

Laboratorio di Elettronica e Tecniche di Acquisizione Dati 2023-2024

Esercitazione 5 "GarageBand"

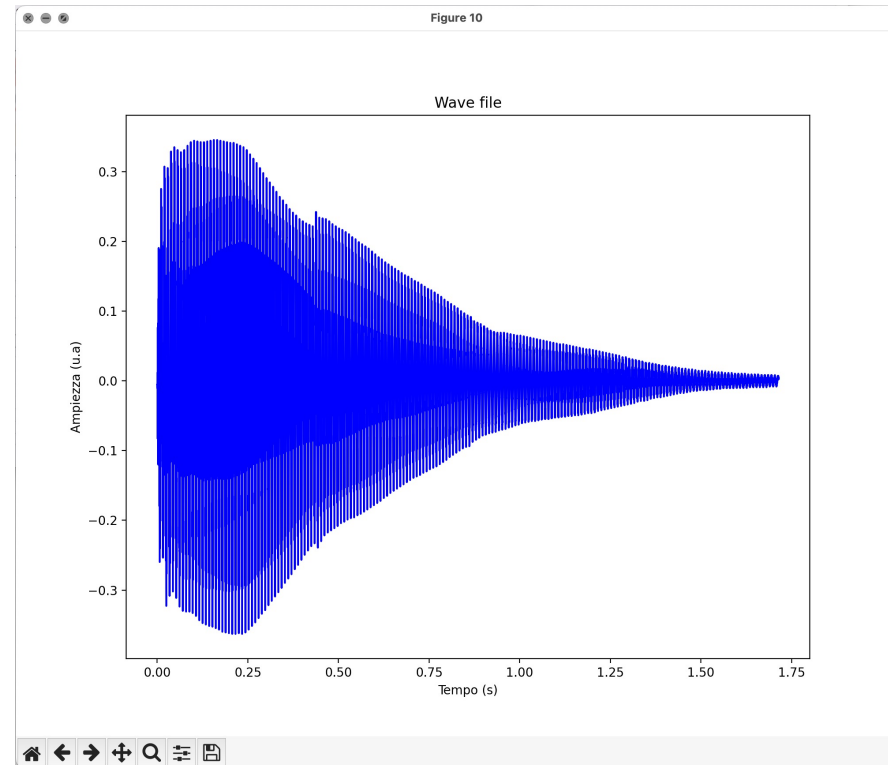
Esercitazione

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_23-24/_slides/diapason.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_23-24/_slides/pulita_semplice.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_23-24/_slides/pulita_media.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_23-24/_slides/pulita_difficile.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_23-24/_slides/pulita_pezzo.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_23-24/_slides/distorta.wav
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_23-24/_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')

