PROJECT REPORT

FOR

"MUSIC NOTE RECOGNITION"

Submitted to

INSTITUTE TECHNICAL SUMMER PROJECT
IIT BOMBAY

A Project on Audio Processing and analysis with Fourier Transformation

BY

Vishal Babu Bhavani	**140050049**
Pushyarag Yadagiri	**140050047**
Divya Somasi	**ROLL NUMBER C**

Under the Mentorship and Guidance of
Web and Coding Club
STAB IIT Bombay

AFFILIATED TO

IIT BOMBAY

June 2015

Acknowledgements

We are profoundly grateful to **Student Technical Activities Body,IIT Bombay** for the expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would like to express deepest appreciation towards **Web and Coding Club**, STAB, IIT Bombay, whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the mentors of ITSP who helped me directly or indirectly during this course of work.

Vishal Babu Bhavani
Pushyarag Yadagiri
Divya Somasi

ABSTRACT

The main target of the project is to get the real time estimation of the frequency of audio signal. Real time estimation will help in maintaining the data related to changes in the frequency. So we designed two different ways of estimating it. Each one has its own applications and is accurate to different types of audio.The sampling frequency is set to 44100 so that it would be compatible with all the devices.The basic approach calculates the period from the superimposition and deviation analysis of the signal.

The other method is more intelligent with respect to the processing part as it uses note detection.Note detection allows us to recognise the portions of the audio sample where we can apply Fast Fourier Transformation algorithms.So this allows us to scale down the region of analysis for efficient run time.Thus we process the data obtained from the Power Spectrum and calculate the fundamental frequency. The frequency obtained from above estimations is used to evaluate the music note names
Keywords: Sampling frequency, Note Detection, Fast Fourier Transform, Power Spectrum.

Contents

List of Figures

Chapter 1

Introduction

1.1 AUDIO PROCESSING

1.1.1 Sampling Theorem

In order to perform any form of processing by digital computers, the signals must be reduced to discrete samples of a discrete-time domain. The operation that transforms a signal from the continuous time to the discrete time is called sampling, and it is performed by picking up the values of the continuous-time signal at time instants that are multiple of a quantity T, called the sampling interval. The quantity Fs = 1/T is called the sampling rate.

1.1.2 Discrete Fourier transform

In mathematics, the discrete Fourier transform (DFT) converts a finite list of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids, ordered by their frequencies, that has those same sample values. It can be said to convert the sampled function from its original domain (often time or position along a line) to the frequency domain.The input samples are complex numbers (in practice, usually real numbers), and the output coefficients are complex as well. The frequencies of the output sinusoids are integer multiples of a fundamental frequency, whose corresponding period is the length of the sampling interval. The combination of sinusoids obtained through the DFT is therefore periodic with that same period.

