

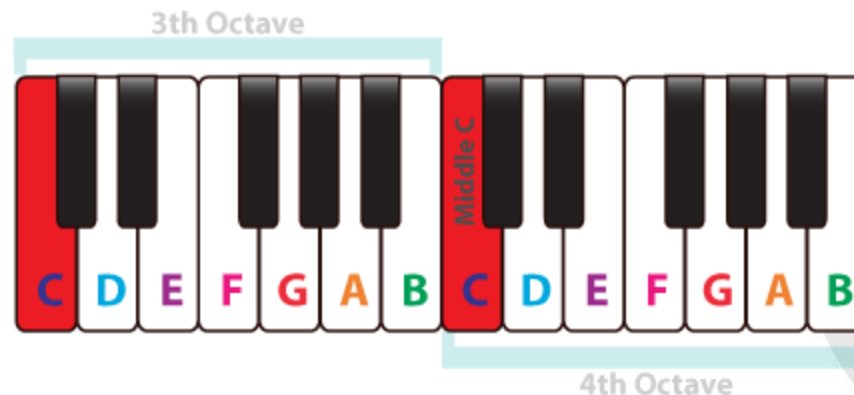


Faculty of Information Engineering and Technology
Department of Communication Engineering

Prof. Dr. Ahmed El-Mahdy

Signals and Systems Theory (COMM401)

Lab Project (PIANO)-Milestone 2 Noise Cancelation





- This project is classified into 2 milestones:

Milestone 1: You are the **PIANIST**, you should mix and match frequencies to generate your own song.




Milestone 2: Noise Cancellation and Frequency Domain





Project Guidelines

- Groups of **THREE** students MAX (From the same Lab group, NO cross labs allowed)
- Copied reports or code means **zero** for both versions ! 
- At the end of both milestones you should submit:
Code: Your Python script/s
Report: A report containing a brief description of what you have implemented followed by the output figure/s obtained

Submissions will be submitted to your TA



Background

- Single tone generation:

Each pressed piano key/note number i , corresponds to a single tone with frequency f_i generated for a certain period of time T_i starting from time t_i over a defined time range t .

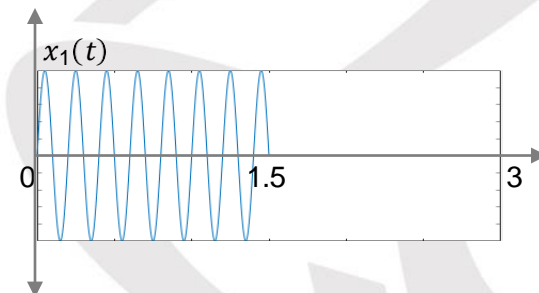
$$x_i(t) = \sin(2\pi f_i t) [u(t - t_i) - u(t - t_i - T_i)]$$

Where the unit step functions defines the playing interval.

Example: Assume that we started pressing the **middle C** key ($f_1 = 261.63 \text{ Hz}$), at time $t_1 = 0$ for a duration of $T_1 = 1.5$ seconds, the resultant signal is:



$$x_1(t) = \sin(2 * 261.63\pi t) [u(t) - u(t - 0.5)]$$



Milestone 1: The Pianist

- Signal/Song Generation:

Generate your own signal using N pairs of notes chosen from the 3rd and 4th piano octaves assuming that you are playing with both hands at the same time (3rd octave with the left hand/ 4th octave with right hand),

Each piano key/note frequency is given below. Your final song could be generated as follows,

$$x(t) = \sum_{i=1}^N [\sin(2\pi F_i t) + \sin(2\pi f_i t)] [u(t - t_i) - u(t - t_i - T_i)]$$

Chosen from the 3rd octave

Chosen from the 4th octave

Note	Frequency	Note	Frequency
C3	130.81	C4	261.63
D3	146.83	D4	293.66
E3	164.81	E4	329.63
F3	174.61	F4	349.23
G3	196	G4	392
A3	220	A4	440
B3	246.93	B4	493.88

3th Octave



4th Octave

Milestone 1: The Pianist

- Procedure:

1) You will need to import the following libraries.

```
import numpy as np
import matplotlib.pyplot as plt
import sounddevice as sd
```



2) Set the total song playing time to 3 seconds starting from 0 for 12×1024 samples.

```
t = np.linspace(0 , 3 , 12 * 1024)
```

3) Customize your own song by setting the number of pairs of notes N . For each pair number i , set the left hand frequency F_i , right hand frequency f_i , the pressing starting time t_i , and how long you will press both keys T_i .

N.B you can freely customize these values to play the song you want.

If you want to play with one hand only, you can set the frequency of the other hand to 0.



Milestone 1: The Pianist

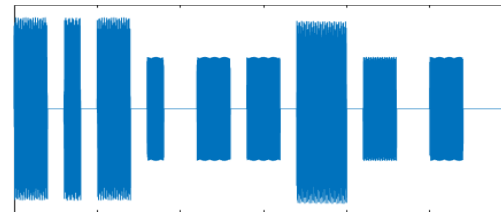
- Procedure:

4) Define your signal/song using the previous parameters:

$$x(t) = \sum_{i=1}^N [\sin(2\pi F_i t) + \sin(2\pi f_i t)] [u(t - t_i) - u(t - t_i - T_i)]$$

5) Plot your signal/song in the time domain your output will look similar to

- the following: `plt.plot(t, x)`



NB. The time separation between the notes will vary from one group to another according to your song. We also can't notice the variations in the sinusoidal tones due to the high frequencies.

