PROJECT REPORT

In this project we used the "Spyder" tool provided from "Anaconda" platform to implement our project.

MILESTONE1 <<The pianist>>

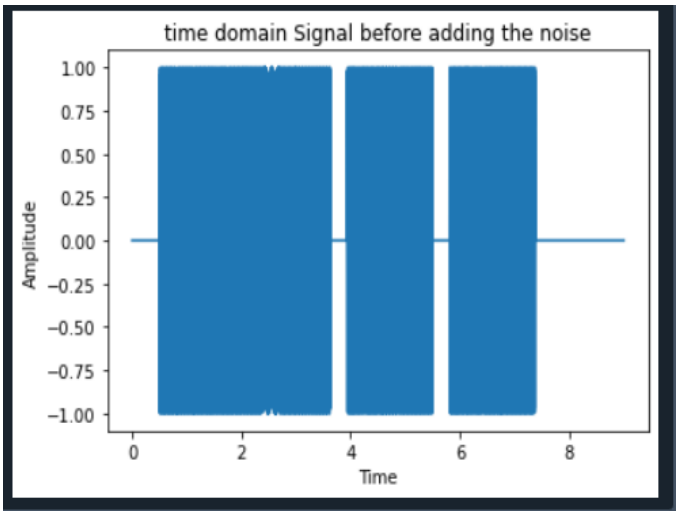
- **We chose a suitable song (fur elisie – Beethoven).**
- **Split the song into piano notes with the 3rd octave for the left hand and 4th for the right one.**
- **Determined each note's corresponding frequency and duration and stored them in 2 arrays, one for each hand (octave).**
- **For silence periods we set the frequency to be zero.**
- **When using one hand at a specific time duration, the other hand frequency in the same duration (time interval) to be zero.**
- **Determined the duration of each note and its corresponding frequency using the personal evaluation (musical sense).**
- **Initiated an empty song X.**
- **Using loops, we managed to derive the final output song as a sum of adding partial nots using the relation:**

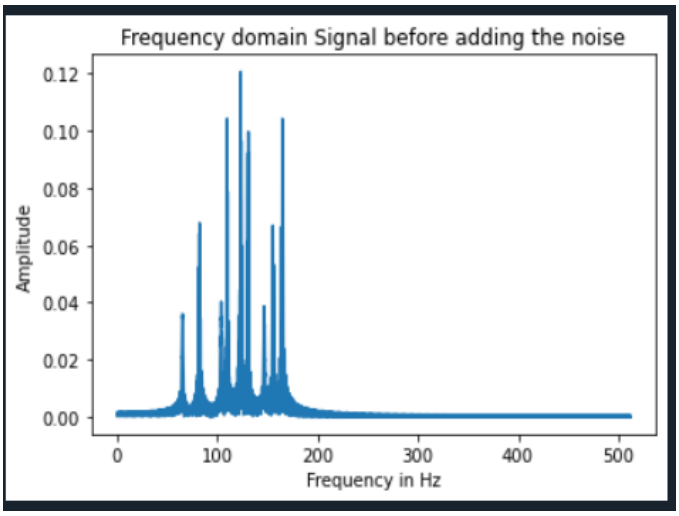
$$x(t) = \sum_{i=0}^{N} [\sin 2\pi \mathrm{Fi}\, t + \sin 2\pi \mathrm{fi}\, t\,][u(t - ti) - u(t - ti - Ti)]$$

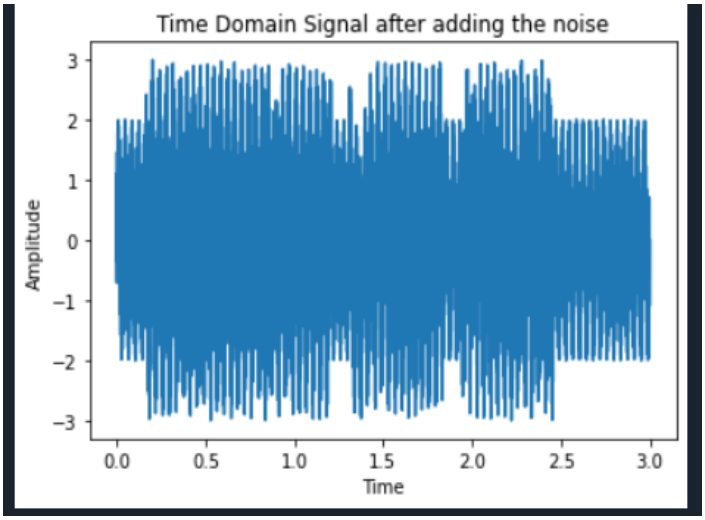
- **We plotted the signal using function plot from matplotlib.**
- **We played the sound of the output song using play from library sounddevice.**

