Faculty of Engineering & Technology Electrical & Computer Engineering Department

Spoken Language Processing- ENCS5344

Assignment #1

Prepared by:

Maha Maher Mali	1200746
Ebaa Taleeb	1203073

Instructors: Dr. Abualseoud Hanani

Section: 1

Date: 20-11-2024

Table of Contents

Table of Figures 3

List Of Tables 5

Part 1: Basic Sound Analysis, Spectrogram 6

Part A: Bandwidth 6

Sound 16

Sound 27

Part B: Spectrogram 8

Sound 3 8

Sound 4 8

Part 2: Filters 10

Sinusoidal Signal 11

White -Noise Signal 11

Combine The Sine and White Noise Signals 12

Design and Implement IIR Low Pass Filter 13

Input Signal Vs Output Signal 14

Freqancy Response 15

Design and Implement FIR Filter 16

16

Input Signal Vs Output Signal 17

Freqancy Response 18

Part 3: Speech Analysis 19

Part A: Formants For Vowels 19

Maha Speech Formants 23

Ebaa Speech Formants 26

Native Vs Our Vowel Quadrilateral 30

Part B: Source and Filter analysis 32

A in dark 32

S in suit 32

Part C: Fundamental Frequency 33

Maha Speech: a in dark 33

Ebaa Speech: 34

Ibaa Speech: e in she 34
Part 4: Speech Analysis 35
Part 5: Sub-Word Level Connection 37
Part 6: Phone-Level Concatenation 38
Part 7: Generation of Vowel Sounds Using the Source-Filter Model 41
7.1 Generating Pulse Train 41
7.2 Generating Bandpass Filters 42
7.3 Exporting Filtered Signals 47
Part 8: Formant synthesis in SFS 48
Part A 48
Part B 51

Table of Figures

Figure 1:Sound 1 wide band spectrogram 6
Figure 2: Sound 2 wide band spectrogram 7
Figure 3:Sound 3 narrow and wide-band spectrogram 8
Figure 4:Sound 4 narrow and wide-band spectrogram 8
Figure 5: generate a sinusoidal signal 10
Figure 6: generate a white noise signal 10
Figure 7:Display Sinusoidal Signal Waveforam , Wideband and Narrwoband Spectogram in SFS 11
Figure 8 : Display White-Noise Signal Waveform, Wideband and Narrowband Spectrogram in SFS 11
Figure 9: Combined Signal in SFS(sin with white noise signal) 12
Figure 10: Input Signal Vs Output Signal after applying IIR Low Pass Filter 14
Figure 11:Frequency response 15
Figure 12:Display Sample1 Spectrogram 19
Figure 13:Source Spectrum, Filter Response, and Signal Spectrum of Sample1 19
Figure 14:E in She (‘sample1.wav’) 20
Figure 15:A in had(‘sample1.wav’) 20
Figure 16:U in suite(‘sample1.wav’) 21
Figure 17:A in Dark(‘sample1.wav’) 21
Figure 18:Display Maha Speech Spectrogram 23
Figure 19: Source Spectrum, Filter Response, and Signal Spectrum of Maha Speech 23
Figure 20:e in she (Maha Speech) 24
Figure 21:a in had (Maha Speech) 24
Figure 22:a in dark (Maha Speech) 24
Figure 23:u in suit (Maha Speech) 25
Figure 24: Maha Speech F1 Vs F2 25
Figure 25: Display Ebaa Speech Spectrogram 26
Figure 26: Source Spectrum, Filter Response, and Signal Spectrum of Ebaa Speech 26
Figure 27: :e in she (Ebaa Speech) 27
Figure 28::a in had (Ebaa Speech) 27
Figure 29::a in dark (Ebaa Speech) 28
Figure 30::u in suit (Ebaa Speech) 28
Figure 31:Ebaa Speech F1 VS F2 29
Figure 32: Native Vowel Quadrilateral 30
Figure 33:Sample1 Vowel Quadrilateral 30
Figure 34: Maha Vowel Quadrilateral 31
Figure 35: Ebaa Vowel Quadrilateral 31
Figure 36:A in Dark source spectrum, filter response and signal spectrum 32
Figure 37:S in Suit source spectrum, filter response and signal spectrum 32
Figure 38:Maha Fundamental Frequency 33
Figure 39: Ebaa Fundamental Frequency 34
Figure 40:AW1.wav spectrogram 35
Figure 41:'capacity' spectrogram 35
Figure 42: /k/ Phoneme Waveform 38
Figure 43: /@/ Phoneme Waveform 38
Figure 44:/t/ Phoneme Waveform 39
Figure 45:waveform and wideband spectrogram to the file “cat.wav" 40
Figure 46:generation of pulse train signal specification 41
Figure 47:pulse train signal spectrogram 41
Figure 48: /a:/ F1=700Hz generations 42

Figure 49:/a:/ F1=700Hz band pass filter..... 42
Figure 50: /a:/ F2=1100Hz generations 42
Figure 51:/a:/ F2=1100Hz band pass filter..... 43
Figure 52: /u:/ F1=300Hz generations 43
Figure 53: /u:/ F1=300Hz band pass filter 43
Figure 54: **/u:/** F2=800Hz generation 44
Figure 55:/u:/ F2=800Hz band pass filter 44
Figure 56: /ə/ F1=500Hz generation..... 44
Figure 57:/ə/ F1=500Hz band pass filter..... 45
Figure 58: /ə/ F2=1500Hz generation..... 45
Figure 59: /ə/ F2=1500Hz band pass filter..... 45
Figure 60: /a:/ Sound..... 47
Figure 61:/ə/ sound 47
Figure 62:/u:/Sound 47
Figure 63:“ah” cross section..... 48
Figure 64: “life” waveform and wideband spectrogram 48
Figure 65:original parameter 49
Figure 66: synthesized original 49
Figure 67: “I” from life word 50
Figure 68: f1 & f2..... 50
Figure 69: synthesized modified..... 51

List Of Tables

Table 1:Sample1.wav F1, F2, F3 frequencies 22
Table 2: Maha Speech F1, F2, F3 frequencies 25
Table 3:Ebaa Speech F1, F2, F3 frequencies 28
Table 4 : start time and end time for each phoneme. 36

Part 1: Basic Sound Analysis, Spectrogram

Part A: Bandwidth

Sound 1

A spectrogram is a graph that represents a two-dimensional shapes matrix that shows a signal's time in x-axis and frequency in y-axis. A spectrogram shows how a signal's frequency varies over time. The sound's lower and higher frequency limits can be determined with a wide-band spectrogram. The spectrogram represents the frequency content of the sound signal through time.

The wide-band spectrogram is a useful tool for measuring the bandwidth of the sound wave. The frequency range included in the sound can be determined by viewing the spectrogram. Also, the spectrogram represents the frequency content of the sound play across time, with darker regions indicating more energy at various frequencies and times. Brighter areas, on the other hand, represent locations with less energy.

Figure 1 shows the Sound1.wav, the waveform in blue color and wide spectrogram in gray color, the wide spectrogram place between 250 Hz to 10900 Hz (±500 Hz). So, from figure 1 we can calculate the Bandwidth according this equation:

$$Bandwidth = Higher\ Frequency - Lower\ Frequency$$
$$= 10900 - 250$$
$$= \mathbf{10650\ Hz}$$

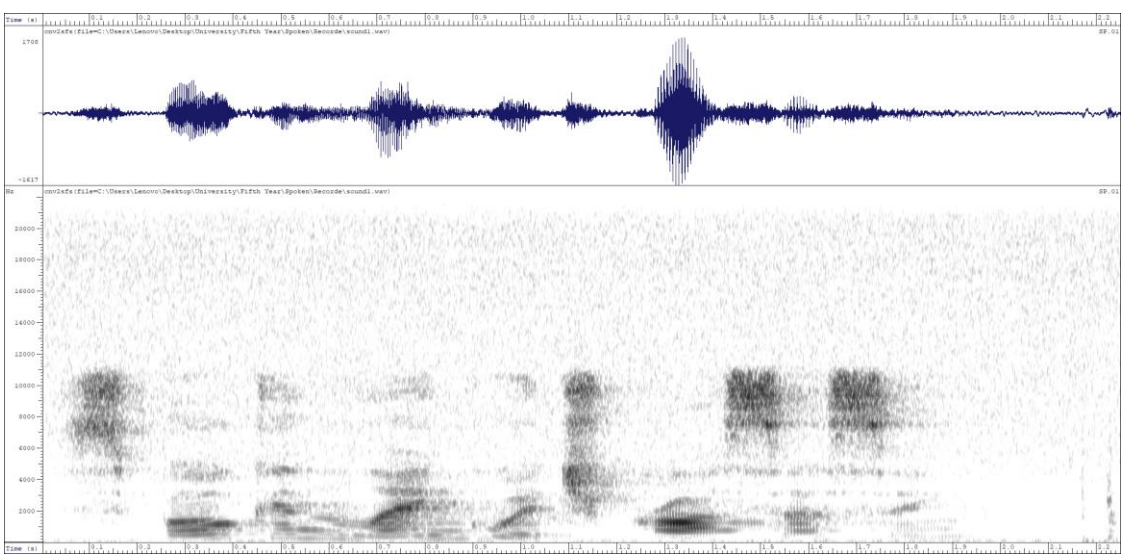


Figure 1:Sound 1 wide band spectrogram

Sound 2

Figure 2 shows the Sound2.wav, the waveform in blue color and wide spectrogram in gray color, the wide spectrogram place between 100 Hz to 6350 Hz (±500 Hz). So, from figure 2 we can calculate the Bandwidth according this equation:

$$Bandwidth = Higher\ F\text{requency} - Lower\ Frequency$$
$$= 6350 - 100$$
$$= \mathbf{6250\ \ Hz}$$

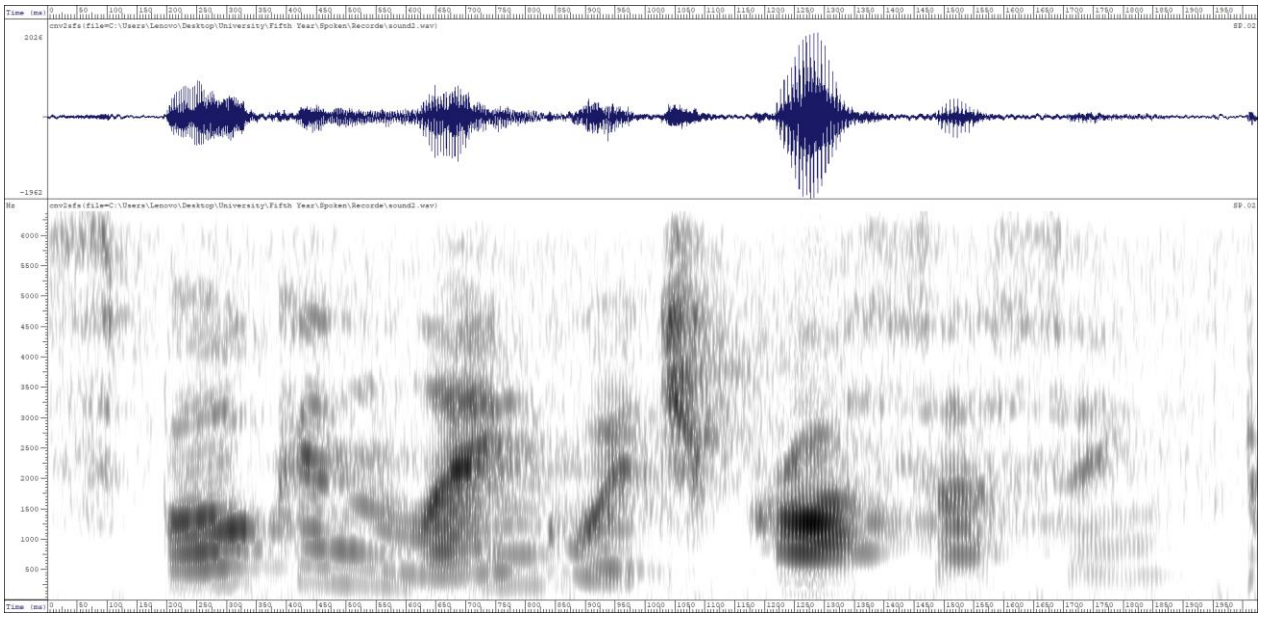


Figure 2: Sound 2 wide band spectrogram

Part B: Spectrogram

Sound 3

As shown in Figure 3, the spectrogram of sound3.wav is bounded between nearly 200Hz Hz and 11 kHz.

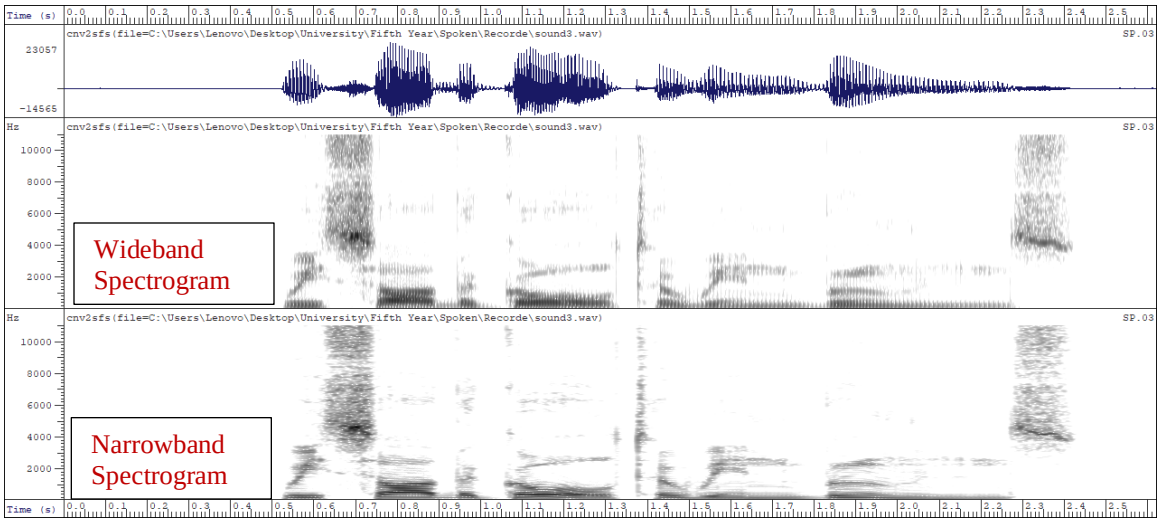


Figure 3:Sound 3 narrow and wide-band spectrogram

Sound 4

As shown in Figure 4, the spectrogram of sound4.wav is bounded between nearly 200 Hz and 11 kHz.

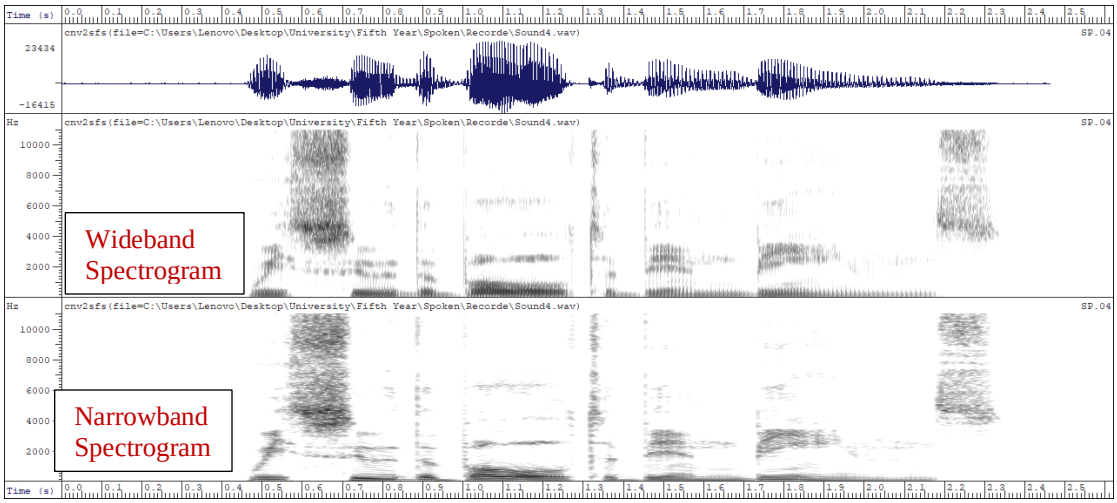


Figure 4:Sound 4 narrow and wide-band spectrogram

The wide-band spectrogram is produced with shorter time segments, resulting in excellent temporal resolution but a decreased frequency resolution. This makes it possible to clearly identify phoneme transitions and speech event timings. Vertical lines on the wide-band spectrogram represent glottal pulses, which are most apparent in spoken sounds. This makes it valuable for analyzing the boundaries and dynamics of speech sounds.

As a result, the narrow-band spectrogram uses longer time segments, giving higher frequency resolution but lower temporal resolution. This depiction shows the harmonic structure of vocal sounds. Individual harmonics, representing the fundamental frequency and its multiples, are visible, making it useful for studying steady-state vowels and pitch.

When comparing the two types of spectrograms, the wide-band spectrogram is better at detecting phoneme boundaries and fast sounds like bursts or formant transitions. The narrow-band spectrogram, on the other hand, is more suited for evaluating pitch and harmonic content, both of which are necessary for studying extended sounds or vowel features.

The spectrograms show significant variations in the articulation of vowels and consonants between the two sentences. For example, the sounds "saw" versus "sue" and "goal" against "bowl" have differing frequency and energy distributions. These articulation differences provide distinct spectrogram fingerprints for each phoneme, displaying how the creation of different sounds affects the overall frequency content of the speech signal. Moreover, the pitch and energy distribution across frequencies change according to the different phonemes in each sentence. For example, the "s" sound in "saw" has a distinct frequency pattern from the "s" sound in "sue". The wide-band spectrogram successfully captures fast changes in the signal, allowing us to better see the transitions between words and phonemes. This high temporal resolution helps to distinguish between quick variations in energy and frequency that occur during speech.

Part 2: Filters

At first, we generated a sinusoidal signal with this specification:

- Signal Type: sine
- Frequency=1000Hz
- Duration= 3s
- Sampling Frequency (Fs)= 16000Hz

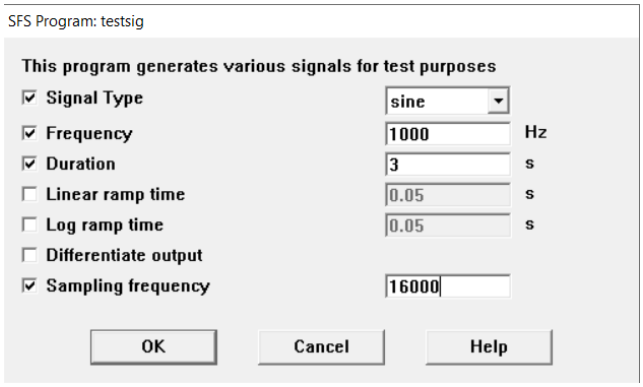


Figure 5: generate a sinusoidal signal

Then we generated a white noise signal with this specification:

- Signal Type: white-noise
- Frequency=1000Hz
- Duration= 3s
- Sampling Frequency (Fs)= 16000Hz

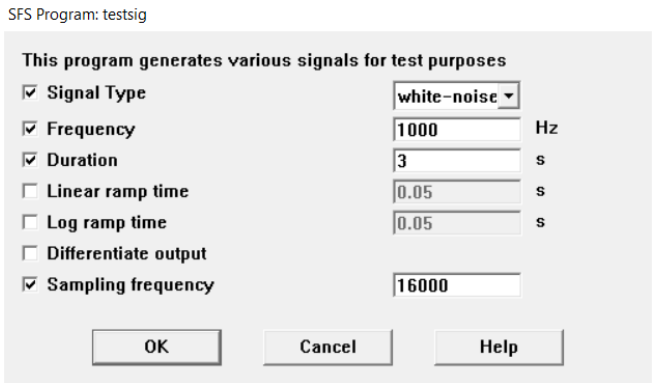


Figure 6: generate a white noise signal

Sinusoidal Signal

After we generated the Signal we import the Sine.wave file which includes the generated signal of sinusoidal, and after that we display the waveforem, narwoband, and wideband of the generated signal in SFS as shown in below figure.

Figure 7:Display Sinusoidal Signal Waveform, Wideband and Narrowband Spectrogram in SFS

White -Noise Signal

Below figure display the waveform, narrowband, and wideband of the generated signal for the white noise signal.

