Speech Processing Using MATLAB

Submitted by:

Hamza Umar

FA19-BCE-026

Rahim Ullah Khan

FA19-BCE-009

Muhammad Kaleem Ullah

FA19-BCE-007

Program: BS in Computer Engineering

Submitted to:

Engr. M Abdul Rehman

A Project submitted in partial fulfillment of the requirements for the degree of bachelors of Science in Computer Engineering

Department of Electrical and Computer Engineering

COMSATS University Islamabad, Attock Campus, Pakistan.

Declaration

I declare that the project report SPEECH PROCESSING USING MATLAB is based on our own work carried out during the course of our study under the supervision of Engr. M Abdul Rehman.
I assert the statements made and conclusions drawn are an outcome of my research work. I further certify that

1. The work contained in the report is original and has been done by us under the general supervision of my supervisor.
2. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of Pakistan or abroad.
3. We have followed the guidelines provided by the university in writing the report.
4. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

Dedication

First and foremost we offer our sincerest gratitude to our course instructor, Engr. M Abdul Rehman, Who encouragement, guidance and support from the initial to the final level enabled us to develop an understanding of the subject. Without his guidance and persistent help this project would not have been possible.

To our parents, we would like to thank to them for supporting us in our daily lives, for going to school every day, and having them by our side to guide us always, their prosperity and love for us.

Acknowledgements

Thanks to ALLAH (s.w.t), the Greatest, the most Merciful and the most Gracious, Whose countless blessings bestowed upon me kind, talented and wise teachers, who provided me sufficient opportunities, and enlighten me towards this research work.

I would like to extend my deepest thanks to our project supervisor, Engr. M Abdul Rehman for giving ua the opportunity of undertaking this project under his determined directions. His support, dedication, encouragement, excellent supervision and guidance are what made this thesis possible.

Thanks to my beloved family, whose prayers, dedication, support and love are the most precious assets, I had (and I have), during the course of my Engineering work and for all of my endeavors.

I am very thankful to the administration and faculty of COMSATS University Islamabad, Attock Campus for providing me a great environment that helped me a lot in conducting our project related activities.

Thank You!

Abstract

Speech processing using MATLAB gives the reader a comprehensive overview of contemporary speech and audio processing techniques with an emphasis on practical implementations and illustrations using MATLAB code. Speech process refers to the analysis of speech signals and their processing to obtain useful information. Speech processing can also be referred as digital signal processing, as the speech signals are digitized for processing. Core concepts are first covered in an introduction to the physics of audio and vibration together with their representations using complex numbers,Z transforms, and frequency analysis transforms such as the FFT.

Speech processing involves identifying the isolated word from the corresponding speech signal. From the speech signal that corresponds to the particular word, one or more features such as linear predictive coefficients (LPC), Mel-frequency cepstral coefficients (MFCC) (refer Chap.3) are collected and are arranged as the elements to form the vector. This is known as the feature extraction, and the corresponding vector is known as feature vector. Every element in the vector is known as the attributes. Suppose we need to design the digit classifier that classifies the word zero to nine (digits) from the corresponding speech signal. About large number (several hundreds) of feature vectors corresponding to the individual digits are collected from the various speakers to design the speaker-independent classifier. Fifty percent of the collected data are used for designing the classifier.

Speech processing is a computerized speech text process in voice is usually recorded with acoustic microphones by capturing air pressure changes. This kind of air transmitted voice signals is prone to two kinds of problems related to voice robustness and applicability. The former means mixing of speech signals and ambient noise usually deteriorate automatic voice recognition system performance. The latter means speech could be overheard easily on air transmission channel and this often results in privacy loss or annoyance to other people.

The following module describes the process behind implementing signal processing techniques in MATLAB. The algorithm utilizes the Discrete Fourier Transform in order to compare the frequency spectra of two voices. Further more, we are comparing and analysing original signal with Fourier Transform to find highest frequency of signal, and decimating re-sampling data at a lower rate after low pass filtering.

Contents

List of Figures

Chapter 1

Introduction

Signals are time-varying quantities which carry information. They may be, for example, audio signals (speech, music), images or video signals, sonar signals or ultrasound, biological signals such as the electrical pulses from the heart, communications signals, or many other types. With the emergence of high-speed, low-cost computing hardware, we now have the opportunity to analyze and process signals via computer algorithms.

The basic idea is straightforward: Rather than design complex circuits to process signals, the signal is first converted into a sequence of numbers and processed via software. By its very nature, software is more easily extensible and more versatile as compared with hard-wired circuits, which are difficult to change. Furthermore, using software, we can build in more intelligence into the operation of our designs and thus develop more human-usable devices.

1.1 Objectives

The objectives of this project are:

1. To get familiar with how to describe signals mathematically and understand how to perform mathematical operations on signals.
2. It will provide knowledge of digital filter
3. To discuss word length issues ,multi rate signal processing and application.

1.2 Introduction

Audio signal processing is the subfield of signal processing in which we work with the audio or what we hear. Different objects produce sound like bird, engines, motors noise in robotic systems. The human can hear these sounds but there is always a mixture of some background noise in the audio signals that must be processed in order to get the desire signals.

Figure 1.1: Audio Signal Processing

The audio sample is taken having some background noise or by adding noise by yourself, then this audio sample is processed by analyzing it in MATLAB or any other tools. We perform some operation of the signal, for more precisely analyzing the signals in Frequency spectrum for analyzing in frequency domain in order to remove noise. Then a filtered signal is produced at the output.

Figure 1.2: Signal Processing Technique

1.2.1 Filters

Bandpass Filters

A bandpass filter is an filter that allows signals between two explicit frequencies, however that oppresses signals at different frequencies. Some bandpass channels require an outer wellspring of intensity and utilize dynamic parts, for example, semiconductors and coordinated circuits; these are known as dynamic bandpass filters.

