Laboratorio di Elettronica e Tecniche di Acquisizione Dati
2025-2026

Esercitazione 5
"Moises"

- parte A: studio di segnali "semplici". Basta la teoria sullo studio di segnali periodici e continui
- parte B: studio di segnali audio. Serve la teoria completa (segnali aperiodici e campionati)

Esercitazione – parte A

realizzare un piccolo programma python per:

- aprire un piccolo file di testo (.txt) e plottarne la waveform

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy import constants, fft

#------------------------------------------------------------------

name = "data1.txt"
#name = "data2.txt"
#name = "data3.txt"

samplerate = 44100

#------------------------------------------------------------------

with open('./'+name, 'r') as data:
    datax = []
    datay = []
    for line in data:
        p = line.split()
        datax.append(float(p[0]))
        datay.append(float(p[1]))

print(samplerate)
#print(datax)
print(len(datax))
#print(datay)
print(len(datay))
```

data1.txt

0	0
2.26757e-05	0.0515434
4.53515e-05	0.10289
6.80272e-05	0.153843
9.07029e-05	0.204209
0.000113379	0.253794
0.000136054	0.30241
0.00015873	0.349871
0.000181406	0.395996
0.000204082	0.440609
0.000226757	0.483541
0.000249433	0.524629
0.000272109	0.563716
0.000294785	0.600655
0.00031746	0.635306
0.000340136	0.667538
0.000362812	0.69723
0.000385488	0.724272
0.000408163	0.748561
0.000430839	0.770009
0.000453515	0.788537
0.00047619	0.804076
0.000498866	0.816573
0.000521542	0.825982
0.000544218	0.832273
0.000566893	0.835426
0.000589569	0.835434
0.000612245	0.832303
0.000634921	0.826051
0.000657596	0.816707

Ogni file (due colonne: tempo in secondi e ampiezza in u.a.) rappresenta 10 s di audio, campionato a 44100 Hz.

Link:

- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/data1.txt
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/data2.txt
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/data3.txt

Esercitazione – parte A

realizzare un piccolo programma python per:

- aprire un piccolo file di testo (.txt) e plottarne la waveform
- fare la FFT dell'array ottenuto dal file e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti

Esercitazione – parte A

realizzare un piccolo programma python per:

- aprire un piccolo file di testo (.txt) e plottarne la waveform
- fare la FFT dell'array ottenuto dal file e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
  - usando la libreria FFT di python

```
datafft  = fft.rfft(datamono) # n/2+1
#print(datafft.size)
print(len(datamono))
#fftfreq = 0.5*fft.rfftfreq(datafft.size, 1.0/samplerate) # this is how Stefano Germani did: the 0.5 he called "nyquist":
#"Unlike fftfreq (but like scipy.fftpack.rfftfreq) the Nyquist frequency component is considered to be positive."
fftfreq = fft.rfftfreq(len(datamono), 1.0/samplerate)

print(datafft)
print(len(datafft))
print(fftfreq)
print(len(fftfreq))

plt.figure(20)
fig = plt.gcf()
fig.set_size_inches(10, 8)
plt.title("FFT")
plt.xlabel('Frequenza (hz)')
plt.ylabel('Ampiezza (parte reale)(u.a)')
plt.plot(fftfreq[:len(datafft)], datafft[:len(datafft)].real)
```

Esercitazione – parte A

realizzare un piccolo programma python per:

- aprire un piccolo file di testo (.txt) e plottarne la waveform
- fare la FFT dell'array ottenuto dal file e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
  - usando la libreria FFT di python
  - che tipi di segnale contiene il file?

Esercitazione – parte A

realizzare un piccolo programma python per:

- aprire un piccolo file di testo (.txt) e plottarne la waveform
- fare la FFT dell'array ottenuto dal file e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
- ri-sintetizzare il segnale a partire da quello in frequenza

Esercitazione – parte A

realizzare un piccolo programma python per:

- aprire un piccolo file di testo (.txt) e plottarne la waveform
- fare la FFT dell'array ottenuto dal file e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
- ri-sintetizzare il segnale a partire da quello in frequenza
  - utilizzando la libreria FFT di python

