate di Fourier. In particolare, $y_1(t)$ si ottiene modulando $x_1(t)$, ovvero $y_1(t) = x_1(t) \cdot \cos(2\pi f_0 t)$, mentre $y_2(t)$ si ottiene modulando in seno $x_2(t)$, ossia $y_2(t) = x_2(t) \cdot \sin(2\pi f_0 t)$. Da notare che $y_1(t)$ ha trasformata di Fourier puramente reale, visto che si ottiene mediante moltiplicazione per un coseno; invece la trasformata di $y_2(t)$ ha solo parte immaginaria poiché si ottiene mediante la moltiplicazione per un seno.

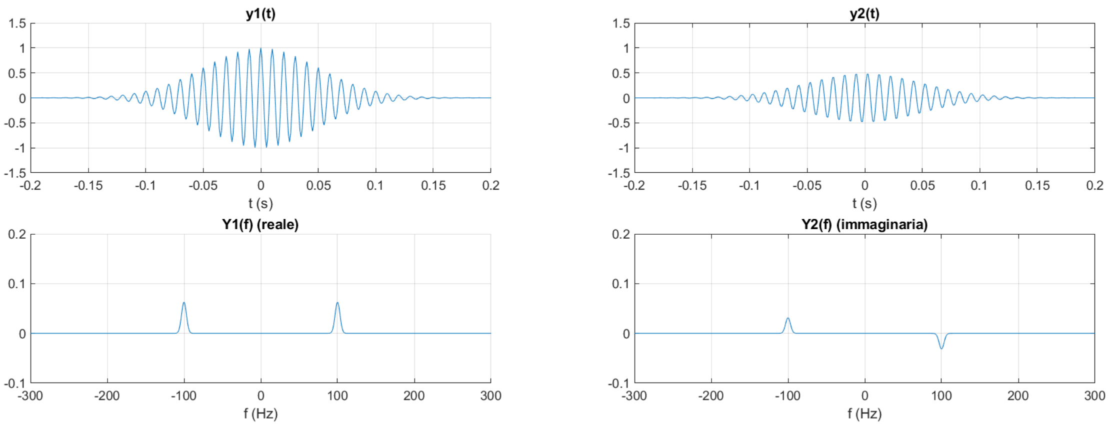

Fig. 5.3: Segnali $y_1(t)$ e $y_2(t)$ da modulare

A questo punto si costruisce il segnale da trasmettere, ovvero $z(t) = y_1(t) + y_2(t)$. In particolare la sua trasformata sarà complessa: la parte reale corrisponde allo spettro di $y_1(t)$, mentre la parte immaginaria corrisponde allo spettro di $y_2(t)$.

