Reza Jahani	Digital Signal Processing	Date: 28/9/1400
810198377	CA2 Report	Teacher: Dr Badiei

Section 1:Short Fourier Transform & Walvet Transform

I)

The audio file is saved in array ***x*** and the sampling frequency in variable ***fs*** as well. Shown below is the result we accuired with plotting the channels and the sampling frequency is shown.

Two channels are saved in arrays ***x1*** and ***x2*** and are plotted on the diagram. The result can be observed in figure 1.1.

```
>> P1_1
Fs=
        44100
```

Sampling Frequency

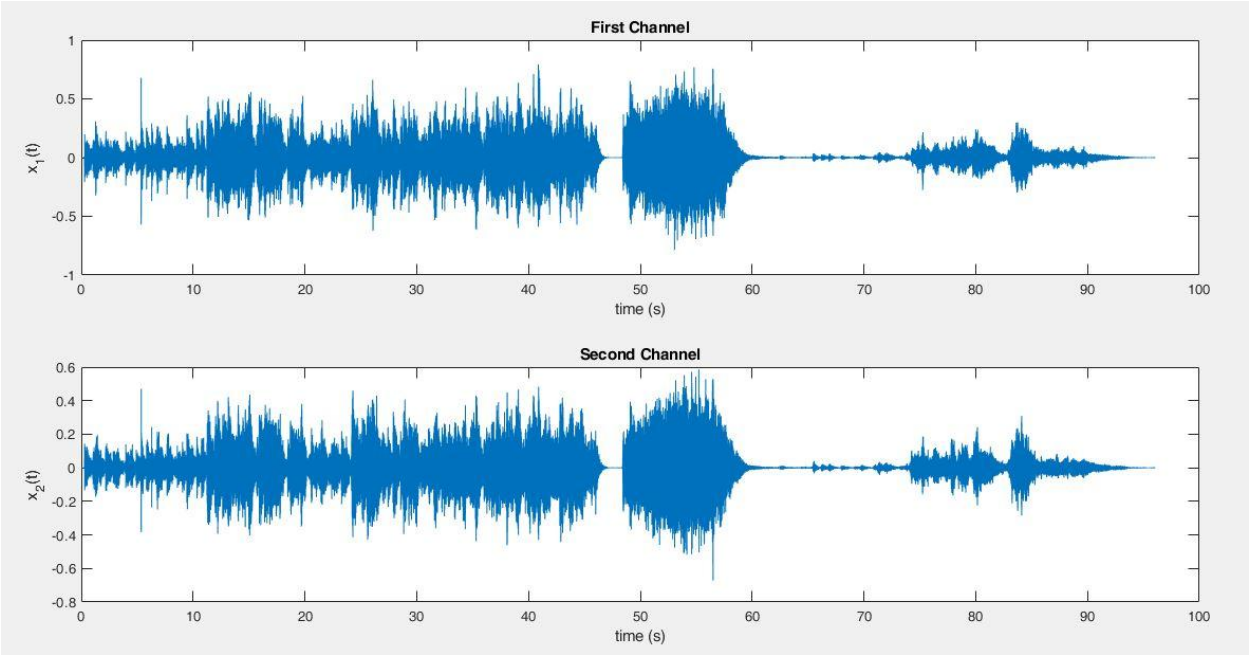

Figure 1.1: Two channels of the array in which audio file is saved

The similarty we can see is the overal form and shape of the signal. On the contrast as a difference to note it is obvious based on the diagram that the amplitude of the signals in 2 channels are different. First channel's signal has a higher amplitude based on what we can observe.

- File used for this part is **P1_1.m**

II)
In this part we are going to compute the fourier transform of the continous-time signals we obtained from the audiofile and compare the results and determine the frequency band. In the end we will also plot the PSD of one of the channels.

The fourier transform of the signals can be observed in figure 1.2.