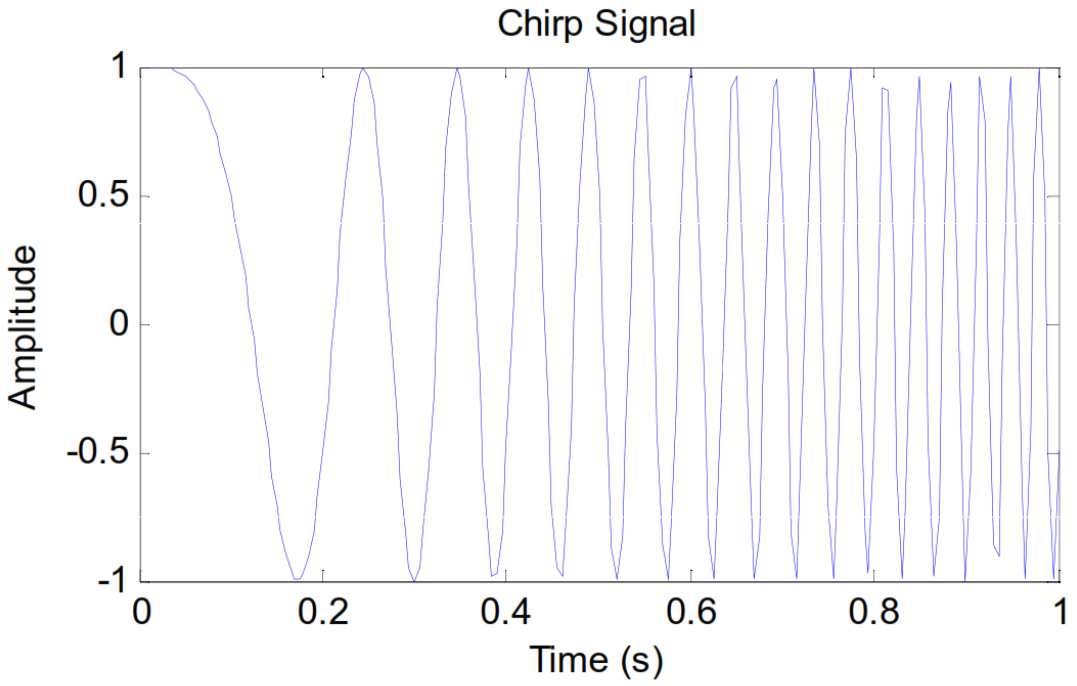


Figure 1.1: Simple Audio signal

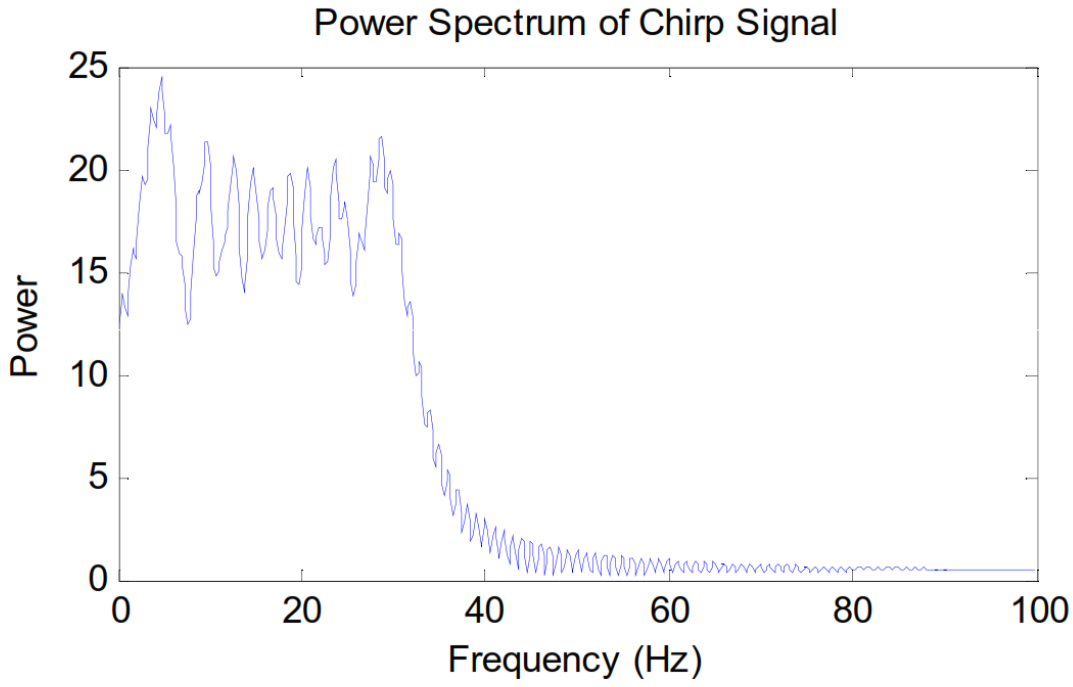


Figure 1.2: Power Spectrum

Chapter 2

Software Requirements Specification

2.1 SPECIFIC REQUIREMENT

2.1.1 Overall Description

The project requires you to use MATLAB Software as the operating environment.The User Interface is easy to handle.User needs to enter the audiofile name in the input section.This section reads the audio file with the given name in the MATLAB project directory using 'audioread' function which is only available in the latest releases(after 2011).So the user should make sure that the MATLAB software he/she is running must be up to date.The project is built in MATLAB R2014a Windows version.It portrays important graphs for advanced anlaysis if required by the user so it is important that the version also contains the basic libraries which are provided by default by the version on which the project is built.

