

Università degli Studi di Perugia



Dipartimento di Matematica e Informatica

Report di Laboratorio - Tecniche di Acquisizione Dati

Trasformata di Fourier

Studenti

Simone Seria

Giulia Diletta Fabbriani

Anno Accademico 2023/2024

Introduzione	3
Scopo	3
Strumenti e Librerie	3
Trasformata di Fourier	4
Parte Prima	5
Odd_k()	6
Calcolo della Sommatoria	6
DFT e Generazione del Grafico	7
Risultato Finale	8
Parte Seconda	9
Sintesi dei Segnali	9
Grafici dei Segnali Sintetizzati	10
Studio in Frequenza	11
Generazione dei Grafici	12
Risultato Finale	14
Conclusione	16

Introduzione

La seconda esperienza di laboratorio ha come scopo la sintesi e lo studio di segnali tramite la Trasformata di Fourier.

Scopo

La prima parte dell'esperienza consiste nella sintesi di un segnale a partire dai suoi coefficienti di Fourier. Vogliamo ottenere un grafico che ci permetta di studiare come varia il segnale in base al numero di coefficienti usati.

La seconda parte, invece, ha come scopo la sintesi di tre segnali distinti:

- Onda Sinusoidale
- Onda Triangolare
- Onda Quadra

Tutti e tre a 100Hz, 200Hz e 440Hz. Vogliamo studiare in frequenza questi segnali tramite la Trasformata di Fourier e anche la somma delle tre onde Sinusoidali.

Strumenti e Librerie

Tutti i programmi realizzati sono scritti in Python e sfruttano le funzioni della libreria "Numpy" per computare la Trasformata di Fourier e tutti i calcoli del caso. Per sintetizzare alcuni segnali sfruttiamo la libreria "Scipy". Per i grafici, invece, la libreria scelta è "Matplotlib". Infine, tutte le immagini del codice sono state generate tramite l'estensione "CodeSnap" di Visual Studio Code.

Trasformata di Fourier

La Trasformata di Fourier è uno strumento matematico utile nello studio dei segnali. Può essere utilizzata per vari scopi, tra cui:

- Scomposizione in Frequenze, ad esempio per analizzare lo spettro di frequenza in fase di Mixing di un brano musicale
- Analisi di Segnali, per capire quali frequenze troviamo in un segnale
- Filtering dei Segnali, ad esempio per rimuovere o isolare particolari frequenze di un segnale

La serie di Fourier è un caso speciale della Trasformata di Fourier. Un segnale, nella realtà, è un'onda periodica che è composta da una somma di armoniche a intensità diverse. Il tono del segnale è determinato dalle frequenze di queste armoniche, mentre il timbro dipende dall'intensità delle stesse.

Nel nostro caso, utilizziamo la Serie di Fourier per approssimare dei segnali e per poi studiarli.

Parte Prima

Vogliamo realizzare un programma Python per sintetizzare un segnale di Treno di Impulsi a partire dai suoi coefficienti di Fourier per poi generare un grafico della sua forma d'onda e studiarlo in frequenza.

$$x(t) = \sum_{k=-\infty}^{\infty} X_k e^{i2\pi k f_o t}$$
$$x(t) = \frac{1}{2} + \frac{2}{\pi} \sum_{k=1}^{\infty} \frac{1}{k} (-1)^{(k-1)/2} \cos(2\pi k f_o t) \quad k \text{ dispari}$$

Figura 1 - Serie di Fourier

Per prima cosa, importiamo le librerie necessarie per poterle utilizzare nel codice:

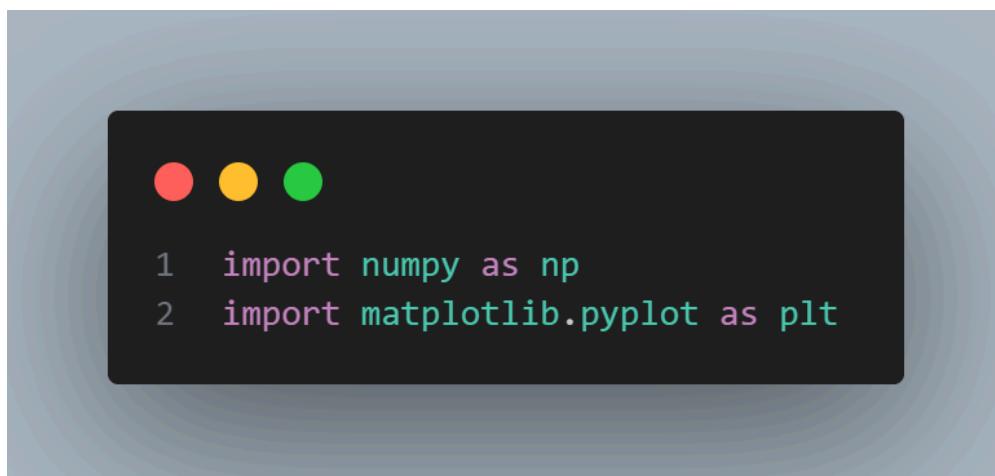
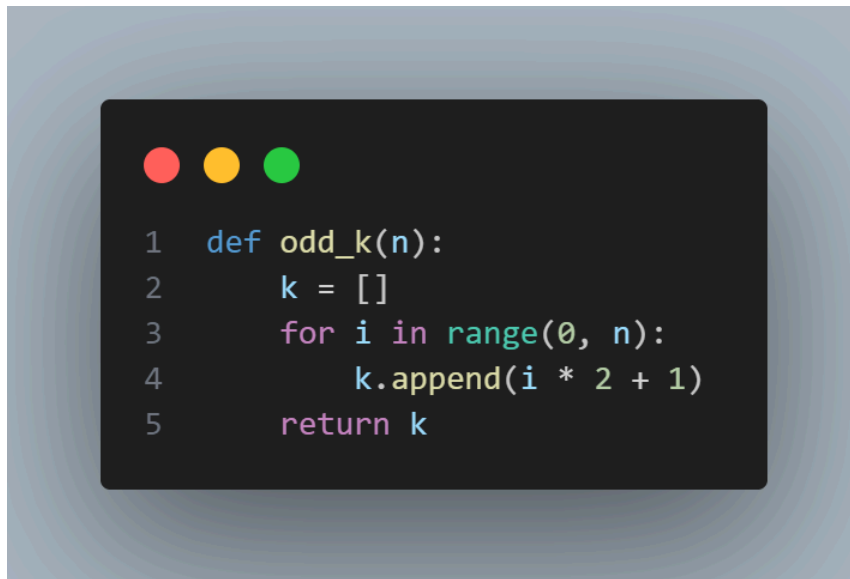


Figura 2

Odd_k()

Ora possiamo entrare nel vivo del codice. Come si vede nella Figura 1, dobbiamo fare in modo da utilizzare solo i “k” dispari. Per fare ciò, è necessaria la funzione “odd_k”, che si occupa di generare una lista di “n” numeri dispari. Il valore “n” va indicato alla chiamata della funzione ed è arbitrario. Lo sfrutteremo più avanti per mostrare come cambia il grafico al variare del numero di coefficienti.

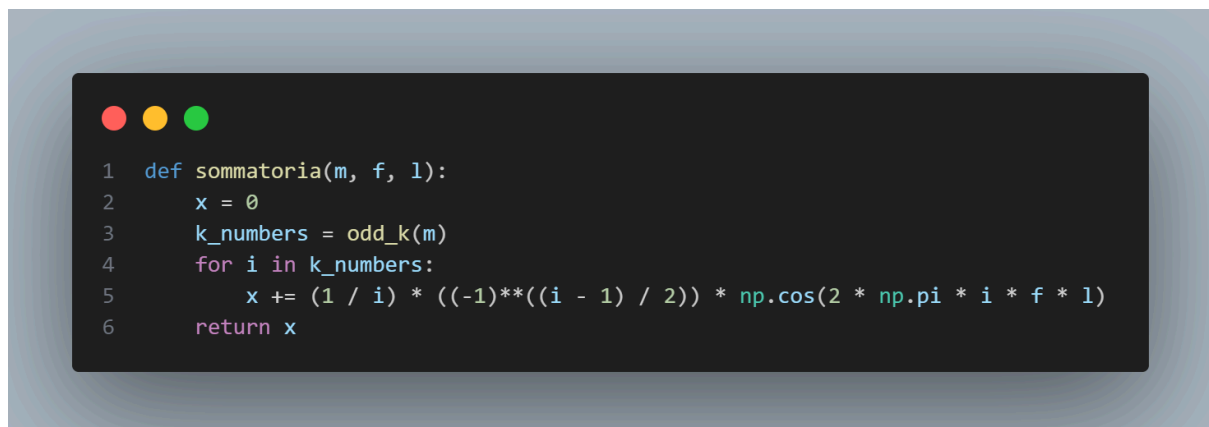


```
1 def odd_k(n):
2     k = []
3     for i in range(0, n):
4         k.append(i * 2 + 1)
5     return k
```