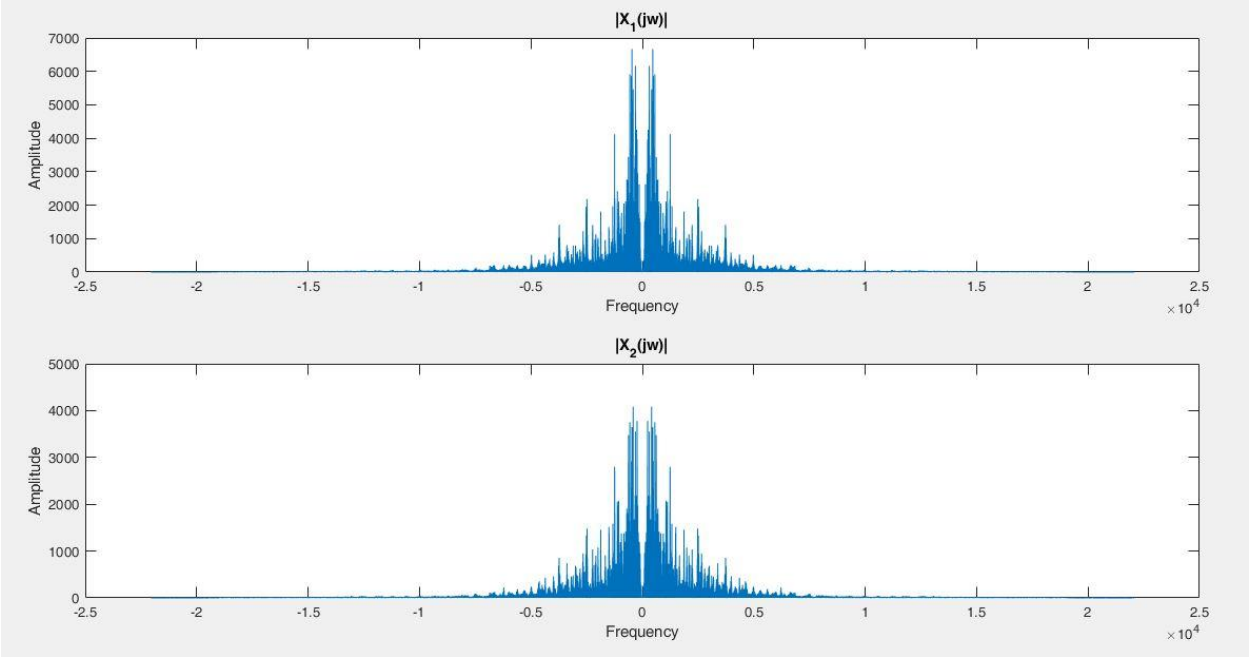

Figure 1.2: Fourier Transform of Channel 1 & Channel 2

Based on the result we gained, the overal shape and form of the fourier transform is the same with a difference in amplitude again. Amplitude of fourier transform of $x_1(t)$ is higher than $X_2(f)$.
Looking at figure 1.2, we can conclude that fourier transform of signals for frequencies which are above 7.5 KHz is zero. Then we can state that frequency band for this signals is

$$F_N = 7.5\ KHz$$

For instance we plot the PSD of channel 1 signal, $x_1(t)$.
As we already know PSD is calculated this way:

$$S_X(f) = \frac{|X_T(f)|^2}{2T}$$

where $X_T(f)$ stands for fourier transform of x(t) cut between the interval [-T,T] and T->∞.

Shown in figure 1.3 is the power spectral density of $x_1(t)$.

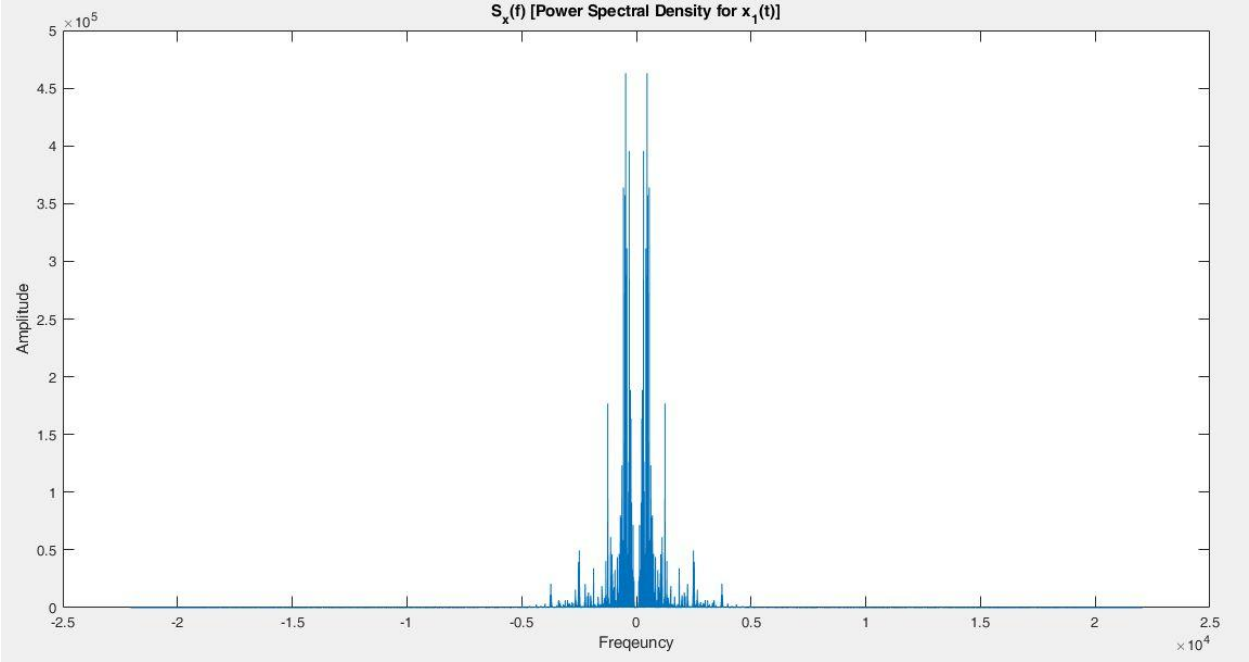

Figure 1.3: PSD of $x_1(t)$

Based on the observations and gained results we can conclude that the main and the stronger frequencies are those frequencies which are below 1KHz.

Main Frequencies: $0.2\ KHz < |f| < 1KHz$

- File used for this part is **P1_2.m**

III)

We choose channel 1.

Shown below in figure 1.4 is the spectrogram of the signal $x_1(t)$. Pictures of both spectrogram and the signal is brought to compare what is going on about the frequencies in the signal and the conclusions are stated.

