NATIONAL INSTITUTE OF TECHNOLOGY
SIKKIM

SPEECH PROCESSING LABORATORY

EC18240

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

SAMEER CHETTRI — B190063EC

INDEX

Experiment 1

Aim: Analysis of Speech Signal using wavesurfer. Plot the waveform, spectrogram, speech analysis with formant frequencies and spectrum section plots (FFT and LPC)

Waveform: A waveform is a graphical representation of the intensity of a sound wave as it varies with time. It depicts the general pattern of the sound and can be helpful in recognizing features like gaps, quiet moments, and shifts in tone.

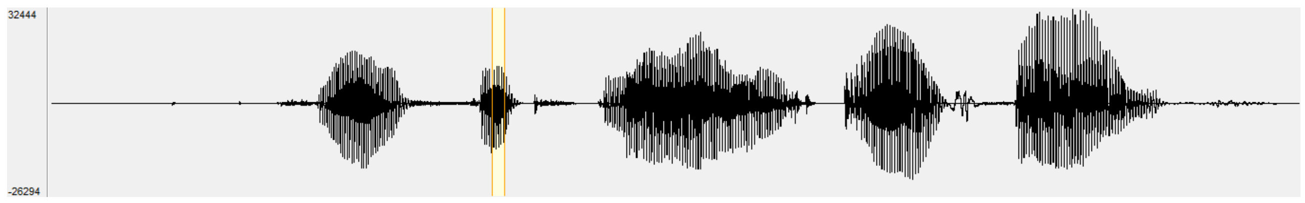
Spectrogram: A spectrogram is a visual representation of the frequency content of a sound signal as it varies over time. It is produced by taking a short-time Fourier transform of the signal and displaying the resulting power spectrum as a function of both frequency and time. The spectrogram helps to recognize certain features of speech such as formants, which are the frequencies that are resonated by the vocal tract and are responsible for the distinct sound of each vowel.

Formant frequencies: The resonant frequencies of the vocal tract that define the sound quality of vowel sounds are known as formants. These formants appear as dark, horizontal bands in a spectrogram. By studying the formant frequencies, it is possible to distinguish which vowel sounds are present in the speech signal.

Spectrum section plots: A plot of the frequency content of a speech signal at a specific moment is referred to as a spectrum section. It is produced by performing a Fourier transform on a small portion of the signal and then displaying the resulting power spectrum. This plot is useful in detecting specific harmonics within the speech signal, which are connected to the fundamental frequency (or pitch) of the speaker's voice.

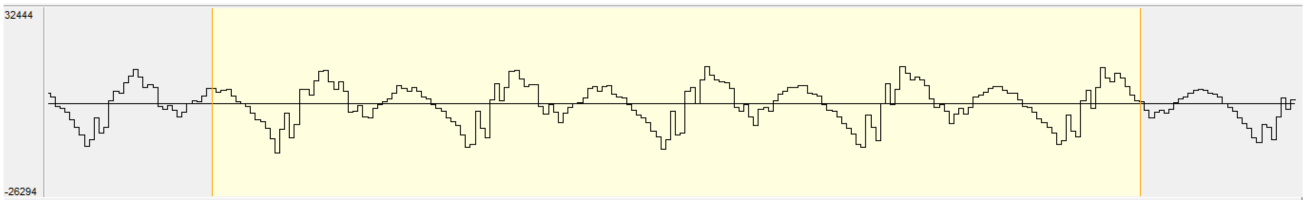
LPC: Linear Predictive Coding (LPC) is a method used to represent the speech signal as a linear mix of past samples with varying weights. It is a valuable tool in studying the spectral envelope of the speech signal, which enables the identification of features such as formants and other spectral peaks. In addition to analysis, LPC is also utilized in speech synthesis and compression.

Speech data is 369o5 and selected area is under analysis.

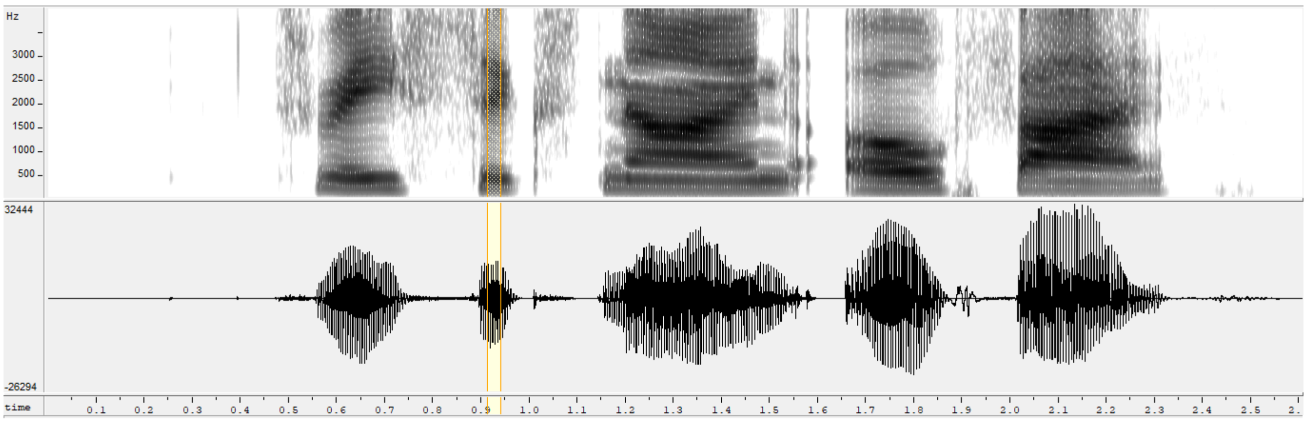


Applied Configurations:

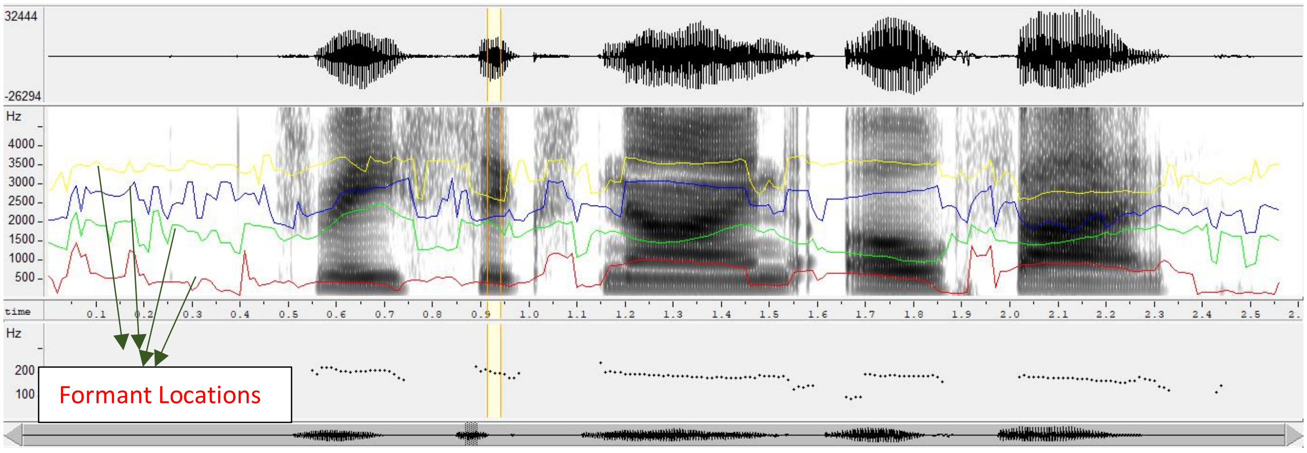
1. Waveform:



2. Spectrogram:

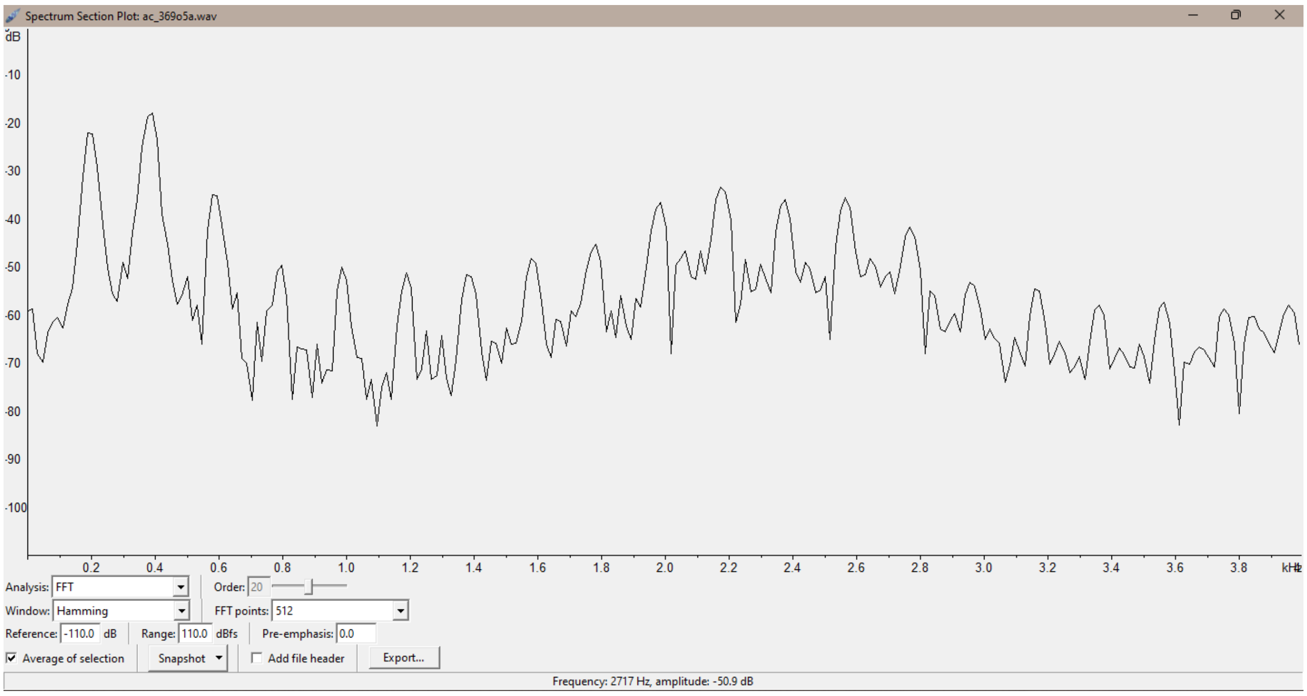


3. Speech Analysis:



4. Spectrum Section Plot: Fast Fourier Transform (FFT)

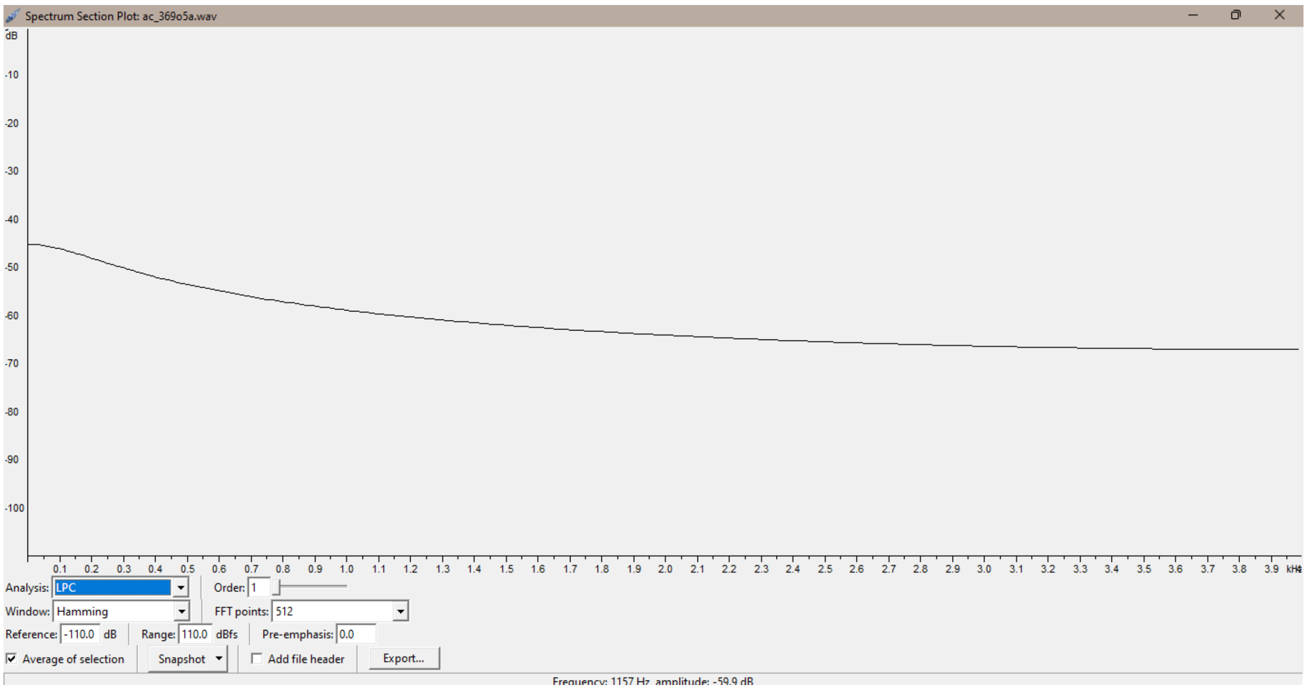
   Using Hamming window with 512 points



Analysis: FFT
Order: 20
Window: Hamming
FFT points: 512
Reference: -110.0 dB
Range: 110.0 dBfs
Pre-emphasis: 0.0
☑ Average of selection
☐ Add file header

Frequency: 2717 Hz, amplitude: -50.9 dB

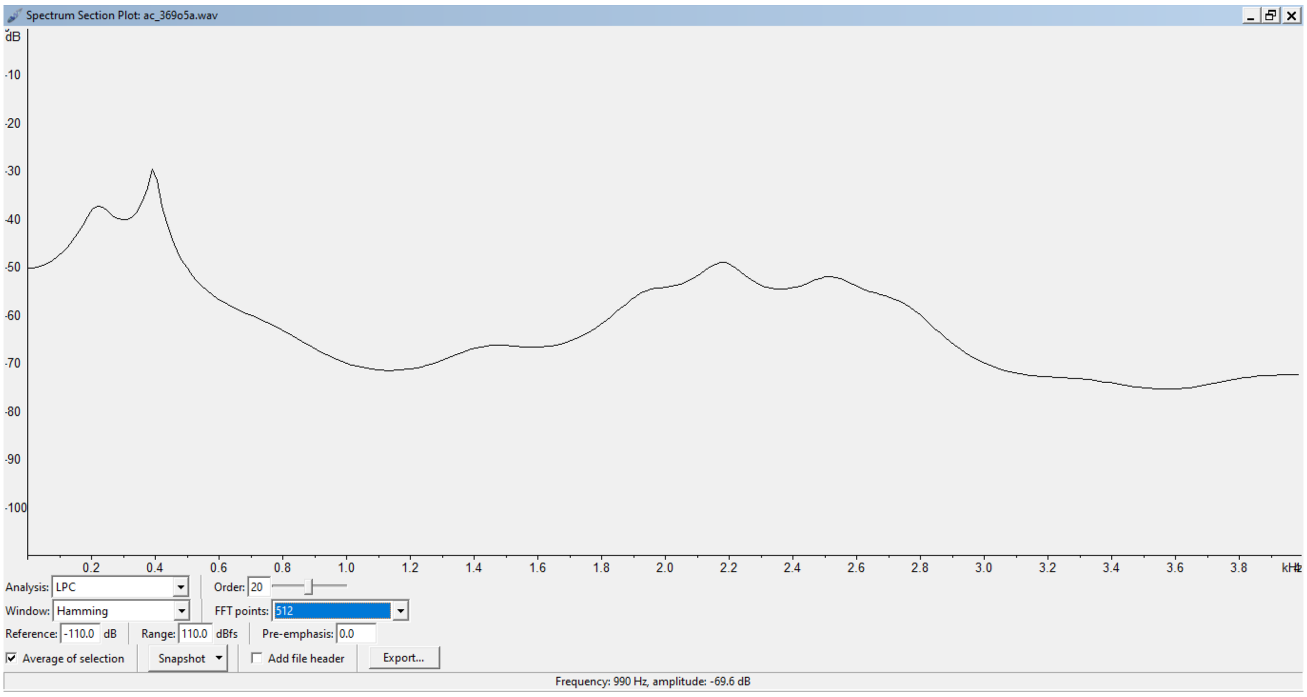
5. Spectrum Section Plot: Linear Productive Coding (LPC) using Hamming window with 512 points.
   1st order



