BIRZEIT UNIVERSITY

Electrical and Computer Engineering

Spoken Language Processing - Spring 2024

Name:	**ID:**
Rahaf Naser	**1201319**
Shatha Khdair	**1200525**

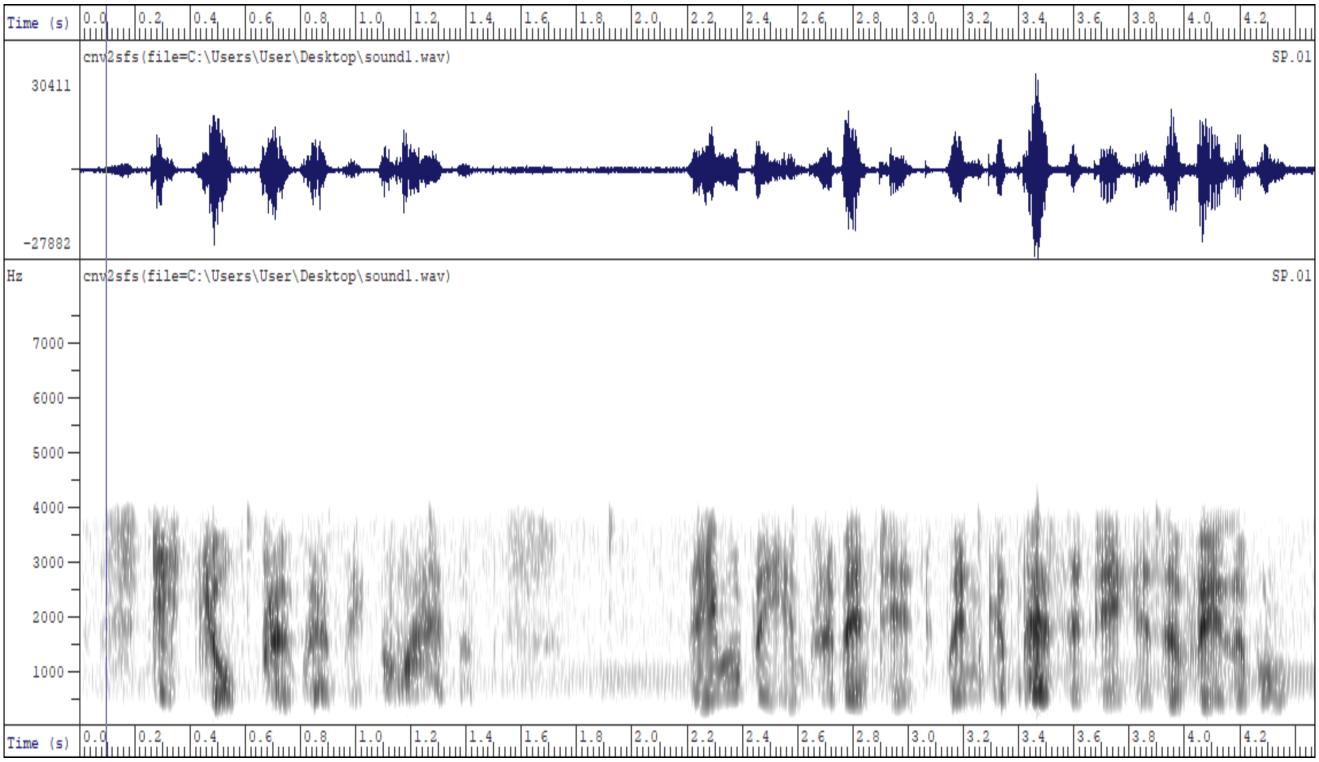
Section: 2

Date: 29/4/2024

Part 1 – Basic sound analysis, Spectrograms

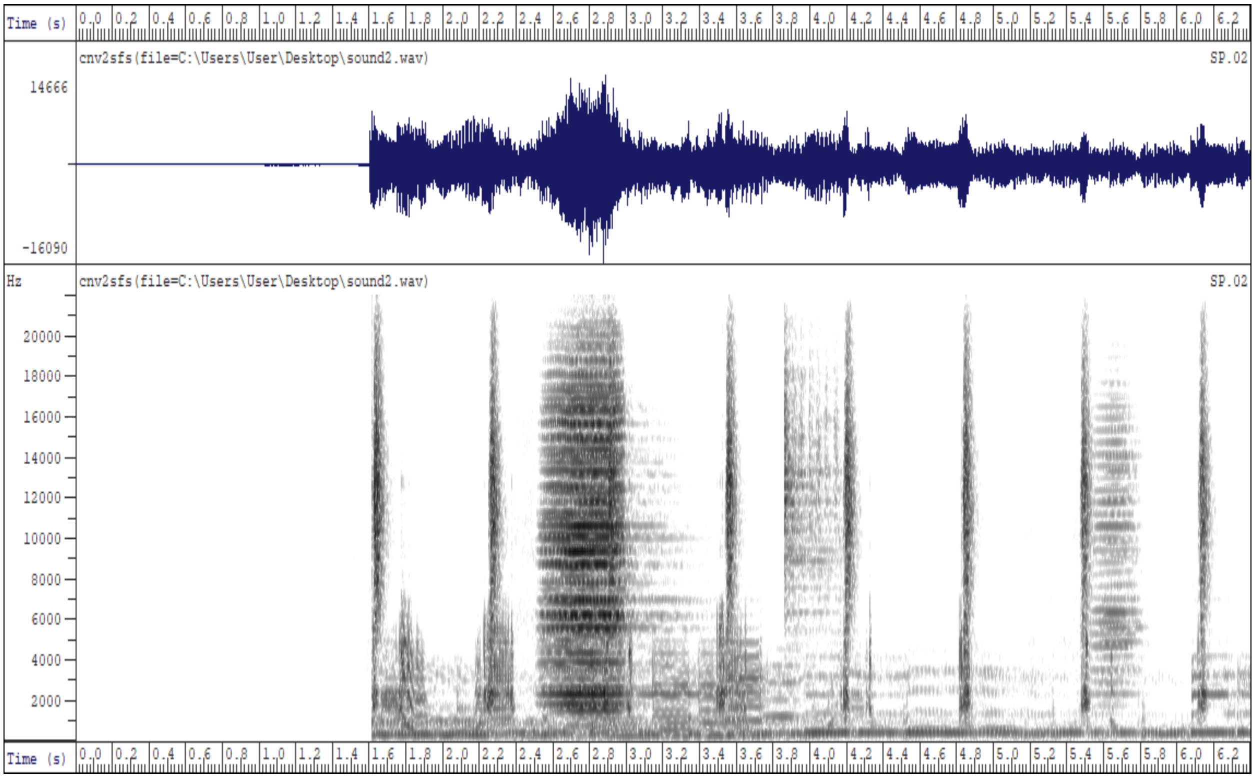
a) Bandwidth

sound1.wav:



this is the waveform and wide band spectrogram of sound1.wav, we find from the spectrogram that the bandwidth equal 4000 Hz, and the lower frequency equal 500 Hz, and upper frequency equal 4000 Hz.

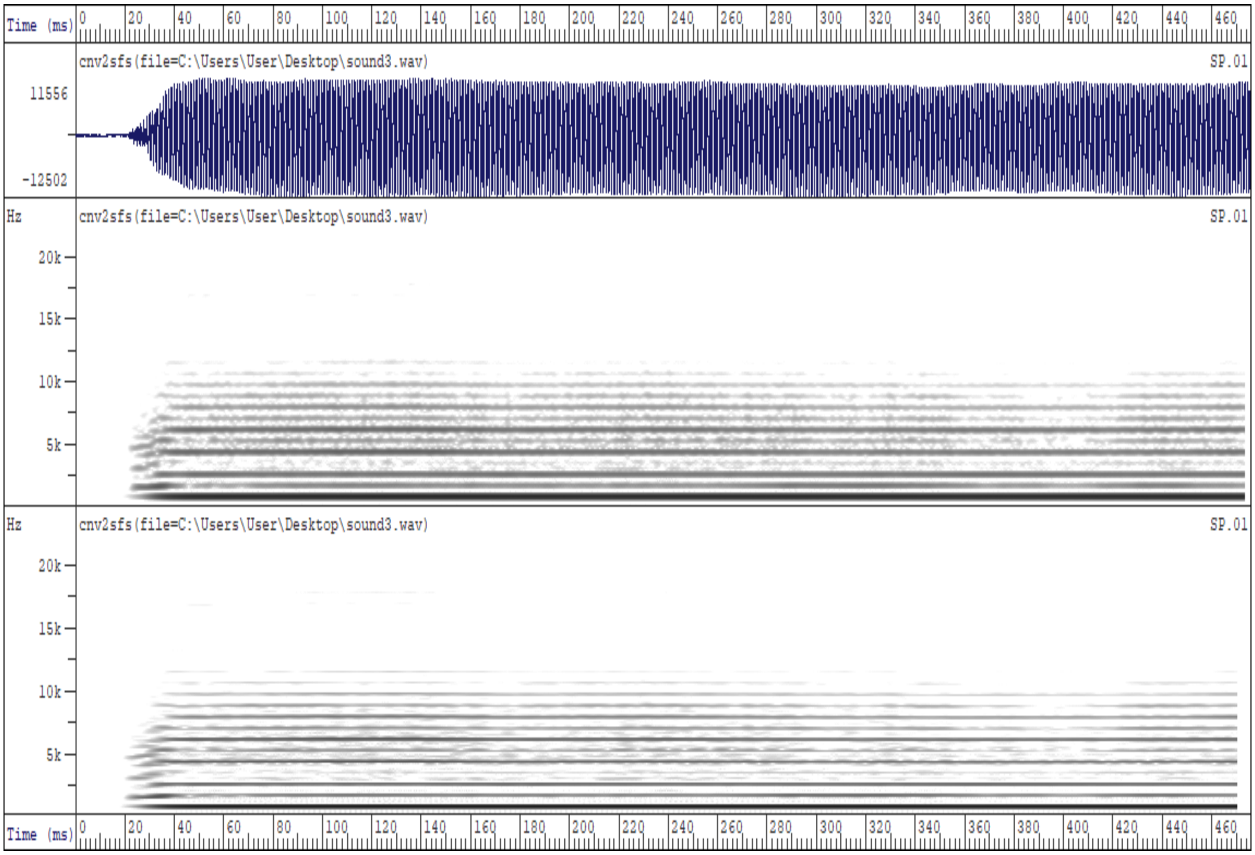
Sound2.wav:



this is the waveform and wide band spectrogram of sound2.wav, we find from the spectrogram that the bandwidth equal 22000 Hz, and the lower frequency equal 100 Hz, and upper frequency equal 22000 Hz.

b) Spectrogram

Sound3.wav

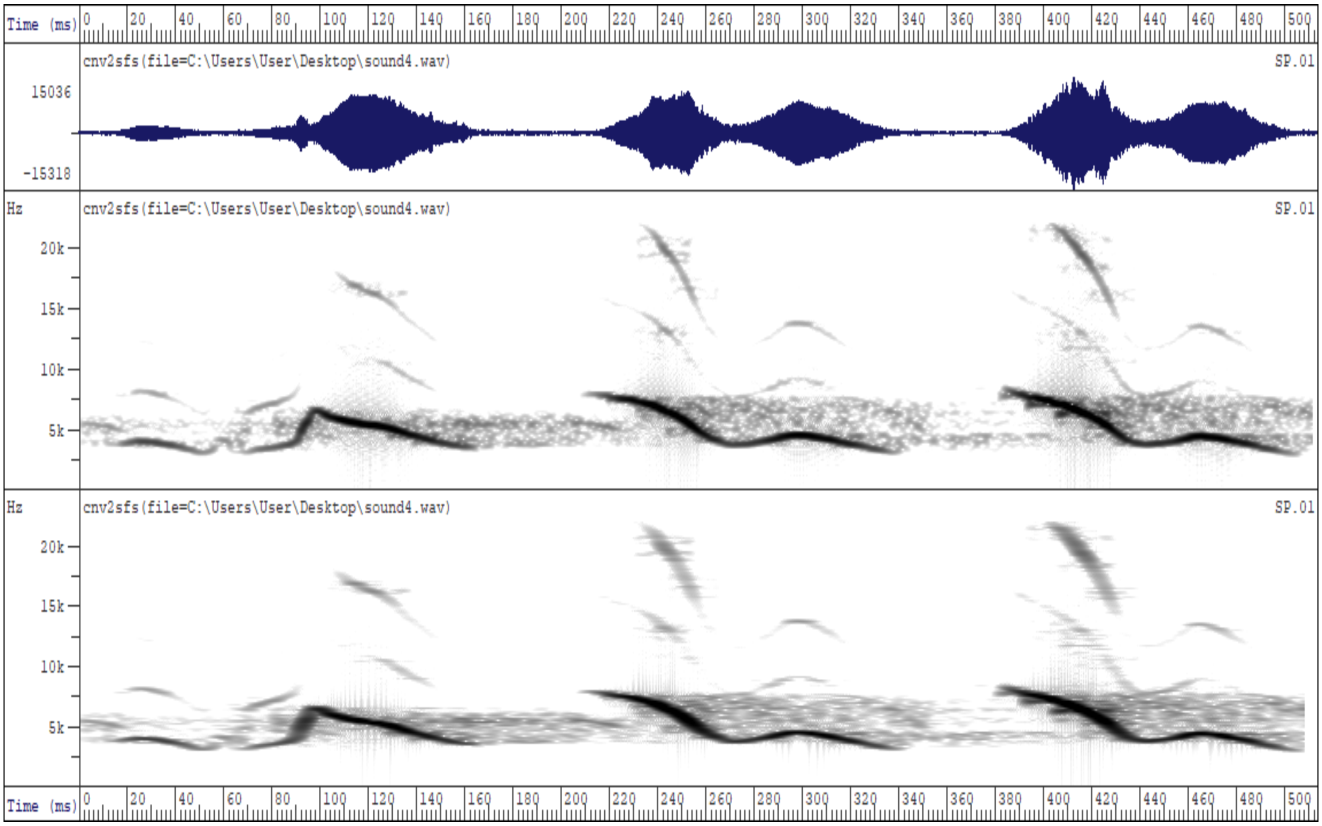


This is the waveform and wide band and narrow band spectrogram of sound3.wav.

