

# SIGNAL PROCESSING AND COMMUNICATION

MCTE 4338

*Mini Project 2*

---

Students Names:

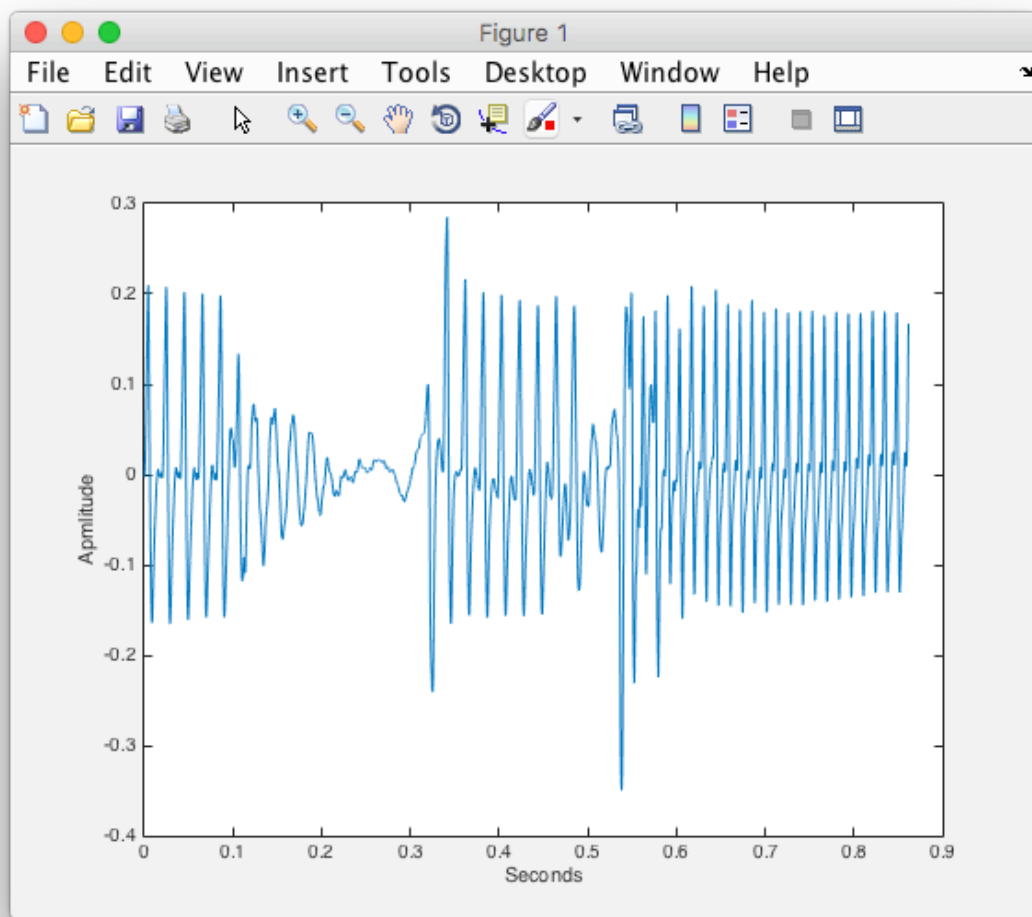
1. Saleem Hadad (1337479)
  2. Ahed AlQaud (1410721)
- 

