

# **Laboratorio di Elettronica e Tecniche di Acquisizione Dati 2024-2025**

## **Esercitazione 5 "Moises"**

# Esercitazione – parte A

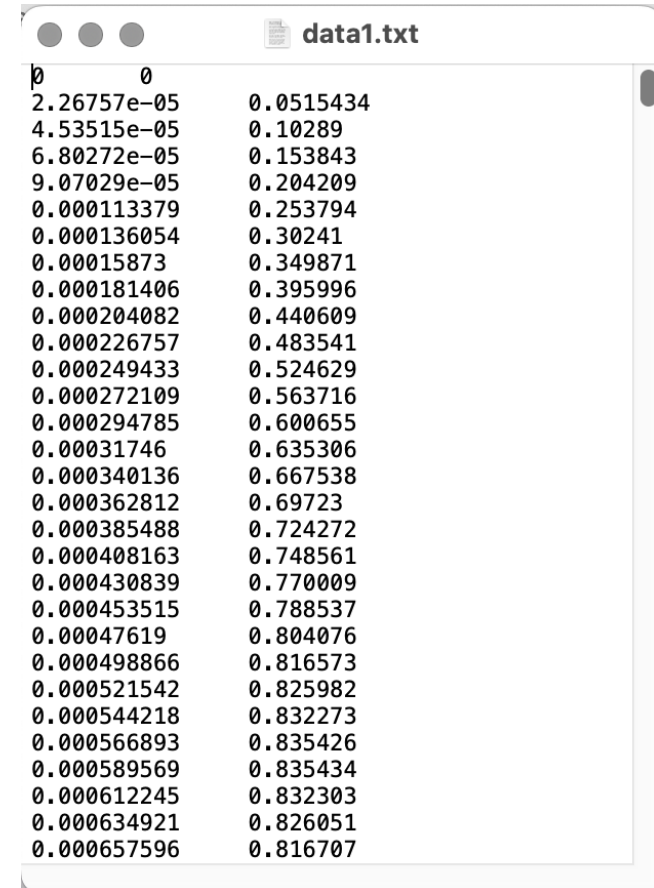
realizzare un piccolo programma python:

- per aprire un piccolo file di testo (.txt) e plottarne la waveform
- fare la FFT dell'array ottenuto dal file e plottare: potenza, parte reale e parte immaginaria dei coefficienti
  - usando la libreria FFT di python
  - che tipo di segnale è quello filtrato?
- ri-sintetizzare il segnale a partire da quello in frequenza
  - utilizzando la libreria FFT di python
  - utilizzando seni e coseni (*np.sin* e *np.cos*)
- 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

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\\_24-25/slides/data1.txt](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/data1.txt)
- [https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio\\_24-25/slides/data2.txt](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/data2.txt)
- [https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio\\_24-25/slides/data3.txt](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/data3.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  |

