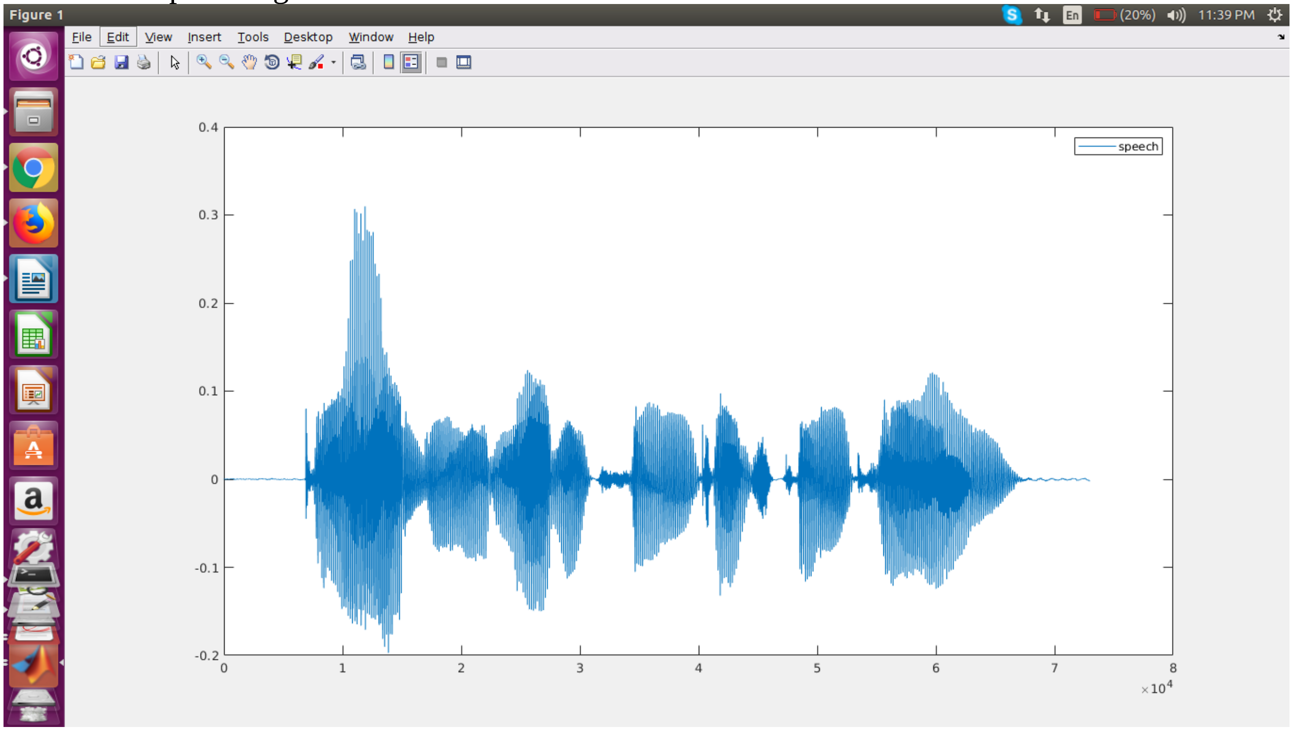
Assignment 3 of Speech Signal Processing
Naila Fatima
201530154

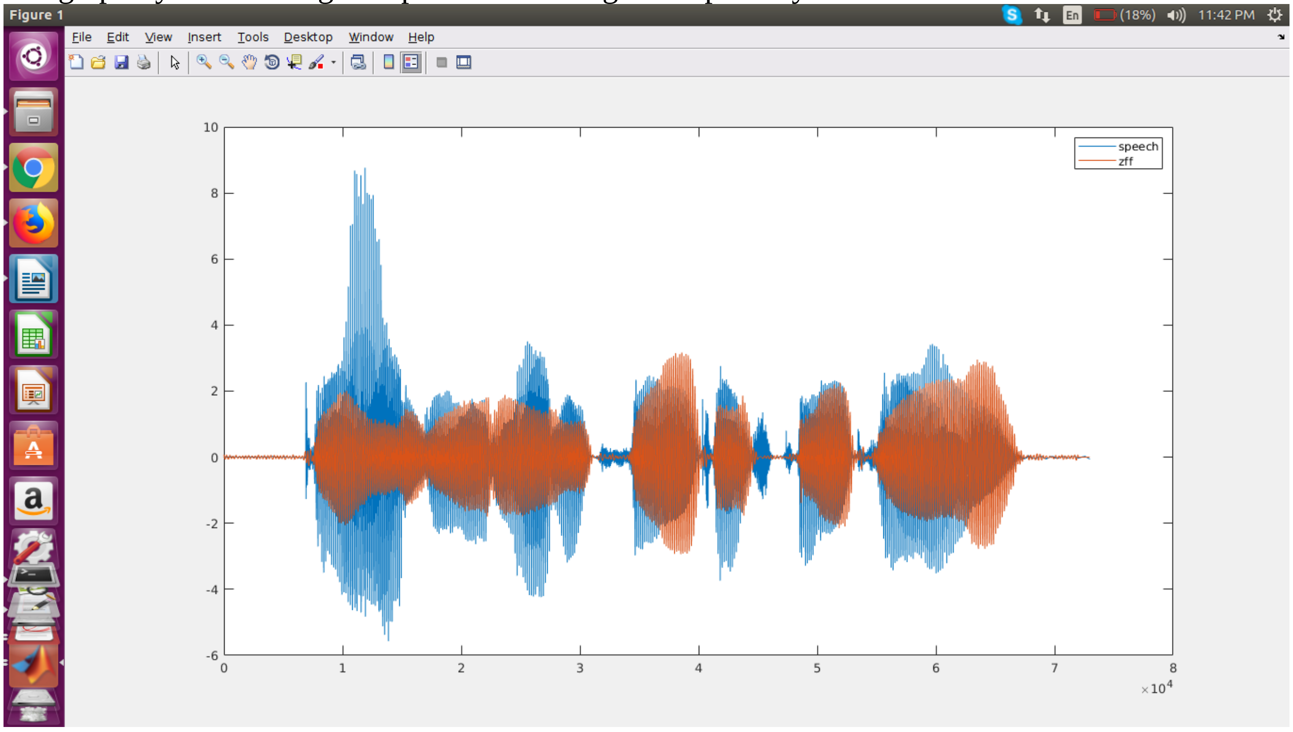
Question 1

a) All experiments have been done on the '*arctic_a0008.wav*' file. The matlab file is q1a.m for this question.