## Question 1

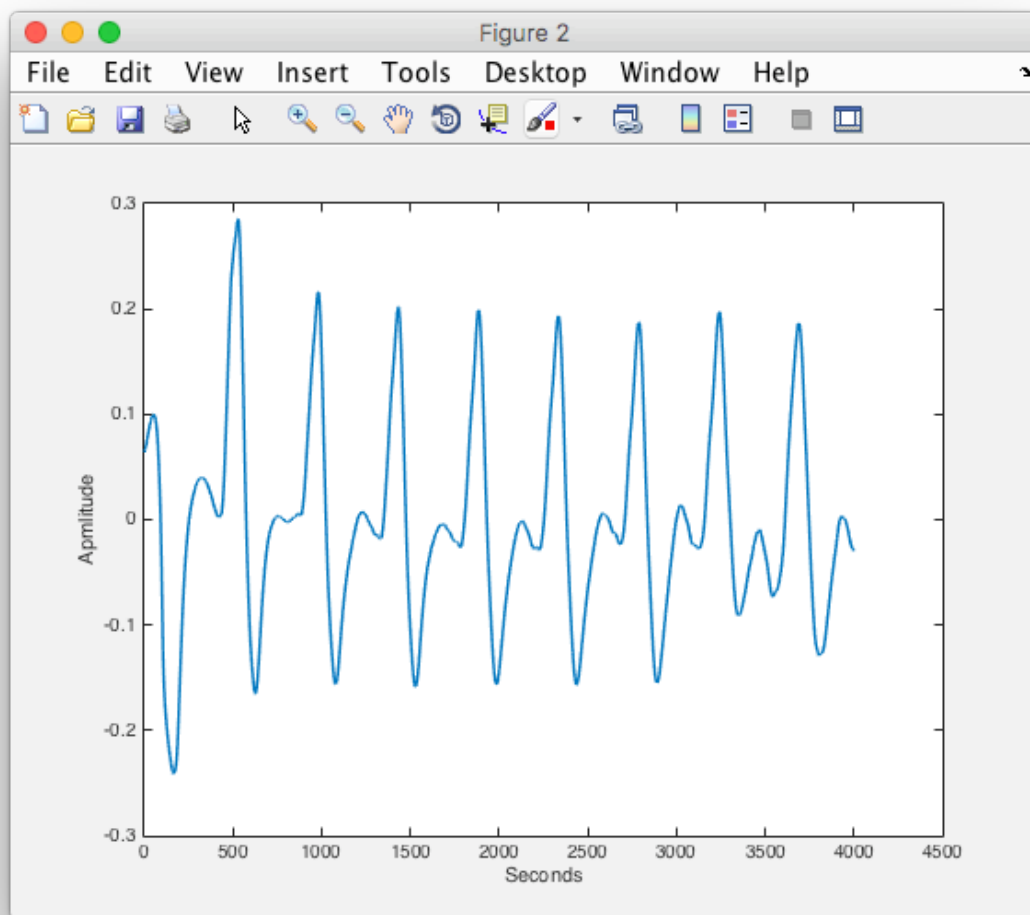
Load 'bass.wav' using `[x,fs]=audioread('bass.wav')`. Obtain and plot the signal for a segment of between `x(670000:689000)`. You should be able to distinguish about 2 notes in the segments. Plot and determine the frequency of each notes (you should process the signal separately).

## Solution

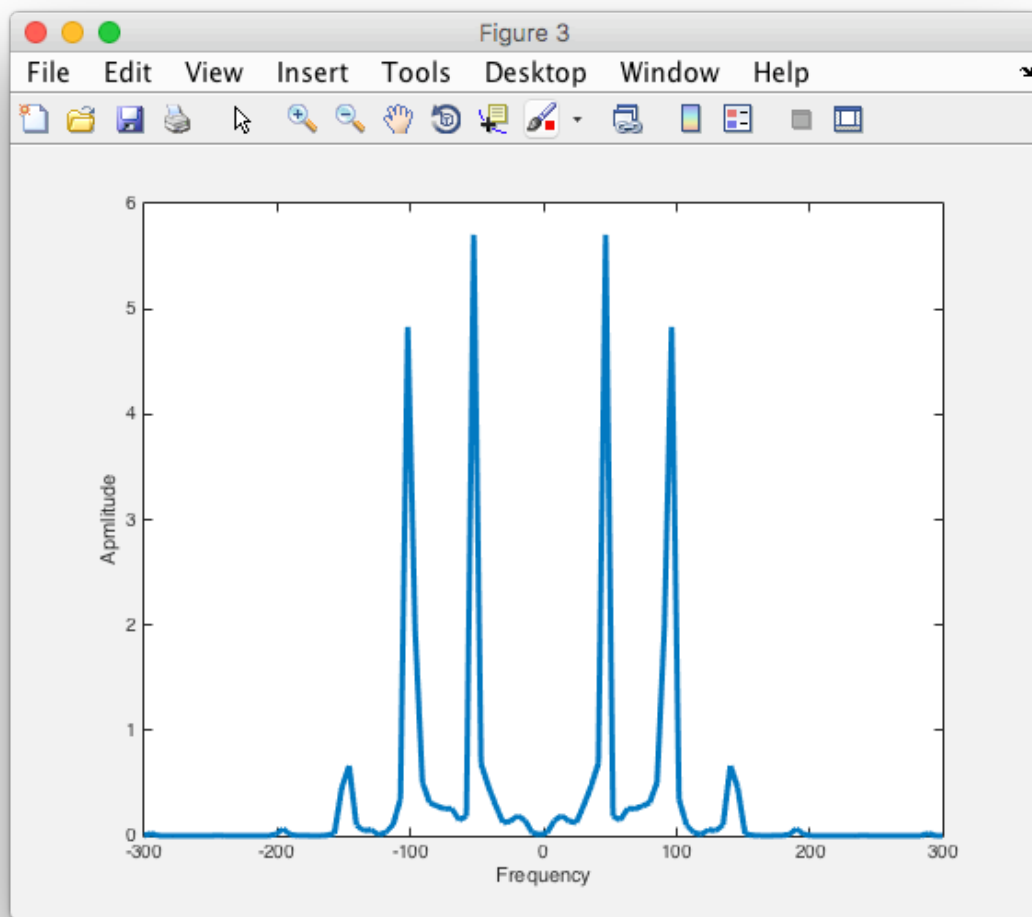
```
1 % base audio
2 [xn, fs]=audioread('audio.wav');
3 x1 = xn(670000:689000);
4 dt = 1/fs;
5 t=0:dt:(length(x1)*dt)-dt;
6 plot(t, x1)
7 xlabel('Seconds');
8 ylabel('Amplitude');
```