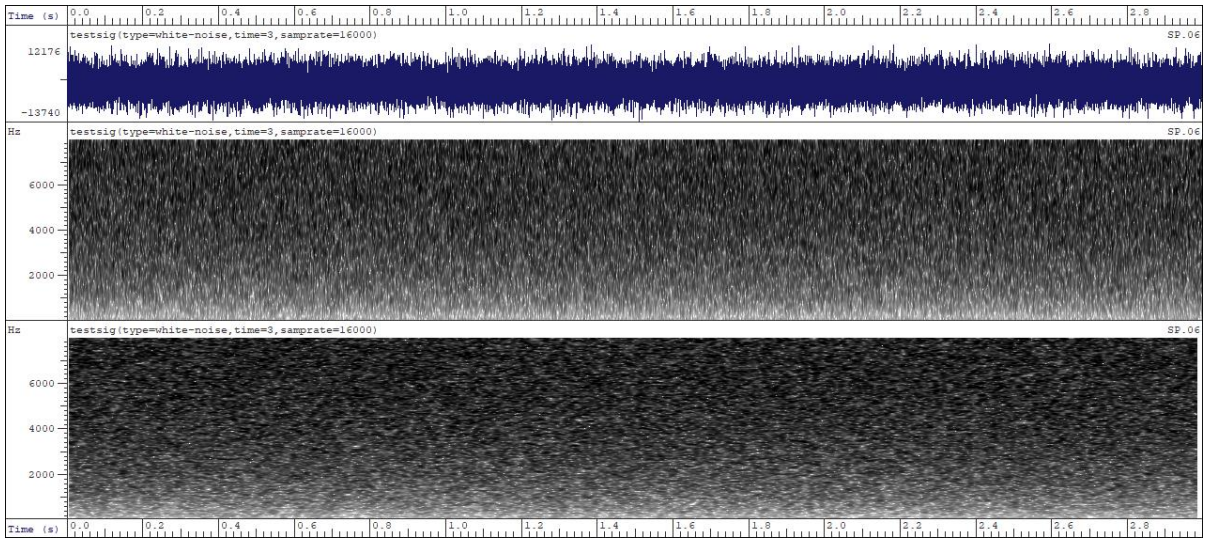


Figure 8 : Display White-Noise Signal Waveform, Wideband and Narrowband Spectrogram in SFS

Combine The Sine and White Noise Signals

The below Code reads the wav files of generated signals which is: Sine.wav, and White-Noise.wav, then make sure the generated signals have the same length. After that combines the generated signals and writes the combined signal to a new wave file name **"Combined_Signal.wav"**.

Code 1:Code for Part 2 Combined Signal

```
%#################### Adding Two Signals ####################
% Maha Mali
%Ebaa Taleeb
% Read the wave files
[sine, fs] = audioread('Sine.wav');
[noise, ~] = audioread('White-Noise.wav');

% Ensure both signals have the same length
minLength = min(length(sine), length(noise));
sine = sine(1:minLength);
noise = noise(1:minLength);

% Add the signals
combined_signal = sine + noise;

% Write the combined signal to a new wave file
audiowrite('Combined_Signal.wav', combined_signal, fs);
```

Combined Signal in SFS: after we combined the signal, we import the "Combined_Signal.wav" file into the SFS and display the waveform, wideband, narrowband spectrogram, to make sure we combined the signal correct.

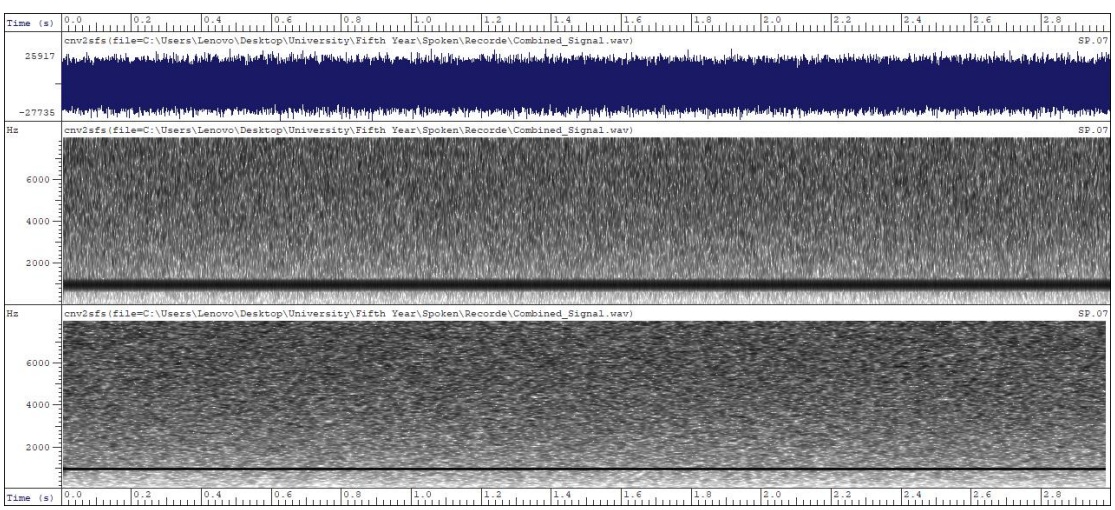


Figure 9: Combined Signal in SFS (sin with white noise signal)

Design and Implement IIR Low Pass Filter

The code below designs and applies an IIR LowPass filter to a signal that includes a 1 kHz sinusoidal wave and white noise. First, it retrieves the combined signal from a file. The IIR LowPass filter is then established using specific coefficients derived from the provided difference equation. The code shows the frequency response of the filter and applies it to the input signal. It then saves the filtered signal as a new file. It also plays the input and filtered signals for audio comparison. Finally, it shows the waveforms and

Code 2: Code for IIR LowPass Code

```
%#################### Read the input signal ####################
[inpSigWav, Fs] = audioread('Combined_Signal.wav');

%#################### Design IIR LowPass Filter #######################
% Coefficients for the IIR lowpass filter based on the difference equation y(n) = 0.99y(n-1) + x(n)
b = [1]; % Numerator coefficients (feedforward)
a = [1, -0.99]; % Denominator coefficients (feedback)

%#################### Display the filter's frequency response ###########
figure;
freqz(b, a, 512, Fs); % Display frequency response of the IIR filter
title('Frequency Response of IIR LowPass Filter');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

%#################### Apply the IIR filter #########################
% Apply the IIR filter to the input signal
filtered_signal_IIR = filter(b, a, inpSigWav);

%#################### Save the filtered signal to a file #########
% Save the output (filtered) signal to a new WAV file
audiowrite('Filtered_Signal_IIR_LowPass.wav', filtered_signal_IIR, Fs);

%#################### Listen to the input and output signals ######
% Listen to the input signal
disp('Playing input signal...');
sound(inpSigWav, Fs);
pause(length(inpSigWav)/Fs + 1); % Wait for the input signal to finish

% Listen to the output (filtered) signal
disp('Playing filtered signal...');
sound(filtered_signal_IIR, Fs);
pause(length(filtered_signal_IIR)/Fs + 1); % Wait for the output signal to finish

%#################### Plot the Waveforms #########################
figure;

% Plot the input signal waveform
subplot(2, 1, 1);
plot((1:length(inpSigWav))/Fs, inpSigWav, 'g');
title('Input Signal: Combined Sine and Noise');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Plot the output (filtered) signal waveform
subplot(2, 1, 2);
plot((1:length(filtered_signal_IIR))/Fs, filtered_signal_IIR, 'b');
title('Output Signal: Filtered (IIR LowPass)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

magnitude frequency characteristics of the input and filtered signals that show the filtering effect.

Input Signal Vs Output Signal

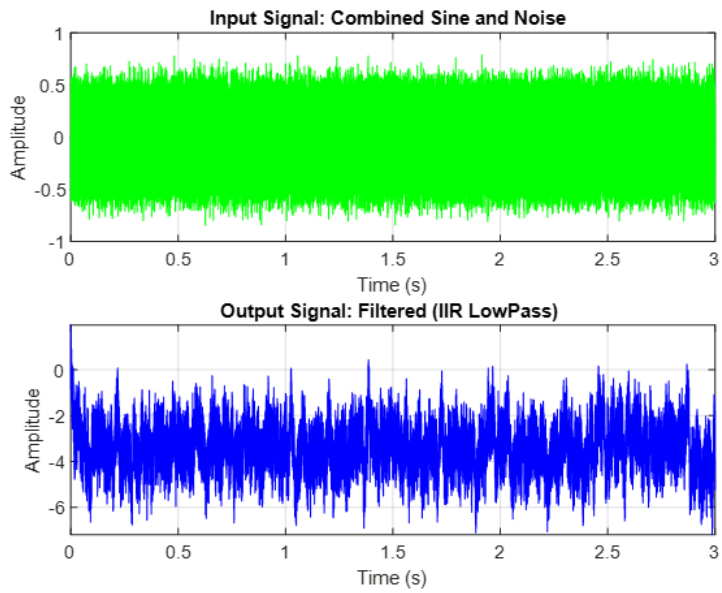


Figure 10: Input Signal Vs Output Signal after applying IIR Low Pass Filter

Input Signal: which is a 1 kHz sine wave combined with white noise. This appears by the green waveform, which varies in amplitude from around -1 to 1 over a time period of 0 to 3 seconds. The combined signal has both low-frequency (sine wave) and high-frequency (noise) components, resulting in a noisy and unpredictable waveform.

Output Signal: represented in the bottom plot, is the same signal after being processed by an IIR LowPass filter. The blue waveform shows that the high-frequency noise has been greatly reduced, resulting in a much clearer and more consistent sine wave. The amplitude of the filtered signal swings from about -7 to 2, indicating the filter's ability to reduce noise while retaining the desired low-frequency components of the original signal.

The input and output signals vary mostly in terms of noise level and sine wave quality. The input signal includes both significant noise and the sine wave, however the output signal has significantly less noise, resulting in a cleaner sine wave. This demonstrates the IIR LowPass filter's efficiency in reducing high-frequency noise while enhancing overall signal quality.

Freqancy Response

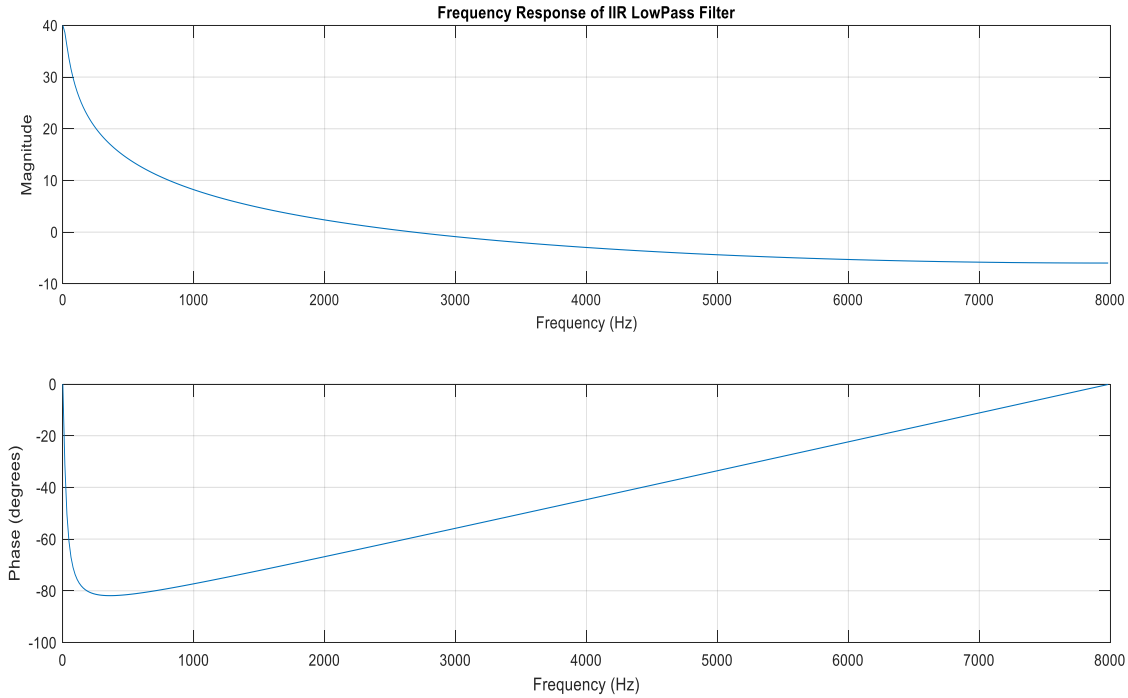


Figure 11:Frequency response for IIR Filter

Magnitude Response: Explains how the filter reduces higher frequencies, with amplitude decreasing as frequency increases.

Phase Response: This is how the filter impacts the phase at different frequencies, starting negatively and growing linearly.

Design and Implement FIR Filter

This code takes an input signal (a combined sine and noise signal) from a WAV file and adds a FIR averaging filter to it. It then saves the filtered signal as a new WAV file. The code also shows the frequency response of the FIR filter, plays both the input and output sounds for auditory comparison, and plots the waveforms of the input and filtered signals on a uniform y-axis scale for simple comparison. The filter's objective is to smooth the signal by averaging around samples, which reduces noise while keeping signal structure.

Code 3: Code for FIR

```
%#################### Read the input signal ####################
% Read the input signal (such as the combined sine and noise signal)
[inpSigWav, Fs] = audioread('Combined_Signal.wav');  % Replace with the correct file name

%#################### Define the FIR filter coefficients ####################
% Coefficients for the averaging filter (FIR filter)
b = [0.2, 0.2, 0.2, 0.2, 0.2];  % Coefficients b0, b1, b2, b3, b4
a = 1;  % FIR filters always have 'a' = 1
%#################### Display the filter's frequency response ###########
figure;
freqz(b, a, 512, Fs);  % Display frequency response of the FIR filter
title('Frequency Response of Averaging FIR Filter');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
%#################### Apply the FIR filter #########################
% Apply the FIR filter to the input signal
filtered_signal_FIR = filter(b, a, inpSigWav);
%#################### Save the filtered signal to a file #########
% Save the output (filtered) signal to a new WAV file
audiowrite('Filtered_Signal_FIR_Averaging.wav', filtered_signal_FIR, Fs);
%#################### Listen to the input and output signals ######
% Listen to the input signal
disp('Playing input signal...');
sound(inpSigWav, Fs);
pause(length(inpSigWav)/Fs + 1); % Wait for the input signal to finish
% Listen to the output (filtered) signal
disp('Playing filtered signal...');
sound(filtered_signal_FIR, Fs);
pause(length(filtered_signal_FIR)/Fs + 1); % Wait for the output signal to finish

%#################### Plot the Waveforms #########################
figure;
% Define unified y-axis limits
ymin = min(min(inpSigWav), min(filtered_signal_FIR));
ymax = max(max(inpSigWav), max(filtered_signal_FIR));

% Plot the input signal waveform
subplot(2, 1, 1);
plot((1:length(inpSigWav))/Fs, inpSigWav, 'g');
title('Input Signal: Combined Sine and Noise');
xlabel('Time (s)');
ylabel('Amplitude');
ylim([ymin ymax]); % Set unified y-axis limits
grid on;