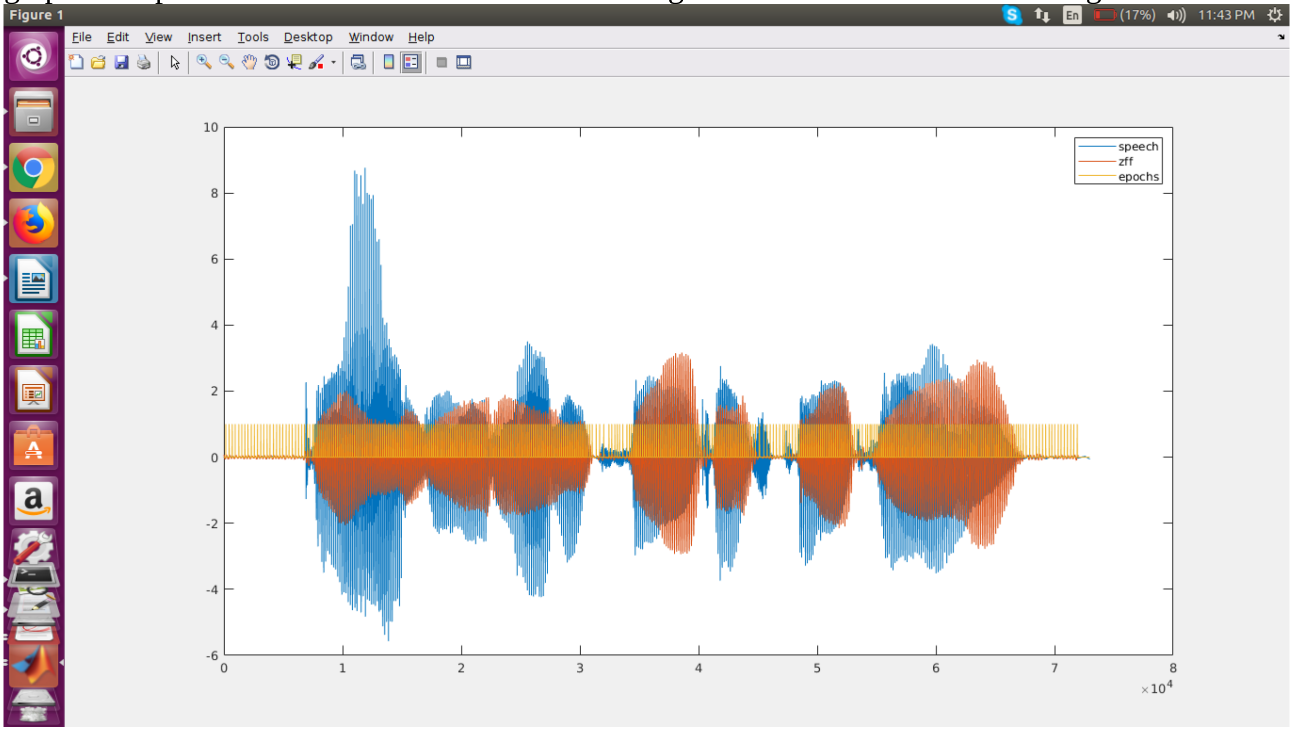
The speech signal is shown below.



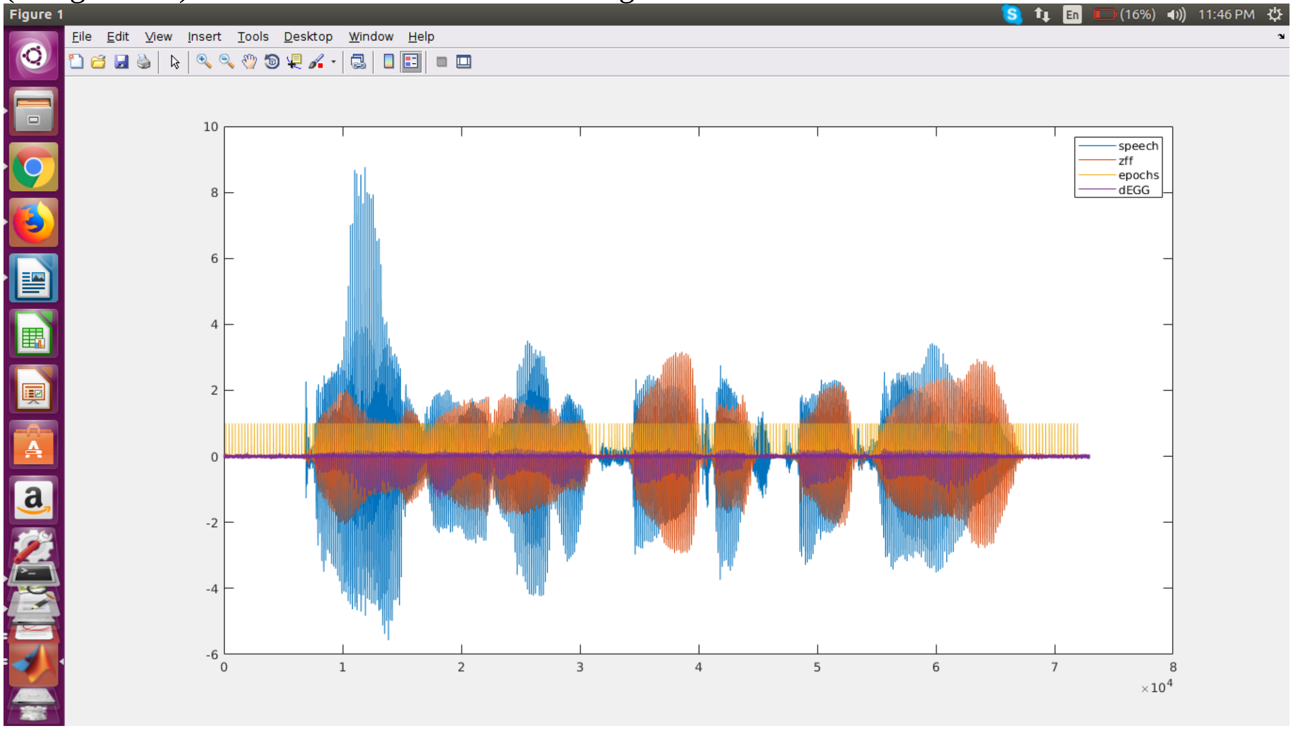
The ZFF signal (which is the output of the ZFF filter) is shown below. Note that I plotted this graph by normalizing the speech and zff signals separately.