```
syntdata = fft.irfft(datafft_cut,  n=len(times))
```

Esercitazione – parte A

realizzare un piccolo programma python per:

- aprire un piccolo file di testo (.txt) e plottarne la waveform
- fare la FFT dell'array ottenuto dal file e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
- ri-sintetizzare il segnale a partire da quello in frequenza
- mascherare (i.e. ponendo a zero i coefficienti associati alla componente di rumore sinusoidale) il segnale

Esercitazione – parte A

realizzare un piccolo programma python per:

- aprire un piccolo file di testo (.txt) e plottarne la waveform
- fare la FFT dell'array ottenuto dal file e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
- ri-sintetizzare il segnale a partire da quello in frequenza
- mascherare (i.e. ponendo a zero i coefficienti associati alla componente di rumore sinusoidale) il segnale
- ri-sintetizzare il segnale a partire da quello in frequenza, filtrato
  - utilizzando la libreria FFT di python

Esercitazione – parte A

realizzare un piccolo programma python per:

- aprire un piccolo file di testo (.txt) e plottarne la waveform
- fare la FFT dell'array ottenuto dal file e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
- ri-sintetizzare il segnale a partire da quello in frequenza
- mascherare (i.e. ponendo a zero i coefficienti associati alla componente di rumore sinusoidale) il segnale
- ri-sintetizzare il segnale a partire da quello in frequenza, filtrato
  - utilizzando la libreria FFT di python
  - utilizzando seni e coseni (*np.sin* e *np.cos*)
    - l'unità di frequenza NON è la frequenza fondamentale del segnale (i.e. f_0), ma è quella di campionamento! Quanto è f_0?
    - *x(t)* è continua ma voi potete sintetizzare e plottare vs t con t ad intervalli regolari

$$x(t) = A_0 + 2\sum_{k=1}^{\infty} A_k \cos(2\pi k f_0 t + \phi_k)$$

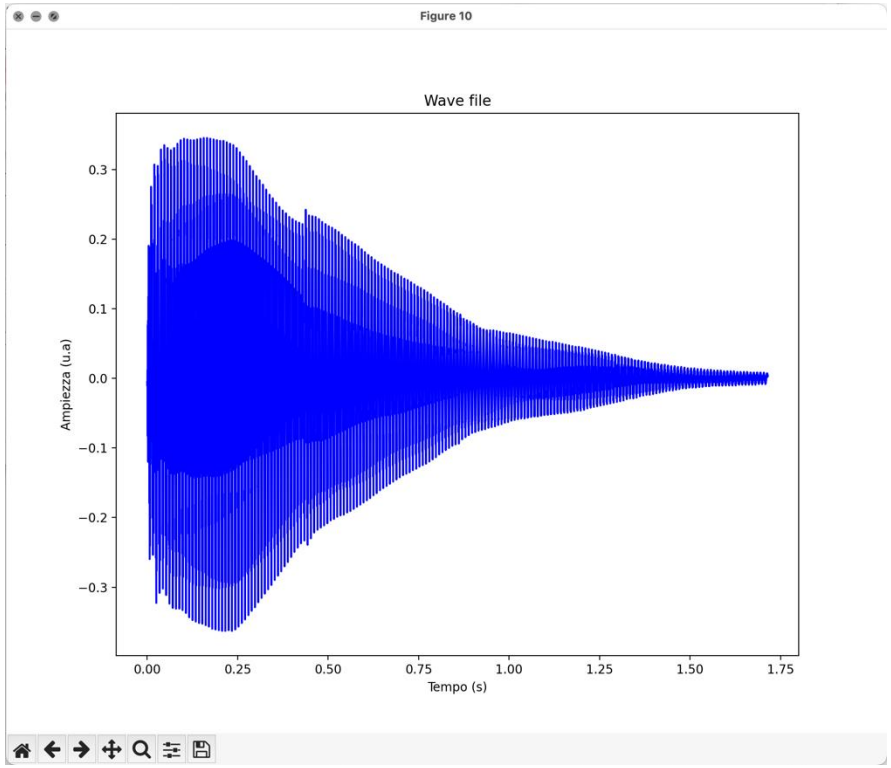
Esercitazione – parte B

realizzare un piccolo programma python:

- per aprire un piccolo file audio (.wav) e plottarne la waveform (solo un canale)
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo

Link parte 1:

- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/diapason.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_26-26/_slides/pulita_semplice.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/pulita_media.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/pulita_difficile.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/distorta.wav



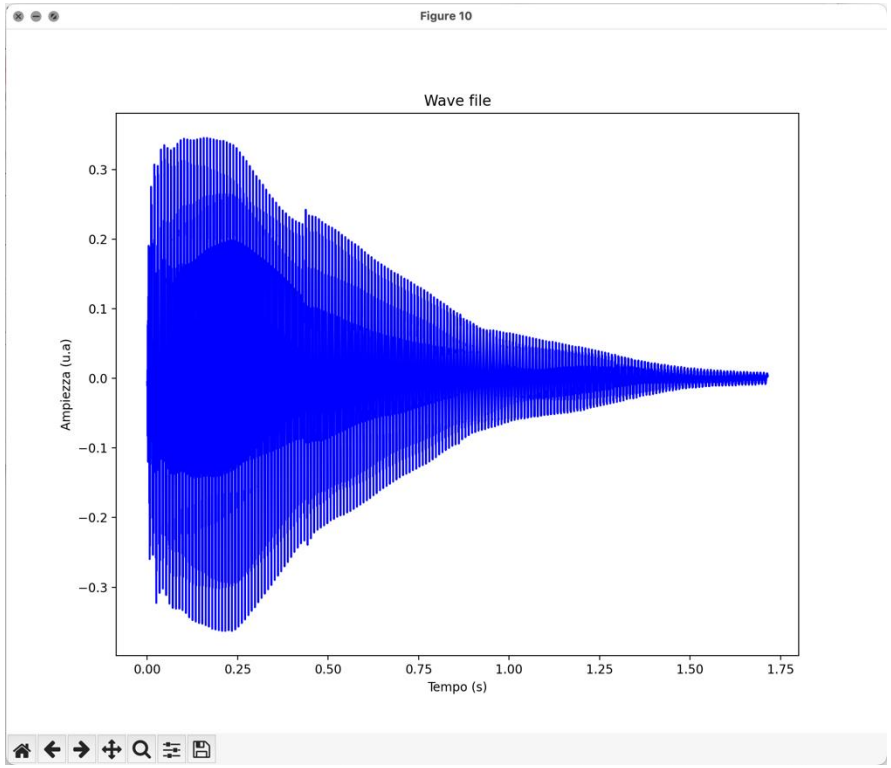
Esercitazione – parte B

realizzare un piccolo programma python:

- per aprire un piccolo file audio (.wav) e plottarne la waveform (solo un canale)
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo

Link parte 2:

- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/pulita_pezzo.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/distorta_pezzo.wav



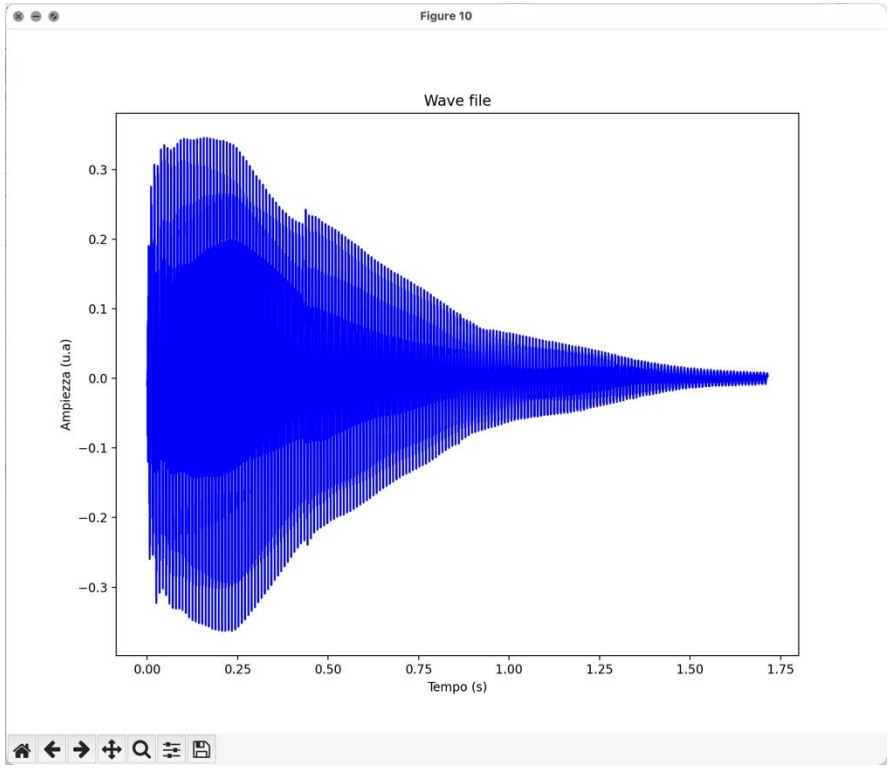
Esercitazione – parte B

realizzare un piccolo programma python:

- per aprire un piccolo file audio (.wav) e plottarne la waveform (solo un canale)
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo

Link parte 3:

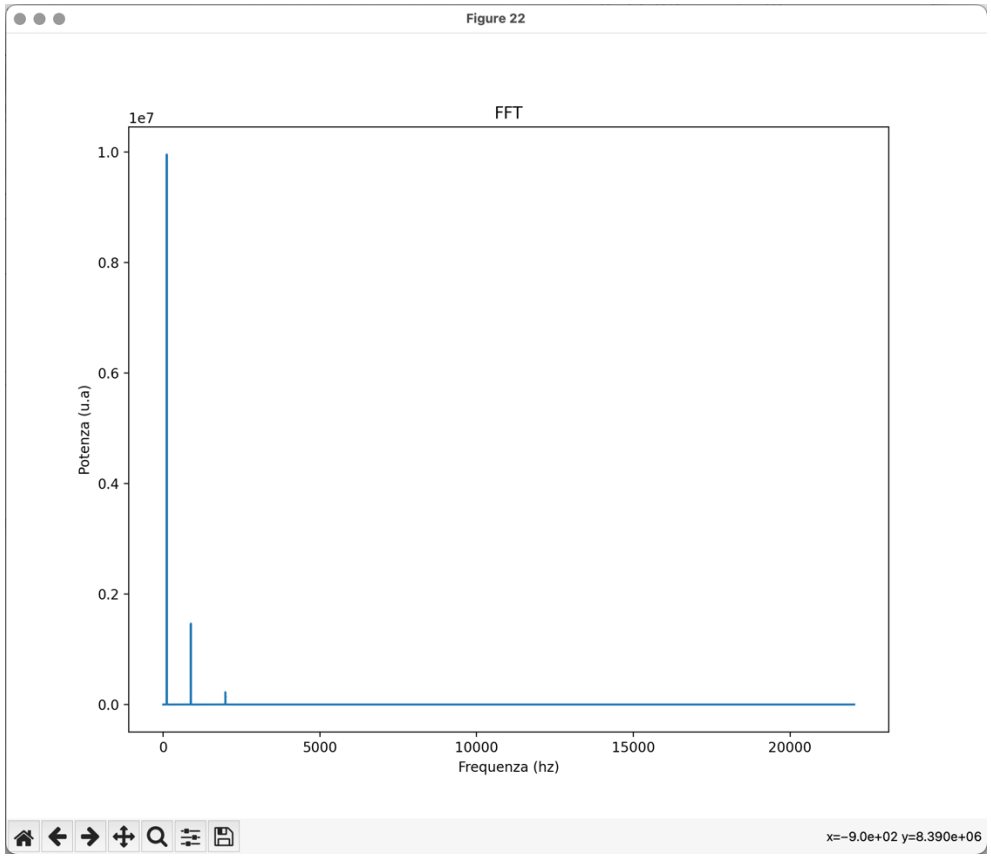
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/primo.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_25-26/_slides/secondo.wav



Esercitazione – parte B

realizzare un piccolo programma python:

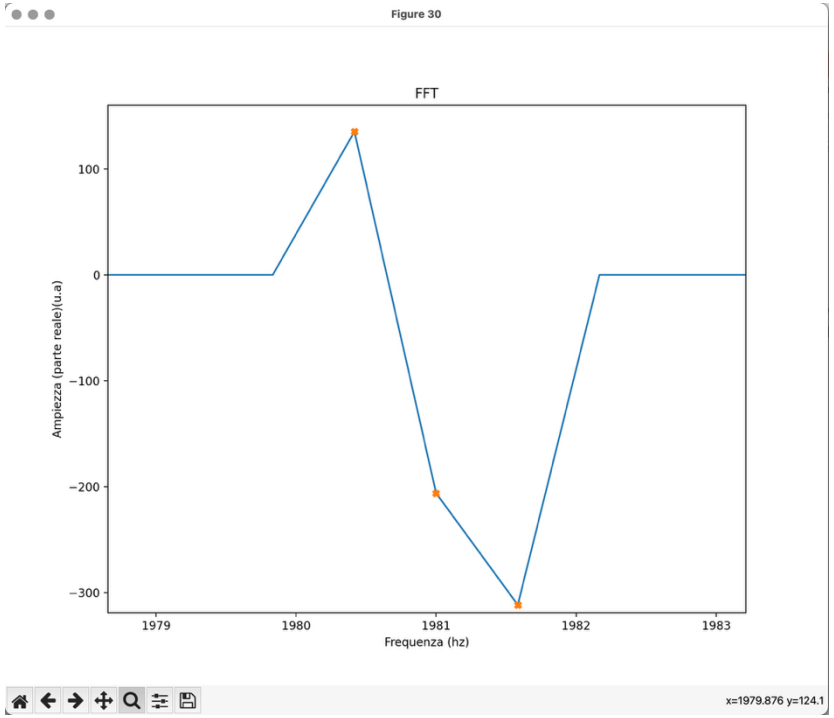
- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
  - usando la libreria FFT di python



Esercitazione – parte B

realizzare un piccolo programma python:

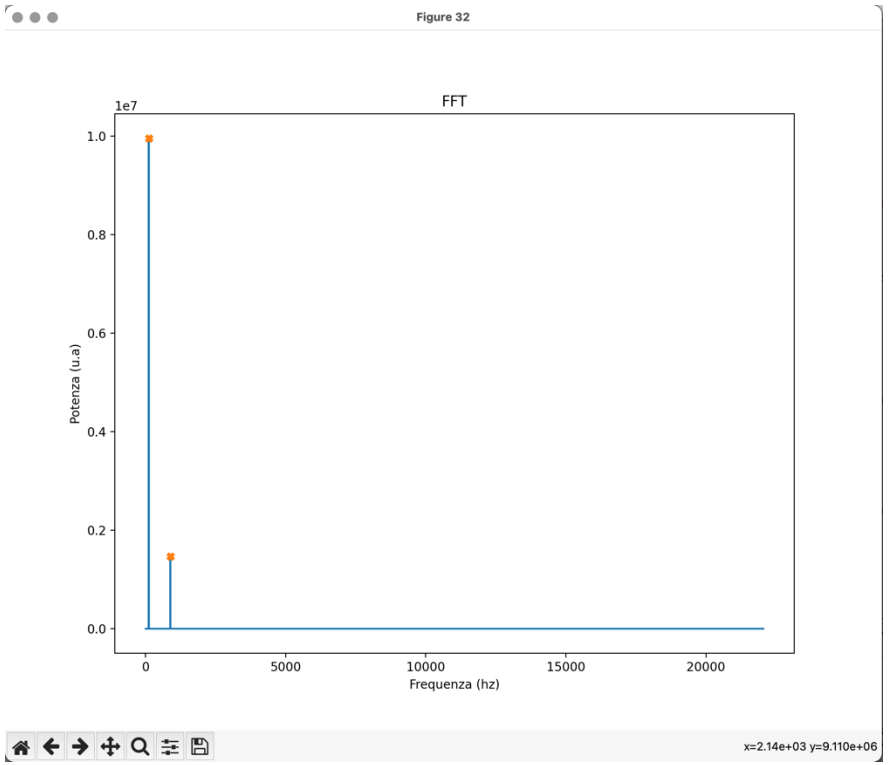
- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
- (solo parte 1) identificare i "picchi"
  - che nota è?
    https://www.audiosonica.com/it/corsoaudio-online/conversione-tra-note-musicali-e-frequenze-appendice-i
  - che accordo è?