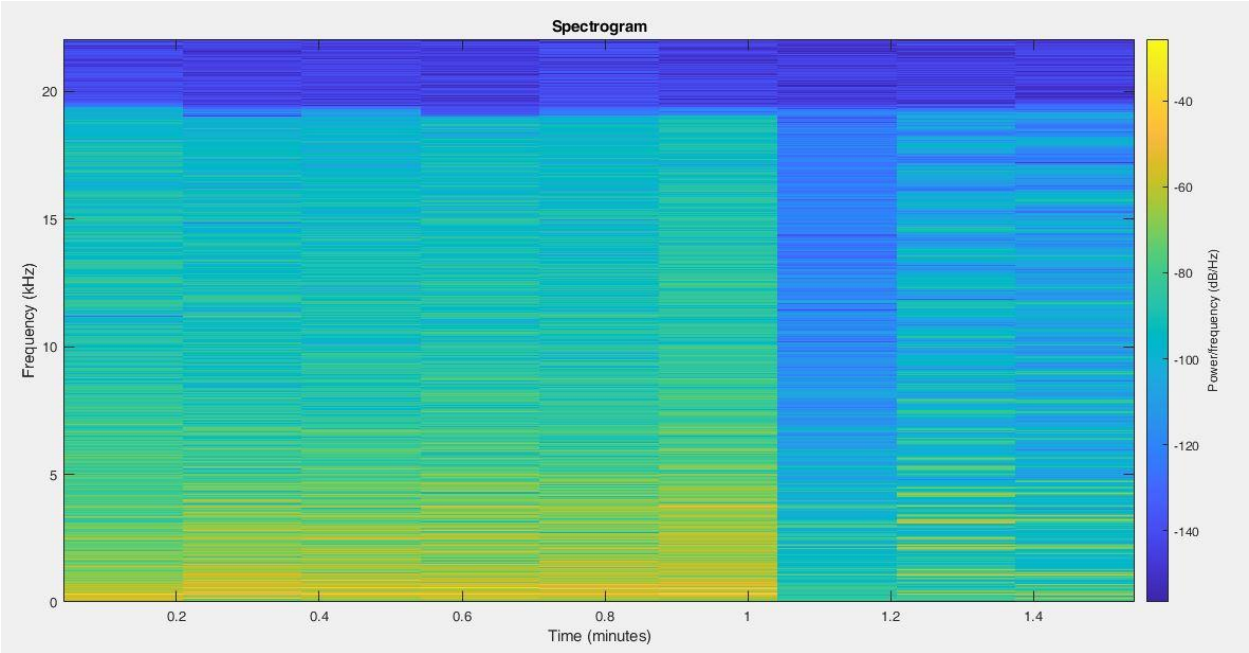

Figure 1.4: Spectrogram of $x_1(t)$

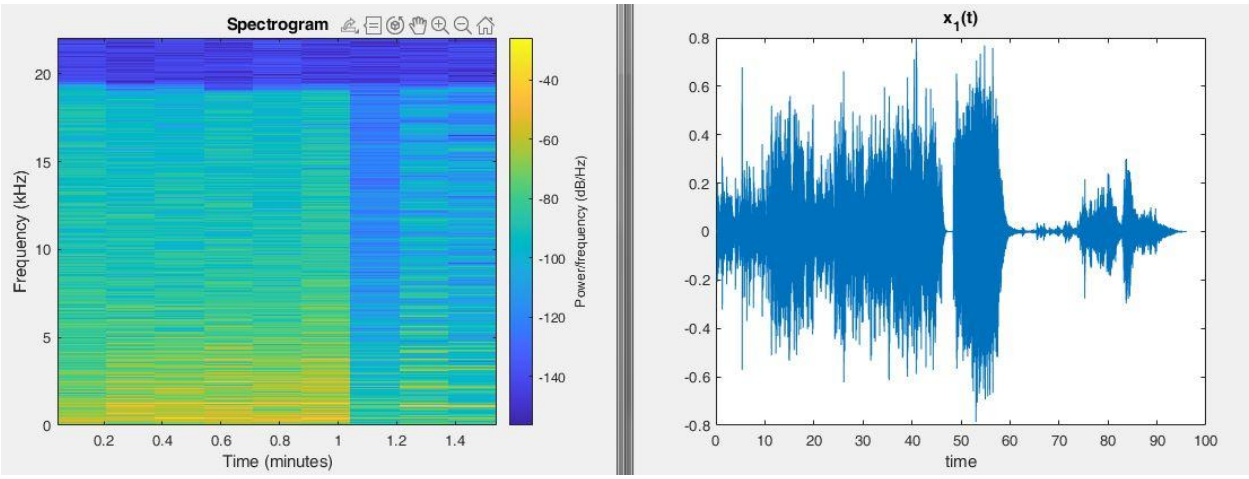

Figure 1.5: $x_1(t)$ & it's spectrogram

Overally based on the spectrogram and the signal itself we can detect the signal changing rate. For instance in the time bound around minute 1, a lot of changing and alteration can be detected. Meanwhile in the spectrogram we plotted, it can be observed that high frequencies appear with high power around that specific time interval.

By listening to the audiofile we can conclude many things most of which are stated below.

- The times where violin is beeing played due to it's nature that holds a lot of change and high frequencies we can detect that in spectrogram high frequencies appear with yellow color which stands for high power frequencies.
- Around the time between 1:10 to 1:25 the whole file goes silent for a short period and it is obvious that in this interval we expect no change and alteration in the signal amplitude which exactly happens based on the computations of spectrogram and we see no high frequencies there in that interval because all frequencies are blue which means low power.
- Among all the instruments which have been played in the file and comparing the frequencies which appear in the spectrogram and the signal carries, we can conclude that violin has the highest speed of beeing played on average.

(+) Since different instruments are beeing played in the audiofile and each instrument and it's voice has it's own alteration and freqeuncy and overally there will be different tones and frequencies two channels are used separately to represent the audio.