Figura 3 - Funzione odd_k()

Calcolo della Sommatoria

Adesso che siamo in grado di generare una lista di numeri dispari arbitraria, il prossimo passo è calcolare la sommatoria della Serie di Fourier nella Figura 1. Per farlo, scriviamo una funzione apposita.



```
1 def sommatoria(m, f, l):
2     x = 0
3     k_numbers = odd_k(m)
4     for i in k_numbers:
5         x += (1 / i) * ((-1)**((i - 1) / 2)) * np.cos(2 * np.pi * i * f * l)
6     return x
```

Figura 4 - Funzione sommatoria()

All'interno del ciclo "for" è presente una somma ripetuta nella variabile "x". Questo è proprio il procedimento tipico di una sommatoria, una somma ripetuta su "m" iterazioni. In questo caso, "m" è proprio il numero di coefficienti "k" che generiamo chiamando "odd_k(m)". I parametri "f" e "l" che passiamo alla funzione "sommatoria()" sono rispettivamente la frequenza del segnale e l'array di valori che rappresenta il tempo da 0 a 1 secondi.

DFT e Generazione del Grafico

L'ultimo passo è computare la Discrete Fourier Transform e la generazione del grafico stesso.



Figura 5

Come si vede in Figura 5, la frequenza del segnale che stiamo cercando di approssimare è 2Hz. Utilizziamo un "for" che itera 3 diverse DFT, proprio per far notare la differenza tra 5, 50 e 5000 coefficienti. Moltiplichiamo la parte fuori dalla sommatoria della Serie di Fourier per la sommatoria stessa, facendo attenzione a passare i parametri corretti alla funzione. Infine generiamo il grafico, specificando la legenda e le descrizioni dei due assi.

Risultato Finale

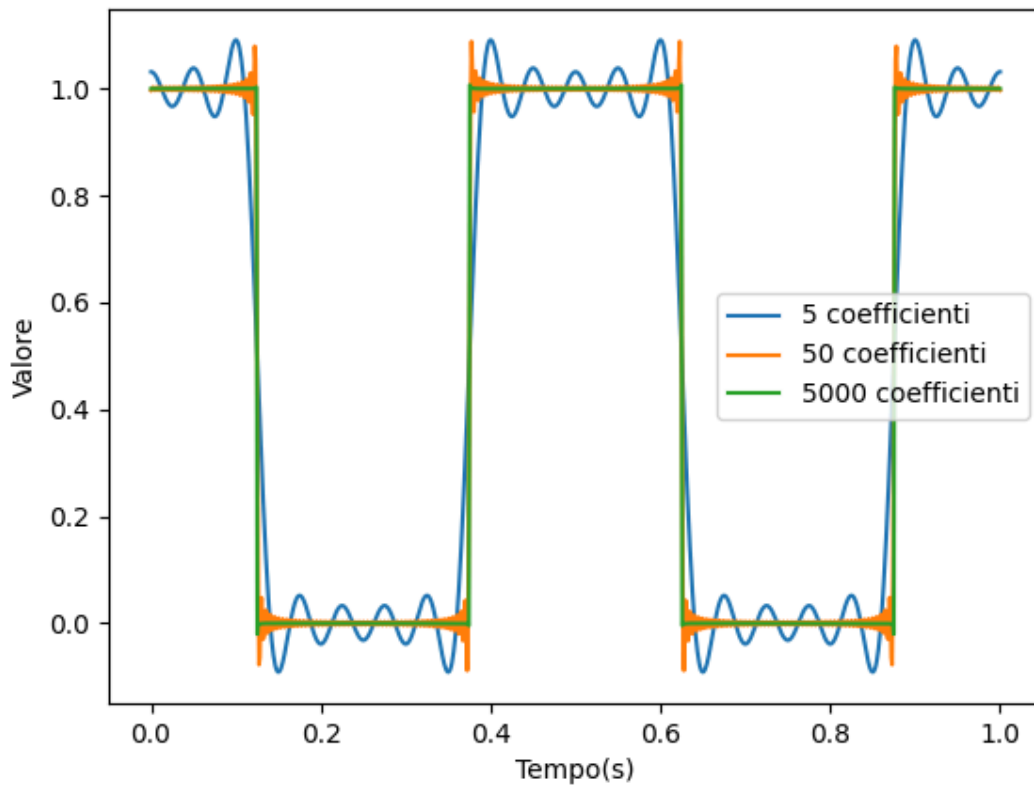


Figura 6

Nel grafico risultante si nota molto bene la differenza tra 5, 50 e 5000 coefficienti. Mentre con pochi coefficienti la forma d'onda è molto approssimativa e poco precisa, all'aumentare di essi otteniamo un risultato sempre più preciso e vicino alla realtà. Possiamo concludere che aumentando il numero di coefficienti, aumenta anche la precisione dell'approssimazione.