  - quanto è "largo" ogni picco?



Esercitazione – parte B

realizzare un piccolo programma python:

- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
- identificare i "picchi"
- (parte 1 e 2) mascherare (i.e. mettere a zero) i coefficienti tranne alcuni "scelti"
  - il picco principale
  - i primi due picchi principali, ma solo il termine "centrale"
  - i picchi principali, ma solo il termine "centrale"
  - i picchi principali con anche 1 o 2 termini, per lato, oltre quello centrale

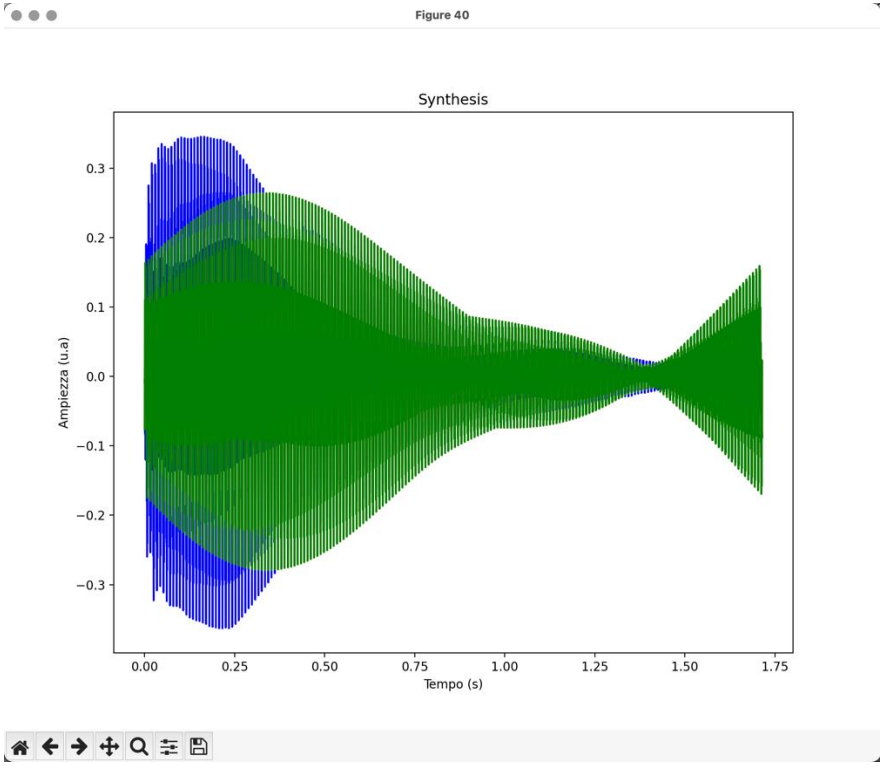


Esercitazione – parte B

realizzare un piccolo programma python:

- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
- identificare i "picchi"
- mascherare (i.e. mettere a zero) i coefficienti tranne alcuni "scelti"
- (parte 1 e 2) "sintetizzare" l'array di dati ("filtrati") e produrre un file audio (.wav)
  - utilizzando la libreria FFT di python
  - utilizzando seni e coseni (*np.sin* e *np.cos*)

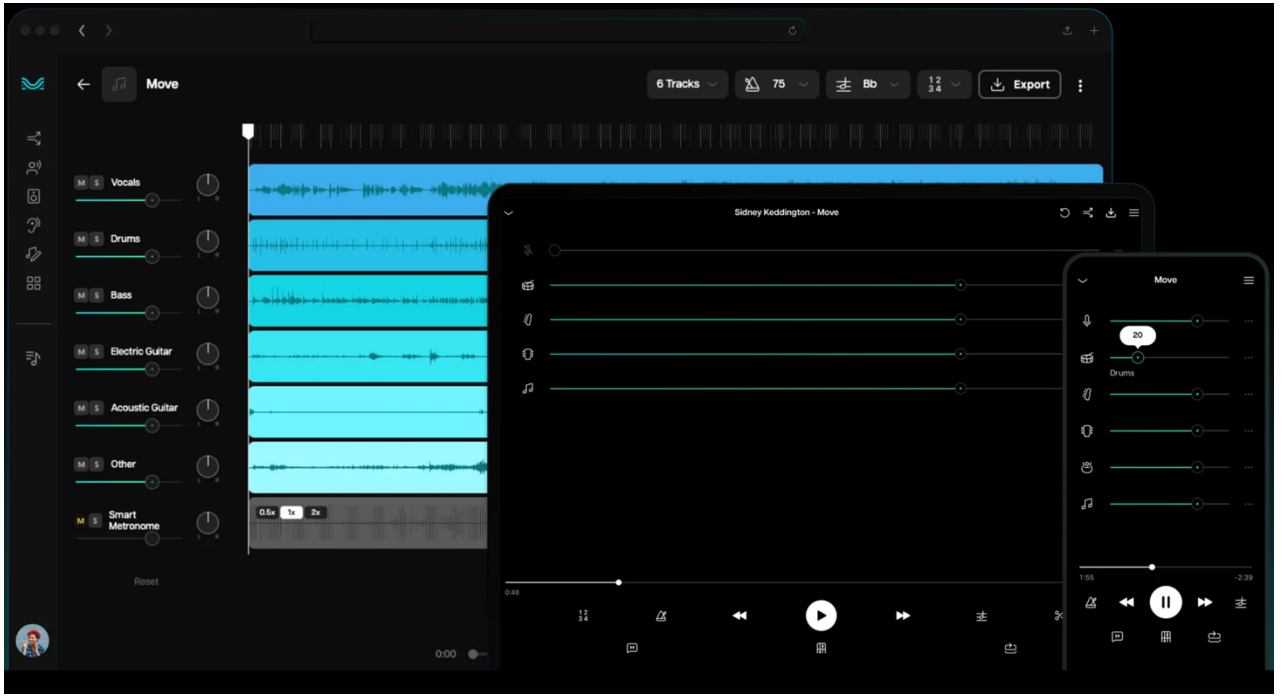
$$x[n] = \sum_{k=0}^{N_0-1} X_{s,k}\, e^{i2\pi nk/N_0}$$



Esercitazione – parte B

realizzare un piccolo programma python:

- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, fase, parte reale e parte immaginaria dei coefficienti
- identificare i "picchi"
- mascherare (i.e. mettere a zero) i coefficienti tranne alcuni "scelti"