% Plot the output (filtered) signal waveform
subplot(2, 1, 2);
plot((1:length(filtered_signal_FIR))/Fs, filtered_signal_FIR, 'b');
title('Output Signal: Filtered (Averaging)');
xlabel('Time (s)');
ylabel('Amplitude');
ylim([ymin ymax]); % Set unified y-axis limits
grid on;
```

Input Signal Vs Output Signal

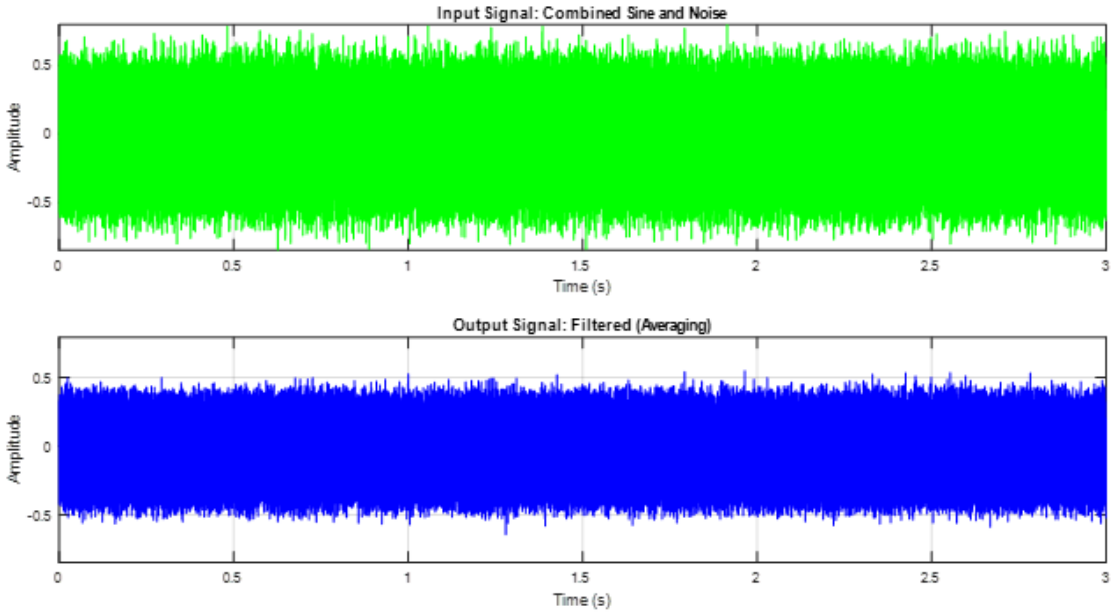


Figure 12: Input Vs Output Signal after applying FIR filter

Input Signal: Combined Sine and Noise" and presented in green, indicates the original signal as a combination of sine wave and noise over a 3-second period. The amplitude covers between -0.5 to 0.5, indicating the existence of both the sine wave and the additional noise.

Output Signal: Filtered (Averaging) displayed in blue, represents the signal after passing through the FIR averaging filter. The amplitude varies between -0.5 and 0.5 throughout the same three-second period. The filtering step smoothed out the input, minimizing noise and emphasizing the sine wave more.

The main variances between the input and output signals are noticeable. The incoming signal has more noise, producing a more unorganized waveform. In comparison, the output signal after filtering seems significantly smoother as a result of the FIR averaging filter's ability to minimize high-frequency noise. As a result, the filtered output signal has a clearer sine wave than the noisy input signal, demonstrating the efficacy of the FIR averaging filter in improving signal quality by reducing noise.

Freqancy Response

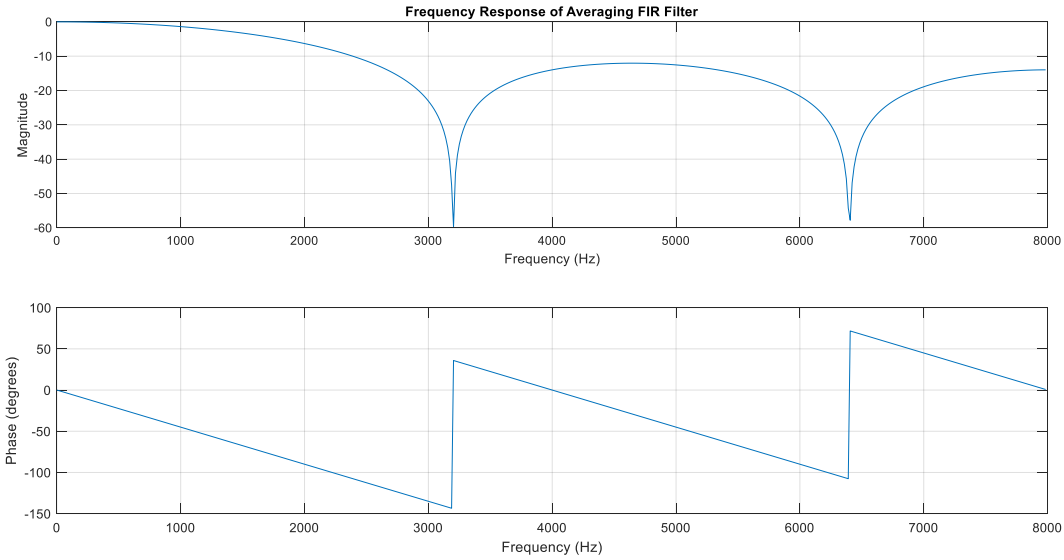


Figure 13: Frequency Response For FIR Filter

Magnitude Response:

➔Low-frequency components (near to 0 Hz) show minimal to no attenuation, indicating that they pass through the filter well.

➔The magnitude of high-frequency components decreases, indicating attenuation. This behavior is indicative of a low-pass filter, which was designed to suppress noise or undesired high-frequency signals.

➔Sharp drops in amplitude response (notch points) occur at specific frequencies where the filter significantly attenuates the signal. These dips are created by the FIR filter's averaging nature, which removes out specific frequency components.

Phase Response:

➔The phase response is nearly linear, which means that the filter delays all frequency components by the same amount of time. This feature ensures that the signal's shape remains unchanged, preserving the waveform's integrity.

➔Phase values decrease linearly with frequency, indicating a proportionate relationship between frequency and delay.

➔Sudden steps in phase, also known as phase wrapping, are artifacts of describing phase in degrees that have no effect on the filter's performance or linearity.

Part 3: Speech Analysis

Part A: Formants For Vowels

Below Figure shows the time domain, wideband and narrowband spectrogram of the speech file ‘sample1.wav’ which contains saying “She had your dark suit in greasy wash water all year” which have many vowels.

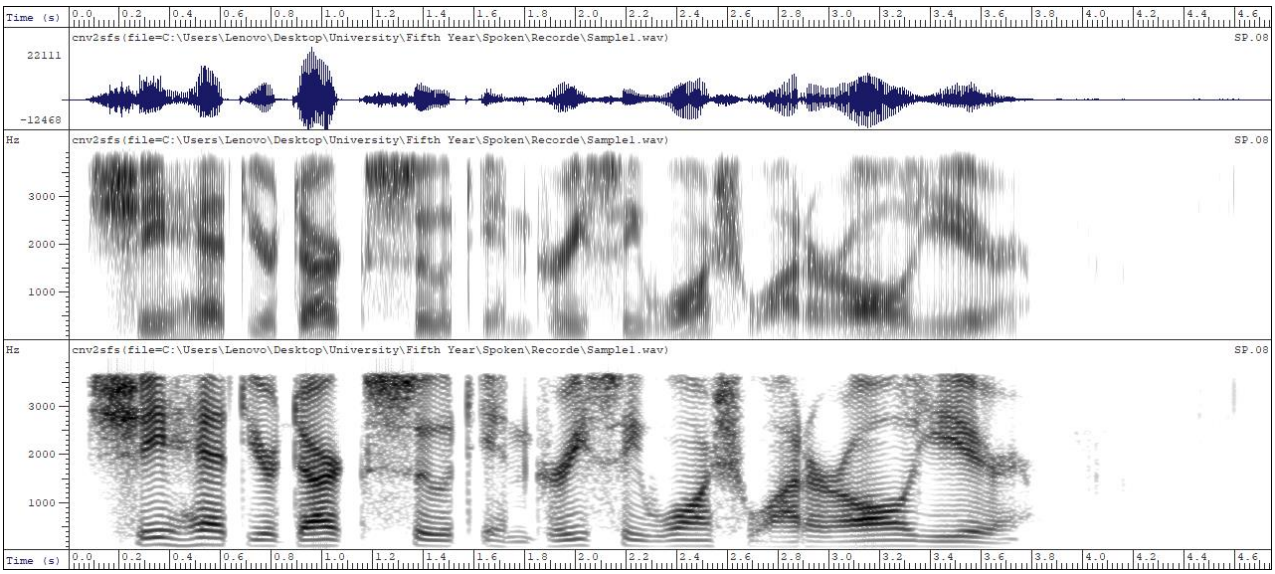


Figure 14:Display Sample1 Spectrogram

Below Figure shows the cross section in purbule color and the source spectrum, filter response, signal spectrum respectivly for the sample1.wav

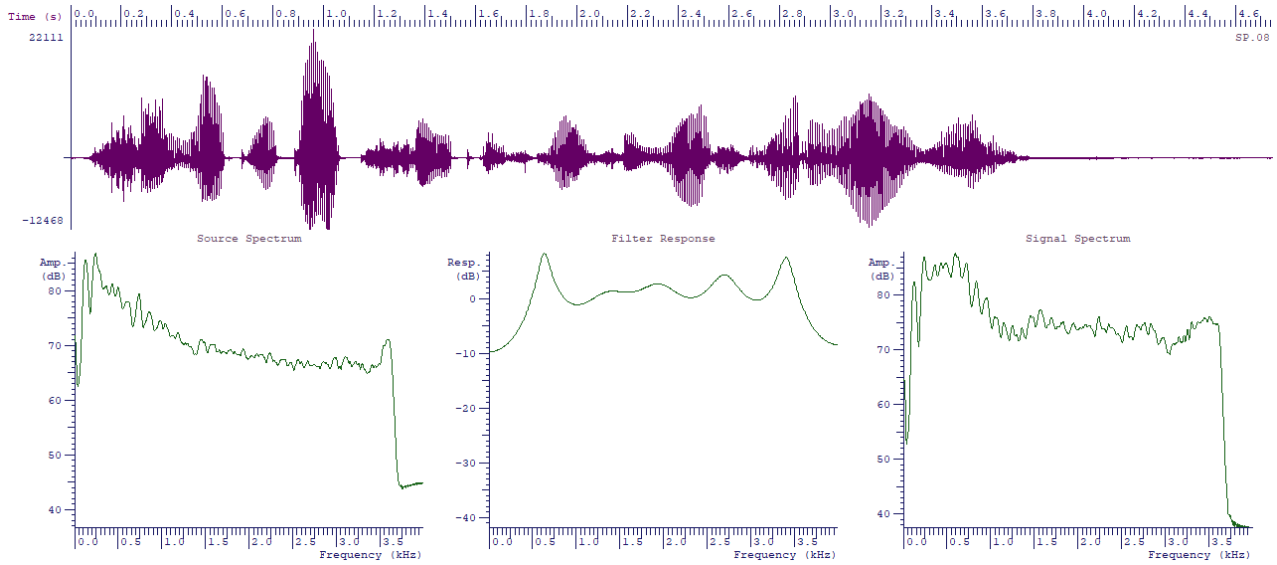


Figure 15:Source Spectrum, Filter Response, and Signal Spectrum of Sample1

Now we display the filter response for each vowel and the fundamental frequency (Fx):

- **e in she**
- **Fx = 148Hz**

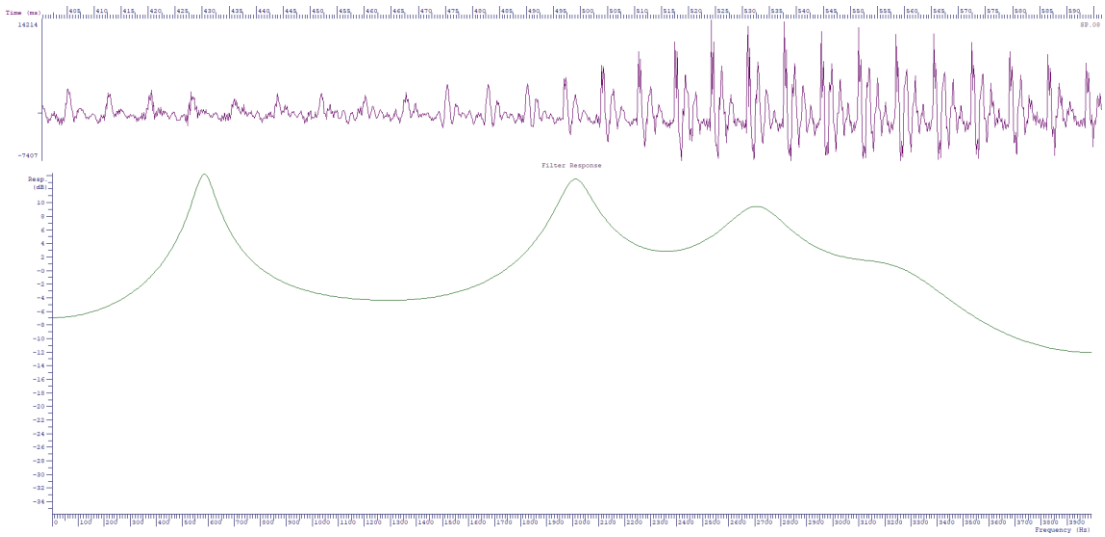


Figure 16:E in She ('sample1.wav')

- **a in had**
- **Fx = 145Hz**

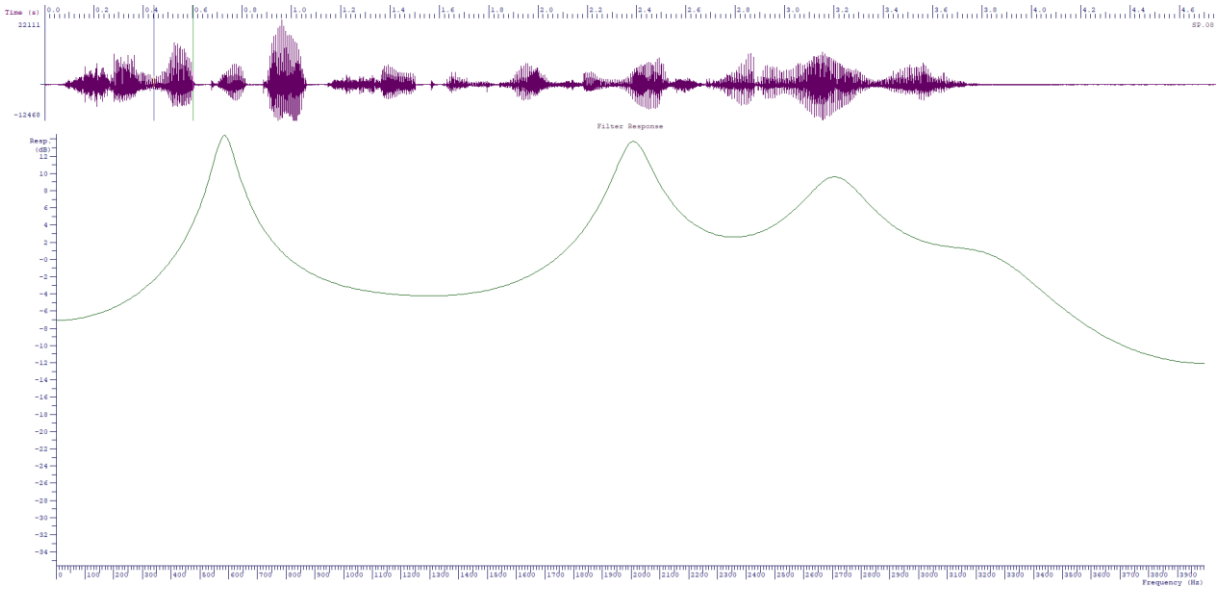


Figure 17:A in had('sample1.wav')

- **u in suite**
- **Fx = 119Hz**

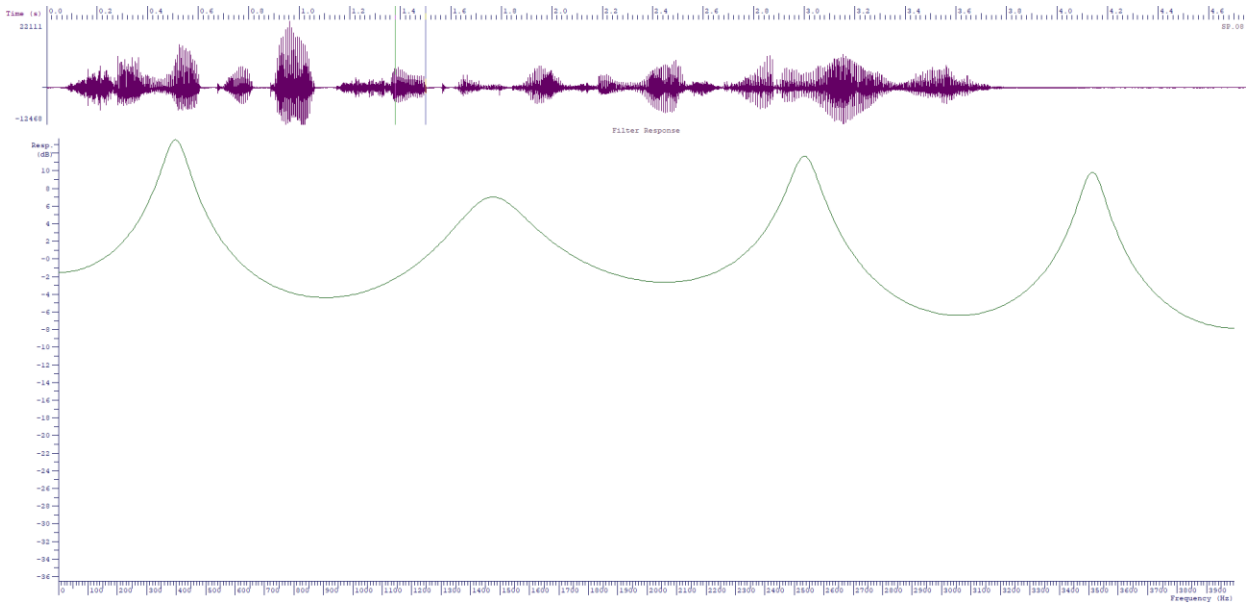


Figure 18: U in suite('sample1.wav')

- **a in Dark**
- **Fx = 127Hz**

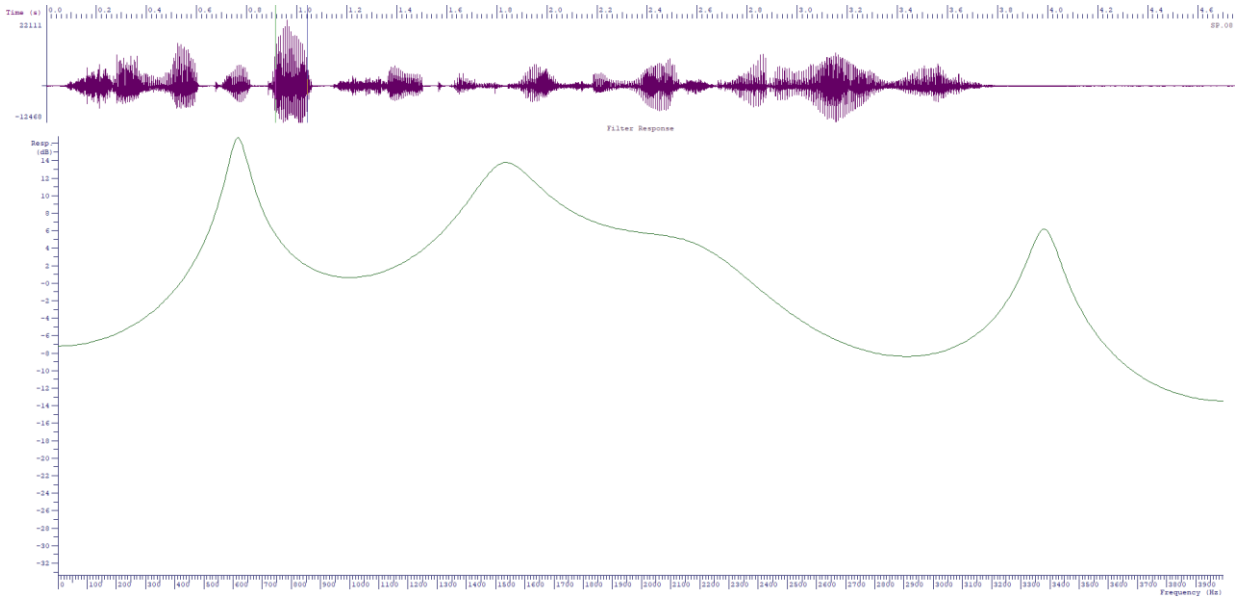


Figure 19: A in Dark('sample1.wav')

This table shows each vowel in specific word and the F1, F2, F3 frequencies respectively. From sample1.

