Alireza Gargoori Motlagh

Stu. ID : 98102176

Armin Panjehpour

Stu. ID : 98101288

Final Project

Signals and Systems

Instructor: Dr. Hamid Karbalaei Aghajan

SHARIF
UNIVERSITY OF
TECHNOLOGY

Department of Electrical Engineering

Semester 99-2

1 Sampling & DFT

Functions:

- HalfBandFFT.m
- CTFourierTransform

HalfBandFFT.m:

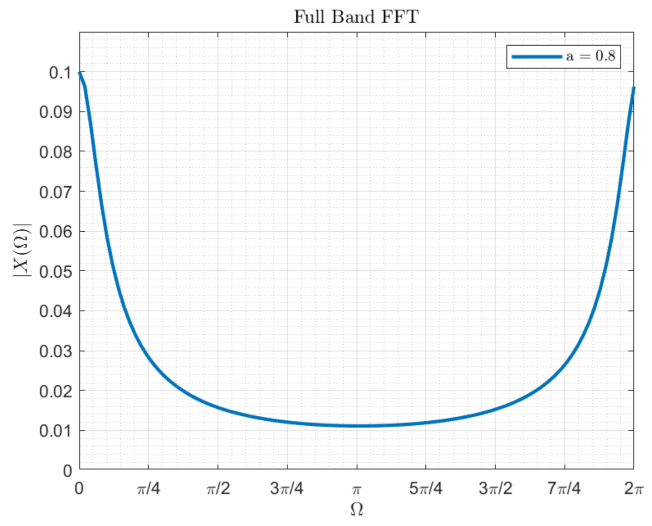
This functions receives a discrete signal as an input and in the output it generates the DFT of signal; as we know the DFT of a discrete signal with length N is computed as follows:

$$X[k] = X(\Omega)\big|_{\Omega=\frac{2k\pi}{N}}$$

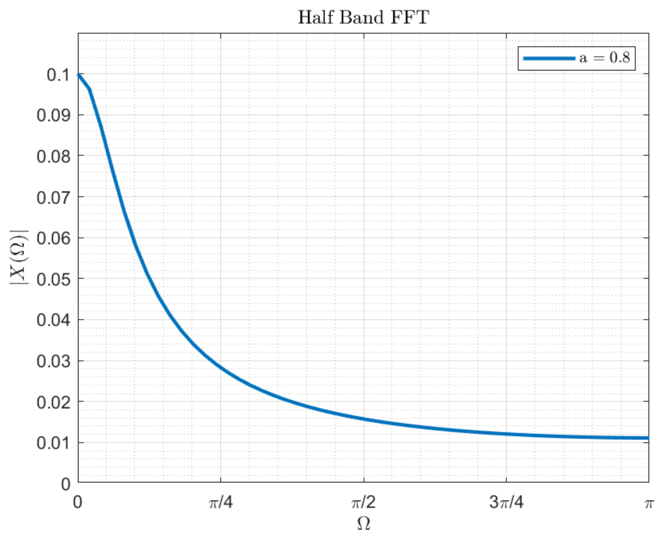
So, to compute the DTFT of a signal, which we know is periodic with period 2π and also symmetric for real signals, we use the DFT samples; which was computed with the FFT algorithm for fast implementation. Also we have normalized the magnitude of fourier transform for better visualization by the factor $\frac{2}{N}$.

Example:

$$x[n] = a^n u[n] \leftrightarrow X(\Omega) = \frac{1}{1 - ae^{-j\Omega}} \qquad |a| < 1$$



As we observe, the DTFT is symmetric with respect to $\Omega = \pi$; so we plot only half band of DTFT:



Nyquist Frequency:

As we know, to be able to reconstruct the signal from the down-sampled signal, which is shifted versions of continuous time Fourier Transform of signal, the following condition should be met:
$(f_s \text{ is sampling frequency and } f_{\max} \text{ is the maximum frequency of signal})$

$$f_{\max} < f_s - f_{\max} \rightarrow \boxed{f_s > 2f_{\max}}$$

Aliasing:

We know that, from what we have learned during the course, if the condition of Nyquist frequency is not met for a signal, its fourier transform, and hence the fourier inverse of that signal, is interfered and that is the phenomenon which we call *Aliasing*. So, the reconstructed signal would not be the same as original signal:

$$\hat{x}[n] \neq x[n]$$

The example below demonstrates the above explanations:

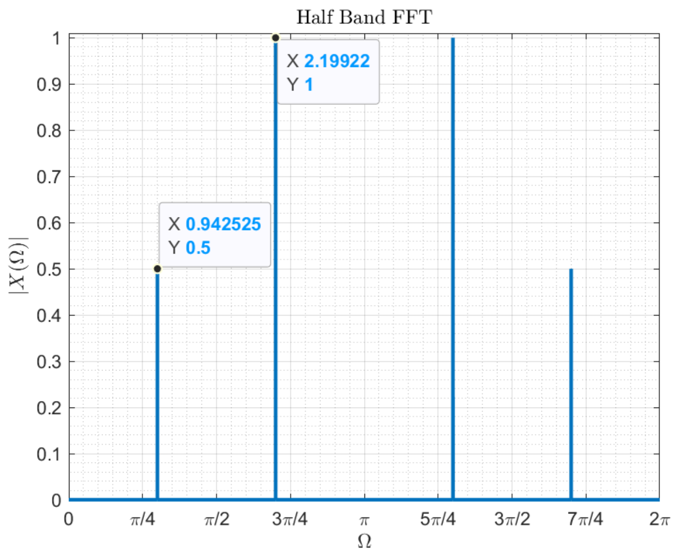
Example:

Consider x(t) = $\frac{1}{2}\sin(2\pi.30t) + \cos(2\pi.70t)$

If we sample x(t) with $F_s = 200Hz$, the Nyquist condition is met, because:

$$F_s = 200 > 2f_{max} = 140\ Hz$$

So, if x[n] = x($\frac{n}{Fs}$) , the DTFT of x[n] has no aliasing, as we see below:



We know that in the discrete signal, each mapped frequency is computed as follows:

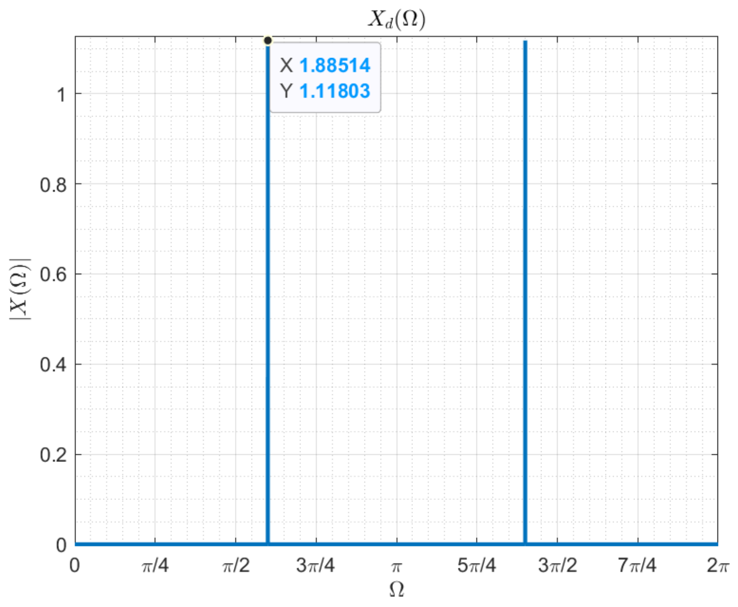
$$\Omega = \frac{w}{Fs} \rightarrow$$

$$\Omega_1 = \frac{2\pi.30}{200} = \frac{3\pi}{10} \cong 0.942 \qquad \Omega_2 = \frac{2\pi.70}{200} = \frac{7\pi}{10} \cong 2.199$$

Which agrees with the above figure; However, if we down-sample x[n] with rate 2, in fact we are sampling x(t) with $F_s' = \frac{F_s}{2} = 100\ Hz$; hence Nyquist condition is not met:

$$F_s' = 100Hz < 2f_{max} = 140\ Hz$$

So, we expect aliasing to happen; which can be seen in the next page:



As it is obvious from the above figure, the frequency in which the DTFT has a peak is not the corresponding continuous frequency. In addition, we have lost the other component frequency and also the value of the peak is not true. So we are not able to reconstruct the original signal due to the aliasing.

