Laboratorio di Elettronica e Tecniche di Acquisizione Dati 2022-2023

Esercitazione 5
"GarageBand"

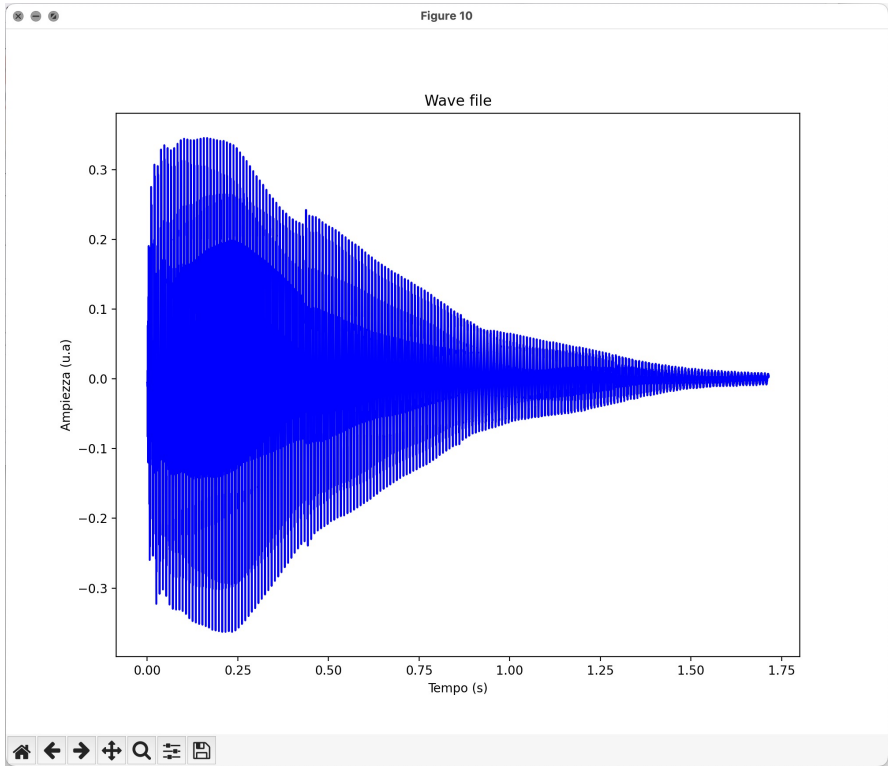
Esercitazione

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

links:

- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_22-23/_slides/diapason.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_22-23/_slides/pulita_semplice.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_22-23/_slides/pulita_media.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_22-23/_slides/pulita_difficile.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_22-23/_slides/pulita_pezzo.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_22-23/_slides/distorta.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_22-23/_slides/distorta_pezzo.wav



Lettura/Scrittura file audio

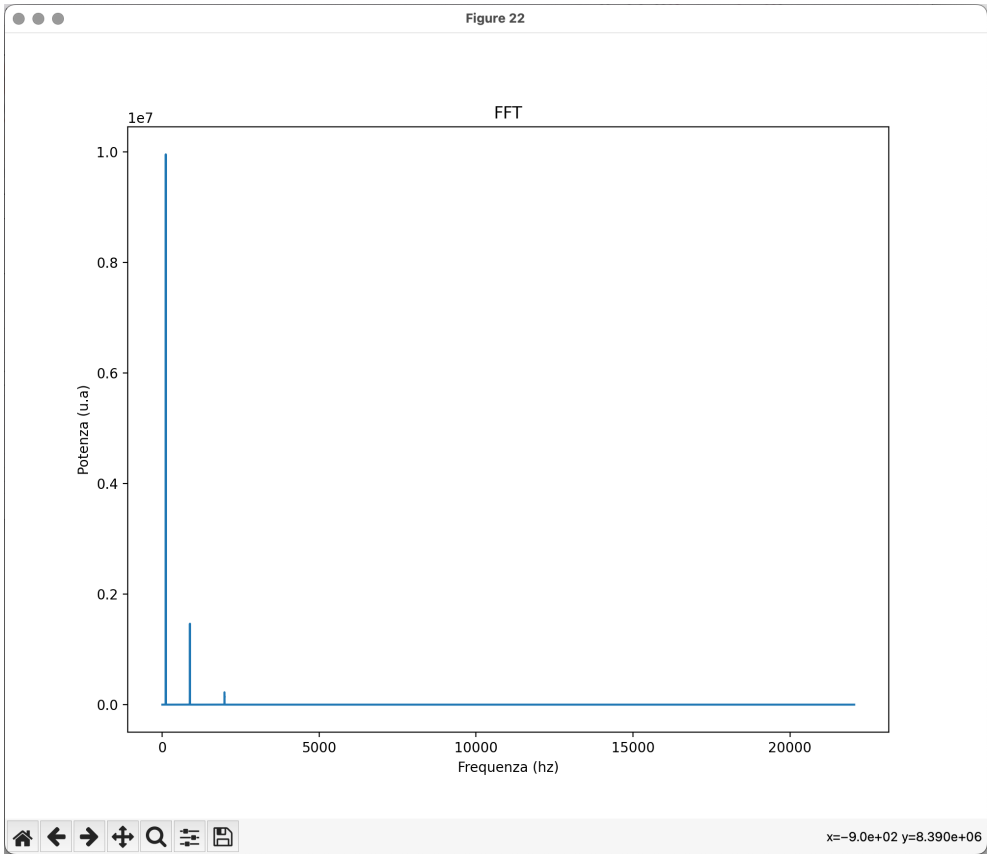
Fonte: https://pysoundfile.readthedocs.io/en/latest/

```
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)
```