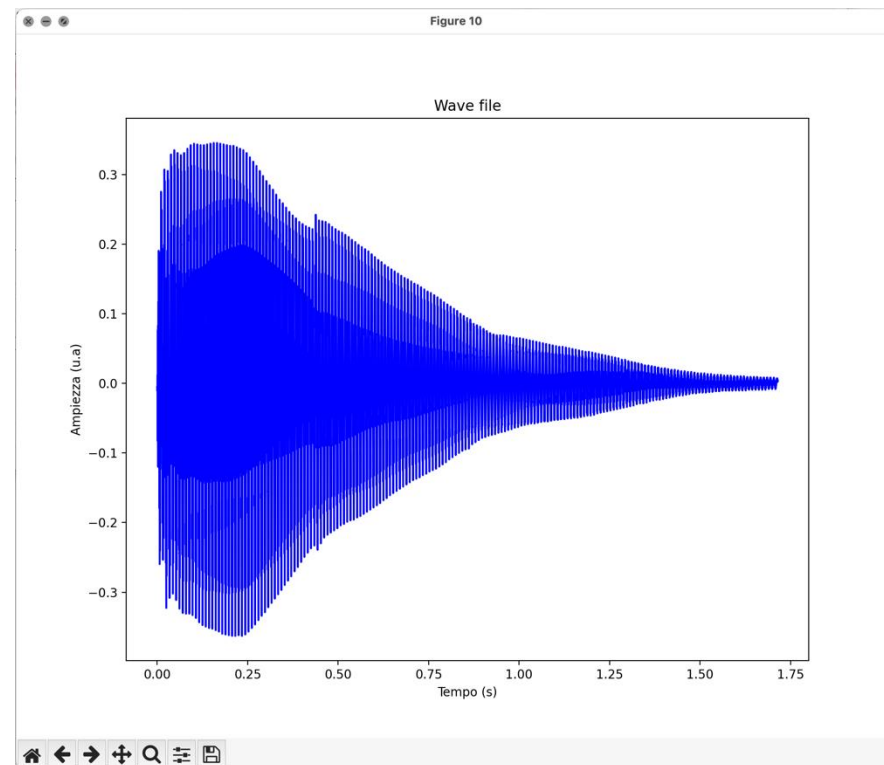
# Esercitazione – parte B

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\\_24-25/slides/diapason.wav](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/diapason.wav)
- [https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio\\_24-25/slides/pulita\\_semplice.wav](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/pulita_semplice.wav)
- [https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio\\_24-25/slides/pulita\\_media.wav](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/pulita_media.wav)
- [https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio\\_24-25/slides/pulita\\_difficile.wav](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/pulita_difficile.wav)
- [https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio\\_24-25/slides/distorta.wav](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/distorta.wav)



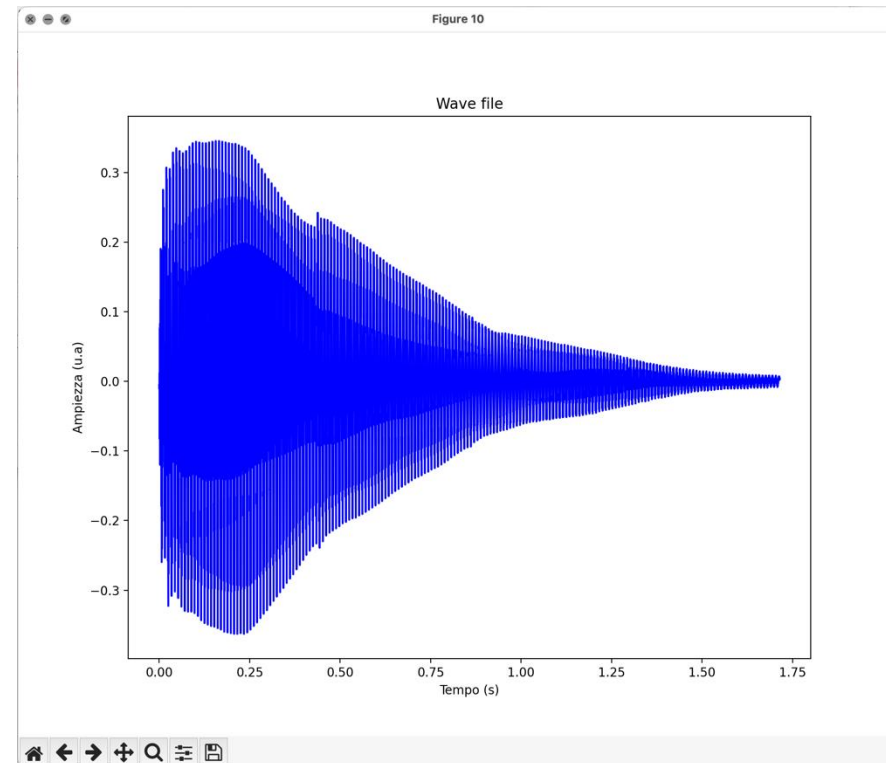
# Esercitazione – parte B

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\\_24-25/slides/pulita\\_pezzo.wav](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/pulita_pezzo.wav)
- [https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio\\_24-25/slides/distorta\\_pezzo.wav](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/distorta_pezzo.wav)