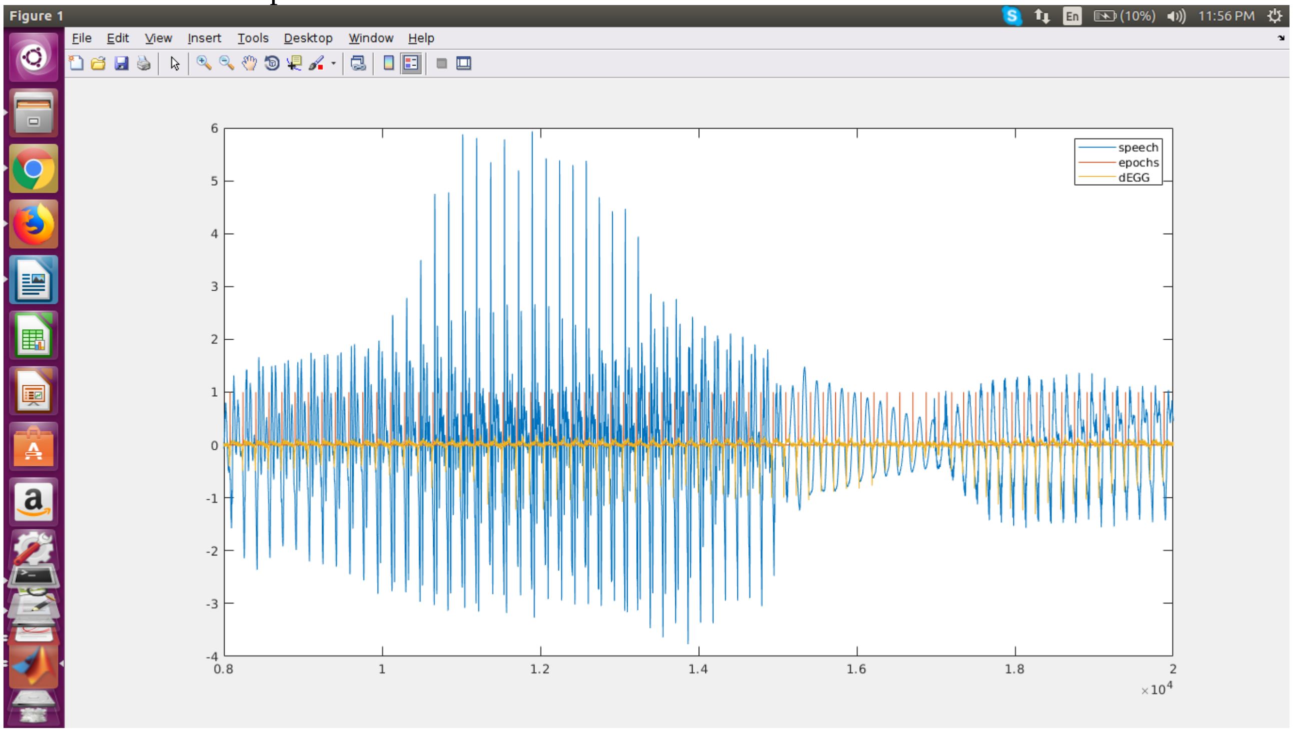
The epochs extracted are shown in the graph below. Note that epochs(i) is 1 if there is an epoch present at that location and it is 0 if there is no epoch at that location. We can see in this graph that epochs are more uniform in the voiced regions and less in the unvoiced regions.



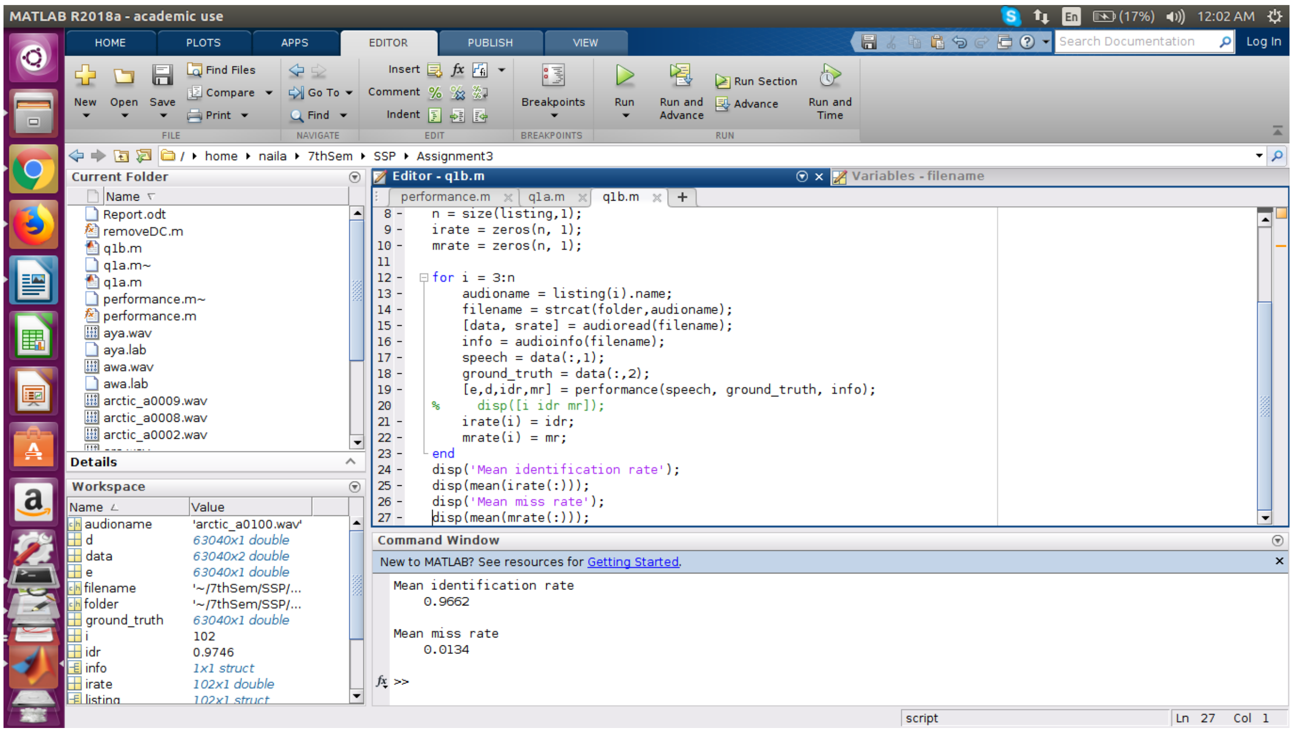
The dEGG signal is shown in the graph below. Note that I had to plot 30*dEGG as the values of the original dEGG signal were shown as a straight line. In this graph we can observe that the dEGG values are more dense in the voiced regions which coincide with the epoch locations (straight lines) and less dense in the unvoiced regions.