We notice that the narrowband spectrogram exhibits the harmonic structure in the form horizontal striations and in this case the frequency resolution will be the best.

The wideband spectrogram exhibits periodic temporal structure in the form of vertical striations and in this case the frequency resolution will be the worst, and it is good for temporal resolution.

Sound4.wav

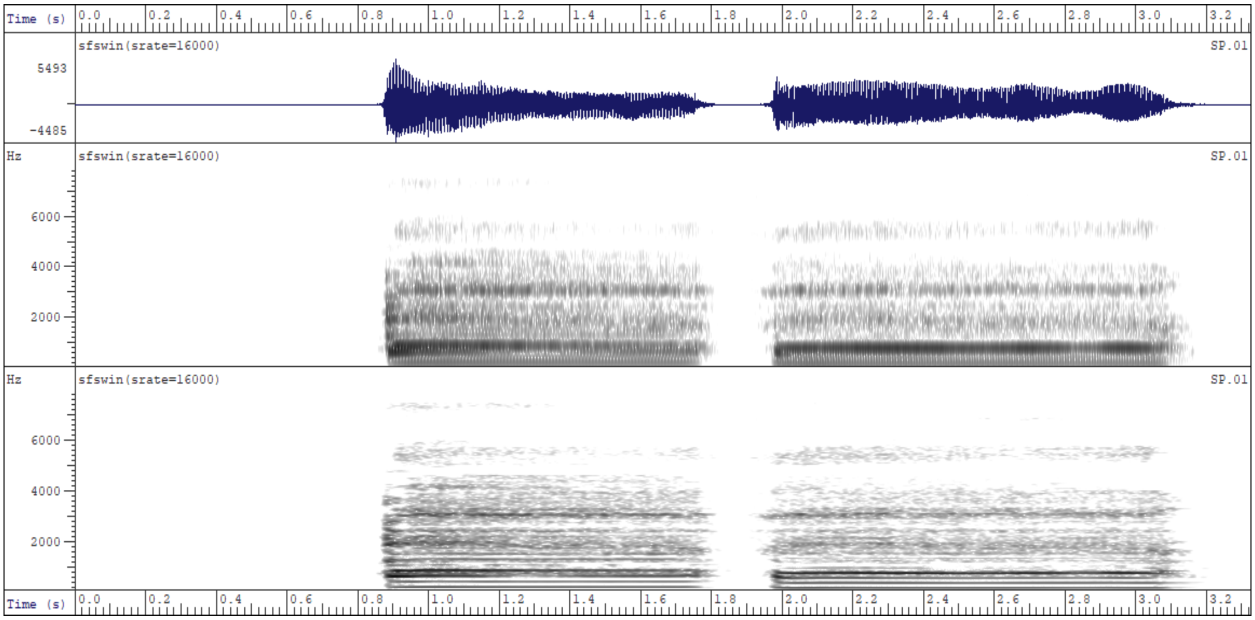


This is the waveform and wide band and narrow band spectrogram of sound4.wav.

We notice that the narrowband spectrogram exhibits the harmonic structure in the form horizontal striations and in this case the frequency resolution will be the best.

The wideband spectrogram exhibits periodic temporal structure in the form of vertical striations and in this case the frequency resolution will be the worst, and it is good for temporal resolution.

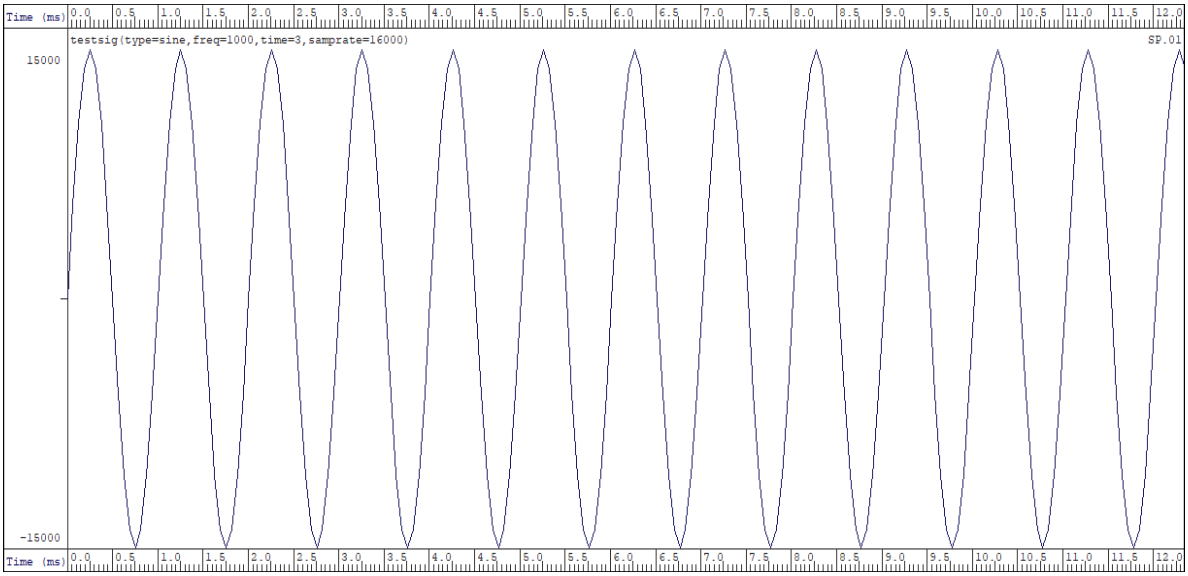
c) In this part I record my speech uttering the syllable 'afa' at sampling rate 16KH



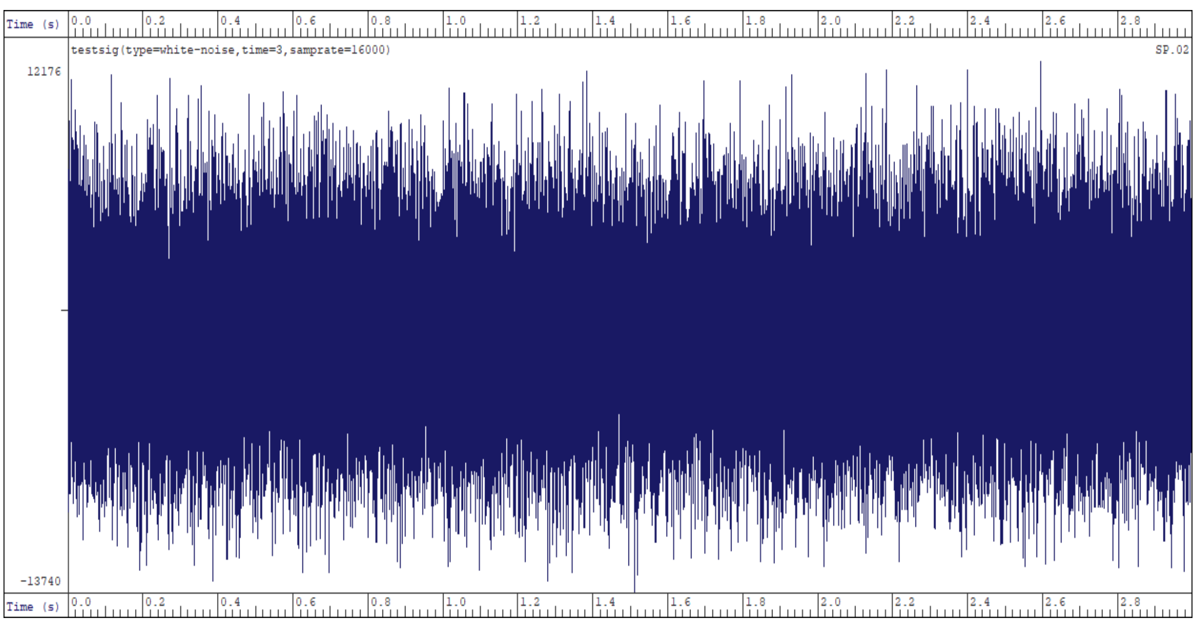
This is the waveform and wide band and narrow band spectrogram of 'afa' we notice that 'afa' contain 2 voiced (periodic) part (vowel 'a') and one unvoiced part 'f'. we notice from spectrogram that in voiced parts the power positioned in low frequencies , and in unvoiced parts the power positioned in high frequencies.

Part 2 – Filters

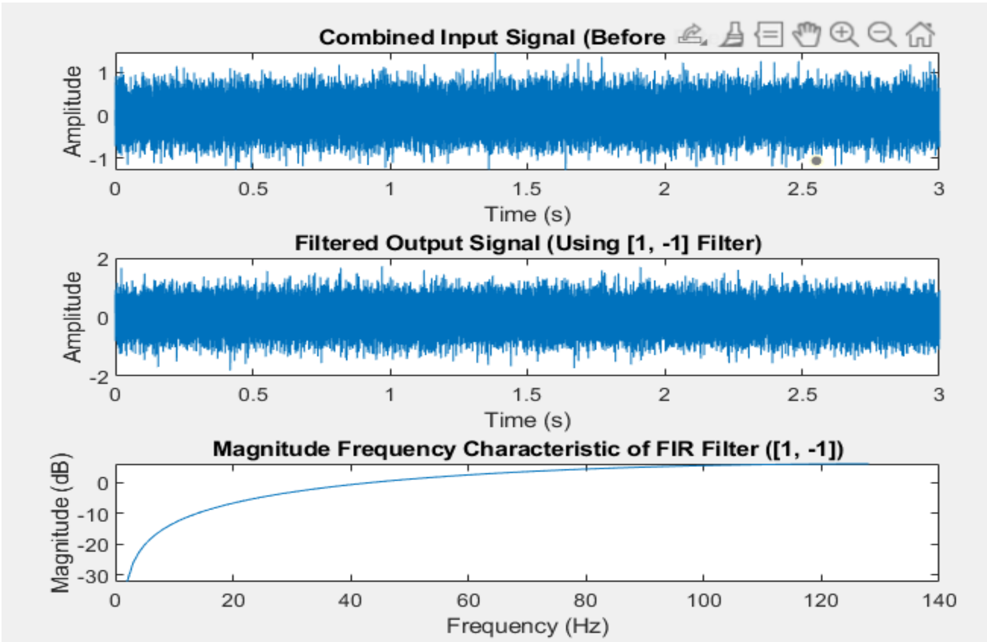
Sinousoidal signal:



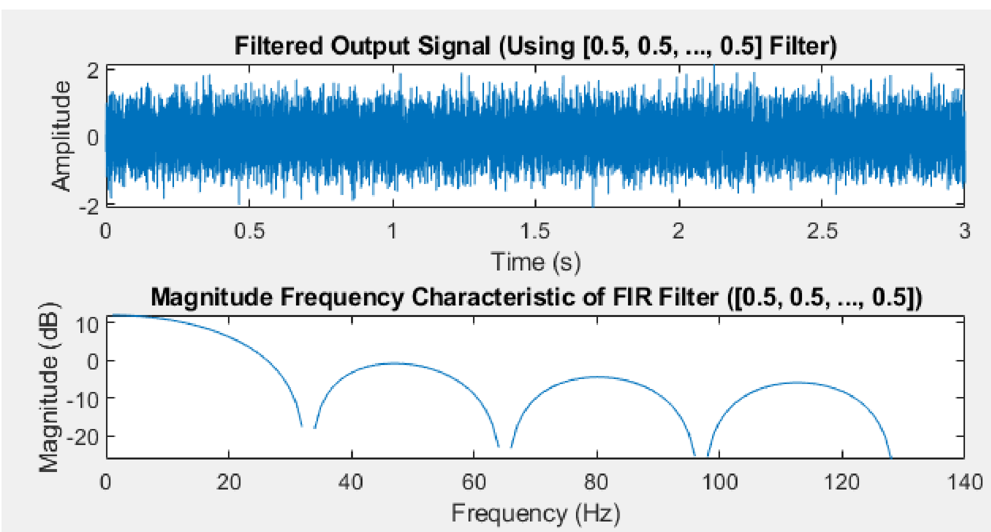
White noise signal:



Combined signal and result of first filter:



Result of second filter:



FIR Filter with Impulse Response {1, -1}:

Sine-wave Strength: This filter is designed to act as a simple differentiator. It emphasizes high-frequency components in the input signal. Therefore, the sine-wave component, being a 1 kHz signal, might be preserved in the output with minimal attenuation.