Analysis: LPC
Order: 1
Window: Hamming
FFT points: 512
Reference: -110.0 dB
Range: 110.0 dBfs
Pre-emphasis: 0.0
☑ Average of selection
☐ Add file header

Frequency: 1157 Hz, amplitude: -59.9 dB

Order 20:

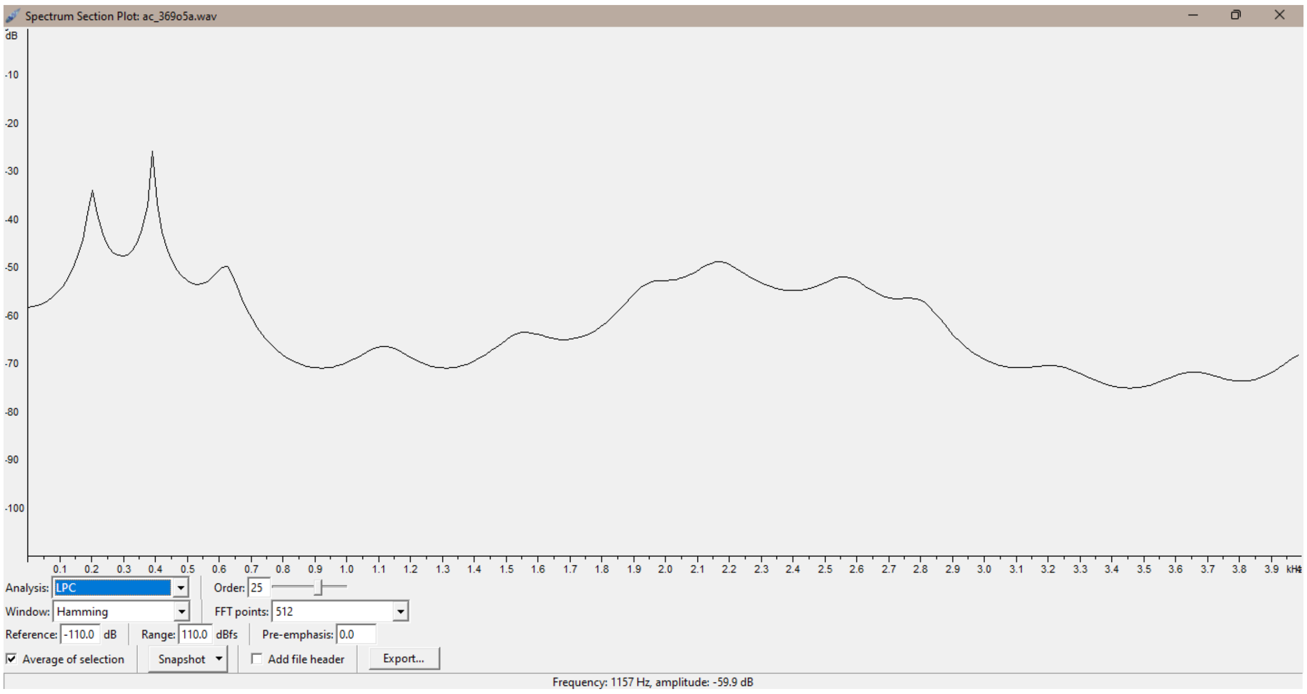


Spectrum Section Plot: ac_369o5a.wav

Analysis: LPC
Order: 20
Window: Hamming
FFT points: 512
Reference: -110.0 dB
Range: 110.0 dBfs
Pre-emphasis: 0.0
☑ Average of selection
☐ Add file header

Frequency: 990 Hz, amplitude: -69.6 dB

Order 25:



Spectrum Section Plot: ac_369o5a.wav

Analysis: LPC
Order: 25
Window: Hamming
FFT points: 512
Reference: -110.0 dB
Range: 110.0 dBfs
Pre-emphasis: 0.0
☑ Average of selection
☐ Add file header

Frequency: 1157 Hz, amplitude: -59.9 dB

Conclusion: On the basis of the above simulations, it can be concluded that with the increase in the order of the system, the formant locations are clearly visible.

Experiment 2

Aim: Write a MATLAB code to read the wave file, take a window size of 25ms, Calculate short term energy and Zero Crossing Rate.

Theory: A spoken sentence is a sequence of phonemes. Speech signals are thus time-variant in character. To extract information from a signal, we must therefore split the signal into sufficiently short segments, such that, heuristically speaking, each segment contains the information that is useful for the observation. **Windowing** is a technique used in signal processing to reduce the leakage effect in the Fourier Transform of a finite length signal. The idea is to multiply the time-domain signal with a window function before taking the Fourier Transform. Windowing can improve the spectral resolution of the signal by reducing the effect of spectral leakage.