- (parte 3) separare i due diversi strumenti presenti e "sintetizzare" l'array di dati ("filtrati") e produrre un file audio (.wav) per ciascuno strumento
  - utilizzando la libreria FFT di python



Lettura/Scrittura file testo

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import constants, fft

#-----------------------------------------------------------------

name = "data1.txt"
#name = "data2.txt"
#name = "data3.txt"

samplerate = 44100

#-----------------------------------------------------------------

with open('./'+name, 'r') as data:
    datax = []
    datay = []
    for line in data:
        p = line.split()
        datax.append(float(p[0]))
        datay.append(float(p[1]))

print(samplerate)
#print(datax)
print(len(datax))
#print(datay)
print(len(datay))

times = datax.copy()
datamono = datay.copy()

#print(times)
print(len(times))

plt.figure(10)
fig = plt.gcf()
fig.set_size_inches(10, 8)
plt.title("Wave file")
plt.xlabel('Tempo (s)')
plt.ylabel('Ampiezza (u.a)')
plt.plot(times, datamono, color="blue")
#plt.show()

#-----------------------------------------------------------------

with open('./'+name+'_syntetized.txt', "w") as txt_file:
    for i in range(len(times)):
        txt_file.write(f'{times[i]}\t{syntdata[i]}\n')
```