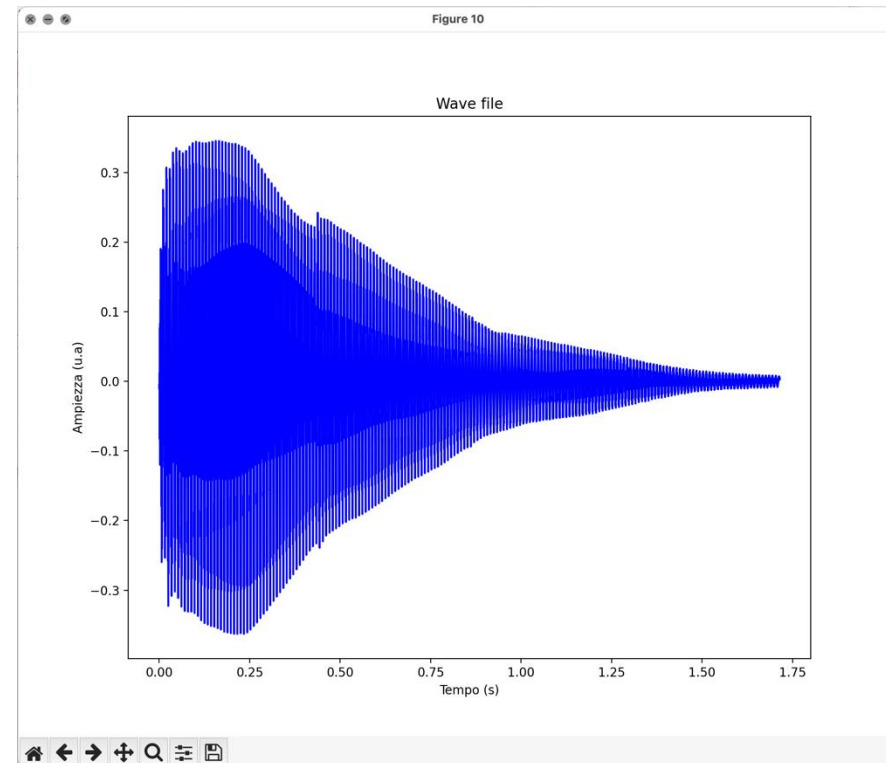
# Esercitazione – parte B

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:

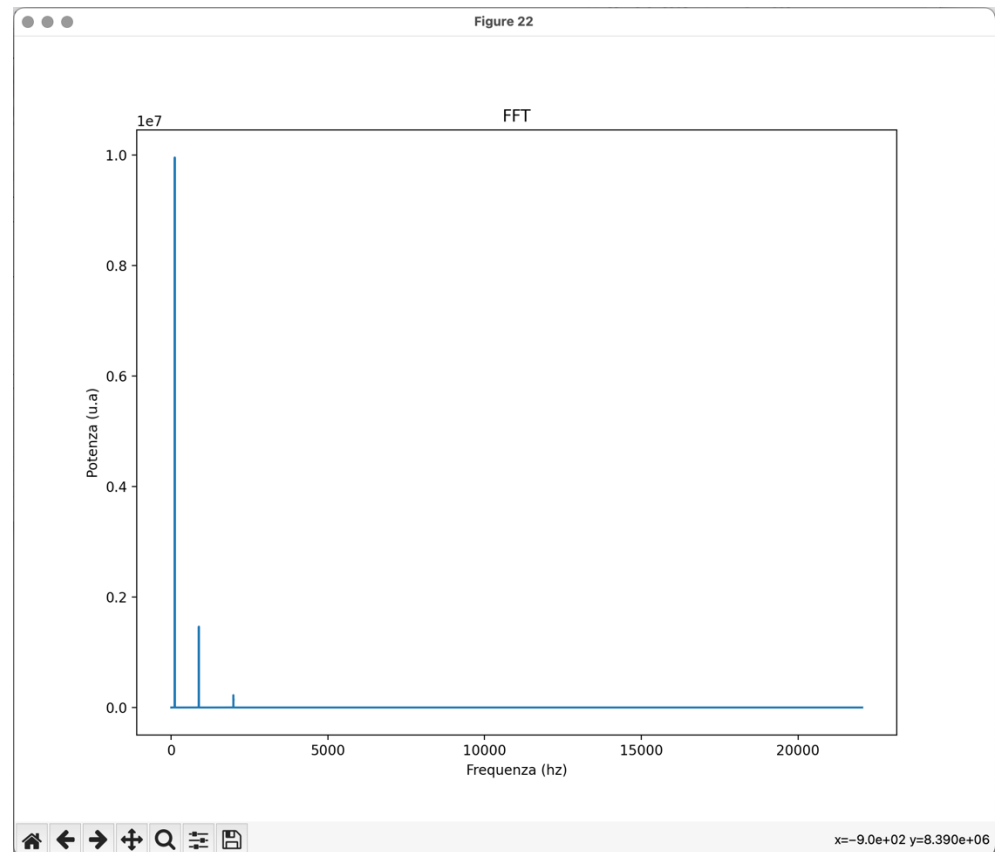
- [https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio\\_24-25/slides/primo.wav](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/primo.wav)
- [https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio\\_24-25/slides/secondo.wav](https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_24-25/slides/secondo.wav)



# Esercitazione – parte B

realizzare un piccolo programma python:

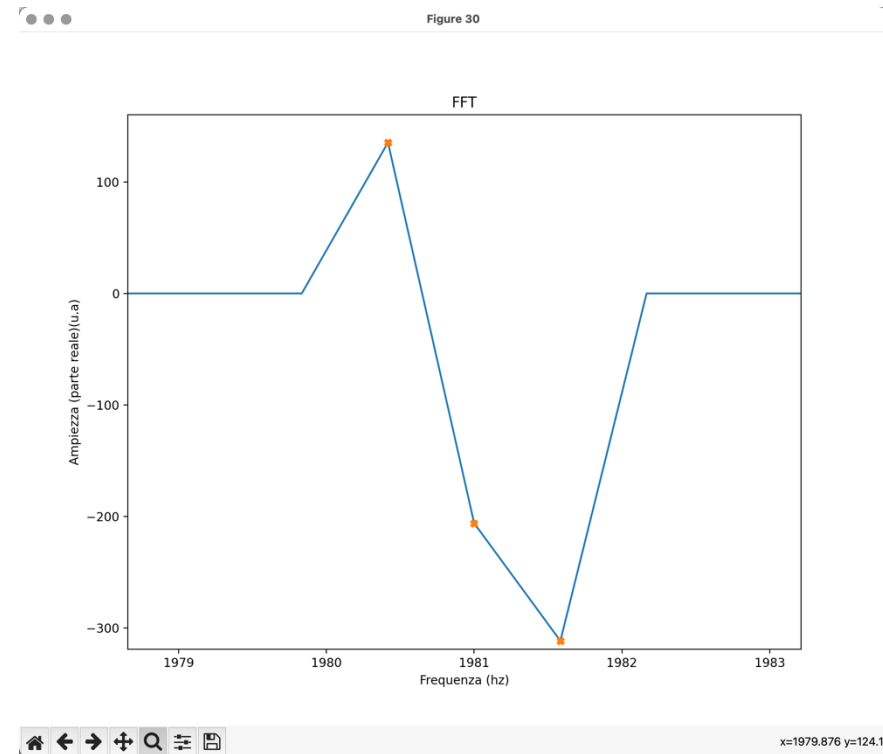
- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, parte reale e parte immaginaria dei coefficienti
  - usando la libreria FFT di python