2.1.2 External Interface Requirement

The UI is developed using MATLAB Graphical User Interface(GUI).It takes the input audio file name and to locate and store the file for processing.The AUTO mode uses intelligent note detection system to find the points where there is a note change.The MANUAL mode allows the user to select the threshold of intensity above which the notes will be detected.The run-time for the latter is more compared to the former.After the note variations are compiled together the PLAY NOTES button creates a new audio file and writes the generated audio file to it.The UI also provides advanced feaatures like plots of power spectrum of each note,note name according to standard sheet music notation,etc.

Chapter 3

Requirement Analysis

3.1 INPUT AUDIO FEATURES

The program uses auto note detection which operates based on the variations in the intensity of sound. In almost all the cases there is an sudden increase in the intensity i.e proportional to the square of the values in the input array.Thus for efficient note detection the input audio signal must have considerable amount of variation when a note begins.This can be assured when the sound intensity of noise in the signal is considerably less than the note sound intensity.

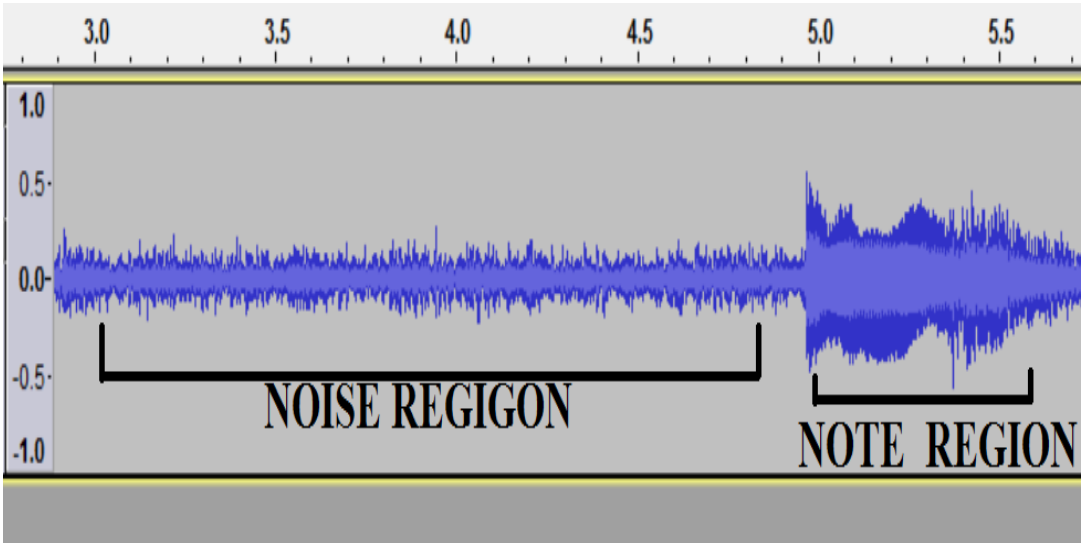


Figure 3.1: Difference in noise and note

The manual tuning requires the user to set a threshold above which the note can be detected.This feature allows him to choose only the required regions to find frequency.It checks if the intensity is greater than the given threshold and plots the graph.The points thus recognised are stored in an array for processing.

Chapter 4

System Testing

WRITE HERE.

4.1 Test Cases and Test Results

Test ID	Test Case File	Output file(NOTES)	Output Sound
T01	happy.wav	notes	out$_h$*appy.wav*
T02	tum.wav	notes	out$_t$*um.wav*
T03	kalhonaho.wav	notes	out$_k$*alho.wav*
T04	manam.wav	notes	out$_m$*anam.wav*

Note: Testing should be performed manually

Chapter 5

Project Planning

5.1 Frequency estimation of a periodic wave

We started by calculating the frequency estimation of a periodic wave.The technique did not require us to use Fourier transform so it made things simple.Our approach to this was the rule that frequency of a peridoc signal is proportional to the number of maxima or minima in a fixed finite time interval.

5.2 Frequency estimation of a non-periodic wave

After we were successful in finding the period of a periodic wave,the next challenge was that the audio signals practically are not exactly periodic because of the minute disturbances in the medium which cause considerable fluctuations in the audio signal in the order of period.So we changed the approach and calculated the period by brute-force method i.e varying period within certain limits and checking which value of period satisfied the required conditions the most.

