Faculty of Engineering & Technology

Electrical & Computer Engineering Department

SPOKEN LANGUAGE PROCESSING: ENCS5344

SFS Assignment

Prepared by:

Mohammad Abu Shams 1200549

Mohammed Owda 1200089

Instructor: Dr. Abualseoud Hanani

Section: 1

Date: 1-5-2024

Birzeit

Table of Contents

List of Figures

List of Tables

Part 1 – Basic sound analysis, Spectrograms

(a) Bandwidth

Sound1.wav

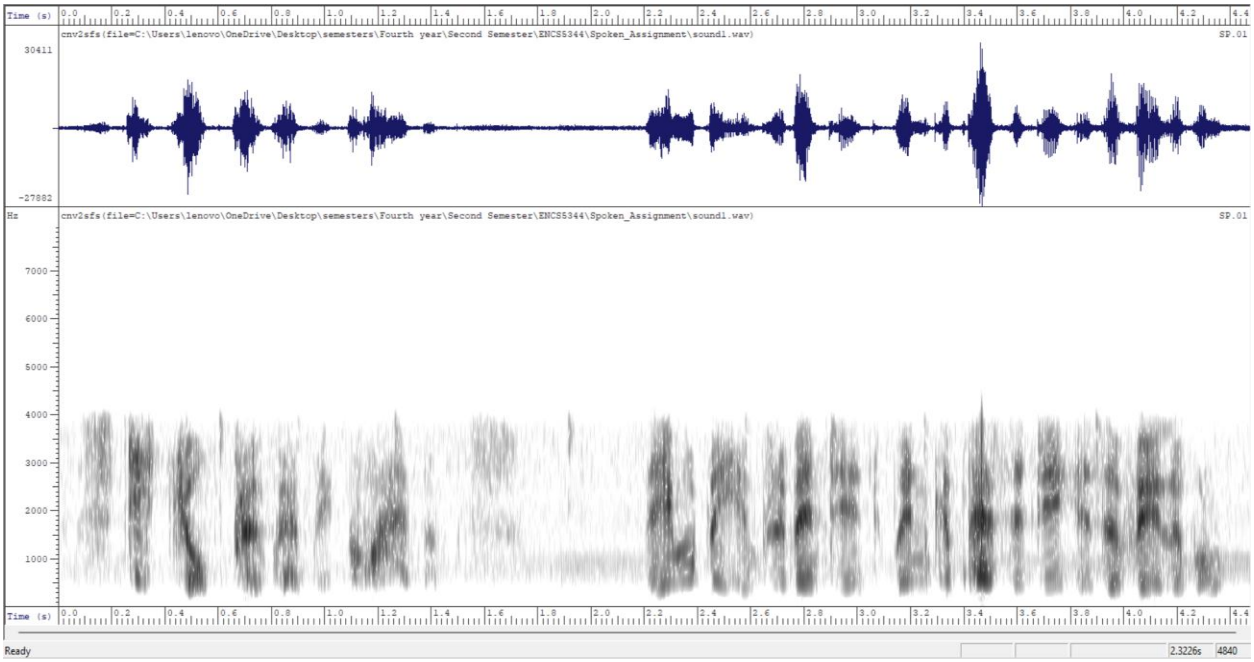


Figure 1: Waveform and wide-band spectrogram for sound1

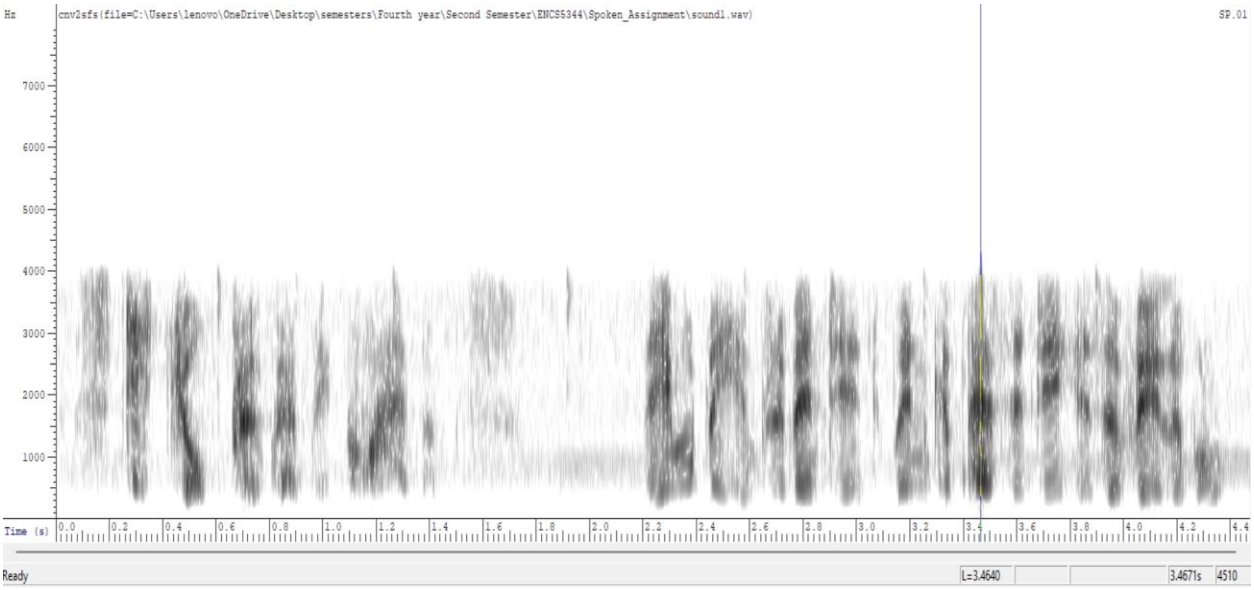


Figure 2: Upper frequency limit for sound1

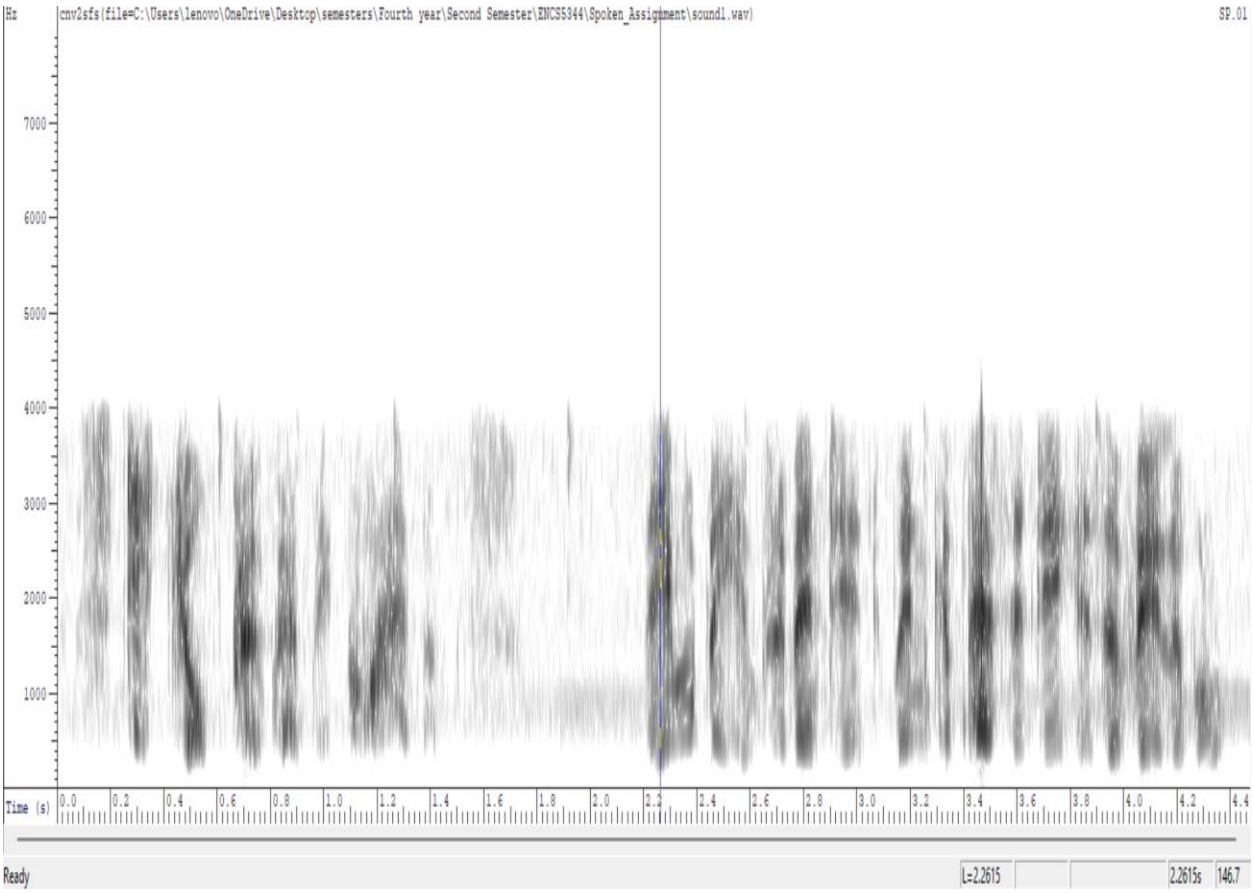


Figure 3: Lower frequency limit for sound1

From above figures for sound1, the upper frequency approximately equal 4510 Hz, and the lower frequency approximately equal 146.7 Hz.

The upper frequency (+/-500Hz) = 4510+500=5010Hz.

The lower frequency (+/-500Hz) = 146.7-500=-353.3Hz

Sound2.wav

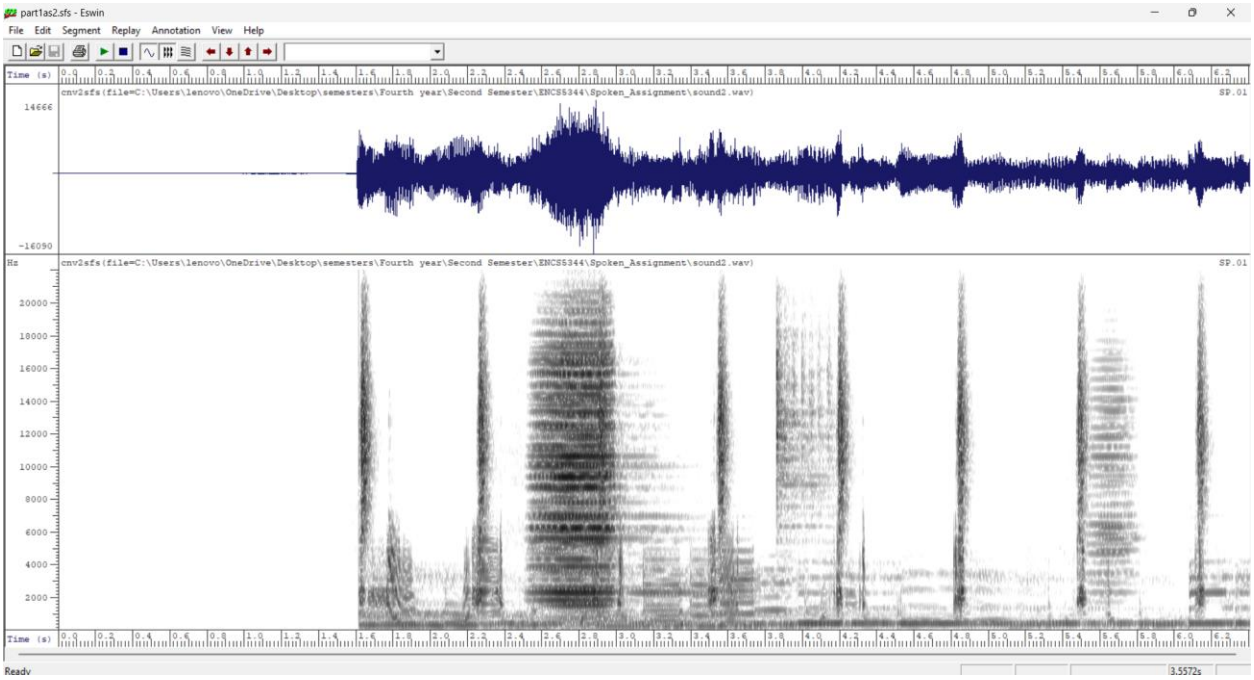


Figure 4: Waveform and wide-band spectrogram for sound2

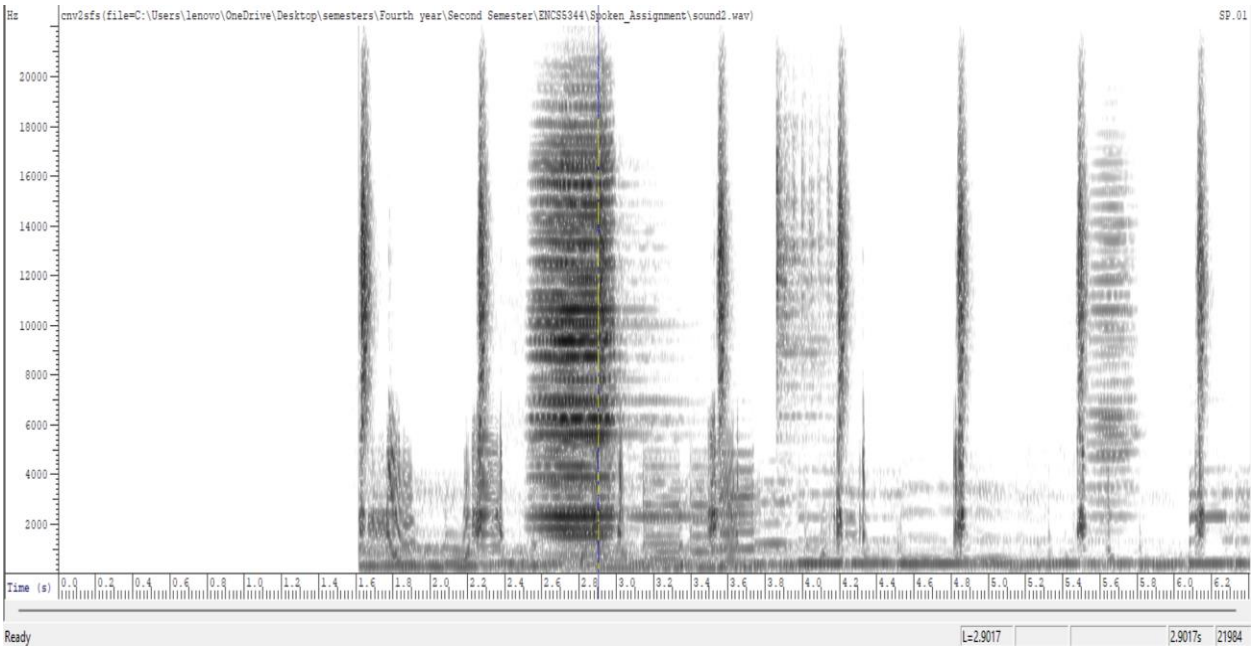


Figure 5: Upper frequency limit for sound2

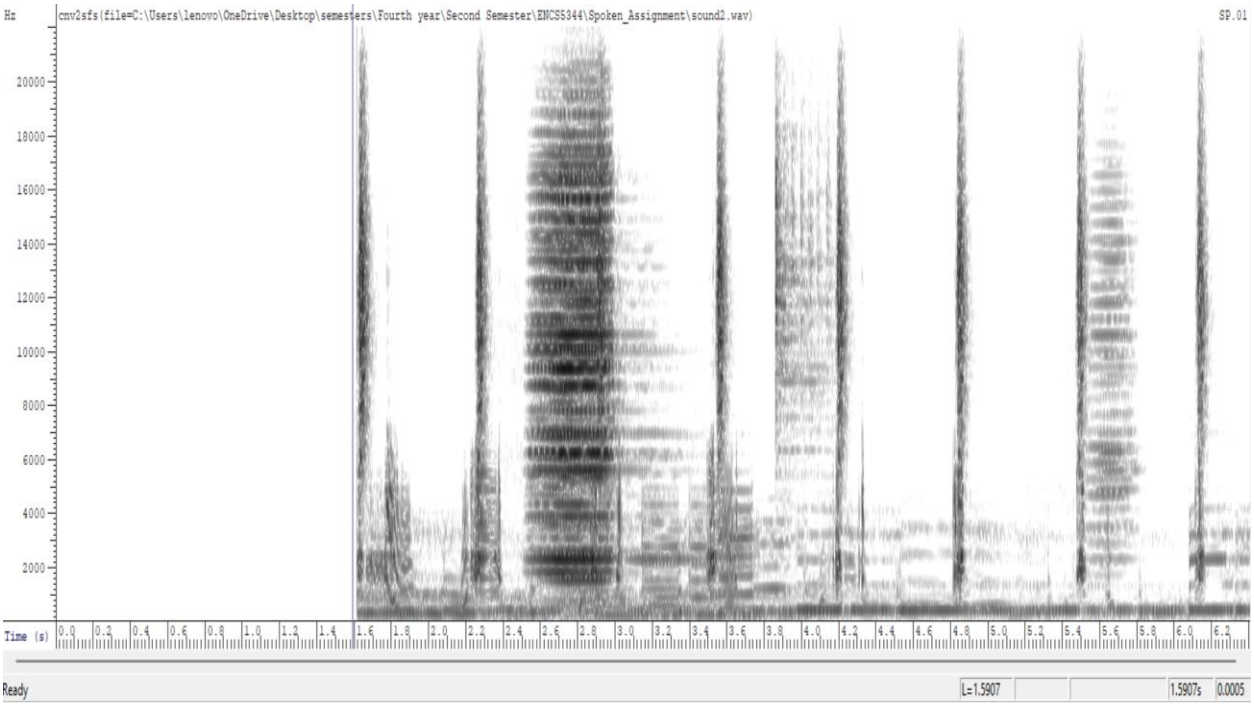


Figure 6: Lower frequency limit for sound2

From above figures for sound2, the upper frequency approximately equal 21984 Hz, and the lower frequency approximately equal 0 Hz.

The upper frequency (+/-500Hz) = 21984+500=22484Hz.

The lower frequency (+/-500Hz) = 0-500=-500Hz.

(b) Spectrogram

Sound3.wav

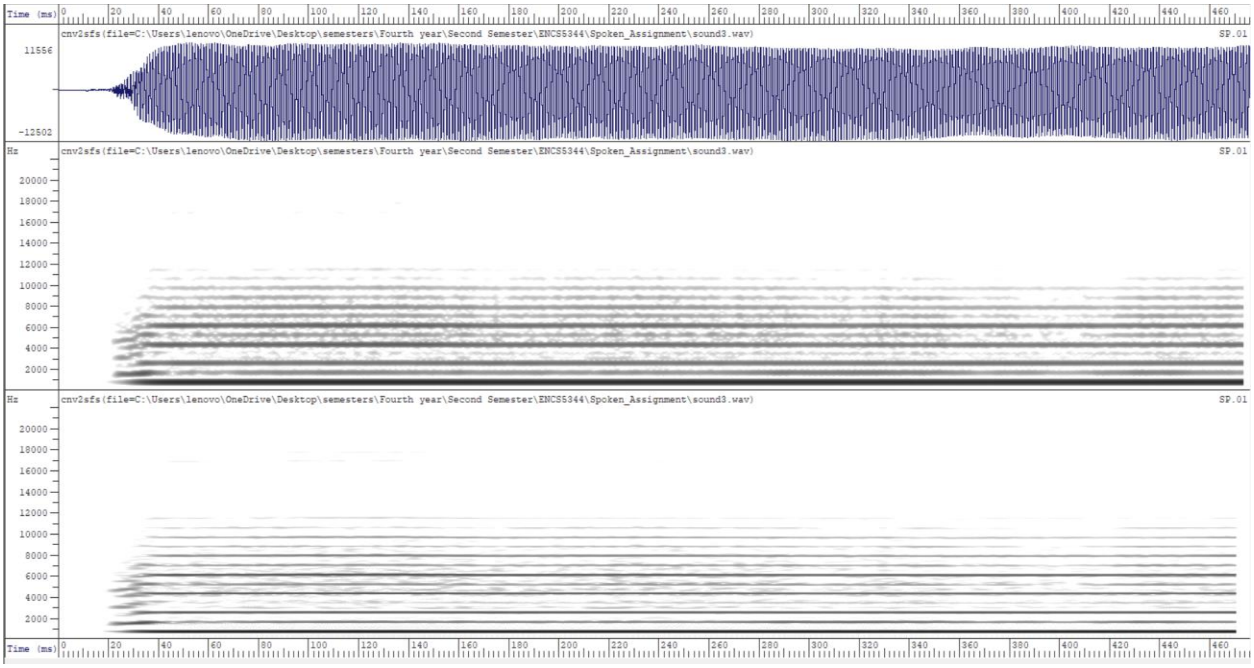


Figure 7: Waveform and both the narrow-band and wide-band spectrogram for sound3

The narrow-band spectrogram in the figure above displays high frequency resolution with clearly defined harmonics as horizontal lines, which indicates its capability to distinguish between closely spaced frequencies, useful for analyzing the spectral content of the sound. This is because narrow-band spectrograms use longer time windows for the Fourier transform, resulting in more detailed frequency information.

In contrast, the wide-band spectrogram shows lower frequency resolution with broader and more blurred frequency bands, sacrificing detail to provide a better view of how sound energy changes over time. This is due to the use of shorter time windows, which gives a better time resolution but less frequency detail, beneficial for analyzing temporal features like the cadence of speech.