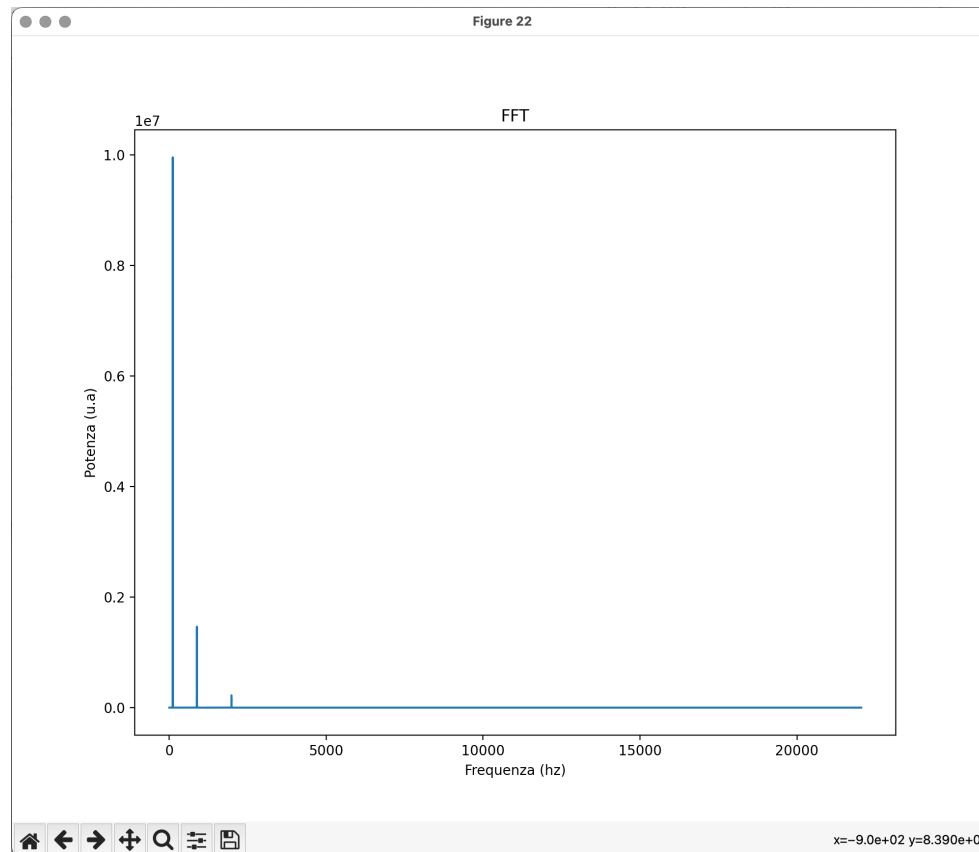
print(samplerate)
print(data)
print(len(data))

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)

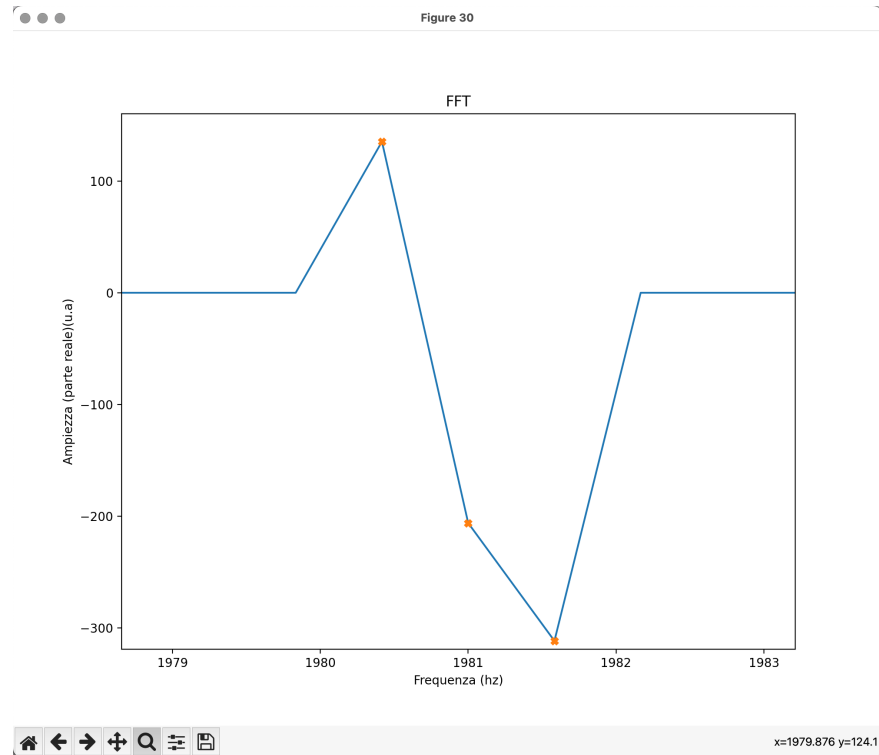
print(datafft)
print(len(datafft))
print(fftfreq)
print(len(fftfreq))

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:

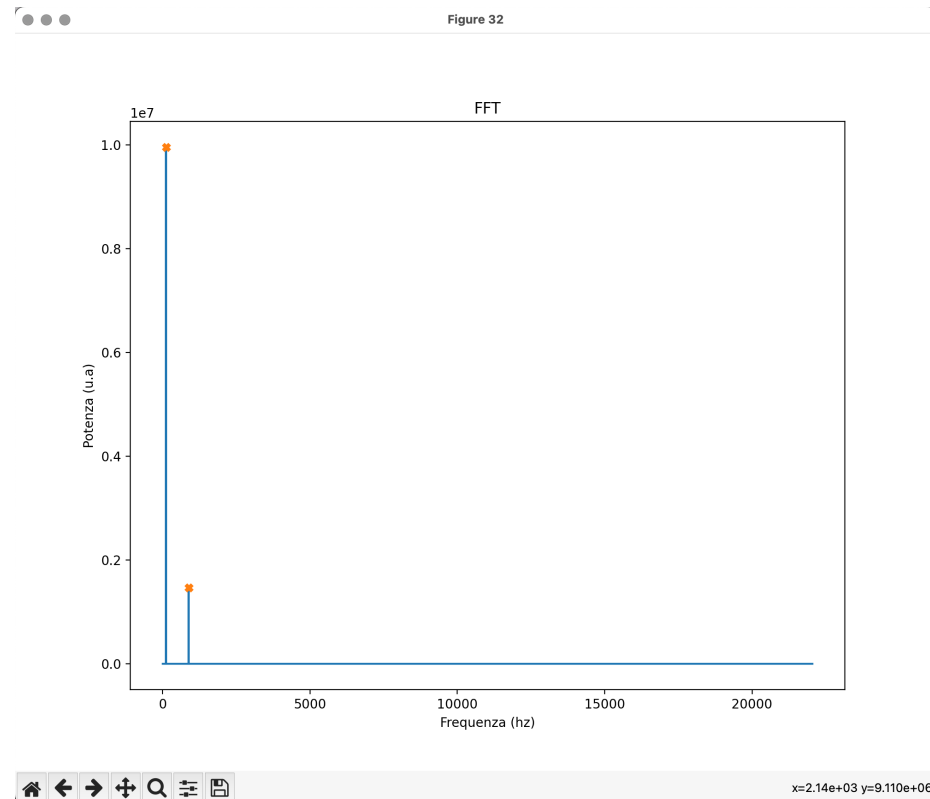
- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, parte reale e parte immaginaria dei coefficienti
- **identificare i "picchi"**
 - che nota è?
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

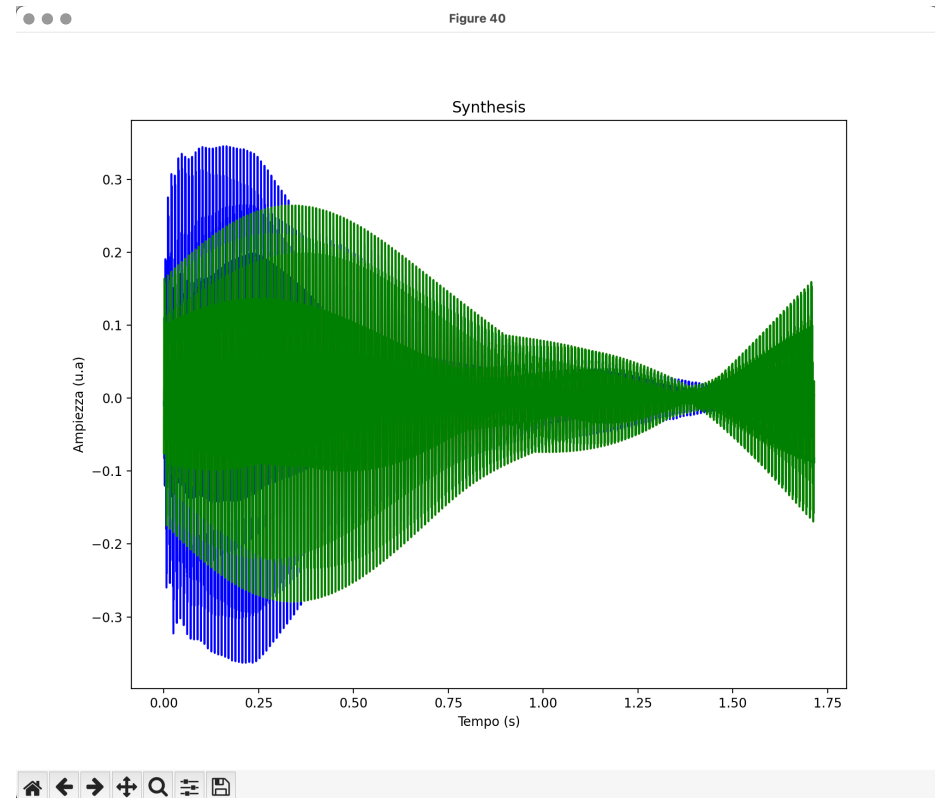
- per aprire un piccolo file audio (.wav) e plottarne la waveform
- utilizzare l'array ottenuto dal file per creare un nuovo file audio (.wav), uguale al primo
- fare la FFT dell'array e plottare: potenza, parte reale e parte immaginaria dei coefficienti
- identificare i "picchi"
- mascherare (i.e. mettere 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
 - 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](https://github.com/s-germani/metodi-computazionali-fisica/blob/main/notebooks/L06%20TrasformateFourier.ipynb)