We can clearly see in the below picture that the epoch locations coincide with the dEGG values. The picture below is a magnified view of the speech, epochs and dEGG signals between 8000 and 20000 samples.



b) The matlab file for this question is q1b.m which uses the function performance.m (both included in folder). This file finds out the identification rate and miss rate of the zff filter for all the wav files present in the dataset given. Note that this gives us the mean of the identification rate and miss rates for all the wav files. In order to calculate the identification rate, the performance.m function finds out how many larynx cycles have their epochs detected correctly. Note that one larynx cycle is defined as 1/100 of a second (standard definition). To calculate the miss rate, it finds how many larynx cycles have no epochs detected at all. The mean identification rate is 96.62% while the mean miss rate is 1.34%.



Current Folder:

```
Report.odt
removeDC.m
q1b.m
q1a.m~
q1a.m
performance.m~
performance.m
aya.wav
aya.lab
awa.wav
awa.lab
arctic_a0009.wav
arctic_a0008.wav
arctic_a0002.wav
```

Workspace:

Name	Value
audioname	'arctic_a0100.wav'
d	63040x1 double
data	63040x2 double
e	63040x1 double
filename	'~/7thSem/SSP/...
folder	'~/7thSem/SSP/...
ground_truth	63040x1 double
i	102
idr	0.9746
info	1x1 struct
irate	102x1 double
listing	102x1 struct

```
n = size(listing,1);
irate = zeros(n, 1);
mrate = zeros(n, 1);

for i = 3:n
    audioname = listing(i).name;
    filename = strcat(folder,audioname);
    [data, srate] = audioread(filename);
    info = audioinfo(filename);
    speech = data(:,1);
    ground_truth = data(:,2);
    [e,d,idr,mr] = performance(speech, ground_truth, info);
%     disp([i idr mr]);
    irate(i) = idr;
    mrate(i) = mr;
end
disp('Mean identification rate');
disp(mean(irate(:)));
disp('Mean miss rate');
disp(mean(mrate(:)));
```

New to MATLAB? See resources for Getting Started.

```
Mean identification rate
    0.9662

Mean miss rate
    0.0134

>>
```

<u>Question 2</u>

a) Epochs refer to the significant excitation of the vocal tract system during the production of speech. Epochs are also known as GCI (glottal closure instant) and instant of significant excitation. Epochs are relevant in speech processing as the successive difference between epochs is known as pitch period. Since the reciprocal of pitch period gives us the pitch frequency or fundamental frequency, the knowledge of epochs allows us to know the frequency at which the vocal cords are vibrating.

Epoch extraction allows us to perform various applications which is why epochs are significant in speech processing. Epoch extraction enables us to calculate the pitch of speech, distinguish between voiced and unvoiced speech and do various speech enhancement techniques. We know that the difference between successive epochs is the pitch period which allows us to find the pitch of a speech as the pitch frequency is the inverse of the pitch period. In short,

fundamental frequency = 1/(difference in successive epochs)

Epoch extraction also enables us to distinguish between voiced and unvoiced speech as voiced speech tends to have uniform epochs (because the vocal cords undergo quasistatic vibrations) whereas unvoiced speech tends to have more random epochs.

Epochs also allow us to create algorithms in order to do prosody manipulation and estimate time-delay between speech signals collected over spatially distributed microphones.

b) There are several different methods used for extracting epochs but most of these algorithms are only used to do extract epochs from voiced speech. Zero frequency filtering, Hilbert envelope-based method, DYPSA algorithm and group delay method are some of the methods available for epoch extraction.

<u>Zero Frequency filtering:</u> One of the most accurate methods used is called zero frequency filtering which is extremely robust (upto 99% accuracy) and simple to implement. In this method, the signal is subjected to a differentiator. This is as the differentiator highlights the higher frequencies while suppressing the lower frequencies thereby making all frequencies nearly the same. This differentiated signal is then subjected to integration 4 times by using an ideal resonator two times. After this step, the resulting signal undergoes trend removal in which the mean of values in a window are subtracted from the signal. The positive zero crossings of the output signal align exactly with the epoch locations. The advantage of this method is that it can work accurately on noisy speech.

<u>LP analysis and Hilbert envelope:</u> Epochs can be extracted by using LP (linear predictive) analysis and Hilbert envelope. This is as LP analysis allows us to predict current samples of speech by using a linear weighted sum of previous samples. The formula for LP analysis is shown below:

$\hat{s}(n) = \Sigma a_k s(n-k)$

The residual/error of LP analysis is defined as the difference between the actual signal and the predicted one. In the below formula, e(n) denotes the residual of LP analysis.

$e(n) = s(n) – \hat{s}(n)$

We should note that the error will be higher in places where the rate of change of the original signal is more. We know that such places occur around the epoch locations and therefore, when the error has maximum change, we can label that location as an epoch. We should however note that there may be spurious peaks which is why we instead use the Hilbert envelope of the LP residual. We should note that the Hilbert envelope of a unit sample sequence or its derivative has a peak at that instant and this property is used to extract epochs from the Hilbert envelope of the LP residual. A differenced gaussian pulse is convolved with the Hilbert envelope of the LP residual and the instants of positive zero crossings in this convolved signal are the epoch locations. We should note that the differenced gaussian pulse has the following formula:

$$g[n] = \frac{(n-N/2)}{\sigma*\text{root}(2\Pi)} \exp\left(- \frac{(n-N/2)^2}{2\sigma^2} \right)$$ where n = 1 to N and σ is the standard deviation.

Group-delay based method: In this method, the LP residual is calculated and the average slope of the unwrapped phase of the STFT (short-time Fourier transform) of the LP residual is computed. The computed signal is known as the phase-slope function. Instants of positive zero crossings of the phase slope function are epochs.