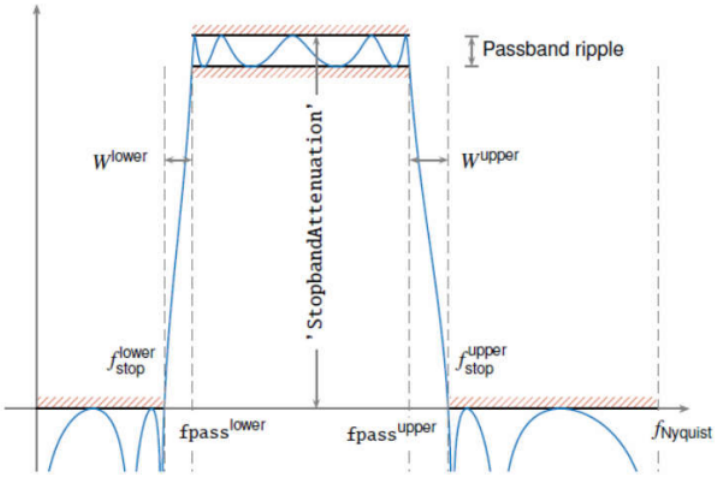

Figure 1.3: Band Pass Filter Frequency Response

Band Stop Filters

The Band Stop Filter, (BSF) is another variety of frequency selective circuit that functions in just the other thanks to the Band Pass Filter we tend to checked out before. The band stop filter, additionally referred to as a band reject filter, passes all frequencies with the exception of these inside a such stop band that are greatly attenuated.
Also, a bit like the band pass filter, the band filter may be a second-order (two-pole) filter having 2 cut-off frequencies, normally referred to as the -3dB or half-power points manufacturing a good stop band information measure between these 2 -3dB points.
So for a wide-band band stop filter, the filters actual stop band lies between its lower and higher -3dB points because it attenuates, or rejects any frequency between these 2 cut-off frequencies. The frequency response curve of a perfect band stop filter is so given as:

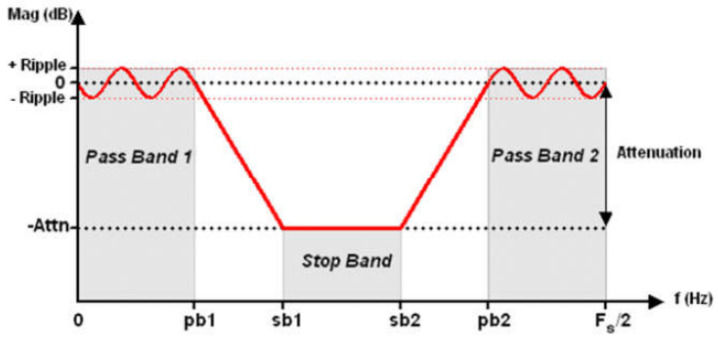

Figure 1.4: Band Stop Filter Frequency Response

Low pass Filter

A low-pass filter is a filter that permits signals under a cutoff frequency (known as the pass band) and lessens signals over the cutoff frequency (known as the stop band). By evacuating a few frequencies, the channel makes a smoothing impact. That is, the channel delivers moderate changes in yield esteems to make it simpler to see patterns and lift the general SNR with insignificant signal noise.

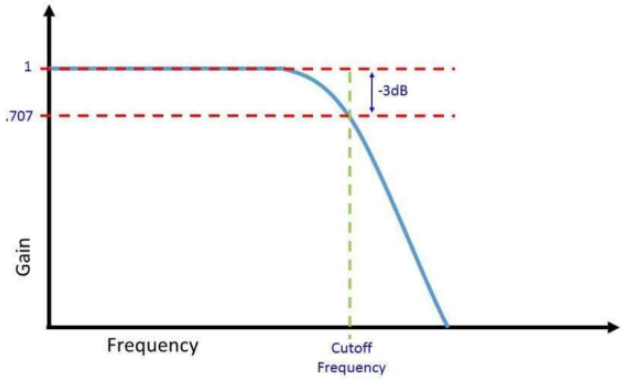

Figure 1.5: Low Pass Filter Frequency Response

1.3 Report Break Down

The major focus of this report is on the findings of the proposed project i.e. Speech Processing Using MATLAB

This Report is organized as follows:

In chapter 2, literature review is provided in detail about the work which is already been done on speech processing using MATLAB and will give a brief details about the articles, papers and literature review.

In Chapter 3, Proposed Methodology is presented in which you will be able to see the method we will work on the designing of a complete project source code to the diagrams.

In Chapter 4, Result and Simulations are being discussed, in which you will see all kind of finding related to the speech processing using MATLAB.

In Chapter 5, we have concluded and summarized the project work and also presented few new research ideas for future studies.

Chapter 2

Literature Review

In the chapter 1 we have given the introduction of our project, objectives and a thesis break down. Our introduction chapter is giving a complete overview of this project report. This chapter is about the work which is already been done on speech processing using MATLAB and will give a brief details about the articles, papers and literature review.

2.1 Literature Review

This chapter is about studies and literatures that are related to the Automatic Meter Reading that the proponents made use of different reading materials (such as thesis, articles, and other web articles) that will help extending the knowledge of the topic. These reading materials will also guide the proponent to improve and develop their proposed system more effectively.

Speech is one of the most important medium by which a communication can take place. With the invention and widespread use of mobiles, telephones, data storage devices etc. has provided a major help in setting up of speech communication and its analysing. The term and the basic concept of speech identification was began in the early 1960‘s with exploration into voiceprint analysis which was somewhat similar to fingerprint concept. It was in 1984 that a science fiction called Star Trek to George Orwell‘s, derived the concept that a machine can recognize the human voice.[1]
Nowadays, with further growth & advancement in the field of speech recognition, the humans who are physically challenged such as blind and deaf can easily communicate with the machines. So in biological terms a voice that is being generated through trachea will be decoded by brain.[2]

The critical need for computationally efficient and user-friendly numerical software is hardly a matter of any controversy, although a number of key questions still remain wide open. In the current computing hardware landscape, any substantial gains in computational efficiency beyond the well-established best-practice techniques and standard software development tools (compilers, libraries, frameworks, etc.) usually come at the price of an increased customization for the target hardware. Such customizations tend to reduce the portability, readability, and maintainability of the code and frequently require specialized software development tools and skills to use them. The efforts to bridge the chasm between the readability/maintainability of numerical solver packages and the optimum performance on a given hardware are very active right now [3] and generally aim to create higher abstraction levels for numeri-

cal software development such as domain specific languages (DSLs) paired with code generation capabilities.[4],[5]

2.2 Concluding Remarks

In this chapter, we have given a overall review about the literature related to Speech Processing Using MATLAB. MATLAB is a programming and numeric computing platform used by millions of engineers and scientists to analyze data, develop algorithms, and create models.

In the chapter 3 we will proposed methodology which we will use for our project with block diagrams, flowcharts, Mathematical Modeling, escudo code and component selection related to the hardware and software.

Chapter 3

Proposed Methodology

In the previous chapter, we have discussed the theories that support this research related to the speech processing using MATLAB . In this chapter we will proposed our methodology for this project.

Basically we are following the given below question step by step. In this Matlab problem you will see the effects of Decimation & Interpolation on an audio signal. Record your speech of about 5 seconds at sampling frequency of 32 kHz. You can use either wavrecord() MATLAB function or windows sound recorder for this purpose. If you use Sound Recorder, then u need to first set its properties for recording sound at 32 kHz and single channel.