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

Esercitazione

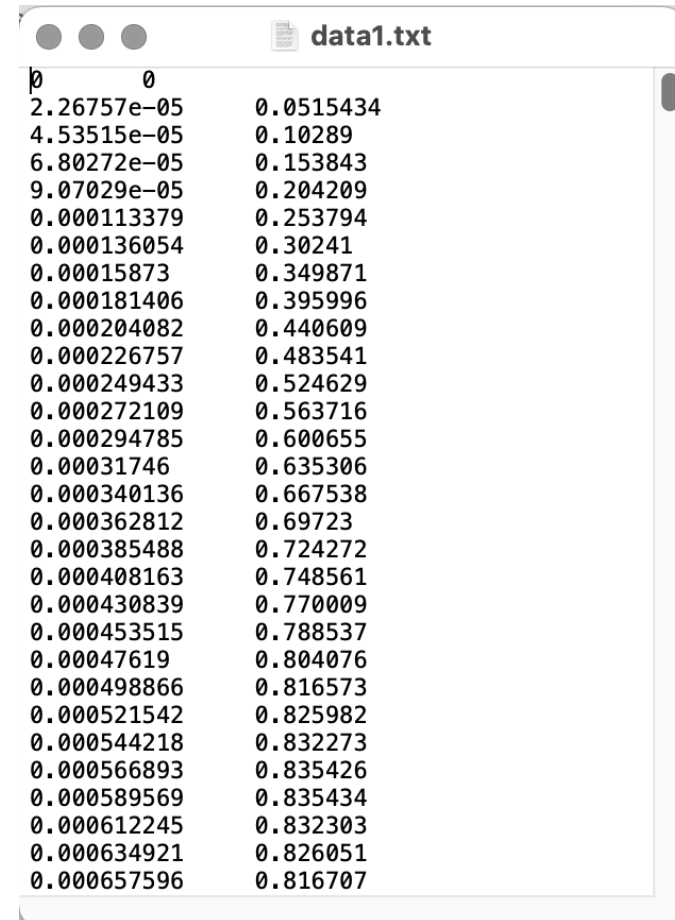
realizzare un piccolo programma python:

- per aprire un file di testo e plottarne la waveform
- studiare in frequenza il segnale
 - che tipo di segnale è quello filtrato?
- ri-sintetizzare il segnale a partire da quello in frequenza
- filtrare (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.

Links:

- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_23-24/_slides/data1.txt
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_23-24/_slides/data2.txt
- https://www.fisgeo.unipg.it/~duranti/laboratoriodue/laboratorio_23-24/_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
```