Noise Attenuation: As white noise contains a broad spectrum of frequencies, this filter may attenuate high-frequency noise components more effectively than low-frequency ones. Thus, the noise may be somewhat reduced in the output.

FIR Filter with Difference Equation Coefficients:

Sine-wave Strength: The coefficients in the difference equation suggest that this filter is a type of low-pass filter. It emphasizes lower frequencies and attenuates higher frequencies. Therefore, the output may have a weaker representation of the 1 kHz sine-wave compared to the input.

Noise Attenuation: As a low-pass filter, it may attenuate higher-frequency noise components more effectively, resulting in a reduction of noise in the output signal. However, it may also introduce some phase distortion due to the recursive nature of the filter.

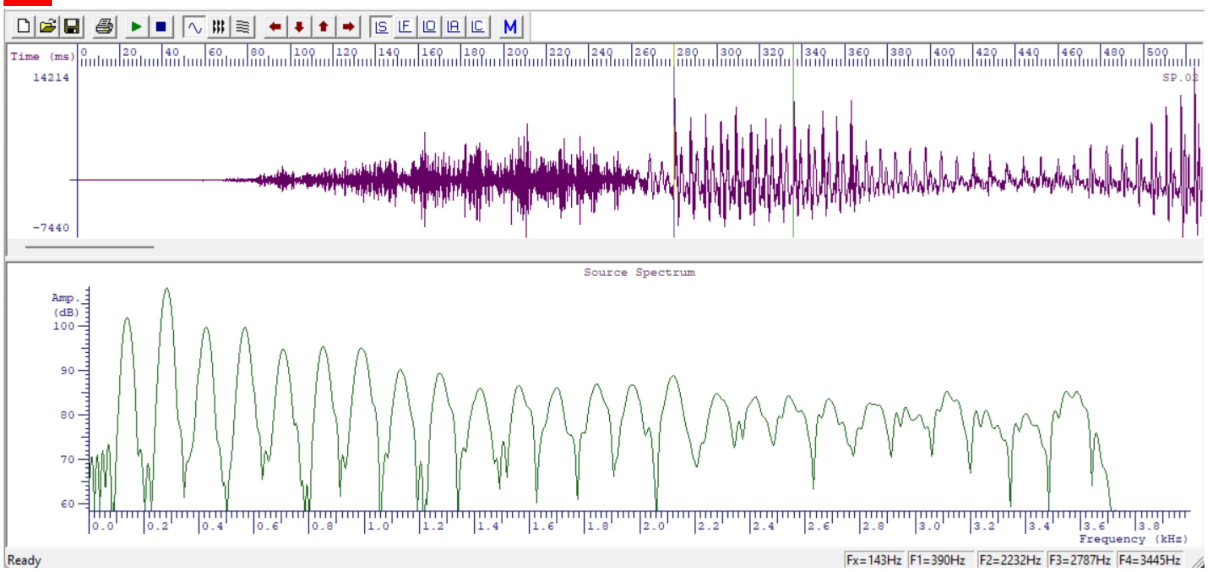
Part 3 – Speech Analysis

a) Formants for vowels

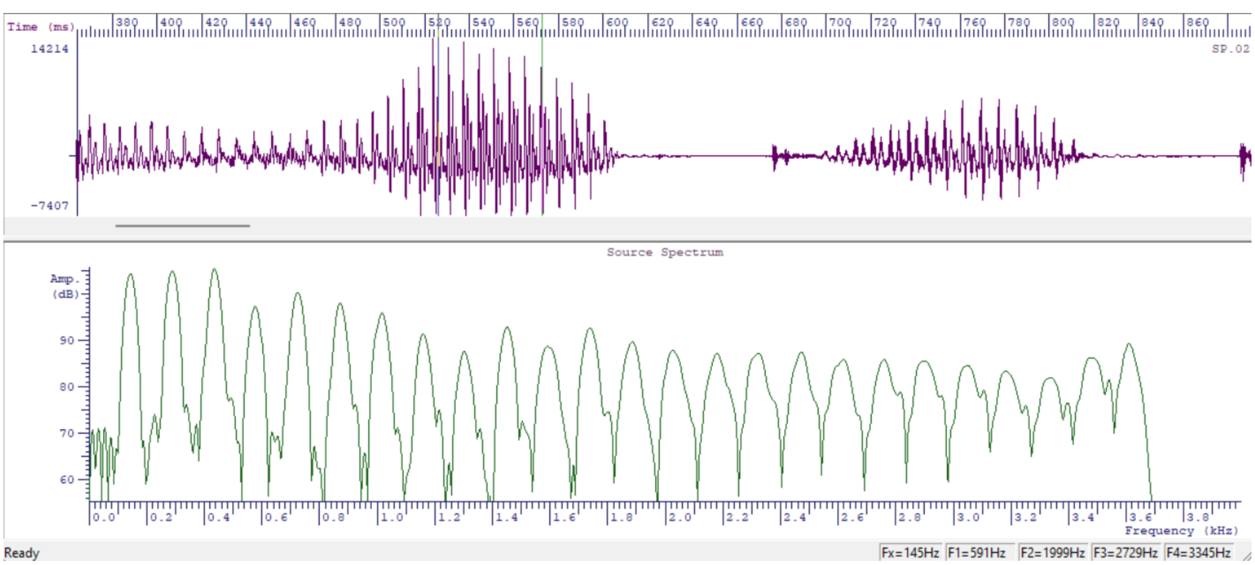
In this part I estimate the formant frequencies F1,F2,F3, of four vowels 'e' in she, 'a' in had , 'u' in suit 'a' in dark from Sample1.wav , my recording , and my partner recording.

Sample1.wav:

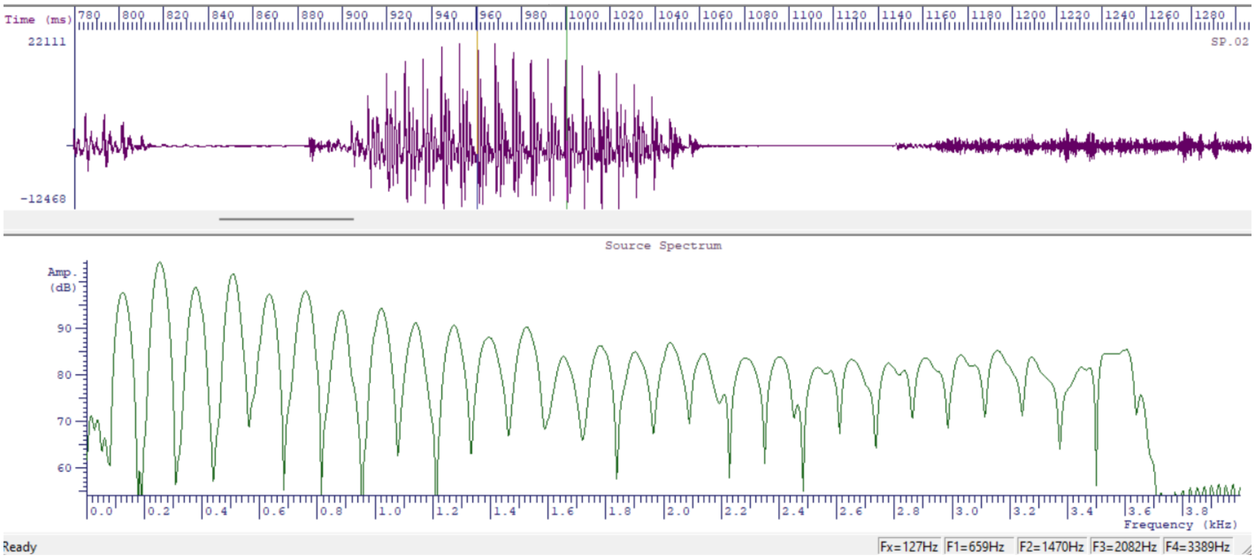
She



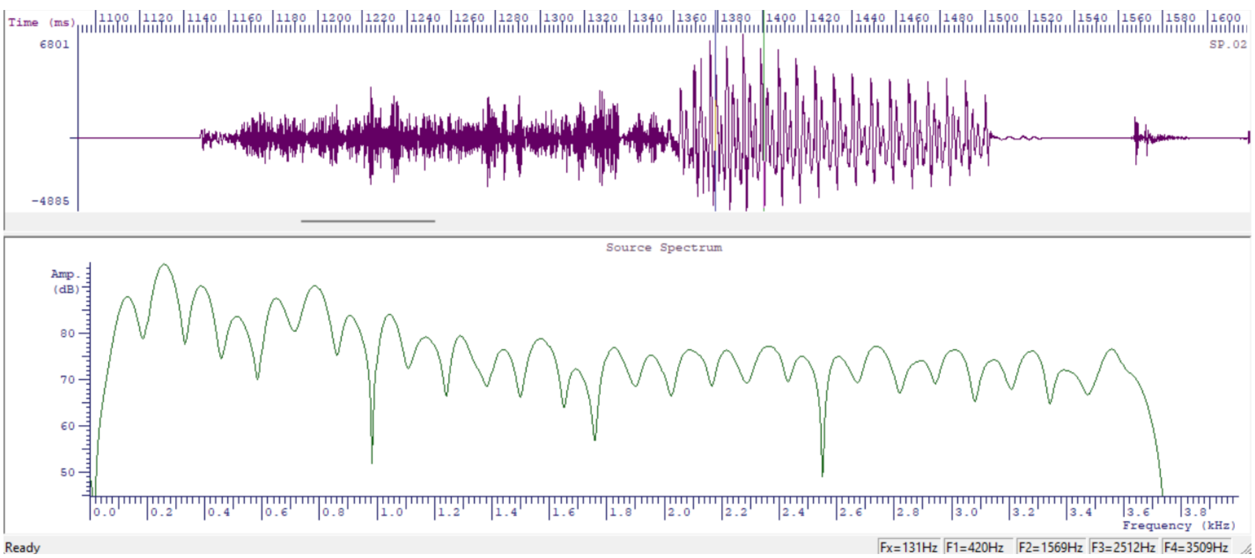
Had



Dark


Time (ms)
22111
-12468
SP.02
Source Spectrum
Amp. (dB)
Frequency (kHz)
Ready
Fx=127Hz F1=659Hz F2=1470Hz F3=2082Hz F4=3389Hz

Suit

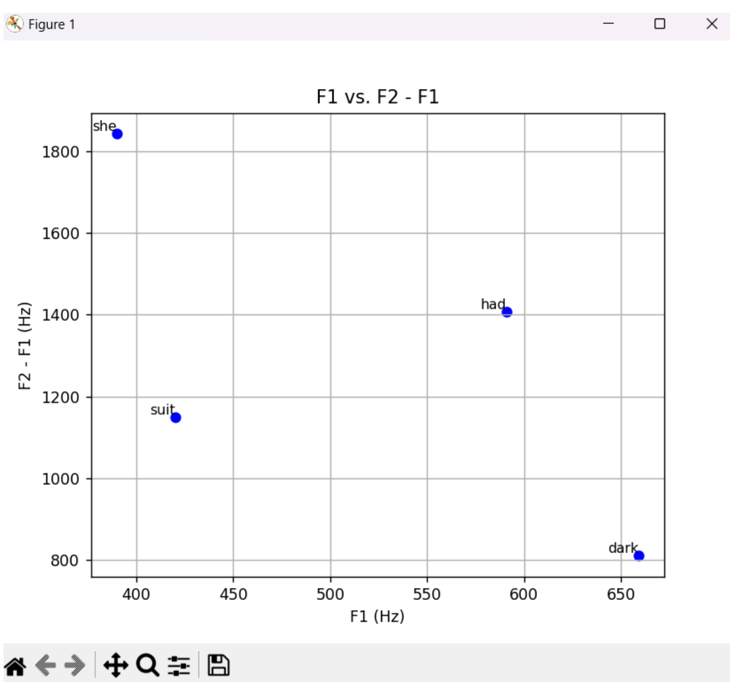

Time (ms)
6801
-4885
SP.02
Source Spectrum
Amp. (dB)
Frequency (kHz)
Ready
Fx=131Hz F1=420Hz F2=1569Hz F3=2512Hz F4=3509Hz

Sample1.wav formants table:

Formants	F1 (HZ)	F2 (HZ)	F3 (HZ)
she	390	2232	2787
had	591	1999	2729
dark	659	1470	2082
suit	420	1569	2512