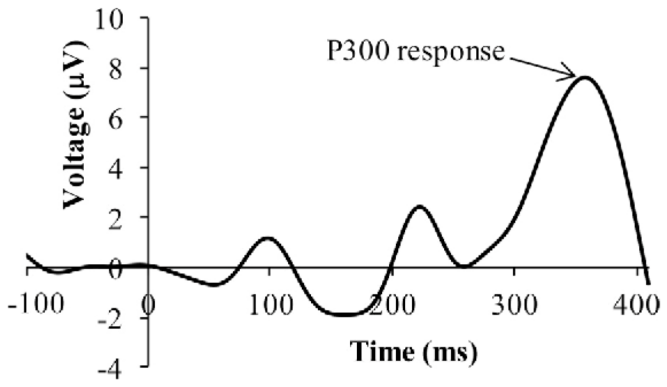
+ **CTFourierTransform.m**

This function computes the fourier transform for a continuous-time signals(i.e. the frequency axis contains all frequencies in Hz) and plots the one-sided amplitude spectrum. This function is required too much in the following parts, so we write that in this part.

2 Introduction to EEG signals

- **ERPs and P300:**

An event related potential is a measurement method to capture the exact result of a sensory, cognitive or motor event. ERPs are measured by averaging over many EEG signals which are the results of many trials. ERP waveforms consist of series of positive and negative voltage bumps which are caused by some components. Most ERP components are shown by a letter(N/P) indicating the polarity followed by a number which indicates either latency in milliseconds or the components ordinal position in the waveform. For example, a positive peak which is the first peak in the waveform that last for 100 milliseconds will be written P100 or P1. P300 is a going positive at 300 ms ERP which is related to processes that involve classifying or updating memory representations of the stimulus.

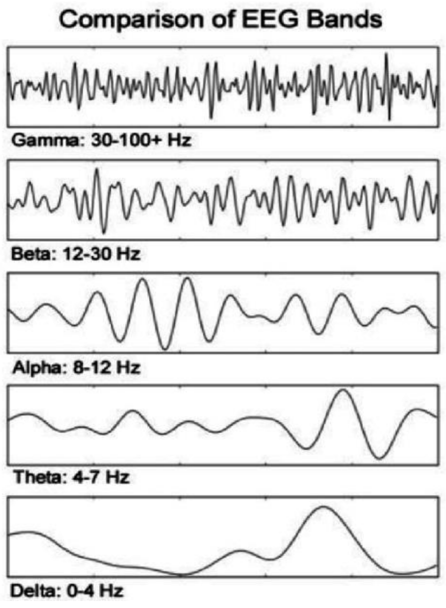


P300 is usually elicited using the **oddball paradigm** which when it is recorded by EEG, it surfaces as a positive deflection in voltage with a latency about 250-500 ms.

Some other ERPs that are used in researches:

- N400 in ELAN (Early Left Anterior Negativity)
- P600

- **Frequency bands:**



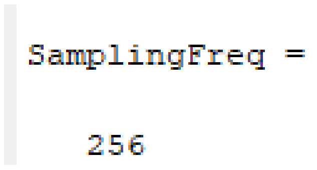
Although there are different classifications for EEG signals frequency bands, one of the classification is the one you see in the image in front. According to this definition, there are 5 frequency bands as you see in the image. Delta, Theta, Alpha, Beta and Gamma. Usually **Delta band**, tends to be the highest in the amplitude and the slowest waves. Normally this frequency band is normal as the dominant rhythm in the infants up to one year old and adults` slow-wave sleep. Also, it has been found during some continues-attention tasks. **Theta band**, is classified as “slow” activity. It may be seen in drowsiness or arousal in older children and adults. It is seen when we`re idling, too. This frequency band is completely normal in children up to 13 years old and in sleep but it`s abnormal in awake adults. **Alpha band**, is usually seen in posterior regions of head in both sides, being higher in amplitude on the dominant side and in central cites(c3, c4) at rest. It appears when closing eyes and relaxing and disappears when opening eyes or getting alerted by any mechanism like thinking or calculating. It is the dominant rhythm seen in normal relaxed adults. This band is presented during most of the life specially after thirteen. **Beta band**, has a low amplitude and its activity classifies as “fast” activities and it is closely linked to the motor behavior. It has been founds on both side of the head with a symmetric distribution and it is mostly evident in the front. It happens during active thinking, focusing, being anxious, have eyes open and when highly alerted. Also, it is accentuated by sedative-hypnotic drugs. **Gamma band**, is found in Somatosensory cortex and shows rest state of motor neurons. Gammas are thought to represent binding of different populations of neurons together into a network for carrying out a motor or cognitive function.

- **Sampling frequency:**

EEG signals are usually recorded at sampling rates between 250 and 2000 Hz in clinical and research things. Why? As you saw in the previous part, maximum frequencies are in Gamma band which they`re about 100 Hz. So according to the Nyquist, sampling rate should be at least twice or 2.5 times bigger than the biggest frequency available which tells us the frequency rate should be at least around 200-250 Hz.

- **Sampling rate of the given data, "SubjectData1":**

```
SamplingFreq = 1/(Subject1.train(1,3) - Subject1.train(1,2))
```