Lettura/Scrittura file audio

Fonte: https://pysoundfile.readthedocs.io/en/latest/

```
import soundfile as sf
import numpy as np
import matplotlib.pyplot as plt
from scipy import constants, fft

#-----------------------------------------------------------------

data, samplerate = sf.read('./audio.wav')

print(samplerate)
print(data)
print(len(data))

sf.write('./audio_recreated.wav', data, samplerate)
```

FFT

Fonte: https://docs.scipy.org/doc/scipy/tutorial/fft.html
https://github.com/s-germani/metodi-computazionali-fisica/blob/main/notebooks/L06_TrasformateFourier.ipynb

```
datafft  = fft.rfft(datamono) # n/2+1
#print(datafft.size)
print(len(datamono))
#fftfreq = 0.5*fft.rfftfreq(datafft.size, 1.0/samplerate) # this is how Stefano Germani did: the 0.5 he called "nyquist":
#"Unlike fftfreq (but like scipy.fftpack.rfftfreq) the Nyquist frequency component is considered to be positive."
fftfreq = fft.rfftfreq(len(datamono), 1.0/samplerate)

print(datafft)
print(len(datafft))
print(fftfreq)
print(len(fftfreq))

plt.figure(20)
fig = plt.gcf()
fig.set_size_inches(10, 8)
plt.title("FFT")
plt.xlabel('Frequenza (hz)')
plt.ylabel('Ampiezza (parte reale)(u.a)')
plt.plot(fftfreq[:len(datafft)], datafft[:len(datafft)].real)
```

inverse-FFT

Fonte: https://docs.scipy.org/doc/scipy/tutorial/fft.html
https://github.com/s-germani/metodi-computazionali-fisica/blob/main/notebooks/L06_TrasformateFourier.ipynb

```
syntdata = fft.irfft(datafft_cut,  n=len(times))
```