Parte Seconda


Vogliamo realizzare un programma Python per generare un plot nel tempo di 9 segnali distinti:

- Onda sinusoidale a 100 Hz, 200 Hz, 440 Hz
- Onda triangolare a 100 Hz, 200 Hz, 440 Hz
- Onda quadra a 100 Hz, 200 Hz, 440 Hz

Infine vogliamo studiare in frequenza i segnali stessi e la somma delle tre Onde Sinusoidali.

Sintesi dei Segnali

Innanzitutto dobbiamo sintetizzare i segnali che vogliamo studiare.

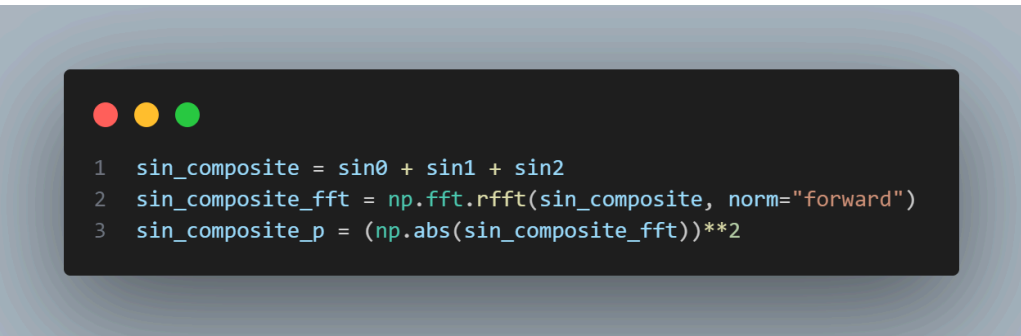


```
1 sin0 = 1 * np.sin(2*np.pi * f0 * t)
2 sin1 = 1 * np.sin(2*np.pi * f1 * t)
3 sin2 = 1 * np.sin(2*np.pi * f2 * t)
4
5 st0 = np.abs(sp.signal.sawtooth(2 * np.pi * f0 * t))
6 st1 = np.abs(sp.signal.sawtooth(2 * np.pi * f1 * t))
7 st2 = np.abs(sp.signal.sawtooth(2 * np.pi * f2 * t))
8
9 sq0 = sp.signal.square(2*np.pi * f0 * t)
10 sq1 = sp.signal.square(2*np.pi * f1 * t)
11 sq2 = sp.signal.square(2*np.pi * f2 * t)
```

Figura 7

I parametri f_0 , f_1 ed f_2 sono le tre frequenze: 100Hz, 200Hz e 440Hz. Il parametro "t" è l'array del tempo e va da 0 a 0.03 secondi.