(+) Let's start with the concept behind each argument. ***window*** stands for the length of segments of the signal and the lower it is more accurate the analysis would be 'cause we can detect frequencies in shorter periods. ***noverlaps*** stands for the number of samples which overlap in each segment and again the higher it is, more accurate our analysis would be. Other arguments such as ***nfft*** and ***fs*** should be the amount which they currently are, cause otherwise our result would not be the way we expect.

- This part is implemented in **P1_3.m**

IV)
(+) Higher frequency signals have more information in one period while low frequency signals don't. In simple words, high frequencies mean high alteration. It is expected that when signal is changing with a high rate, it's multiplication with the shifted wavelet produces an accurate output because more information is being used. On the other hand in this case when computing in frequency domain because of high alteration in signal amplitude we can not expect good accuracy.
In low frequencies alteration of the signal amplitude is low and less information in specific interval is being multiplied by the wavelet. This is reason behind the fact that in this case computations in time domain are not accurate while computation in fourier domain are highly accurate.

Section 2: DTMF Voice Detection

I)

In this part we are going to write a function which detects which key has been pressed. The function is named ***dtmf*** and the code can be seen in figure 2.1.

```
function keyboard=dtmf(y,fs)
        key=['1','2','3','A'       %keys
            '4' '5' '6' 'B'
            '7' '8' '9' 'C'
            '*' '0' '#' 'D'];
        fh=[1209,1336,1477,1633];  %high frequencies
        fl=[697,770,852,941];      %low frequencies
        %%-------------Computation of FT-----------------
        t_start=0;
        t_end=length(y)/fs;
        T=t_end-t_start;
        N=fs*T;
        Y=fft(y);
        f=0:fs/N:fs-fs/N;
        Y=Y(1:length(Y)/2);        %cutting the FT to half
        f=f(1:length(f)/2);
        Y=abs(Y);
        %%-------------Finding f_high f_low-------------
        X=sort(Y);
        peaks=X(end-1:end);
        index1= Y==peaks(1);
        index2= Y==peaks(2);
        F=[f(index1) f(index2)];
        F=sort(F);
        f_high=F(2);
        f_low=F(1);
        %%---------------finding the key---------------
        for i=1:4
            flag=0;
            for j=1:4
                if f_high>fh(i)-10 && f_high<fh(i)+10
                    if f_low>fl(j)-10 && f_low<fl(j)+10
                        flag=1;
                        break;
                    end
                end
            end
            if flag==1
                break;
            end
        end
        %%--------------Assigning the output-----------
        keyboard=key(j,i);
end
```

Figure 2.1: DTMF Function

- Now let's take a look at the code we have written. First we initialize some matrixes as the variables we need such as the keys and the frequencies. In second stage Fourier Transform computation is conducted. Meanwhile in this stage we cut half of the transform to keep only 2 deltas in the specific tones. In next stage peaks which are the location of the tones are detected and finally we achieve the frequencies that are available in the signal(The high frequency and low frequency). Then we find the index of the key based on the frequencies we managed to find as the tone of the signal and in the end we assign the key we found to the output of out function.

For checking the functionality of this function all the signals from '1.wav' to 'D.wav' are checked. For instance we have brought '3.wav' and 'hash.wav' in **P2_1.m** and results can be observed in figure 2.2

```
>> P2_1
key 1=
3
key 2=
#
```

Figure 2.2: Test case for dtmf

- The function is written in file **dtmf.m**
- Test case is written in file **P2_1.m**

II)
In this part we aim to detect the number dialed in audiofile **phone_number** using the function we wrote in previous part(dtmf).
By listening to the audiofile we can detect this fact that the code we are trying to decrypt has length of 8. Therfore we need to divide the whole signal into 8 segments with same length and decode each part.

After decoding each part conclusion in figure 2.3 was obtained.

```
>> P2_2
key =
82084180
```

Figure 2.3: Phone number

- This part is implemented with file **P2_2.m**

Section 3: Compressing Image Via 2 Dimensional Fourier Transform

I)
(+)Based on the pictures given in the question, matrixes are defined to present the images and the result can be observed in figure 3.1

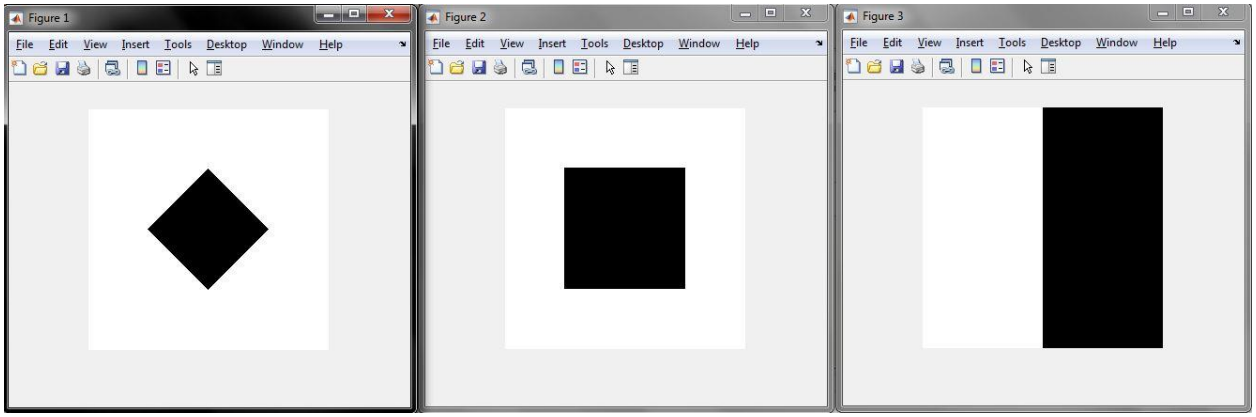

Figure 3.1: Black & White Images

- Code for this part is in file **P3_1.m**

II)
(+)
Now we plot 2 dimensional fourier transform amplitude of these 0 and 1 sequence we created. The result can be observed in figure 3.2.