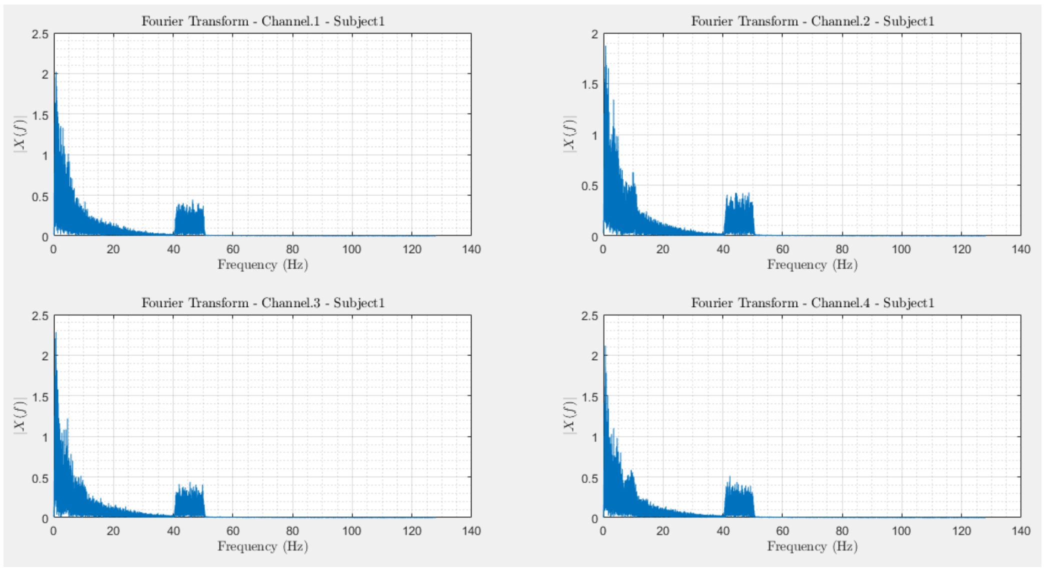
```
SamplingFreq =

   256
```

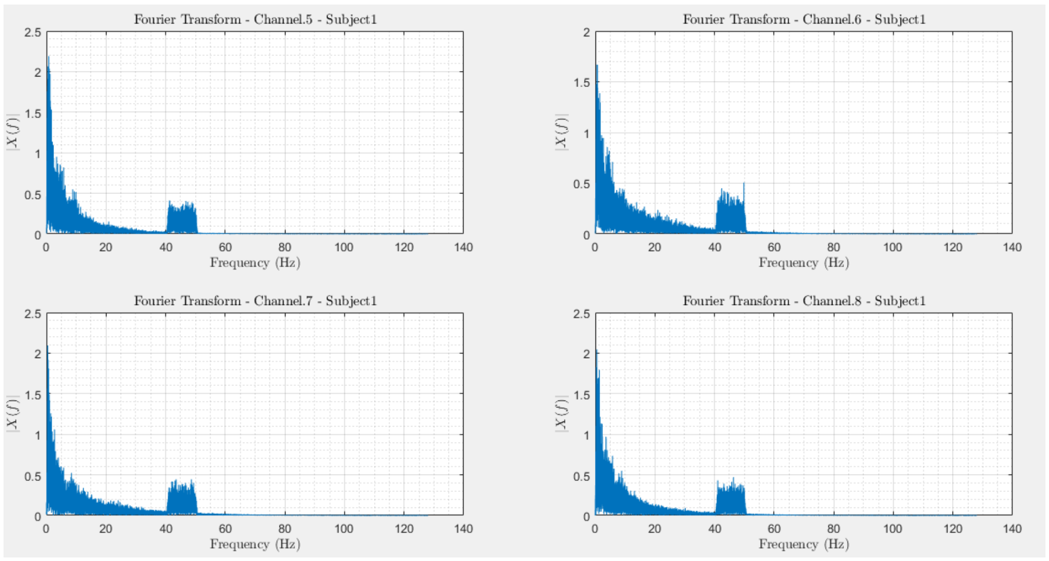
- **Guessing cut off frequency Depend on frequency bands information:**

Depend on the information gathered about frequency bands, cut off frequency could be for example 70 Hz because the task is not that complex. We`re not sure until we check the Fourier transform of the data.

- **Finding cut off frequency using Fourier transform:**

Here, Fourier transforms of each 8 channels are plotted using CTFourierTransform function which we saw in previous parts:





As the plots show, our data highest frequency is about 50 Hz. In range of 40 to 50 - Hz, a noise(maybe containing power line noise around 49-50 Hz) is on the data which it has to be removed. So the best cut off frequency is **40 Hz**.

- **Frequency range with the highest energy:**

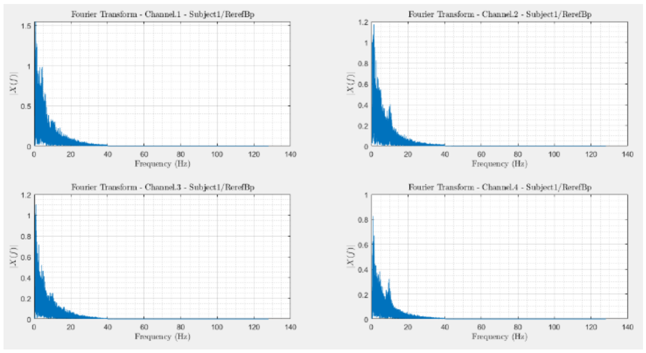
As you can see in the plots in the previous part, our signal`s energy is mostly between 0.5 to 40 Hz.

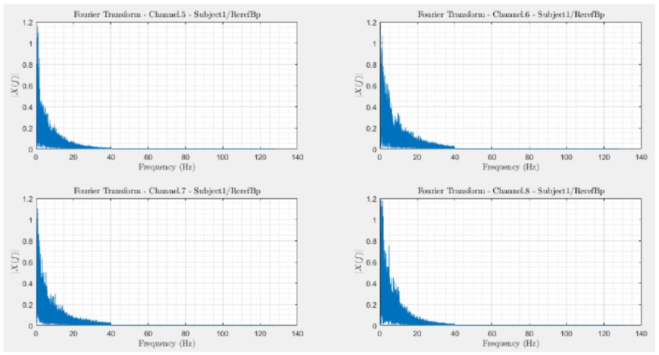
- **Low Pass filter cut off frequency:**

Low pass filter cut off frequency would be 0.5 Hz according to the previous part.

- **Re-referencing and Band Pass filter:**

To remove the DC component of the signal, re-referencing to the average of all 8 channels is done. Also, instead of applying a low pass and a high pass filter for filtering, we can apply a band pass filter with this cut off frequencies: [0.5 40] Hz using Matlab **bandpass** function:





- **Is removing average enough for removing DC component?**

After re-referencing, average of all 8 channels is removed from the channels, so we could still have DC component. Actually frequencies between 0 to 0.5 or 1 should be removed using a high pass filter, too. Why? Because in EEG processing there is a method called ICA and this method is very sensitive to big amplitudes which this can cause problems finding components. So this part of the signal should be removed. What cause this big amplitudes? For example EEG electrodes were not ideally(correctly) placed on the head or Conduction gel between the head and electrodes were too much(more than enough).

- **Down Sampling:**

According to the previous parts` results, our data sampling rate is 256 Hz and the highest frequency after band pass filtering is 40 Hz. Nyquist says sampling rate should be at least 80 Hz. So we are just capable of applying a down sampling with the rate of 2 or 3. So we down sample our channels` data with the rate of 2 here using Matlab function **down-sample**. The new sampling rate is 128 Hz. (We didn't choose 3 because it would have been so close to the Nyquist limit …)

- **Epoching:**

We epoch our data to a 3d matrix which the first dimension of it is the number of channels which is 8, the second dimension is time of the data captured and the doc has said 200ms before giving the stimulus up to 800ms after giving the stimulus which with the sampling rate of 128 Hz it would be 128 samples and the third dimension is the number of times stimulus were given to the subject (number of trials). Epoching is done using the function below:

```
function dataEpoched =
epoch(inputSignal,backwardSamples,forwardSamples,StimuliTimes,Fs)
```

at the end, the output is a 8×128×7200 matrix which is our epoched data:

EpchedData   8x128x2700 double

- **Why do we do filtering before down sampling?**

Without filtering, we can`t know what is the lowest sampling which Nyquist tells us. If we down sample before filtering, some useless frequencies in our signal increase the sampling rate limit or aliasing could happen. But after filtering we have our useful data without any noise or useless info, So down sampling can be done according to the highest frequency we have.

- **Why Filtering, then Epoching?**

Filtering has to be done before epoching because after epoching we just have a part of the data (1s for each stimuli) and filtering would be done depend on just some parts of the data which would have a wrong result.

- **Why do we have to wait a little to do the experiment after the EEG starts capturing signals?**

Because we want to have 200 ms before each stimuli time of the channels` data in our epoch, Thus we have to wait at least 200 ms before giving the stimuli. Also, this time helps the EEG recorder and the subject to be in a stable situation.

3 Clustering based on Correlation

For two discrete-time signals, the value of correlation function at $\tau = 0$ is computed as follows:

$$R_{XY}[0] = \sum_{n=-\infty}^{+\infty} X[n]Y[n] = X.Y$$

which is simply the inner product of them.

