Milestone One Report

In this milestone, we have implemented a part of the twinkle, twinkle little star song using the NumPy, matplotlib, and soundDevice libraries using Anaconda's Spyder.

Firstly, we started by importing the three libraries as follows: import numpy as np, import matplotlib.pyplot as plt, and import sounddevice as sd.

Then, we set the time t to be from 0 to 3 seconds with a sample space of 12 * 1024 as follows: t = np.linspace(0, 3, 12 * 1024).

Thirdly, we set the 3rd Octave (Left hand) frequencies Fi and the 4th Octave (Right Hand) frequencies fi with the frequencies of the 1st 20 notes of the song as follows Fi = np.array([164.81, 164.81, …]) and fi = np.array([261.63, 261.63, …]).

After that, we initial time to zero like that ti = 0.

Then we set the duration of each press of each note as an array of 20 elements as follows: Ti = np.array([0.2, 0.2, …]).

Then we initialize the signal with and initial value zero: signal = 0;

After that, we loop through fi, Fi, and Ti using a for loop starting from 0 to 20 exclusive. With each loop we setup the unit step function that we will use to cut the signal for each frequency. Then we add the

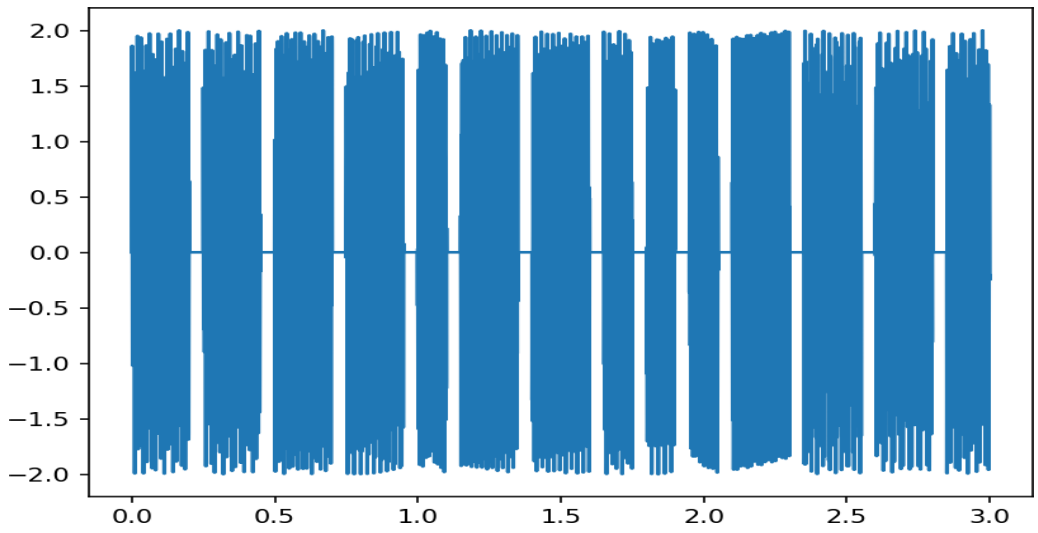
sinusoidal signals with each new frequency and multiply it by the unit step function, then we increment the initial time ti by the duration Ti plus 0.05. All of this is done as follows:

```
for i in range (0, 20):
    u = [t >= ti] * [t <= ti + Ti]
    signal = (np.sin(2*np.pi*Fi*t) + np.sin(2*np.pi*fi*t)) * u
    ti += Ti + 0.05
```

After the summation process is done, we plot the graph of the signal and show it as follows: plt.plot(t, signal) and plt.show().

At the end once the graph is shown, we play the song generated by the signal as follows: sd.play(signal, 3 * 1024)

Below is a picture of the output of the plt.plot(t, signal).



The reason why we cannot see the waves of the sinusoidal function is that the high frequencies.

Report prepared by Ziad Elsabbagh.