Table 1:Sample1.wav F1, F2, F3 frequencies

Vowel in the Word	F1	F2	F3
e in she	492HZ	2254Hz	2772Hz
a in had	583HZ	2005HZ	2711HZ
u in suit	398HZ	1453	2538
a in dark	615Hz	1526Hz	2203Hz

The below figure shows F2 vs F1 in “sample1.wav”

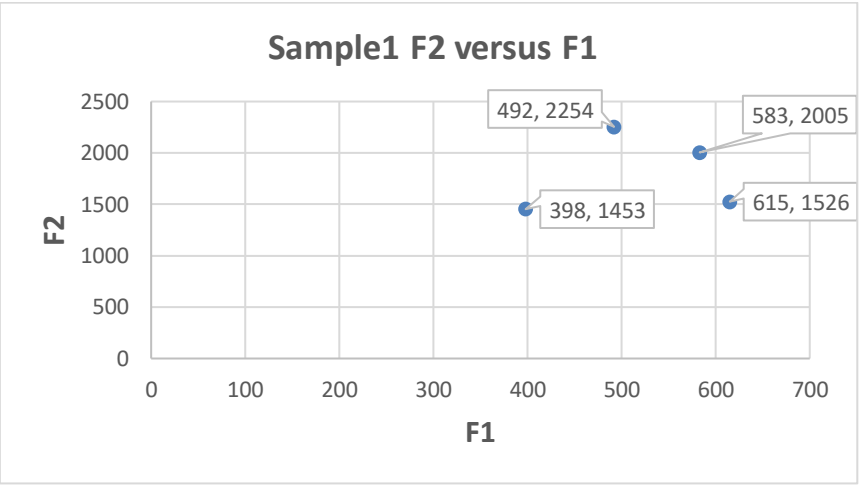


Figure 20: Sample 1 F2 vs F1

Maha Speech Formants

Below Figure shows the time domain, wideband and narrowband spectrogram of the speech file ‘Maha_speech.wav’ which contains saying “She had your dark suit in greasy wash water all year” which have many vowels.

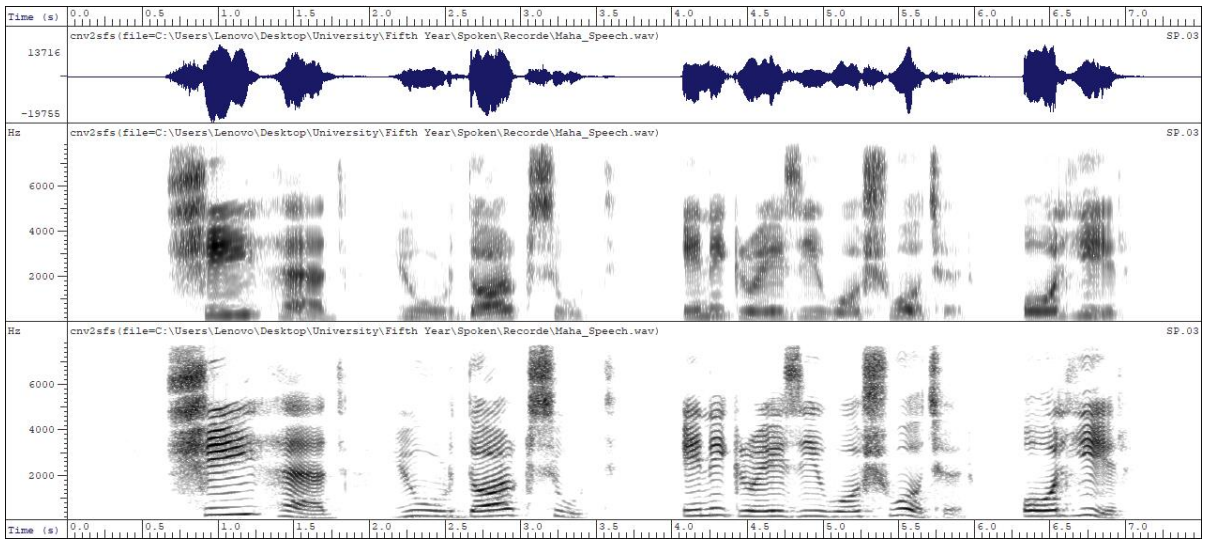


Figure 21:Display Maha Speech Spectrogram

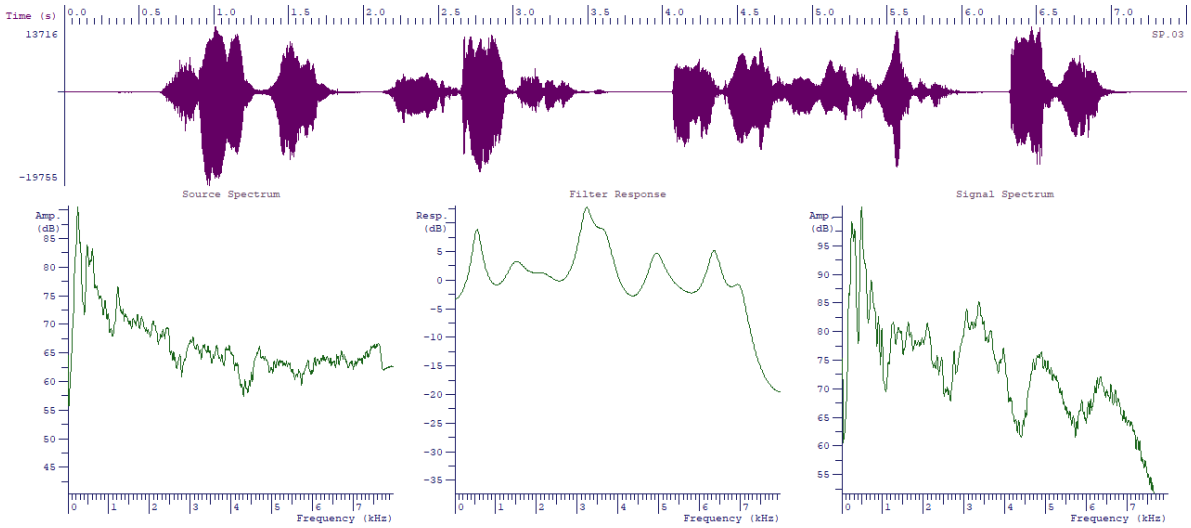


Figure 22: Source Spectrum, Filter Response, and Signal Spectrum of Maha Speech

We start to choose the period which contains the specific vowels in the word as shown in these steps:

- **e in she**

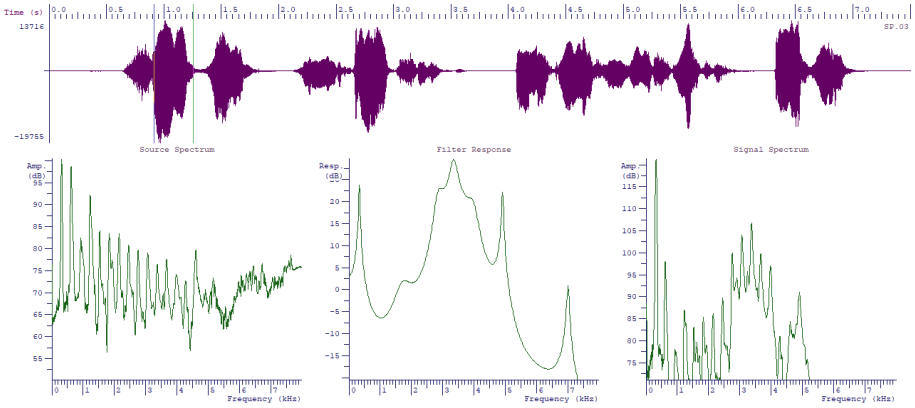


Figure 23:e in she (Maha Speech)

- **a in had**

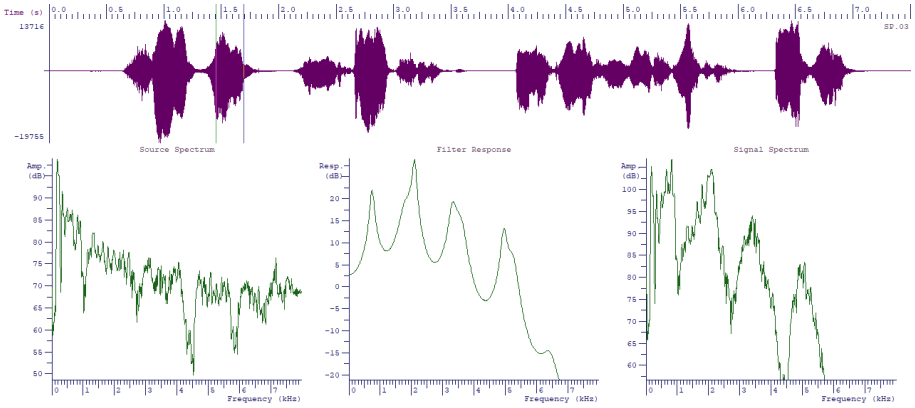


Figure 24:a in had (Maha Speech)

- **a in dark**

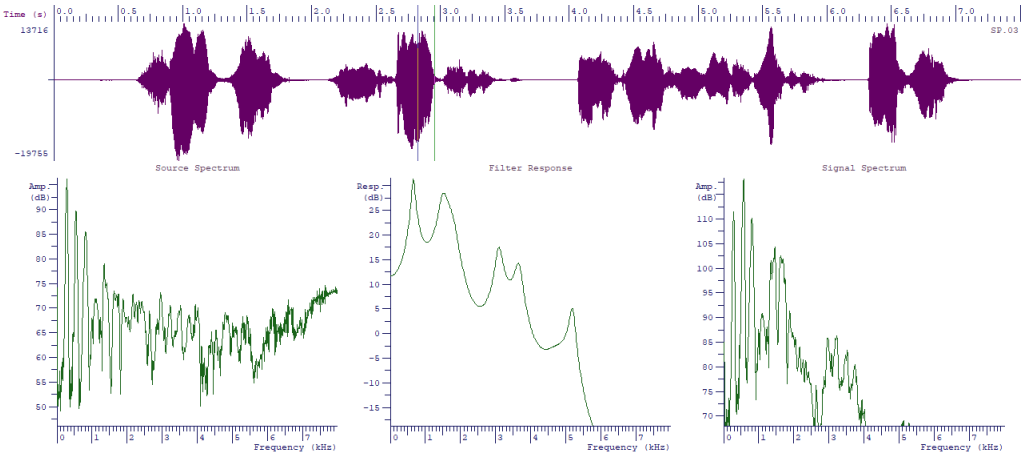


Figure 25:a in dark (Maha Speech)

- **u in suit**

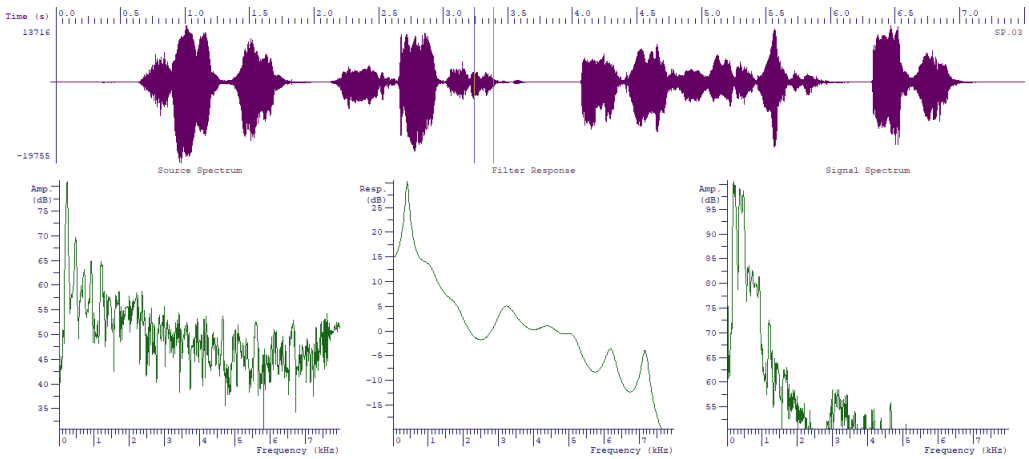


Figure 26:u in suit (Maha Speech)

This table shows each vowel in specific word and the F1, F2, F3 frequencies respectively. From Maha speech.

Table 2: Maha Speech F1, F2, F3 frequencies

Vowel in the Word	F1	F2	F3
e in she	333	2839	3345
a in had	738	2099	3308
u in suit	402	1035	3181
a in dark	625	3704	3108

The below figure shows F2 vs F1 in "Maha_speech.wav"

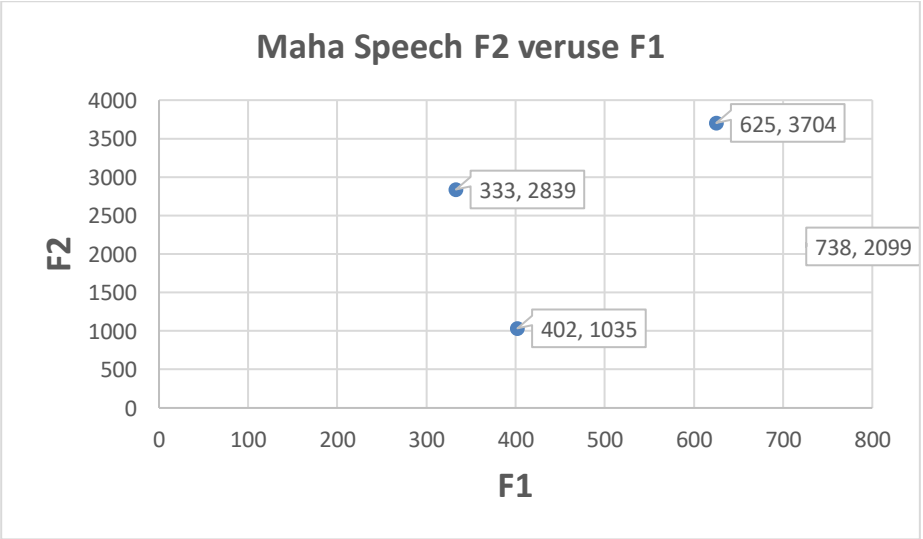


Figure 27: Maha Speech F1 Vs F2

Ebaa Speech Formants

I was used ‘Ibaa speech.wav’ which contains saying “She had your dark suit in greasy wash water all year” which have many vowels to show the time domain, wideband and narrowband spectrogram of the speech file.

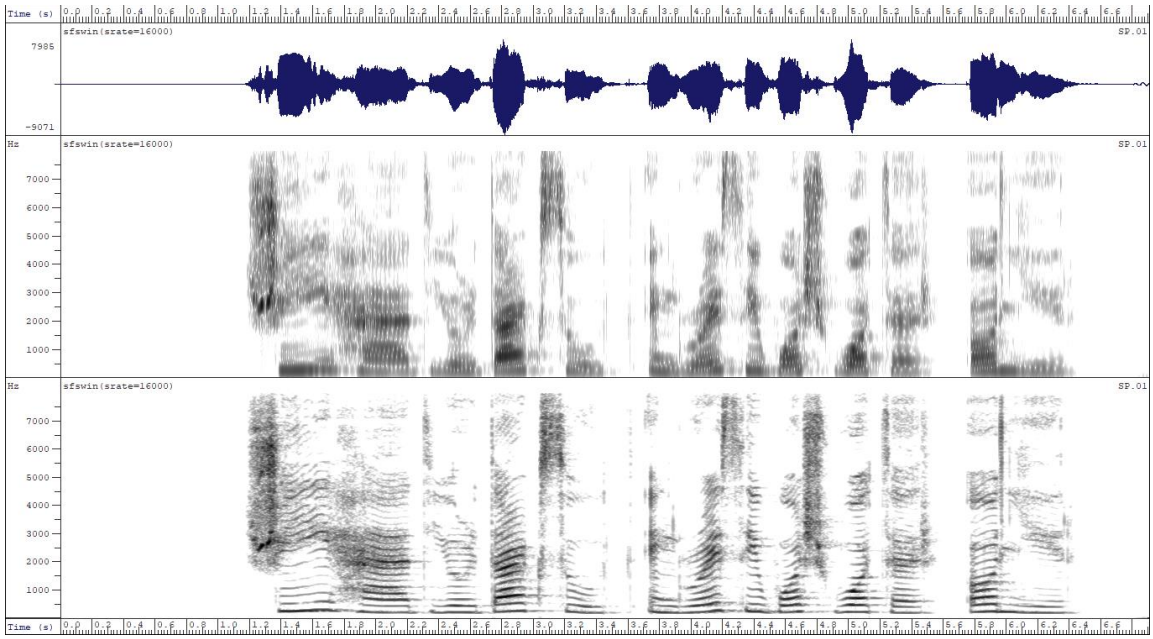


Figure 28: Display Ebaa Speech Spectrogram

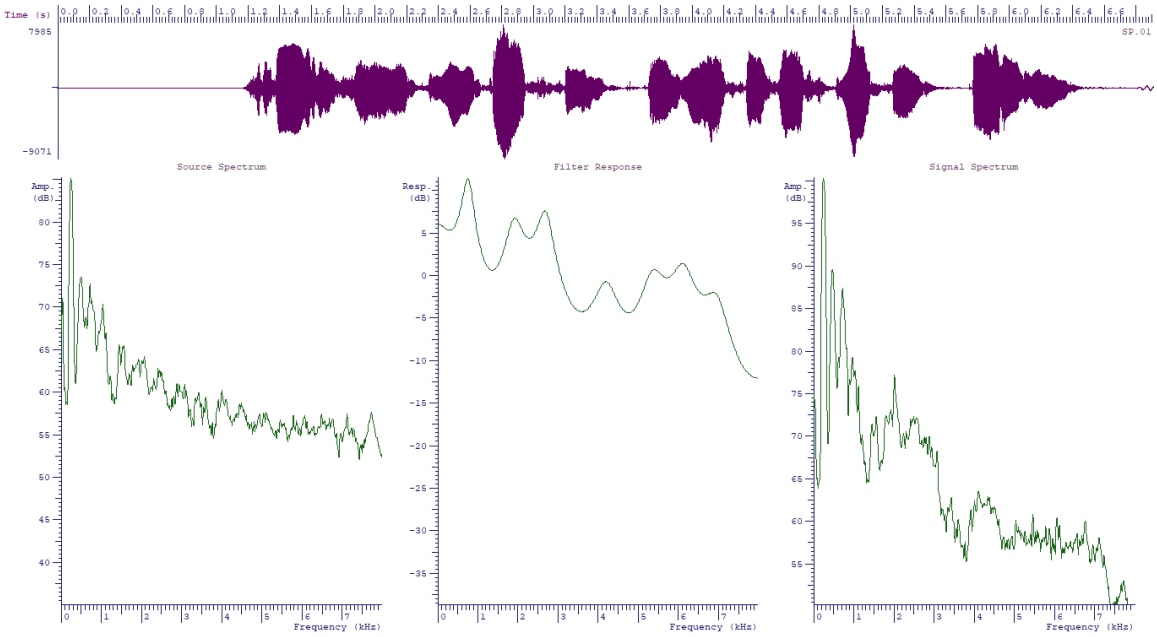


Figure 29: Source Spectrum, Filter Response, and Signal Spectrum of Ebaa Speech

I start to choose the period which contains the specific vowels in the word as shown in these steps:

- **e in she**

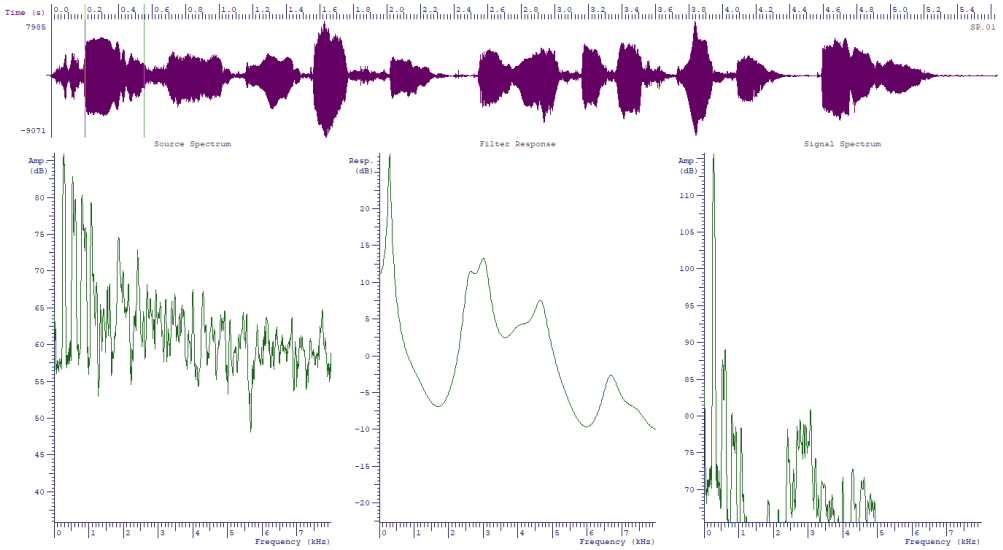


Figure 30:e in she (Ebaa Speech)

- **a in had**

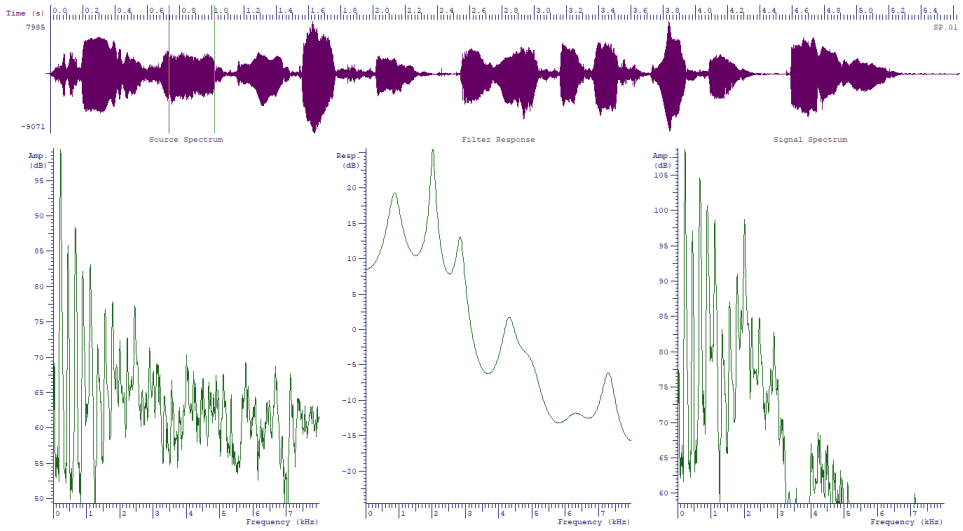


Figure 31::a in had (Ebaa Speech)

- **a in dark**

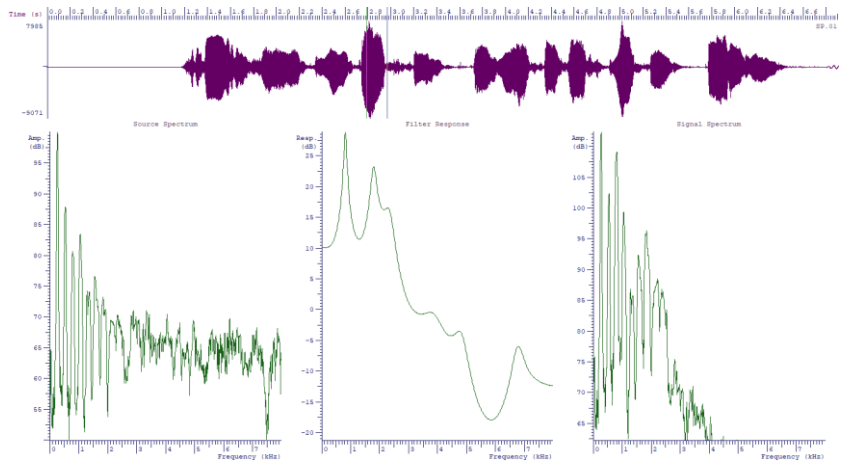


Figure 32::a in dark (Ebaa Speech)

- **u in suit**

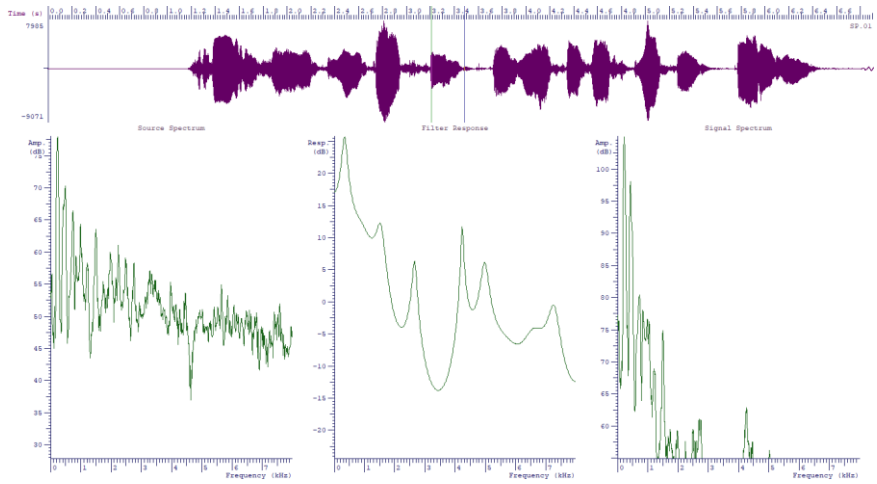


Figure 33::u in suit (Ebaa Speech)

This table shows each vowel in specific word and the F1, F2, F3 frequencies respectively from Ebaa speech.

Table 3:Ebaa Speech F1, F2, F3 frequencies

Vowel in the Word	F1	F2	F3
e in she	292	2579	3029
a in had	869	2023	2856
u in suit	341	2658	4231
a in dark	814	1788	2348

The below figure shows F2 vs F1 in “Ibaa Speech.wav”

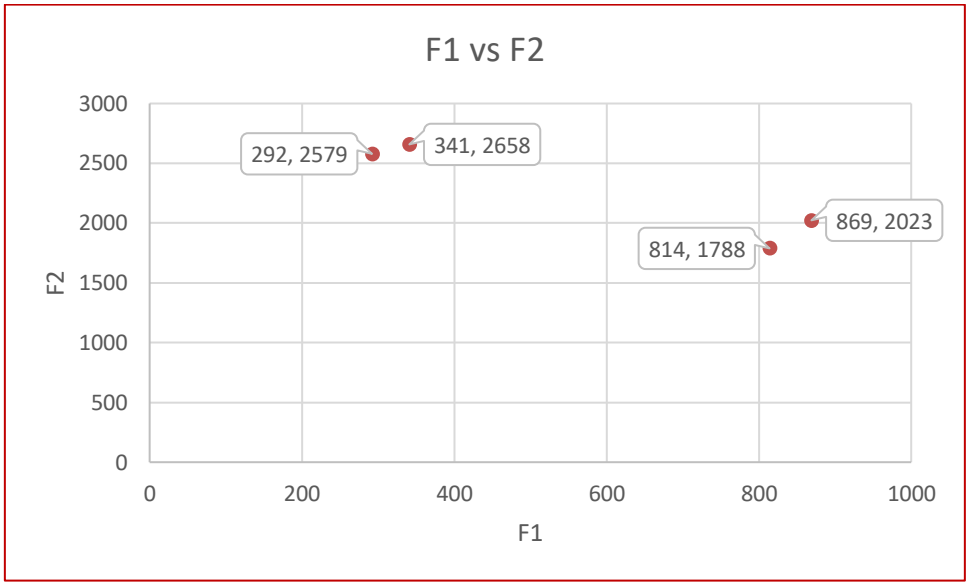


Figure 34:Ebaa Speech F1 VS F2

Native Vs Our Vowel Quadrilateral

➔Native Vowel Quadrilateral

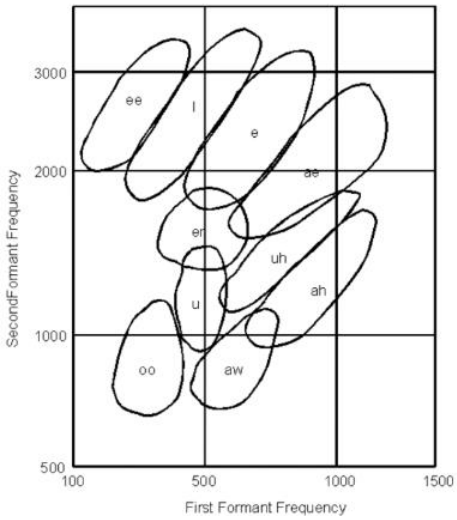


Figure 35: Native Vowel Quadrilateral

➔Sample1 Vowel Quadrilateral

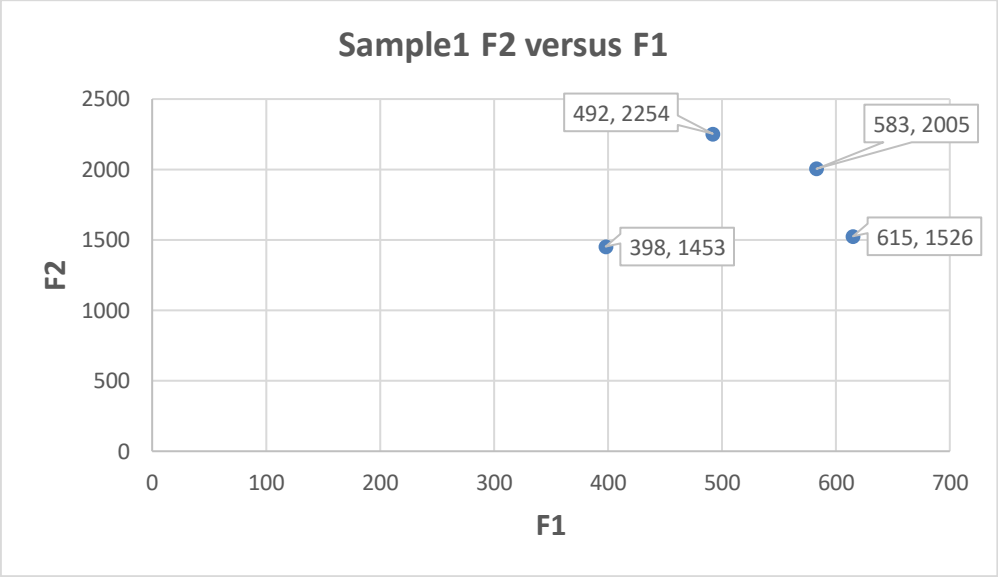


Figure 36:Sample1 Vowel Quadrilateral

➔Maha Vowel Quadrilateral

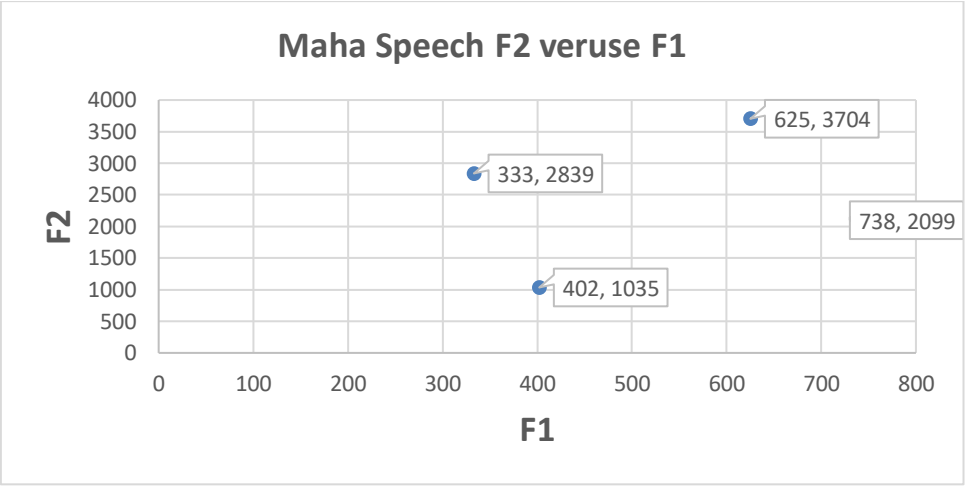


Figure 37: Maha Vowel Quadrilateral

➔Ebaa Vowel Quadrilateral

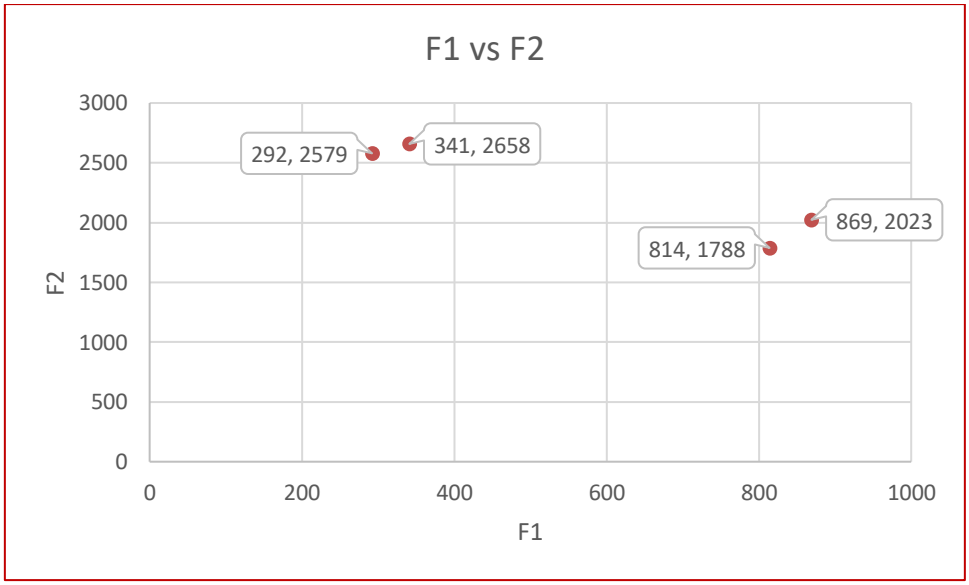


Figure 38: Ebaa Vowel Quadrilateral

Part B: Source and Filter analysis

A in dark

Below figure shows the source spectrum, filter response, signal spectrum respectively of the **/a/** in the word dark, **a** is voiced sound which show higher peaks in low frequencies and lower peaks in higher frequencies, and the filter response is decreasing as the frequency increasing. Which indicates energy concentration toward lower frequencies found in vowels.

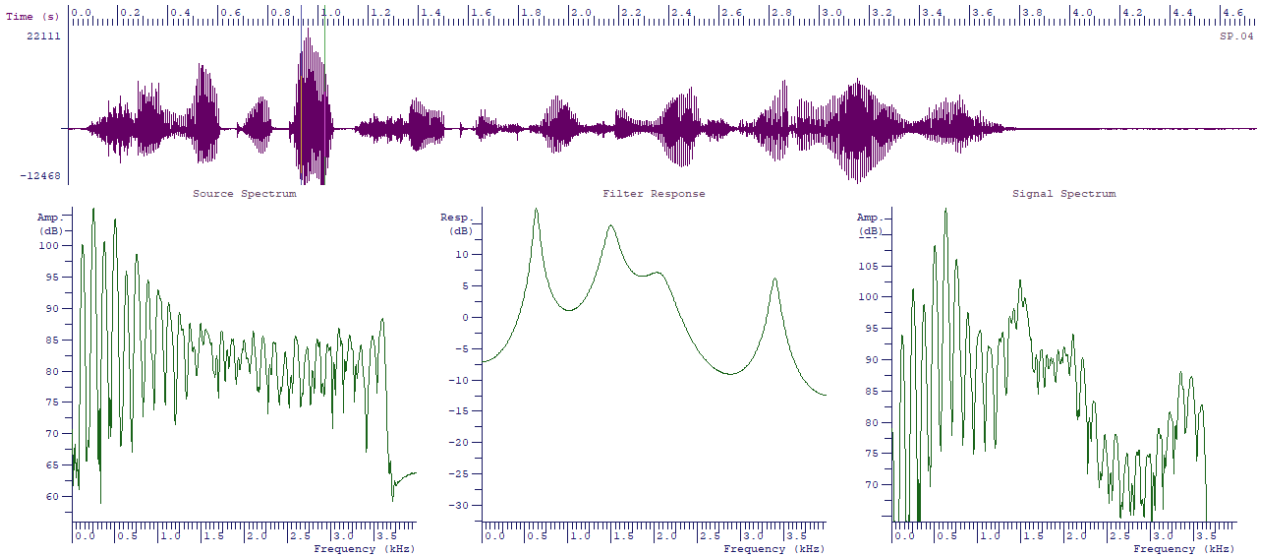