Correlation Coefficient is defined as:

$$r_{XY} = \frac{(\sum_{n=-\infty}^{+\infty} X[n]Y[n])}{\sqrt{(\sum_{n=-\infty}^{+\infty} X^2[n])(\sum_{n=-\infty}^{+\infty} Y^2[n])}} = \frac{R_{XY}[0]}{\sqrt{R_{XX}[0]\, R_{YY}[0]}}$$

Based on the definition of inner product we can simplify the above equation:

$$r_{XY} = \frac{X.Y}{\sqrt{(X.X)(Y.Y)}} = \frac{X.Y}{\sqrt{|X|^2|Y|^2}} = \frac{X.Y}{|X||Y|} = \cos(X,Y)$$

In fact, correlation coefficient of two signals is an indicator of the angle between them, where we have considered each signal as a vector with its values as elements of this vector. So, this coefficient shows how much two signals are aligned with each other.

As we show that $r_{XY} = \cos(X,Y)$, we conclude:

$$-1 \le r_{XY} \le 1$$

Earlier in this part, we stated that correlation coefficient shows how much two signals (i.e. two vectors) are parallel or in the same direction. So the magnitude of r_{XY} is 1 if and only if X and Y are parallel and hence the angle between them would be 0 or π :

$$|r_{XY}| = 1 \leftrightarrow \cos(X,Y) = \pm 1 \leftrightarrow X[n] = \alpha Y[n]\,, \alpha \in \mathbb{R}$$

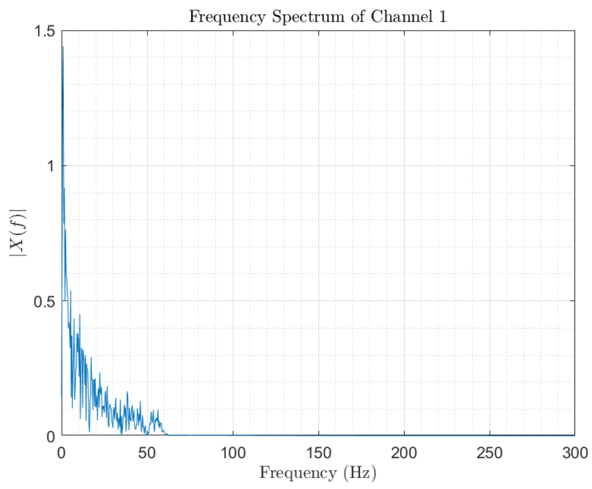
+ In the continuous form, the above results remain the same; because the inner product of two continuous signals is defined as:

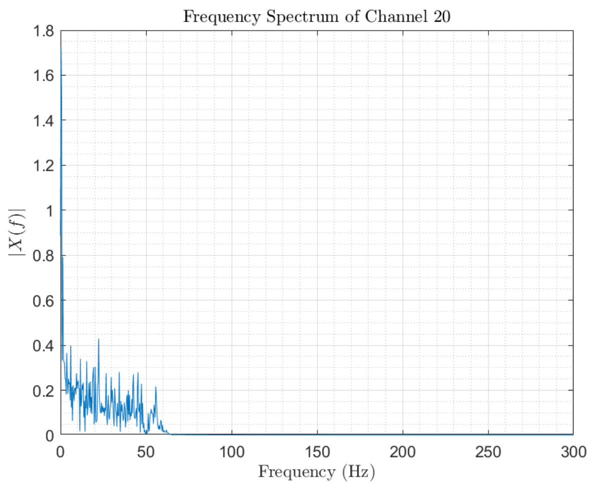
$$< X, Y > = \int_{-\infty}^{+\infty} X(t)Y(t)dt$$

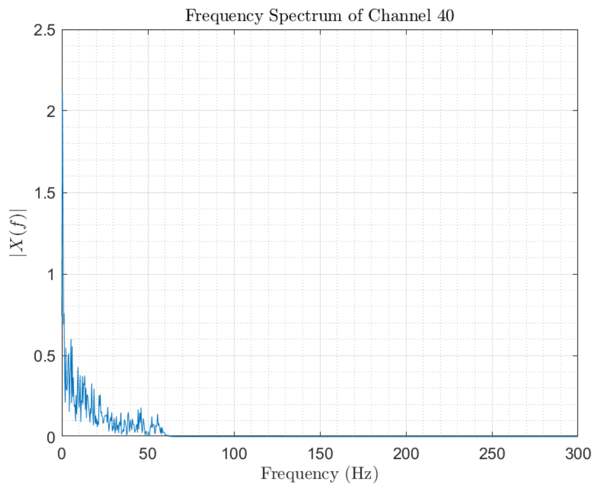
As we mentioned before, correlation coefficient is an indicator of the alignment of two signals. So this coefficient can be used to measure the similarity between them and is good a parameter for our purpose of clustering.

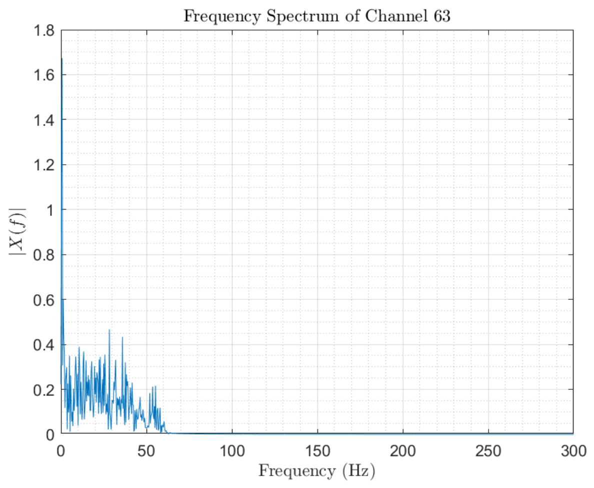
Correlation Clustering on 63-channel Data form BCI experiment:

By plotting different channels' frequency responses, we observe no filtering is needed and even the 50-Hz noise is filtered; for example:









First of all, we average signals over trials; so that the probable noise of experiment is removed and therefore we have a 63×1800 data.

From the channels' frequency responses, we observe that maximum frequency for each channel is less than 70 or 65 Hz. So to decrease the sampling frequency, the Nyquist condition would be:

$$F_s' > 2 \times 70\ Hz = 140\ Hz$$

So we can down-sample the data by rate 4 (i.e. choose 1 sample out of 4 consecutive samples) and still have the same frequency response. Now our data would be 63×450 and we now compute the *Correlation Coefficient* matrix which is simply a matrix that contains correlation coefficient of each pair of channels:

$$\text{Correlation Coefficient Matrix } \boldsymbol{R} \therefore \begin{bmatrix} r_{X_1X_2} & \cdots & r_{X_1X_n} \\ \vdots & \ddots & \vdots \\ r_{X_nX_1} & \cdots & r_{X_nX_n} \end{bmatrix}$$

- **CorrCoefMat.m**

This function computes the correlation coefficient matrix of a 2D data (like channels of EEG signals) based on the formula:

$$r_{X_iX_j} = \frac{X_i . X_j}{|X_i||X_j|}$$

So defining three matrixes as below

$$R_1 = \begin{bmatrix} X_1X_1 & \cdots & X_nX_1 \\ \vdots & \ddots & \vdots \\ X_nX_1 & \cdots & X_nX_n \end{bmatrix} \qquad R_2 = \begin{bmatrix} |X_1|^2 & \cdots & |X_1|^2 \\ \vdots & \ddots & \vdots \\ |X_n|^2 & \cdots & |X_n|^2 \end{bmatrix} \qquad R_3 = \begin{bmatrix} |X_1|^2 & \cdots & |X_n|^2 \\ \vdots & \ddots & \vdots \\ |X_1|^2 & \cdots & |X_n|^2 \end{bmatrix} = R_2^T$$

We can find the correlation matrix as:

$$\boldsymbol{R} = R_1./\sqrt{R_2.* R_3}$$

(The dot indicates elementwise operation.) This kind of implementation is much faster than the simple loops for each pair channel computation due to the high-speed matrix computations of Matlab. Our function does the above algorithm, and in the output, we have the correlation coefficient matrix.