Sound4.wav

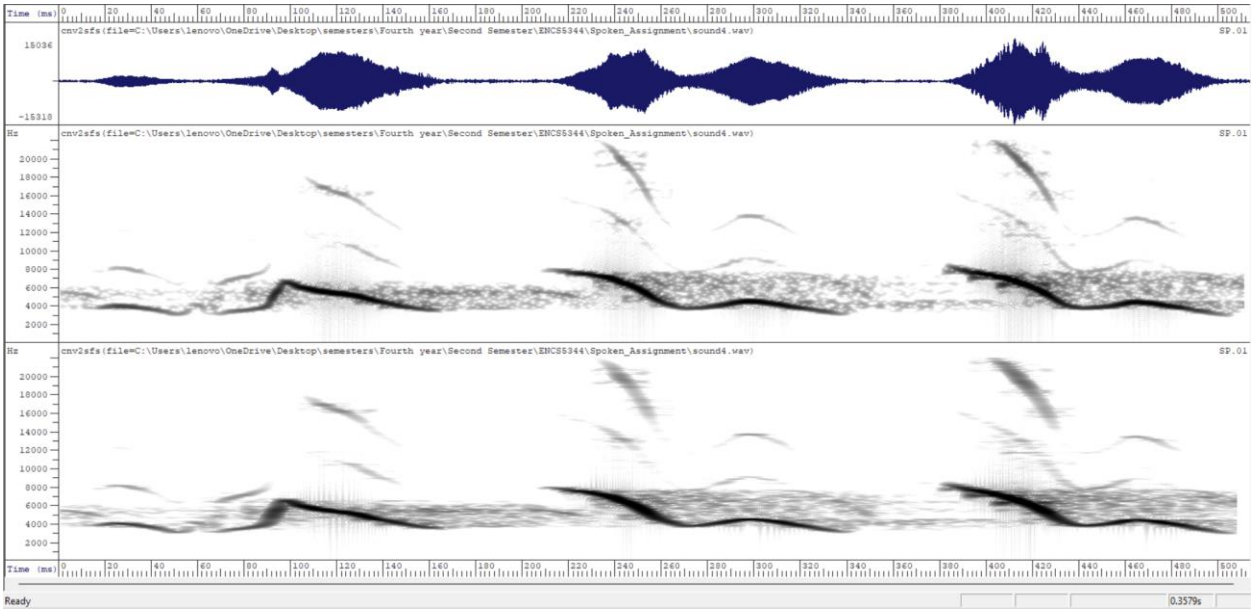


Figure 8: Waveform and both the narrow-band and wide-band spectrogram for sound4

The narrow-band spectrogram shows higher frequency resolution, evident from the distinct horizontal lines, which represent individual harmonics. This is due to a larger analysis window capturing finer frequency details.

Conversely, the wide-band spectrogram prioritizes time resolution, evident from the less distinct, smeared frequency bands, which are better for analyzing rapid changes over time. The wider bands in the wide-band spectrogram indicate a smaller window size in the analysis, trading off frequency clarity for temporal precision.

afa

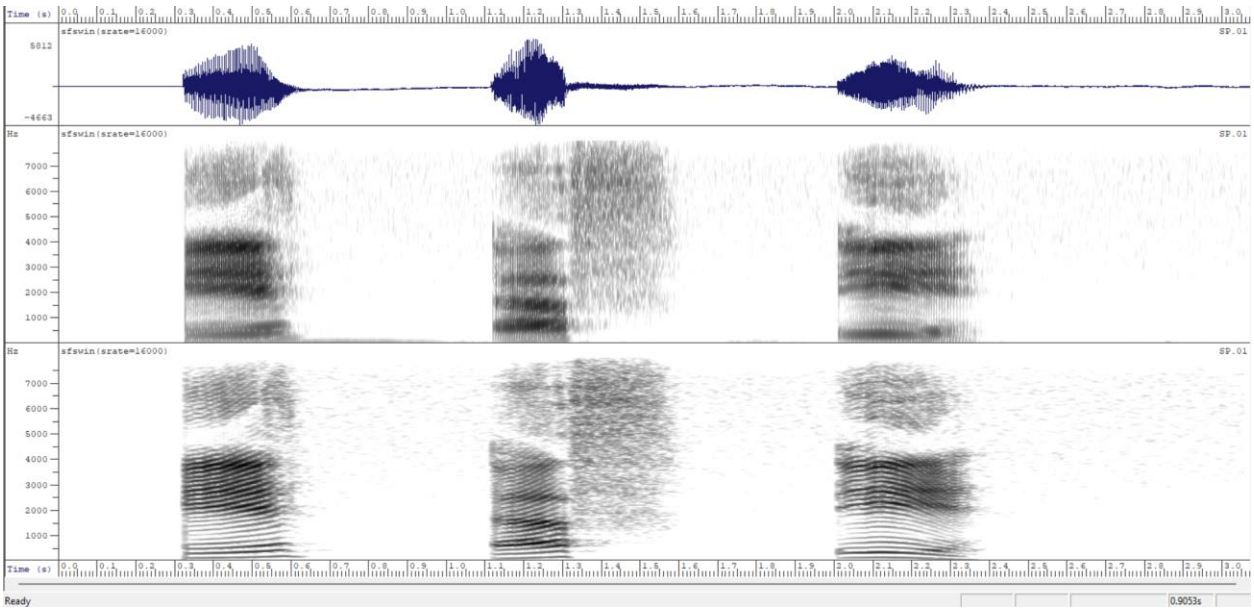


Figure 9: Waveform and both the narrow-band and wide-band spectrogram for the syllable 'afa'

In the sound wave, when people speak, the parts where they make vowel sounds have a regular up-and-down pattern because of their vocal cords vibrating. But when they make consonant sounds, there's more randomness and less of a pattern, like noisy sounds made without vocal cord vibrations.

In the narrow-band spectrogram, the vowel sounds show clear lines because they have a regular pattern of vibrations, like the fundamental note and its harmonics. But for consonant sounds, there aren't clear lines like that. Instead, there's a more scattered pattern across different frequencies, showing the noisy airflow.

This wav files was attached in the folder part1 wav.

Part 2 – Filters

We generate a sinusoidal signal with 1KHz at Fs=16KHz and duration of 3 seconds.

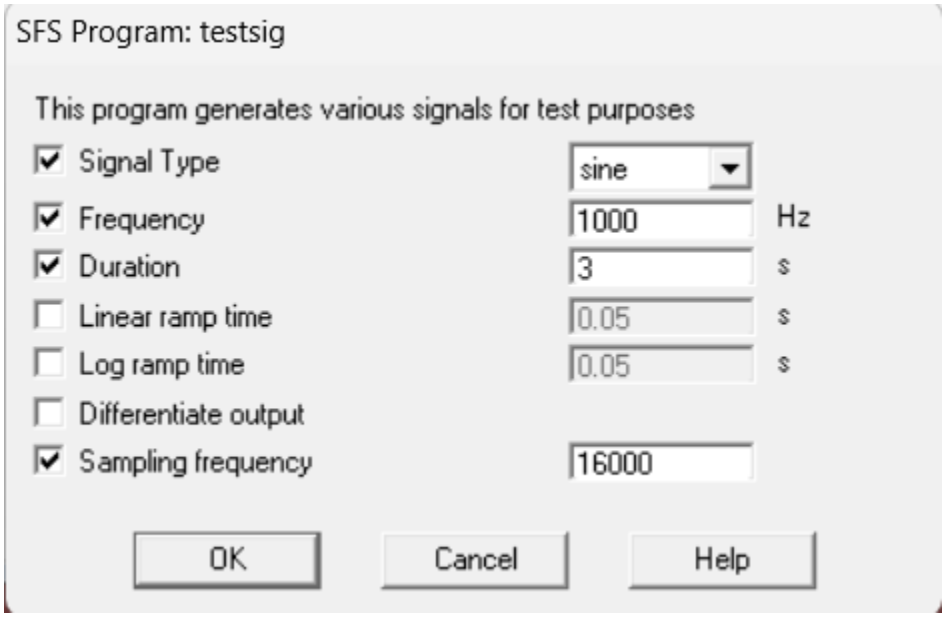


Figure 10: sinusoidal signal settings

We also generate a white noise signal at the same Fs and the same duration.

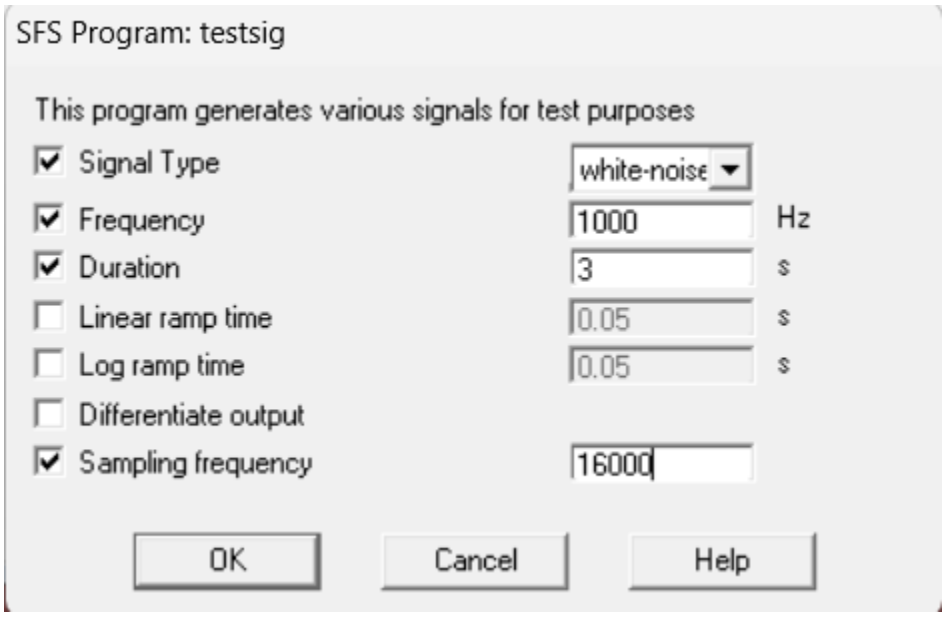


Figure 11: white noise signal settings

We add the two signals together into one signal

```
PART2FIRST_P.m
% Mohammad Abu Shams 1200549.
% Mohammed Owda 1200089.
% Section1.


% Sine wave signal
[sine_signal, Fs] = audioread('sine_signal.wav');

% White noise signal.
[white_noise, Fs_noise] = audioread('white_signal.wav');

% Combined two signals.
combined_signal = sine_signal + white_noise;

% Normalize the combined signal to prevent clipping.
combined_signal = combined_signal / max(abs(combined_signal));

% Write the combined signal.
audiowrite('combined_signal.wav', combined_signal, Fs);

Command Window
>> PART2FIRST_P
>>
```

Figure 12: Add two Signals together code

First Filter

We design and implement the first FIR Filter.

```
PART2SECOND_P.m
% Mohammad Abu Shams 1200549.
% Mohammed Owda 1200089.
% Section1.

% Load the combined signal.
[combined_signal, Fs] = audioread('combined_signal.wav');

% Define the FIR filter with impulse response {1, -1}.
b = [1, -1];  % Coefficients of the FIR filter.
a = 1;        % Denominator is simply 1.

% Apply the FIR filter to the combined signal.
filtered_signal = filter(b, a, combined_signal);

% Normalize the filtered signal to prevent clipping.
filtered_signal = filtered_signal / max(abs(filtered_signal));

% Write the filtered signal.
audiowrite('filtered_signal.wav', filtered_signal, Fs);

% Listen to the signals.
sound(combined_signal, Fs);
pause(length(combined_signal)/Fs + 1);  % Wait for the sound to finish.
sound(filtered_signal, Fs);

% Perform the FFT of the impulse response.
N = 256;  % Number of points in FFT
magFreqChar = abs(fft(b, N));  % Calculate the magnitude of the FFT.

% Plot the magnitude frequency characteristic.
figure;
plot(20*log10(magFreqChar(1:N/2)), 'r', 'LineWidth', 1);  % Convert magnitude to dB and plot for half the spectrum.
title('Magnitude Frequency Characteristic of FIR Filter');
xlabel('Frequency Index');
ylabel('Magnitude (dB)');
grid on;

Command Window
>> PART2SECOND_P
>>
```

Figure 13: First FIR Filter code

The waveforms of the input and output signals for the first filter:

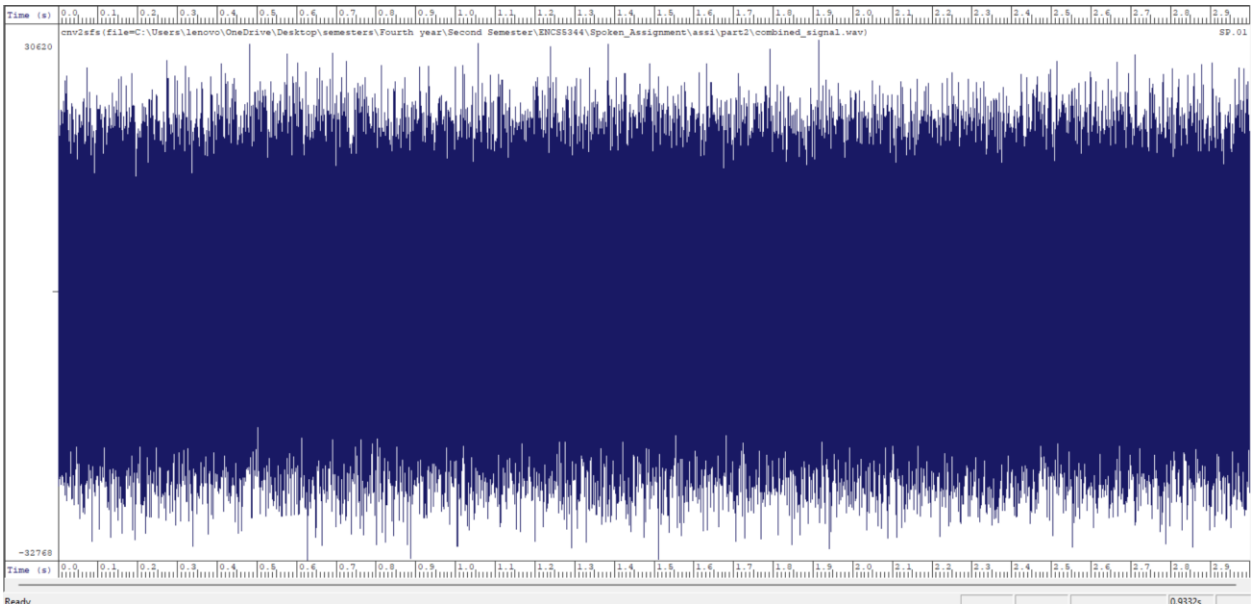


Figure 14: waveforms of the input signal for the first filter

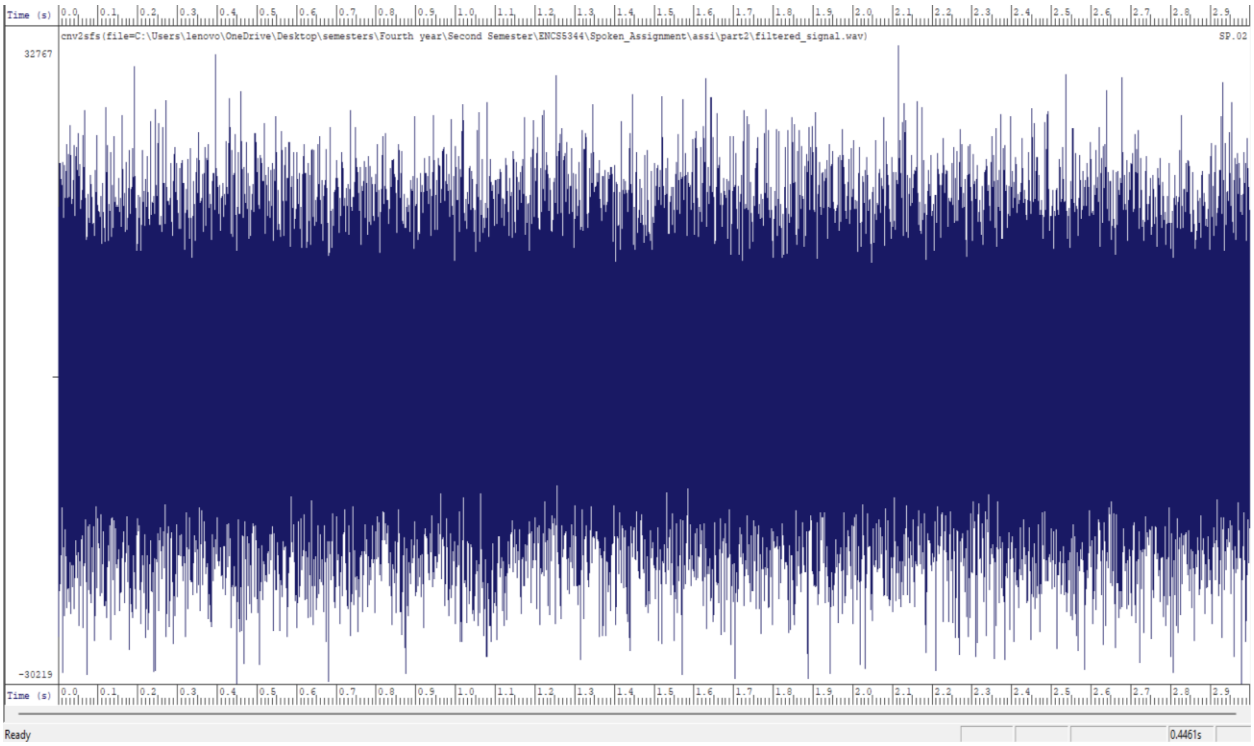


Figure 15: waveforms of the output signal for the first filter

The magnitude frequency characteristics of the first filter:

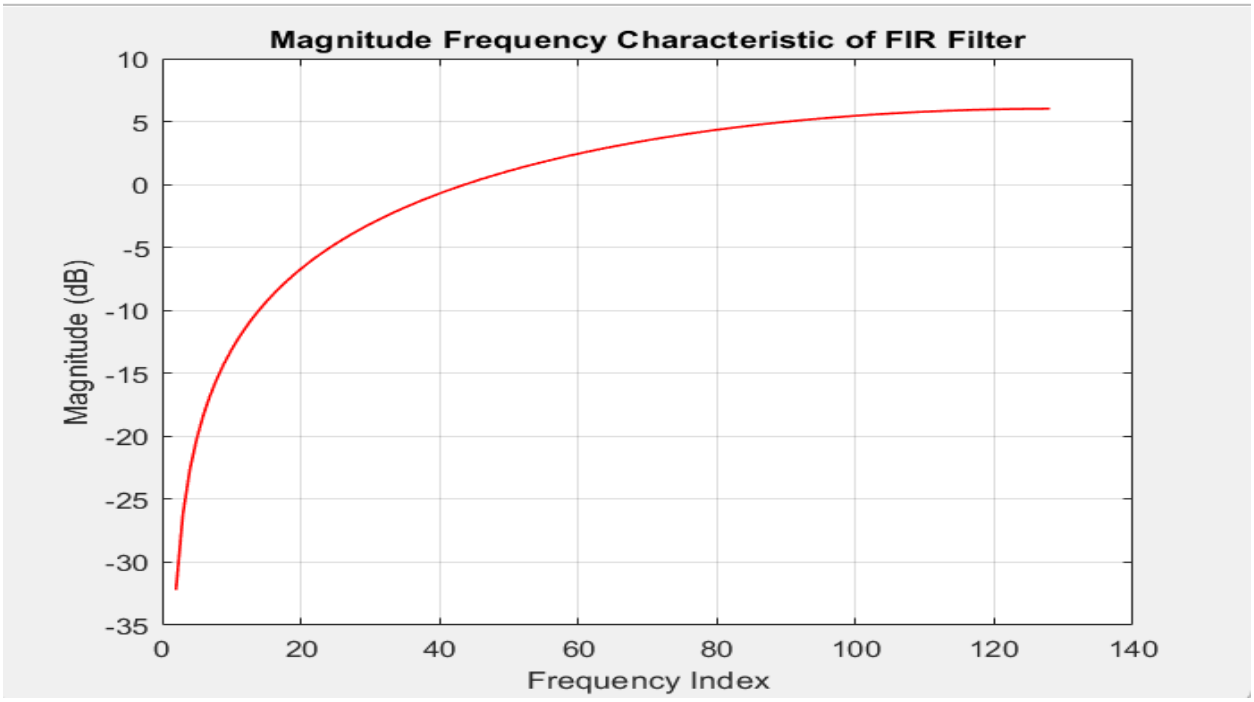


Figure 16: The magnitude frequency characteristics of the first filter.

The FIR filter with impulse response {1, -1} is a basic high-pass filter. This filter works by subtracting each sample from the one before it, which makes sudden changes in the signal more noticeable. This enhances high-frequency parts of the signal and reduces low-frequency parts.

In the output signal, high-frequency sounds like those found in higher-pitched tones will be more noticeable, while lower-frequency sounds (like bass tones) might be quieter. This filter can help reduce low-frequency noise, but high-frequency noise might become more noticeable.

Overall, the effect of this filter depends on the input signal's frequencies. High-pass filtering can improve clarity and sharpness, but it might also lead to a loss of bass tones and an increase in sharpness due to the focus on high-frequency noise.