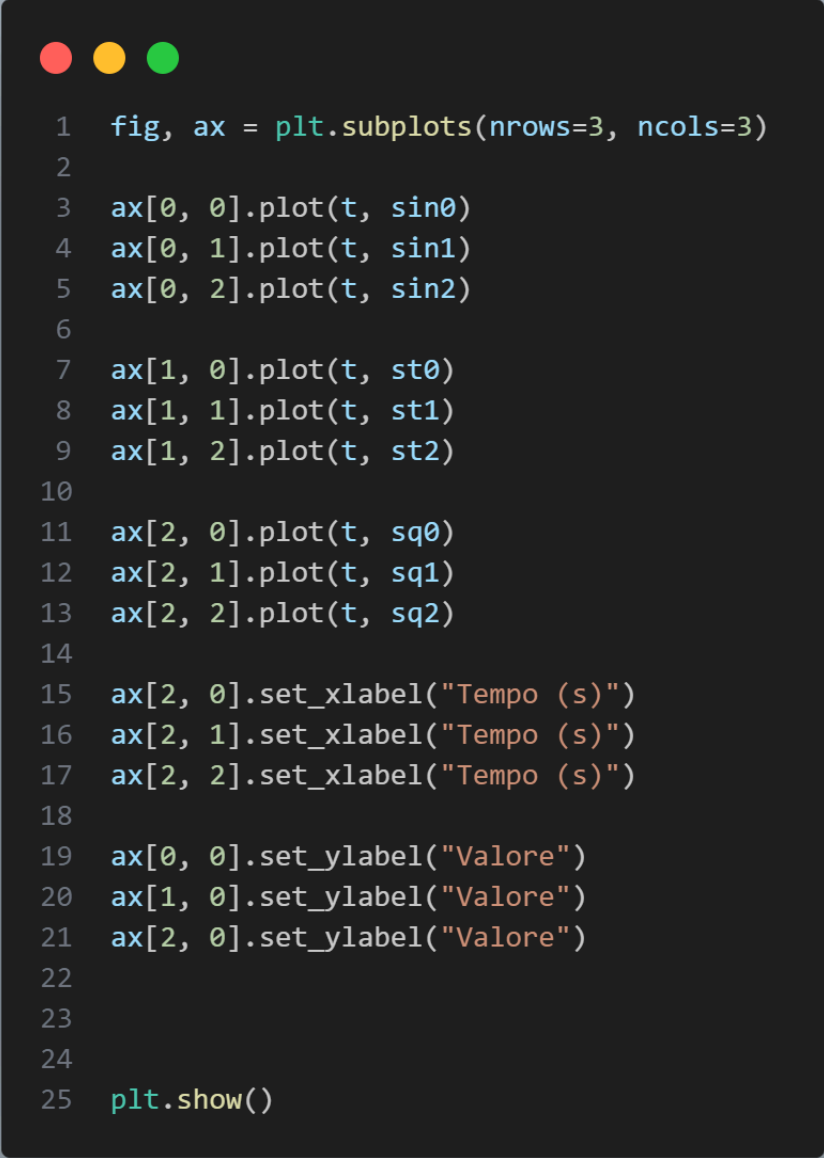
L'unica differenza per la somma delle onde Sinusoidali è che per ottenere il segnale richiesto dobbiamo calcolare la somma:



```
1 sin_composite = sin0 + sin1 + sin2
2 sin_composite_fft = np.fft.rfft(sin_composite, norm="forward")
3 sin_composite_p = (np.abs(sin_composite_fft))**2
```

Grafici dei Segnali Sintetizzati

Il prossimo passo è generare i grafici dei segnali precedentemente sintetizzati. Utilizziamo la funzione “subplots” di Matplotlib per suddividere i 9 grafici.



```
1  fig, ax = plt.subplots(nrows=3, ncols=3)
2
3  ax[0, 0].plot(t, sin0)
4  ax[0, 1].plot(t, sin1)
5  ax[0, 2].plot(t, sin2)
6
7  ax[1, 0].plot(t, st0)
8  ax[1, 1].plot(t, st1)
9  ax[1, 2].plot(t, st2)
10
11 ax[2, 0].plot(t, sq0)
12 ax[2, 1].plot(t, sq1)
13 ax[2, 2].plot(t, sq2)
14
15 ax[2, 0].set_xlabel("Tempo (s)")
16 ax[2, 1].set_xlabel("Tempo (s)")
17 ax[2, 2].set_xlabel("Tempo (s)")
18
19 ax[0, 0].set_ylabel("Valore")
20 ax[1, 0].set_ylabel("Valore")
21 ax[2, 0].set_ylabel("Valore")
22
23
24
25 plt.show()
```

Figura 8

Il risultato finale saranno 9 grafici, uno per ogni frequenza e 3 per ogni tipo di onda.