5.3 Frequency estimation with Fourier Analysis

5.3.1 No Harmonics

The above method was successful but it was slow,then we finally resorted to using the fast Fourier transform provided by MATLAB,our operating environment.But even that was not enough because if T is the period then nT, where n=2,3,4..... can also be the period.So we were not sure if we got the right frequency.So we created an artificial sine wave with the detected frequency and compared with the input.As doubted errors occurred due to above reason.

5.3.2 In presence of Harmonics

Fortunately,the solution to the above problem was the problem itself.If more than one harmonics are present then we can find the fundamental frequency by two methods i.e by finding the least frequency in them or by calculating the difference between two consecutive harmonics.In almost all the cases both give the same result.But in few cases the first methods fails if the least frequency is undetected.If something like that happens the second method ensures the correctness of the frequency.

5.4 Note Detection

Once we were able to calculate the frequency of a note the next challenge was to isolate the note from the audio input.Since our code works only if the input contains a single note and no other signals.Initially we limited the domain of the input and designed a note detection method which works on the fact that a note begins with fast increase in intensity and ends with decrease in intensity below a threshold.The method worked perfectly except when a two notes were very close i.e when a note started before a note ended.So two notes were taken as a single note and thereby resulted in errors while detected frequency in that region.Since the note detection was not the main agenda of our project we learned from external sources to develop the note detection which used Gaussian filters and other advanced functions provided by MATLAB.

Chapter 6

Implementation

The project follows the following implementation.Input audio file which is in the project directory is taken as read and converted into an array with one or two streams.If it contains two streams then they are merged into one by taking average of the two.The array is further any type of processing or analysis.The array stores the audio signal with sampling frequency 44100 samples per second.

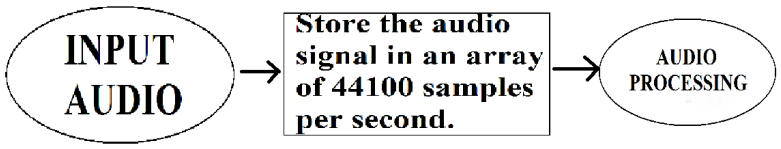


Figure 6.1: INPUT

Audio processing involves two types of note detection.Only the regions where a note is present is taken for processing.Once a note region is isolated we apply Fast Fourier Transform to the region and analyse the data by plotting the power spectrum of the signal.Then frequency is estimated by two different fast algorithms and the appropriate frequency is taken.