Esercitazione

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



FFT

Fonte: https://docs.scipy.org/doc/scipy/tutorial/fft.html

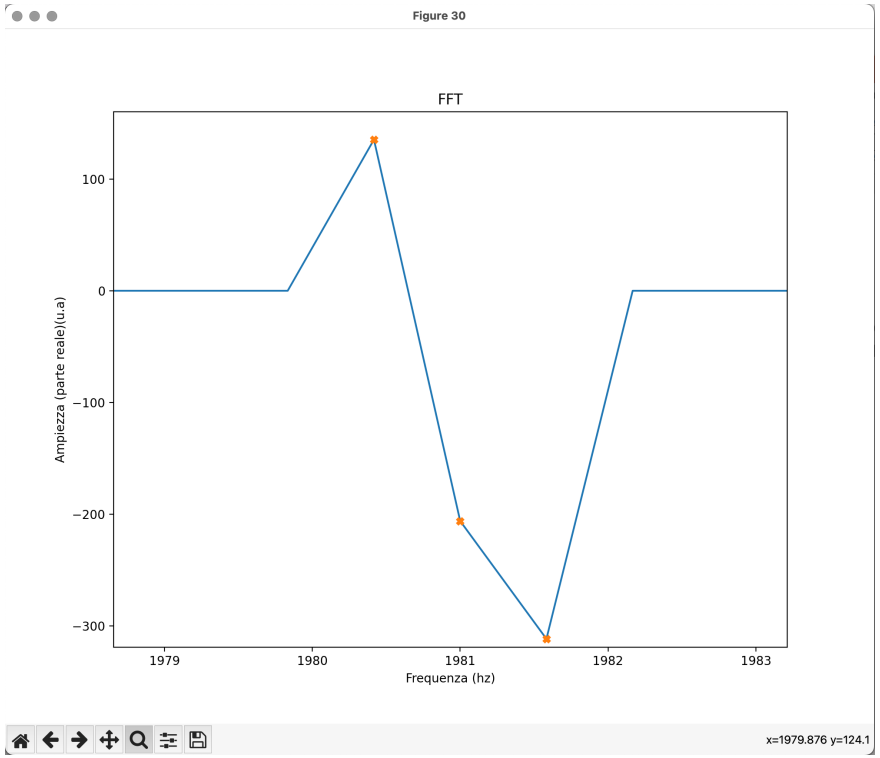
https://github.com/s-germani/metodi-computazionali-fisica/blob/main/notebooks/L06_TrasformateFourier.ipynb

```
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)
```

Esercitazione

realizzare un piccolo programma python:

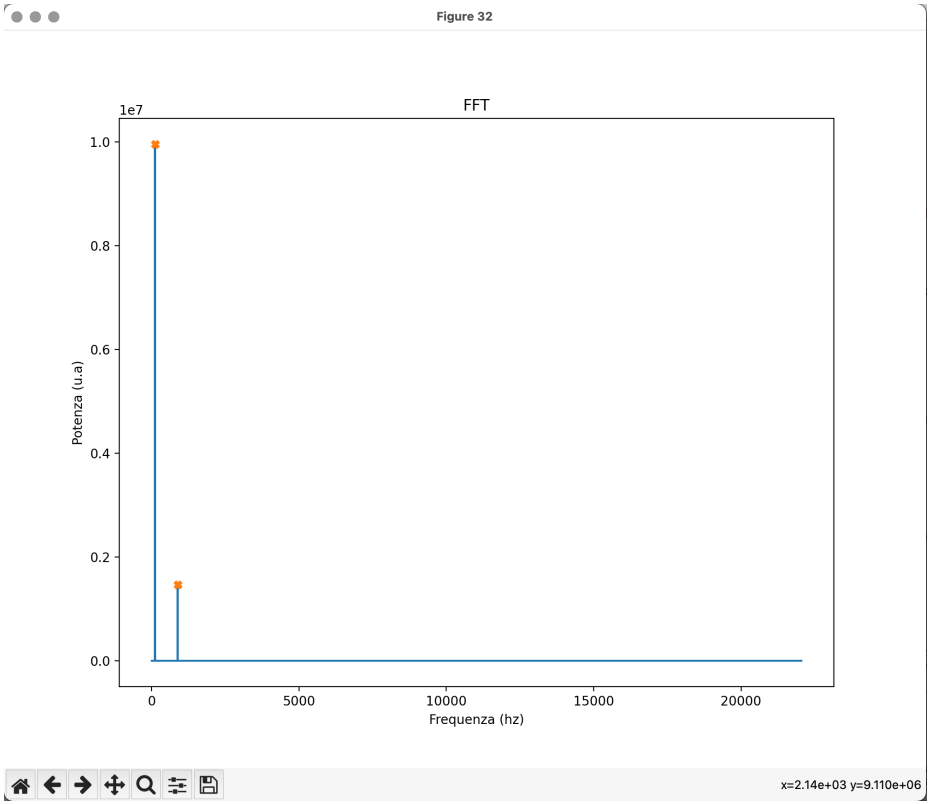
- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, parte reale e parte immaginaria dei coefficienti
- identificare i "picchi"
  - che nota è?
   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



Esercitazione

realizzare un piccolo programma python:

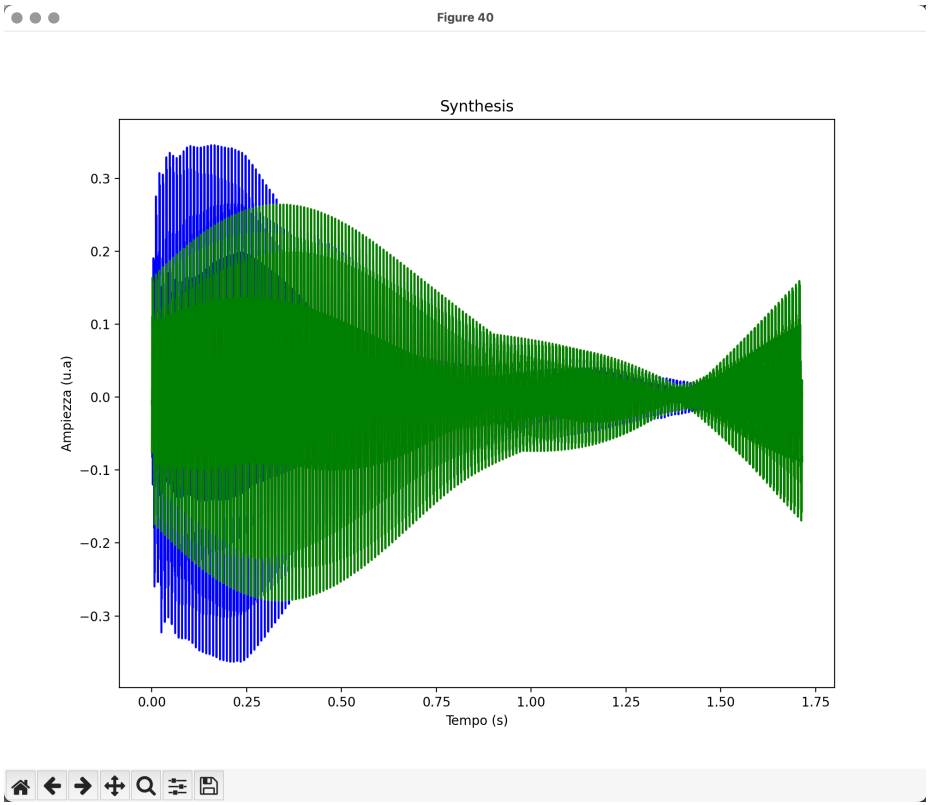
- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, parte reale e parte immaginaria dei coefficienti
- identificare i "picchi"
- mascherare (i.e. mettere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

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"
- "sintetizzare" l'array di dati ("filtrati") e produrre un file audio (.wav)
  - utilizzando la libreria FFT di python
  - [iper-facoltativo] utilizzando seni e coseni (*np.sin* e *np.cos*)



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))
```