Figure 39:A in Dark source spectrum, filter response and signal spectrum

S in suit

Below figure shows the source spectrum, filter response, signal spectrum respectively of the **/s/** in the word suit, **s** is unvoiced sound, filter response is increasing as the frequency increasing. Which indicates energy concentration toward higher frequencies found in consonants.

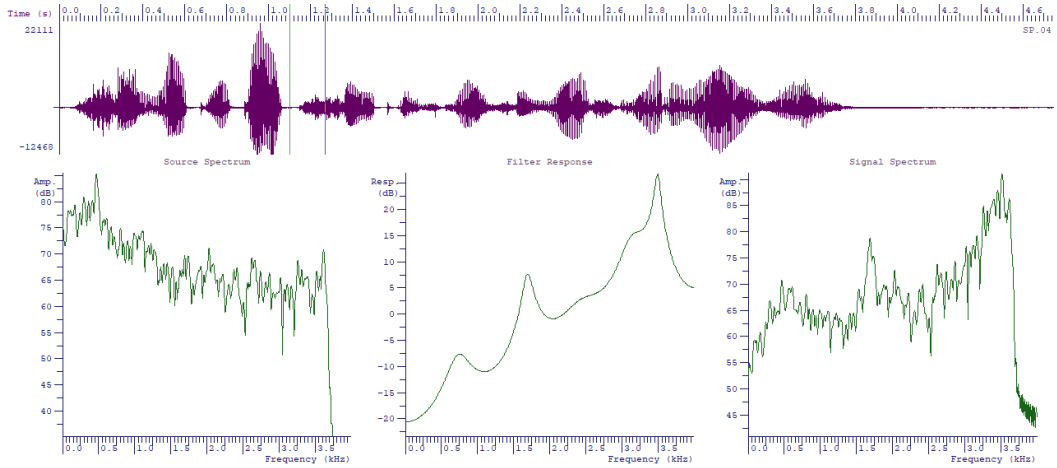


Figure 40:S in Suit source spectrum, filter response and signal spectrum

Part C: Fundamental Frequency

Maha Speech: a in dark

- Chose the period between two vertical red line which is 2745.5 ms to 2765.5 ms.
- Calculate the period:

$$period = End\ Time - Start\ Time$$
$$= 2765.5 - 2745.5 = \mathbf{20\ ms}$$

- Find the number of repeated symbols (Peaks) which is 5 (purple circle)
- Calculate the duration for each peak:

$$Duration = \frac{period}{Number\ of\ repeated\ Symbols} = \frac{20ms}{5} = \mathbf{4ms}$$

- Calculate the fundamental frequency (Fx):

$$Fx = \frac{1}{Duration} = \frac{1}{4ms} = \mathbf{250Hz}$$

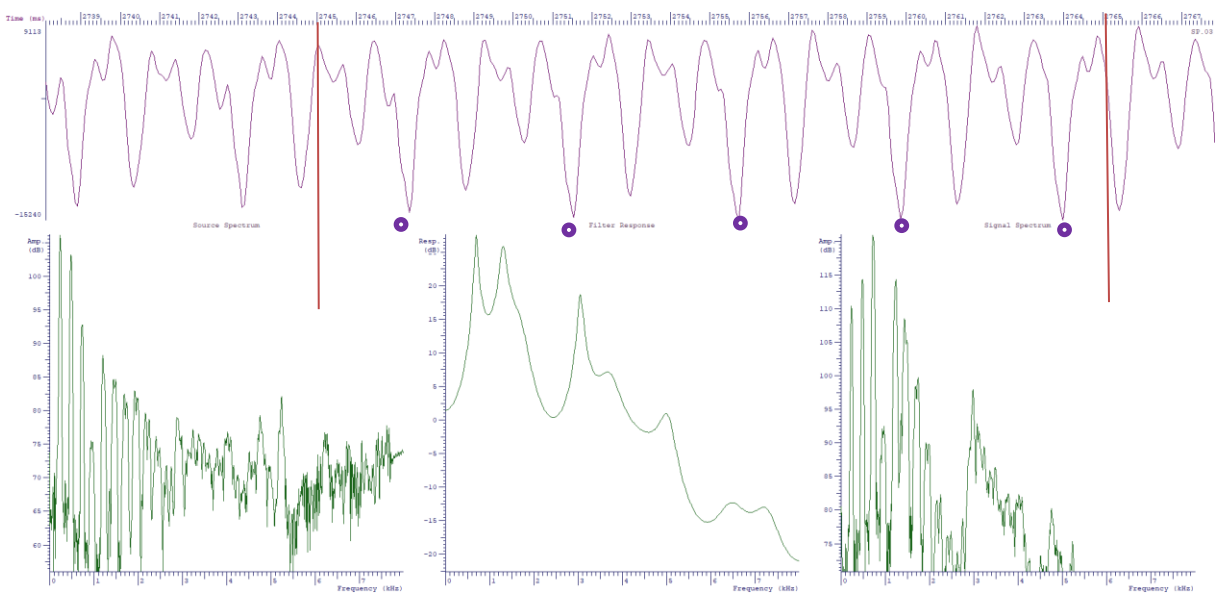


Figure 41:Maha Fundamental Frequency

Ebaa Speech:
Ibaa Speech: e in she

- Chose the period between two verticals blue and green lines which is 235 ms to 265.25 ms.
- Calculate the period:

$$period = End\ Time - Start\ Time$$
$$= 265.25 - 235 = \mathbf{30.25\ ms}$$

- Find the number of repeated symbols (Peaks) which is 8
- Calculate the duration for each peak:

$$Duration = \frac{period}{Number\ of\ repeated\ Symbols} = \frac{30ms}{8} = \mathbf{3.8ms}$$

- Calculate the fundamental frequency (Fx):

$$Fx = \frac{1}{Duration} = \frac{1}{3.8ms} = \mathbf{263Hz}$$

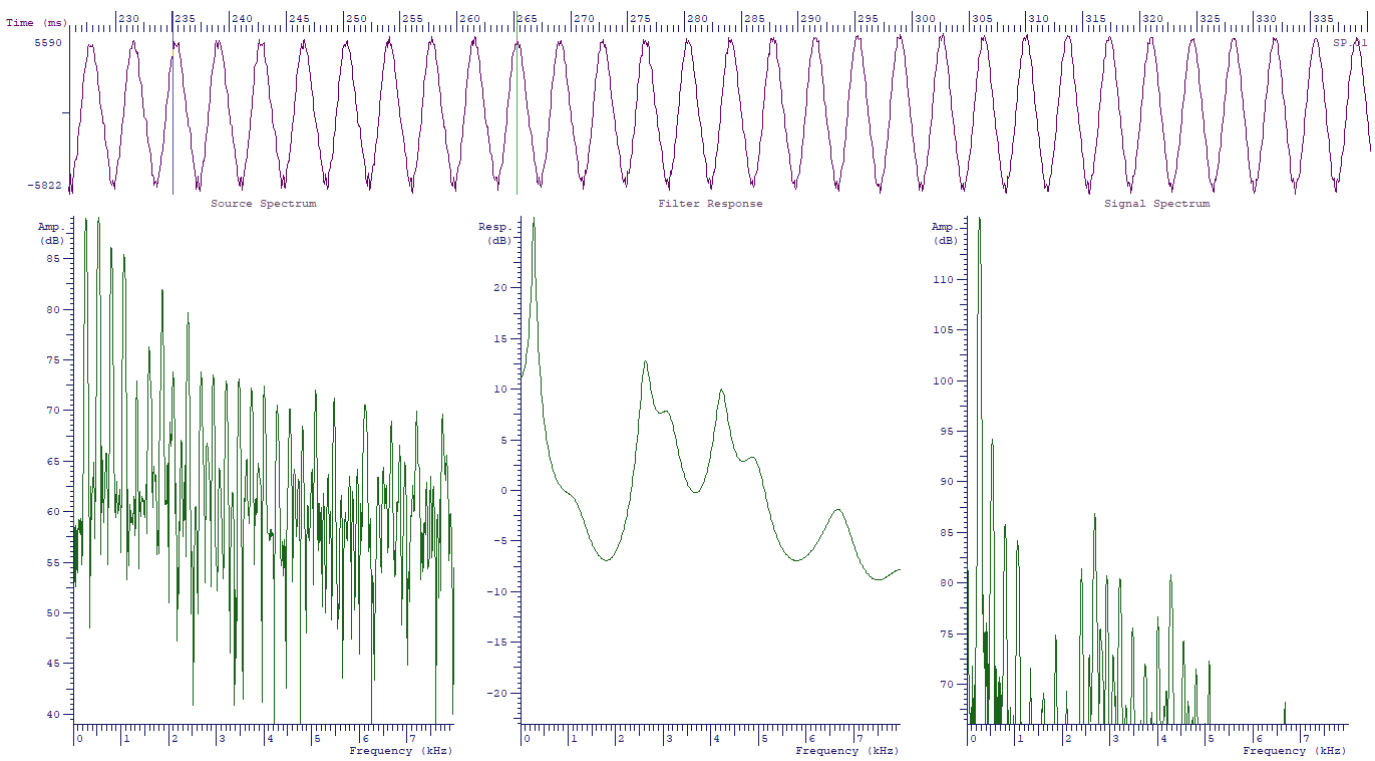
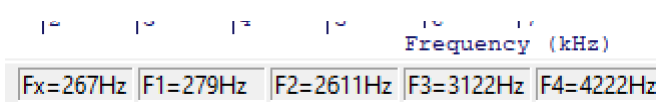


Figure 42: Ebaa Fundamental Frequency

The figure below show that Fx using display cross section very close to the calculated Fx



Part 4: Speech Analysis

Figure shows the waveform for the file AW1.wav. We selected, the period between 3290 – 3800 ms because this part containing the word “Capacity”. Then we copy the segment into the file “capacity. wav” which contains the word “Capacity”, to identify the start and end time for each phoneme.

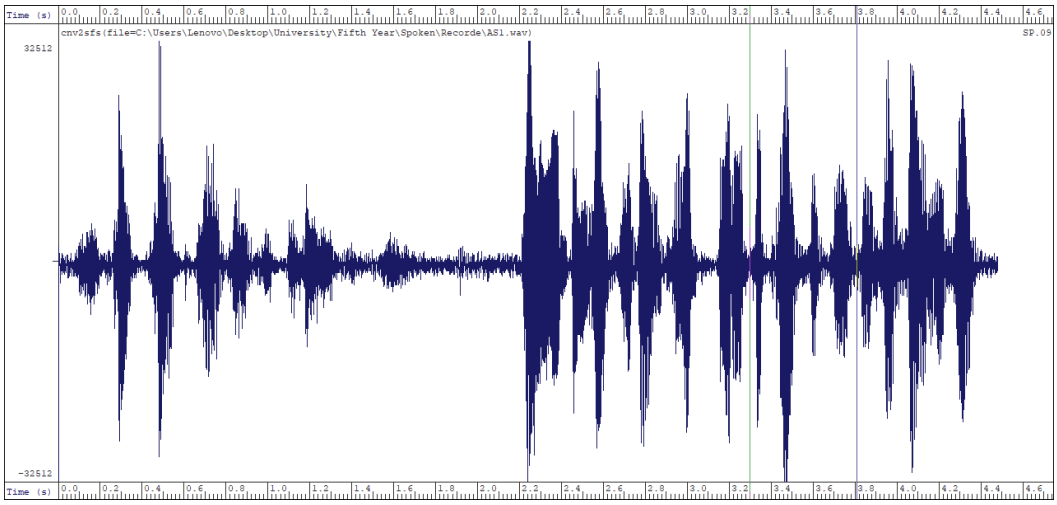


Figure 43: AW1. wav spectrogram

After that we import the file “capacity. wav” as shown in below figure, then we display the waveform, wideband spectrogram, narrowband spectrogram respectively.

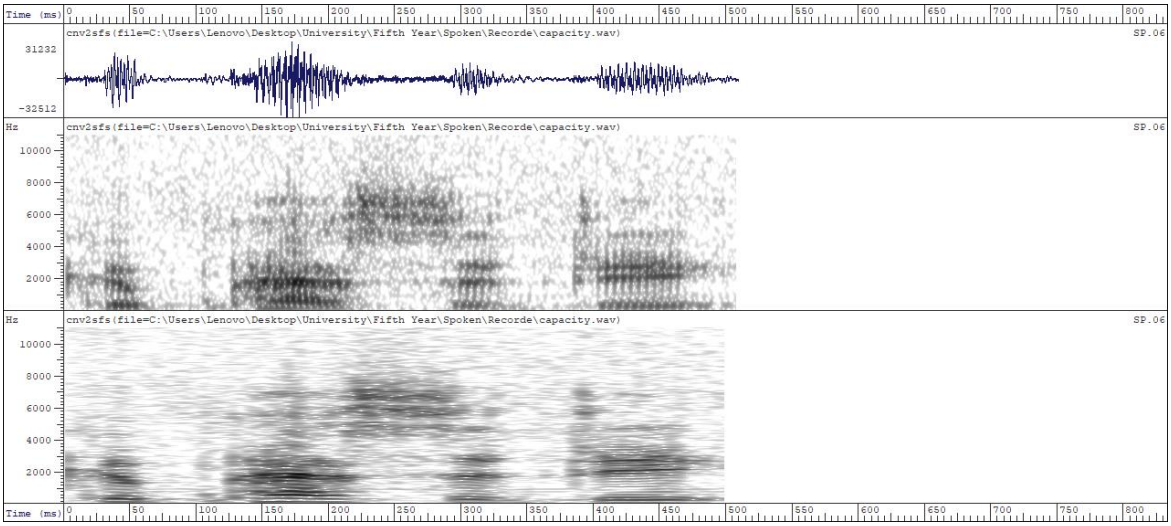


Figure 44: 'capacity' spectrogram

Then we start to identify the start time and end time for each phoneme. Below table shows the start time and end time of each phoneme in the word **capacity.**

Table 4 : start time and end time for each phoneme.

Phonemes	Start time (ms)	End Time (ms)
/k/	0	74.4
/@/	74.4	127.5
/p/	127.5	206
/{/	206	248.3
/s/	248.1	340.5
/I/	340.5	397.4
/t/	397.4	459.5
/i/	459.5	515.4

Part 5: Sub-Word Level Connection

In this part, we recorded the words "life" and "night" using the carrier phrase "Say ... again," which ensuring consistent breathing and emphasis during each recording. We first recorded the utterance "Say life again" and then isolated the segment corresponding to the word "life" and exported it as li.wav file. Next, I repeated the process for the word "night", extracting the segment corresponding to "night" and saving it as a ight.wav file. With these two isolated segments ("li" from "life" and "ight" from "night"), I used MATLAB to combine them into a new synthesized word, "light", by concatenating the "li" part from "life" and the "ight" part from "night." In MATLAB, I wrote a script that read the two wav files, check the sampling rate of both audio then concatenated them, and saved the resulting waveform as a “light.wav” file. This process is known as concatenative synthesis, where parts of different speech signals are joined together to create a new word.

Code 4: Part 5 Sub Word Level Connection