Clustering using Correlation Coefficient matrix

For clustering method, we need to have a measure for distances between each pair of channels. To do this, we use the Distance Matrix defined as below:

Distance Matrix **D** = **1**- **R**

where **1** is a matrix of the same size of **R** which all its elements are 1. This is a good measurement of distance for two signals; since for two aligned signals we have:

$$r_{XY} = 1 \rightarrow d_{XY} = 0$$

and for two opposite signals (Y = -X) we got:

$$r_{XY} = -1 \rightarrow d_{XY} = 2$$

Now we cluster data channels to K clusters based on the following algorithm:

1. Put each channel in one cluster.
2. Merge the closest pair of clusters. (with the least distance)
3. Update the distance matrix.
4. Repeat the two above steps for K times.

For step 3, updating the distance matrix, we implement 2 methods:

1. **UPGMA**

   In this algorithm, at each clustering step, the updated distance between the joined clusters $A \cup B$ and a new cluster X is given by the proportional averaging of the $d_{A,X}$ and $d_{B,X}$distances:

$$d_{(A\cup B),X} = \frac{|A|.\, d_{A,X} + |B|.\, d_{B,X}}{|A| + |B|}$$

2. **WPGMA**

   In this algorithm, at each clustering step, the updated distance between the joined clusters $A \cup B$ and a new cluster X is given by the arithmetic mean of the average of the $d_{A,X}$ and $d_{B,X}$ distances:

$$d_{(A\cup B),X} = \frac{d_{A,X} + d_{B,X}}{2}$$

- **CorrelationClustering.m**

This function computes the clustering of the input data; it first finds the correlation matrix of data, and hence the distance matrix, with the help of *CorrCoef.m* and then it starts to cluster the data into K clusters. To indicate the method of updating the distance matrix, it has an input argument which can be 'UPGMA' or 'WPGMA'.

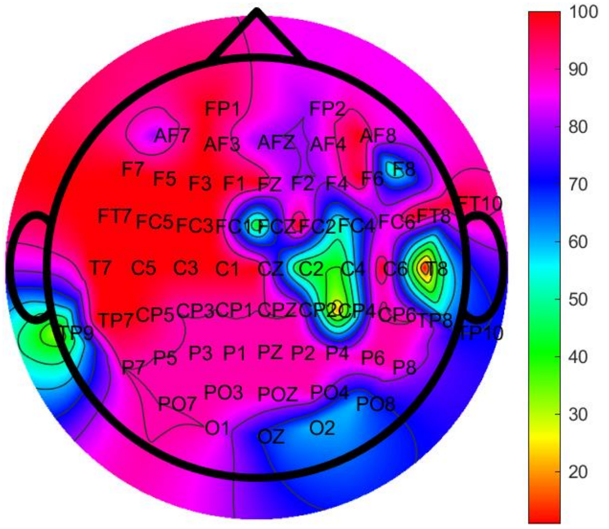
  - **UPGMA_Update.m**

    It updates the distances of a distance matrix where the row and column of the least value of the old distance matrix is given as an input for the function using *UPGMA* algorithm described in the last page.

  - **WPGMA_Update.m**

    It updates the distances of a distance matrix where the row and column of the least value of the old distance matrix is given as an input for the function using *WPGMA* algorithm described in the last page.

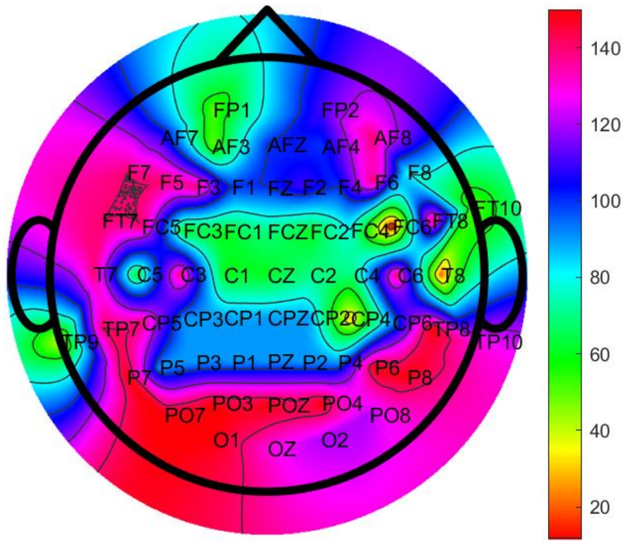
Example 1: Clustering the 63-channel Data into 10 clusters using *UPGMA* method



(Note that the electrodes of the same cluster are indicated by a same number and hence their color is the same.)

As we observe, generally the neighboring neurons are in the same cluster and provide more similar signal to an excitation. This is in phase with our expectations of the neurons' group functions which will be explained soon.

Example 2: Clustering the 63-channel Data into 15 clusters using *WPGMA* method



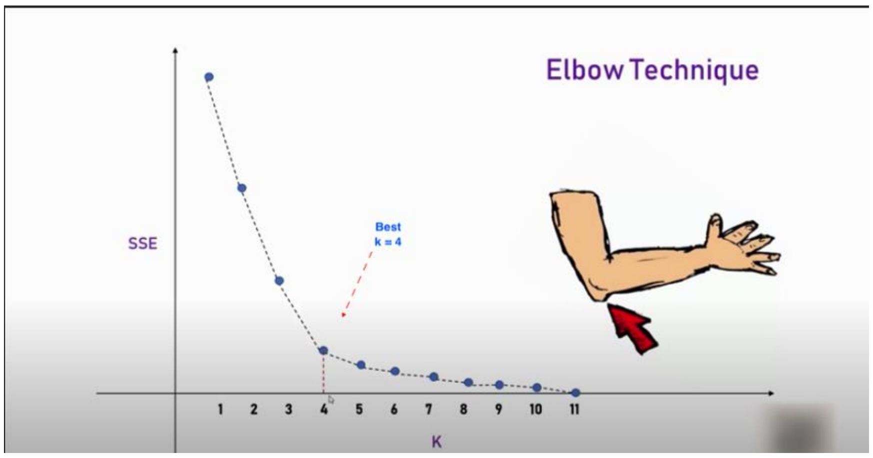
(Note that the electrodes of the same cluster are indicated by a same number and hence their color is the same.)

As it is noticeable, the effect of increasing the number of channels can be inferred from the above figure. Increase in the number of clusters caused the output to be less in phase with our expectation and the biological functions of neurons. However, it is not true that decrease in the number of clusters would lead to a better clustering and there is a trade-off. We will talk about this in the next section.

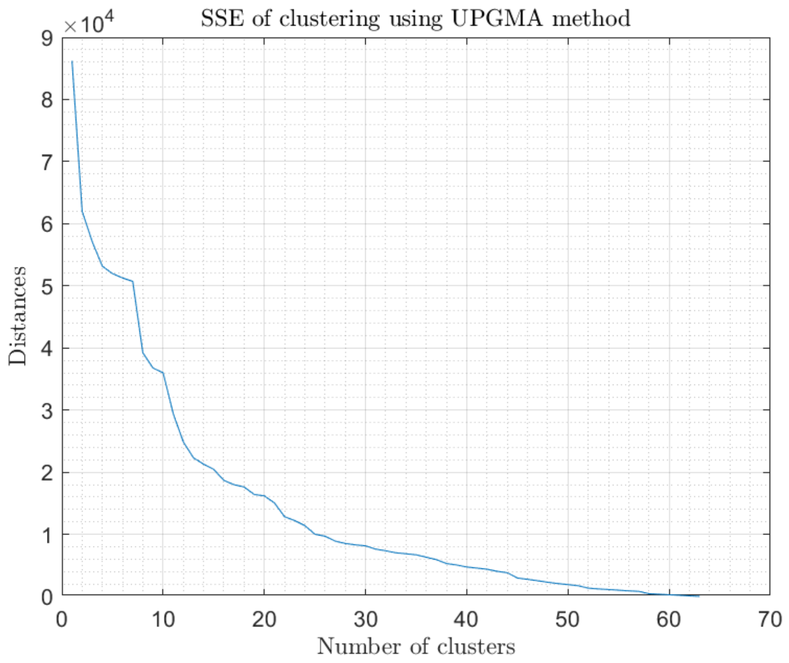
A Decision Criterion for the Optimal Number of Clusters

As we discussed, there is a trade-off for decreasing or increasing the number of clusters in our methods and generally, in any clustering method. One of the good options for finding the optimal number of clusters in hierarchical clustering algorithms is based on the *Sum of Squared Errors*, known as SSE.