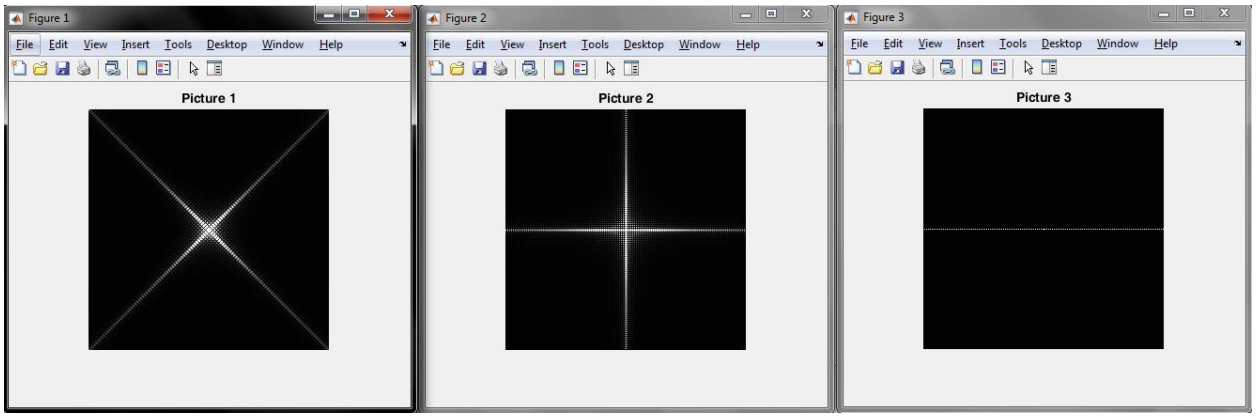

Figure 3.2: 2 dimensional Fourier Transform of the Pictures

- Code for this part is in file **P3_2.m**

As it was already stated and we expected in the locations where the color changes high frequencies appear while in locations where the color is stable in one place, low frequencies appear in the transform. Looking at picture 1 fourier transform we can observe that high frequencies appear near the edge where color changes from black to white and the same thing happens about the 2 other pictures.

III)

Now we want to load the image of the tiger given as prepared data and implement some algorithms based on fft2 to compress the image.

First 2 dimensional fourier transform of the pictur is computed and saved in a matrix. For 5 percent compression we look for those elements in fourier transform that have lower absolute value than factor of 1/3000 of the maximum of fourier transform and for 50 percent compression we look for those indexes with factor of 1/500. This way those elements in fourier transform which are not significant will turn zero and the picture is compressed. Below in figure 3.3 we can see compressed images. The code we have written will also save the compressed images with suitable names.

Figure 3.3: Compressed Image with factor of 5%

Figure 3.4: Compressed Image with factor of 50%

Now let's compare the outputs we gained with the original picture.
As we can observe the pictures are compressed and the quality of the picture has decreased a little. The one which is compressed with factor of 5 percent still carries the same quality, however a little lower. But the compression is completely obvious in Figure 3.4 since the picture is compressed 50%.

- Code for this part is in file **P3_3.m**

(+) Because fft2 of colored pictueres would be different from grayscale images and if we turn some elements zero in their fourier transform, it means we are losing some data that is non-reconstructable and not only won't the picture be compressed but also the picture will change completely and no longer would be the picture we wanted.

Section 4: Brain Signal Frequencies Analysis

I)

In this part we choose 3 EEG signals to process. The signals we choose are **v1.mat** & **v2.mat** & **v4.mat**

(+) We load these signals into arrays in matlab. For the rest of the process we've got up ahead I have chosen the first row.

Now in **x1 x2 x3** EEG signals are stored for further processes.

(+) Now we plot the signal in time and frequency domain and the results have been brought in figures 4.1 and 4.2