# Esercitazione – parte B

realizzare un piccolo programma python:

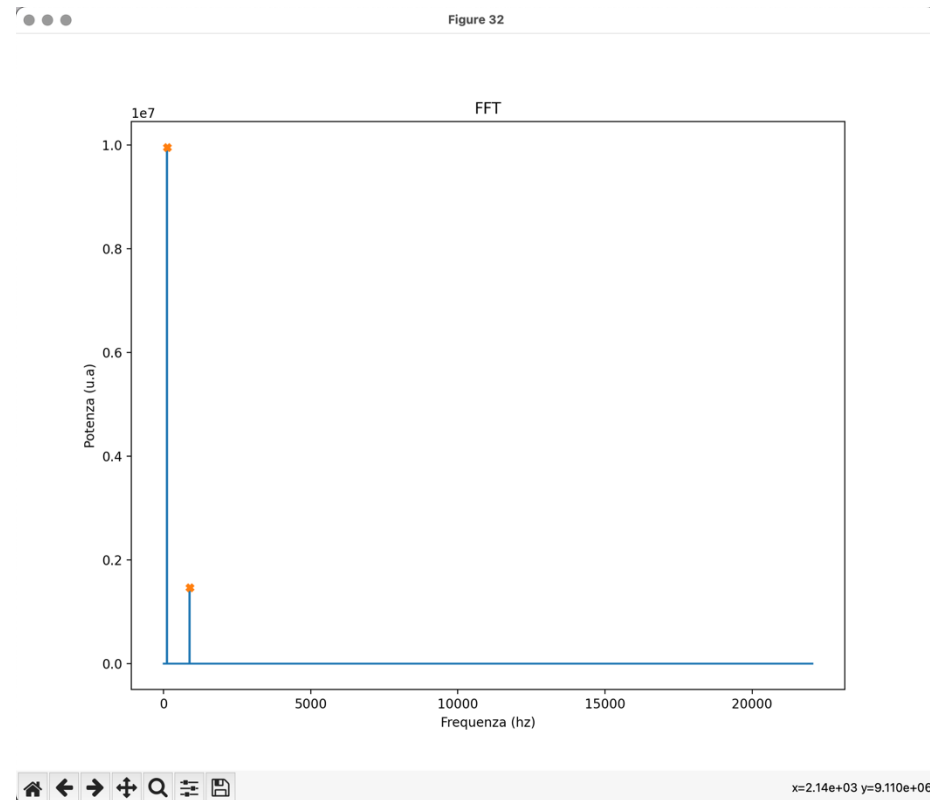
- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, parte reale e parte immaginaria dei coefficienti
- (solo parte 1) identificare i "picchi"
  - che 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, 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