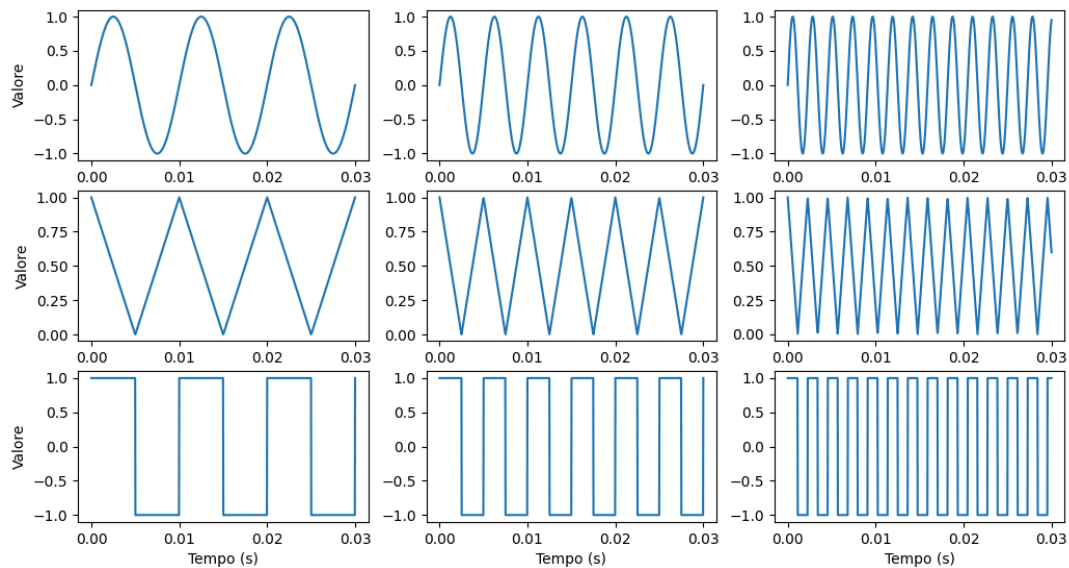


Figura 9

Studio in Frequenza

Per poter studiare in frequenza i nostri segnali, dobbiamo prima computarne la Trasformata di Fourier, visto che la useremo per mostrare nel grafico la sua parte reale, immaginaria e lo spettro di potenza dei segnali.

```
1 sin0_fft = np.fft.rfft(sin0, norm="forward")
2 sin0_p = (np.abs(sin0_fft))**2
```

Figura 10

“sin0_fft” è il calcolo della Trasformata di Fourier. “sin0_p” è invece il calcolo della Potenza, che si ottiene elevando al quadrato il valore assoluto della Trasformata di Fourier.

Ripetiamo questo calcolo per tutti gli 8 restanti segnali, esattamente alla stessa maniera ma cambiando semplicemente “sin0” in “sin1, 2” e così via, anche con gli altri segnali.

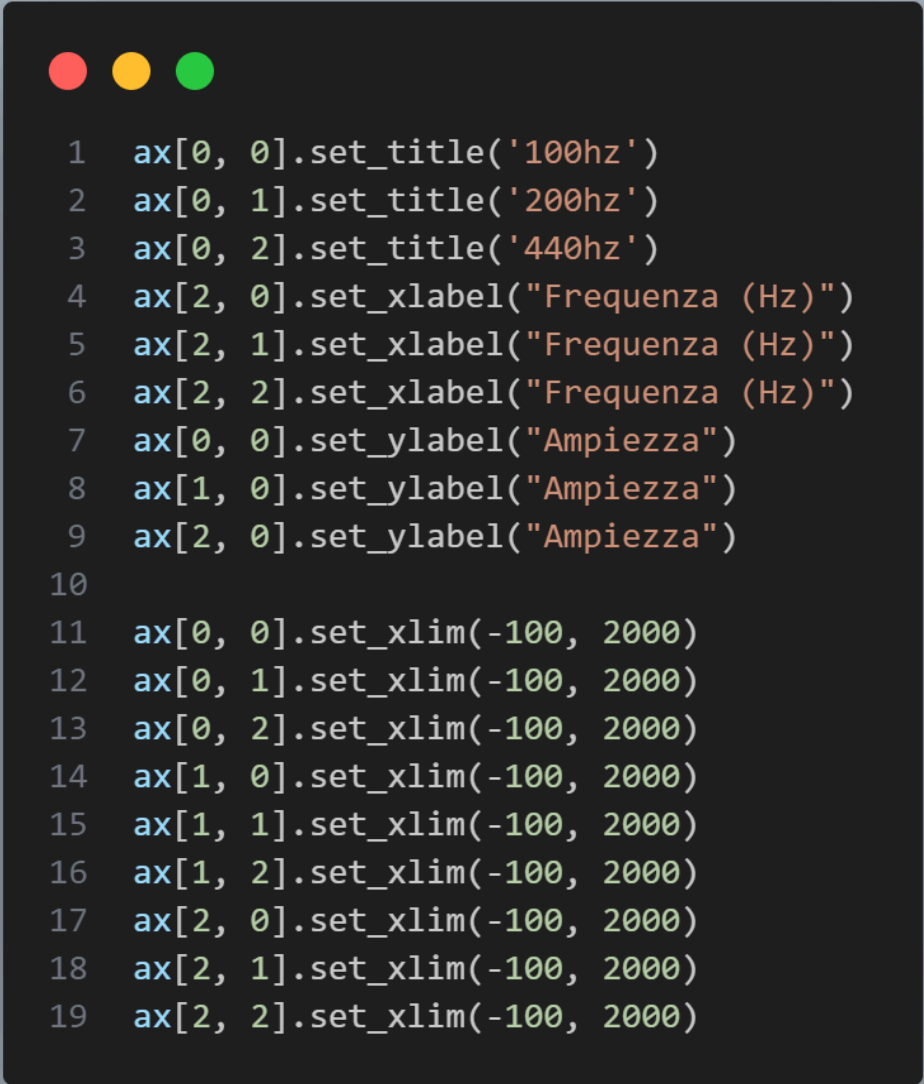
Generazione dei Grafici

Anche in questo caso facciamo uso della funzione “subplots” di Matplotlib per gestire i grafici da generare.