1. Plot the time and frequency domain magnitude spectrums of this speech signal. (Use wavread() to read the speech signal. Use freqz() to plot the frequency spectrum). What is highest frequency of this signal? Play this signal using wavplay().

2. Decimate the signal by a factor of 2. Again plot the time and frequency domain spectrum of the signal. Play the sound. What do you observe?

3. Again decimate the signal by a factor of 2. Plot the time and frequency domain spectrum of the signal. Play the sound. What do you observe?

4. Now interpolate the signal by a factor of 2. Plot the time and frequency domain spectrum of the signal. Play the sound. What do you observe?

5. Again interpolate the signal by a factor of 2. Plot the time and frequency domain spectrum of the signal. Play the sound. What do you observe?

3.1 Block Diagram

Let us look at the simplified block diagram in Figure 3.1, which illustrates the main components involved in recording an audio signal with MATLAB. A microphone converts acoustic sound waves (which are essentially variations in air pressure over time) into continuous electronic signals (voltages). These voltages are then filtered using a so-called low-pass filter with cut-off frequency fs/2.

In non-technical terms, frequencies in the analog voltage signal higher than fs/2 will be removed completelyfrequencies below fs/2 are left untouched. The filtered signal is then sampled at a sampling rate of fs in a so-called analog-to-digital converter (ADC, for short). Since frequencies higher than fs/2 cannot be represented when sampling a continuous signals, the low-pass filter is essential in the sampling process. (Maybe you recall the activity above where you sampled your voice at fs = 32000 Hz; high frequencies were completely filtered out!) The digitized samples are then transmitted to MATLAB and stored in a vector.

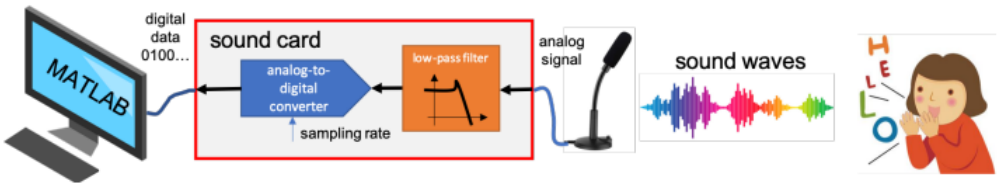

Figure 3.1: Simplified block diagram of a microphone, sound-card, and computer. The microphone converts air pressure into voltages, which are filtered and sampled. The samples are then transferred to MATLAB.

3.2 Flow Diagram

Figure 3.3 shows the flow of the decimate function. In this project decimate function is the main function which we are using to process our signal with different factors.Decimation reduces the original sampling rate for a sequence to a lower rate, the opposite of interpolation. The decimation process filters the input data with a low pass filter and then resamples the resulting smoothed signal at a lower rate.

y = decimate(x,r) reduces the sample rate of x, the input signal, by a factor of r. The decimated vector, y, is shortened by a factor of r so that

$$length(y) = ceil(length\frac{x}{r})$$

By default, decimate uses a low pass Chebyshev Type I infinite impulse response (IIR) filter of order 8.