Short term Energy: It is used to provide the information about the overall magnitude or amplitude of the speech signal over the short time intervals. Basically, it calculates the energy of a window.

$$\textbf{Energy, } E = \sum_{m=-\infty}^{\infty} x^2(m)$$

$$\textbf{Short-Time Energy, } E = \sum_{m=n-N+1}^{n} x^2(m)$$

Order: $E_{(\text{voiced})} > E_{(\text{unvoiced})} > E_{(\text{silence})}$

Zero Crossing Rate: The zero-crossing rate refers to the frequency at which a signal passes through the zero-axis. This is a common feature used in speech and music analysis, as it is correlated with the pitch and rhythm of the signal. A higher zero crossing rate typically implies a higher frequency or faster rhythm. The zero-crossing rate is computed by calculating the number of times the signal changes polarity within a small-time window, usually between 10 to 30 milliseconds.

The measurement is essentially a way to determine how often two adjacent values in A change sign from positive to negative or vice versa. This is determined by using the function sgn(.), which returns a value of 1 if one input is positive and the other is negative, and 0 if they are both positive or both negative.

$$Z_n = \sum_{m=-\infty}^{\infty} \left|\operatorname{sgn}[x(m)] - \operatorname{sgn}[x(m-1)]\right| w(n-m)$$

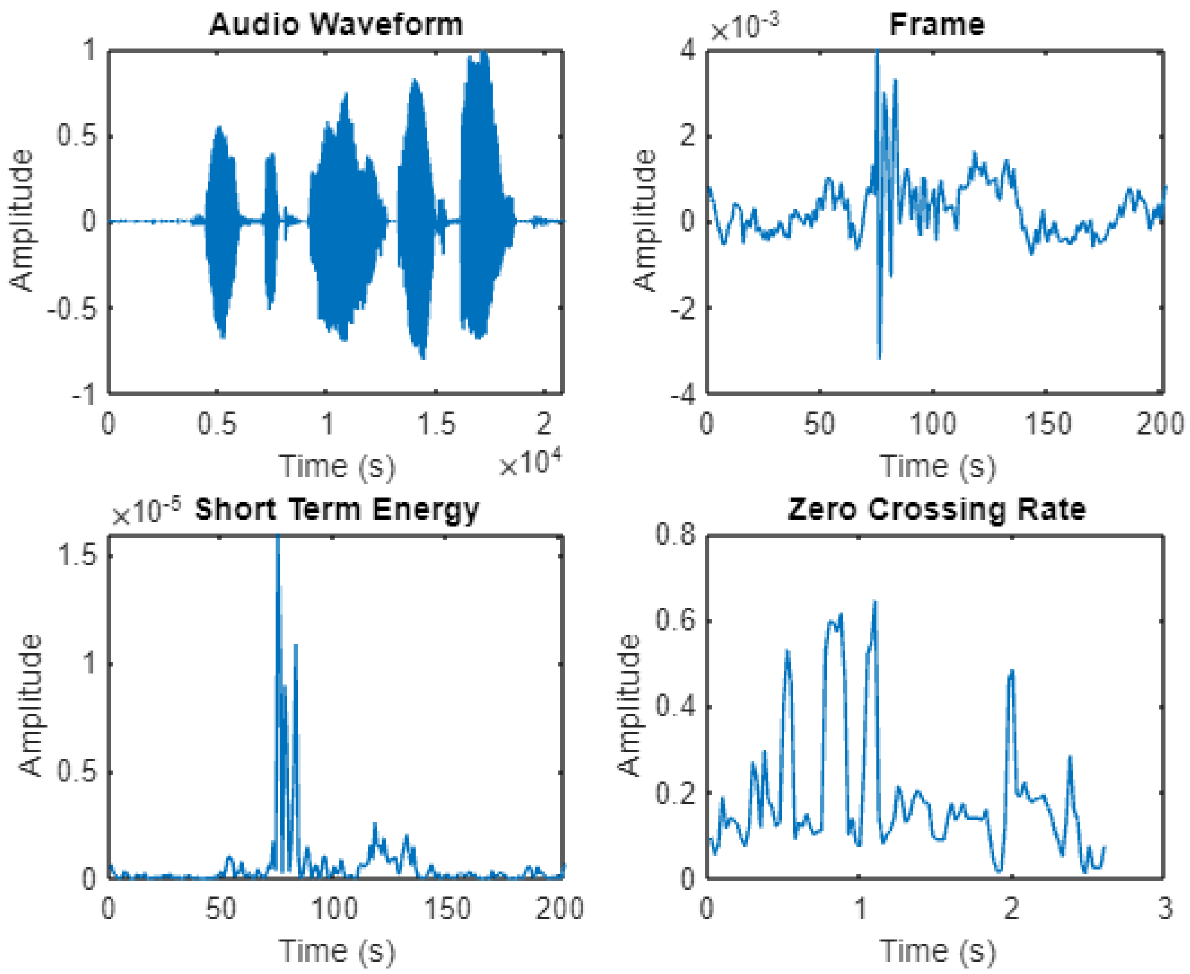
Order: $Z_{(\text{unvoiced})} > Z_{(\text{silence})} > Z_{(\text{voiced})}$