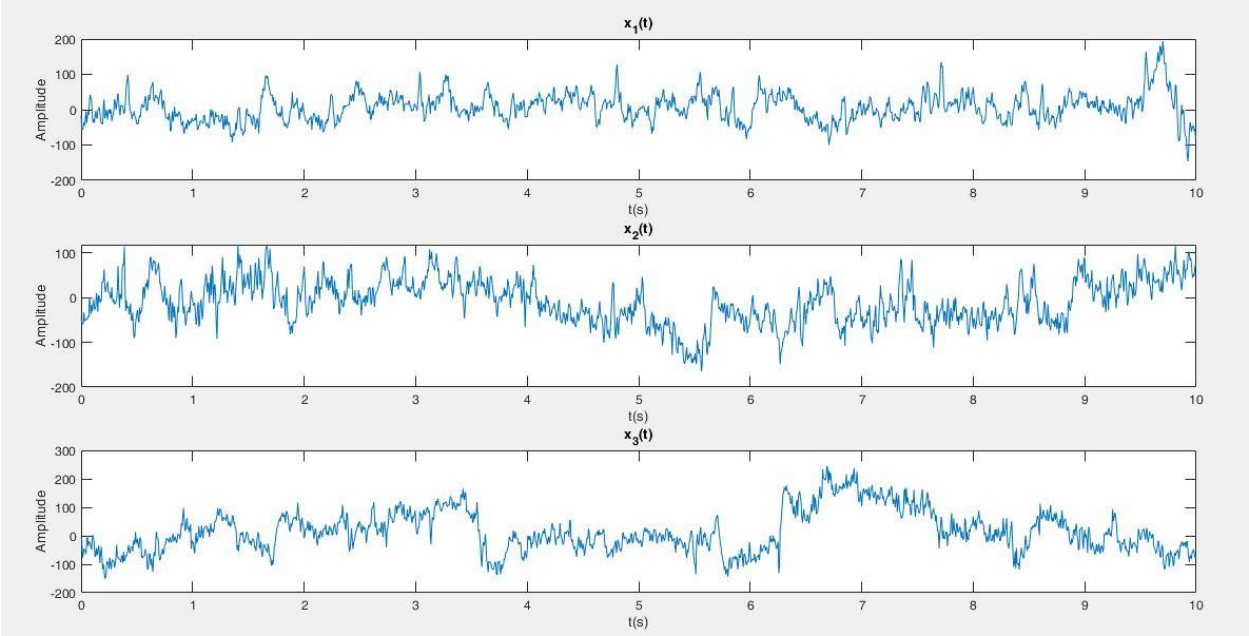

Figure 4.1: EEG signals in time domain

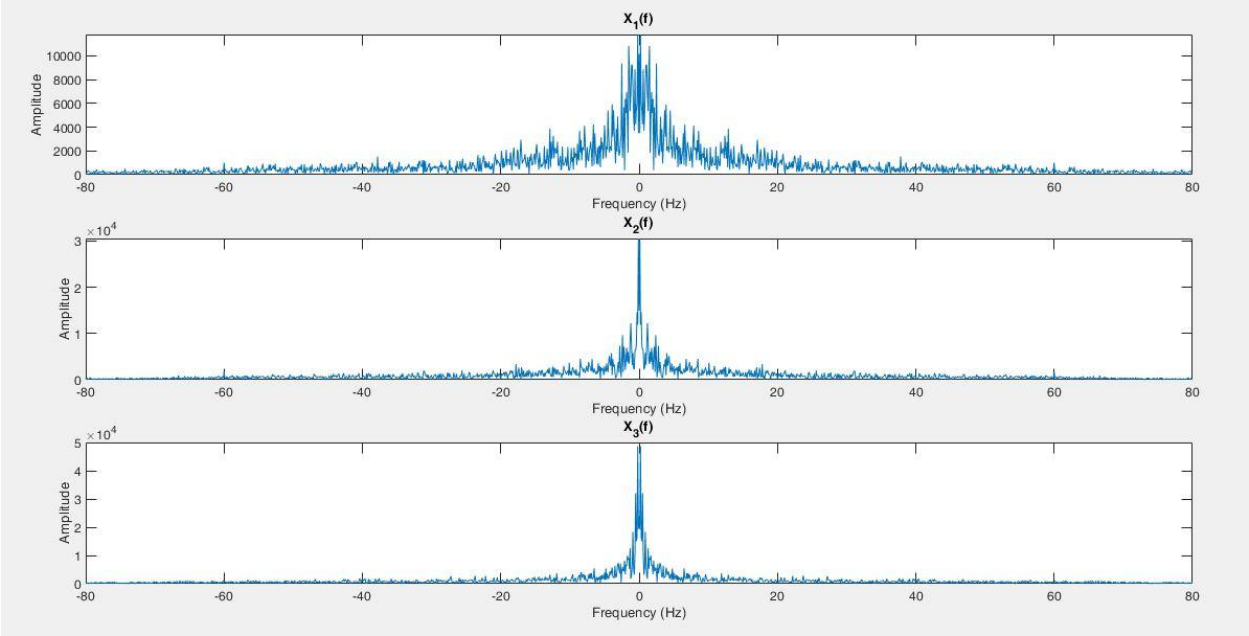

Figure 4.2: Fourier Transform of EEG signals

- This part code is in file **P4_1.m**

(+) Categorizing the changing rate in 5 levels descending from level 1 to level 5. We can say level 1 refers to gamma, level 2 for beta, level 3 for alpha, level 4 for theta and level 5 for delta.
Observing the time domain signal changing rate of level 1 appears in the time domain amplitude around 30. Level 2 appears around 50. Level 3 and 4 and 5 appear around amplitude of 50 to 100
So we can guess that gamma's amplitude might be around 30. beta's amplitude might be around 50 and the 3 remaining signal carry the amplitude of around 50 to 100.

II)

Now we want to extract alpha, beta, delta, theta and gama signal from the EEG signal.

Extraction has been conducted and in next page results can be observed in figure 4.3
The algorithm we used to extract these signals is obvious in the code and straightforwad. Making the fourier transform amplitude at the frequencies which should not appear in the signal zero and reconstructing the time domain signal.