Second Filter

We design and implement the Second FIR Filter.\

PART2THIRD_P.m

```
% Mohammad Abu Shams 1200549.
% Mohammed Owda 1200089.
% Section1.

% Load the combined signal.
[combined_signal, Fs] = audioread('combined_signal.wav');

% Define the filter coefficients.
b = 0.5 * ones(1, 8);  % Coefficients b0 to b7 set to 0.5.
a = 1;                 % Denominator is simply 1.

% Apply the filter to the combined signal.
filtered_signal2 = filter(b, a, combined_signal);

% Normalize the filtered signal to prevent clipping.
filtered_signal2 = filtered_signal2 / max(abs(filtered_signal2));

% Write the filtered signal.
audiowrite('filtered_signal2.wav', filtered_signal2, Fs);

% Listen to the signals.
sound(combined_signal, Fs);
pause(length(combined_signal)/Fs + 1);  % Wait for the sound to finish.
sound(filtered_signal, Fs);

% Perform the FFT of the impulse response.
N = 256;  % Number of points in FFT.
magFreqChar = abs(fft(b, N));  % Calculate the magnitude of the FFT.

% Plot the magnitude frequency characteristic.
figure;
plot(20 * log10(magFreqChar(1:N/2)),'g', 'LineWidth', 1);  % Convert magnitude to dB and plot for half the spectrum.
title('Magnitude Frequency Characteristic of 8-Tap FIR Filter');
xlabel('Frequency Index');
ylabel('Magnitude (dB)');
grid on;
```

Command Window

```
>> PART2THIRD_P
>>
```

Figure 17: Second Filter Code

The waveforms of the input and output signals for the first filter:

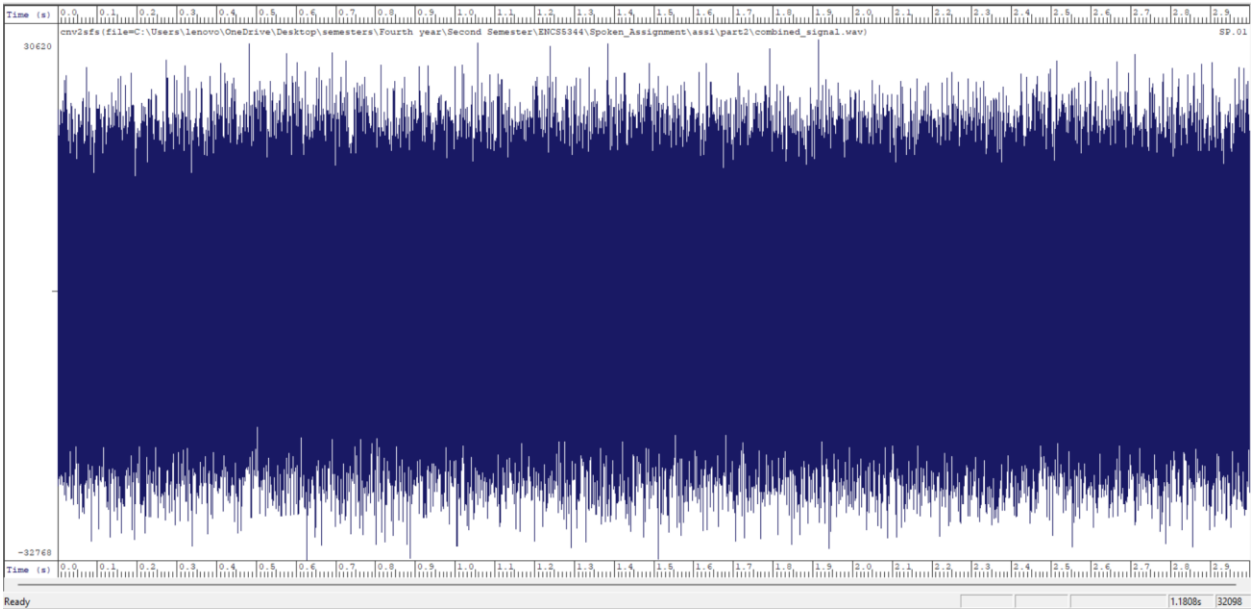


Figure 18: The waveforms of the input signals for the second filter

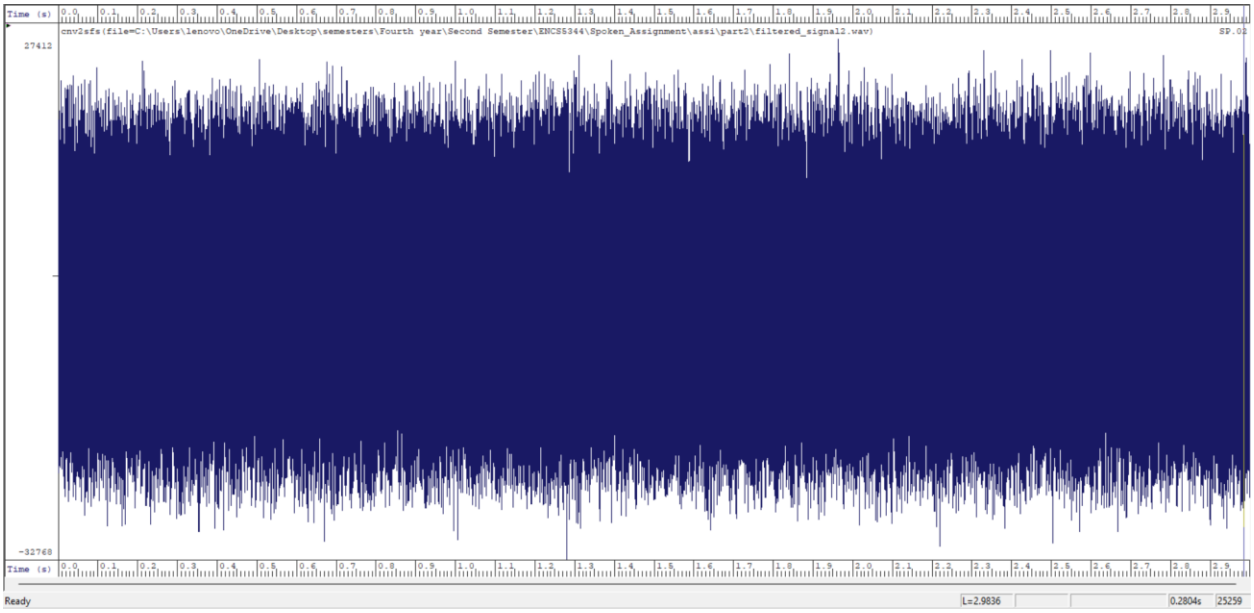


Figure 19: The waveforms of the output signals for the second filter

The magnitude frequency characteristics of the second filter:

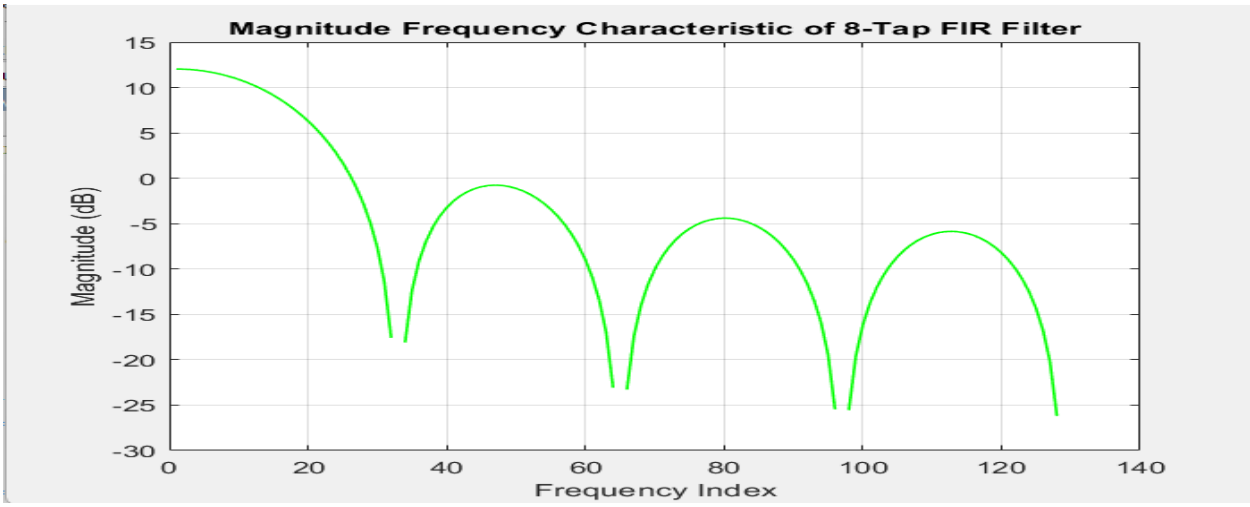


Figure 20: The magnitude frequency characteristics of the second filter

The 8-tap FIR filter with equal coefficients (0.5) acts like a moving average filter. It smooths out the input signal by averaging it over 8 samples. This filter helps reduce high-frequency noise, making the signal smoother.

High-frequency sounds like those in high-pitched tones may have lower volume in the output because the filter lowers high-frequency components. Lower-frequency sounds, like bass tones, are less affected and keep more of their original volume.

The filter might cause some phase distortion, which is a delay in the signal due to the averaging of multiple samples. This can slightly change the shape of the signal in the time domain.

Overall, the output signal will be smoother and less noisy, with reduced high-frequency content and some changes in phase characteristics.

These wav files was attached in the folder part2 wav.

Part 3 – Speech Analysis

(a) Formants for vowels

"She had your dark suit in greasy wash water all year" from 'sample1.wav'.

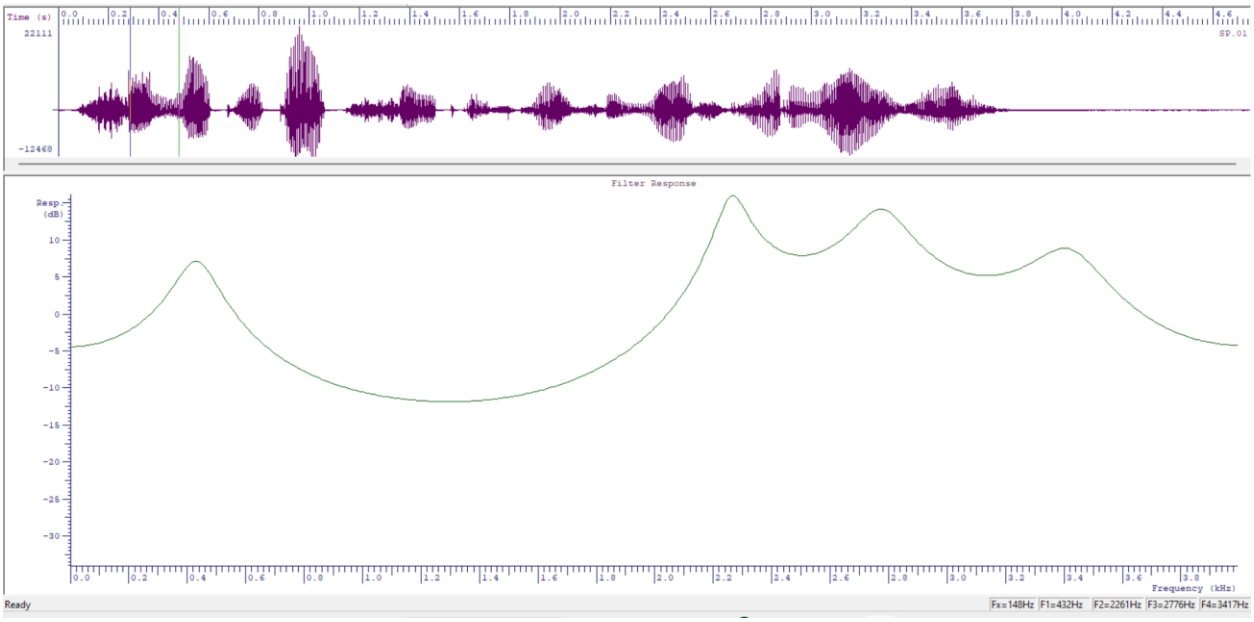


Figure 21: Vowel 'e' in she from 'sample1.wav'

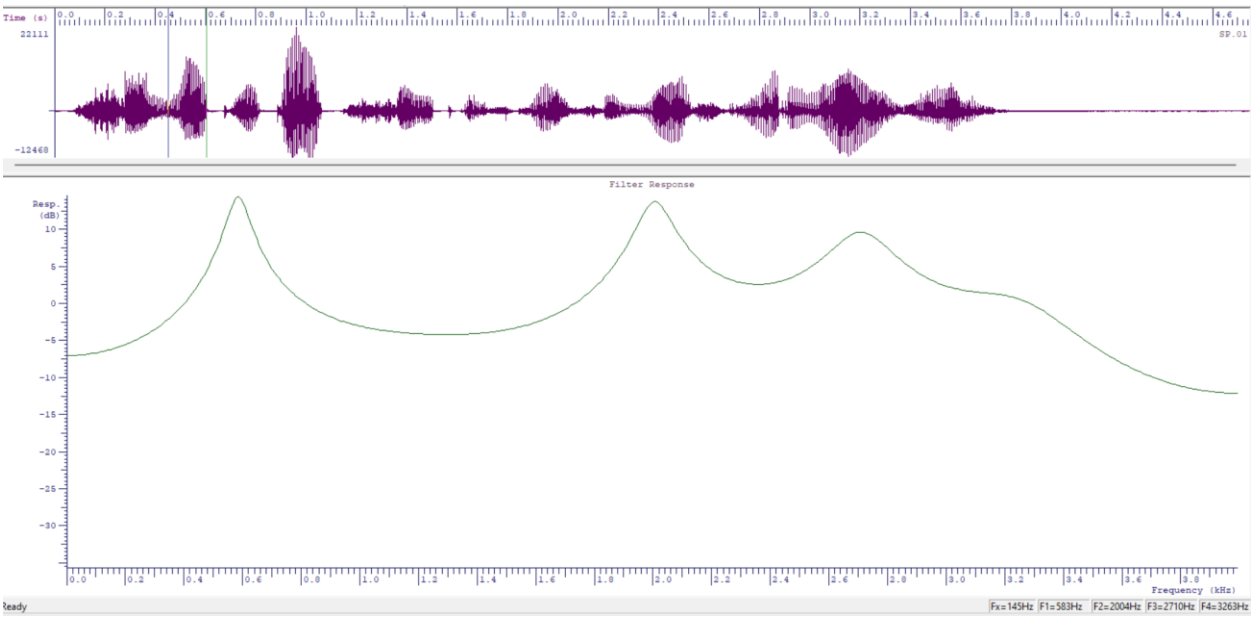


Figure 22: Vowel 'a' in had from 'sample1.wav'

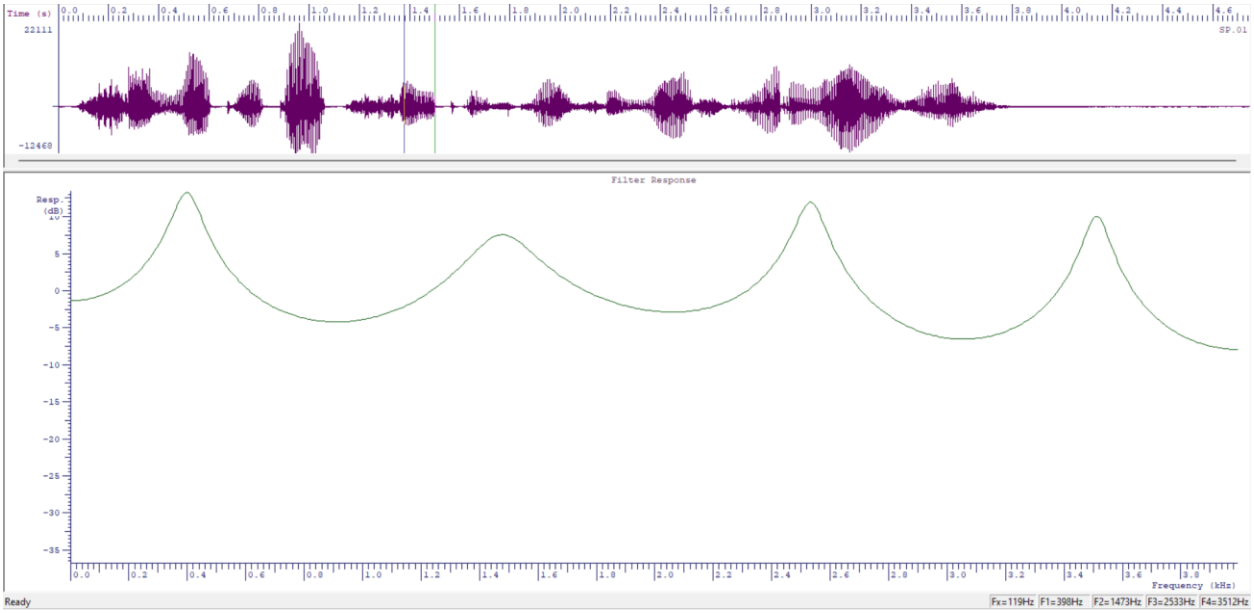


Figure 23: Vowel ‘u’ in suit from 'sample1.wav'

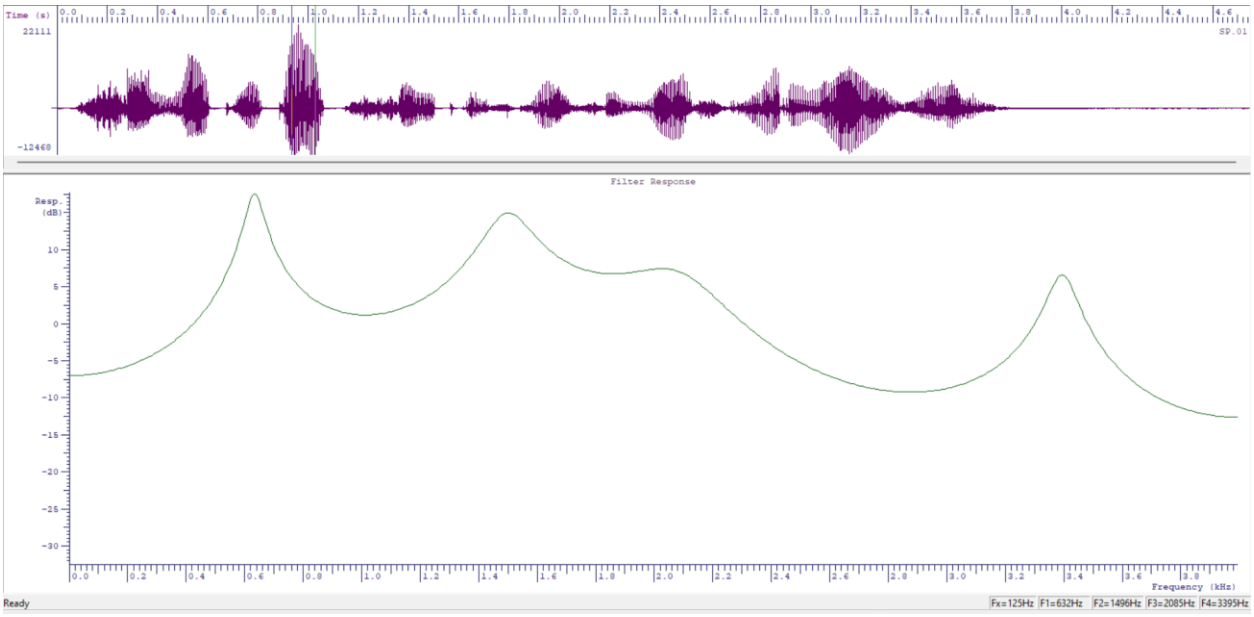


Figure 24: Vowel 'a' in dark from 'sample1.wav'

Table 1: the first three formants F1, F2 and F3 for the four vowels in sample.wav

	F1(Hz)	F2(Hz)	F3(Hz)
E in she	432	2261	2276
A in had	583	2004	2710
U in suit	398	1473	2533
A in dark	632	1496	2085

We Plot F1 versus F2-F1 for these four vowels:

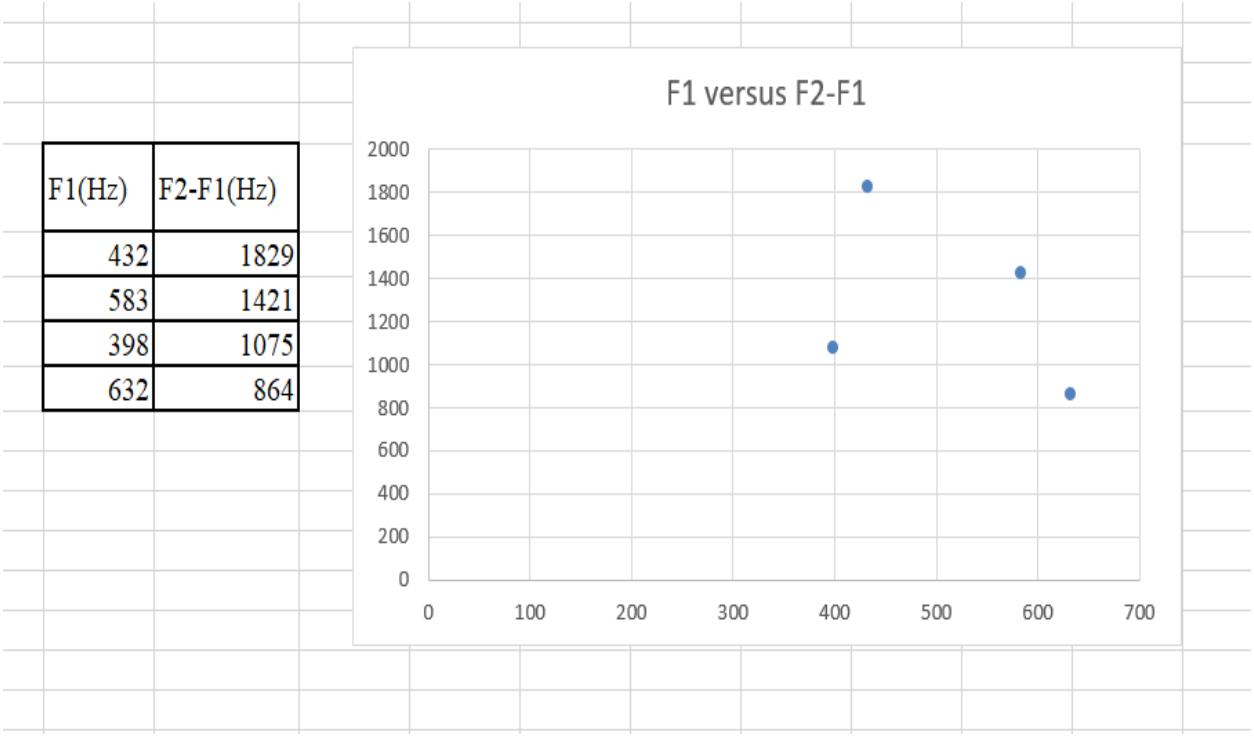


Figure 25: F1 versus F2-F1 plotting in sample.wav

Now, Mohammed Owda record his voice:

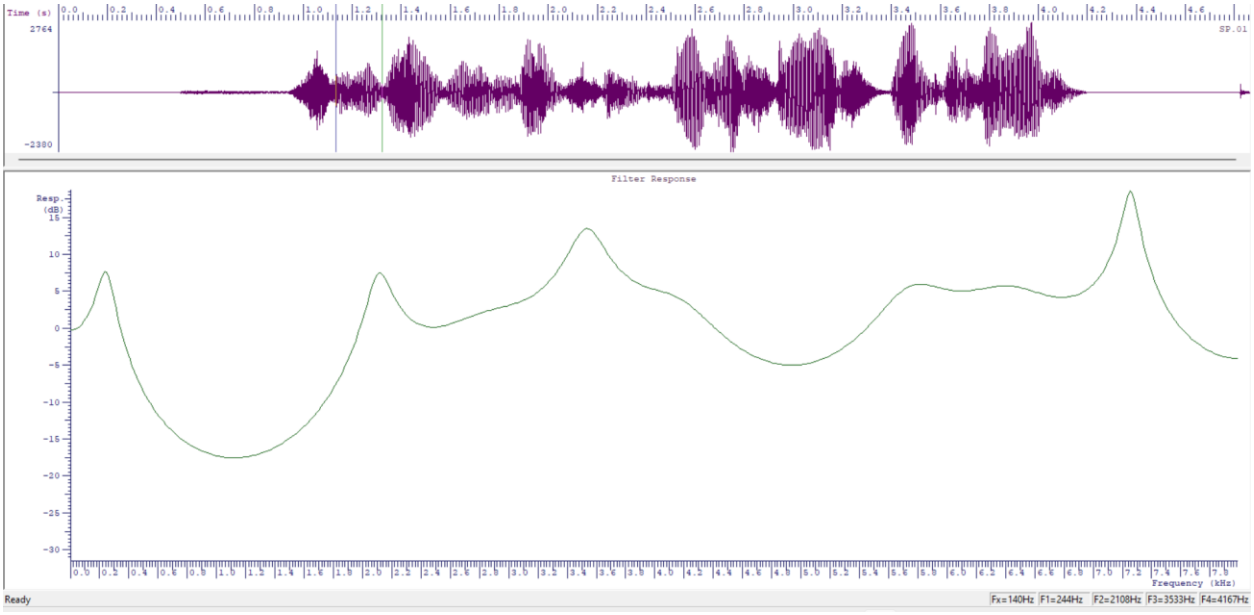


Figure 26: Vowel 'e' in she from 'owda.wav'

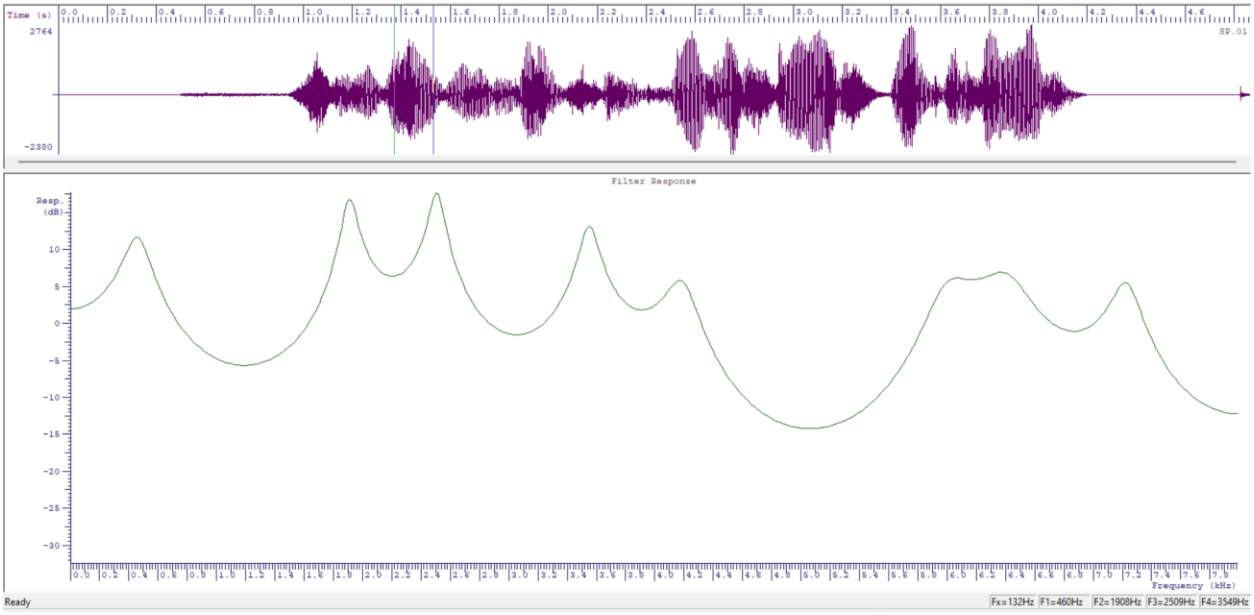


Figure 27: Vowel 'a' in had from 'owda.wav'

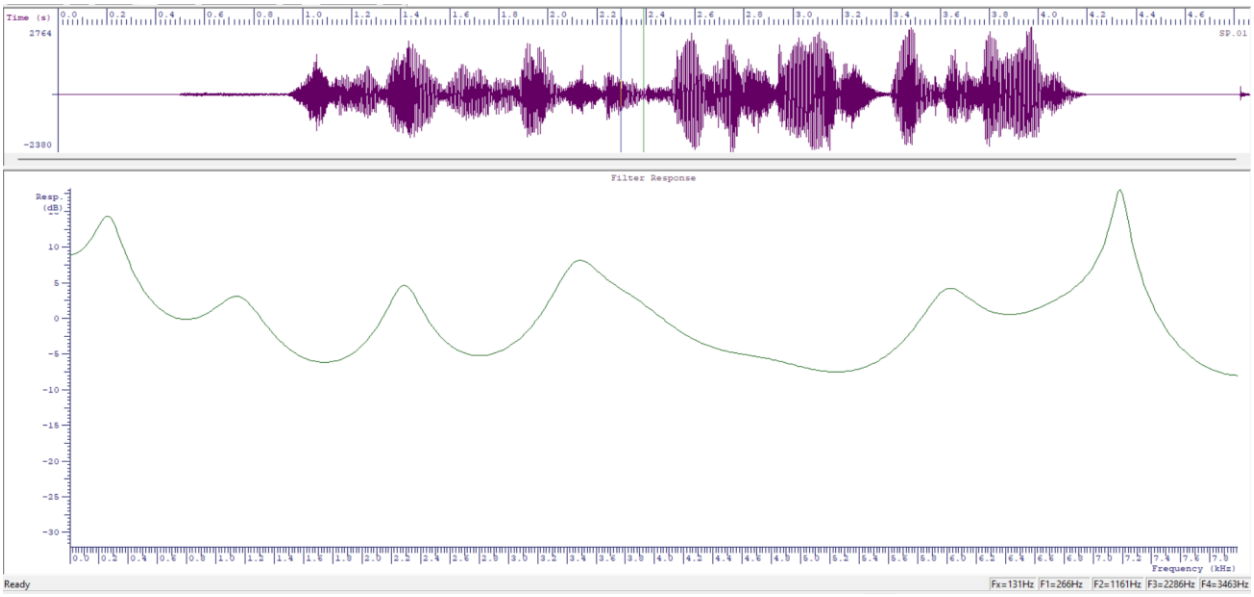


Figure 28: Vowel u'' in suit from 'owda.wav'

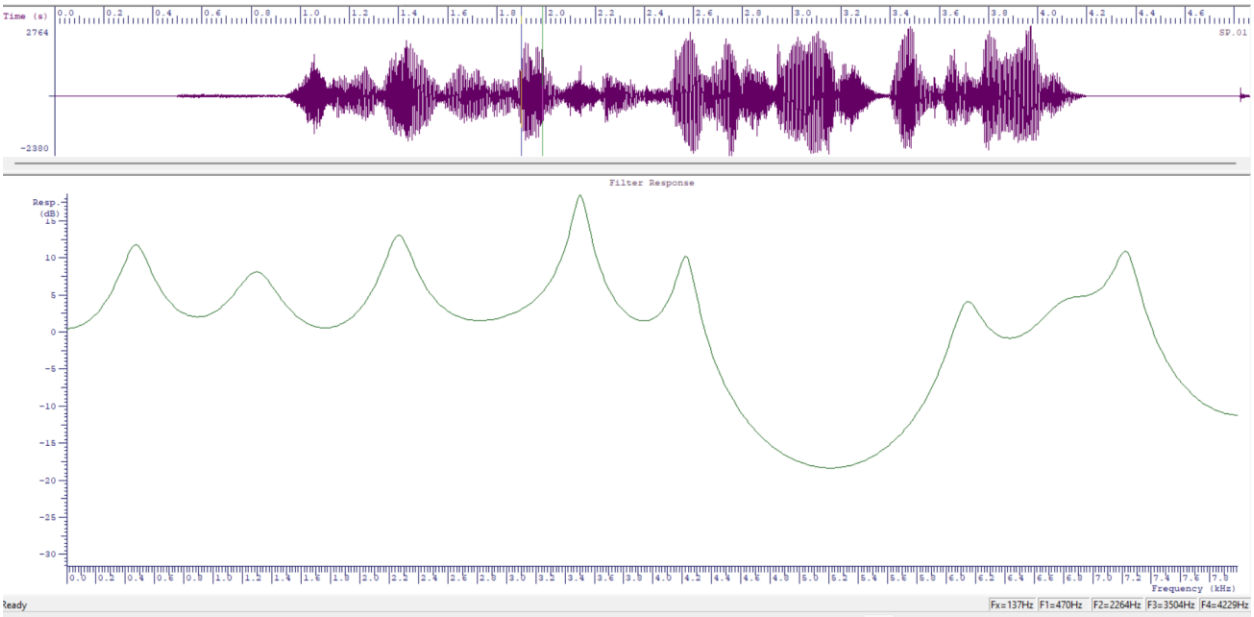


Figure 29: Vowel 'a' in dark from 'owda.wav'

Table 2: the first three formants F1, F2 and F3 for the four vowels in owda.wav

	F1(Hz)	F2(Hz)	F3(Hz)
E in she	244	2108	3533
A in had	460	1908	2509
U in suit	266	2286	3463
A in dark	470	2264	3504

We Plot F1 versus F2-F1 for these four vowels:

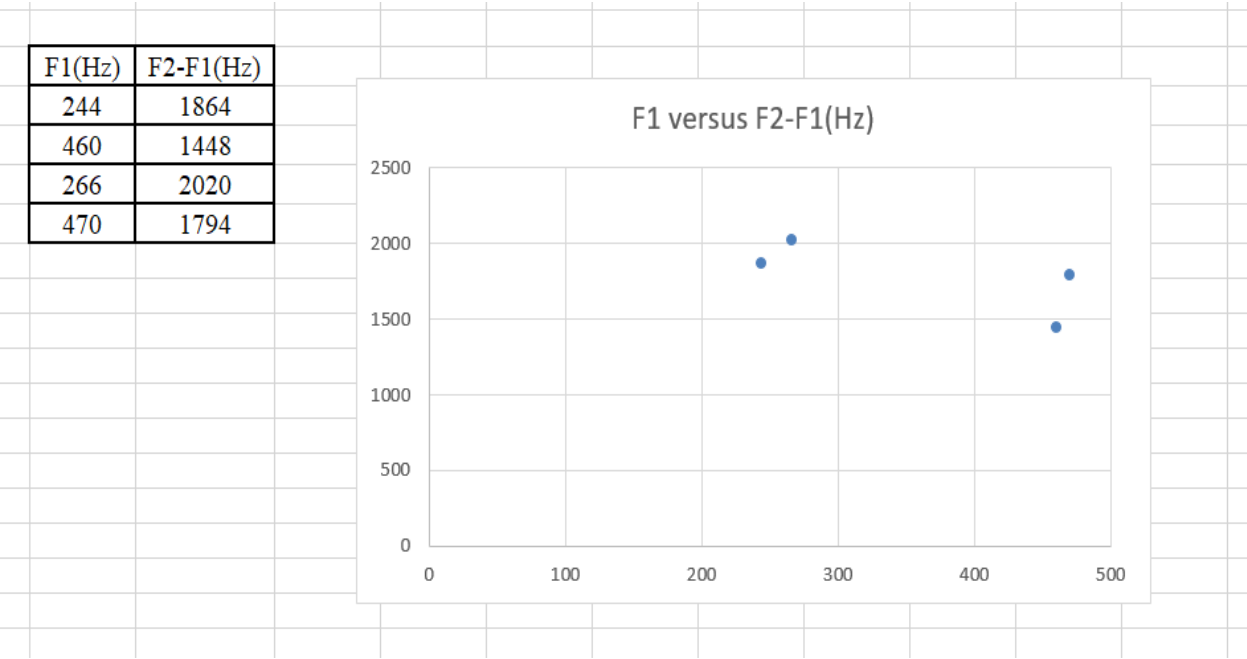


Figure 30: F1 versus F2-F1 plotting in owda.wav

Now, Mohammad Abu Shams record his voice:

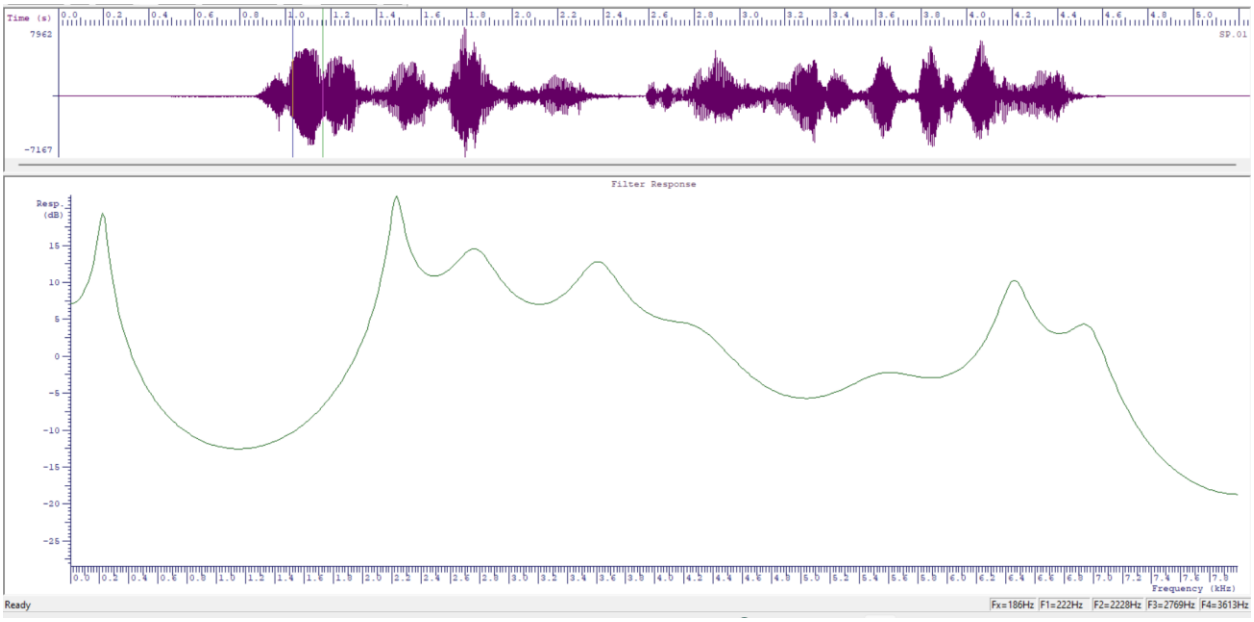


Figure 31: Vowel 'e' in she from 'shams.wav'

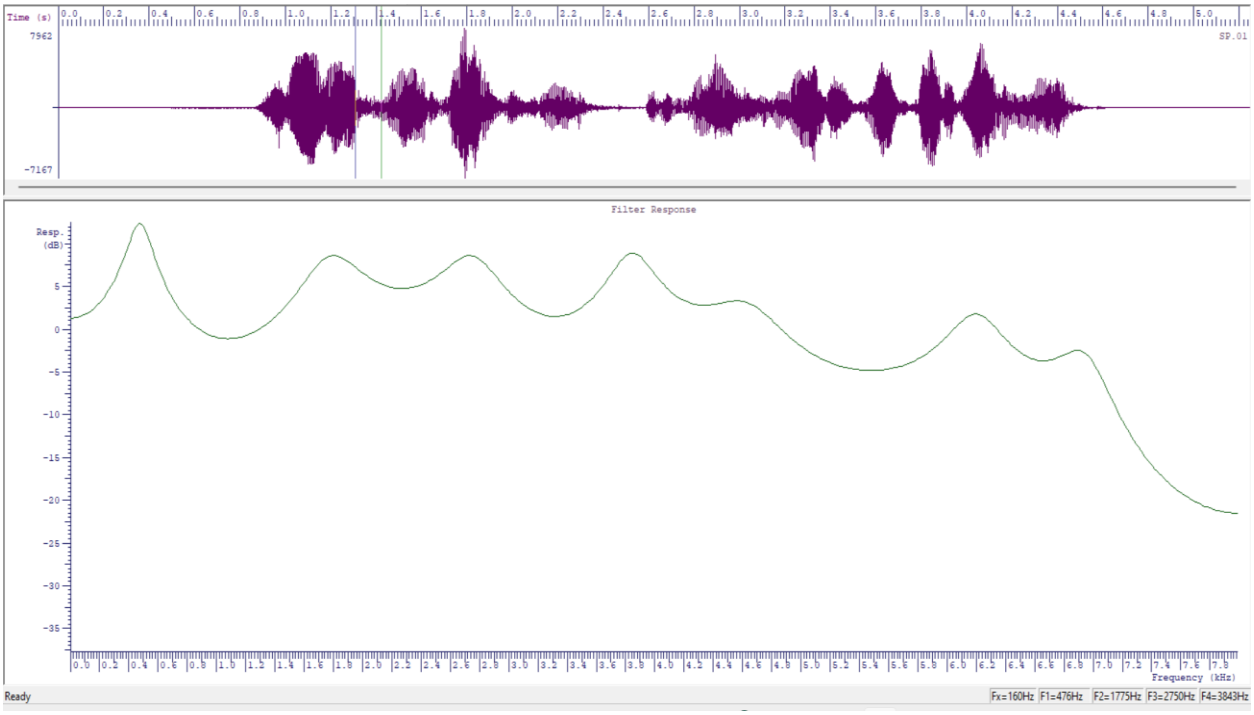


Figure 32: Vowel 'a' in had from shams.wav'

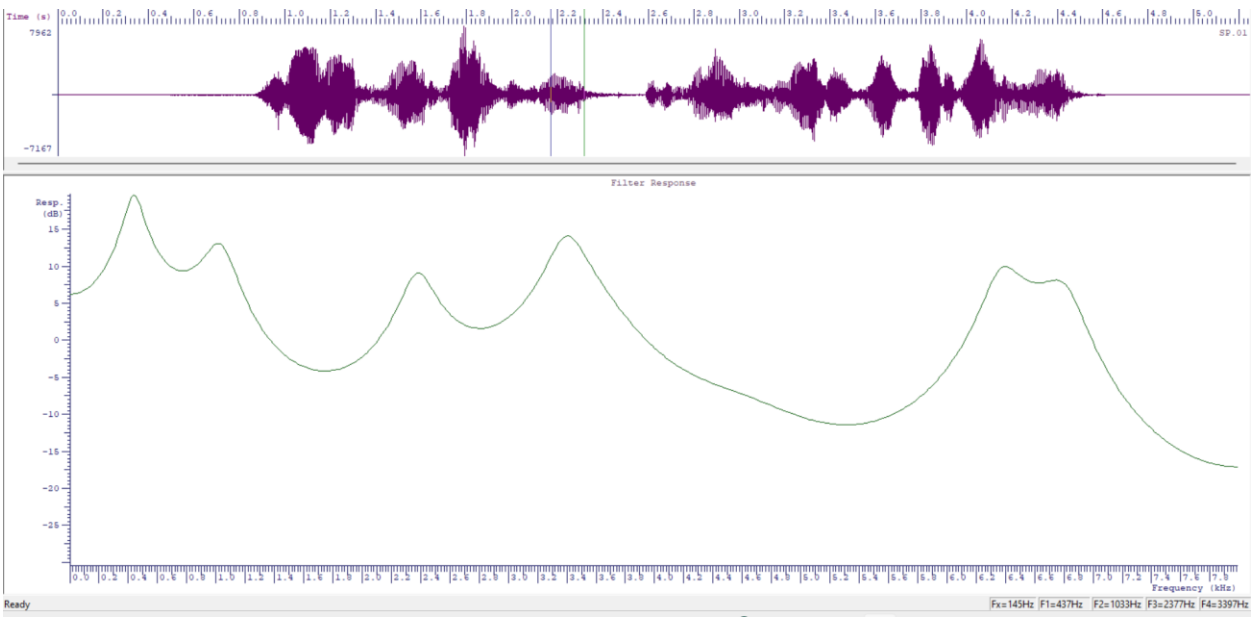


Figure 33: Vowel u'' in suit from shams.wav'