One of the famous and popular methods for this purpose is the Elbow Method. This criteria finds the average distance of cluster members from the cluster's centroid for each number of clusters and the optimal number would be the one with a rapid and steep fall in the SSE. The figure below describes our purpose better:



For this algorithm, we have implemented the function **Elbow_Method.m** ; This function finds the SSE for each number of clusters using Correlation Clustering function we defined earlier. By plotting the results we can choose the optimal number of clusters:



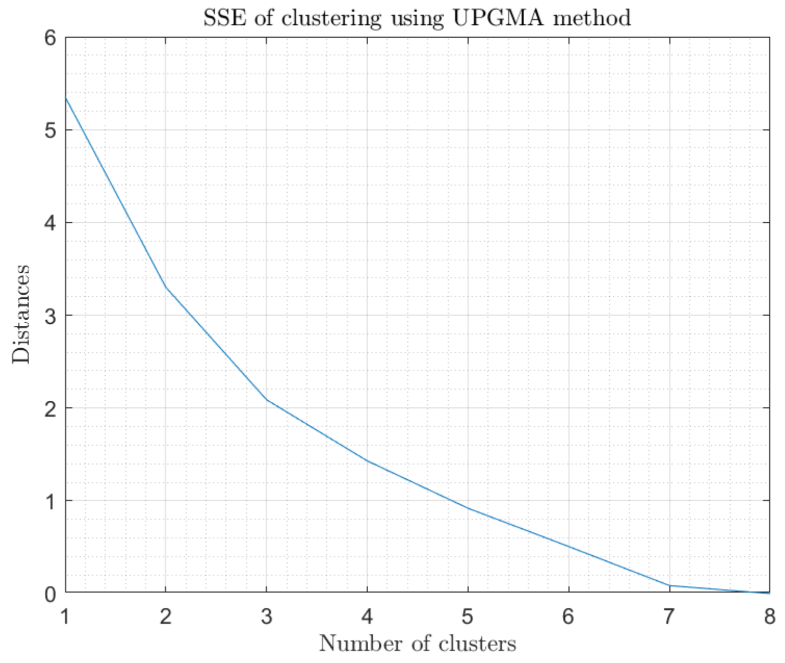
From the above figure, we can infer that the best number of clusters could be 10 for 63-channel data; Hence we used this number of clusters in the first topography of scalp and we observed a good clustering based on our knowledge. There are also some other methods like *Silhouette Method* which can a more exact number.

- Electrical activity correlates linearly with distance within the brain, i.e. when distance increases the correlation decreases. So there is an inverse relationship between Cross-correlation and distance. The second conclusion from this work is that the correlation is independent of brain hemisphere. This suggests that most probably the electrical signals are transmitted through the white matter of the brain. We assume signal transmission is through white matter because of the commissural tracts within the white matter which connect the two hemispheres of the brain. This means in practice it does not matter which side of the medial plane you place the electrodes. (1)

- As we observe, the electrodes in the same cluster are generally the ones that have small physical distances between them on the scalp. This phenomenon matches our expectation; because neighboring neurons, which are close to each other, can be considered as a network which its neurons are responsible for a same task.

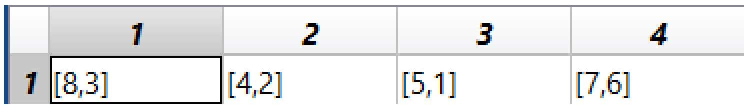
Correlation Clustering on 8-channel Data to find relationship between channels

First of all, we use elbow method to find the best number of clusters. By plotting the corresponded SSE we get:

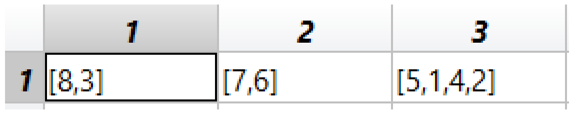


(1) : The Correlation between EEG signals as Measured in Different Positions on Scalp Varying with Distance (*Ronakben Bhavsar, Yi Sun, Na Helian, Neil Davey, David Mayor and Tony Steffert*)

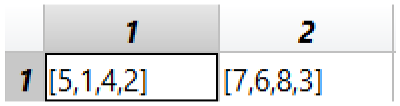
We infer from the above plot that the best number of clusters is 2, 3 or 4 clusters; which we get:



	1	2	3	4
1	[8,3]	[4,2]	[5,1]	[7,6]



	1	2	3
1	[8,3]	[7,6]	[5,1,4,2]



	1	2
1	[5,1,4,2]	[7,6,8,3]

By comparing to the original all-channel data, we infer that closer channels are more likely to be in the same cluster; so using the physical locations of the channels, for example for 4-group clustering, it might probably be that channels are grouped together like this:

$$\{PO7, P3\}, \{PO8, P4\}, \{CZ, FZ\}, \{OZ, PZ\}$$

4 Group Delay & z-Phase Filter

- **Calculating output of the given filter:**

- First filter:

$$\boxed{\begin{array}{c}\text{Given filter : } H(jw) = Ae^{-j\frac{\pi}{3}\mathrm{sgn(w)}} \text{ ,} \\ \text{Given signal : } x(t) = \cos(w_0 t) + \cos(2w_0 t) \\ \text{Output = y(t) = ?}\end{array}}$$

$\rightarrow\ X(jw) = \pi(\delta(w - w_0) + \delta(w + w_0) + \delta(w - 2w_0) + \delta(w + 2w_0))$

$\rightarrow\ Y(jw) = H(jw) \times X(jw) \rightarrow$

$$\boldsymbol{Y(jw) = \pi A(e^{-j\frac{\pi}{3}}(\delta(w - w_0) + \delta(w - 2w_0)) + e^{j\frac{\pi}{3}}(\delta(w + w_0) + \delta(w + 2w_0)))}$$

$$\rightarrow\ y(t) = \frac{A}{2}(e^{jw_0\left(t-\frac{\pi}{3w_0}\right)} + e^{j2w_0\left(t-\frac{\pi}{6w_0}\right)} + e^{jw_0\left(-t+\frac{\pi}{3w_0}\right)} + e^{j2w_0\left(-t+\frac{\pi}{6w_0}\right)})$$

$$\boxed{\rightarrow\ y(t) = A(\cos(w_0\left(t - \frac{\pi}{3w_0}\right)) + \cos(2w_0\left(t - \frac{\pi}{6w_0}\right))}$$

As we can see, the cosine functions are shifted $-\frac{\pi}{3w_0}$ and $-\frac{\pi}{6w_0}$.