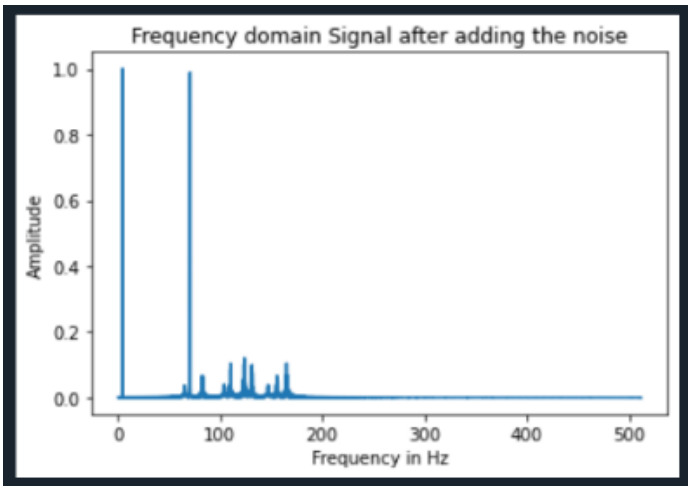
MILESTONE2<<Noise cancelation>>

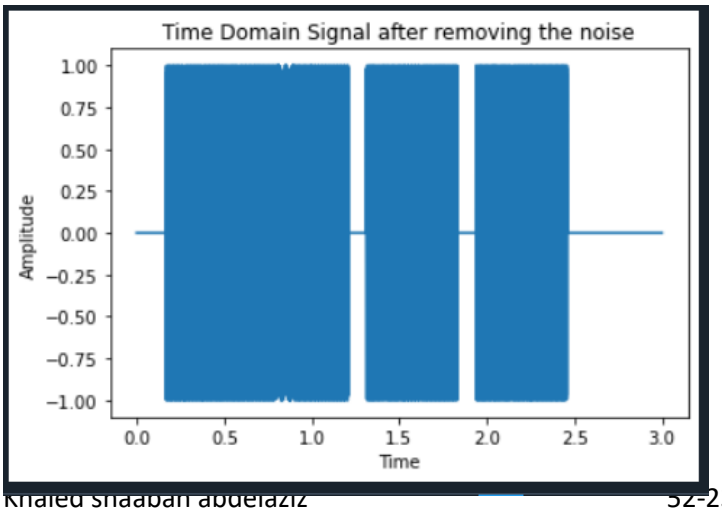
- **We converted the plot of the signal form the time domain(fig1) to the frequency domain(fig2) using the previously imported function "fft" from the library "SciPi" with Nyquist Rate equal to 1024.**
- **We generated a two random numbers in order to use them as a frequency of the noise source which was calculated form the relation: $n(t)$ = sin ($2fn1\pi t$) + sin ($2fn2\pi$t).**
- **We added the noise to the output song signal obtained in the first milestone to get the new time domain signal output (fig3) and frequency domain (fig4).**
- **After comparing the sounds and the plots of the output before and after adding the noise we conserved that the noise has an effect of the plots and the sound.**
- **We then cancel the noise form the output by removing it from the equation and obtaining a new song signal filtered which we called Xfiltered.**
- **The output of the plot when using the Xfiltered song was similar to the first output before adding the noise from the perspective of the sound played and the plots either in the time domain (fig5) or the frequency domain (fig6).**

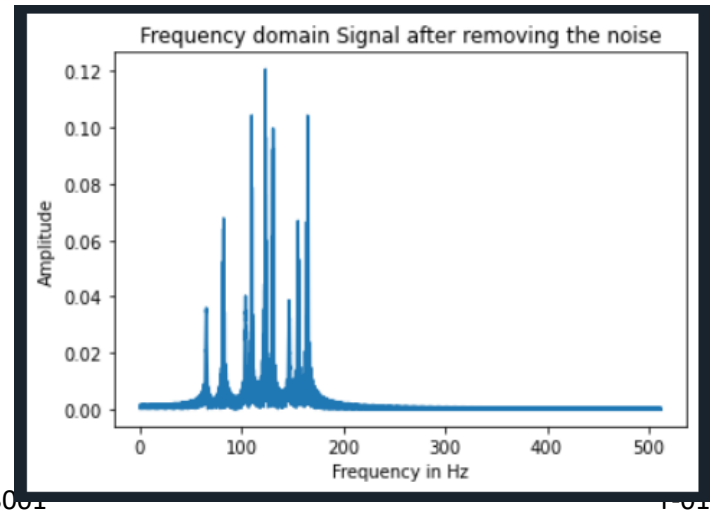

time domain Signal before adding the noise
Amplitude
Time


Frequency domain Signal before adding the noise
Amplitude
Frequency in Hz


Time Domain Signal after adding the noise
Amplitude
Time


Frequency domain Signal after adding the noise
Amplitude
Frequency in Hz


Time Domain Signal after removing the noise
Amplitude
Time


Frequency domain Signal after removing the noise
Amplitude
Frequency in Hz