```
% File names of the input .wav files
file1 = 'li.wav';
file2 = 'ight.wav';
% Read audio data from the input .wav files
[y1, Fs1] = audioread(file1);
[y2, Fs2] = audioread(file2);
% Check if the sampling rates of both audio files match
if Fs1 ~= Fs2
    error('Sampling rates of input files do not match.');
end
% Concatenate the audio data
y_concatenated = [y1; y2];

% Define the file name for the output .wav file
output_file = 'light.wav';

% Write the concatenated audio data to the output .wav file
audiowrite(output_file, y_concatenated, Fs1);

% Play the concatenated audio
sound(y_concatenated, Fs1);
```

Part 6: Phone-Level Concatenation

In this part we choose these phoneme: **/k/, /@/, /t/**, to generate the sound **"cat"** after concatenate these three phonemes. From part 4 we saved the sound of each phoneme after identify the start and end time into the file, such as the sound for phoneme /k/ saved in file "k_capacity.wav", and the same for the rest of the phonemes.

In addition to that, we display the waveform for each phoneme which is: **/k/, /@/, /t/**, through import the wave file for the phonemes: **/k/, /@/, /t/**

➔ **/k/ Phoneme Waveform:**

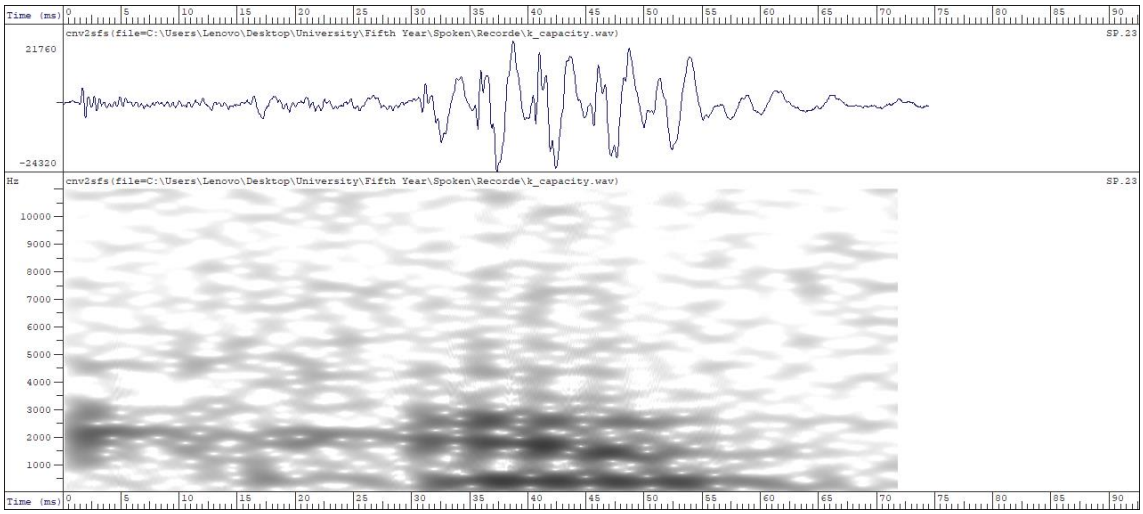


Figure 45: /k/ Phoneme Waveform

➔ **/@/ Phoneme Waveform:**

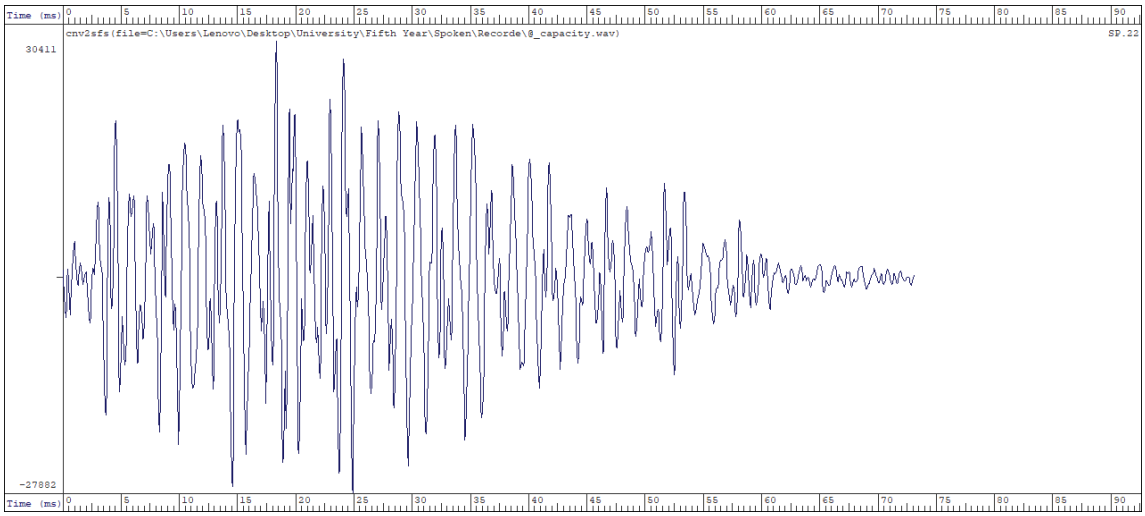


Figure 46: /@/ Phoneme Waveform

➔ **/t/ Phoneme Waveform:**

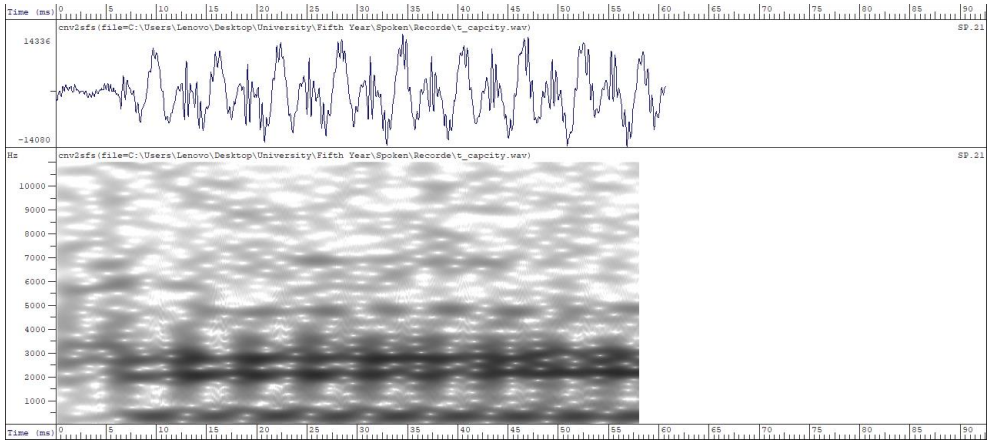


Figure 47:/t/ Phoneme Waveform

Below code shows how we connected three phonemes together in MATLAB.

Code 5: Part6 to Connected three Phonemes

```
% Read the contents of each .wav file
[c_audio, fs] = audioread('k_capacity.wav');
[a_audio, ~] = audioread('@_capacity.wav');
[t_audio, ~] = audioread('t_capcity.wav');

% Adjust the length of each audio segment to match the shortest segment
min_length = min([length(c_audio), length(a_audio), length(t_audio)]);
c_audio = c_audio(1:min_length);
a_audio = a_audio(1:min_length);
t_audio = t_audio(1:min_length);

% Concatenate the audio data
cat_audio = [c_audio; a_audio; t_audio];

% Write the concatenated audio data to a new .wav file
output_File = 'cat.wav';
audiowrite(output_File, cat_audio, fs);

% Play the synthesized word
sound(cat_audio, fs);