DYPSA algorithm: The DYPSA algorithm is a dynamic programming method used to extract the epochs from voiced speech. Candidate epochs are generated from the phase-slope function (which is computed by using the group-delay method). A phase-slope projection is used to find candidates which are not zero crossings of the phase slope function. A cost function is minimized by using dynamic programming in order to find the true epochs from this set of candidates (which are true and spurious epochs).

c) In order to create a zero frequency filter we use a differentiator, an integrator and trend removal. The differentiator converts the input signal s[n] into the differenced signal x[n] by using:

$$x[n] = s[n] – s[n-1]$$

This differenced signal x[n] is then passed through an ideal resonator 2 times which is equivalent to 4 successive integrations.

$y1[n] = \Sigma a_k y1[n-k] + x[n]$ where k=1,2 and a_1= -2 and a_2 = 1

$y2[n] = \Sigma a_k y2[n-k] + y1[n]$ where k =1,2

The above integration process is also known as filtering at zero frequency. The next process is trend removal where we subtract the average of the y2 signal over an interval from y2 itself. We can take any interval length between 5-15 ms. The number of samples N in the interval will be equivalent to the product of the duration of the interval and the sampling rate. The formula for this step is given by:

$y[n] = y2[n] – 1/(2N+1)\ \Sigma y2[n+m]$ where m ranges from -N to N.

The signal y[n] is known as the zero frequency filtered signal. Note that the differentiator is highlighting the high frequencies and removing any low frequency bias.

We choose the zero frequency so that the characteristics of the time-varying vocal tract system do not affect the characteristics of the filter output. It should be noted that the vocal tract system has almost no influence on any sound in the frequencies between 0-4 Hz. We pass the speech signal through the zero frequency filter 2 times (this is equal to 4 integrations) to remove any high frequency resonances. We use 0 Hz resonator to separate the vocal tract system which resonates at higher frequencies.

Question 3

a) Nasals and approximants are particular types of speech sounds in phonetics. Nasals are speech sounds which are produced with a lowered velum (a membrane which separates the nasal and oral cavity), thereby allowing air to pass through the nose. Nasals include the consonants ‘n’ and ‘m’ as in order to produce these sounds, we tend to block the air from escaping through the mouth causing it to escape through the nose. The oral cavity acts as a resonance chamber for the sound produced.

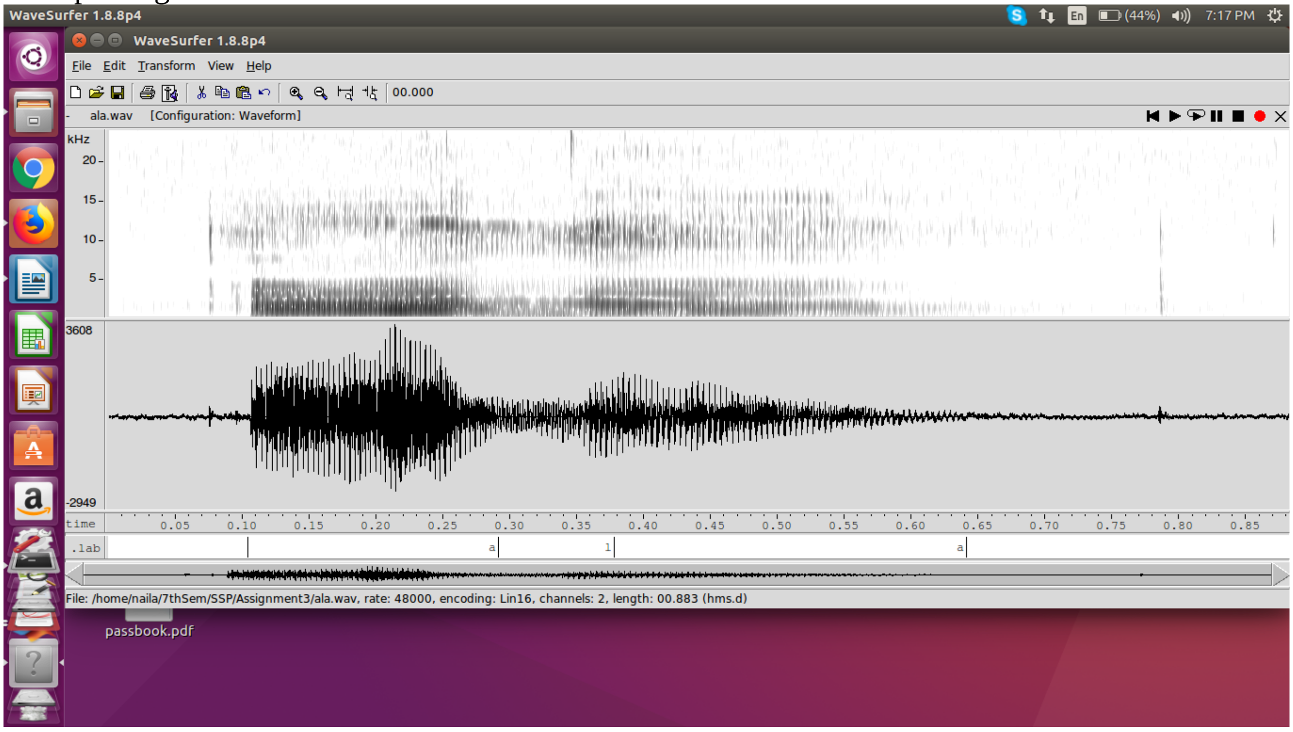
Approximants are speech sounds which require the articulators (lips, tongue, etc) to approach each other while still maintaining some separation in order to not cause any audible friction. Approximants include the phonemes ‘w’, ‘y’, ‘r’ and ‘l’. We can observe that in order to produce the sounds ‘w’ and ‘y’, we require lips to come together (but not touch) and in order to produce the sounds ‘l’ and ‘r’, we require the tongue to come close to the to top and bottom of our mouths, respectively.

The sounds ‘*ana*’, *ama*’ ‘*ara*’, ‘*ala*’, ‘*aya*’ and ‘*awa*’ are stored in the corresponding files: *ana.wav*, *ama.wav, ara.wav, ala.wav, aya.wav* and *awa.wav*. The transcriptions are in the files with corresponding names. These VCV sounds are different as they have different places and manners of articulation. If we are comparing the manner of articulation of the sounds, ‘n’ and ‘m’ are nasals while ‘y’, ‘r’, ‘l’ and ‘s’ are semivowels which are also approximants. If we compare the sounds by the place of articulation, ‘n’ and ‘l’ are dental, ‘r’ is alveolar, ‘m’ is bilabial, ‘y’ is palatal and ‘w’ is labio-dental. These sounds are different as they require different positions of articulators and have different manners of articulation.