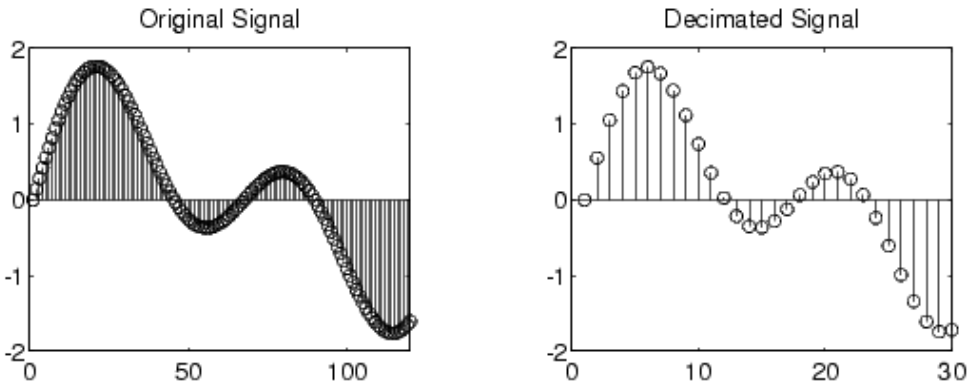

Figure 3.2: Example of Decimating A Signal

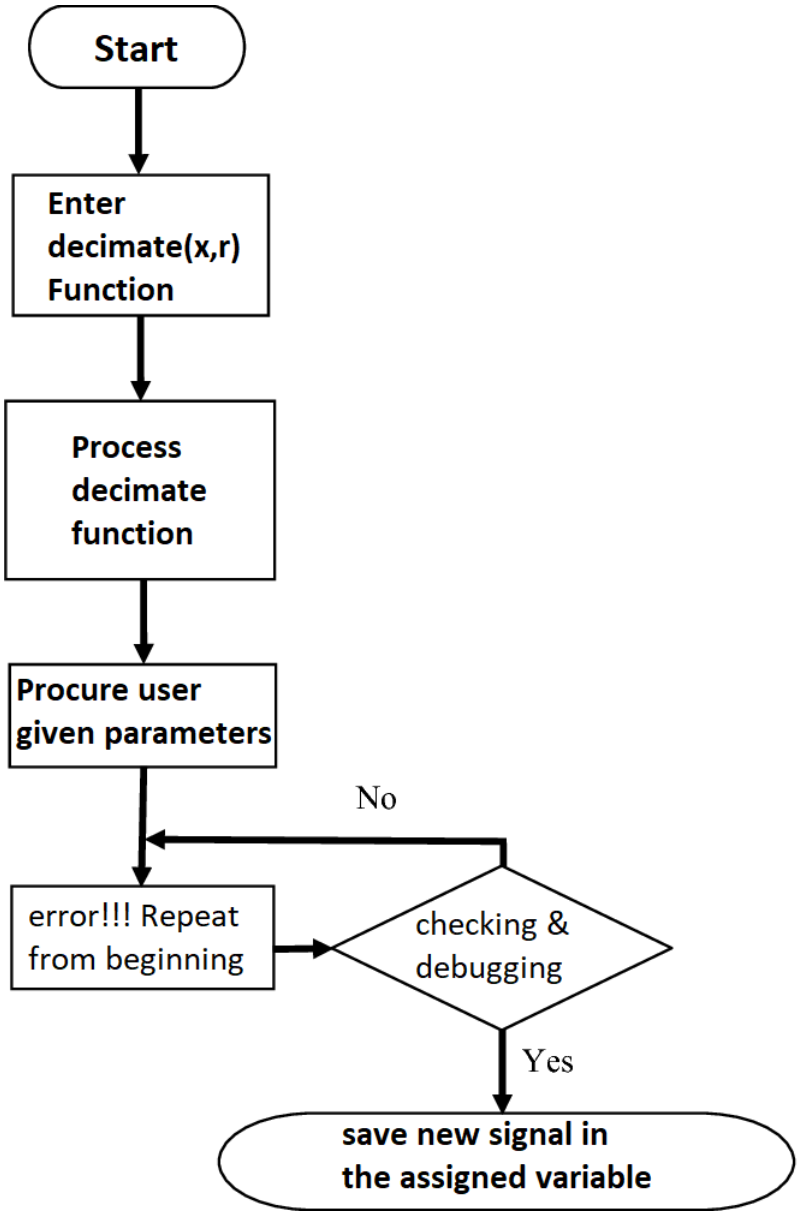

Figure 3.3: Flow Diagram of the Project

3.3 System Model

DSP System Toolbox provides algorithms and tools for the design and simulation of signal processing systems. These capabilities are provided as MATLAB functions, MATLAB System objects, and Simulink blocks. The system toolbox includes design methods for specialized FIR and IIR filters, FFTs, multirate processing, and DSP techniques for processing streaming data and creating real-time prototypes.

You can design adaptive and multirate filters, implement filters using computationally efficient architectures, and simulate floating-point digital filters. Tools for signal I/O from files and devices, signal generation, spectral analysis, and interactive visualization enable you to analyze system behavior and performance. For rapid prototyping and embedded system design, the system toolbox supports fixed-point arithmetic and C or HDL code generation.

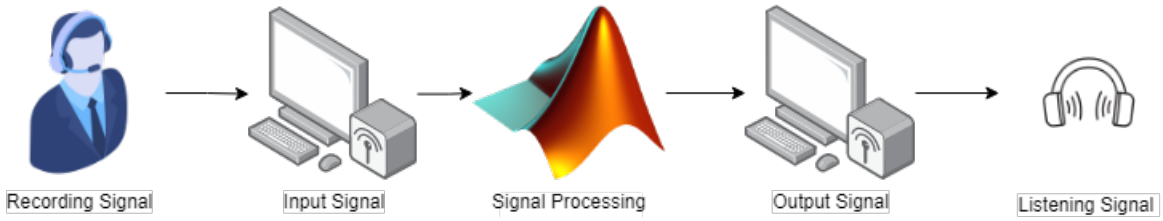

Figure 3.4: System Model of the Project

3.4 Component Selection

Signal processing engineers use MATLAB and Simulink at all stages of development-from analyzing signals and exploring algorithms to evaluating design implementation tradeoffs for building real-time signal processing systems. MATLAB and Simulink offer:

1. Built-in functions and apps for analysis and preprocessing of time-series data, spectral and time-frequency analysis, and signal measurements
2. Apps and algorithms to design, analyze, and implement digital filters (FIR and IIR) from basic FIR and IIR filters to adaptive, multirate, and multistage designs
3. An environment to model and simulate signal processing systems with a combination of programs and block diagrams
4. Capabilities to model fixed-point behavior and automatically generate C/C++ or HDL code for deploying on embedded processors, FPGAs, and ASICs
5. Tools for developing predictive models on signals and sensor data using machine learning and deep learning workflows

Note: Make sure you are using MATLAB 2013 or any less Version because wavrecord, wavplay are being replaced with audiorecorder/getaudiodata.

Figure 3.5: Signal processing engineer using MATLAB and Simulink

Chapter 4

Simulation and Results

In the previous chapter, we have discussed about our methodology for this project while giving details about block diagram, system model, flow chart, software selection of this project. In this chapter, we will provide source code, and simulation and results of the project.

4.1 Simulation Results

As discussed in the previous chapter in the section of software selection, we have used MATLAB to simulate our project. We have used two MATLAB function. In the first file, we have used that for the recording of the input signal where as the second file is accessing the input signal while calling that audio signal file and is performing the rest of the operation on that. Each line is properly comment to understand the use of that command.

Note: Make sure both file should be in the same folder.