disp('Concatenation complete');
disp(['Concatenated audio written to ' output_File]);
```

This figure displays the waveform and wideband spectrogram to the file “cat.wav”, this file generated after concatenation of three phonemes.

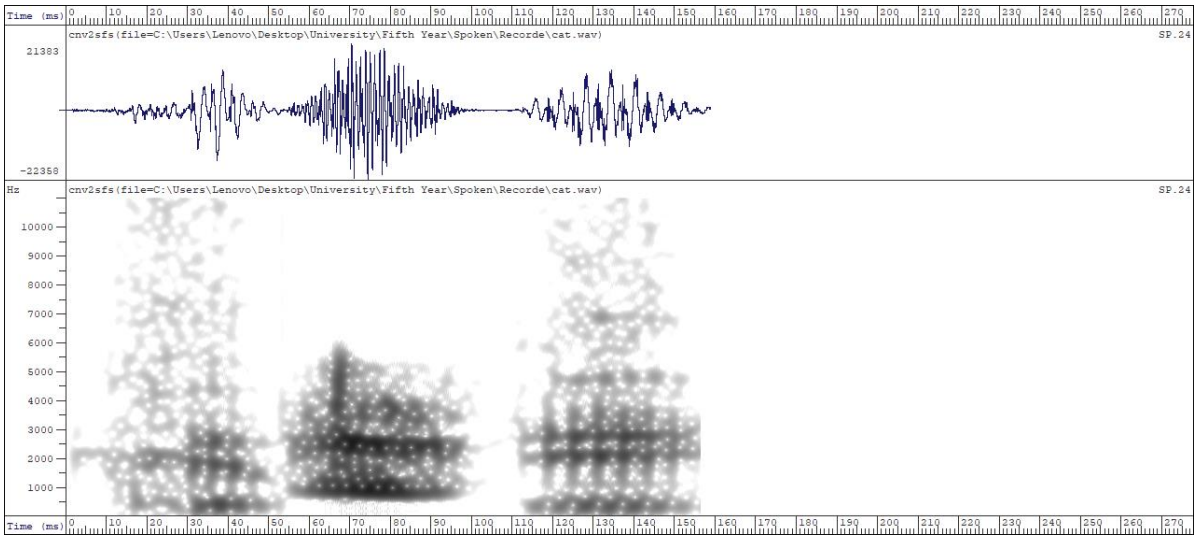


Figure 48: waveform and wideband spectrogram to the file “cat.wav"

Quick changes in sound quality or volume might impact speech quality. Zero-crossing locations may not always be optimal for joining sounds. This can result in unwanted sounds or distortions. To improve the quality of synthesized speech by concatenating audio parts, many strategies might be considered:

- Doing listening tests and getting feedback from listeners helps identify areas for enhancement and guide repeated developing of the synthesis process for better speech quality.
- We can combine many recordings of the same letter and average them. Then we overlap or link these recordings, making the sound appear smoother and more connected.
- We can improve audio processing by aligning waveforms, matching tone and quality, and utilizing machine learning to identify optimal locations to combine sounds based on natural speech scenarios.
- We can change the crossfade length and use a database of small sound units to achieve smoother and more accurate speech transitions.

Part 7: Generation of Vowel Sounds Using the Source-Filter Model

7.1 Generating Pulse Train

At first, we generated a pulse train signal with this specification:

- Signal Type: pulse train
- Frequency=110Hz
- Sampling Frequency (Fs)= 8000Hz

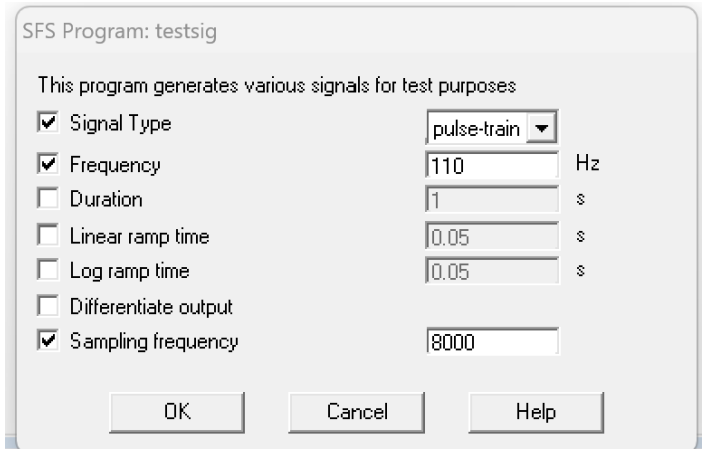


Figure 49:generation of pulse train signal specification

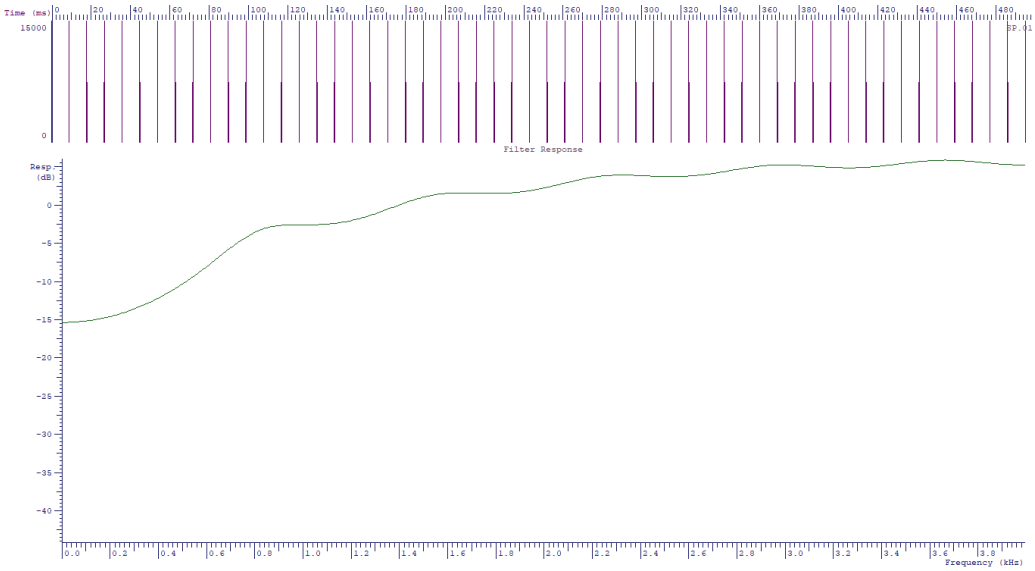


Figure 50:pulse train signal spectrogram

7.2 Generating Bandpass Filters

We defined formant frequencies using band-pass filter

- ⇨ For /a:/ F1=700Hz

Lower band-pass frequency: 700 Hz - 200 Hz = 500 Hz.
Higher band-pass frequency: 700 Hz + 200 Hz = 900 Hz.

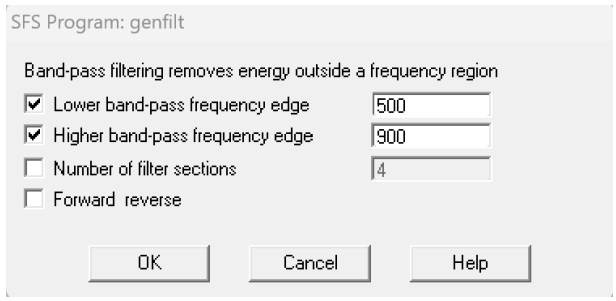


Figure 51: /a:/ F1= 700Hz generations

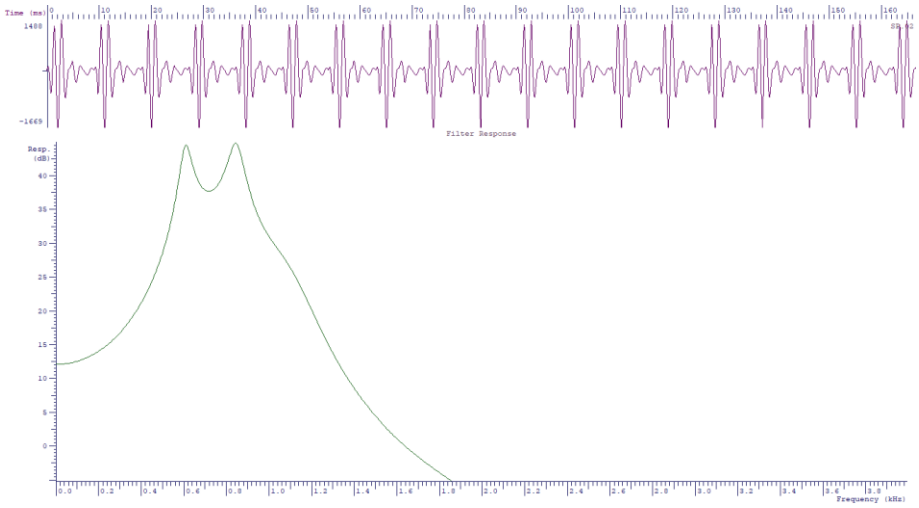


Figure 52:/a:/ F1= 700Hz band pass filter

- ⇨ For /a:/ F2=1100Hz:

Lower band-pass frequency: 1100 Hz - 200 Hz = 900 Hz.
Higher band-pass frequency: 1100 Hz + 200 Hz = 1300 Hz.

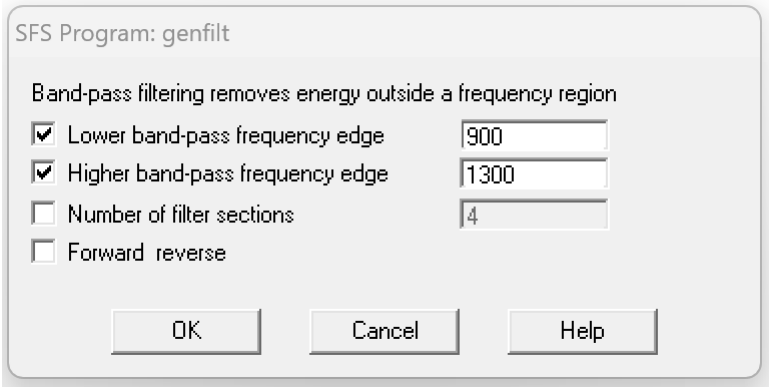


Figure 53: /a:/ F2= 1100Hz generations

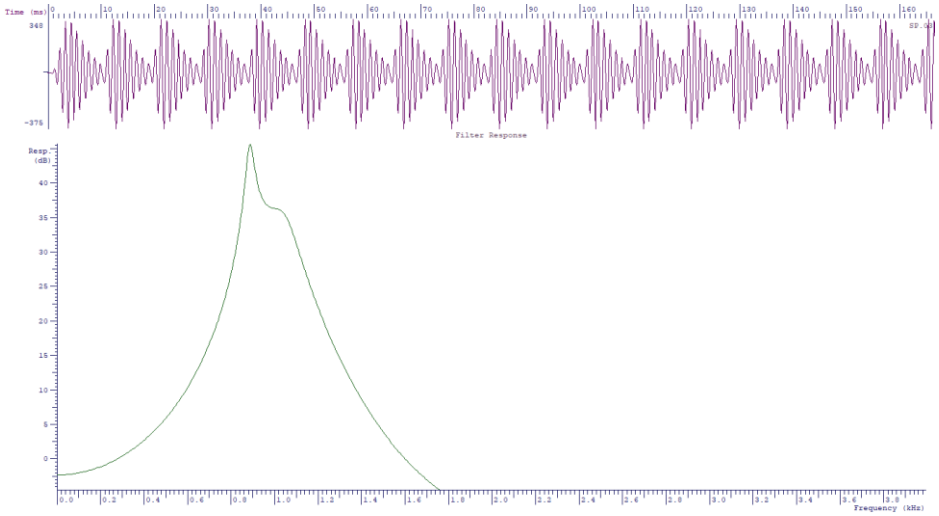


Figure 54:/a:/ F2=1100Hz band pass filter

- **/u:/ F_1=300Hz**

Lower band-pass frequency: 300 Hz - 200 Hz = 100 Hz.
Higher band-pass frequency: 300 Hz + 200 Hz = 500 Hz.

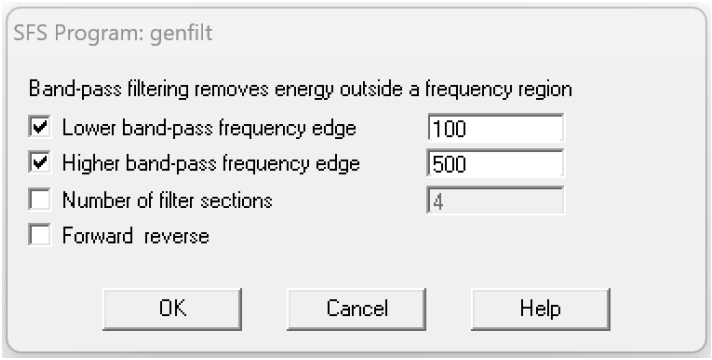


Figure 55: /u:/ F1=300Hz generations

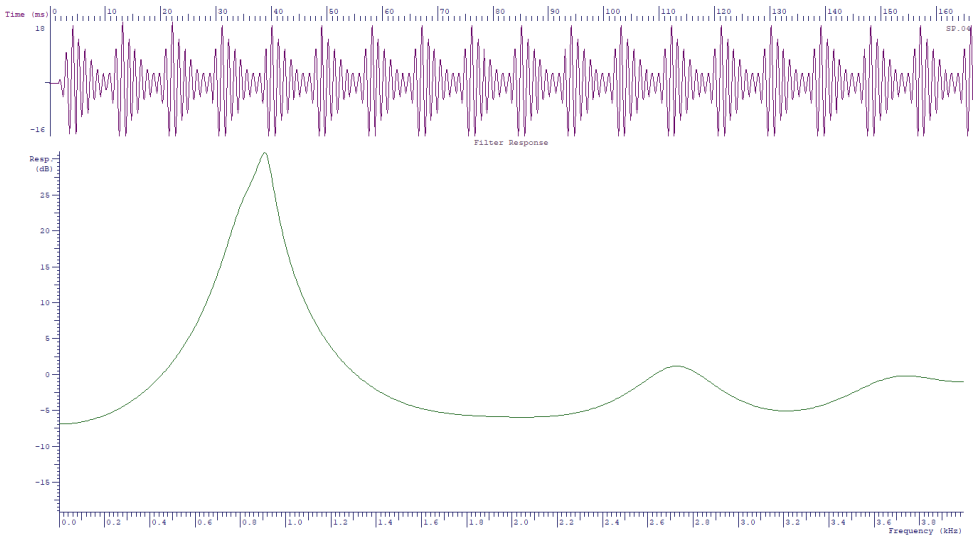


Figure 56: /u:/ F1=300Hz band pass filter

⇨ **/u:/** F2=800Hz:

Lower band-pass frequency edge: 800 Hz - 200 Hz = 600 Hz.
Higher band-pass frequency edge: 800 Hz + 200 Hz = 1000 Hz.

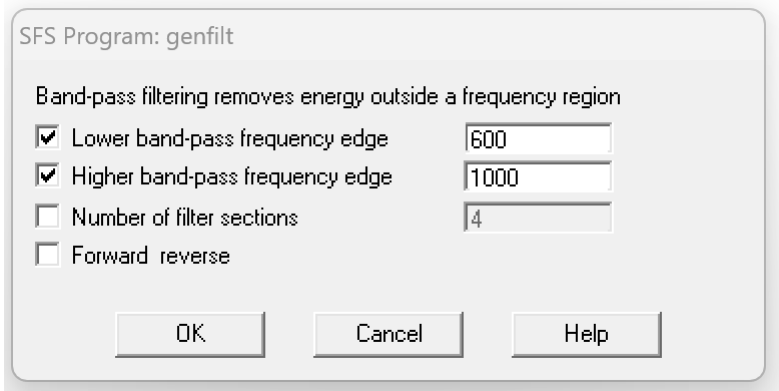


*Figure 57: /**u:**/ F2=800Hz generation*

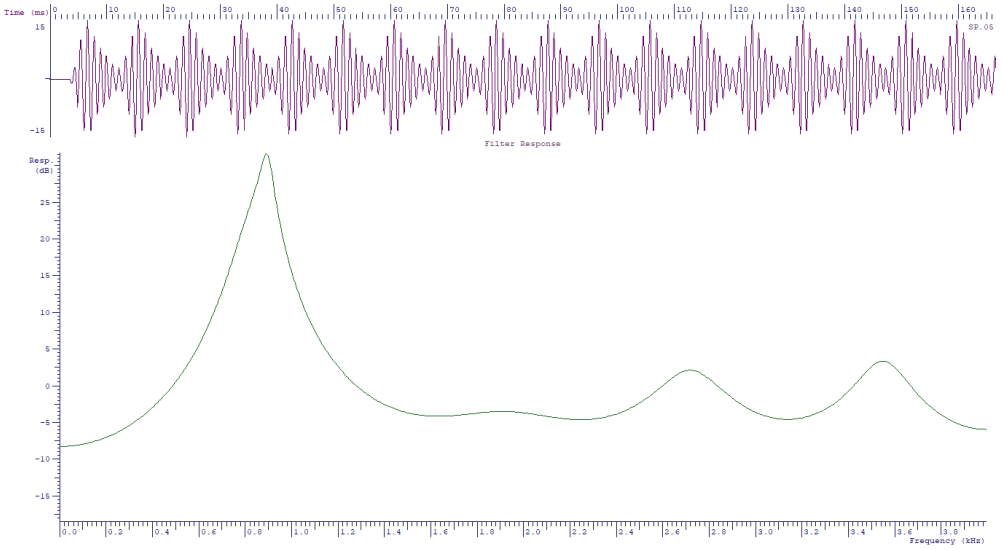


Figure 58:/u:/ F2=800Hz band pass filter

⇨ **/ə/ F_1=500Hz**

Lower band-pass frequency: 500 Hz - 200 Hz = 300 Hz.
Higher band-pass frequency: 500 Hz + 200 Hz = 700 Hz.

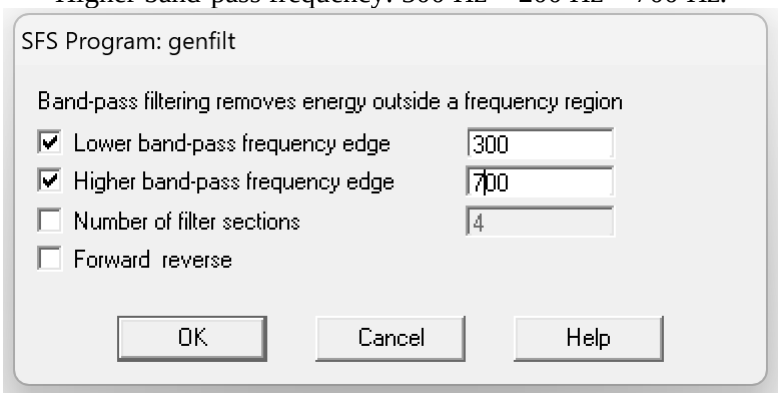


Figure 59: /ə/ F1=500Hz generation

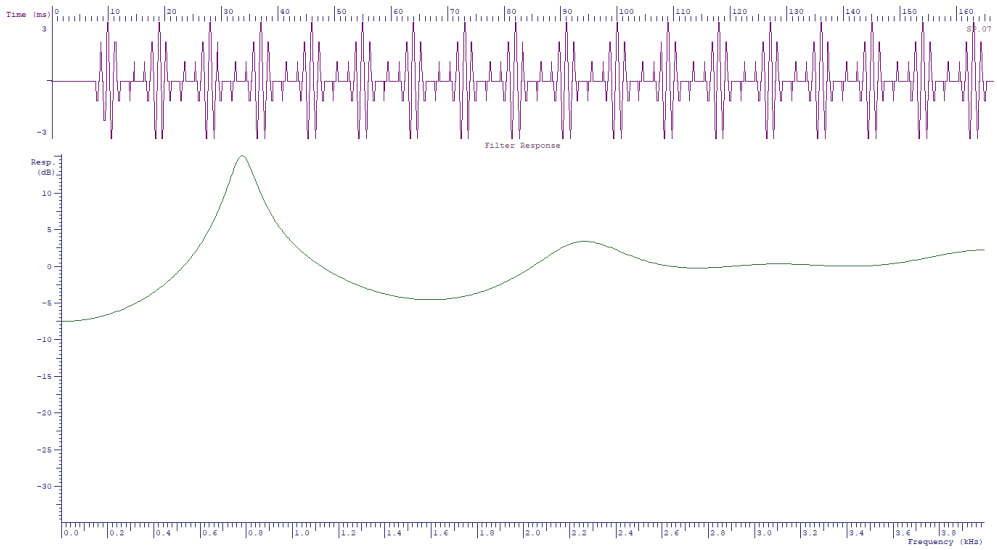


Figure 60:/ə/ F1=500Hz band pass filter

- **/ə/ F_2=1500Hz**

Lower band-pass frequency: 1500 Hz - 200 Hz = 1300 Hz.
Higher band-pass frequency: 1500 Hz + 200 Hz = 1700 Hz.

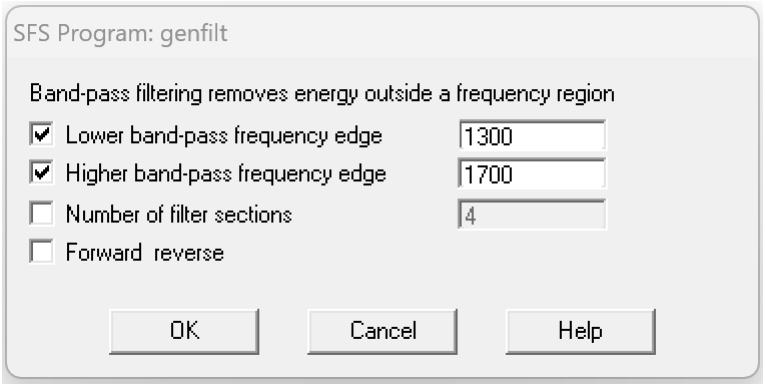


Figure 61: /ə/ F2=1500Hz generation

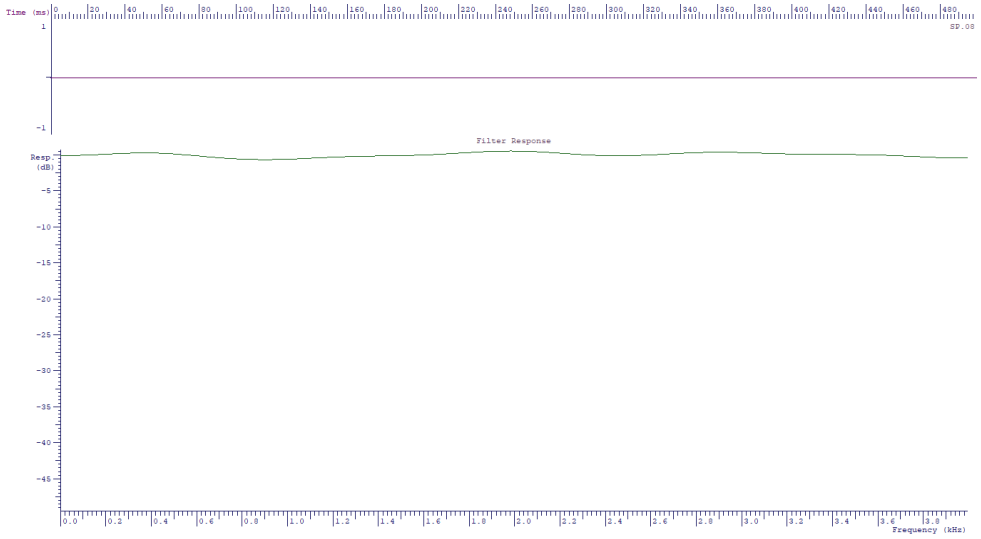


Figure 62: /ə/ F2=1500Hz band pass filter

To synthesize vowel sounds, we applied band-pass filters to a pulse train source signal, at first, read the necessary filter files and pulse train .wav, then processes the source signal for each vowel's formants, and generate audio files for each vowel. The provided code below to Generate the vowel sounds using MATLAB.

Code 6 :Part 7 Code

```
% Read the source signal from impulse.wav
[source_signal, fs] = audioread('pulseTrain.wav');

% Define formant frequencies for each vowel
formants = struct('A', [700, 1100], 'u', [300, 800], 'schwa', [500, 1500]);

% Load band-pass filters for each formant dynamically
filters = struct();
vowels = fieldnames(formants);

for i = 1:length(vowels)
    for j = 1:2 % For F1 and F2
        formant_freq = formants.(vowels{i})(j);
        filter_file = sprintf('%d.wav', formant_freq);

        if ~isfile(filter_file)
            error('Filter file %s does not exist in the current directory.', filter_file);
        end

        % Dynamically load the filter
        filters.(vowels{i}).(sprintf('F%d', j)) = audioread(filter_file);
    end
end

% Initialize final speech signal
final_signal = zeros(size(source_signal));
final_signals = cell(length(vowels), 1);

% Filter source-signal using band-pass filters for each vowel and sum signals
for i = 1:length(vowels)
    vowel_name = vowels{i};

    % Dynamically access filters for F1 and F2
    F1_filter = filters.(vowel_name).F1;
    F2_filter = filters.(vowel_name).F2;

    % Filter source signal with band-pass filters
    filtered_signal_F1 = filter(F1_filter, 1, source_signal);
    filtered_signal_F2 = filter(F2_filter, 1, source_signal);

    % Sum the filtered signals
    vowel_signal = filtered_signal_F1 + filtered_signal_F2;

    % Normalize the final speech signal to ensure maximum amplitude of 1
    max_value = max(abs(vowel_signal));
    if max_value > 1
        vowel_signal = vowel_signal / max_value; % Scale down if necessary
    end

    % Store the final speech signal
    final_signals{i} = vowel_signal;

    % Write the final speech signal to a WAV file
    filename_out = sprintf('%s_vowel.wav', vowel_name);
    audiowrite(filename_out, vowel_signal, fs);

end
```

7.3 Exporting Filtered Signals

A_vowel.wav

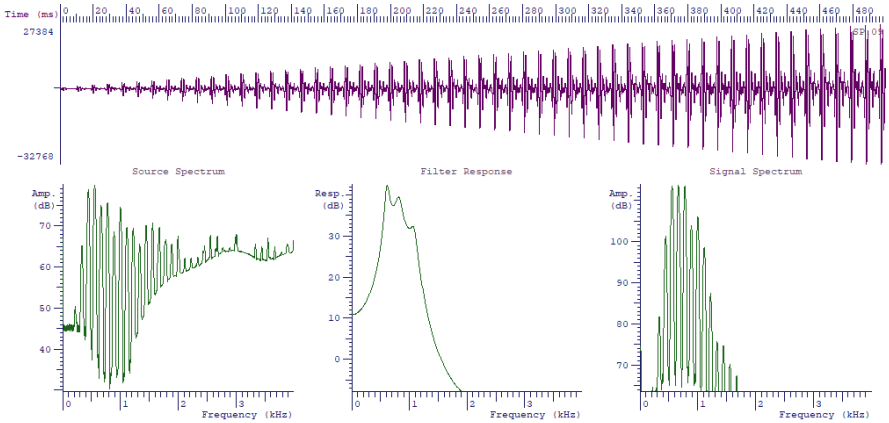


Figure 63: /a:/ Sound

schwa_vowel.wav

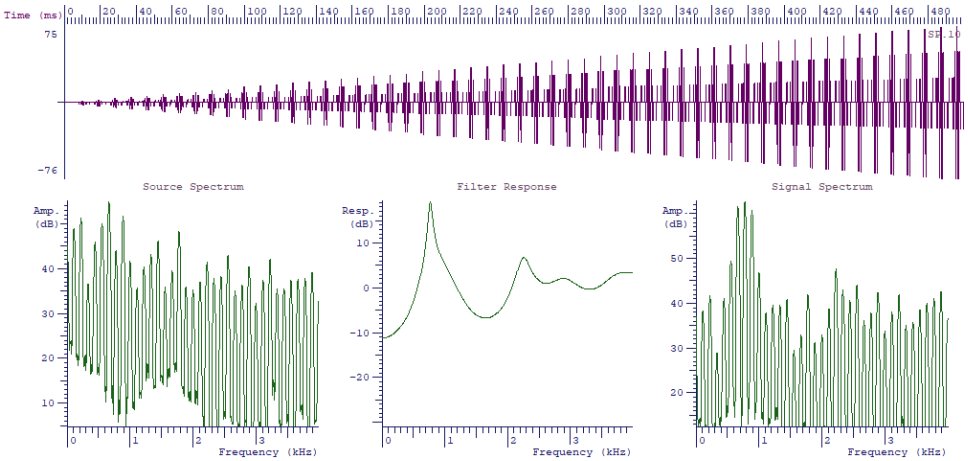


Figure 64:/ə/ sound

u_vowel.wav

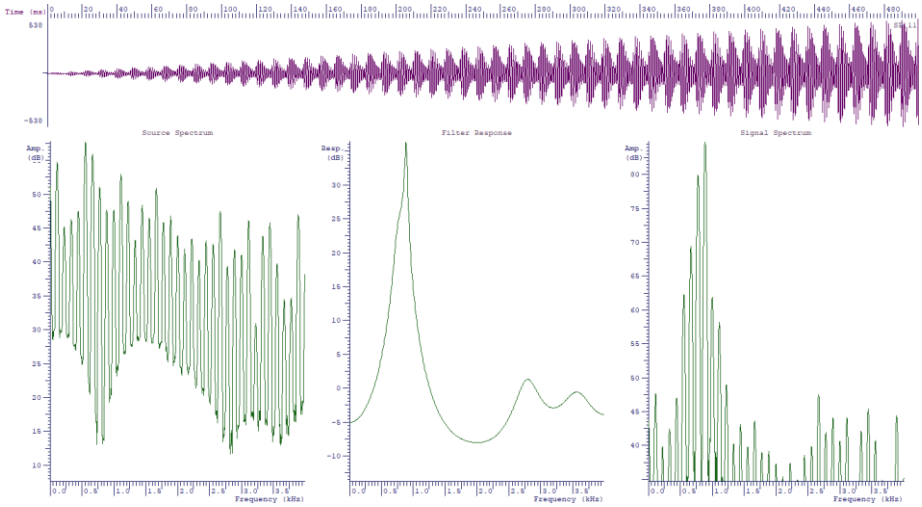


Figure 65:/u:/Sound

Part 8: Formant synthesis in SFS

Part A

- At First, we display the cross section of “ah” sound, thee we select the “a” vowel from “ah” sound. From cross section we found that the frequency F1=784Hz and frequency F2 = 1410 Hz.

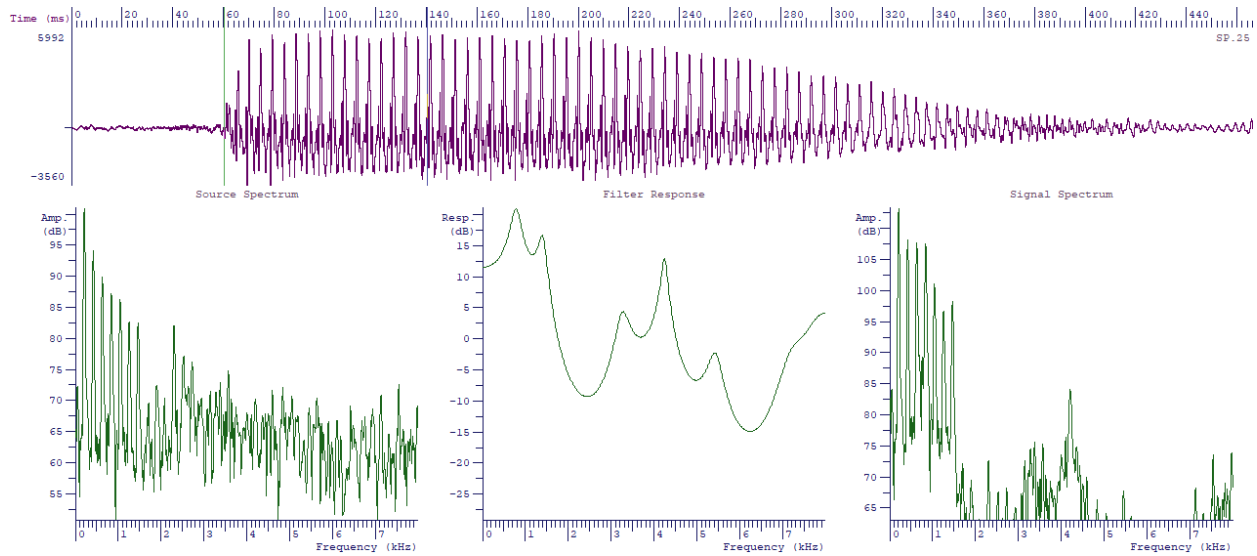


Figure 66:“ah” cross section

- After that, we recorded the life word and then display the waveform and wideband spectrogram.

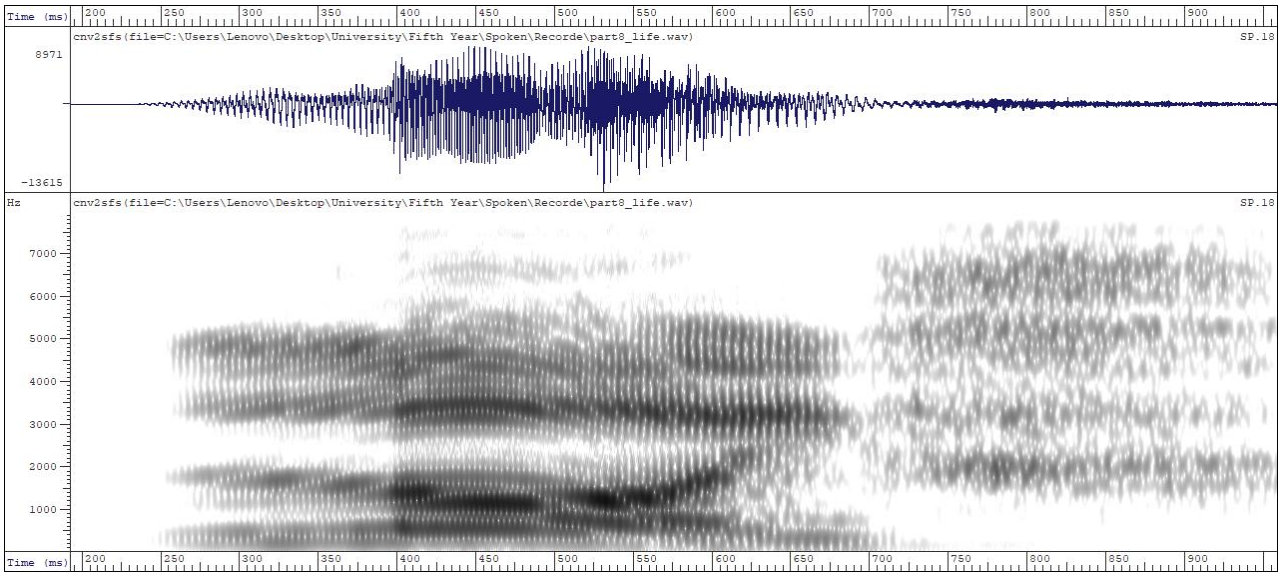


Figure 67: “life” waveform and wideband spectrogram

orginal_parameter_paer8.txt - Notepad

File Edit Format View Help