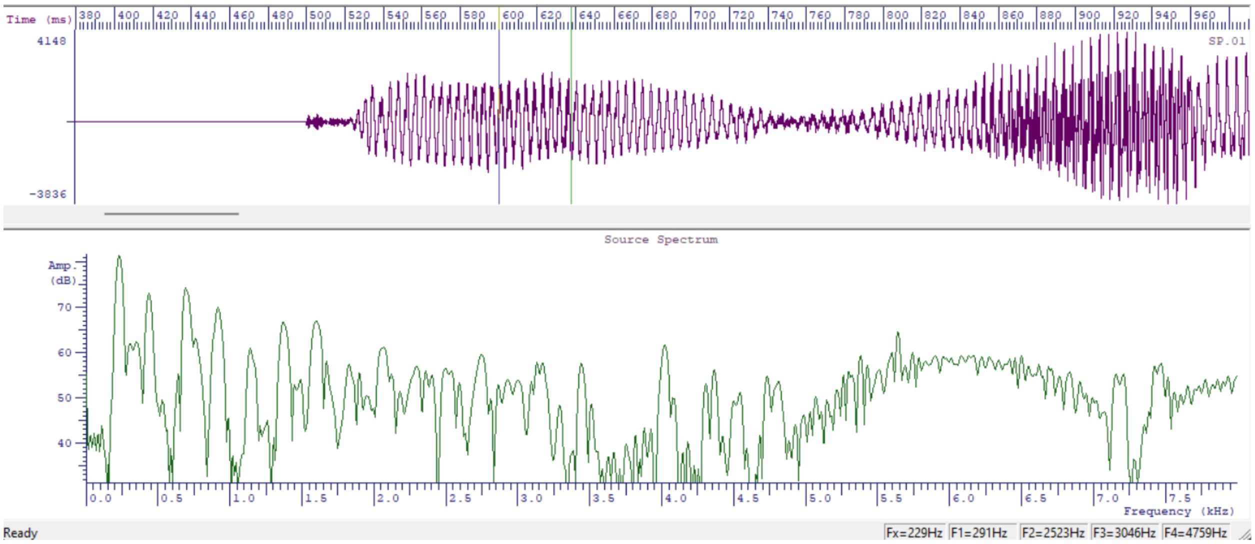
Python code to plot F1 versus F2-F1:

```python
import matplotlib.pyplot as plt

# Formant data
words = ['she', 'had', 'dark', 'suit']
f1_values = [390, 591, 659, 420]
f2_values = [2232, 1999, 1470, 1569]

# Calculate F2 - F1
f2_minus_f1 = [f2 - f1 for f1, f2 in zip(f1_values, f2_values)]

# Plot
plt.figure(figsize=(8, 6))
plt.scatter(f1_values, f2_minus_f1, color='blue')
plt.title('F1 vs. F2 - F1')
plt.xlabel('F1 (Hz)')
plt.ylabel('F2 - F1 (Hz)')
for i, word in enumerate(words):
    plt.text(f1_values[i], f2_minus_f1[i], word, fontsize=9, ha='right', va='bottom')
plt.grid(True)
plt.show()
```

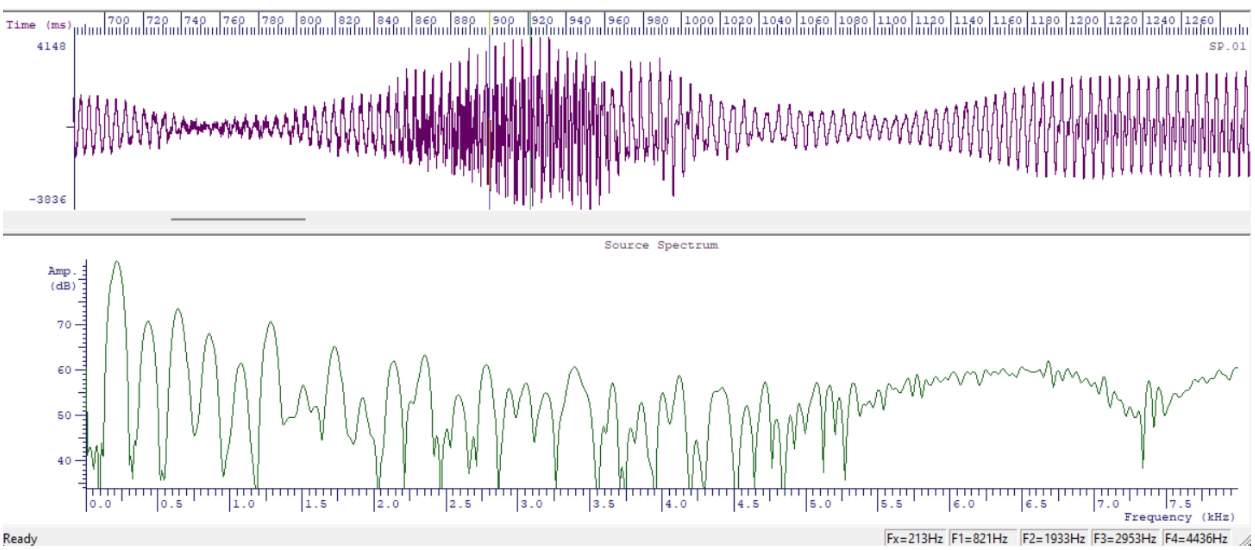


Redording1.wav

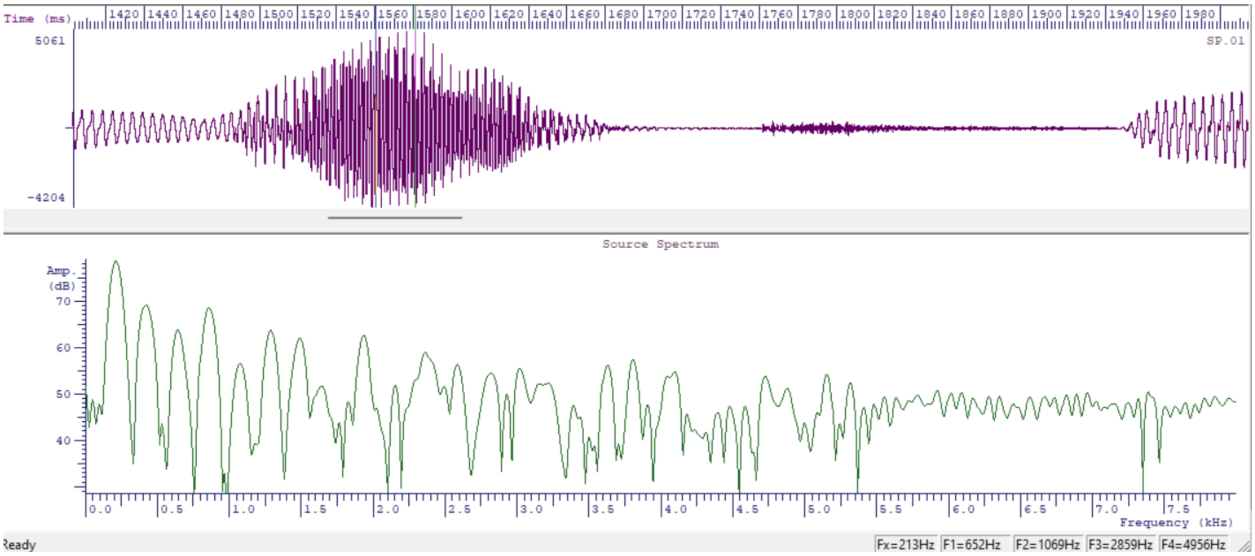
she



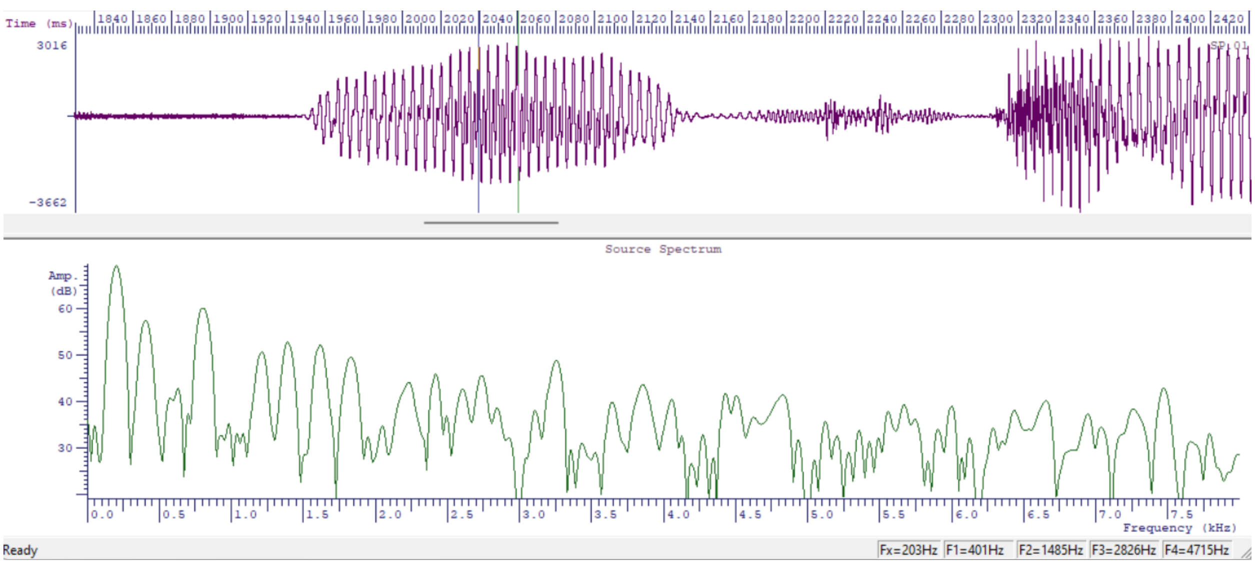
Had



Dark

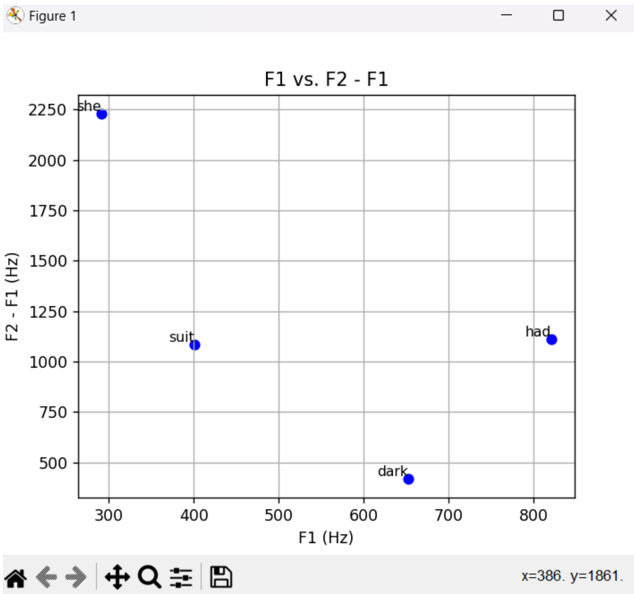

Time (ms)
5061
-4204
SP.01
Source Spectrum
Amp. (dB)
Frequency (kHz)
Ready
Fx=213Hz
F1=652Hz
F2=1069Hz
F3=2859Hz
F4=4956Hz

Suit


Time (ms)
3016
-3662
Source Spectrum
Amp. (dB)
Frequency (kHz)
Ready
Fx=203Hz
F1=401Hz
F2=1485Hz
F3=2826Hz
F4=4715Hz

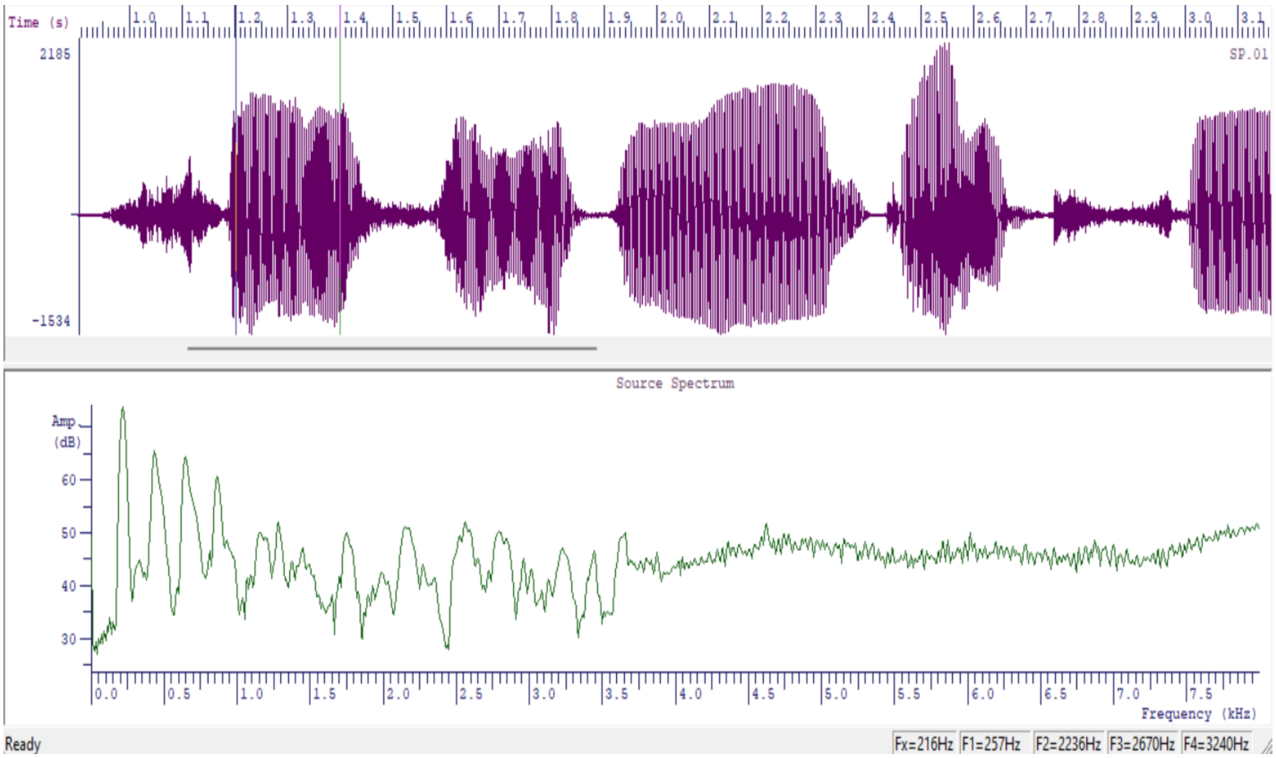
Recording1.wav formants table:

Formants	F1 (HZ)	F2 (HZ)	F3 (HZ)
she	291	2523	3046
had	821	1933	2953
dark	652	1069	2859
suit	401	1485	2826

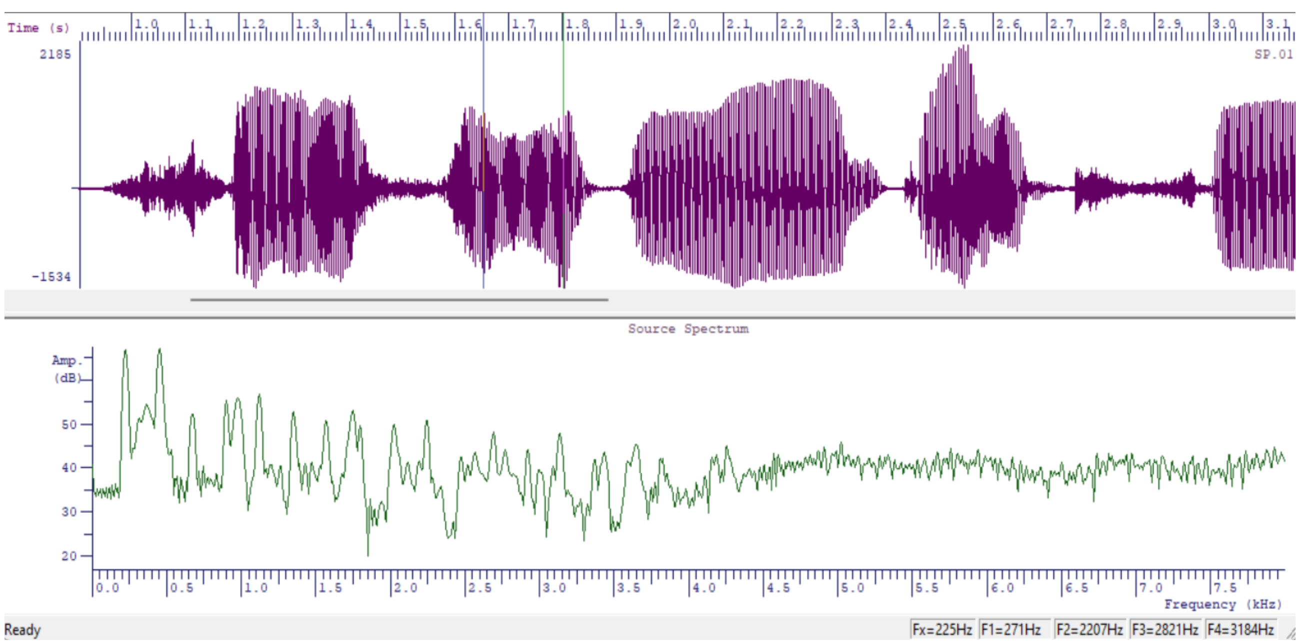


Recording2.wav

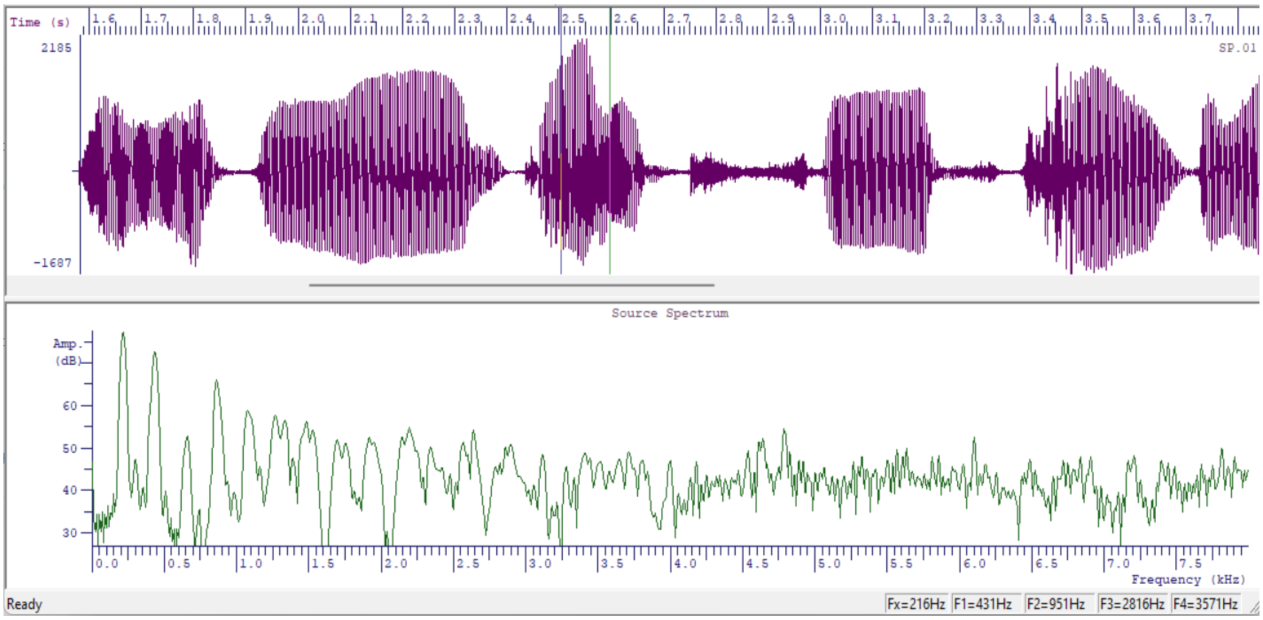
She



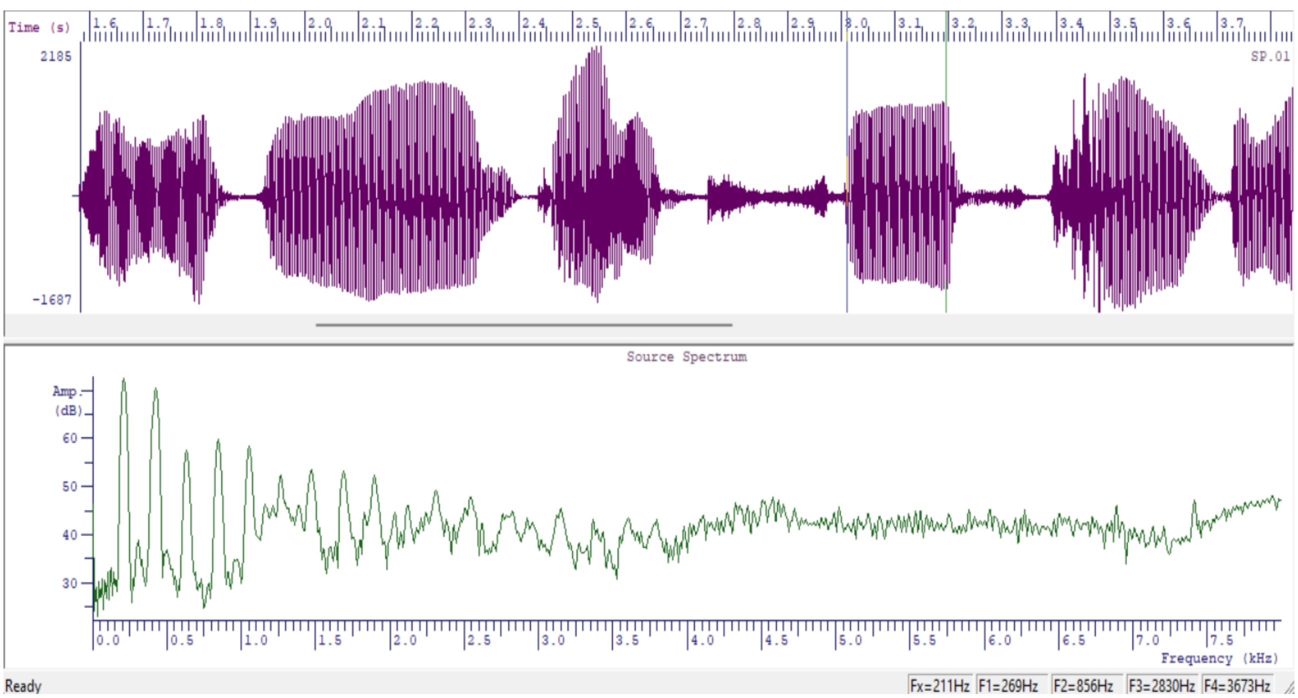
Had



dark

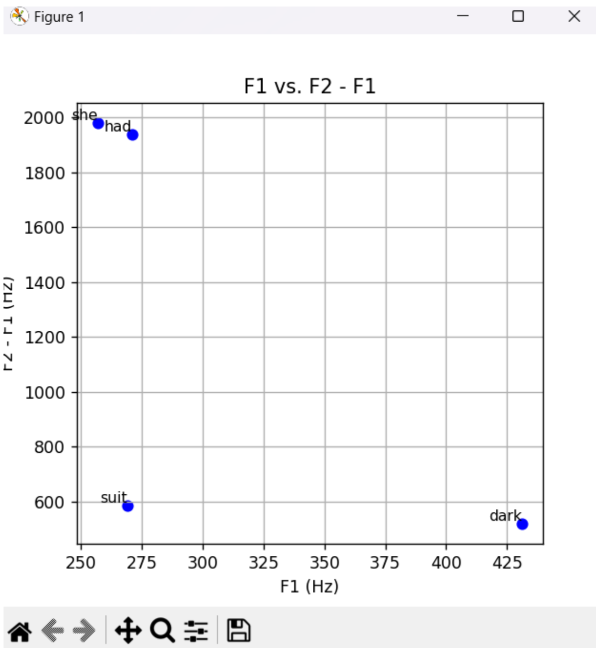

Time (s)
2185
-1687
SP.01
Source Spectrum
Amp. (dB)
Frequency (kHz)
Ready
Fx=216Hz F1=431Hz F2=951Hz F3=2816Hz F4=3571Hz

Suit


Time (s)
2185
-1687
SP.01
Source Spectrum
Amp. (dB)
Frequency (kHz)
Ready
Fx=211Hz F1=269Hz F2=856Hz F3=2830Hz F4=3673Hz

Recording2.wav formants table:

Formants	F1 (HZ)	F2 (HZ)	F3 (HZ)
she	257	2236	2670
had	271	2207	2821
dark	431	951	2816
suit	269	856	2830



From this part after measuring formants frequencies for vowels in 3 files different speakers, we find that the formants frequency vary for different vowels and for the same vowel across different speakers.

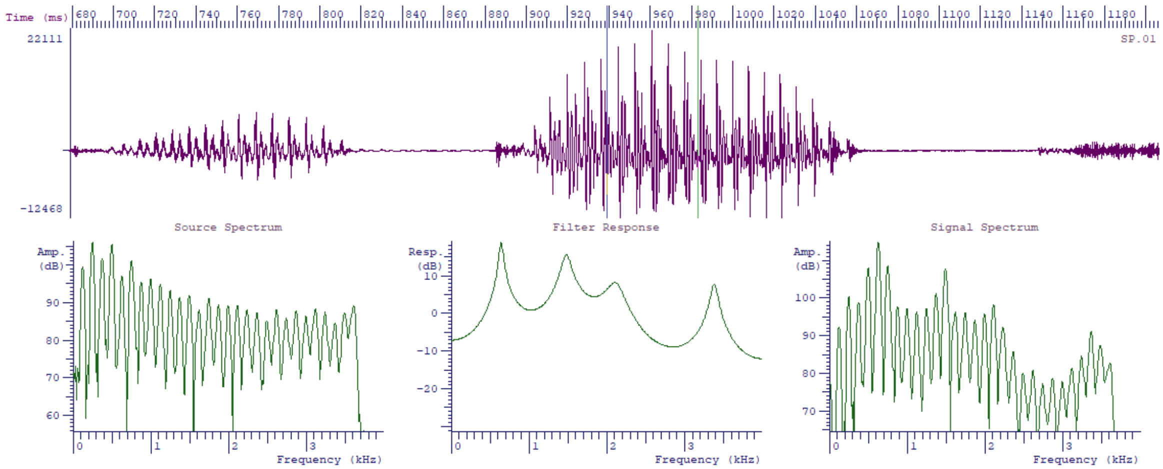
For voiced speech the magnitude of the lower formants frequencies is larger than magnitude of higher formants frequencies .