Figura 11

Lo stesso procedimento va eseguito per tutti i segnali. Infine, basta impostare i titoli degli assi dei grafici e dei limiti per l'asse x (al fine di rendere più leggibili i grafici).



```
1 ax[0, 0].set_title('100hz')
2 ax[0, 1].set_title('200hz')
3 ax[0, 2].set_title('440hz')
4 ax[2, 0].set_xlabel("Frequenza (Hz)")
5 ax[2, 1].set_xlabel("Frequenza (Hz)")
6 ax[2, 2].set_xlabel("Frequenza (Hz)")
7 ax[0, 0].set_ylabel("Ampiezza")
8 ax[1, 0].set_ylabel("Ampiezza")
9 ax[2, 0].set_ylabel("Ampiezza")
10
11 ax[0, 0].set_xlim(-100, 2000)
12 ax[0, 1].set_xlim(-100, 2000)
13 ax[0, 2].set_xlim(-100, 2000)
14 ax[1, 0].set_xlim(-100, 2000)
15 ax[1, 1].set_xlim(-100, 2000)
16 ax[1, 2].set_xlim(-100, 2000)
17 ax[2, 0].set_xlim(-100, 2000)
18 ax[2, 1].set_xlim(-100, 2000)
19 ax[2, 2].set_xlim(-100, 2000)
```