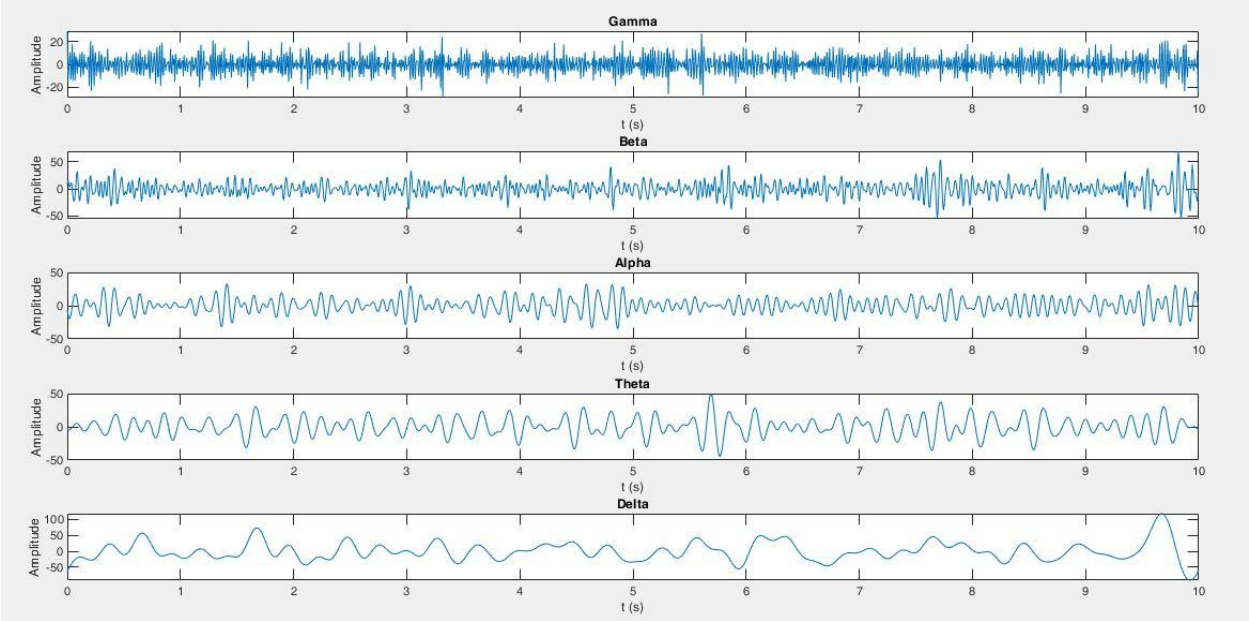

Figure 4.3: Gamma/Delta/Alpha/Theta/Delta Brain Signals

- Code for this part is **P4_2.m**

(+) As we guessed in 2 previous parts that each signal holds what sort of amplitude, we extracted these signals from the first signal and as we can see we could somehow guess the amplitude correctly with a little error. Based on Figure 4.3 the amplitude of gamma, beta, alph, theta and delta signals are respectively around 20, 50, 50, 50, 100.

(+) Yes we can detect how the brain is functioning in this 10 second. Since the amplitude of the gamma signal and beta signal are low we can deduce that the brain is not in situation of beeing highly aware or problem solving however he might be a little. As the frequency decrease and we reach out to signals alpha, theta and delta the amplitude of these signals rise and we can conclude that the brain is more in a situation of relaxing and chilling or maybe sleeping.
In conclusion if we want to present the state of brain in this 10 second we can mention the sentence below:

- Brain is relaxing and sleeping, however some thinking might be in progress while in sleep.

Section 5: Introduction to Cepstrum

I)

In first part we want to code a function to implement Cepstrum Transform on an input signal. The code is brought below in figure 5.1

```
function y=Cepstrum(x)
        X=fft(x);
        X=abs(X);
        X=log10(X);
        y=ifft(X);
end
```

Figure 5.1: Cepstrum Transform

- The function file is **Cepstrum.m**

II)

Using the function written in figure 5.1, cepstrum transform of EEG signals from previous section is to be plotted. Result can be observed in figure 5.2