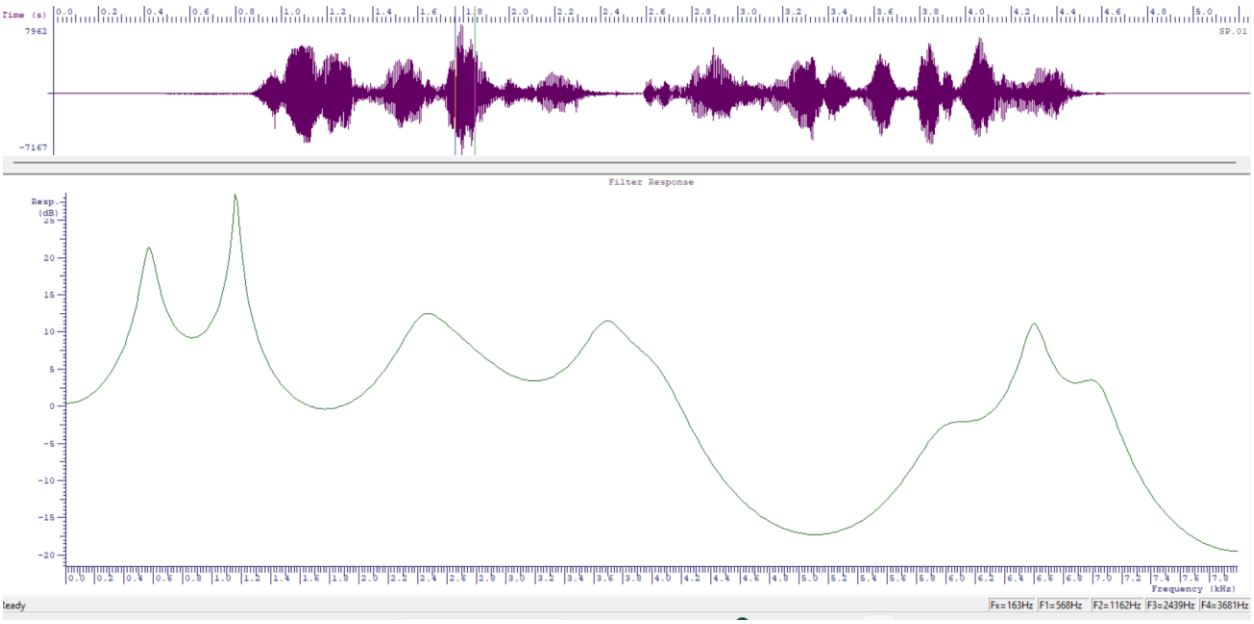


Figure 34: Vowel 'a' in dark from shams.wav'

Table 3: the first three formants F1, F2 and F3 for the four vowels in shams.wav

	F1(Hz)	F2(Hz)	F3(Hz)
E in she	222	2228	2769
A in had	476	1775	3843
U in suit	437	1033	2377
A in dark	568	1162	2439

We Plot F1 versus F2-F1 for these four vowels:

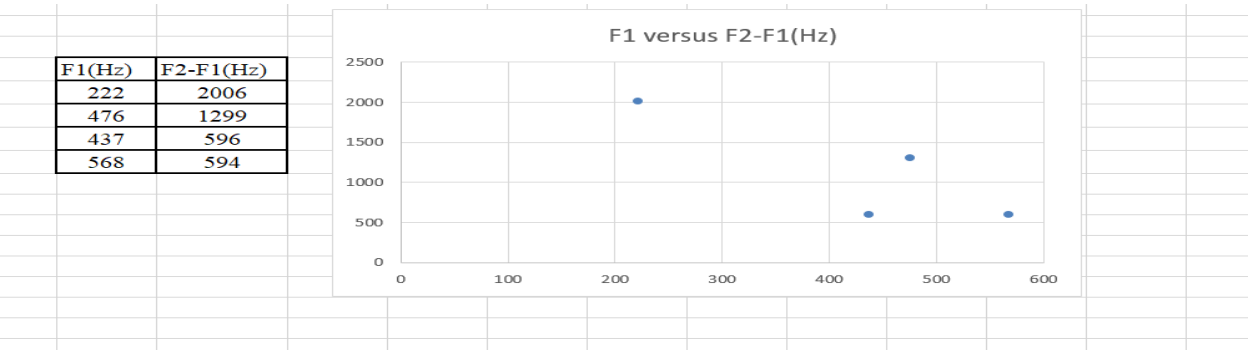


Figure 35: F1 versus F2-F1 plotting in shams.wav

The analysis of vowel sounds in words like "sample," "shams," and "owda" shows how different these sounds can be depending on accent, pronunciation style, or how they were measured. In "sample," the vowel sounds are close to what is expected on a typical chart, suggesting a normal or standard accent. In "shams," the vowel sounds are different, especially for vowels like "e" in "she" and "a" in "had," showing a more pronounced or exaggerated accent. The vowel sounds in "owda" are quite varied, especially for "u" in "suit," which suggests a very different pronunciation style or the speaker having a unique vocal setup. These differences are important to understand because they can impact how people learn accents or how speech recognition systems work.

These wav files was attached in the folder part3 wav.

(b) Source and Filter analysis

We Depict figure of the source (S) spectrum and filter (F) spectrum:

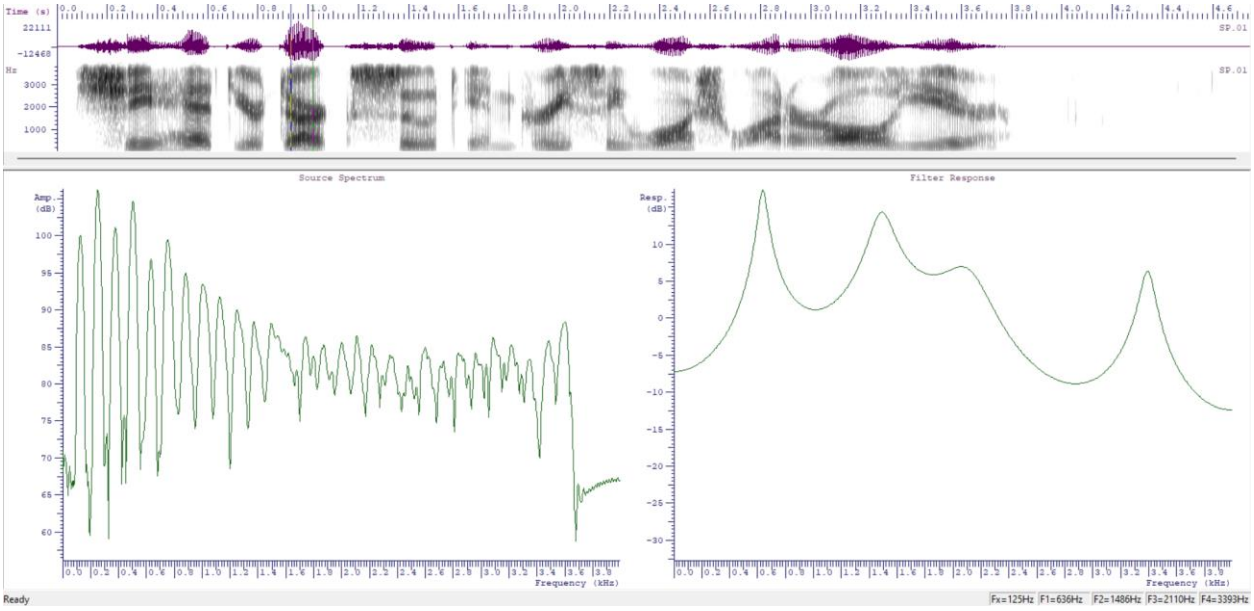


Figure 36: the source (S) spectrum and filter (F) spectrum for the 'a' in dark

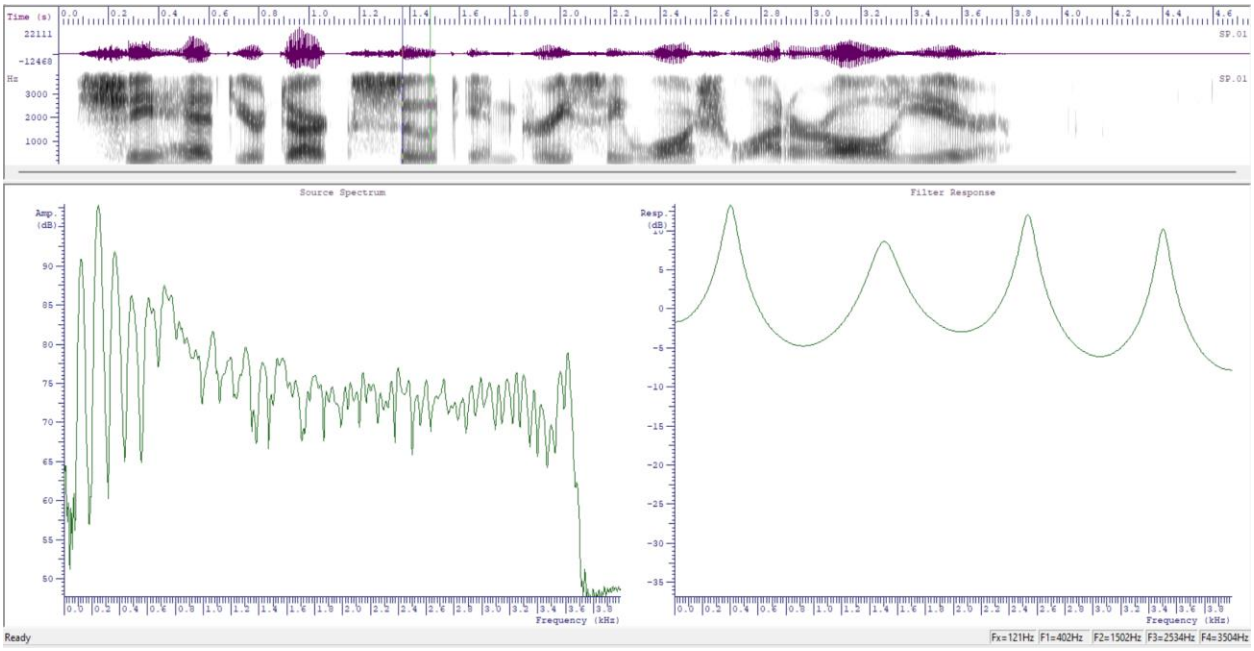


Figure 37: the source (S) spectrum and filter (F) spectrum for the 'u' in suit

The production differences in the sounds are as follows:

Harmonic Content: The first sound has clear harmonics, which indicates it might be a musical note or a tonal sound. The second sound has a more even spread of harmonics, which could mean it's less tonal or has a different quality.

Energy Distribution: In the original sound, the first sound has more energy in the lower frequencies, while the second sound loses energy in the middle frequencies. This could mean different instruments or ways of playing.

Filter Characteristics: Both filter responses have peaks and valleys, which affects the final sound by boosting or reducing certain frequencies. The first filter is smoother, so it gently changes the sound, while the second has sharper peaks, so it changes the sound more dramatically.

Timbral Changes: The filter responses suggest each sound will change in its own way. The first sound keeps more of its original quality with small boosts in certain frequencies, while the second sound undergoes more significant changes due to the sharper filter, possibly creating stronger resonances.

Sound Production: The differences in the original sounds suggest they came from different sources or methods. The filter responses show they were processed differently, leading to unique characteristics in the final sounds.

(c) Fundamental Frequency
From Owda.wav:

'a' in dark:

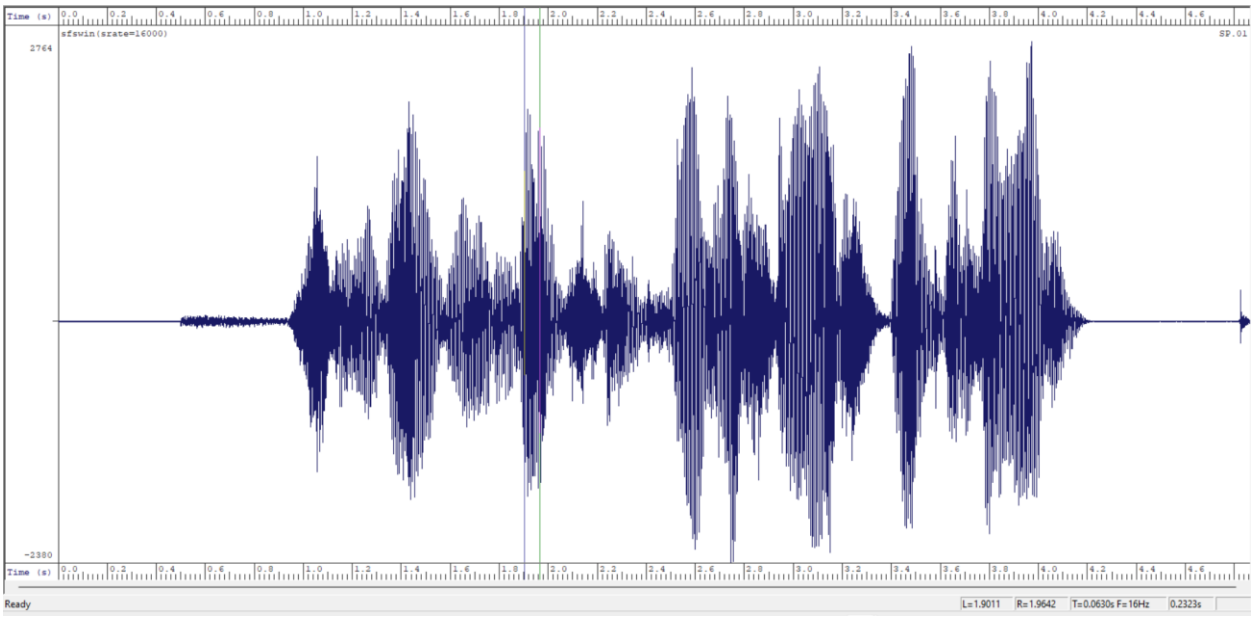


Figure 38: Waveform for 'a' in dark from owda.wav

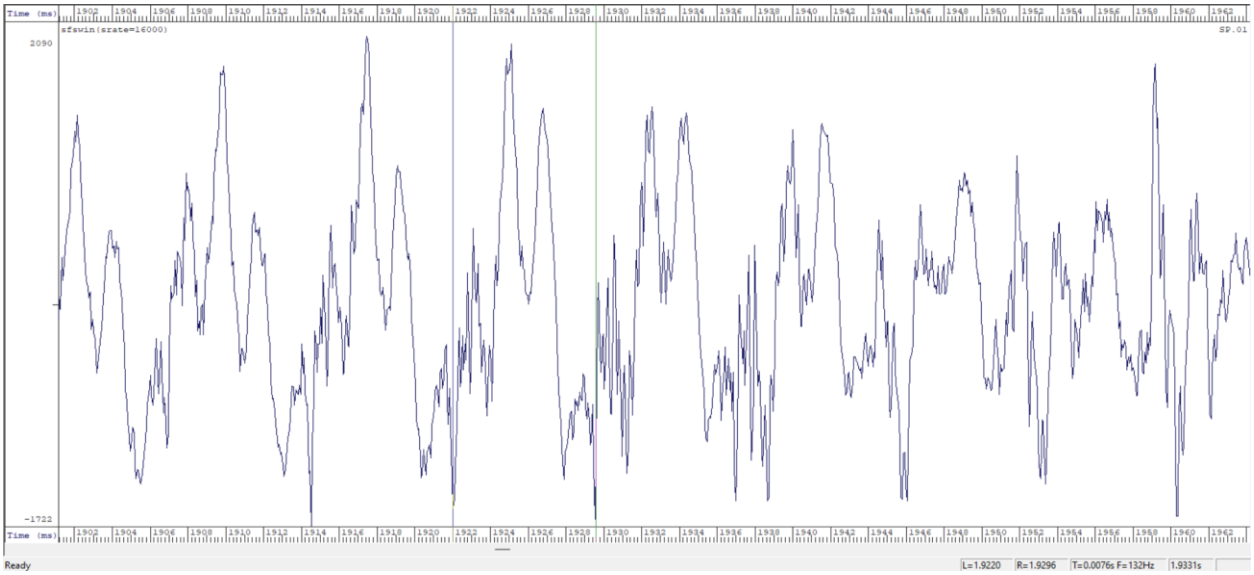


Figure 39: fundamental frequency of 'a' in owda.wav

Fundamental frequency=132Hz.

From Shams.wav:

‘a’ in dark:

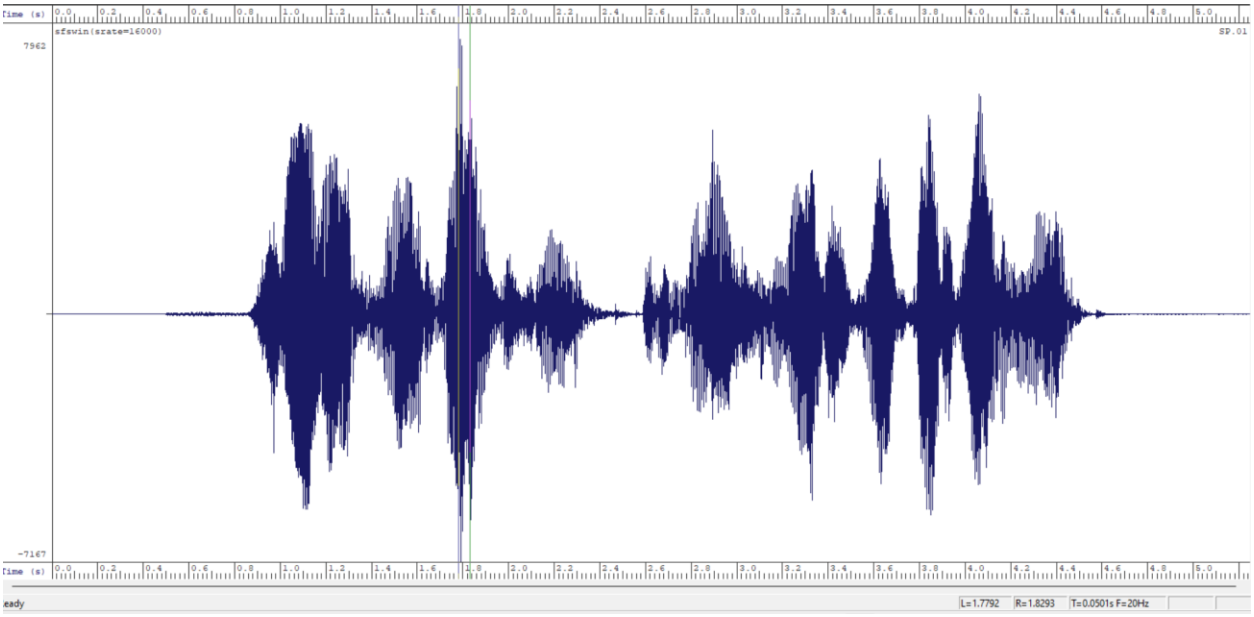


Figure 40: Waveform for 'a' in dark from shams.wav

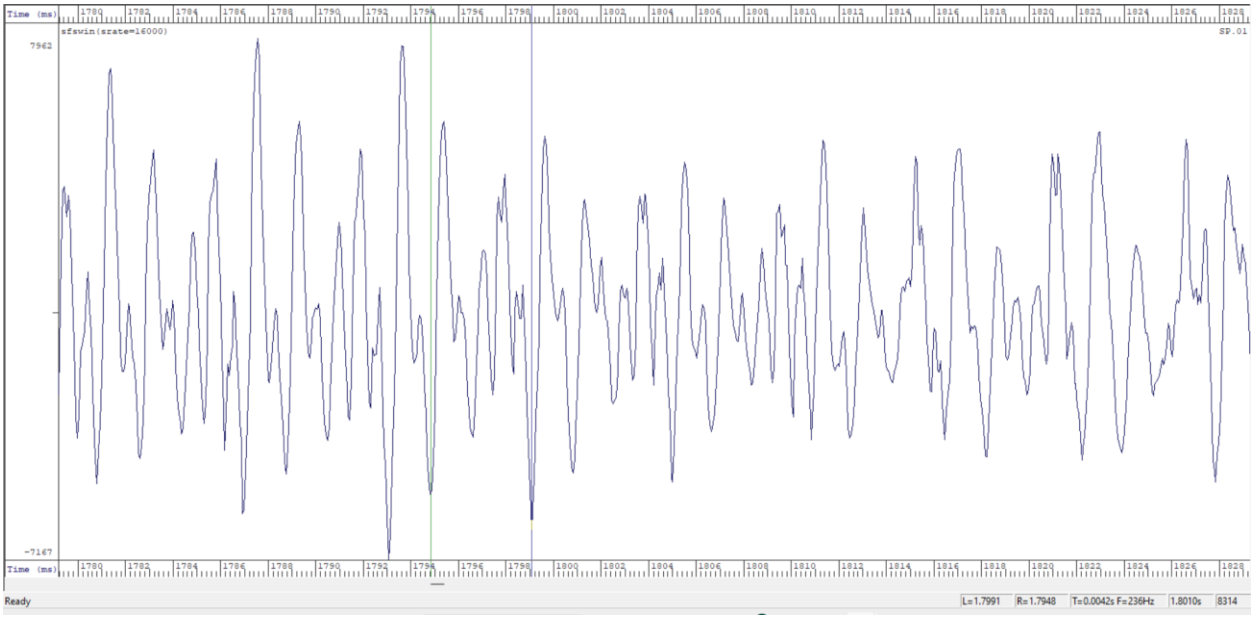


Figure 41: fundamental frequency of 'a' in shams.wav

Fundamental frequency=236Hz.

Part 4 – Speech analysis

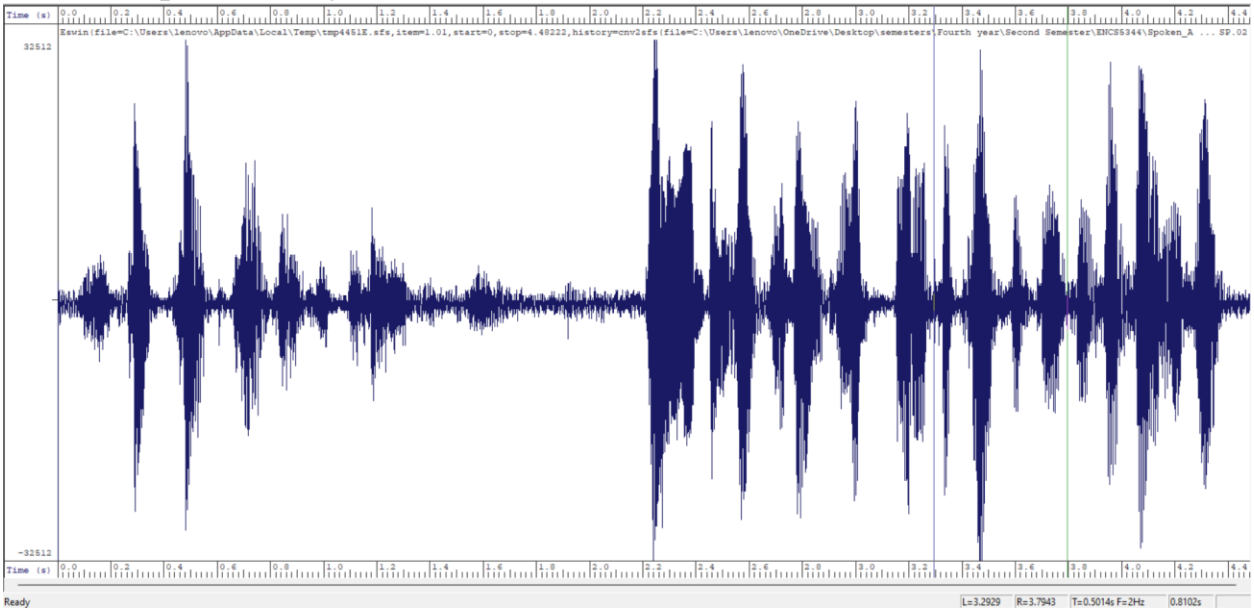


Figure 42: Waveform for Aws1.wav

Capacity: / k @ p { s I t i /

Figure 43: Waveform for capacity

Figure 44: Waveform for /k/

Figure 45: Waveform for /@/

Figure 46: Waveform for /p/

Figure 47: Waveform for / {/

Figure 48: Waveform for /s/

Figure 49: Waveform for /I/

Figure 50: Waveform for /t/

Figure 51: Waveform for /i/

We identify precisely the start and end times of each of the eight phonemes in capacity in the below table:

Table 4: the start and end times of each of the eight phonemes in capacity

	Start time (ms)	End Time (ms)
k	0	88.7
@	88.7	116.2
p	116.2	205.8
{	205.8	243.6
s	243.6	344.5
l	344.5	378.1
t	378.1	462.1
i	462.1	525.1

These wav files was attached in the folder part4 wav.