Figura 12

Risultato Finale

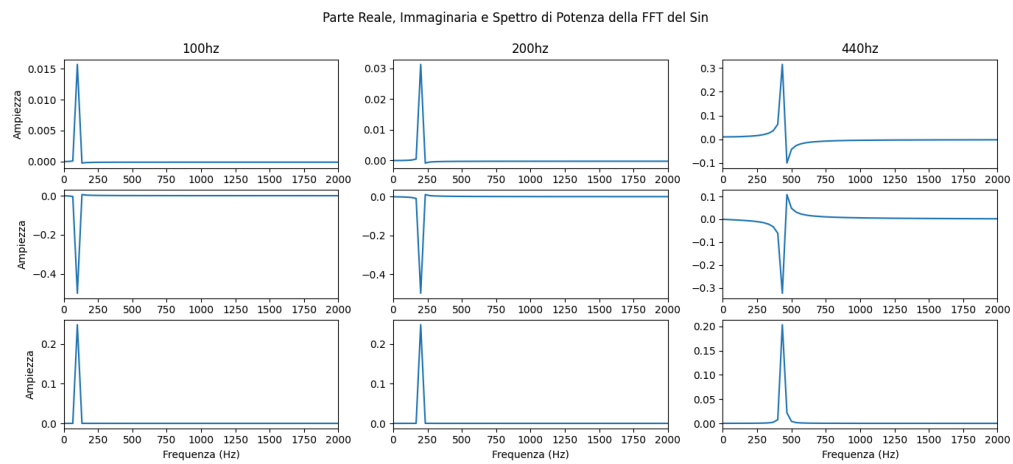


Figura 13

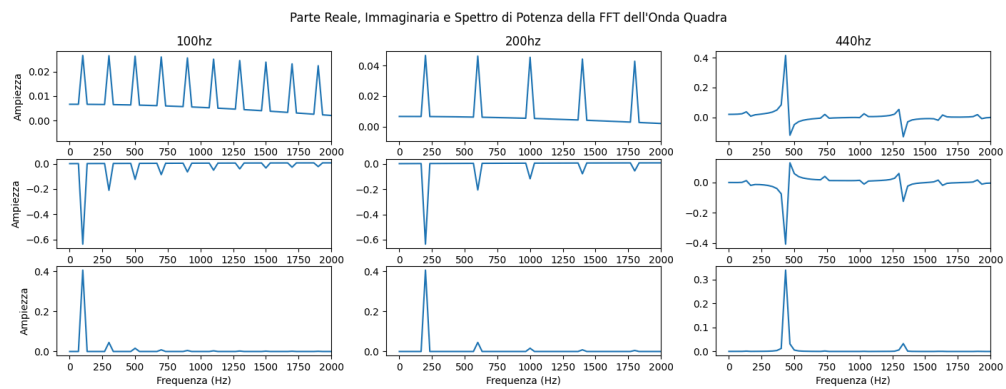


Figura 14

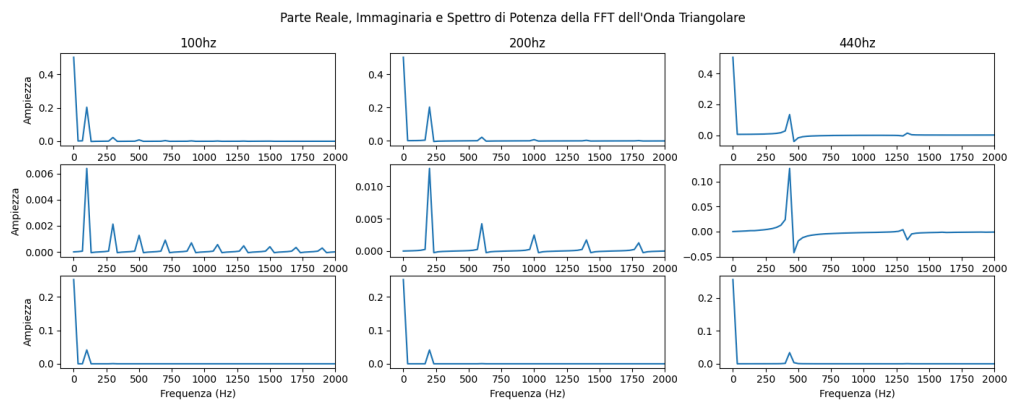


Figura 15

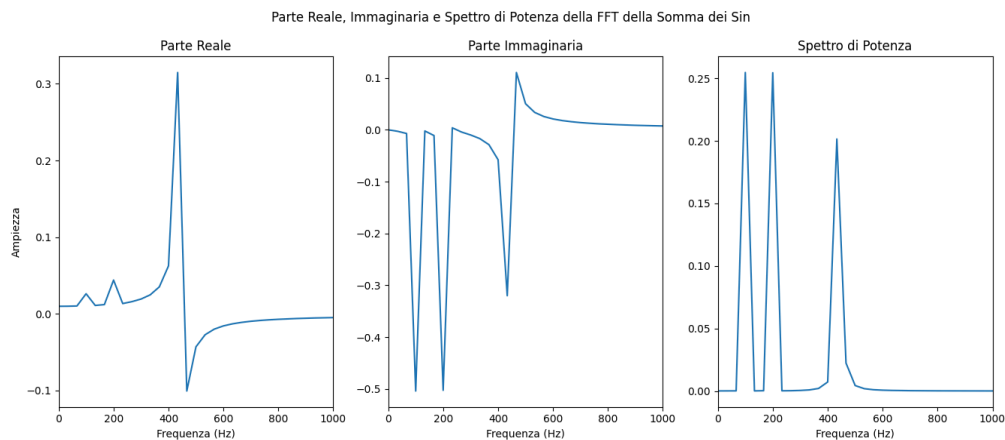


Figura 16

Come è possibile notare dalle Figure 13, 14, 15 e 16, aumentando la frequenza anche la Trasformata di Fourier (e di conseguenza lo spettro di potenza) cambia. Il “picco” del grafico si sposta verso la frequenza del segnale stesso. Nella Figura 16, in particolare, notiamo come, essendo il segnale una somma dello stesso segnale a 3 frequenze differenti, il risultato sarà diverso dai tre grafici precedenti. Ci sono tre differenti picchi nei grafici, rispettivamente a circa 100Hz, 200Hz e 440Hz. Il motivo è chiaramente la somma stessa: infatti, nello spettro di potenza i tre picchi sono quasi equivalenti e sono localizzati esattamente nelle nostre tre frequenze scelte.

Conclusioni

Grazie a questa esperienza, abbiamo notato come la Trasformata e la Serie di Fourier siano uno strumento matematico molto utile sia per studiare che per approssimare segnali.

Nella Parte Prima, ci siamo accorti come all'aumentare dei coefficienti della serie l'approssimazione del segnale diventi sempre più precisa.

Nella Parte Seconda, abbiamo notato che la Trasformata di Fourier e lo Spettro di Potenza di un segnale dipendono fortemente dalla frequenza del segnale stesso. Infatti, sommando tre segnali identici ma a frequenze differenti, il risultato che otteniamo rispecchia esattamente quello che ci eravamo immaginati: tre picchi alle tre frequenze dei tre segnali iniziali sommati.