```
item=12.01 file=C:\Users\Lenovo\Desktop\University\Fifth Year\Spoken\Assigment.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   45.1; 1585   89 49; 3328  227 44; 4745  251 41; 6559  791 26;    0    0  0;
 20.0 20.0 1   45.1;  999  189 32; 1643   43 40; 3225  149 27; 4311  109 28; 4937  309 15;
 30.0 20.0 0   44.5; 1539  111 47; 3386  163 46; 4550  133 45; 7038  808 23;    0    0  0;
 40.0 20.0 0   42.2; 1670  169 41; 3599  441 38; 4678  213 42; 7200 1313 21;    0    0  0;
 50.0 20.0 1   42.1; 1028  265 30; 1848  112 32; 3182  129 24; 4278  150 22; 4977  401 11;
 60.0 20.0 1   42.0; 1689   78 32; 3303  181 21; 4298  206 18; 4977  161 17; 6979  136  4;
 70.0 20.0 1   41.8; 1052  108 33; 1822  120 28; 3223  264 17; 4861  245 12; 6989  371  0;
 80.0 20.0 1   42.6; 1097  185 28; 1731  136 27; 3503  266 14; 4854  147 11; 5968  259  3;
 90.0 20.0 1   40.3;  932  171 29; 1795  238 21; 3508  444  9; 4831  286  6; 6919  222  3;
100.0 20.0 0   39.6; 1107  262 35; 2319 1280 24; 3857  657 24; 6035  413 25;    0    0  0;
110.0 20.0 0   39.5; 1531 1275 34; 1547  399 34; 4413  665 24; 6564  833 23;    0    0  0;
120.0 20.0 1   37.9; 1172  211 25; 1861  243 18; 3051  238  9; 4021  245  6; 5130  270  3;
130.0 20.0 1   38.3; 1153  222 24; 1841  285 16; 3083  233  7; 4289  194  5; 7009  180  3;
140.0 20.0 1   40.0; 1010   84 30; 1768   95 23; 2890  380  4; 5773  241  3; 6952  144  4;
150.0 20.0 1   41.5;  958  103 29; 1935  139 19; 2907  295  8; 5894  356  3; 7462  331  0;
160.0 20.0 1   41.0; 1054   88 32; 1941  103 22; 3727  150  6; 4898  179  2; 6714  265  0;
170.0 20.0 1   40.5; 1141   40 36; 1955  289 14; 3097  286  7; 4087  285  4; 6846  226  1;
180.0 20.0 1   39.3; 1062   75 33; 1908  243 19; 3776  290  3; 5172  302  1; 7321  426  0;
190.0 20.0 1   37.8; 1098  106 27; 1971  267 13; 4679  316  2; 5815  196  3; 6784  277  0;
200.0 20.0 1   39.2; 1078  231 24; 2085  164 16; 3336   88 15; 5973  159  6; 7006  138  5;
210.0 20.0 1   41.0; 1123  176 27; 2033  320 14; 4425  188  7; 5323  273  3; 6239  194  5;
220.0 20.0 1   39.9; 1046  135 27; 1839  187 21; 3591  348  3; 4884  201  8; 7123  330  1;
230.0 20.0 0   44.1; 1542  650 30; 2455 2220 23; 4784   56 43; 6990  624 21;    0    0  0;
240.0 20.0 1   61.7;  301   44 61; 1101  277 31; 1847   88 30; 3414   38 28; 4745   26 31;
250.0 20.0 1   67.6;  349   45 66; 1773   17 52; 3329   79 34; 4616  111 35; 4955   65 35;
260.0 20.0 1   72.4;  361   50 70; 1776   22 59; 3347   53 47; 4610  148 46; 4894   54 49;
270.0 20.0 1   75.7;  355   70 72; 1789   18 65; 3370   34 57; 4644   99 50; 4949   33 55;
280.0 20.0 1   79.0;  371   60 76; 1793   24 67; 3373   42 60; 4701   61 58; 4969   40 58;
290.0 20.0 1   81.2;  408   49 79; 1779   76 63; 3323   39 62; 4931   53 60; 6609  121 26;
300.0 20.0 1   83.1;  412   39 82; 1746   84 65; 3330   29 66; 4657  110 58; 4961   59 59;
310.0 20.0 1   85.0;  371   68 81; 1677   39 72; 3358   24 69; 4714   90 61; 4969  167 54;
320.0 20.0 1   87.0;  303   87 83; 1684   20 78; 3402   23 71; 4588  144 58; 4951   88 57;
330.0 20.0 1   87.6;  341   83 84; 1701   15 79; 3390   49 66; 4630   96 61; 4954   69 60;
340.0 20.0 1   87.0;  378   84 82; 1678   57 71; 3363   48 67; 4596   97 62; 4953  101 57;
350.0 20.0 1   85.3;  383   99 79; 1616  106 70; 3340   31 70; 4583  101 62; 4899   57 63;
360.0 20.0 1   85.8;  429   82 81; 1555   33 79; 3385   30 72; 4589   78 65; 4919   46 64;
370.0 20.0 1   88.5;  455   93 83; 1567   22 82; 3414   42 69; 4692  130 67; 4875   85 67;
380.0 20.0 1   89.8;  448   76 86; 1563   56 77; 3329   53 69; 4483   94 66; 4897   31 69;
390.0 20.0 1   90.5;  482   58 87; 1408   66 80; 3319   41 70; 4427   98 65; 4901   62 62;
400.0 20.0 1   95.1;  591   42 93; 1392   61 89; 3303   38 77; 4374  119 69; 4852  117 61;
410.0 20.0 1   96.7;  608   71 91; 1384   57 91; 3335   54 78; 4454   97 67; 5084  142 58;
420.0 20.0 1   96.2;  621  101 89; 1332  101 88; 3308   69 76; 3663  191 69; 4576   75 69;
430.0 20.0 1   96.0;  606   87 90; 1196   56 90; 3302   63 76; 3651  120 71; 4607   51 71;
440.0 20.0 1   96.6;  606   78 91; 1180   40 93; 3281   55 78; 3656  107 73; 4545   84 68;
450.0 20.0 1   97.5;  616   83 91; 1172   31 95; 3290   60 77; 3668  122 72; 4444   82 68;
460.0 20.0 1   97.5;  623   77 90; 1164   24 97; 3295   64 77; 4446   51 70; 5253  110 55;
470.0 20.0 1   97.0;  610   65 91; 1150   18 98; 3339   56 77; 4474   57 67; 5314   90 56;
480.0 20.0 1   96.3;  596   75 89; 1140   19 97; 3360   29 81; 4435   75 65; 5264  107 54;
```

Figure 68:original parameter

- synthesized original

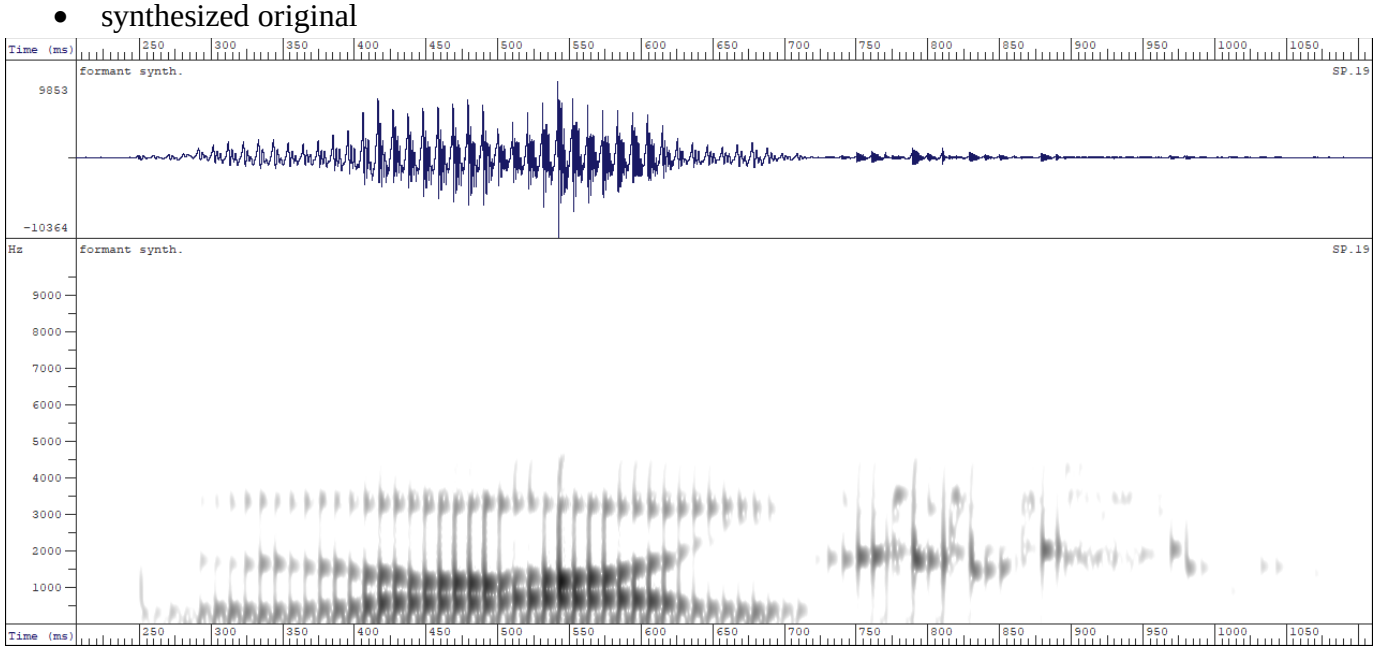


Figure 69: synthesized original

- From life word we found the ‘i’ time from 440ms to 720ms.

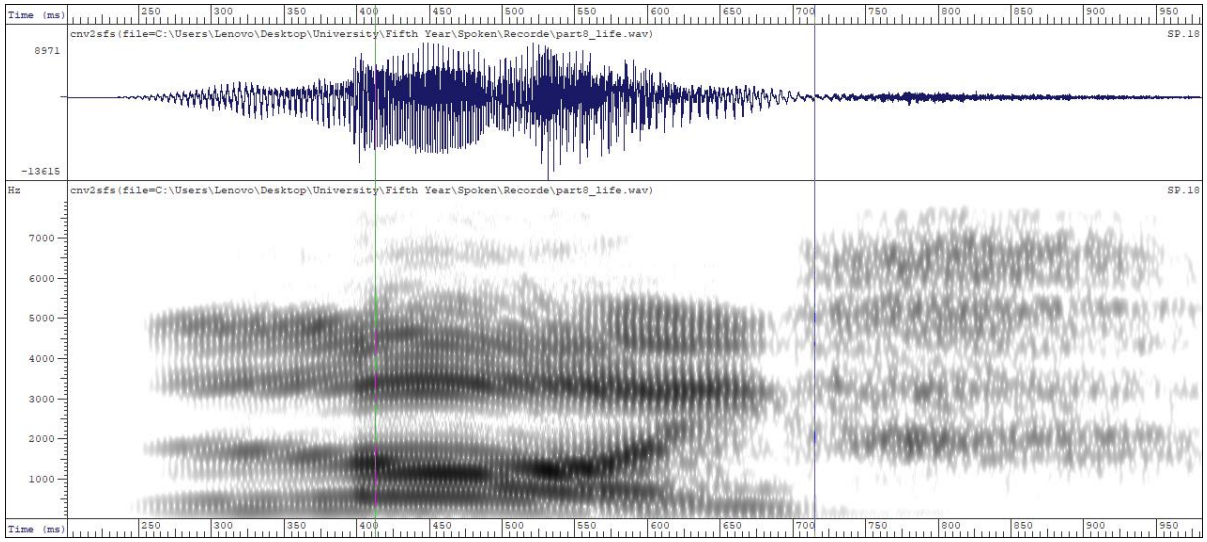


Figure 70: “I” from life word

- Then we put the value of F1= 784 Hz and F2= 1410 Hz in the file from 440ms to 720 ms.

```
430.0 20.0 1   96.0;  606   87 90;  [illegible]   56 90; 3302   63 76; 3651   120 71; 4607   51 71;
440.0 20.0 1   96.6;  784   78 91; 1410   40 93; 3281   55 78; 3656   107 73; 4545   84 68;
450.0 20.0 1   97.5;  784   83 91; 1410   31 95; 3290   60 77; 3668   122 72; 4444   82 68;
460.0 20.0 1   97.5;  784   77 90; 1410   24 97; 3295   64 77; 4446    51 70; 5253  110 55;
470.0 20.0 1   97.0;  784   65 91; 1410   18 98; 3339   56 77; 4474    57 67; 5314   90 56;
480.0 20.0 1   96.3;  784   75 89; 1410   19 97; 3360   29 81; 4435    75 65; 5264  107 54;
490.0 20.0 1   94.1;  784  124 85; 1410   70 87; 3316   24 79; 4430    70 65; 5305  148 50;
500.0 20.0 1   93.6;  784  101 89; 1410  116 80; 3286   68 73; 4417    60 66; 6569  178 39;
510.0 20.0 1   94.6;  784   66 90; 1410   42 87; 3369   52 75; 4447   164 62; 6601  439 36;
520.0 20.0 1   95.8;  784   54 91; 1410   14 96; 3307   71 73; 4419   165 60; 5417  284 46;
530.0 20.0 1   97.6;  784   41 92; 1410    7 104; 3337   38 77; 4466   115 63; 6759  143 37;
540.0 20.0 1   97.0;  784   56 92; 1410   33 94; 3326   46 77; 4399   110 65; 6687  119 40;
550.0 20.0 1   95.4;  784   57 89; 1410   14 97; 3282   49 77; 4402   112 64; 5162  248 49;
560.0 20.0 1   95.1;  784   65 88; 1410   21 96; 3274   32 81; 4379    98 64; 5196  128 53;
570.0 20.0 1   94.2;  784   62 90; 1410   40 91; 3217   53 78; 4488   163 60; 5212  106 56;
580.0 20.0 1   93.7;  784   54 89; 1410   65 86; 3209   46 78; 4611    73 67; 5173   64 62;
590.0 20.0 1   92.9;  784   55 89; 1410   58 86; 3212   54 78; 4532    77 68; 5185   75 62;
600.0 20.0 1   91.9;  784   51 87; 1410   42 85; 3170   54 80; 4452    67 68; 5114  145 61;
610.0 20.0 1   89.4;  784   72 83; 1410   99 76; 3182   75 78; 4426    81 68; 5137   62 64;
620.0 20.0 1   87.0;  784   80 80; 1410  111 71; 3202   75 79; 4453    94 67; 5088   49 65;
630.0 20.0 1   86.2;  784   72 81; 1410  299 64; 3217   77 78; 4448    87 66; 5002   73 62;
640.0 20.0 1   85.5;  784  123 77; 1410  403 59; 3237   62 78; 4504   152 62; 5009   80 60;
650.0 20.0 1   85.7;  784  118 79; 1410  198 68; 3299   85 74; 4438   128 62; 4987  100 60;
660.0 20.0 1   86.1;  784   71 82; 1410  132 67; 3287   62 72; 4410   111 61; 4928   95 57;
670.0 20.0 1   85.7;  784   54 83; 1410  198 61; 3216   60 70; 4399   152 53; 4904   70 54;
680.0 20.0 1   84.4;  784  117 73; 1410  207 53; 3200   36 66; 4400    92 50; 4920   50 51;
690.0 20.0 1   81.8;  784  173 72; 1410  132 52; 3158   96 53; 4373   113 42; 5009   92 40;
700.0 20.0 1   79.1;  784  154 [illegible] 1410 [illegible] 51; 3196  101 49; 5070    74 45; 6790  134 32;
710.0 20.0 1   74.[illegible]  784  [illegible]8 5[illegible] 1410  82 52; 5087   131 48; 6111   217 41; 6723   97 43;
720.0 20.0 1   71.4;  784  168 52; [illegible]  78 54; 5101   113 48; 6087   149 45; 6746   143 40;
```

Figure 71: Modify f1 & f2

- Below figure shows **synthesized modified** the after update the parameter.txt file, with F1 and F2 values from 440ms to 720ms.

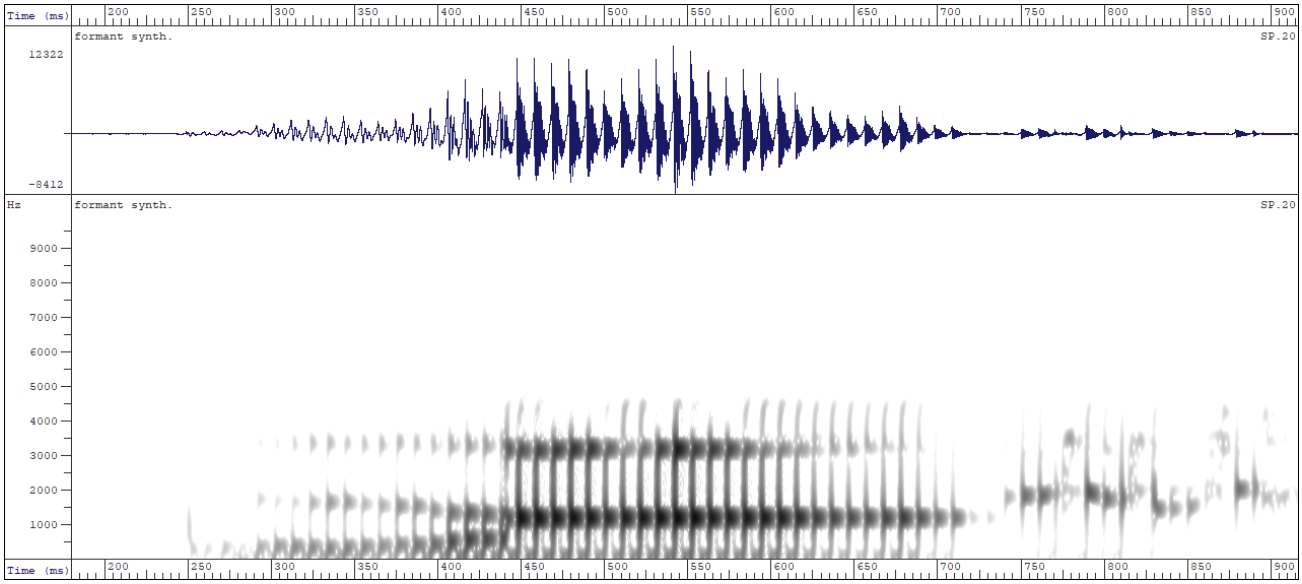


Figure 72: ***synthesized modified***

Modified formant data will produce speech that differs from the original word "life" (/laIf/). Instead of transitioning from /a/ to /I/, the diphthong maintains a consistent /a/ vowel sound. This modification can make speech less natural and difficult to understand, perhaps resulting in a longer /a/ sound or another term.

Part B

To change a speaker's voice in a formant-based speech synthesizer, alter the formant filter parameters to approximate the resonance frequencies and vocal tract characteristics of the new speaker. A formant-based speech synthesizer adjusts formant frequencies, bandwidths, and amplitudes proportionally based on voice variations between the source and target speakers. For example, if the target speaker has higher formant frequencies than the source speaker, the synthesizer's formant frequencies would be accordingly increased to match. To maintain the organic feel of the synthetic voice, proportional modifications to bandwidths and amplitudes will be performed.