4.2 Source Code

```
    % First(1) File
clc %Clear command window.
close all %closes all the open figure windows.

% sampling frequency
F = 32000;

%  recording speech on MATLAB
%  5*F is recording speech of 5 secs of sampling frequency
%  datatype to store the sound

y  = wavrecord(5*F, F, 'int16');
%Record sound using Windows audio input device.
%Y = wavrecord(N,FS,CH) records N audio samples at FS Hertz from
%CH number of input channels from the Windows WAVE audio device.
%playing recorded speech
```

```

wavplay(y, F);
% wavplay Play sound using Windows audio output device.
%wavplay(Y,FS) sends the signal in vector Y with sample frequency
%of FS Hertz to the Windows WAVE audio device.

audiowrite('project.wav',y, F)
%it will write the speech signal of the current file(save in memory)
%audiowrite write audio files
%audiowrite(FILENAME,Y,FS)  writes data Y to an audio
%file specified by the file name FILENAME, with a sample rate
%of FS Hz.
```

```
    % Second(2) File
clc
% (part a)
% wavread will read the .wav file in which speech is recorded
% y is samples or the input signal
% fs is the sampling frequency
[y,fs] = wavread('project.wav');
wavplay(y,fs);   %wavplay Play sound using Windows audio output device.
subplot(2,1,1)
% plotting frequency spectrum
% y is input signal
%freqz Frequency response of digital filter
freqz(abs(y))
% setting axis limits
x = linspace(0,0.1);    %linspace Linearly spaced vector.
xlim([0  0.1])   % xlim X limits.
ylim([-80  80])  % ylim Y limits.
title('Freq doamin magnitude spectrum of speech.');
subplot(2,1,2)% subplot Create axes in tiled positions.
% plotting time spectrum
plot(y),grid on;              % Linear plot &  Grid lines.
title('Time domain');         %Graph title.
xlabel('Seconds');           %X-axis label.
ylabel('Apmlitude');          %ylabel

% figure used to make the graphs appear in different windows
figure;
% fft is Fourier Transform to find highest frequency of signal
z=fft(y);
plot(abs(z)),grid on;              % Linear plot &  Grid lines.
title('Highest frequency');       %graph title
xlabel('Freq Hz');                %x-axis label
ylabel('Power');                  %y-axis label
```

```
34
35 figure;
36 % (part b)
37 subplot(2,1,1)% subplot Create axes in tiled positions.
38 % decimating or downsampling the speech by facor 2
39 y1 = decimate(y,2)
40 %decimate is a low pass filter
41 freqz(abs(y1))%freqz frequency response of digital filter
42 x = linspace(0,0.1);    %linspace Linearly spaced vector.
43 xlim([0  0.1])          %x-axis limit
44 ylim([-80  80])         %y-axis limit
45 title('Freq doamin magnitude spectrum after decimating by 2')
      ;%graph title
46 subplot(2,1,2)% subplot Create axes in tiled positions.
47 plot(y1),grid on;% Linear plot &  Grid lines.
48 title('Time domain');       %graph title
49 xlabel('Seconds');          %x-axis label
50 ylabel('Apmlitude');        %y-axis label
51 wavplay(y1,fs);   %wavplay Play sound using Windows audio
      output device.
52
53
54 figure;          % figure Create figure window.
55 % (part c)
56 subplot(2,1,1)
57 % decimating by 4
58 y2 = decimate(y,2*2)
59 %decimate Resample data at a lower rate after lowpass
      filtering.
60 freqz(abs(y2))%freqz frequency response of digital filter.
61 x = linspace(0,0.1); %linearly spaced vector.
62 xlim([0  0.1]) %x-axis limit
63 ylim([-80  80])%y-axis limit
64 title('Freq Doamin Magnitude Spectrum After Decimating by 4')
      ;%graph title
65 subplot(2,1,2)% subplot Create axes in tiled positions.
66 plot(y2),grid on;% Linear plot &  Grid lines.
67 title('Time domain'); %graph title
68 xlabel('Seconds'); %x-axis label
69 ylabel('Apmlitude');%y-axis label
70 wavplay(y2,fs); %wavplay play sound using windows audio
      output device.
71
72
73 figure; %create figure window
74 % (part d)
75 subplot(2,1,1)% subplot Create axes in tiled positions.
76 % decimating by 8
77 y3 = decimate(y,2*2*2)
```

```
78 %decimate resample data at a lower rate after lowpass
       filtering
79 freqz(abs(y3)) %frequency response of digital filter.
80 x = linspace(0,0.1); %linearly spaced vector
81 xlim([0  0.1]) %x-axis limit
82 ylim([-60  60])%y-axis limit
83 title('Freq doamin magnitude spectrum after decimating by 8')
       ;%graph title
84 subplot(2,1,2)% subplot Create axes in tiled positions.
85 plot(y3),grid on;% Linear plot &  Grid lines.
86 title('Time domain'); %graph title
87 xlabel('Seconds');%x-axis label
88 ylabel('Apmlitude');%y-axis label
89 wavplay(y3,fs); %plays sound using windows audio output
       device
90
91
92 figure;%create figure window
93 % (part e)
94 subplot(2,1,1)% subplot Create axes in tiled positions.
95 % decimating by 16
96 % decimate resample data at a lower rate after lowpass
       filtering
97 y4 = decimate(y,2*2*2*2)
98 freqz(abs(y4))%frequency response of digital filter.
99 x = linspace(0,0.1);%linearly spaced vector
100 xlim([0  0.1])%x-axis limit
101 ylim([-50  50])%y-axis limit
102 title('Freq doamin magnitude spectrum after decimating by 16'
       );%graph title
103 subplot(2,1,2)% subplot Create axes in tiled positions.
104 plot(y4),grid on;% Linear plot &  Grid lines.
105 title('Time domain');%graph title
106 xlabel('Seconds');%x-axis label
107 ylabel('Apmlitude');%y-axis label
108 wavplay(y4,fs);%plays sound using windows audio output device
109
110 z=fft(y4);
111 figure;
112 plot(abs(z)),grid on;
```

4.3 Statistical Analysis

Frequency spectrum of Input signal and input signal in time domain is Fig. 4.1 .

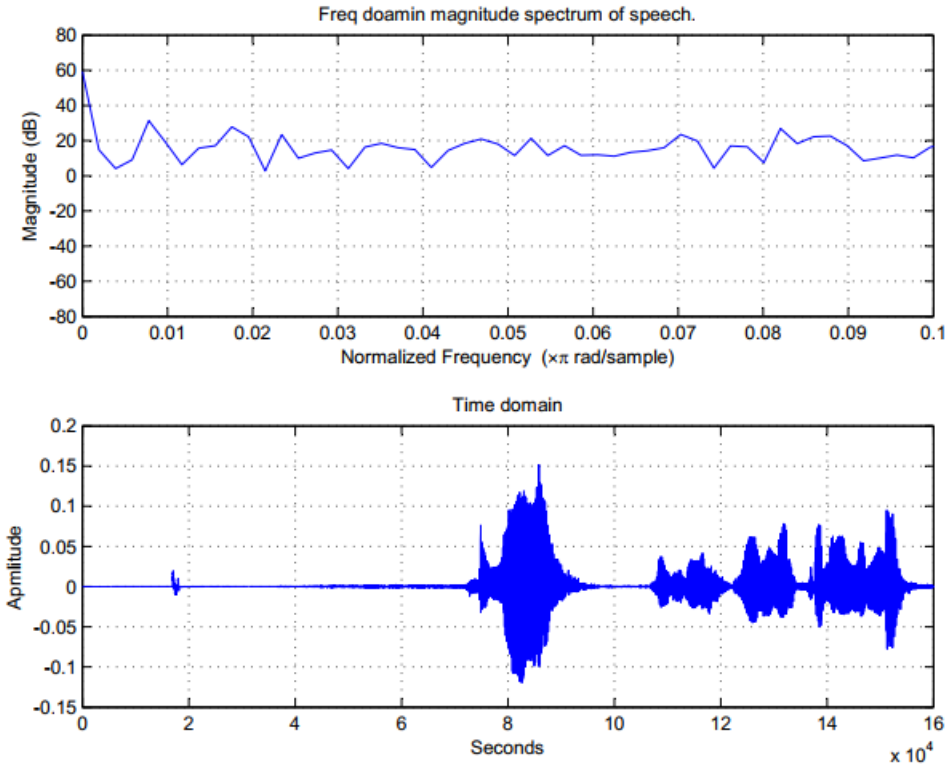

Figure 4.1: Comparison of Frequency Spectrum and Time Domain Graph of Input Signal

The highest frequency of recorded speech signal is 327kHz from fft is Fourier Transform to find highest frequency of signal which is shown in Fig. 4.2