Code:

```
close all;
clear all;

% read wavefile
[Y, fs] = audioread('/MATLAB Drive/ac_369o5a.wav');
subplot(2,2,1);
plot(Y);
xlabel('Time (s)');
ylabel('Amplitude');
title('Audio Waveform');

%taking a window of 25ms
fd = 0.025;
f_size = fd*fs;
fr= Y(2500:2501+f_size);
subplot(2,2,2)
plot(fr);
xlabel('Time (s)');
ylabel('Amplitude');
title('Frame');

%Energy plot
%short term energy
[r,c]=size(fr);
for i=1:r
    ste(i)=sum(fr(i,:).^2);
end
subplot(2,2,3)
plot(ste)
xlabel('Time (s)');
ylabel('Amplitude');
title('Short Term Energy')

%Zero Crossing Rate

% Calculate the window size in samples
win_samples = round(fd*fs);

% Calculate the number of windows
num_win = floor(length(Y)/win_samples);

% Initialize the Zero Crossing Rate array
zcr = zeros(num_win,1);

% Iterate over the windows
for i = 1:num_win
    % Calculate the start and end indices for the current window
    start_index = (i-1)*win_samples + 1;
    end_index = min(start_index+win_samples-1, length(Y));

    % Extract the current window
    x = Y(start_index:end_index);

    % Calculate the Zero Crossing Rate for the current window
    zcr(i) = sum(abs(diff(sign(x))))/(2*win_samples);
end
```

```
% Plot the Zero Crossing Rate
subplot(2,2,4)
plot((1:num_win)*fd, zcr);
xlabel('Time (s)');
ylabel('Amplitude');
title('Zero Crossing Rate');
```

Output:



Conclusion:

In the above simulation, we have calculated the short term energy and Zero-crossing rate concluding that STE is greater in voiced region compared to unvoiced and silent regions.

Experiment 3

Aim: Write the MATLAB code for reading a wave file and then calculating its Short-term Fourier Transform (STFT).

Theory: A short-time Fourier transform (STFT) is a signal processing method used to examine the frequency content of a signal at different points in time. This is achieved by dividing the signal into short, overlapping sections and applying the Fourier Transform to each section. The resulting spectrogram displays how the spectral content of the signal changes over time.

STFT is often used in speech processing to extract spectral features from the speech signal, such as formant frequencies and spectral energy distributions.

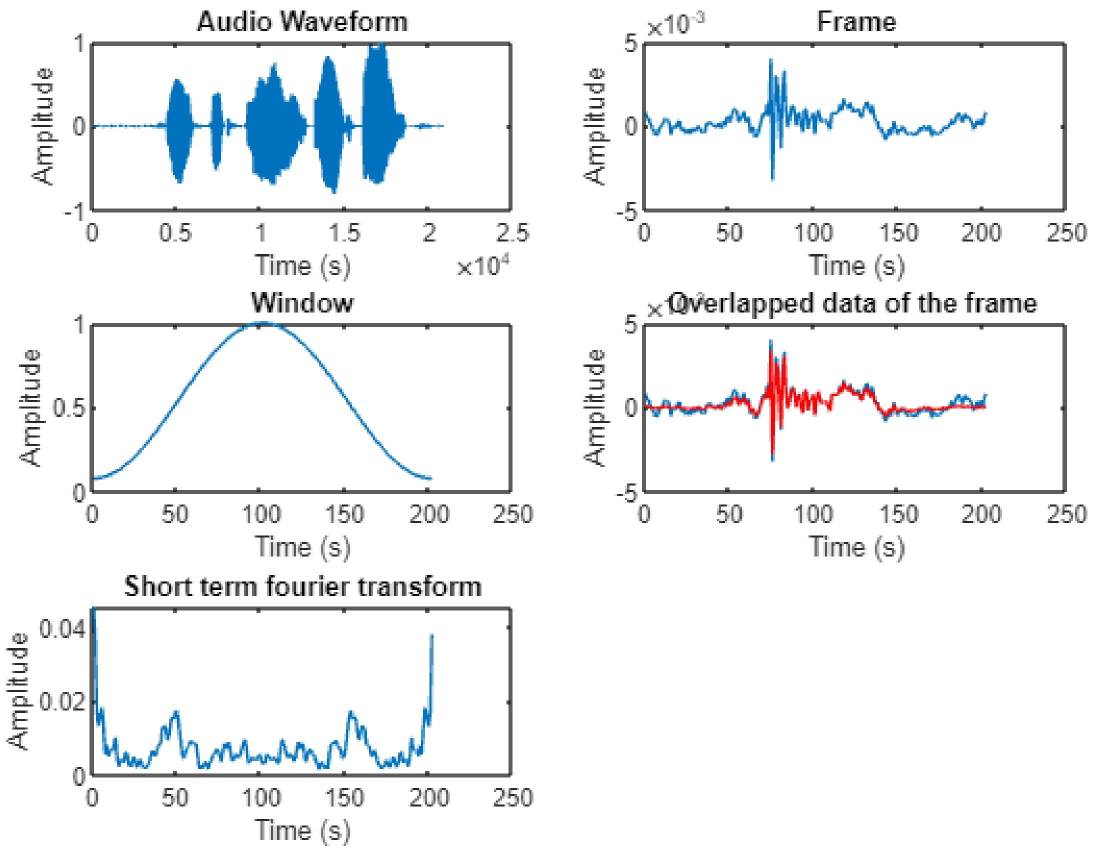
$$X_n e^{j\omega} = \sum_{m=-\infty}^{\infty} \mathrm{x}(m)w(n-m)e^{-j\omega m}$$