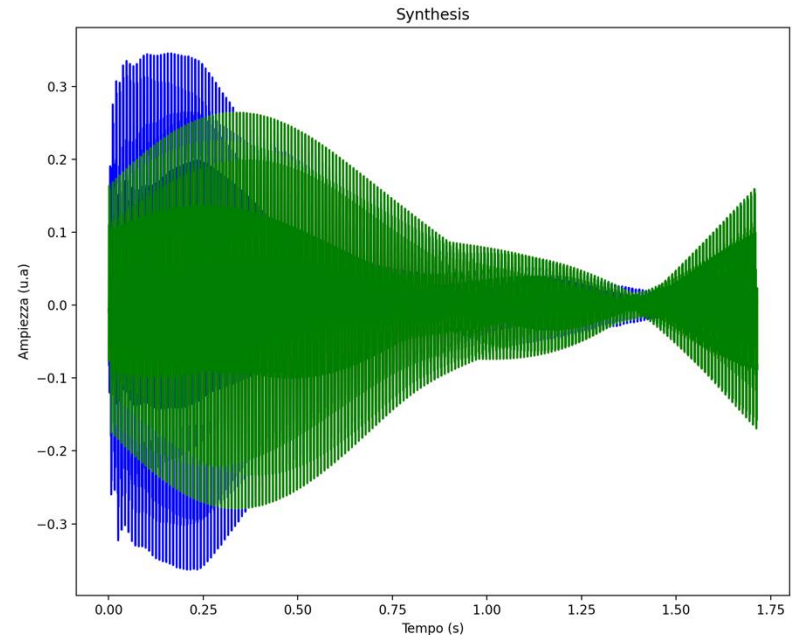
# Esercitazione – parte B

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, 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*)

...

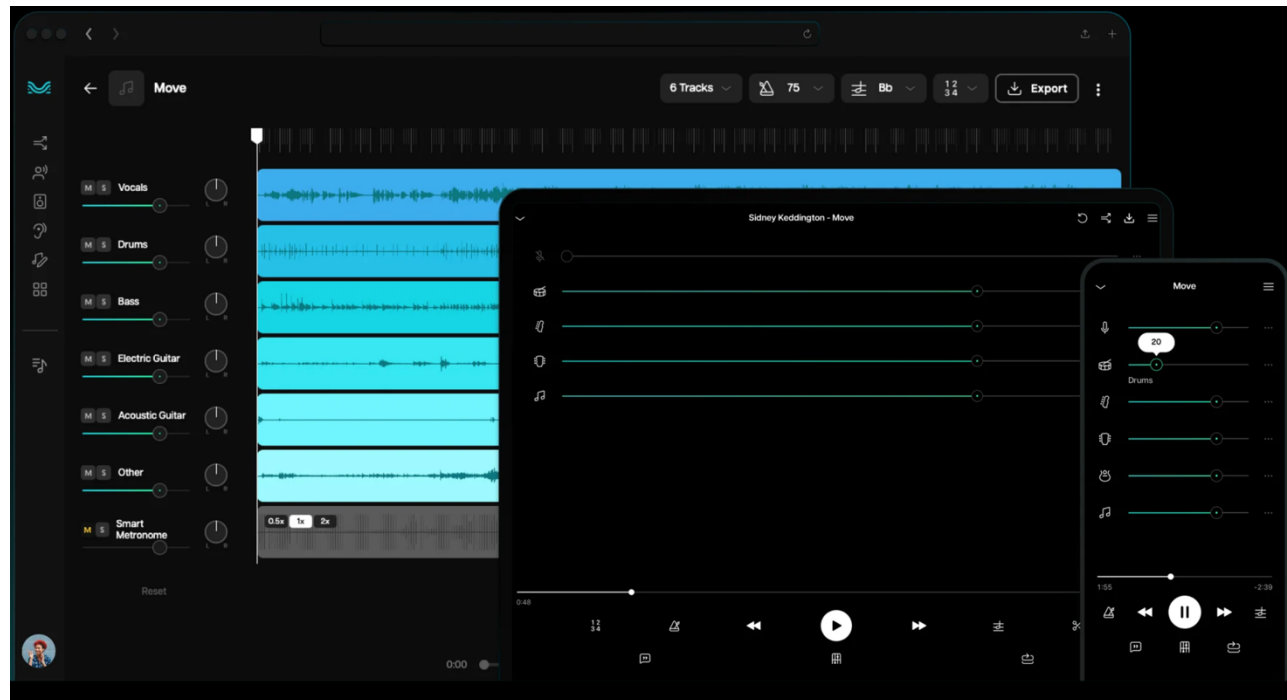
Figure 40



# 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, 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 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](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](https://github.com/s-germani/metodi-computazionali-fisica/blob/main/notebooks/L06%20TrasformateFourier.ipynb)

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