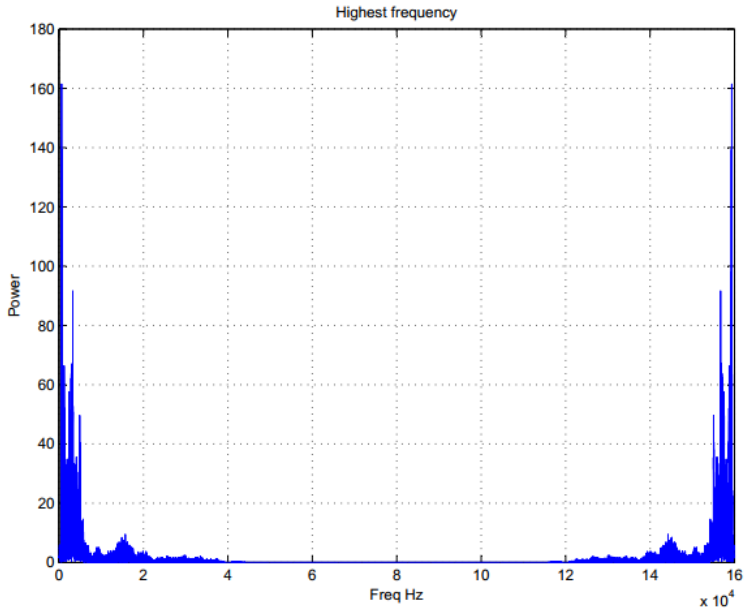

Figure 4.2: Fourier Transform of the Input Signal

Data transmitted per unit time is increased so does the speed of transmission or the speed of speech is increased. Another thing we observed by decimating by 2 is that the amplitude of frequency magnitude spectrum is got lowered from amplitude of original frequency magnitude spectrum.

For example, if we take any sample point lets say at 0.1, the amplitude of original frequency magnitude spectrum at that sample point is almost 30dB but when decimated by 2 the amplitude of decimated frequency magnitude spectrum goes to 28dB. Therefore, speech is not that much clear to understand than original speech and time is reduced as shown in the Fig. 4.3.

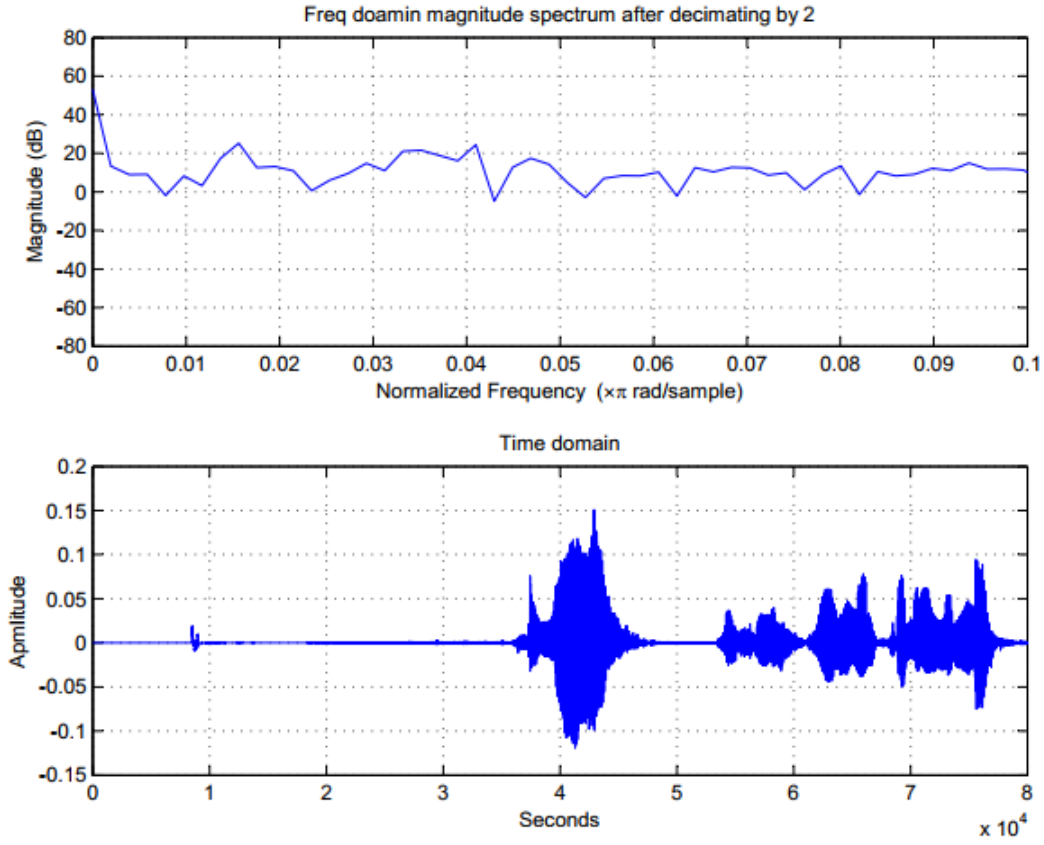

Figure 4.3: Comparison of Frequency Spectrum and Time Domain Graph of Input Signal After Decimating by 2

Data transmitted rate is much more increased. By decimating by 4 the amplitude of frequency magnitude spectrum is got much lowered from amplitude of original frequency magnitude spectrum.

For example, if we take any sample point lets say at 0.1, the amplitude of original frequency magnitude spectrum at that sample point is almost 30dB but when decimated by 4 the amplitude of decimated frequency magnitude spectrum goes to 25dB. The decimated speech is very rough and not clear enough to understand because time is more decreased as shown in the Fig. 4.4.