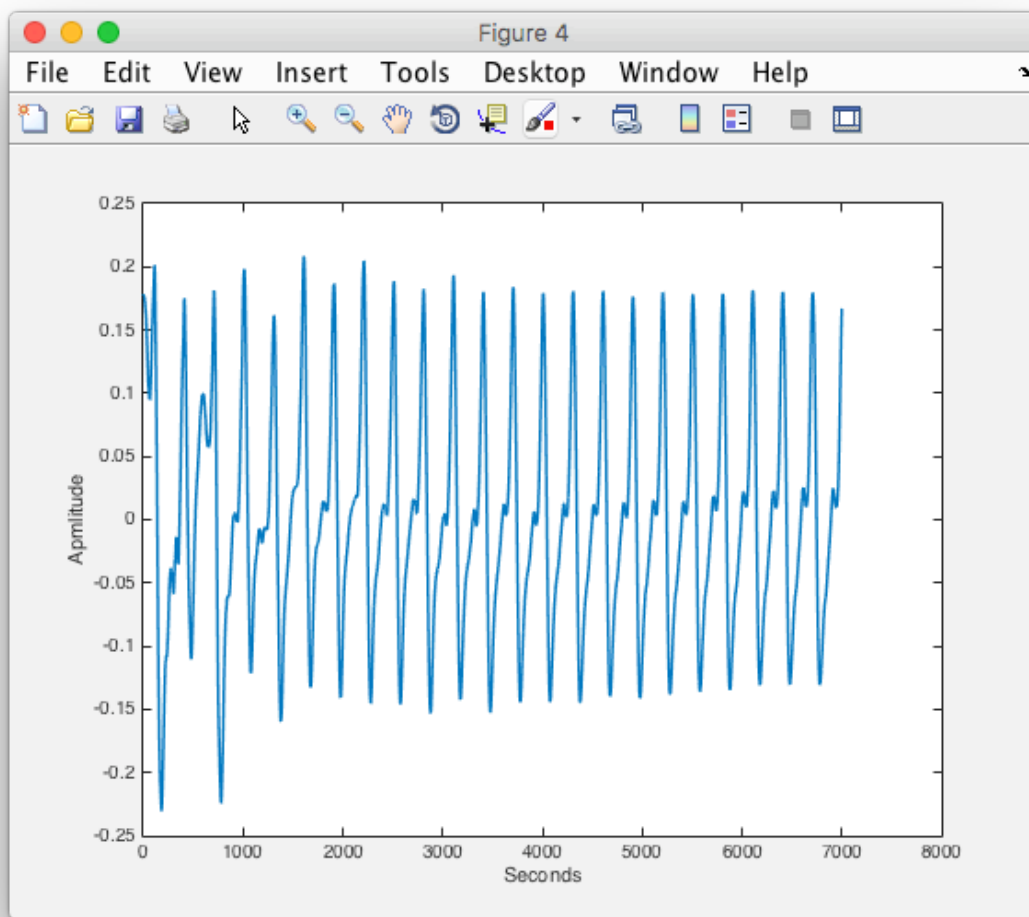
```
1 % note 1
2 figure;
3 N1 = xn(677000:681000);
4 t=1:1:length(N1);
5 plot(t, N1, 'linewidth', 1.5);
6 xlabel('Seconds');
7 ylabel('Apmlitude');
```