Code:

```
close all;
clear all;

% read wavefile
[Y, fs] = audioread('/MATLAB Drive/ac_369o5a.wav');
subplot(3,2,1);
plot(Y);
xlabel('Time (s)');
ylabel('Amplitude');
title('Audio Waveform');

%take frame of 25ms
fd = 0.025;
f_size = fd*fs;
fr= Y(2500:2501+f_size);
subplot(3,2,2)
plot(fr);
xlabel('Time (s)');
ylabel('Amplitude');
title('Frame');

%short term fourier transform
subplot(3,2,3)
plot(hamming(length(fr)));
xlabel('Time (s)');
ylabel('Amplitude');
title('Window')
fr_win = fr.*hamming(length(fr));
subplot(3,2,4)
```

```
plot(fr)
hold on;
plot(fr_win, 'r');
xlabel('Time (s)');
ylabel('Amplitude');
title('Overlapped data of the frame ')
stft = fft(fr_win);
subplot(3,2,5)
plot(abs(stft))
xlabel('Time (s)');
ylabel('Amplitude');
title('Short term fourier transform')
```

Output:



Conclusion:

With the window of 25ms, we have successfully calculated the STFT with the help of Fourier transform.

Experiment 4

Aim: Write the MATLAB code for reading a wave file and do source to system separation.

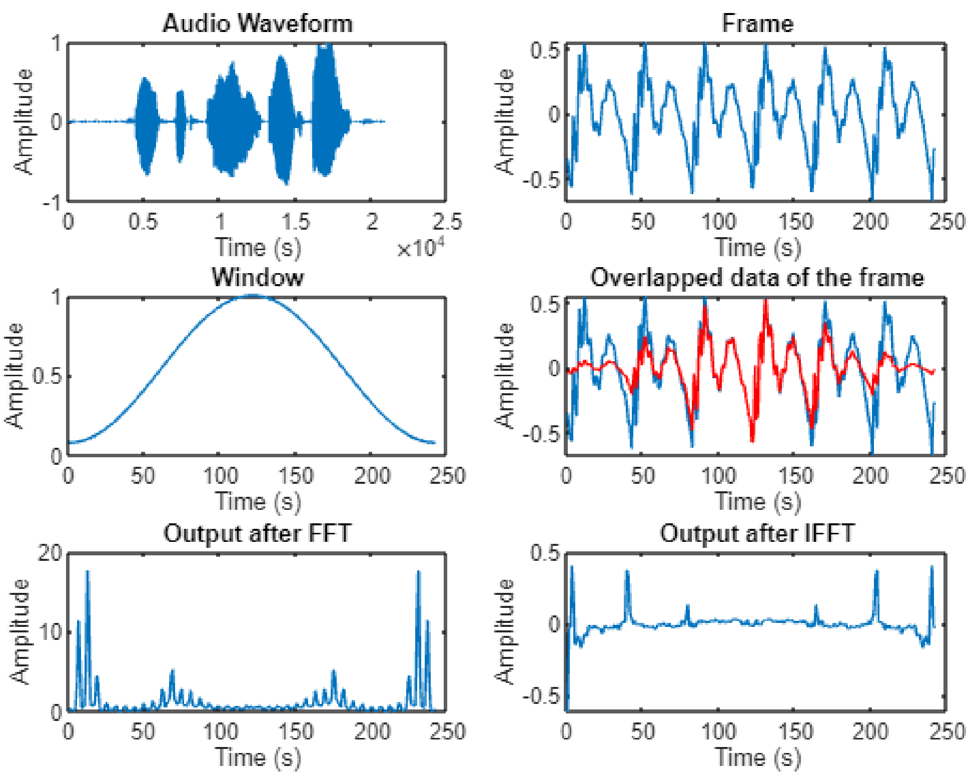
Theory: Source to system separation refers to the process of separating the information related to the vocal tract (known as the source information) from the excitation information. This process involves several steps, starting with taking the Fast Fourier Transform (FFT) of the signal at the beginning. Then, we calculate the magnitude and take the logarithm of the resulting output. Finally, we perform the Inverse Fast Fourier Transform (IFFT) to achieve the separation..

Code:

```
%System and source separation


close all;
clear all;
clc;

% read wavefile
[Y, fs] = audioread('/MATLAB Drive/ac_369o5a.wav');
subplot(3,2,1);
plot(Y);
xlabel('Time (s)');
ylabel('Amplitude');
title('Audio Waveform');

%take frame of 25ms
fd = 0.03;
f_size = fd*fs;
fr= Y(5000:5001+f_size);
subplot(3,2,2)
plot(fr);
xlabel('Time (s)');
ylabel('Amplitude');
title('frame');

%short term fourier transform
subplot(3,2,3)
plot(hamming(length(fr)));
xlabel('Time (s)');
ylabel('Amplitude');
title('window')

fr_win = fr.*hamming(length(fr));
subplot(3,2,4)
plot(fr)
hold on;
plot(fr_win, 'r');
xlabel('Time (s)');
ylabel('Amplitude');

hw = fft(fr_win);
subplot(3,2,5)
plot(abs(hw))
xlabel('Time (s)');
ylabel('Amplitude');
```

```
title('output after FFT')
L= log(abs(hw));

C = ifft(L);
display(C);
subplot(3,2,6)
plot(C)
xlabel('Time (s)');
ylabel('Amplitude');
title('output after IFFT')
```

Output:



Conclusion:

Learn to separate the source from system, The path of widowing is taken at first, lead with calculating the fast Fourier transform log and magnitude of the extracted output and finally calculated the IFFT.