- Second filter:

$$\boxed{\begin{array}{c}\text{Given filter : } H(jw) = Ae^{-j\frac{\pi}{3}w} \text{ ,} \\ \text{Given signal : } x(t) = \cos(w_0 t) + \cos(2w_0 t) \\ \text{Output = y(t) = ?}\end{array}}$$

$\rightarrow\ X(jw) = \pi(\delta(w - w_0) + \delta(w + w_0) + \delta(w - 2w_0) + \delta(w + 2w_0))$

$\rightarrow\ Y(jw) = H(jw) \times X(jw) \rightarrow$

$$\boldsymbol{Y(jw) = \pi A e^{-j\frac{\pi}{3}w}(\delta(w - w_0) + \delta(w + w_0) + \delta(w - 2w_0) + \delta(w + 2w_0))}$$

$$\rightarrow\ y(t) = \frac{A}{2}\left(e^{jw_0\left(t-\frac{\pi}{3}\right)} + e^{-jw_0\left(t-\frac{\pi}{3}\right)} + e^{j2w_0\left(t-\frac{\pi}{3}\right)} + e^{-j2w_0\left(t-\frac{\pi}{3}\right)}\right)$$

$$\boxed{\rightarrow\ y(t) = A\left(\cos\left(w_0\left(t-\frac{\pi}{3}\right)\right) + \cos\left(2w_0\left(t-\frac{\pi}{3}\right)\right)\right)}$$

As we can see, the cosine functions are both shifted $-\frac{\pi}{3}$

Thus, if the generalized phase is linear, all the frequency components would shift with the same value but if the generalized phase is non-linear, frequency components would shift differently and it causes distortion.

Group Delay:

$$\text{gd(w)} = -\frac{d\emptyset(w)}{dw}$$

- Checking this equation for the filters in the previous question:
  - Filter 1:
    gd(w) = 0 when w ≠ 0 and so group delay cannot be defined for filters with non-linear generalized phase.

  - Filter 2:
    gd(w) = $\frac{\pi}{3}$ which is equal to the shift value that we calculated in the previous part.

- **Group Delay Implementation – prove the equation below:**

$$\text{gd(w)} = \text{Re}\left\{\frac{j\frac{d}{dw}H(w)}{H(w)}\right\}$$

$$\rightarrow \frac{d}{dw}H(w) = jA(w)\frac{d\emptyset(w)}{dw}e^{j\emptyset(w)} + \frac{dA(w)}{dw}e^{j\emptyset(w)}$$

$$\rightarrow H(w) = A(w)e^{j\emptyset(w)}\ \rightarrow gd(w) = Re\left\{\frac{-A(w)\frac{d\emptyset(w)}{dw}e^{j\emptyset(w)} + j\frac{dA(w)}{dw}e^{j\emptyset(w)}}{H(w)}\right\}$$

$$\boxed{\rightarrow gd(w) = -\frac{d\emptyset(w)}{dw}}$$

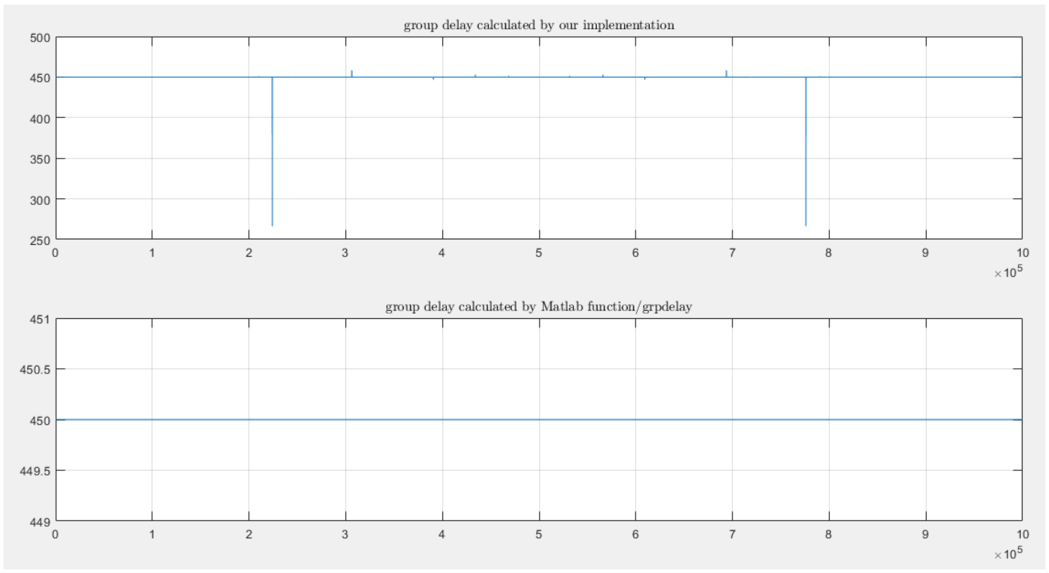
- **Group Delay Implementation:**

Using this formula, gd is calculated:

$$gd(w) = \mathrm{Re}\{\frac{j\frac{d}{dw}H(w)}{H(w)}\}$$

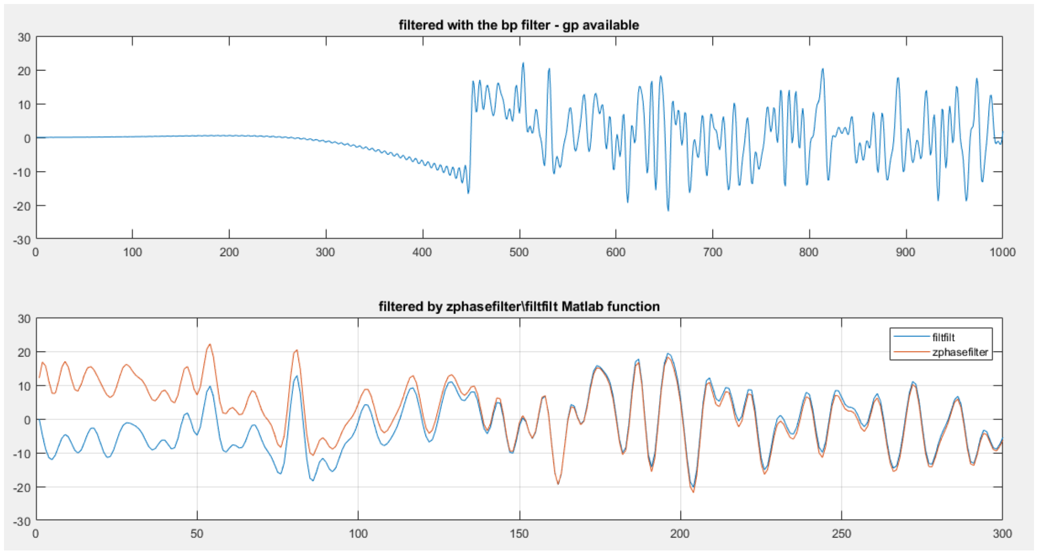
```
function gd = groupdelay(h,N)
    n = 0:size(h,2)-1;
    numerator = fft(h.*n,N); % N-point DFT
    denominator = fft(h,N);
    gd = real(numerator./denominator);
end
```

Here we use the BPfilter attached in the project files and calculate its gd. The output is:



As we see, group delay is 450 and our function and matlab function results are the same except for a little samples which is not that important. This inequality is maybe caused by non-linearity of the filter.

Z Phase Filter:



As we can see from the first plot, our signal is filtered but has 450 group delay. Then we implement *zphasefilter* which removes group delay and as you see it has the same output with the built-in function, *filtfilt* and it`s working well. (the difference between the two results at the first, is just caused by difference of the filters. The point here is the that the two signals are on each other and we don’t have group delay.)