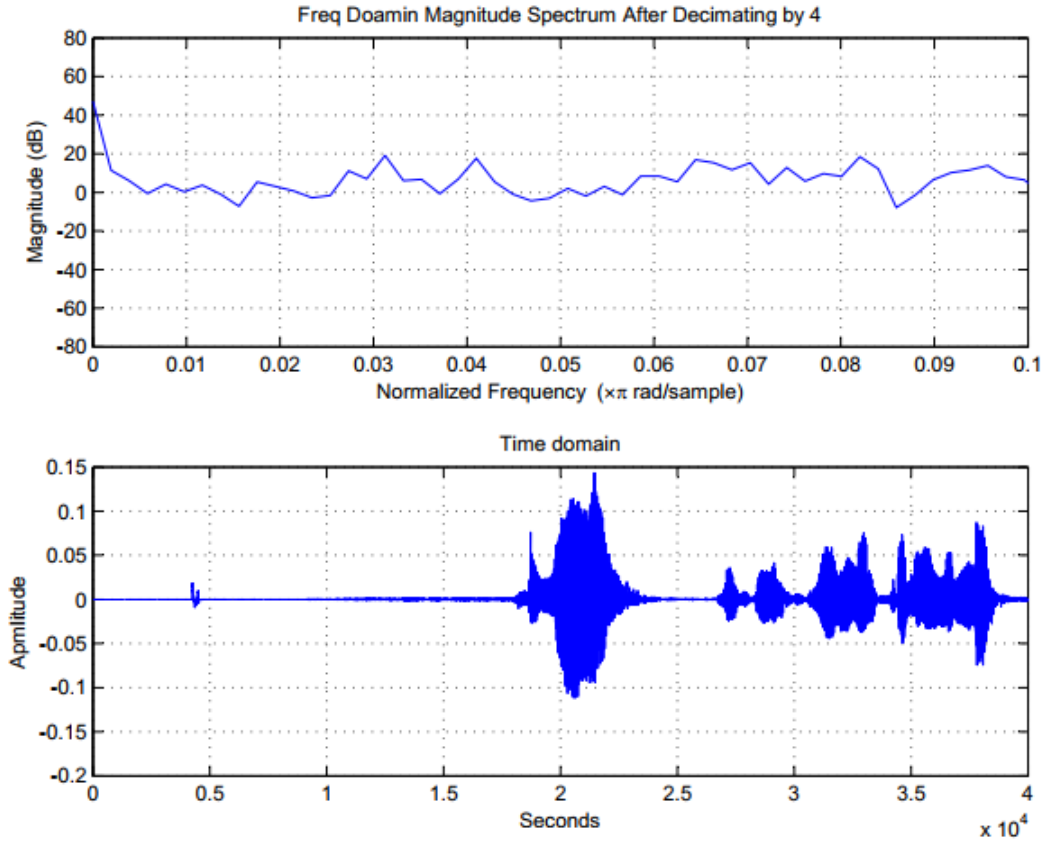

Figure 4.4: Comparison of Frequency Spectrum and Time Domain Graph of Input Signal After Decimating by 4

Decimation reduces the original sample rate of a sequence to a lower rate. It is the opposite of interpolation. decimate lowpass filters the input to guard against aliasing and downsamples the result. When using the FIR filter, decimate filters the input sequence in only one direction. This conserves memory and is useful for working with long sequences. In the IIR case, decimate applies the filter in the forward and reverse directions using filtfilt to remove phase distortion. In effect, this process doubles the filter order. In both cases, the function minimizes transient effects at both ends of the signal by matching endpoint conditions

Data transmitted per unit time is much more increased and the speed of transmission or the speed of speech is increased. Another thing we observed by decimating by 8 is that the amplitude of frequency magnitude spectrum is got lowered from amplitude of original frequency magnitude spectrum.

For example, if we take any sample point lets say at 0.1, the amplitude of original frequency magnitude spectrum at that sample point is almost 30dB but when decimated by 8 the amplitude of decimated frequency magnitude spectrum goes to 18dB. The speech is played so fast that the speech is almost impossible to understand, time is much more reduced as shown in the Fig.4.5.

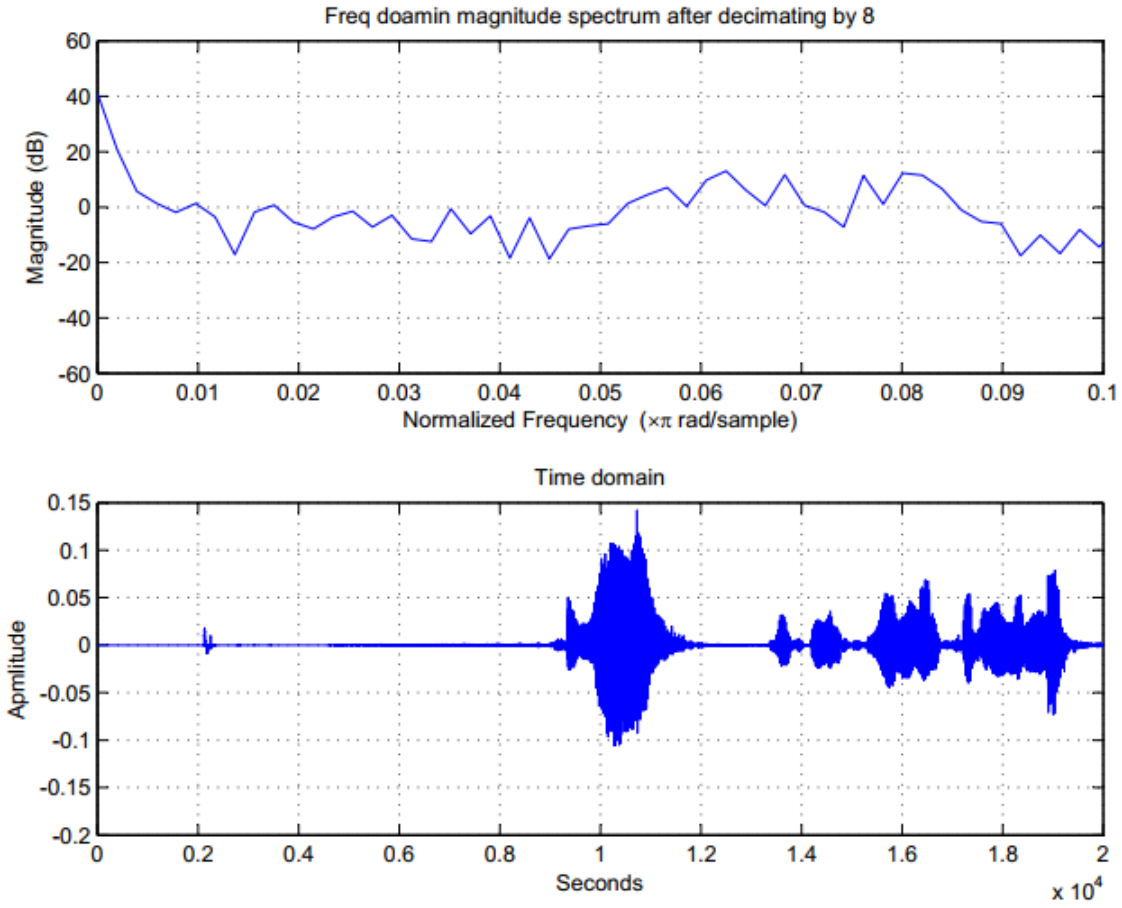

Figure 4.5: Comparison of Frequency Spectrum and Time Domain Graph of Input Signal After Decimating by 8

decimate creates a lowpass filter. The default is a Chebyshev Type I filter designed using cheby1. This filter has a normalized cutoff frequency of 0.8/r and a passband ripple of 0.05 dB. Sometimes, the specified filter order produces passband distortion due to round-off errors accumulated from the convolutions needed to create the transfer function. decimate automatically reduces the filter order when distortion causes the magnitude response at the cutoff frequency to differ from the ripple by more than $1/10^6$.
When the 'fir' option is chosen, decimate uses fir1 to design a lowpass FIR filter with cutoff frequency $1/r$.