Experiment 5

Aim: Write the MATLAB code for reading a wave file and extract the MFCC features.

Theory: MFCC is an acronym for Mel-frequency Cepstral Coefficients, which is a method of extracting features from speech signals that captures crucial spectral features. It works by examining the spectral characteristics of an audio signal and mapping them to a logarithmic scale called the mel-frequency scale, which approximates the frequency response of the human auditory system. The amplitudes of different frequency bands are then computed and utilized as input features for various machine learning models.

Reading a wave file: A wave file is a digital audio file format that stores audio signals as a sequence of samples. The first step in extracting MFCC features from a wave file is to read the audio data into memory.

Pre-emphasis: Pre-emphasis is a filtering technique used in audio processing to boost the high-frequency content of a signal. The idea is to apply a high-pass filter to the audio signal to amplify the high-frequency components before down sampling it. This can help to improve the signal-to-noise ratio and enhance the clarity of the signal.

Framing: Once the audio data is loaded into memory, it is typically divided into small, overlapping frames of fixed duration (e.g., 20-30 ms) to capture the time-varying nature of the signal.

Windowing: Windowing is a method that is utilized to minimize the spectral leakage impact, which happens when a signal with a limited length is transformed with Fourier. The concept is to apply a window function (such as Hamming, Hann, or Blackman) to every frame of the audio signal to attenuate the signal at the edges and reduce the sidelobes in the frequency domain.

Discrete Fourier Transform (DFT): The DFT is a mathematical method that transforms a signal in the time-domain to its corresponding representation in the frequency-domain. For each frame of the audio signal, the DFT can be calculated using the fast Fourier transform (FFT) algorithm.

Mel-frequency warping: To account for the human ear's varying sensitivity to changes in frequency across different frequencies, a mel-frequency scale is often used to map the DFT coefficients. This scale compresses higher frequencies and expands lower frequencies using triangular filters placed at intervals on the mel-frequency scale.

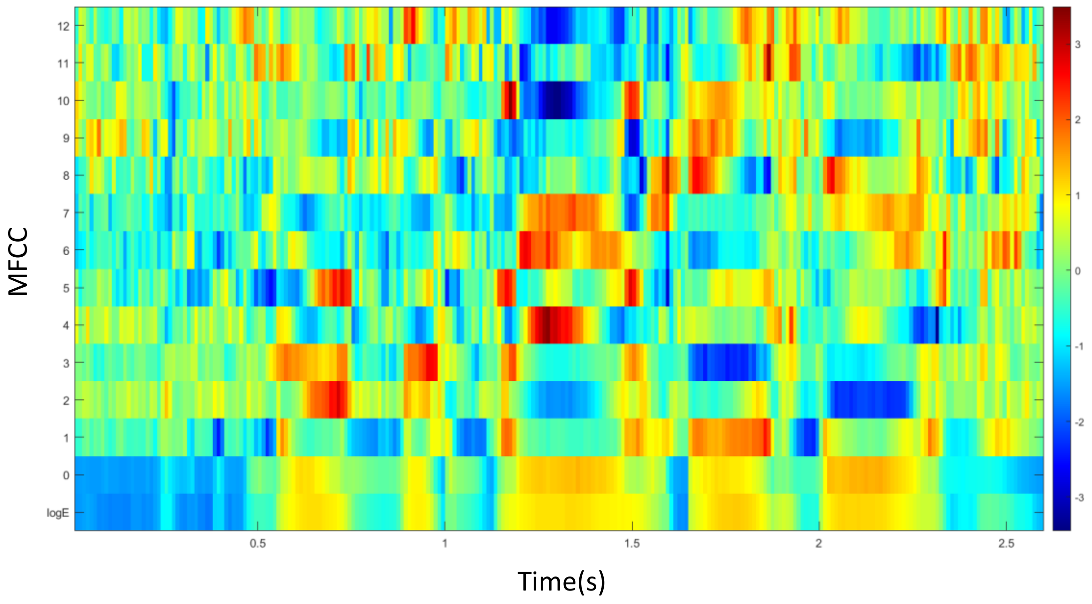
Cepstral analysis: Cepstral analysis is a technique used to extract the spectral envelope of a signal. The MFCC features are obtained by taking the inverse DFT of the mel-scaled log-spectrum of each frame, and then calculating the cepstral coefficients. The MFCC coefficients can be further processed by applying various techniques, such as mean normalization, delta features, and feature selection, to improve the performance of the speech recognition or audio analysis system.

To summarize, extracting MFCC features from a wave file involves several steps such as pre-emphasis, framing, windowing, DFT, mel-frequency scaling, and cepstral analysis. These features are useful as input to machine learning algorithms that perform various audio analysis tasks, such as speaker identification, speech recognition, and music genre classification.

Code:

```
%Melscale frequency cepstral coefficient: MFCC

close all;
clear all;
clc

[Y, fs] = audioread('/MATLAB Drive/ac_369o5a.wav');

coff = mfcc(Y,fs,LogEnergy = "replace");
mfcc(Y,fs);
```

Output:



Conclusion:

We learnt to read the wav file and extracted the MFCC features following the steps below:

- Pre-emphasis
- Framing
- Windowing
- DFT
- Mel-frequency warping
- Cepstral analysis