Part 5 – Sub-word level concatenation

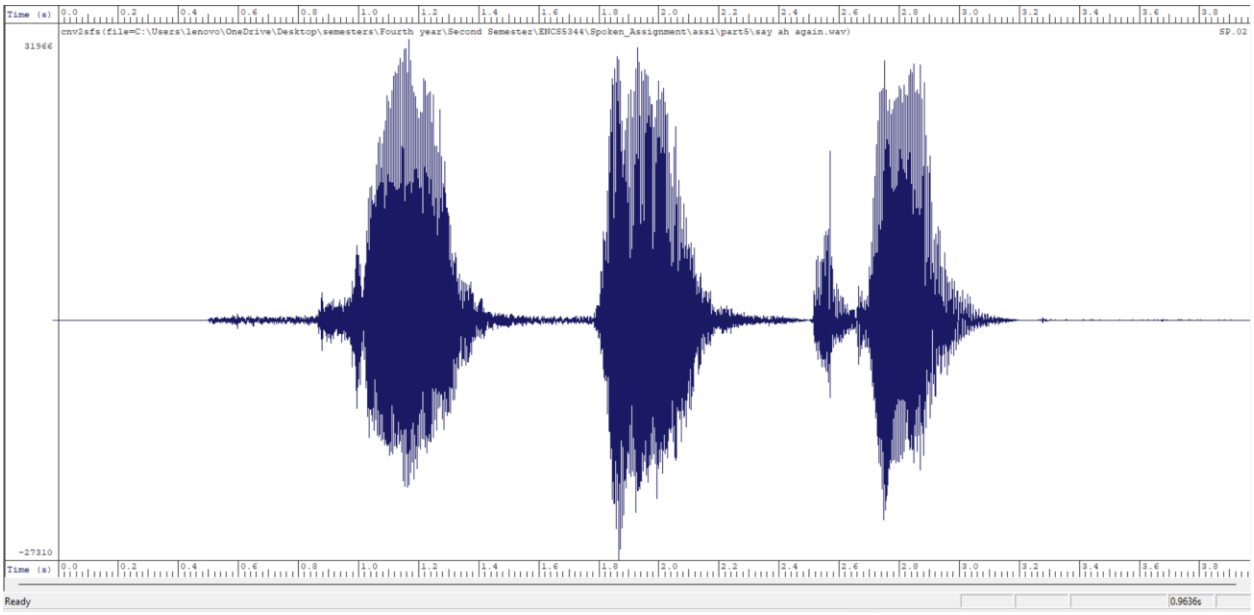


Figure 52: Waveform for say ah again

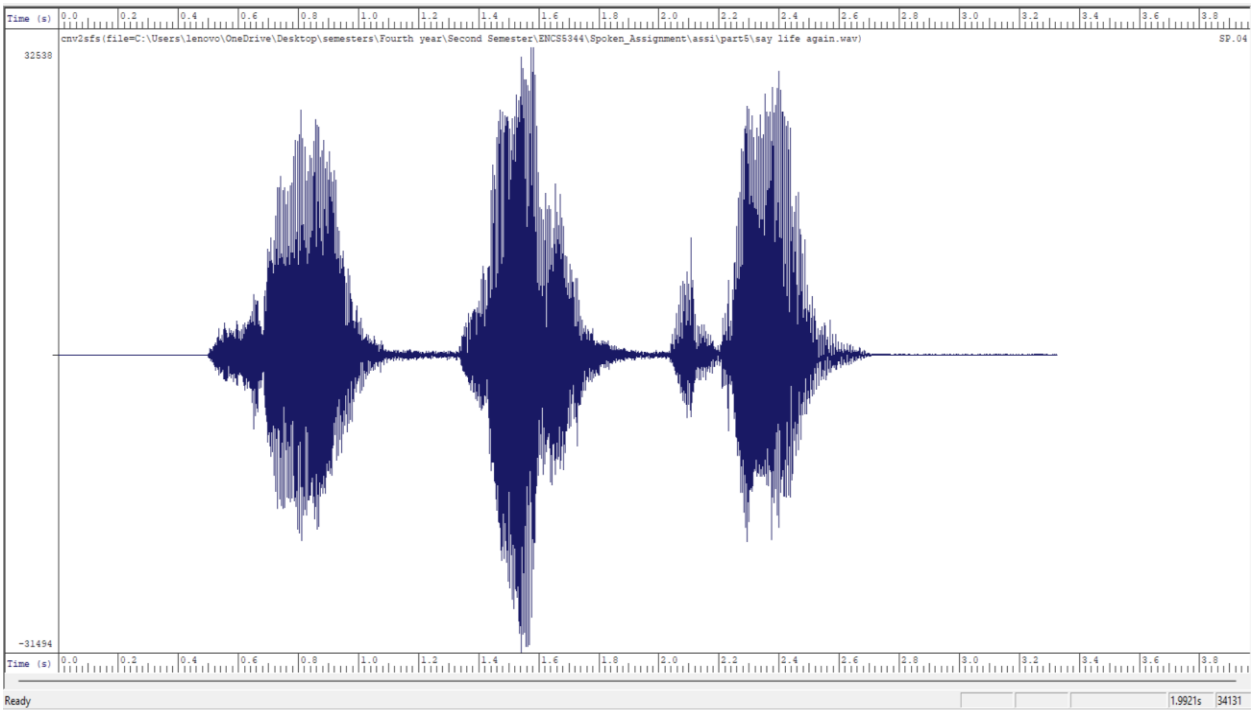


Figure 53: Waveform for say life again

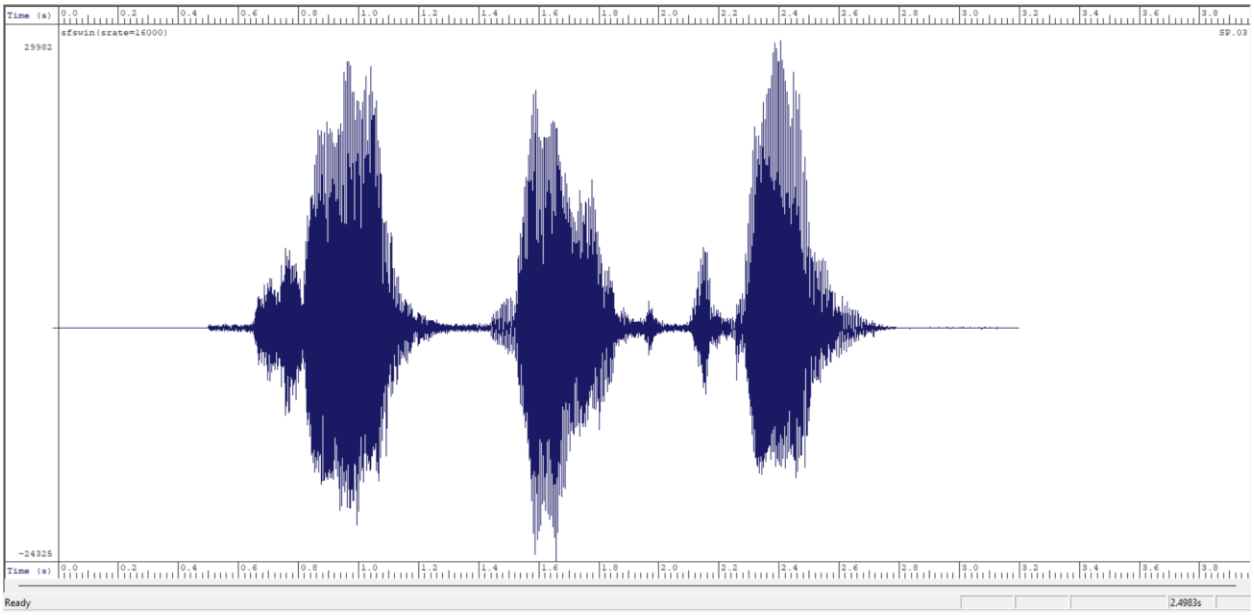


Figure 54: Waveform for say night again

Figure 55: Waveform for ah

Figure 56: Waveform for life

Figure 57: Waveform for night

Figure 58: Waveform for li

Figure 59: Waveform for ght

We concatenate two waves together:

```
% Mohammad Abu Shams 1200549.
% Mohammed Owda 1200089.
% Section1.

% Load the sounds.
[li, Fs] = audioread('li.wav');    % li from life.
[ght, ~] = audioread('ght.wav');   % ght from night.

% Find zero-crossing points near the end of li and the start of ght.
li_end = find(li(end-1000:end) .* li(end-1001:end-1) <= 0, 1, 'last') + (length(li)-1001);
ght_start = find(ght(1:1000) .* ght(2:1001) <= 0, 1, 'first');

% Cut the segments at zero-crossings.
li = li(1:li_end);
ght = ght(ght_start:end);

% Cross fade window size.
crossfade_samples = floor(Fs * 0.02);  % 20 ms cross fade.
fade_in = linspace(0, 1, crossfade_samples)';
fade_out = linspace(1, 0, crossfade_samples)';

% Create the cross-fade section.
crossfade_section = li(end-crossfade_samples+1:end) .* fade_out + ght(1:crossfade_samples) .* fade_in;

% Concatenate 'li' without its fade out, the cross fade section, and 'ght' without its fade in
light = [li(1:end-crossfade_samples); crossfade_section; ght(crossfade_samples+1:end)];

% Normalize the concatenated signal.
light = light / max(abs(light));

% Listen.
sound(light, Fs);

% Write the sound.
audiowrite('light.wav', light, Fs);
```

```
>> PART5_1
>>
```

Figure 60: concatenate li and ght in matlab

When combining sound clips to create speech, we need to make sure they match in speed and natural rhythm. This helps the speech sound smooth and realistic. Transitions between clips should be seamless to avoid noticeable clicks or harsh changes in sound. Aligning the tone and loudness of different clips keeps the voice quality consistent throughout. It's important to carefully choose where the clips connect, adjusting their volume and phase to match each other, and use cross-fade techniques to blend them smoothly. This way, the final speech will sound clear and natural.

These wav files was attached in the folder part5 wav.

Part 6 – Phone-level concatenation

We choose a word “cat” wich contain 3 phonemes: /k/, /@/, /t/.

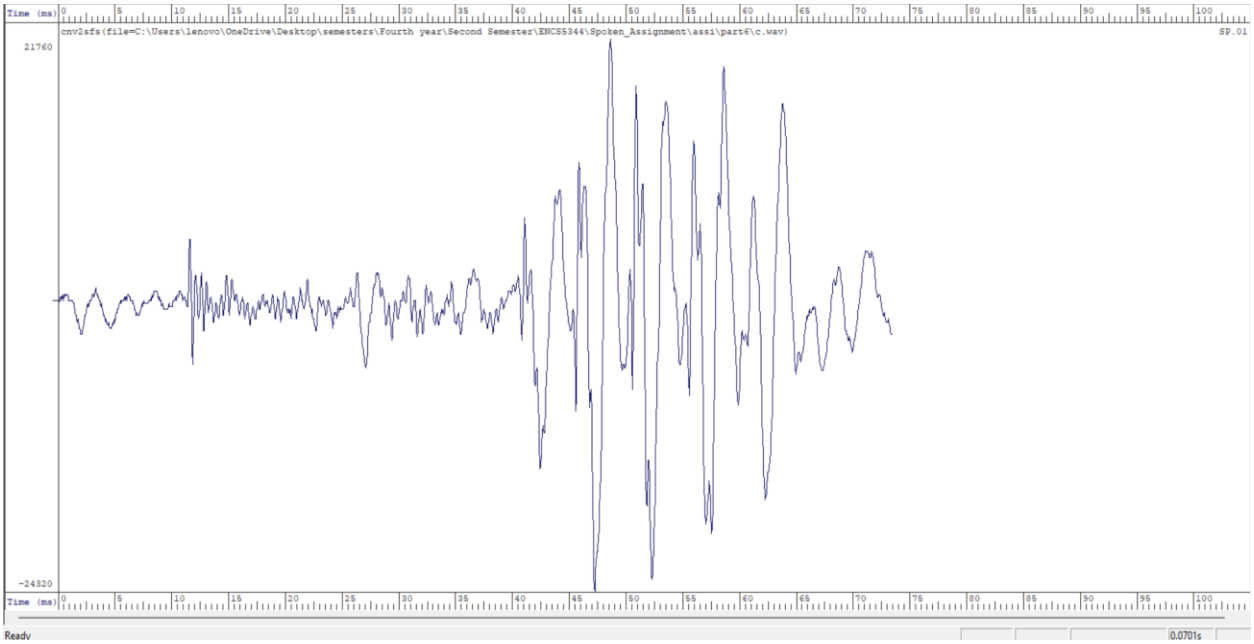


Figure 61: Waveform for c

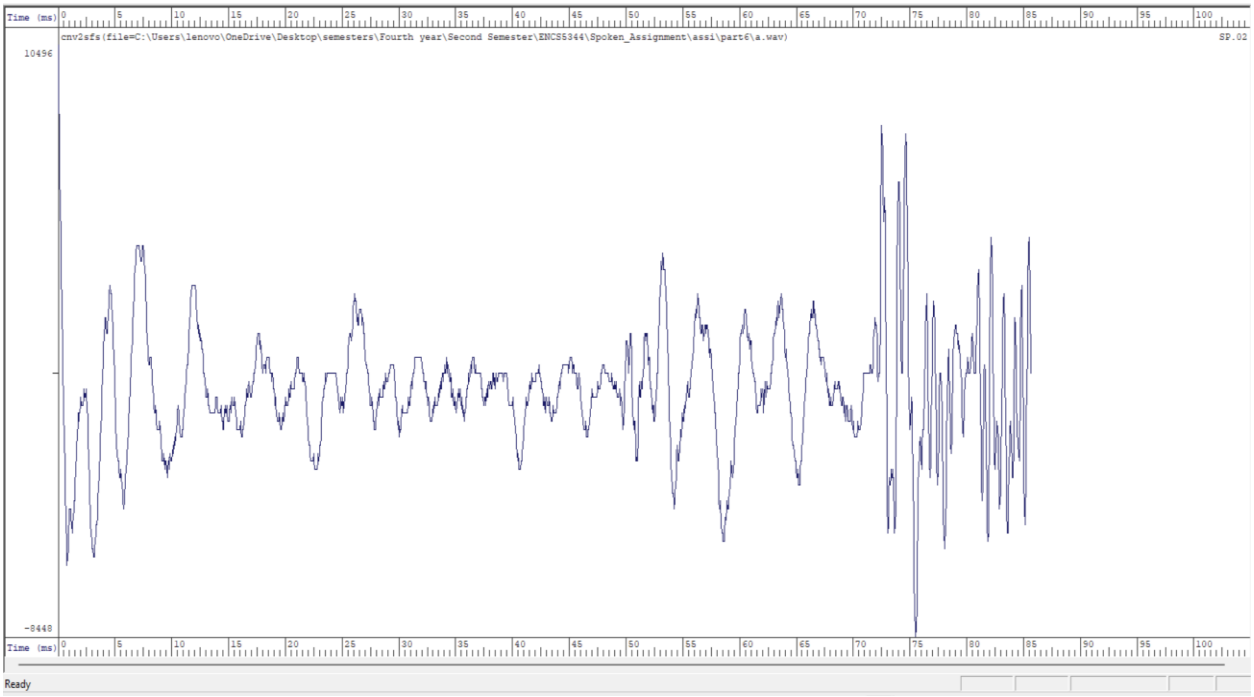


Figure 62: Waveform for a

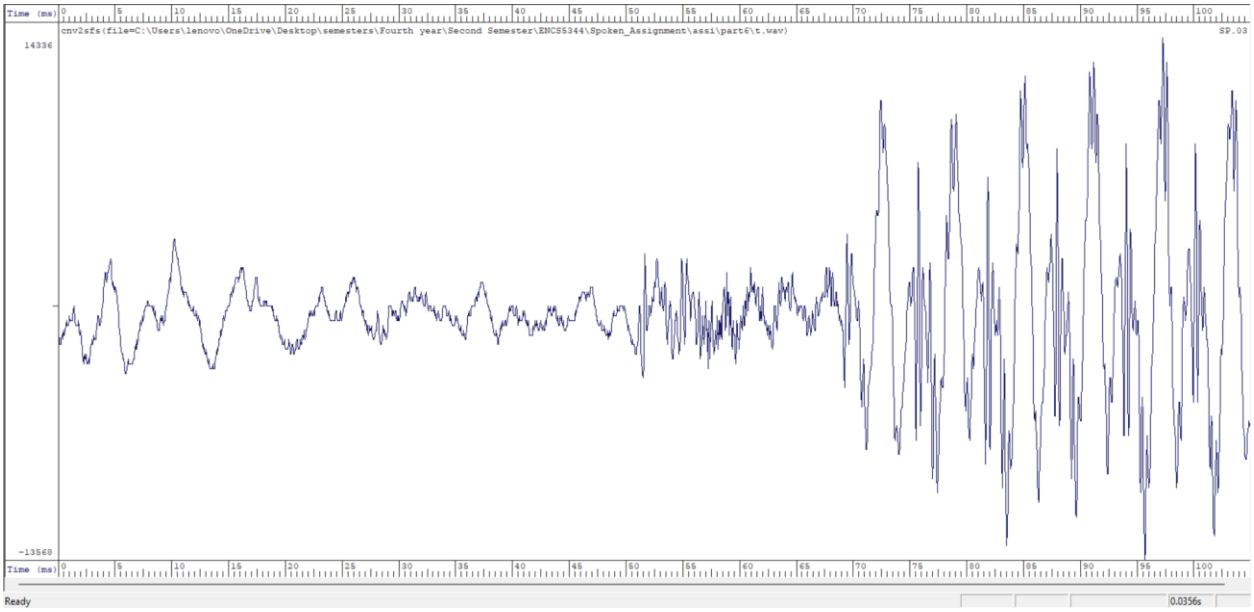


Figure 63: Waveform for t

part6.m

```
% Mohammad Abu Shams 1200549.
% Mohammed Owda 1200089.
% Section1.

% Load the sounds.
[c, Fs] = audioread('c.wav');    % Load 'c' sound.
[a, ~] = audioread('a.wav');    % Load 'a' sound.
[t, ~] = audioread('t.wav');    % Load 't' sound.

% Define crossfade duration in seconds.
crossfade_duration = 0.02;  % 20 ms crossfade.

% Calculate crossfade window size in samples.
crossfade_samples = floor(Fs * crossfade_duration);

% Generate crossfade windows.
fade_in = linspace(0, 1, crossfade_samples)';
fade_out = linspace(1, 0, crossfade_samples)';

% Concatenate 'c' and 'a' with crossfade.
ca = crossfade_sounds(c, a, fade_in, fade_out, crossfade_samples);

% Concatenate 'ca' and 't' with crossfade.
cat = crossfade_sounds(ca, t, fade_in, fade_out, crossfade_samples);

```

```
% Normalize the concatenated signal.
cat = cat / max(abs(cat));

% Listen to the concatenated audio.
sound(cat, Fs);

% Write the resulting audio to a file.
audiowrite('cat.wav', cat, Fs);

% Function definition to crossfade sounds.
function result = crossfade_sounds(sound1, sound2, fade_in, fade_out, crossfade_samples)
    % Define search ranges.
    % Ensure the search range does not exceed the length of the arrays.
    sound1_search_range = min(length(sound1), 1000);
    sound2_search_range = min(length(sound2), 1000);

    % Find zero-crossing points in the defined ranges.
    sound1_end = find(sound1(end - sound1_search_range + 1:end) .* sound1(end - sound1_search_range:end - 1) <= 0, 1, 'last');
    sound2_start = find(sound2(1:sound2_search_range) .* sound2(2:sound2_search_range + 1) <= 0, 1, 'first');

    % Adjust found indices if necessary.
    if ~isempty(sound1_end)
        sound1_end = length(sound1) - sound1_search_range + sound1_end - 1;
    else
        sound1_end = length(sound1);
    end

    if ~isempty(sound2_start)
        sound2_start = sound2_start;
    else
        sound2_start = 1;
    end

    % Cut the segments at zero-crossings.
    sound1 = sound1(1:sound1_end);
    sound2 = sound2(sound2_start:end);

    % Create the crossfade section.
    crossfade_section = sound1(end - crossfade_samples + 1:end) .* fade_out + sound2(1:crossfade_samples) .* fade_in;

    % Concatenate 'sound1' without its fade out, the crossfade section, and 'sound2' without its fade in.
    result = [sound1(1:end - crossfade_samples); crossfade_section; sound2(crossfade_samples + 1:end)];
end
```

```
Command Window
>> part6
>>
```

Figure 64: concatenate c and a and t in matlab

The quality of the speech can be affected by sudden changes in sound quality or volume, and zero-crossing points may not always be the best places to join sounds together. This can lead to unwanted sounds or distortions. To improve the quality, we can use advanced audio processing techniques such as aligning the waveforms more accurately, matching the tone and quality of the sounds, and using machine learning to find the best points to join sounds based on natural speech examples. We can also adjust the length of the crossfade based on the context of the sounds being joined, and use a database of small sound units for smoother and more natural speech transitions.