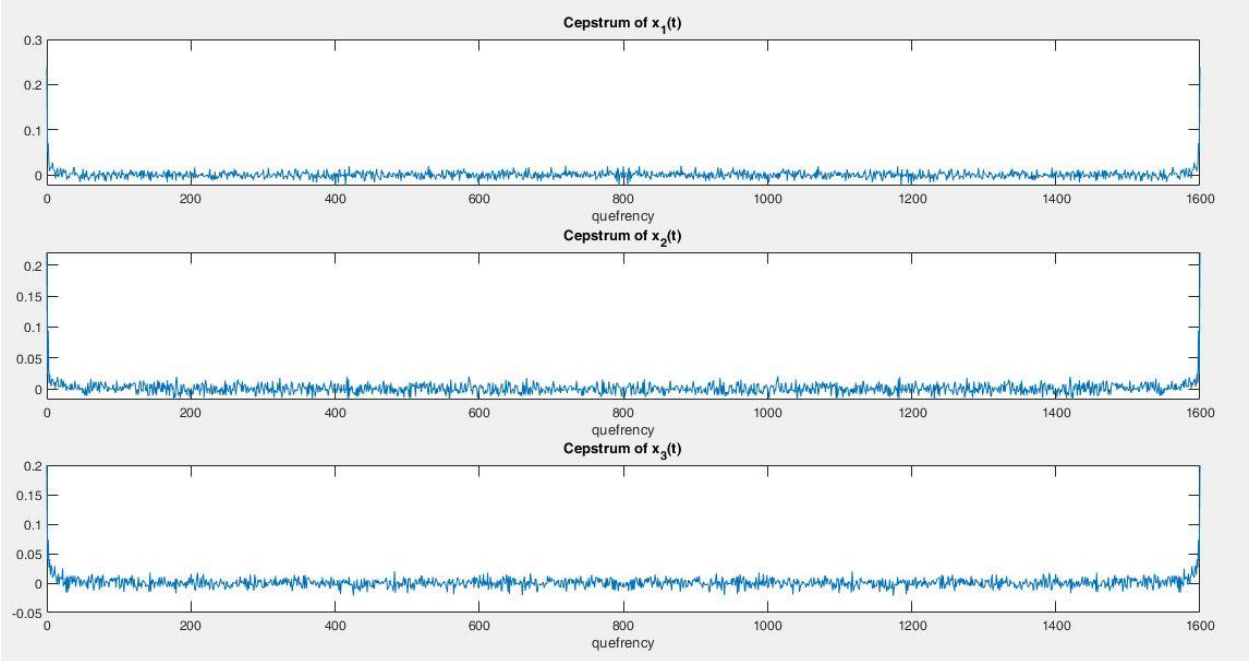

Figure 5.2: Cepstrum of EEG Signals

- This part is done via the file **P5_1.m**

(+) Based on figure 5.2, we want to detect the quefrencies where there are peaks in cepstrum of EEG signals.In the listed quefrencies interval or specific quefrencies below, there are peaks detectable.

for $x_1(t)$:

38
104
168.2
420
529 and other intervals

for $x_2(t)$:

[160,200]
323
416
[560,600]
[1000,1050]
1420 and …

for $x_3(t)$:

143
[200,250]
482
[600,650]
990
1120
[1300,1400] and …

(+) For further analysis of finding the frequencies in which the fourier transform of EEG signal has higher amplitude, diagram of the transform have been brought in figure 5.3

Assuming the maximum of fourier transfrom of signal is M. We consdier those frequencies strong in which the fourier transform amplitude is higher than 0.7M.

This way using figure 5.3, we can detect main and strong frequencies in FTs of EEG signals

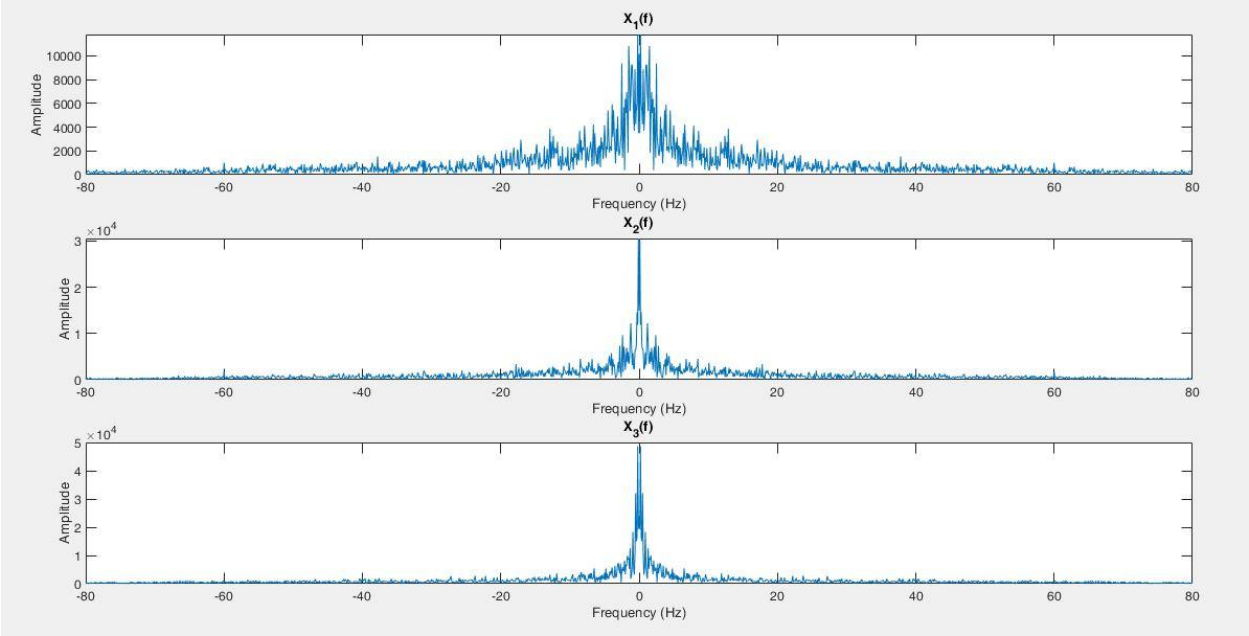

Figure 5.3: Fourier Transform of EEG signals

Based on the assumption made before bringing figure 5.3, strong frquencies are as listed below for each signal:

for $x_1(t)$:

- $|f| < 4\ Hz$

for $x_2(t)$:

- $|f| < 1.2\ Hz$

for $x_3(t)$:

- $|f| < 1\ Hz$

Based on what we can deduce from observing figures 5.2 and 5.3, we can state that the frequencies which are strong and the signal's fourier transform have a high amplitude in, can be estimated by a division of sampling frequency to those quefrency in which the cepstrum have peaks.
For instance as we already determined frequencies around 4Hz-2Hz for $x_1(t)$ we can see that if divide fs=160 to the peak in quefrency=38 or quefrency=104 we can reach the same result as 4Hz and 2Hz and frequencies around this amount. The same equations and estimations can be dervied for $x_2(t)$ and $x_3(t)$.

(+) In simple words to explain what quefrency is all about we can mention the deduction we made in previous question that there is a relation between quefrencies carrying high amplitude in cepstrum with the frequencies carrying high amplitude in FT. In conclusion we can state that quefrency is a scaled form of frequencies of the signal. It is a measure of time though not a sense of time domain.