For unvoiced speech the magnitude of the higher formants frequencies is larger than magnitude of lower formants frequencies .

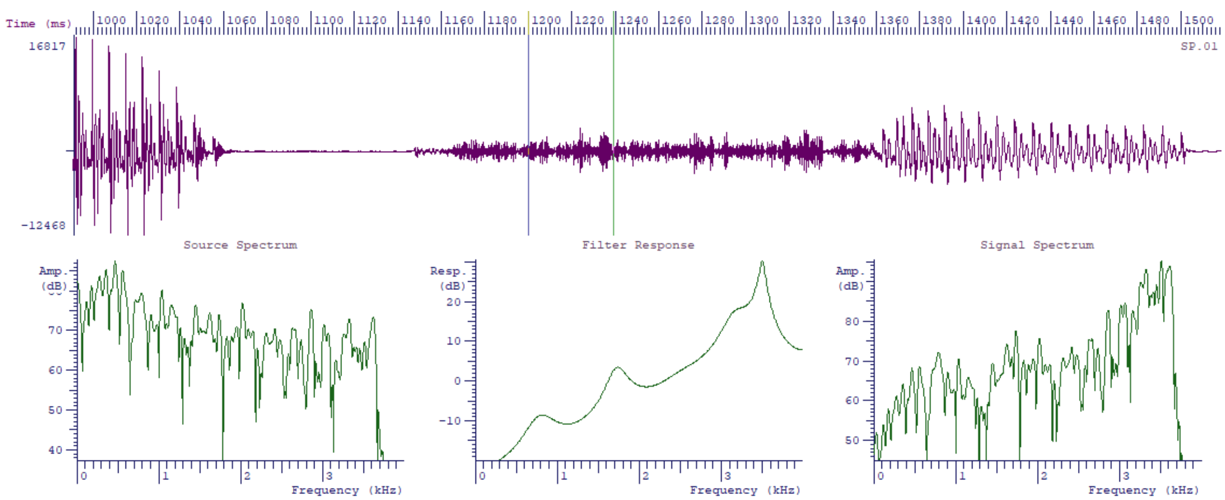
b) Source and Filter analysis

In this part I perform the source filter analysis for phoneme /s/ in ‘suit’ and /A/ in ‘dark’ .

Dark ‘a’ (vowel periodic)



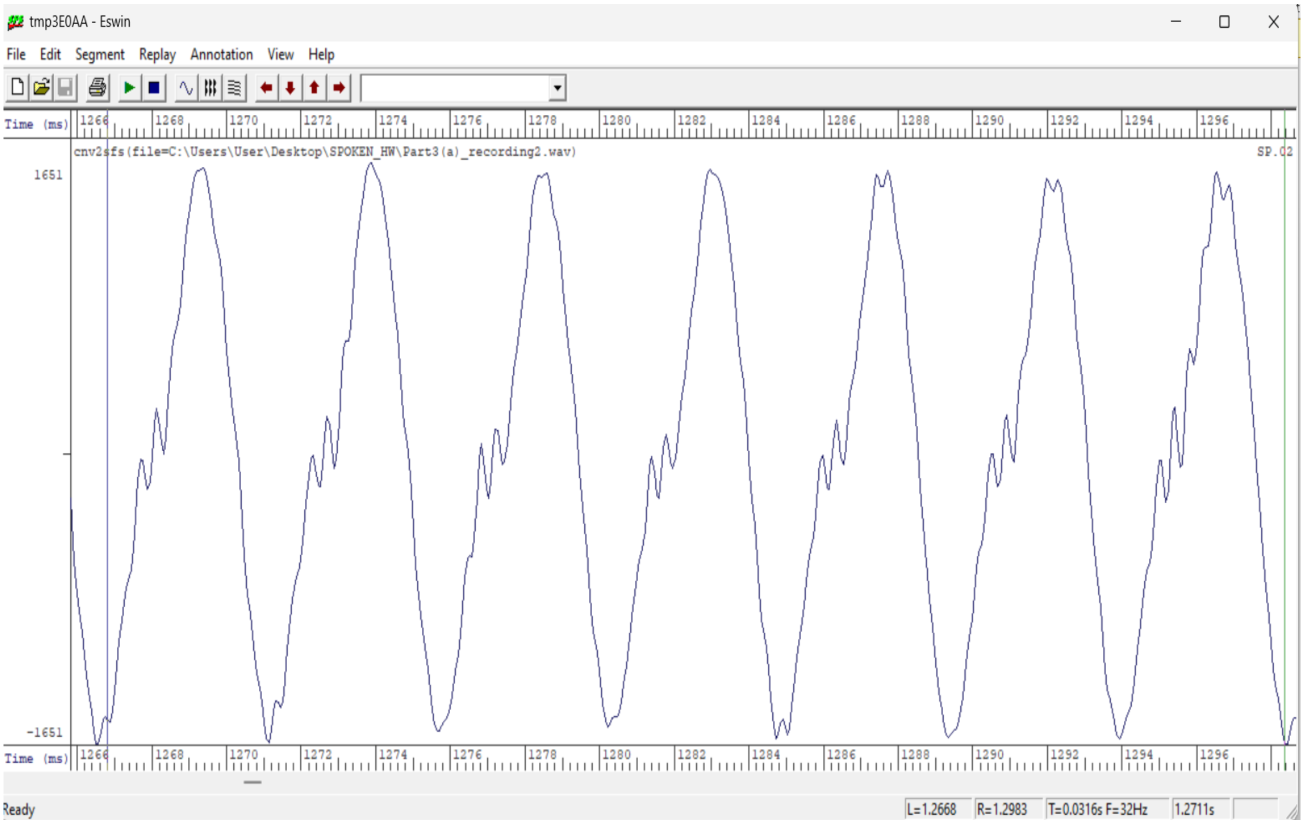
Suit ‘s’ (noise)



There is difference between filter response for each phoneme.

c) Fundamental Frequency

RECORDING1



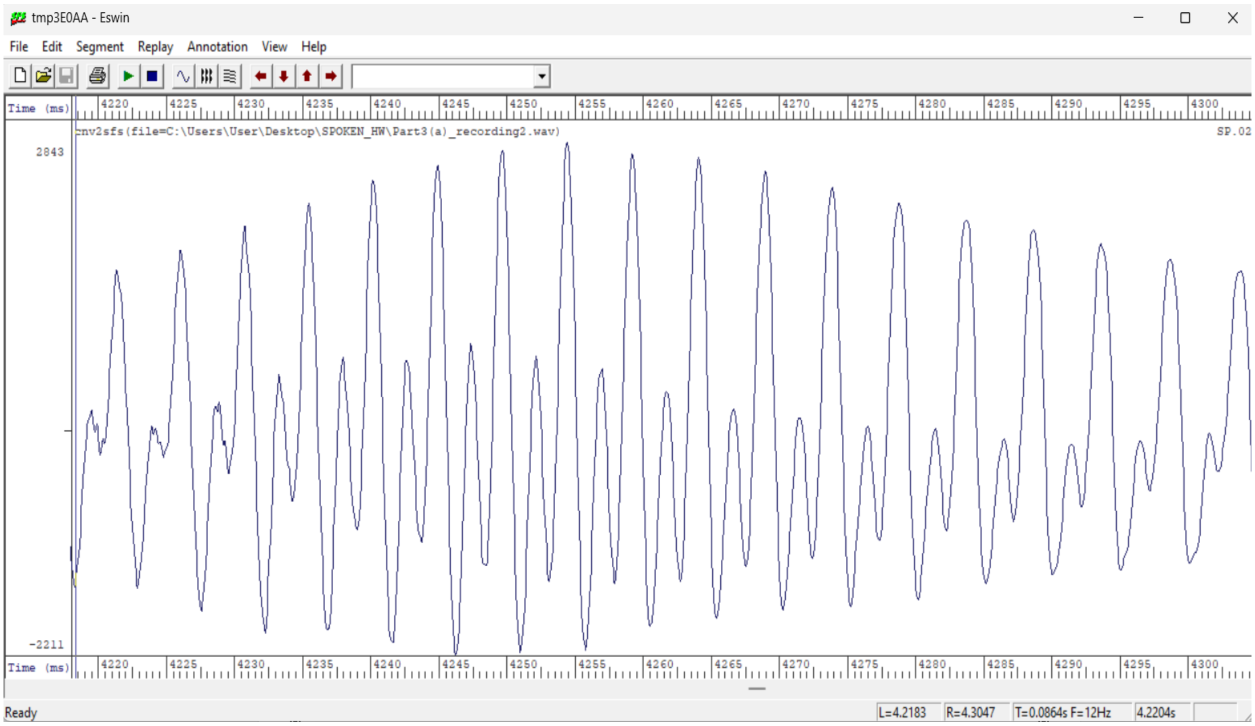
Fundamental period= duration/ number of periods

= 0.0316/7=4.514 * 10^-3

Fundamental frequency=1/ fundamental period= 1/(4.514 * 10^-3)

= 221.5 HZ

RECORDING2



Fundamental period= duration/ number of periods

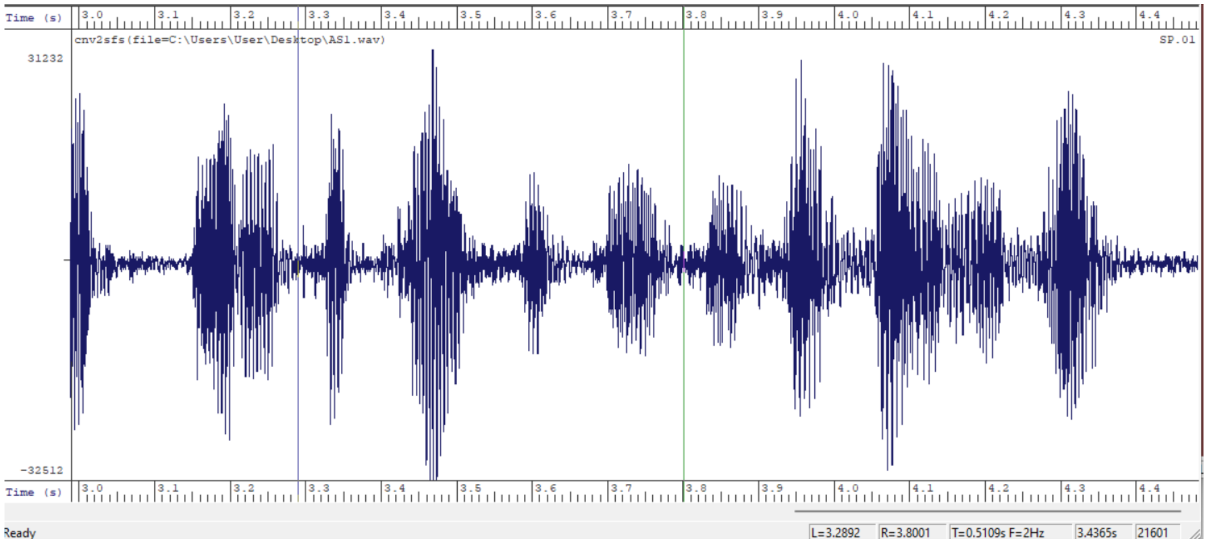
= 0.0864/18=4.8 * 10^-3

Fundamental frequency=1/ fundamental period= 1/(4.8 * 10^-3)

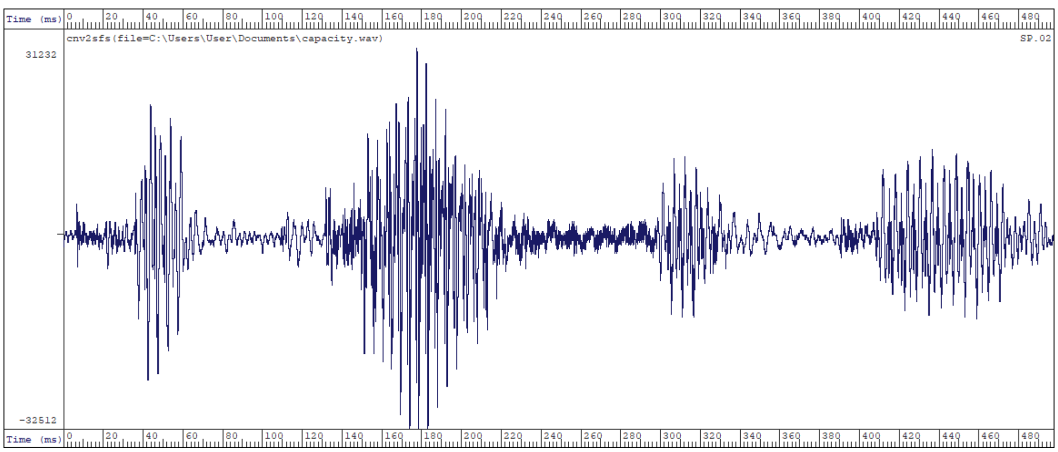
= 208.33 HZ

Part 4 – Speech analysis

All waveform in AS1.wav



'Capacity' waveform:

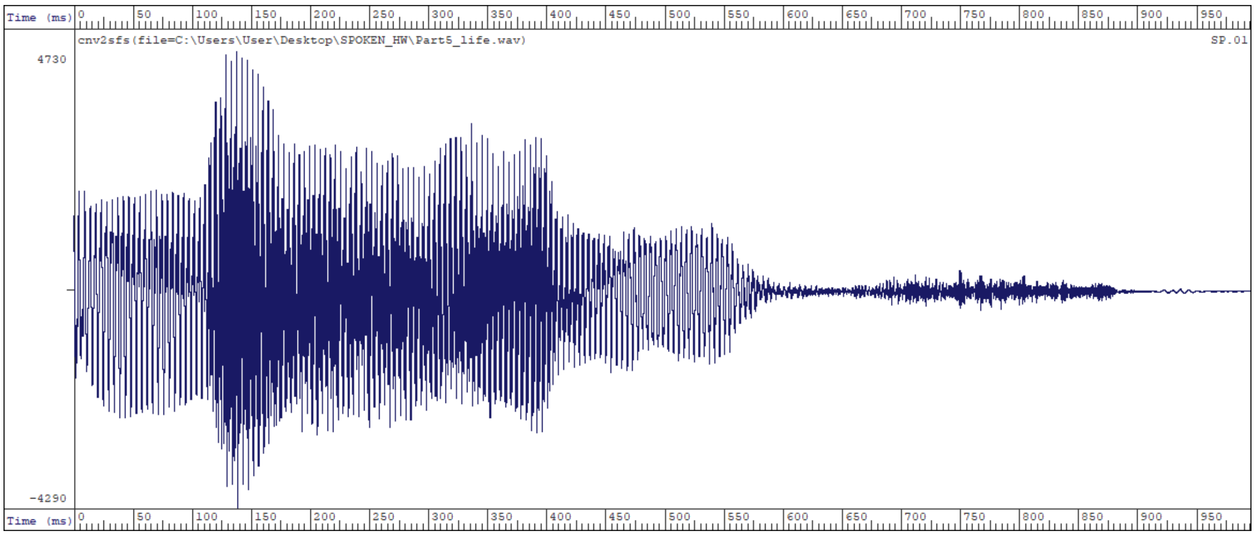


Phoneme		Start time (ms)	End time (ms)
/k/	C	0	17
/@/	a	18	62
/p/	p	63	130
/{/	a	131	222
/s/	c	223	298
/I/	i	299	232
/t/	t	233	292
/i/	y	293	508

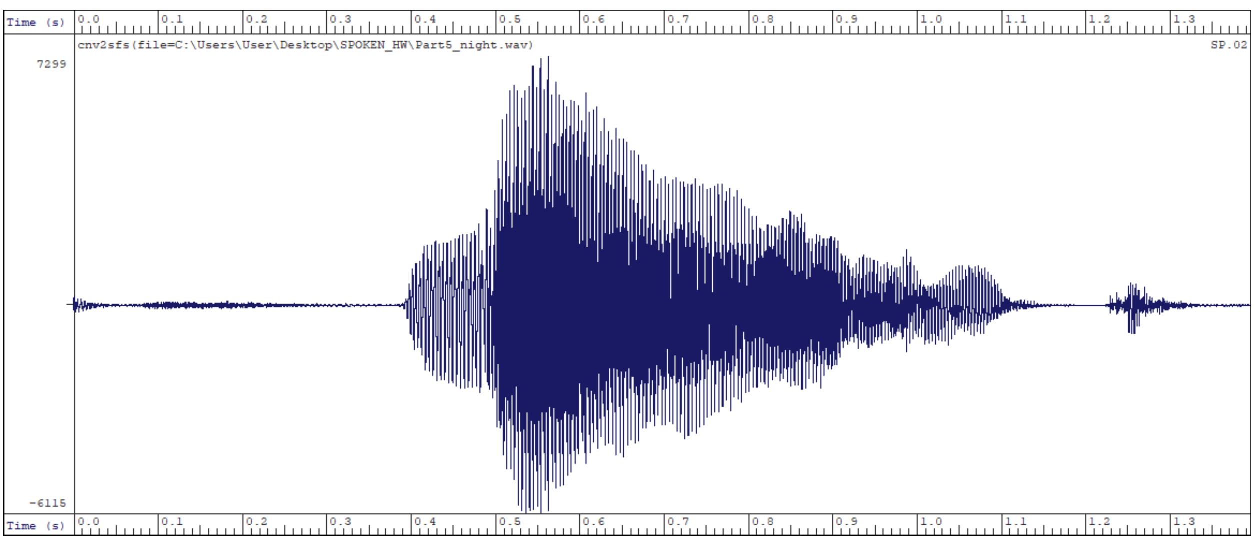
Part 5 – Sub-word level concatenation

In this part I record ‘life’ and night’ words to make a new word ‘light’ by make concatenation between ‘li’ from ‘life’ and ‘ght’ from ‘night’ and I used python code to make concatenation between the two sub word level .wav files.

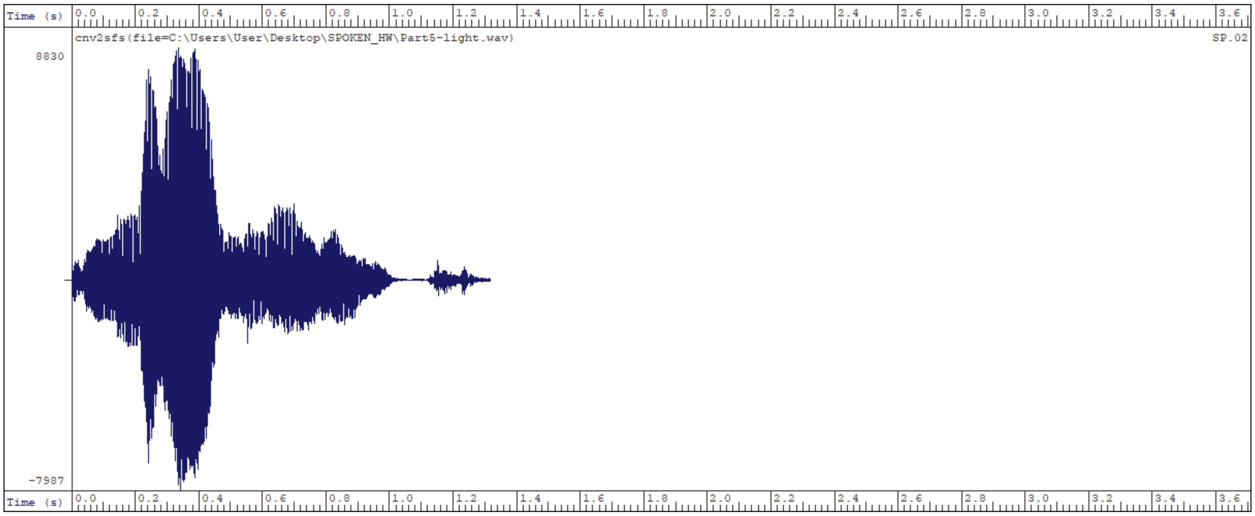
Life waveform>



Night waveform>



Result of concatenation : light



Python code for concatenation:

```
from pydub import AudioSegment
from tkinter import Tk, filedialog

1 usage
def get_file_paths():
    Tk().withdraw() # Hide the Tkinter root window
    print("Please select the first .wav file:")
    first_file_path = filedialog.askopenfilename(filetypes=[("WAV files", "*.wav")])
    print("Please select the second .wav file:")
    second_file_path = filedialog.askopenfilename(filetypes=[("WAV files", "*.wav")])
    return first_file_path, second_file_path
# Function to concatenate two audio files
1 usage
def concatenate_audio(first_file_path, second_file_path):
    first_audio = AudioSegment.from_wav(first_file_path)
    second_audio = AudioSegment.from_wav(second_file_path)
    concatenated_audio = first_audio + second_audio
    return concatenated_audio
1 usage
def save_audio(output_audio, output_file_path):
    output_audio.export(output_file_path, format="wav")
    print(f"Concatenated audio saved as '{output_file_path}'.")

if __name__ == "__main__":
    first_file_path, second_file_path = get_file_paths()
    output_file_path = filedialog.asksaveasfilename(defaultextension=".wav",   filetypes=[("WAV files", "*.wav")])
    concatenated_audio = concatenate_audio(first_file_path, second_file_path)
    save_audio(concatenated_audio, output_file_path)
```

We produce a high quality speech from using two sub word level concatenation process.

Part 6 – Phone-level concatenation

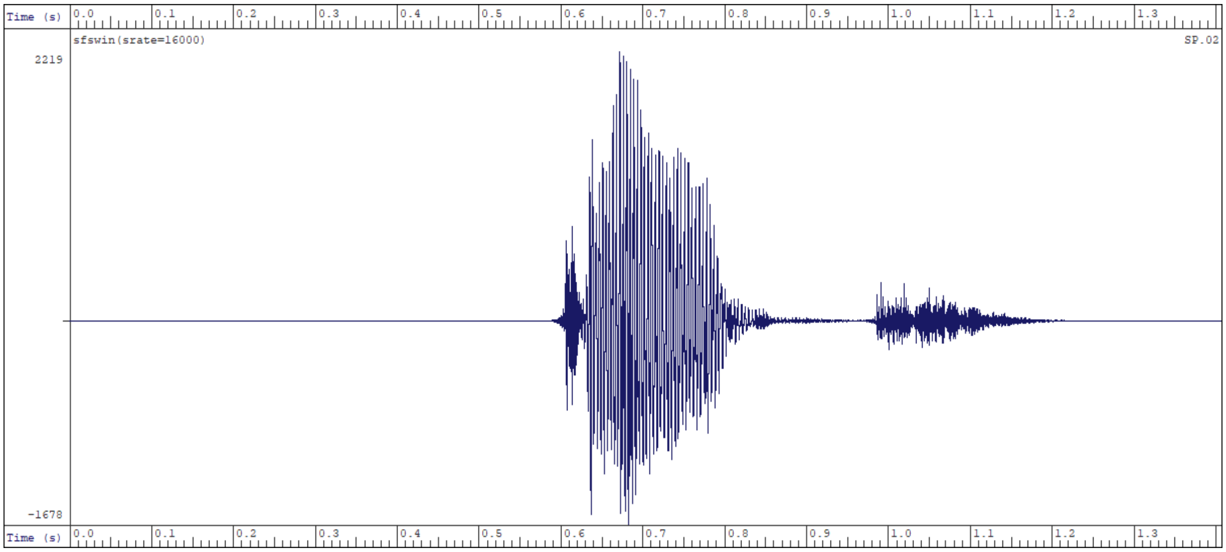
In this part I make a different word from capacity word phonemes {c,a,p,a,c,i,t,y}

And the new word is 'cat' by making concatenation between 'c' 'a' 't' parts.

Python code to make concatenation between phonemes:

```
from pydub import AudioSegment
from tkinter import Tk, filedialog
1 usage
def get_file_paths():
    Tk().withdraw() # Hide the Tkinter root window
    print("Please select the first .wav file:")
    first_file_path = filedialog.askopenfilename(filetypes=[("WAV files", "*.wav")])
    print("Please select the second .wav file:")
    second_file_path = filedialog.askopenfilename(filetypes=[("WAV files", "*.wav")])
    print("Please select the third .wav file:")
    third_file_path = filedialog.askopenfilename(filetypes=[("WAV files", "*.wav")])
    return first_file_path, second_file_path, third_file_path
# Function to concatenate two audio files
1 usage
def concatenate_audio(first_file_path, second_file_path):
    first_audio = AudioSegment.from_wav(first_file_path)
    second_audio = AudioSegment.from_wav(second_file_path)
    concatenated_audio = first_audio + second_audio
    return concatenated_audio
1 usage
def save_audio(output_audio, output_file_path):
    output_audio.export(output_file_path, format="wav")
    print(f"Concatenated audio saved as '{output_file_path}'.")
if __name__ == "__main__":
    first_file_path, second_file_path, third_file_path = get_file_paths()
    output_file_path = filedialog.asksaveasfilename(defaultextension=".wav",   filetypes=[("WAV files", "*.wav")])
    concatenated_audio = concatenate_audio(first_file_path, second_file_path, third_file_path)
    save_audio(concatenated_audio, output_file_path)
```

C+a+t: cat waveform



We produce a low quality speech from using phone level concatenation process. We can improve quality by select high quality speech units , and implement algorithms that select a most appropriate speech units for concatenations.

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

In this exercise, the source-filter model was employed to generate three vowel sounds (/A:/, /u:/, and /ə/) using a pulse-train source signal passed through vocal-tract filters.

Generation Procedure:

A periodic pulse-train source signal at 110 Hz frequency was generated to mimic the vibrating vocal cords during voiced sounds.

Two band-pass filters were applied in parallel to the source signal, each corresponding to one of the first two formant frequencies of the respective vowel.

The first two formant frequencies used were: /A:/ (F1=700Hz, F2=1100Hz), /u:/ (F1=300Hz, F2=800Hz), and /ə/ (F1=500Hz, F2=1500Hz).

The filtered signals were exported as separate wav files.

Signal Combination:

The filtered signals were then loaded into Matlab using the audioread function.

Each signal was summed element-wise to obtain the final speech signal representing the vowel sound.

The final signal was stored in the wav file format using the audiowrite function.