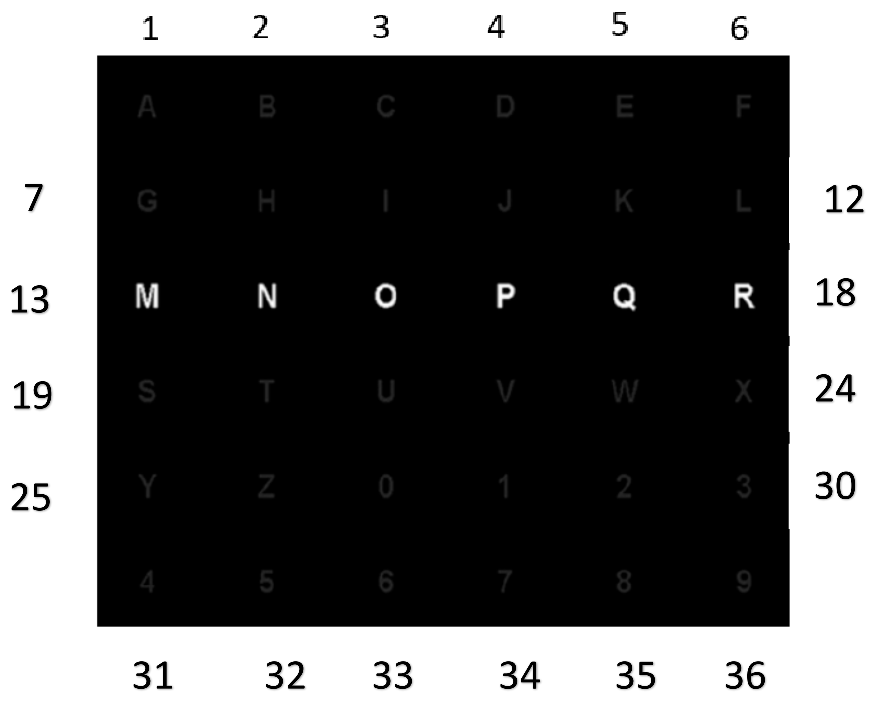
- **a Real Causal Zero Phase Filter:**

filter with impulse response: $\delta(t)$!

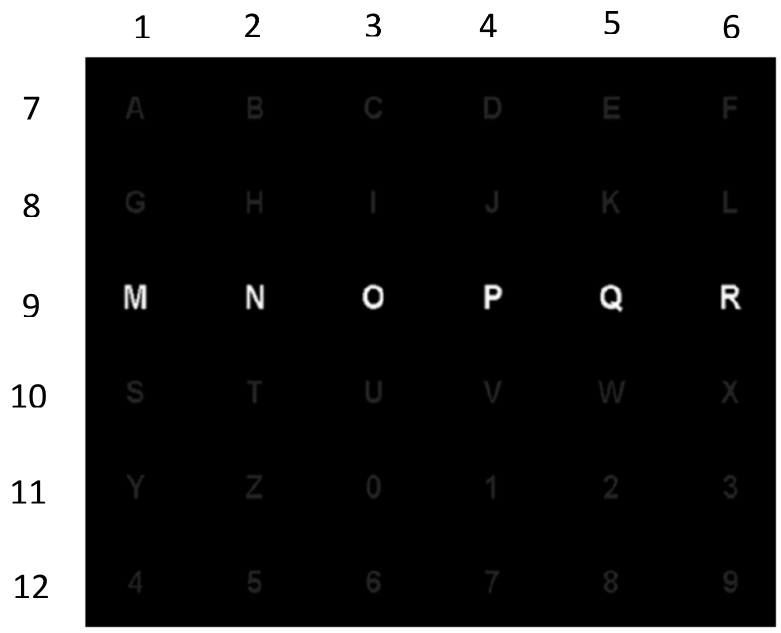
5 Word Recognition

- In the 10th row of data set, there are some non-zero numbers which are mapped into A-Z and 0-9 system. Since in the SC paradigm there are 36 characters that need to be shown but in the RC paradigm there are 6 rows and 6 columns and so 12 values. So, if the maximum value of the 10th row of each dataset is more than 12, the experiment is done based on SC paradigm and if this maximum is 12, then it is done based on RC paradigm.
  By the above method, we infer that Subject1 and Subject2 are experimented based on the SC paradigm, while other Subjects are experimented using RC paradigm.

- Since the train word is 'LUKAS', we can infer the mapping in each paradigm using the ones in the last row of train data based on the paradigm. Using one of the subjects which is trained with SC paradigm, we observe that the mapping system is like the below picture:



Using the same strategy for one of the RC based experiments, we conclude:



- **IndexExtraction.m**

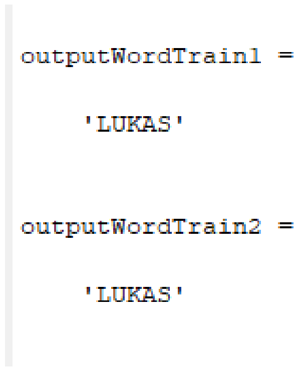
This function extracts the indices of target and non-target values of each stimuli and adds these fields to the general struct which includes all the structs for each subject. This function is much useful for epoching the Train data into Target and Non-Target epochs.

- To start the learning algorithm, we first preprocess data; removing possible noises by using bandpass filters and also we epoch our data into Target and Non-Target epochs for further uses. Also it is good to down-sample the data by 4; since each stimuli takes 4 indices of the 10th row of train and test data.

Implementing Machine Learning Algorithm

- To train our model, we use LDA and SVM learning algorithms with built-in matlab functions fitcsvm and fitcdiscr. To achieve this purpose, we need a train feature matrix; to do so, we use the time-series signals of each channel for target and non-target values and create the featureMatrix, then we create labels including target and non-targets. At the end we give the feature matrix and the labels to the `fitcdiscr & fitcsvm` and get the classification models in output and give it to predict with a feature and get the output labels.

- If we give trainFeatures to the predict function, we except to get the right word in output and this will happens too if we see the code output:



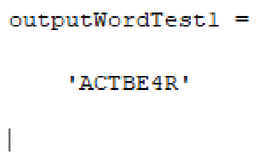
```
outputWordTrain1 =

    'LUKAS'


outputWordTrain2 =

    'LUKAS'
```

With both fitcdiscr & fitcsvm

- Because of the more non-target labels than target labels, we must give more weight for target values; since they are the ones we are looking for in general. So the cost function should consider more value for target labels than non-target ones as well.

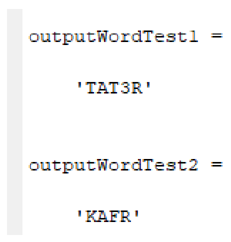
- **wordCreator.m** is the function for printing the recognized spelled word. It creates the result depend on a Method which gets in its input. This method can be 'SC' or 'RC' depend on the experiment. After that, it will get the samples saved in the 10th row of the data, which the samples are selected depend on the target and non-target status of each stimuli. Those numbers are something between 1 to 36 or 1 to 12 depend on the type of the experiment. For SC experiments we simply convert that number to its mapped letter or number, and for RC, the number shows the column or the row that is turned on in that specific stimuli. For RC experiments, we simply cross each column and row and get the character. (We saw the mappings in the previous parts.)

- Apply the trained models to all the objects:
    - Sub5:



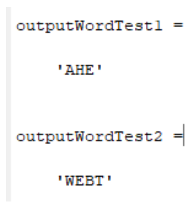
```
outputWordTest1 =

    'ACTBE4R'
```

    - Sub8:



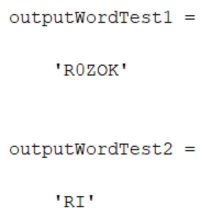
```
outputWordTest1 =

    'TAT3R'

outputWordTest2 =

    'KAFR'
```

    - Sub9:



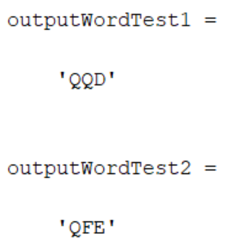
```
outputWordTest1 =

    'AHE'

outputWordTest2 =

    'WEBT'
```

    - Sub7:



```
outputWordTest1 =

    'R0ZOK'

outputWordTest2 =

    'RI'
```

    - Sub6:



```
outputWordTest1 =

    'QQD'

outputWordTest2 =

    'QFE'
```