The formants for the above sounds are given in the table below.

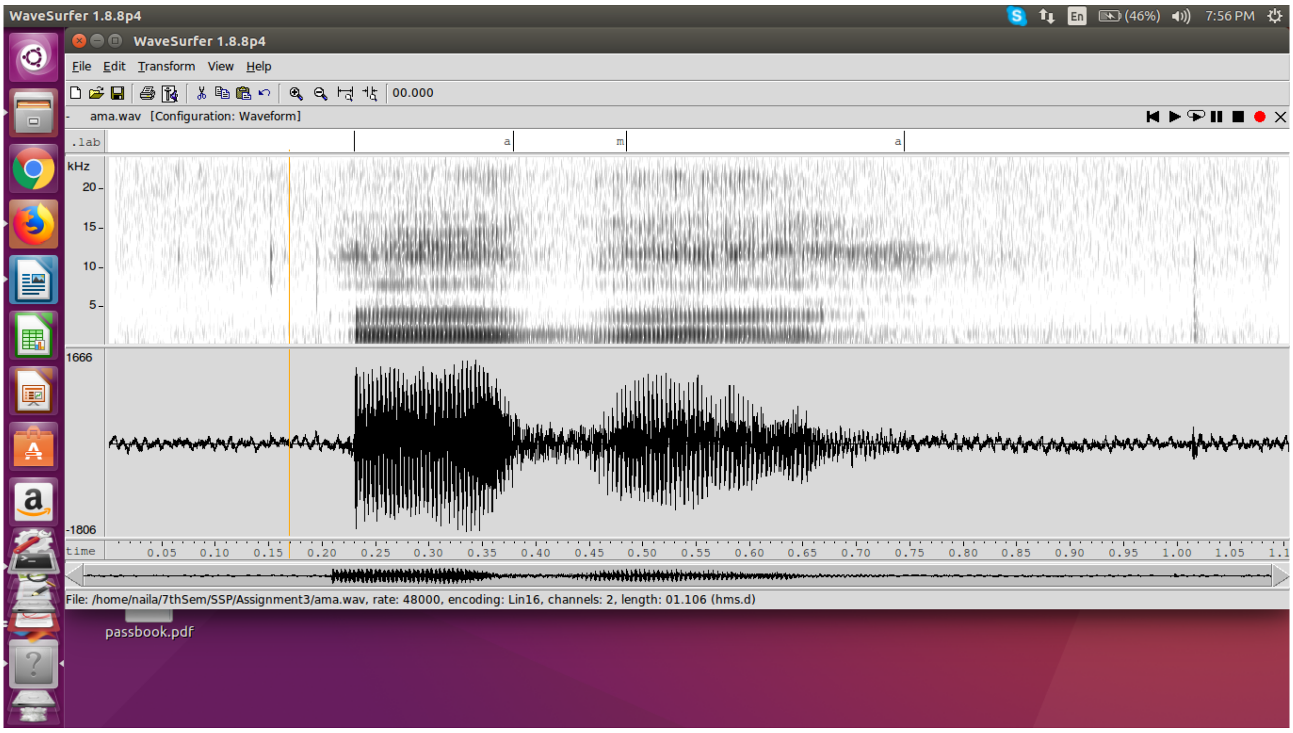
	F1	**F2**	**F3**
ana	493 Hz	1688 Hz	3425 Hz
ama	512 Hz	1288 Hz	3421 Hz
ara	626 Hz	1726 Hz	3220 Hz
ala	526 Hz	1755 Hz	3126 Hz
aya	770 Hz	2018 Hz	3198 Hz
awa	518 Hz	1075 Hz	2739 Hz

The spectrogram for ‘ala’ is shown below.



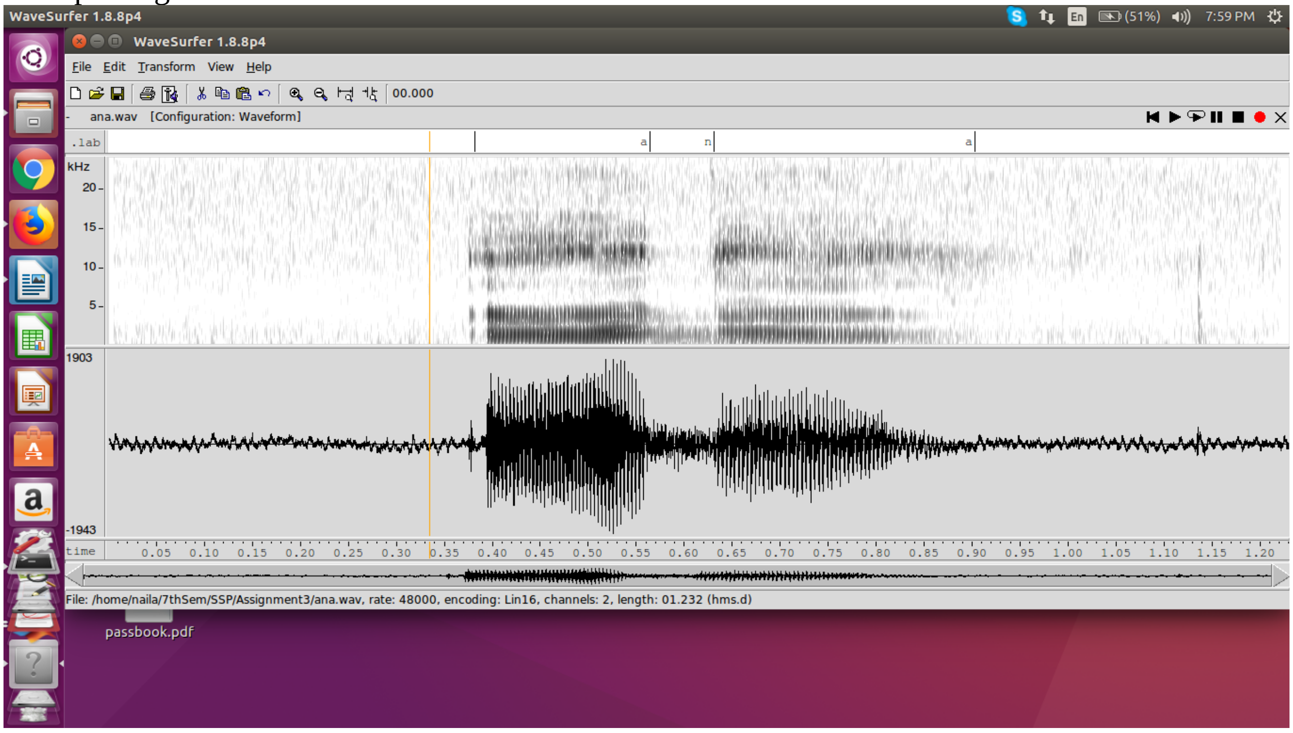
File: /home/naila/7thSem/SSP/Assignment3/ala.wav, rate: 48000, encoding: Lin16, channels: 2, length: 00.883 (hms.d)

The spectrogram for ‘ama’ is shown below.