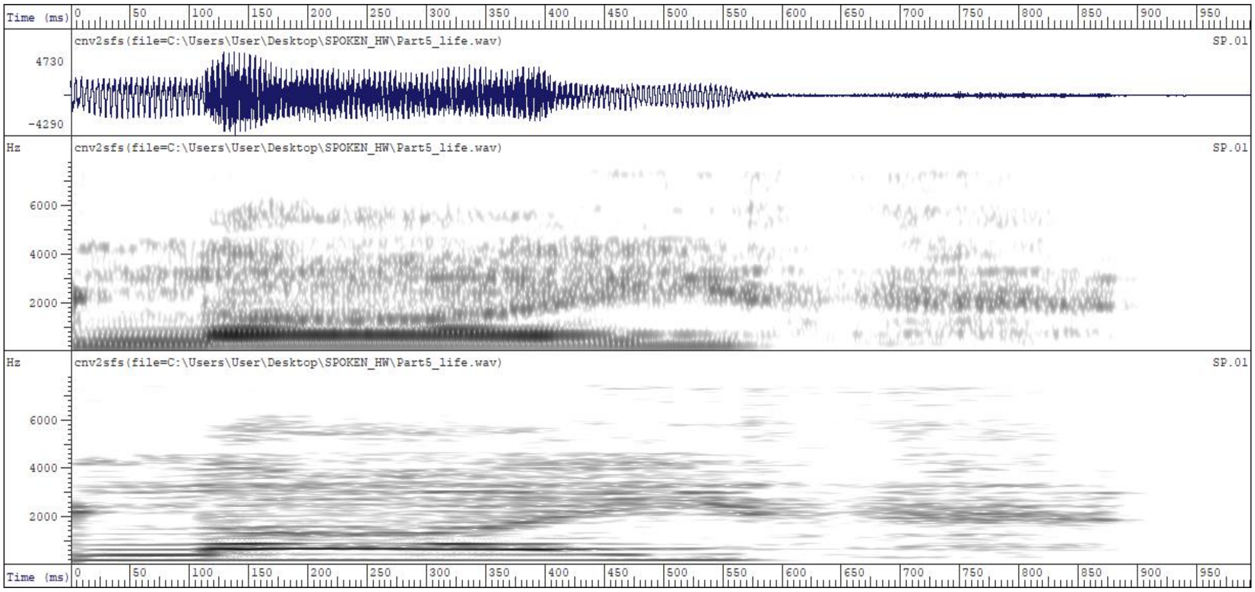
Output:

The generated wav files for the three vowels were submitted as the deliverable.

In conclusion, This exercise demonstrates the concept of vowel generation using the source-filter model, where the source signal (vibrating vocal cords) is filtered through the vocal tract (formant frequencies) to produce distinct vowel sounds. By adjusting the formant frequencies, different vowels can be synthesized, showcasing the importance of formant frequencies in speech production and perception. Additionally, the option to use white noise as the source signal provides insight into the generation of whispered vowels, where vocal fold vibration is absent. Overall, this exercise enhances understanding of the acoustic characteristics of vowels and their synthesis using the source-filter model.

Part 8 – Formant synthesis in SFS

‘life’ waveform and spectrogram:

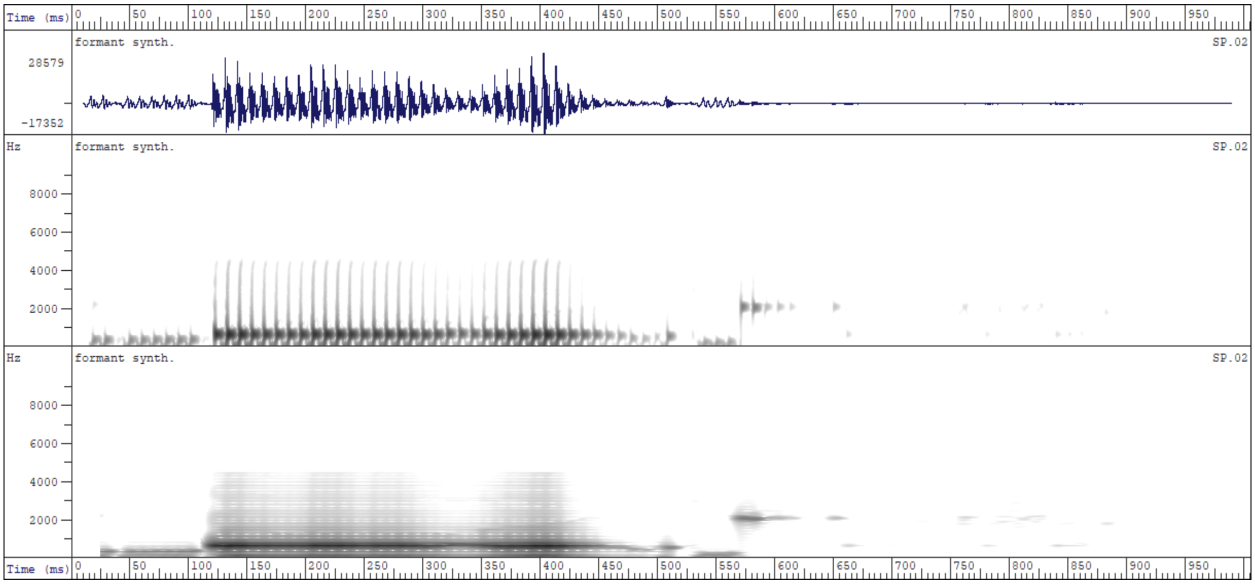


Formants.txt of original signal ‘life’ :

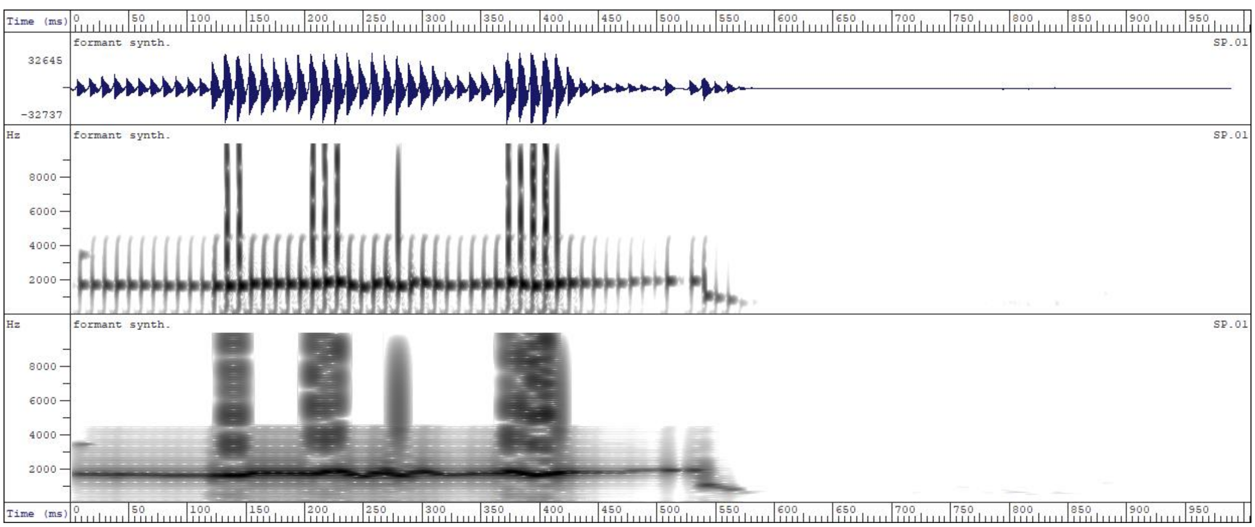
```
! item=12.01 file=C:\Users\User\AppData\Local\Temp\tmpA79A9.sfs
! posn size v   gain;   F1   B1 A1;   F2   B2 A2;   F3   B3 A3;   F4   B4 A4;   F5   B5 A5;
!
  10.0 20.0 1   85.0;  294   62 82; 2224   24 67; 4239   27 49; 5645  321 24; 7220  242 25;
  20.0 20.0 1   85.8;  293   77 83; 2266   98 47; 3171  140 42; 4321   57 44; 6766  317 23;
  30.0 20.0 1   86.8;  292   59 85;  685  139 65; 2232   76 48; 3131  148 42; 4286   16 51;
  40.0 20.0 1   86.9;  393   80 82; 2462  180 41; 3295   91 43; 4209   72 41; 5282  288 25;
  50.0 20.0 1   86.7;  391   74 82; 2530  253 39; 3146   67 48; 4230   49 42; 5304  440 21;
  60.0 20.0 1   86.9;  386   82 82; 3084  103 48; 4498   58 39; 6129  262 21; 7349  255 18;
  70.0 20.0 1   86.9;  384   76 83; 2803  298 42; 3280  137 44; 4390   29 44; 7298  109 23;
  80.0 20.0 1   86.9;  383   77 83; 2453   89 48; 3237   68 47; 4377   76 41; 7168  281 21;
  90.0 20.0 1   86.4;  418   78 82; 2477  173 45; 3214   78 49; 4326  132 41; 7074  279 21;
 100.0 20.0 1   86.2;  378   78 82;  761  242 64; 2508  246 43; 3161   76 51; 4315  246 34;
 110.0 20.0 1   86.4;  481  195 79;  677  190 75; 1929  202 50; 3171   77 55; 4190  255 40;
 120.0 20.0 1   89.4;  692   28 91; 3132   92 51; 4220  115 47; 5513  137 35; 7295  229 23;
 130.0 20.0 1   91.6;  687   19 95;  979  331 70; 3238  214 47; 4243  133 47; 5520  154 36;
 140.0 20.0 1   92.2;  680   22 94;  972  176 74; 3069  235 46; 4064   55 53; 5699  265 38;
 150.0 20.0 1   91.4;  691   28 92; 4101   72 51; 5337  111 41; 5833   58 42; 7260   89 24;
 160.0 20.0 1   90.8;  693   27 92; 3368  164 50; 4229  111 44; 5539  162 37; 5969  181 34;
 170.0 20.0 1   89.8;  699   22 91; 1363  143 65; 2821  188 48; 3838  182 48; 6085  145 36;
 180.0 20.0 1   88.4;  694   13 92; 1341   76 66; 2640  185 49; 4147  170 41; 5529  185 39;
 190.0 20.0 1   88.0;  690   12 92; 1348   94 64; 3270   80 50; 5501  132 37; 6043  161 33;
 200.0 20.0 1   88.1;  686    8 94; 1425  105 61; 2588  247 44; 3370   74 50; 5529   79 40;
 210.0 20.0 1   88.3;  684    7 94; 1392   79 63; 2705  170 48; 3384  138 49; 5449   65 41;
 220.0 20.0 1   88.0;  689    8 94; 1448  205 56; 2565  182 48; 3338  147 46; 5515   51 43;
 230.0 20.0 1   87.6;  681   11 91; 1481  204 55; 2626  259 45; 3384  116 49; 5501  378 32;
 240.0 20.0 1   87.2;  686   15 90; 1380   91 63; 2758  172 49; 4099  228 38; 5513  105 39;
 250.0 20.0 1   87.3;  688    7 94; 1457  170 55; 2677  193 47; 3339  176 45; 5510  149 38;
 260.0 20.0 1   87.4;  679   11 91; 1379   81 63; 2723  197 47; 3361  137 49; 5410  267 35;
 270.0 20.0 1   87.3;  682    8 93; 1374   64 64; 2824  175 45; 3720  176 44; 5694  128 35;
 280.0 20.0 1   87.3;  676   13 90; 1381   80 63; 2709  170 47; 3620  153 49; 5467  213 37;
 290.0 20.0 1   86.7;  673   14 90; 1360   72 64; 3349  208 46; 3939  172 43; 5710  208 32;
 300.0 20.0 1   86.7;  678   22 88; 1357   81 66; 3217  141 50; 4352  198 37; 5509   86 40;
 310.0 20.0 1   86.8;  713   42 85; 1335  172 65; 3026   89 56; 3854  245 41; 5344  234 32;
 320.0 20.0 1   87.1;  730   76 84;  945  280 73; 2205  348 49; 3049  108 53; 4018  195 40;
 330.0 20.0 1   87.2;  736   76 84; 1730  279 55; 3025  128 54; 5604  230 29; 7368  298 19;
 340.0 20.0 1   87.2;  700   50 86; 3114  196 51; 3888  272 41; 5646  149 33; 7317  313 22;
 350.0 20.0 1   87.3;  693   21 89; 1567   96 65; 3145  144 52; 4269   97 41; 5614   84 33;
 360.0 20.0 1   87.2;  673   12 91; 1589  113 60; 4141  201 39; 5578   80 34; 7290  119 20;
 370.0 20.0 1   87.4;  667    7 94; 1762   79 62; 3101  169 52; 4290  130 44; 5570   56 39;
```

Ln 8, Col 20 | 9,272 characters

‘synthesized original’ waveform and spectrogram:



‘synthesized modified’ waveform and spectrogram:



To change the speaker's voice in a formant-based speech synthesizer, we use speech analysis tools to extract formant data from the reference recordings. Formants represent the resonant frequencies of the vocal tract and play a crucial role in determining the characteristics of a speaker's voice.