These wav files was attached in the folder part6 wav.

Part 7 – Generation of vowel sounds using the source-filter model (parallel formant synthesiser)

We generate pulse train signal as below:

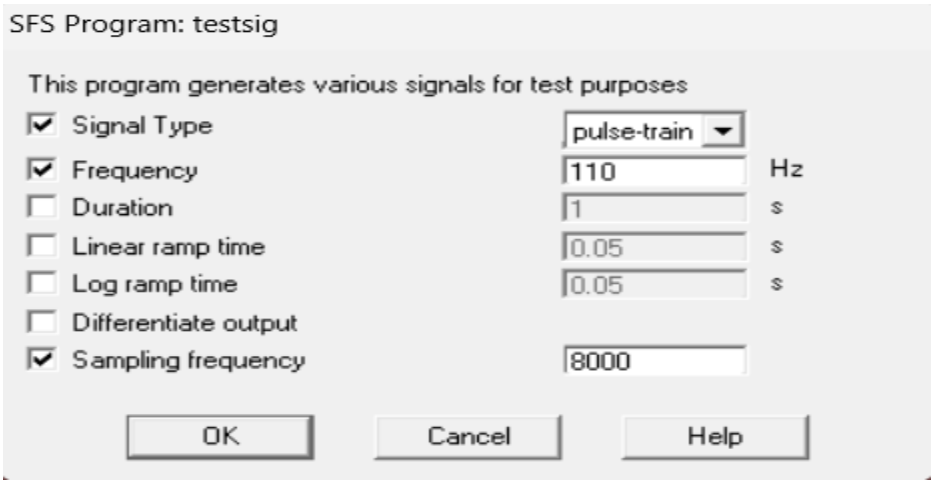


Figure 65: Pulse Train Signal

We defined formant frequencies using band-pass filter.

For F1=700Hz:

Lower band-pass frequency edge: 700 Hz - 200 Hz = 500 Hz.

Higher band-pass frequency edge: 700 Hz + 200 Hz = 900 Hz.

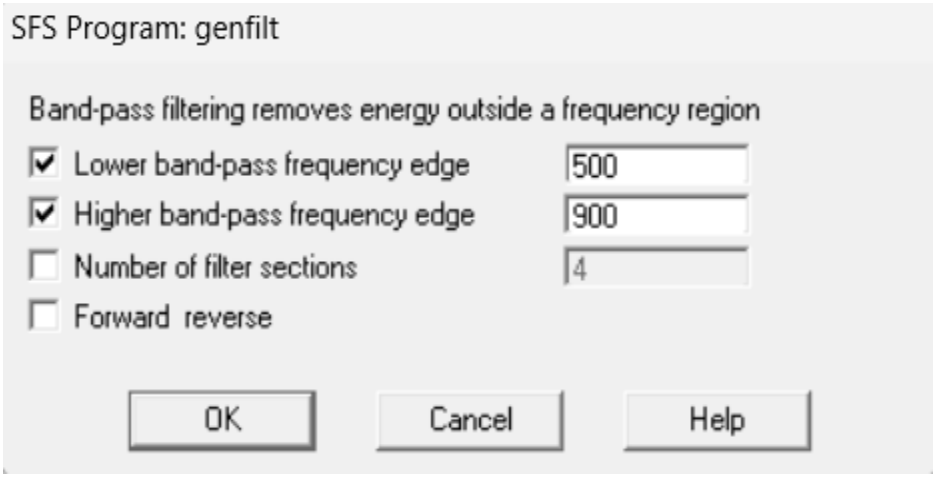


Figure 66: Band-pass frequencies for F1 'a'

For F2=1100Hz:

Lower band-pass frequency edge: 1100 Hz - 200 Hz = 900 Hz.

Higher band-pass frequency edge: 1100 Hz + 200 Hz = 1300 Hz.

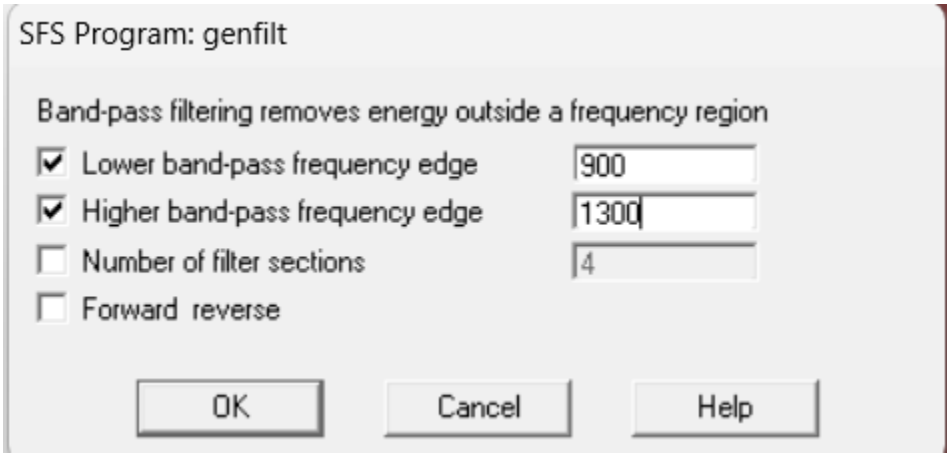


Figure 67: Band-pass frequencies for F2 'a'

For F1=300Hz:

Lower band-pass frequency edge: 300 Hz - 200 Hz = 100 Hz.

Higher band-pass frequency edge: 300 Hz + 200 Hz = 500 Hz.

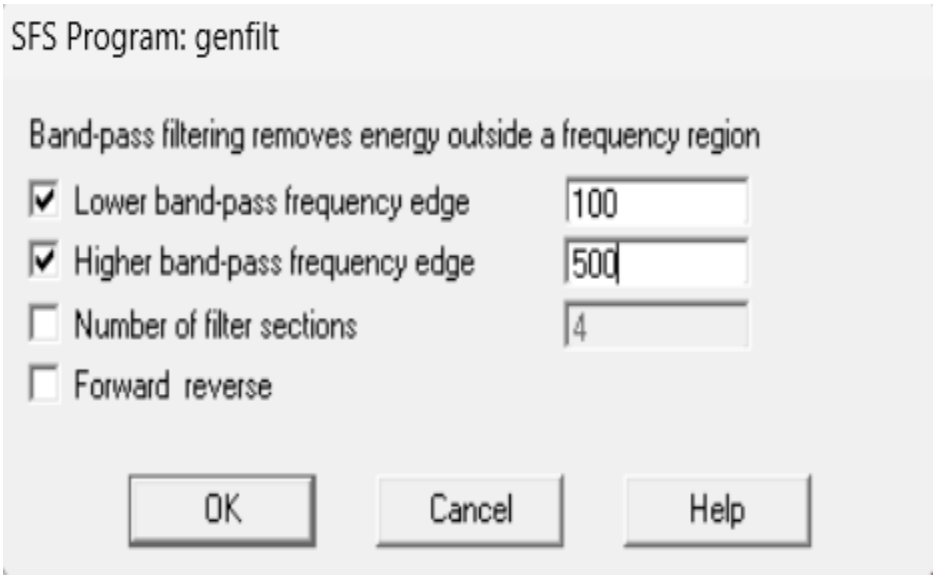


Figure 68: Band-pass frequencies for F1 'u'

For F2=800Hz:

Lower band-pass frequency edge: 800 Hz - 200 Hz = 600 Hz.

Higher band-pass frequency edge: 800 Hz + 200 Hz = 1000 Hz.

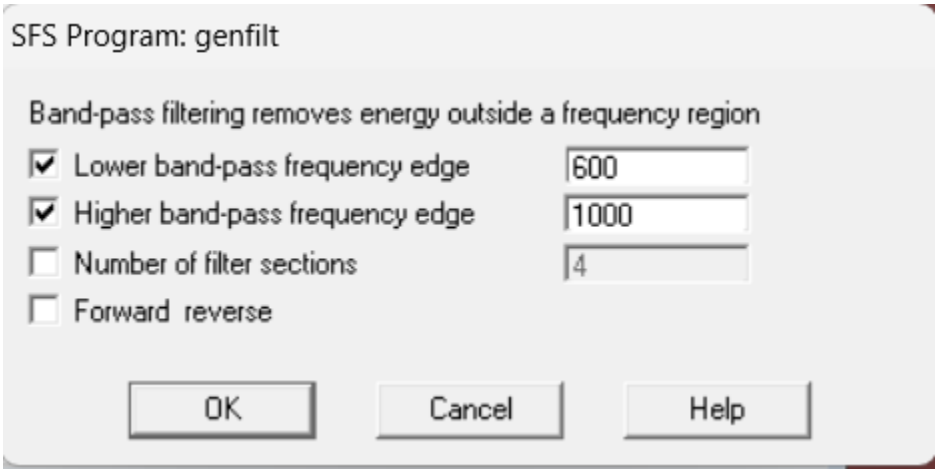


Figure 69: Band-pass frequencies for F2 'u'

For F1=500Hz:

Lower band-pass frequency edge: 500 Hz - 200 Hz = 300 Hz.

Higher band-pass frequency edge: 500 Hz + 200 Hz = 700 Hz.

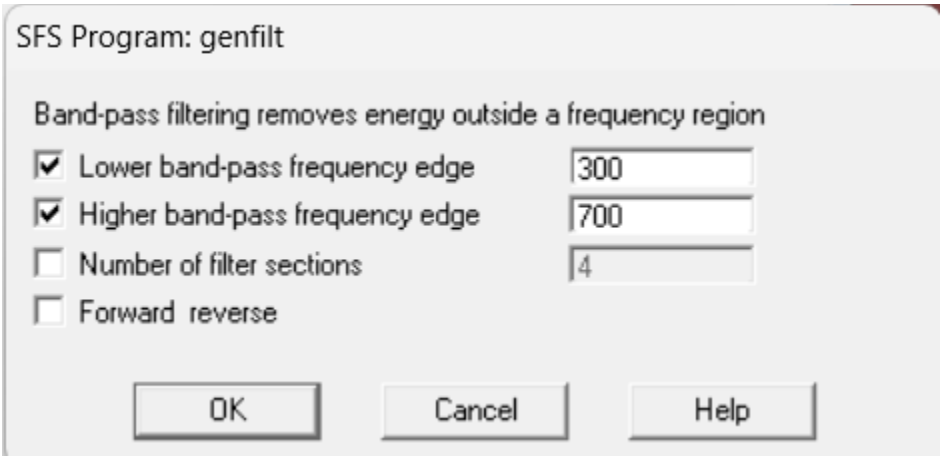


Figure 70: Band-pass frequencies for F1 'ea'

For F2=1500Hz:

Lower band-pass frequency edge: 1500 Hz - 200 Hz = 1300 Hz.

Higher band-pass frequency edge: 1500 Hz + 200 Hz = 1700 Hz.

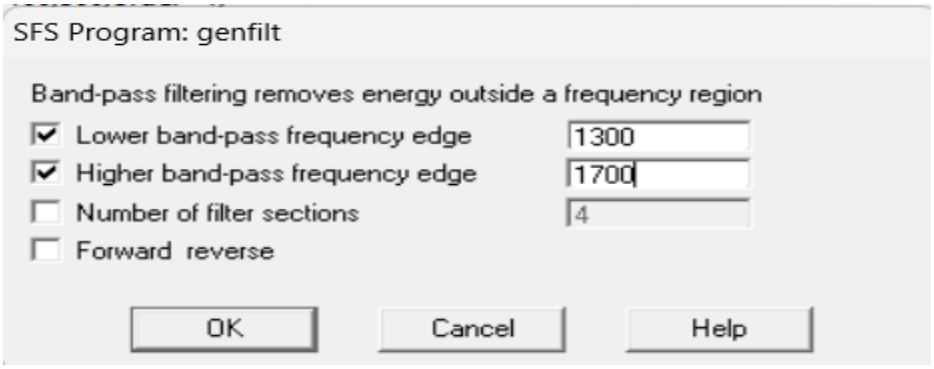


Figure 71: Band-pass frequencies for F2 'ea'

We write a matlab code to make the sounds: a, u and ea.

```
part7.m
 1      % Mohammad Abu Shams 1200549.
 2      % Mohammed Owda 1200089.
 3      % Section1.
 4
 5 -    file1 = 'a500_900.wav';
 6 -    file2 = 'a900_1300.wav';
 7 -    fileNameOut = 'vowel_A.wav';  % Output WAV file.
 8
 9      % Load the individual WAV files.
10 -    [formant1, fs1] = audioread(file1);
11 -    [formant2, fs2] = audioread(file2);
12
13      % Ensure both signals have the same sample rate and length.
14 -    if fs1 ~= fs2
15 -        error('Sample rates of the two files do not match.');
16 -    end
17
18 -    min_length = min(length(formant1), length(formant2));
19 -    formant1 = formant1(1:min_length);
20 -    formant2 = formant2(1:min_length);
21
22      % Sum the two signals sample-by-sample.
23 -    outSigWav = formant1 + formant2;
24
25      % Normalize the final signal to prevent clipping.
26 -    outSigWav = outSigWav / max(abs(outSigWav));
27
28      % Save the final signal.
29 -    audiowrite(fileNameOut, outSigWav, fs1);
30
31      % Play.
32 -    sound(outSigWav, fs1);
33
34
Command Window
>> part7
>>
```

Figure 72: Vowel a code in matlab

part7_u.m

```
% Mohammad Abu Shams 1200549.
% Mohammed Owda 1200089.
% Section1.

file1 = 'u100_500.wav';
file2 = 'u600_1000.wav';
fileNameOut = 'vowel_U.wav';  % Output WAV file.

% Load the individual WAV files.
[formant1, fs1] = audioread(file1);
[formant2, fs2] = audioread(file2);

% Ensure both signals have the same sample rate and length.
if fs1 ~= fs2
    error('Sample rates of the two files do not match.');
end

min_length = min(length(formant1), length(formant2));
formant1 = formant1(1:min_length);
formant2 = formant2(1:min_length);

% Sum the two signals sample-by-sample.
outSigWav = formant1 + formant2;


% Normalize the final signal to prevent clipping.
outSigWav = outSigWav / max(abs(outSigWav));

% Save the final signal.
audiowrite(fileNameOut, outSigWav, fs1);

% Play.
sound(outSigWav, fs1);
```

Command Window

```
>> part7_u
>>
```

Figure 73: Vowel u code in matlab

part7ea.m

```
% Mohammad Abu Shams 1200549.
% Mohammed Owda 1200089.
% Section1.

file1 = 'ea300_700.wav';
file2 = 'ea1300_1700.wav';
fileNameOut = 'vowel_ea.wav';  % Output WAV file.

% Load the individual WAV files.
[formant1, fs1] = audioread(file1);
[formant2, fs2] = audioread(file2);

% Ensure both signals have the same sample rate and length.
if fs1 ~= fs2
    error('Sample rates of the two files do not match.');
end

min_length = min(length(formant1), length(formant2));
formant1 = formant1(1:min_length);
formant2 = formant2(1:min_length);

% Sum the two signals sample-by-sample.
outSigWav = formant1 + formant2;

% Normalize the final signal to prevent clipping.
outSigWav = outSigWav / max(abs(outSigWav));

% Save the final signal.
audiowrite(fileNameOut, outSigWav, fs1);

% Play.
sound(outSigWav, fs1);
```

Command Window

```
>> part7ea
>>
```

Figure 74: Vowel ea code in matlab

These wav files was attached in the folder part7 wav.

Part 8 – Formant synthesis in SFS

Deliverable (a):

This is the cross section for vowel ‘a’ in ‘ah’.

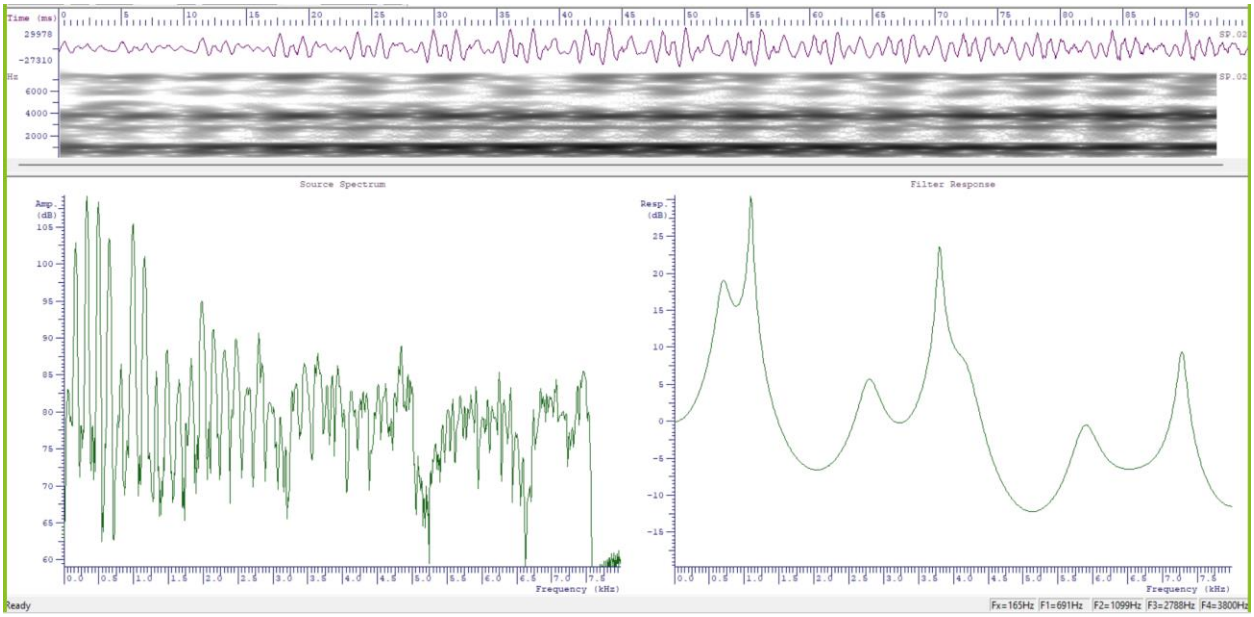


Figure 75: the cross section for vowel ‘a’ in ‘ah’

From the Cross section we found that the frequency F1=691Hz and frequency F2=1099Hz.

We plot the waveform and wide spectrogram for (original, synthesized original and synthesized modified).