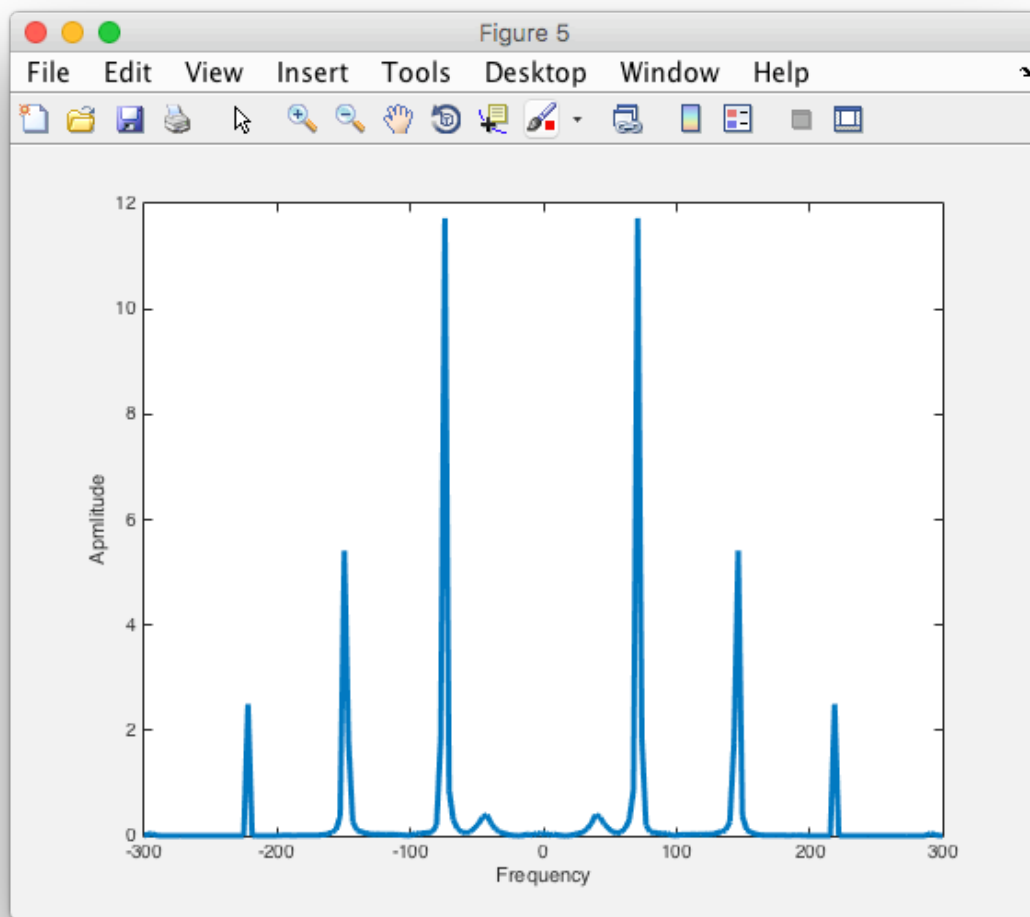
```
1 % note 1: spectrum analysis
2 figure;
3 N_1 = length(N1);
4 x_1 = fftshift(fft(N1));
5 fshift = (-N_1/2:N_1/2-1) * (fs/N_1);
6 powershift=abs(x_1).^2/N_1;
7 plot(fshift, powershift,'linewidth',3)
8 xlim([-300, 300]);
9 xlabel('Frequency');
10 ylabel('Apmlitude');
```



```
1 % note 2
2 figure;
3 N2 = xn(682000:689000);
4 t=1:1:length(N2);
5 plot(t, N2, 'linewidth', 1.5);
6 xlabel('Seconds');
7 ylabel('Amplitude');
```



```
1 % note 2: spectrum analysis
2 figure;
3 N_2 = length(N2);
4 x_2 = fftshift(fft(N2));
5 fshift = (-N_2/2:N_2/2-1) * (fs/N_2);
6 powershift=abs(x_2).^2/N_2;
7 plot(fshift, powershift,'linewidth',3)
8 xlim([-300, 300]);
9 xlabel('Frequency');
10 ylabel('Apmlitude');
```



## Question 2

Considering there are 10 workers in the office. You are required to design a biometric security system that will only allow the 10 workers to enter the office. Write a code that could recognize who is speaking using the method of Euclidean distance of their speech magnitude spectrum.