6) **Finally:** Play your song using:

`sd.play(x, 3 * 1024)`

Background

- **Noise generation:**

Assume that additional noise is generated by a child who kept pressing the same 2 random piano keys during your song playing interval such that the noise is represented as follows,

$$n(t) = \sin(2f_{n_1}\pi t) + \sin(2f_{n_2}\pi t)$$



Where, f_{n_1} and f_{n_2} are two frequencies selected randomly from the 3rd and 4th octaves frequencies list respectively. Your final song after the noise becomes,

$$x_n(t) = x(t) + n(t)$$

Goal: Noise Cancellation: using frequency domain conversion



Milestone 2: Frequency Conversion

- Procedure:

0) You will need to add this line at the beginning of your code:

```
from scipy.fftpack import fft
```

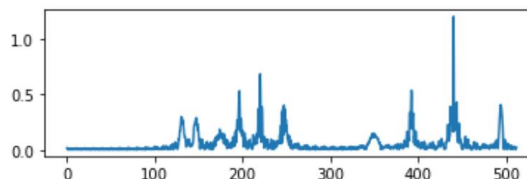
1) Set the number of samples N to *your_song_duration* x 1024. Then set the frequency axis range to,

```
N = 3x1024  
f = np.linspace(0, 512, int(N/2))
```

2) Convert the time signal $x(t)$ to the frequency signal $X(f)$ from,

```
x_f = fft(x)  
x_f = 2/N * np.abs(x_f[0:np.int(N/2)])
```

Your final output will look as following,



Milestone 2: Noise Generation

- Procedure:

- 3) Generate the noise signal as described in slide 3 where the two random frequencies are selected as follows,

$$f_{n_1} \& f_{n_2} = \text{np.random.randint}(0, 512, 2)$$

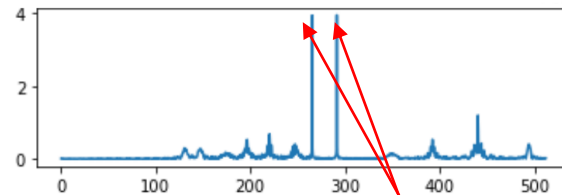
Hint: Each time you run the code you will get different noise.

- 4) Add the generated noise $n(t)$ to the signal $x(t)$,

$$x_n(t) = x(t) + n(t)$$

Hint: Play the new signal to notice the noise effect on the sound.

- 5) Convert the noise contaminated signal $x_n(t)$ to the signal $X_n(f)$ in frequency domain, it looks as follows,



Hint: You will notice that the signal contains **two very high frequency peaks**

Milestone 2: Noise Cancelation

- Procedure:

6) Find the two random noise frequencies: by finding the frequency indices that corresponds to the peaks of the signal $X_n(f)$ which is higher than the maximum peak of the original signal $X(f)$ without noise rounded to the next integer (to avoid including the maximum value due to double precision errors).

7) Round the frequency values at that indices in f to get \hat{f}_{n_1} and \hat{f}_{n_2} (since we generate random integer values of frequencies)

8) Filter the noise by subtracting two tones with the two found frequencies

$$x_{filtered}(t) = x_n(t) - [\sin(2\hat{f}_{n_1}\pi t) + \sin(2\hat{f}_{n_2}\pi t)]$$

9) Play the filtered signal to make sure that you get the same original sound without noise



`sd.play(xfiltered, 4 * 1024)`

Project Submission

- Copied reports or code means **zero** for both versions !

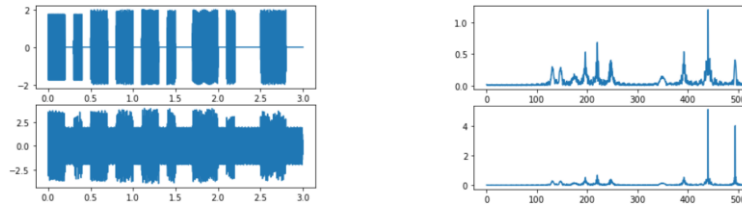


- Submit a HARD COPY report that consists of:

Code(s): Your Python script/s of both of milestone 1 and 2 together

Figures: All figures plotted should be printed

Report (Word Typed): Update your report with a brief description of the carried out steps followed by the output figure/s obtained (Hence: You should have six plots divided into two figures, the first figure contains the time domain signal with and without noise. And the second figure contains the frequency domain signal with and without noise.)



- Deadline for submitting the report to your TA is Tuesday 14/5/2024
- Evaluations will be held in your corresponding Lab Slot on the week starting 14/5/2024 till 20/5/2024
- Attendance of ALL team members together is a must
- Late Reports will be deducted 50% of the report grade



GOOD LUCK