WaveSurfer 1.8.8p4
File Edit Transform View Help
00.000
ama.wav [Configuration: Waveform]
.lab
a
m
a
kHz
time
passbook.pdf

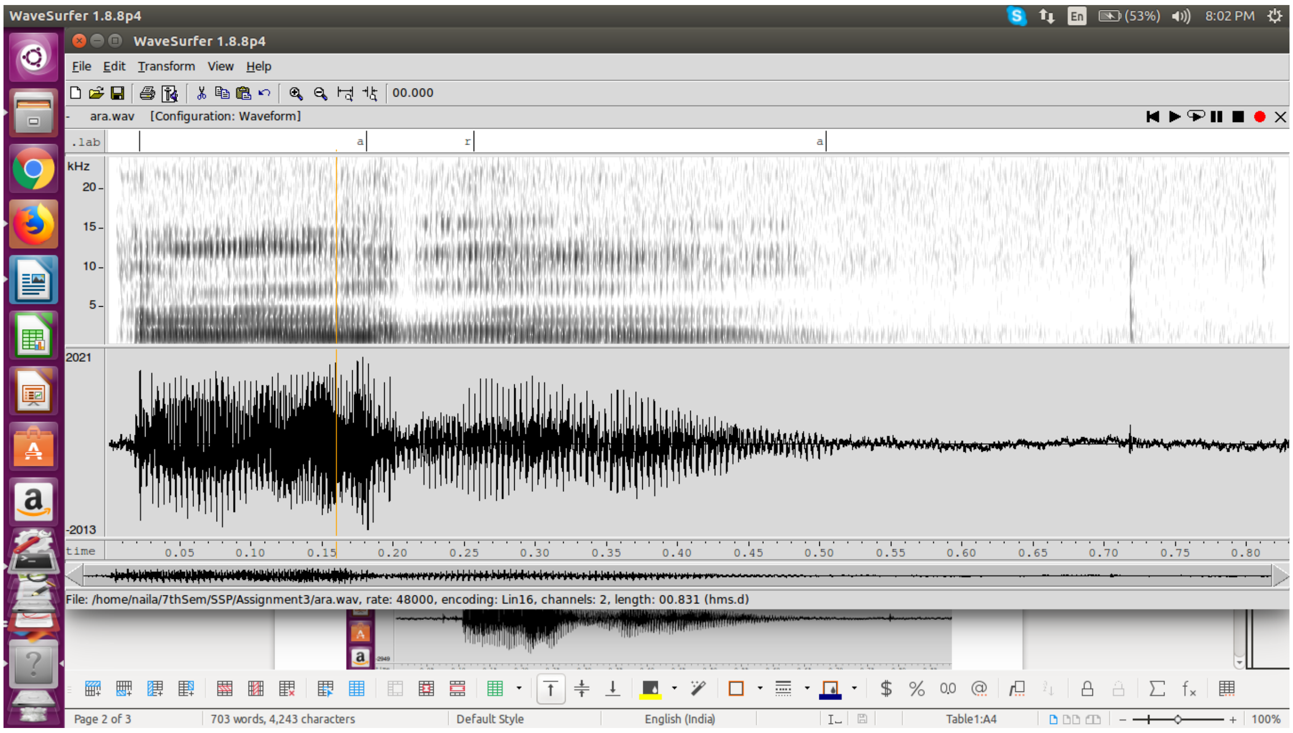
File: /home/naila/7thSem/SSP/Assignment3/ama.wav, rate: 48000, encoding: Lin16, channels: 2, length: 01.106 (hms.d)

The spectrogram for ‘ana’ is shown below.


WaveSurfer 1.8.8p4
File Edit Transform View Help
00.000
ana.wav [Configuration: Waveform]
.lab
a
n
a
kHz
time
passbook.pdf

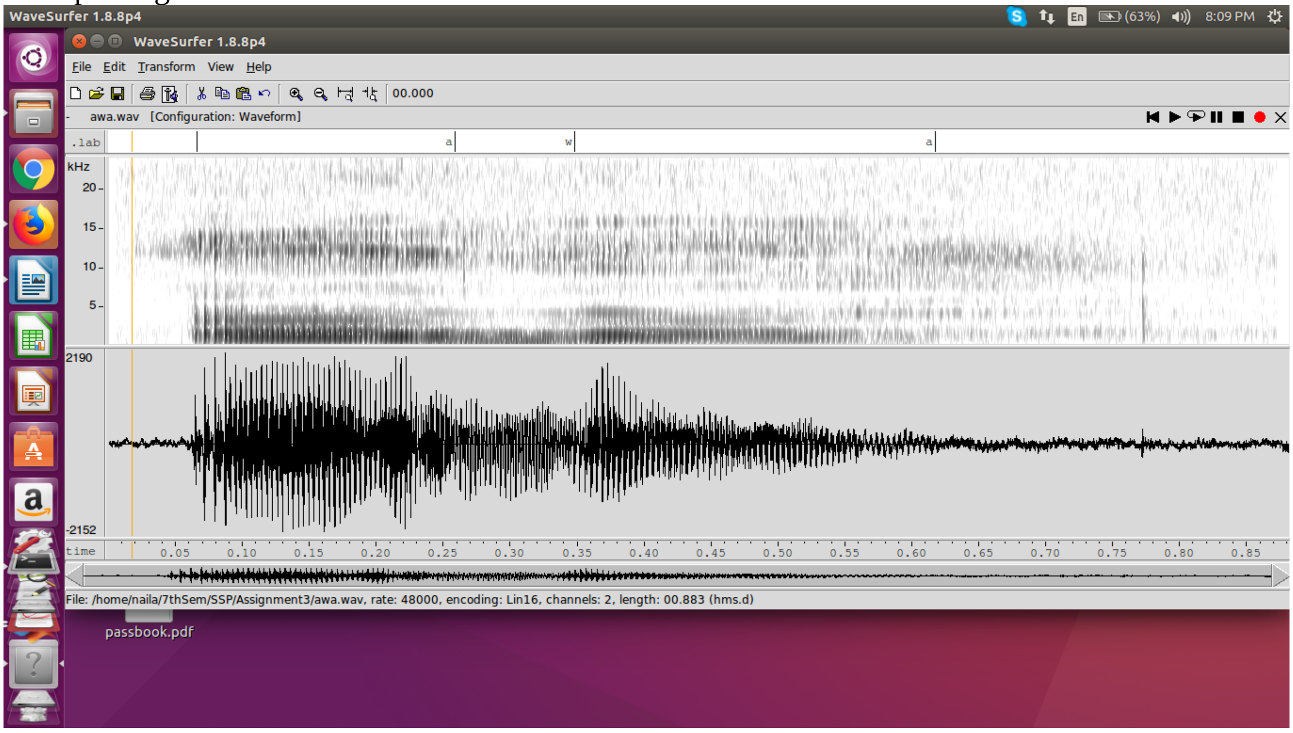
File: /home/naila/7thSem/SSP/Assignment3/ana.wav, rate: 48000, encoding: Lin16, channels: 2, length: 01.232 (hms.d)