## Dataset

We have collected 6 (the last column used for testing) audio files for 5 persons as follow:

Person	audio 1	audio 2	audio 3	audio 4	audio 5	audio 6
Saleem	✓	✓	✓	✓	✓	✓
Tariq	✓	✓	✓	✓	✓	✓
Khaled	✓	✓	✓	✓	✓	✓
Muhammed	✓	✓	✓	✓	✓	✓
Google	✓	✓	✓	✓	✓	✓

Python language used to do this project.

## Code

```
1  # import useful librarys and tools
2  import numpy as np
3  from scipy.fftpack import fft
4  import matplotlib.pyplot as plt
5  from scipy.io import wavfile as wav
6
7  NTTF = 1000
8
9  # stored names in the dataset.
10 names = [
11     'saleem',
12     'tariq',
13     'khaled',
14     'muhammed',
15     'google'
16 ]
17
18 # calculate the FFT for a given audio file name.
19 def calc_fft(audio):
20     # read and process the audio file
21     fs, data = wav.read(audio)
22     fft_out = fft(data)
23     k = np.arange(len(data))
24     T = len(data)/fs
25     frqLabel = k/T
26     fft_out = fft_out[:len(fft_out)//2-1]
27     frqLabel= frqLabel[:len(frqLabel)//2-1]
28
29     # plot the spectrum
30     plt.plot(frqLabel, np.abs(fft_out))
31     return np.abs(fft_out)
32
33 # helper method used to calculate Euclidean distance between two variables.
34 def calc_dist(s1, s2):
35     return np.linalg.norm(s1 - s2)
36
37 # used to compute the average FFT real values.
38 def calc_avg_fft(s):
39     return np.sum(s/len(s))
40
41 # used to calcualte the average of all resulted fft for a given person name.
42 def avg_fft(name):
43     avg_ffts = []
44     plt.figure(name)
45     for i in range(5):
46         avg_ffts.append(
```

```

47         calc_avg_fft(
48             calc_fft('./dataset/' + name + str(i+1) + '.wav')
49         )
50     )
51     return np.average(avg_ffts)
52
53     # normalize output result
54     def normalized(a, axis=-1, order=2):
55         l2 = np.atleast_1d(np.linalg.norm(a, order, axis))
56         l2[l2==0] = 1
57         return a / np.expand_dims(l2, axis)
58
59     # calculate the percentage of match between each unknown audio file and known
    dataset.
60     def process_unknowns(known):
61         n = len(known)
62         result = []
63         for i in range(n):
64             dists = []
65             avg_fft = calc_avg_fft(calc_fft('./dataset/unknown' + str(i+1) +
    '.wav'))
66             for j in range(n):
67                 dists.append(calc_dist(known[j], avg_fft))
68             result.append((1-normalized(dists))*100)
69         return result
70
71     result = process_unknowns([
72         avg_fft(names[0]),
73         avg_fft(names[1]),
74         avg_fft(names[2]),
75         avg_fft(names[3]),
76         avg_fft(names[4]),
77     ])
78
79     # print results
80     for i in range(len(result)):
81         print("unknown " + str(i+1) + ", matched "),
82         for j in range(len(result[i][0])):
83             print(str(int(result[i][0][j])) + "% with " + names[j] + " "),
84         print
85
86     # show plotted graphs
87     plt.show()

```

## Results

Based on the result we obtain, the performance of the system perform very well in identifying the correct person. However, there are several limitations to this implementation such as the process time and the achieved accuracy, we can improve the speed performance by implementing it on DSP based system in real-time using C or C++ language which is way fast than Python or Matlab. In addition to that, we can increase the dataset size and implement machine learning algorithm with



can beat the implemented handcraft feature extraction.

Unkown	Saleem	Tariq	Khaled	Muhammed	Google
1	97%	68%	81%	95%	7%
2	80%	97%	92%	74%	5%
3	89%	85%	96%	81%	3%
4	91%	81%	93%	95%	3%
5	54%	41%	47%	58%	100%