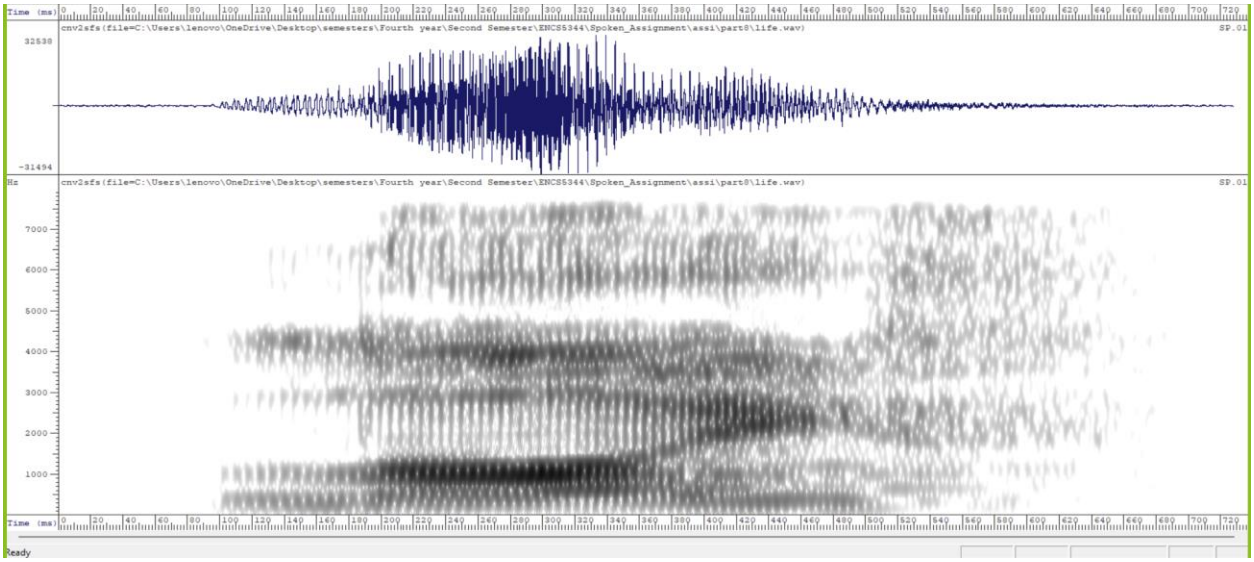


Figure 76: Waveform and Wide spectrogram for original life

life_format.txt

File Edit View

```
! item=12.01 file=C:\Users\lenovo\OneDrive\Desktop\semesters\Fourth year\Second Semester\ENCS5344\Spoken_Assignment\assi\part8\part8life.sfs
! posn size v   gain;   F1   B1 A1;   F2   B2 A2;   F3   B3 A3;   F4   B4 A4;   F5   B5 A5;
!
  10.0 20.0 0   67.7; 1929  374 58; 3742  368 57; 5866  345 55;    0    0  0;    0    0  0;
  20.0 20.0 0   66.9; 1670  445 59; 3315 1113 53; 4520  720 53; 6330  339 52;    0    0  0;
  30.0 20.0 0   68.2; 1430  857 57; 2780  513 57; 4887  474 53; 6436  391 51;    0    0  0;
  40.0 20.0 1   68.3; 2667  263 43; 3681  195 38; 5499  217 32; 6457  179 31; 7100  184 26;
  50.0 20.0 0   69.5; 1414  457 58; 3186  429 55; 4893  420 53; 6628  299 52;    0    0  0;
  60.0 20.0 0   70.2; 1616  520 58; 3135  700 54; 4820  480 54; 6615  376 51;    0    0  0;
  70.0 20.0 0   68.2;  710  762 55; 2725  435 59; 4350  619 56; 6094  640 52;    0    0  0;
  80.0 20.0 1   68.0;  482  155 59; 2230  172 46; 4285  116 44; 5562  184 34; 7038  169 28;
  90.0 20.0 0   72.4;  661  532 58; 2411  436 57; 4274  115 66; 6288  391 53;    0    0  0;
 100.0 20.0 1   82.5;  366   47 81; 1043   67 66; 4390  146 44; 6344  142 33; 7193  101 29;
 110.0 20.0 1   87.5;  379   27 89; 1043   49 71; 2836   70 54; 3948   87 54; 4341   50 56;
 120.0 20.0 1   90.0;  377   27 91; 1047   51 73; 2895  119 53; 4000   91 59; 4310   40 64;
 130.0 20.0 1   90.7;  374   25 93; 1047   38 77; 3981   82 61; 4322   18 72; 6345   93 46;
 140.0 20.0 1   92.3;  356   28 93; 1053   31 79; 2924  107 58; 4073  114 64; 4286   28 74;
 150.0 20.0 1   93.6;  344   39 93; 1027   42 78; 2918  127 58; 4026  106 64; 4293   22 75;
 160.0 20.0 1   94.7;  327   30 96;  993   34 80; 2886  103 59; 3983   72 64; 4332   44 66;
 170.0 20.0 1   95.4;  338   38 95; 1008   19 87; 2933   38 69; 4252   37 70; 6407   95 48;
 180.0 20.0 1   94.2;  383   70 90; 1013   17 90; 2962   86 65; 4271   45 70; 6357   75 51;
 190.0 20.0 0   96.2;  936  128 93; 3226  433 80; 4217   57 95; 6348  138 77;    0    0  0;
 200.0 20.0 1  100.4;  516   97 94; 1077   19 100; 2871  110 71; 4264   89 70; 6475   71 62;
 210.0 20.0 1  102.6; 1086   17 104; 2894   73 73; 3895   99 70; 4424   93 68; 7263   96 58;
 220.0 20.0 1  104.3;  688  132 94; 1078   23 105; 2884   78 76; 5757  103 61; 7301  127 54;
 230.0 20.0 1  105.3;  729  167 95; 1062   33 104; 2842  100 74; 4158  133 71; 7303   58 60;
 240.0 20.0 1  105.9; 1057   41 104; 2776  122 72; 4059   60 79; 5733   94 62; 6464  158 58;
 250.0 20.0 1  106.3; 1040   31 106; 2882   54 79; 3980   97 78; 5710  135 60; 7265  125 55;
 260.0 20.0 1  106.4; 1044   31 106; 2874   67 77; 3901   55 82; 5740   97 66; 7200  101 57;
 270.0 20.0 1  106.8; 1026   20 108; 2916   48 82; 3888   39 87; 4250  149 73; 5834  143 64;
 280.0 20.0 1  107.1; 1005   13 110; 2935   67 80; 3873   71 86; 5751  130 63; 7198  118 56;
 290.0 20.0 1  108.0; 1016   17 111; 2962  124 73; 3937   40 87; 5734  101 64; 7257  162 55;
 300.0 20.0 1  108.9; 1044   23 110; 2959  117 75; 4037   89 79; 5757   94 65; 7283   80 60;
 310.0 20.0 1  108.4;  749   94 99; 1036   18 110; 4020   79 81; 5766   92 65; 7306   53 66;
 320.0 20.0 1  107.5;  776   72 100; 1105   24 107; 4018  115 79; 5823   76 70; 7320   55 64;
 330.0 20.0 1  106.5;  749   61 101; 1141   25 105; 4030   82 80; 5827   61 70; 7335   54 63;
 340.0 20.0 1  105.2;  728   57 101; 1152   30 103; 4084  101 77; 5826   67 65; 7343   62 64;
 350.0 20.0 1  103.4;  714   77 98; 1144   43 99; 2831   91 80; 4083  136 74; 5826   62 65;
 360.0 20.0 1  101.2; 1235  109 91; 2745   93 82; 4119  103 73; 5855   77 66; 7284   99 55;
 370.0 20.0 1   98.7; 2712  111 81; 3539   62 81; 4127   72 73; 5840   59 67; 7127  140 53;
```

Figure 77: Original Formant data

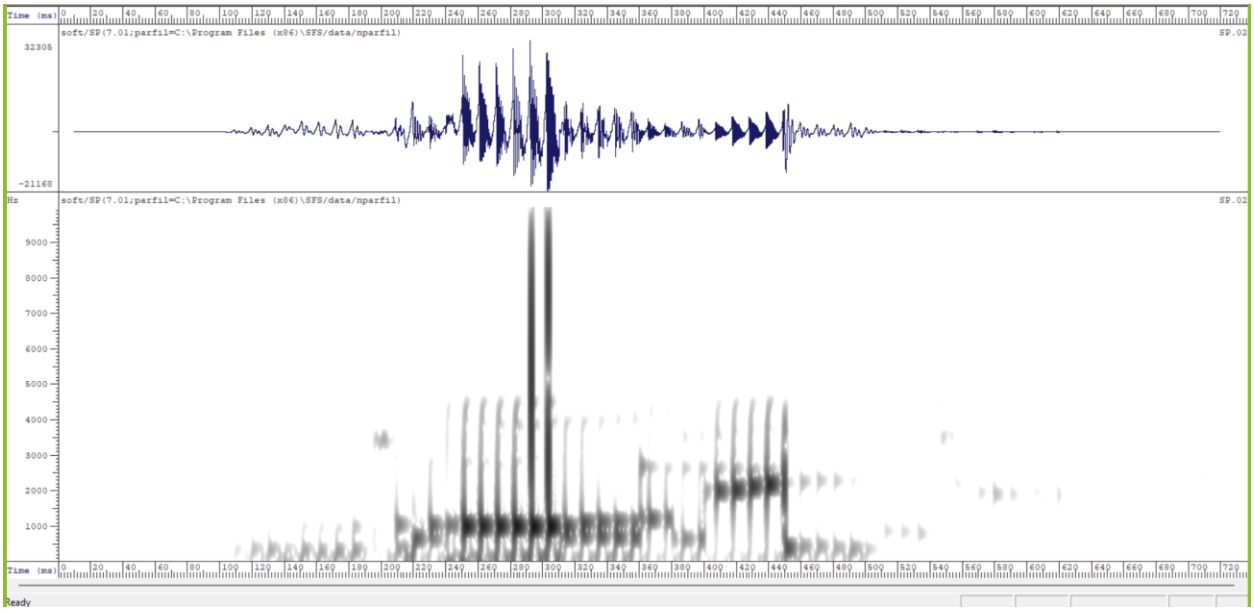


Figure 78: Waveform and Wide spectrogram for synthesized original life

From life we found the ‘i’ time it from 210-560 ms as below

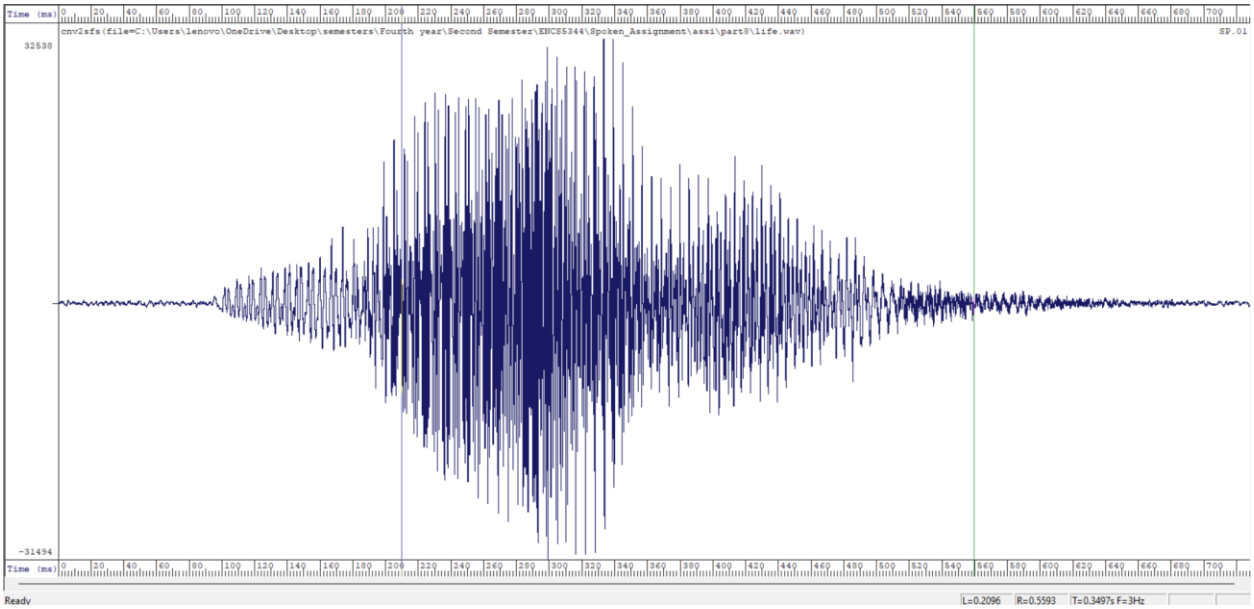
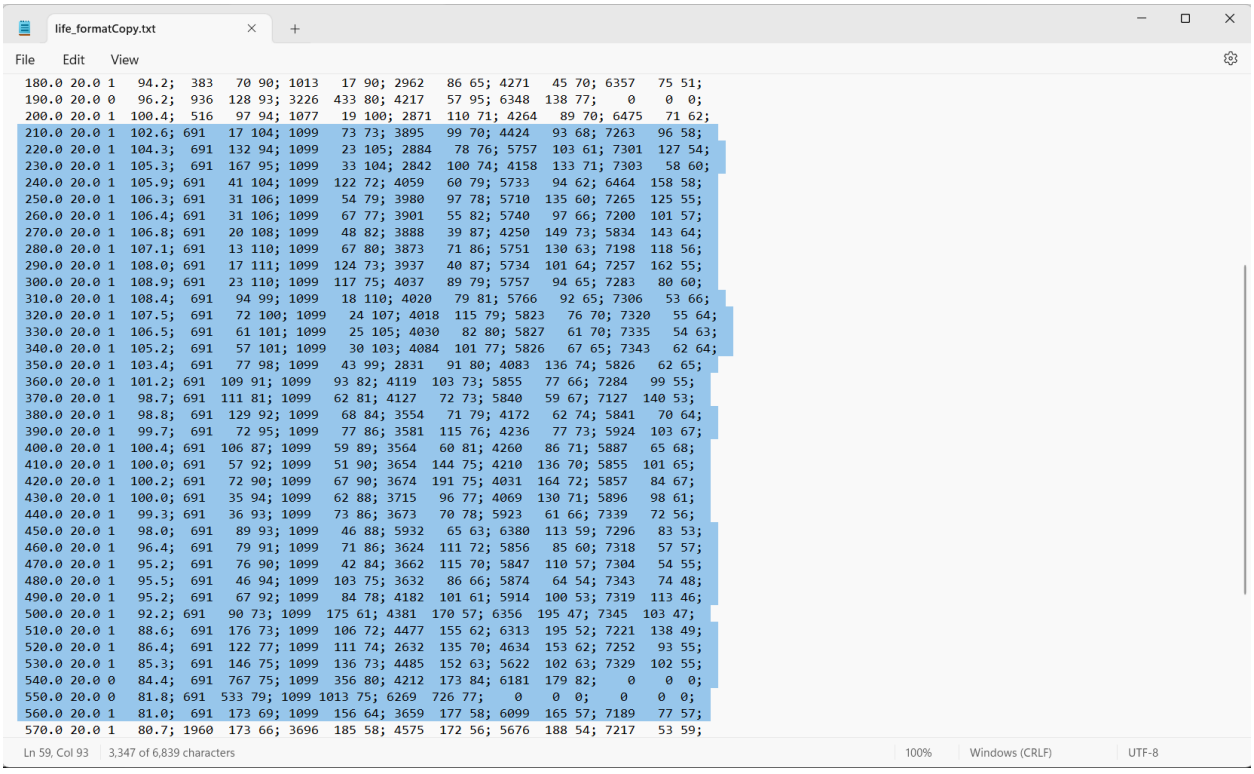


Figure 79: vowel 'i' interval

We put the values of F1=691 and F2=1099 in this file from 210-560.



life_formatCopy.txt

File Edit View

```
180.0 20.0 1   94.2;  383   70 90; 1013   17 90; 2962   86 65; 4271   45 70; 6357   75 51;
190.0 20.0 0   96.2;  936  128 93; 3226  433 80; 4217   57 95; 6348  138 77;    0    0  0;
200.0 20.0 1  100.4;  516   97 94; 1077   19 100; 2871  110 71; 4264   89 70; 6475   71 62;
210.0 20.0 1  102.6; 691   17 104; 1099   73 73; 3895   99 70; 4424   93 68; 7263   96 58;
220.0 20.0 1  104.3;  691  132 94; 1099   23 105; 2884   78 76; 5757  103 61; 7301  127 54;
230.0 20.0 1  105.3;  691  167 95; 1099   33 104; 2842  100 74; 4158  133 71; 7303   58 60;
240.0 20.0 1  105.9; 691   41 104; 1099  122 72; 4059   60 79; 5733   94 62; 6464  158 58;
250.0 20.0 1  106.3; 691   31 106; 1099   54 79; 3980   97 78; 5710  135 60; 7265  125 55;
260.0 20.0 1  106.4; 691   31 106; 1099   67 77; 3901   55 82; 5740   97 66; 7200  101 57;
270.0 20.0 1  106.8; 691   20 108; 1099   48 82; 3888   39 87; 4250  149 73; 5834  143 64;
280.0 20.0 1  107.1; 691   13 110; 1099   67 80; 3873   71 86; 5751  130 63; 7198  118 56;
290.0 20.0 1  108.0; 691   17 111; 1099  124 73; 3937   40 87; 5734  101 64; 7257  162 55;
300.0 20.0 1  108.9; 691   23 110; 1099  117 75; 4037   89 79; 5757   94 65; 7283   80 60;
310.0 20.0 1  108.4;  691   94 99; 1099   18 110; 4020   79 81; 5766   92 65; 7306   53 66;
320.0 20.0 1  107.5;  691   72 100; 1099   24 107; 4018  115 79; 5823   76 70; 7320   55 64;
330.0 20.0 1  106.5;  691   61 101; 1099   25 105; 4030   82 80; 5827   61 70; 7335   54 63;
340.0 20.0 1  105.2;  691   57 101; 1099   30 103; 4084  101 77; 5826   67 65; 7343   62 64;
350.0 20.0 1  103.4;  691   77 98; 1099   43 99; 2831   91 80; 4083  136 74; 5826   62 65;
360.0 20.0 1  101.2; 691  109 91; 1099   93 82; 4119  103 73; 5855   77 66; 7284   99 55;
370.0 20.0 1   98.7; 691  111 81; 1099   62 81; 4127   72 73; 5840   59 67; 7127  140 53;
380.0 20.0 1   98.8;  691  129 92; 1099   68 84; 3554   71 79; 4172   62 74; 5841   70 64;
390.0 20.0 1   99.7;  691   72 95; 1099   77 86; 3581  115 76; 4236   77 73; 5924  103 67;
400.0 20.0 1  100.4; 691  106 87; 1099   59 89; 3564   60 81; 4260   86 71; 5887   65 68;
410.0 20.0 1  100.0; 691   57 92; 1099   51 90; 3654  144 75; 4210  136 70; 5855  101 65;
420.0 20.0 1  100.2; 691   72 90; 1099   67 90; 3674  191 75; 4031  164 72; 5857   84 67;
430.0 20.0 1  100.0; 691   35 94; 1099   62 88; 3715   96 77; 4069  130 71; 5896   98 61;
440.0 20.0 1   99.3; 691   36 93; 1099   73 86; 3673   70 78; 5923   61 66; 7339   72 56;
450.0 20.0 1   98.0;  691   89 93; 1099   46 88; 5932   65 63; 6380  113 59; 7296   83 53;
460.0 20.0 1   96.4;  691   79 91; 1099   71 86; 3624  111 72; 5856   85 60; 7318   57 57;
470.0 20.0 1   95.2;  691   76 90; 1099   42 84; 3662  115 70; 5847  110 57; 7304   54 55;
480.0 20.0 1   95.5;  691   46 94; 1099  103 75; 3632   86 66; 5874   64 54; 7343   74 48;
490.0 20.0 1   95.2;  691   67 92; 1099   84 78; 4182  101 61; 5914  100 53; 7319  113 46;
500.0 20.0 1   92.2; 691   90 73; 1099  175 61; 4381  170 57; 6356  195 47; 7345  103 47;
510.0 20.0 1   88.6;  691  176 73; 1099  106 72; 4477  155 62; 6313  195 52; 7221  138 49;
520.0 20.0 1   86.4;  691  122 77; 1099  111 74; 2632  135 70; 4634  153 62; 7252   93 55;
530.0 20.0 1   85.3;  691  146 75; 1099  136 73; 4485  152 63; 5622  102 63; 7329  102 55;
540.0 20.0 0   84.4;  691  767 75; 1099  356 80; 4212  173 84; 6181  179 82;    0    0  0;
550.0 20.0 0   81.8; 691  533 79; 1099 1013 75; 6269  726 77;    0    0  0;    0    0  0;
560.0 20.0 1   81.0;  691  173 69; 1099  156 64; 3659  177 58; 6099  165 57; 7189   77 57;
570.0 20.0 1   80.7; 1960  173 66; 3696  185 58; 4575  172 56; 5676  188 54; 7217   53 59;
```

Ln 59, Col 93 | 3,347 of 6,839 characters | 100% | Windows (CRLF) | UTF-8

Figure 80: Modified Formant data

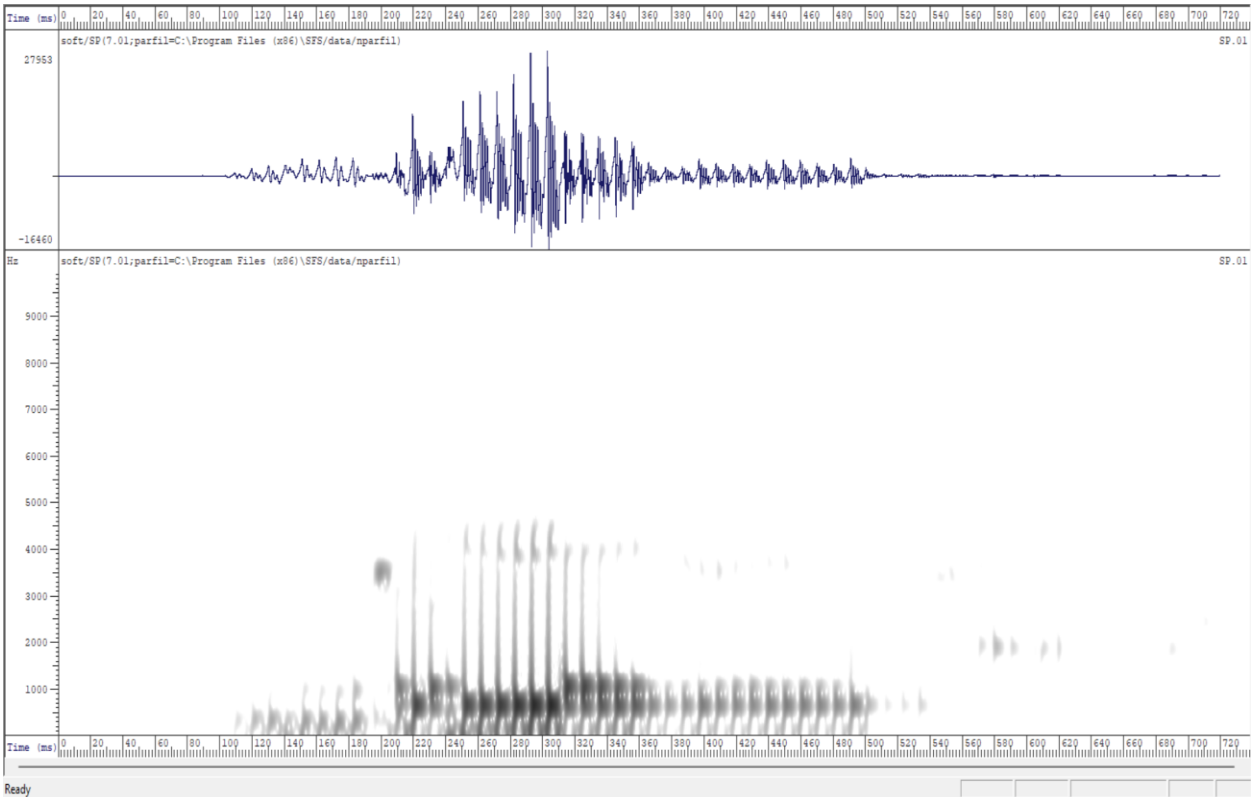


Figure 81: Waveform and Wide spectrogram for synthesized modified life

The synthesized speech using modified formant data will sound different from the original word "life" (/laIf/). Instead of transitioning from the /a/ to the /I/ sound in the diphthong, it keeps a constant /a/ vowel sound. This change can make the speech sound less natural and harder to understand, producing a longer /a/ sound or a different word.

Deliverable (b):

To change the speaker's voice in a formant-based speech synthesizer, start by adjusting the formant frequencies and bandwidths so they match the target speaker's voice. Change the pitch (or fundamental frequency) to match the pitch range that the new speaker usually uses. Adjust the loudness and dynamics of the formants to reflect the speaker's loudness and expressiveness. Add variations in voice quality like breathiness or harshness through changes in the sound spectrum. Lastly, adjust the timing and length of speech sounds to mimic the unique speaking style of the new speaker.

These wav files and formants files was attached in the folder part8 wav.