The spectrogram for ‘ara’ is shown below.



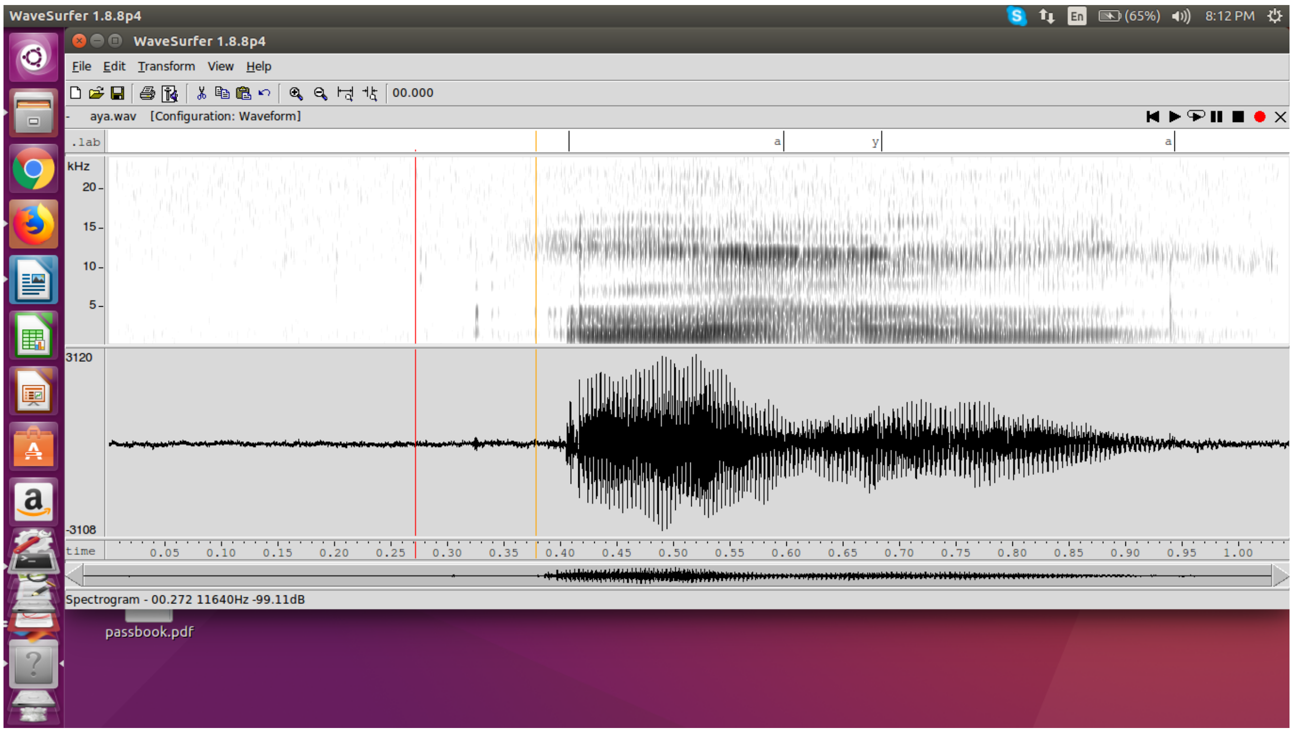
File: /home/naila/7thSem/SSP/Assignment3/ara.wav, rate: 48000, encoding: Lin16, channels: 2, length: 00.831 (hms.d)

The spectrogram for ‘awa’ is shown below.



File: /home/naila/7thSem/SSP/Assignment3/awa.wav, rate: 48000, encoding: Lin16, channels: 2, length: 00.883 (hms.d)

The spectrogram for ‘aya’ is shown below.



We know that if a spectrogram is dark, the formants are stronger. By observing the spectrograms, we can notice that ‘aya’ tends to have the darkest spectrogram indicating stronger values of the formants (which is indeed seen in the observation table as F1 and F2 are highest for ‘aya’). The spectrogram of ‘ara’ is dark thereby indicating that its formants are also strong. The spectrogram for ‘ana’ is the lightest thereby showing that it has weaker formants. We can observe that ‘ala’ and ‘ama’ have moderate spectrograms so their formants will also be moderate-levelled.

‘n’ and ‘m’ are different in the acoustics as ‘n’ is dental while ‘m’ is bilabial. We can observe that we require the tongue to press against the backs of the upper teeth to produce ‘n’ and our lips to press together to produce ‘m’. ‘n’ and ‘m’ are also different as they have different formants. F3 is approximately similar for both sounds but F1 is greater for ‘m’ while F2 is greater for ‘n’. By judging the spectrograms, we can also see that the spectrogram for ‘n’ is slightly lighter than that for ‘m’ showing that the formants may be slightly weaker. We can also see that bilabials like ‘m’, ‘b’ and ‘p’ tend to have lower F2 and F3 frequencies whereas dental sounds like ‘d’, ‘n’ and ‘s’ tend to have higher F2 and F3. The above statement is proved by our observation table.