Data transmitted per unit time is so much increased Another thing we observed by decimating by 16 is that the amplitude of frequency magnitude spectrum is got lowered from amplitude of original frequency magnitude spectrum.

For example, if we take any sample point lets say at 0.1, the amplitude of original frequency magnitude spectrum at that sample point is almost 30dB but when decimated by 16 the amplitude of decimated frequency magnitude spectrum almost goes to 1dB. The rate of speech is so less that the speech is impossible to understand, time is much more reduced as shown in the Fig. 4.6.

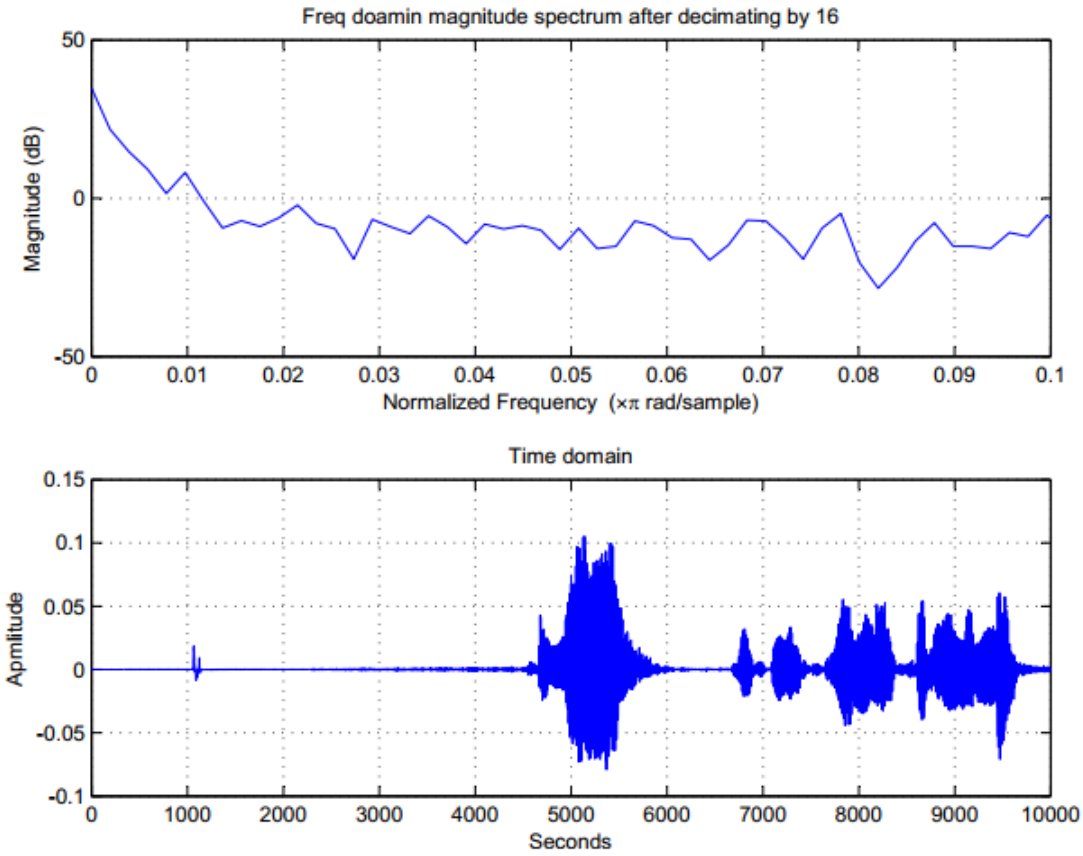

Figure 4.6: Comparison of Frequency Spectrum and Time Domain Graph of Input Signal After Decimating by 16

Chapter 5

Conclusion and Future Work

In the previous chapter, we have shown all the simulation and results of our proposed project. From introduction, Literature Review, Proposed Methodology, Result and Simulations. In this chapter, we will conclude our project and will

5.1 Conclusion

This work describes the speech processing using MATLAB. We have:

1. Describe signals mathematically and understand how to perform mathematical operations on signals.
2. This project provides knowledge of Digital filter
3. discuss word length issues ,multi rate signal processing and application

In this project, we have record our speech of about 5 seconds at sampling frequency of 32 kHz. we have use either wavrecord() MATLAB function
After that we have:

1. Plotted the time and frequency domain magnitude spectrums of this speech signal. And found 327kHz is highest frequency of this signal and played that signal using wavplay().
2. We have decimated the signal by a factor of 2 repeatedly. And again plotted the time and frequency domain spectrum of the signal. And played the sound. Observation of this step is mentioned in detail in the Chapter 4.

5.2 Future Work

Speech processing using MATLAB can be done in many other ways and the current method/project can be improve in many ways. Following are the some of the things we can try to improve this system:

1. Different method of the signal processing can be used to improve it.
2. High quality microphone can be used to improve the system accuracy.
3. Instead of using wavrecord, wavplay we can create the same project with the updated commands like audiorecorder/getaudiodata in MATLAB.
4. We can try on implementing the processing technique on multi subjects.

Bibliography

[1] D. Mandalia, P. Gareta, and R. Sharma, "Speaker recognition using mfcc and vector quantization model," *Electronics, vol. Program, C, no. May*, p. 75, 2011.

[2] A. Saxena, A. K. Sinha, S. Chakrawarti, and S. Charu, "Speech recognition using matlab," *International Journal of Advances in Computer Science and Cloud Computing*, vol. 1, no. 2, pp. 26–30, 2013.

[3] B. N. Lawrence, M. Rezny, R. Budich, P. Bauer, J. Behrens, M. Carter, W. Deconinck, R. Ford, C. Maynard, S. Mullerworth, *et al.*, "Crossing the chasm: how to develop weather and climate models for next generation computers?," *Geoscientific Model Development*, vol. 11, no. 5, pp. 1799–1821, 2018.

[4] S. Kuckuk and H. Köstler, "Whole program generation of massively parallel shallow water equation solvers," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 78–87, IEEE, 2018.

[5] T. C. Schulthess, "Programming revisited," *Nature Physics*, vol. 11, no. 5, pp. 369–373, 2015.