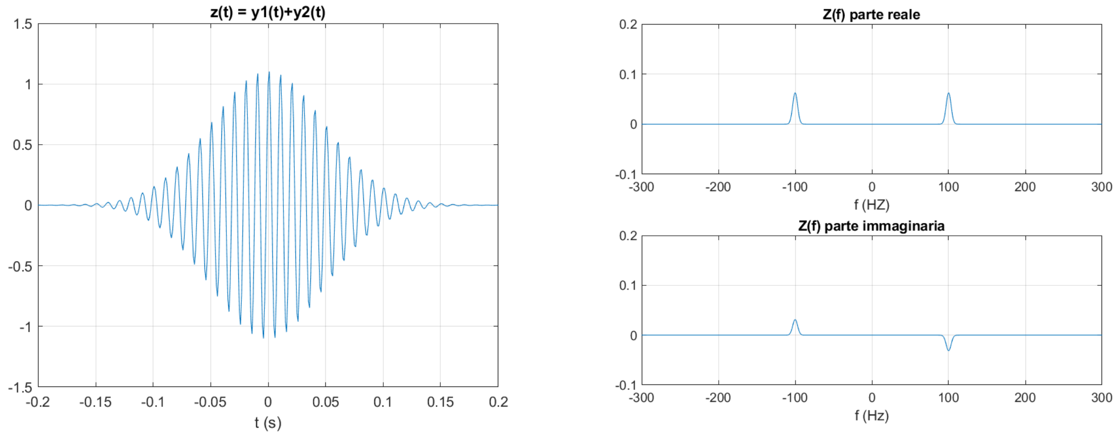

Fig. 5.4: Segnale $z(t)$

Una volta ricevuto, il segnale $z(t)$ deve essere demodulato. Per fare ciò si moltiplica il segnale sia per un coseno, che deve avere la stessa fase e frequenza del coseno usato per modulare il segnale $x_1(t)$, così da ottenere il segnale $z_1(t) = z(t) \cdot \cos(2\pi f_0 t)$, sia per un seno avente la stessa fase e frequenza del seno usato per modulare $x_2(t)$, così da ottenere $z_2(t) = z(t) \cdot \sin(2\pi f_0 t)$.
Queste condizioni sulla fase e sulla frequenza servono per non ritrovare in $z_1(t)$ e in $z_2(t)$ i segnali $x_1(t)$ e $x_2(t)$ mescolati: in $z_1(t)$ deve essere presente solo $x_1(t)$, e in $z_2(t)$ deve essere presente solo $x_2(t)$. Verranno mostrati in seguito i due segnali:

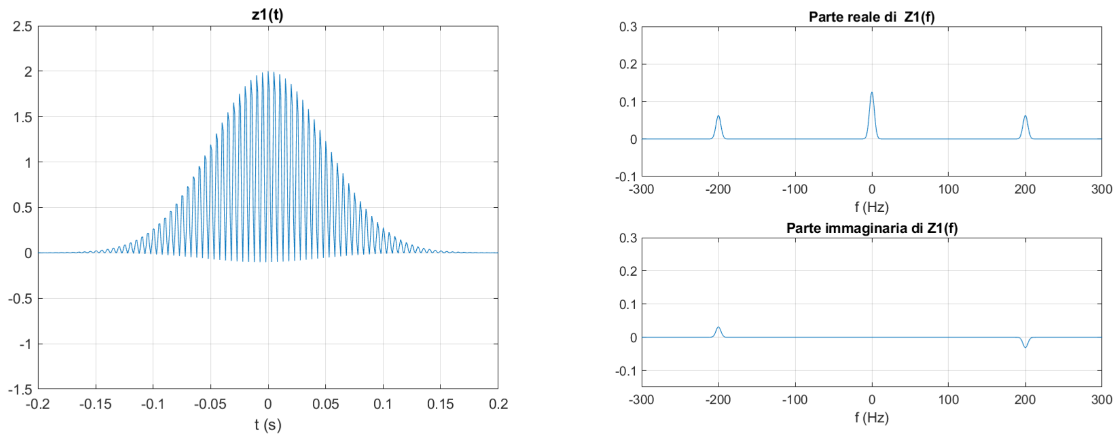

Fig. 5.5: Segnale $z_1(t)$ e relativa trasformata

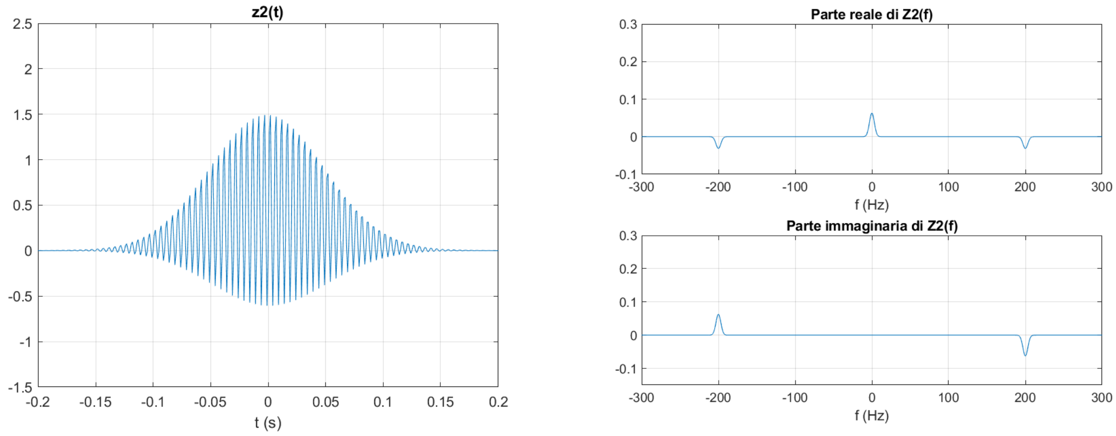

Fig. 5.6: Segnale $z_2(t)$ e relativa trasformata

Verrà mostrato in seguito un confronto tra i segnali $z_1(t)$ e $x_1(t)$, $z_2(t)$ e $x_2(t)$. Da notare che i segnali sono diversi: questo perché sia in $z_1(t)$ che in $z_2(t)$ sono presenti le componenti frequenziali del coseno e del seno, che dovranno poi essere tagliate con un filtro passa basso.