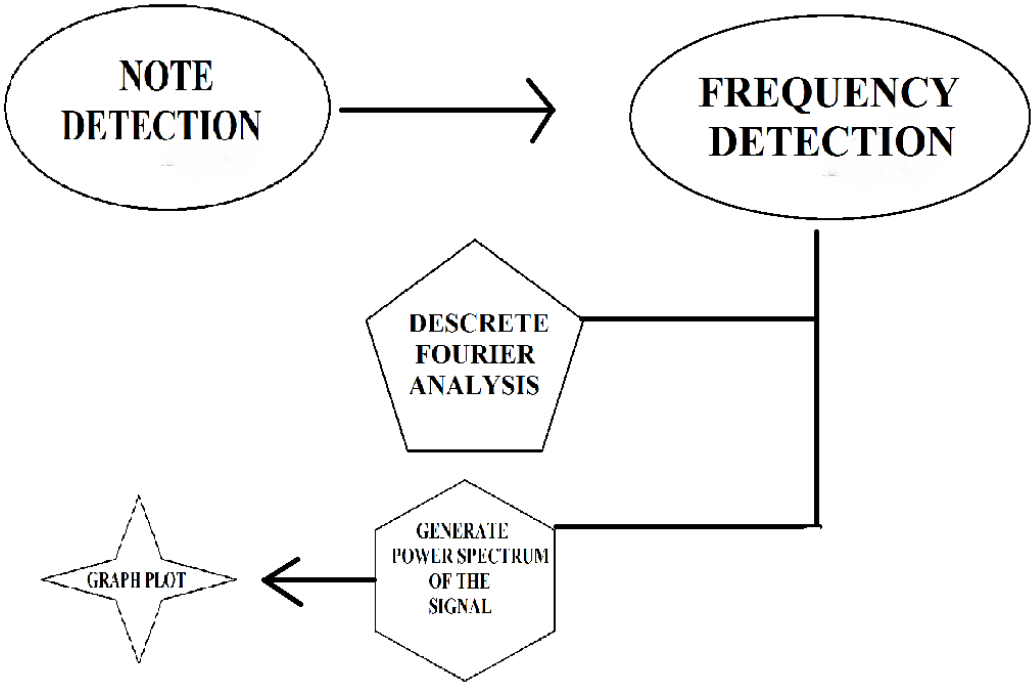


Figure 6.2: Audio Processing

```
  input='filename.wav';
  %takes file name
[z,Fs]=audioread(input);
%reads input audio file
z=z(44100*5:44100*40);
%z=(z(:,1)+z(:,2))/2;
%%activate this line if input contains two streams
 points = timeout2(z,44100,1);
%detects the points where there is a note change
note=1;factor=1;
zz=z*0;
%empty audio array
while(note<length(points))
    fprintf('note number %d \n',note);
    x=z(points(note):points(note+1));
    %audio segment containing the note


     [freq,amp]=compile2(x);
     %store the frequency and amplitude calculated from discrete Fourier
         analysis
  sin_note=artificial(length(x),freq,amp,factor);
%create a exponentially decaying sine wave with detected frequency
  zz(points(note):points(note+1))=sin_note;
%write the empty array with the artificially generated signal

  note=note+1;
end
%Fs=2*Fs;
%%activate this if Fs=22050
```

```
sound(zz,Fs);
%play the generated sound
output=strcat('art_',input);
audiowrite(output,zz,Fs);
%write the artificial to a new audio file
```

The fundamental frequency is calculated directly by using maximum power point in the power spectrum and by taking the difference between consecutive harmonics.The frequency thus obtained is used to calculate the note name of according to the standard sheet music notation.

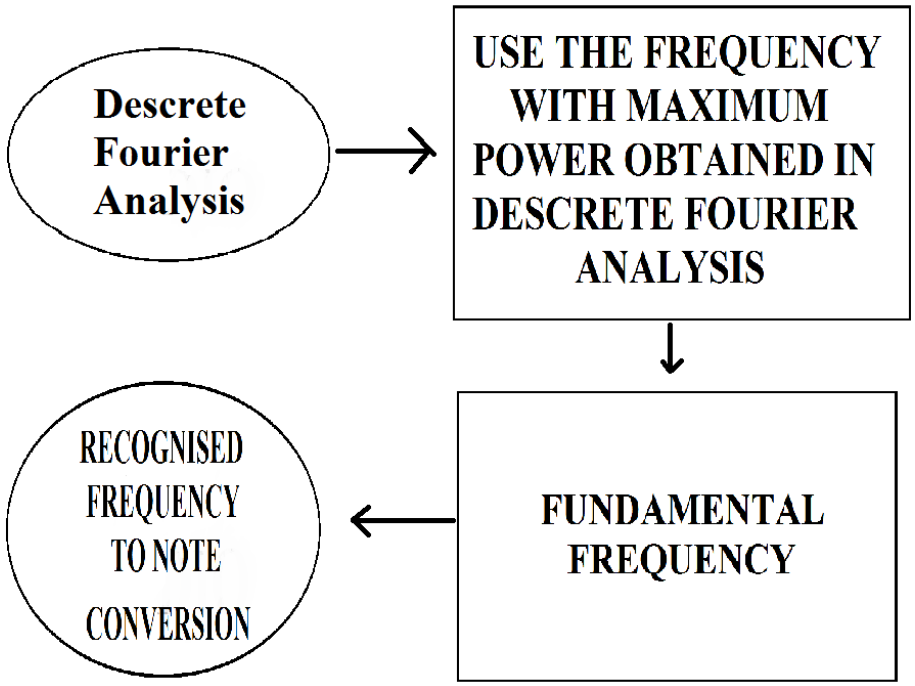


Figure 6.3: Note name from Frequency

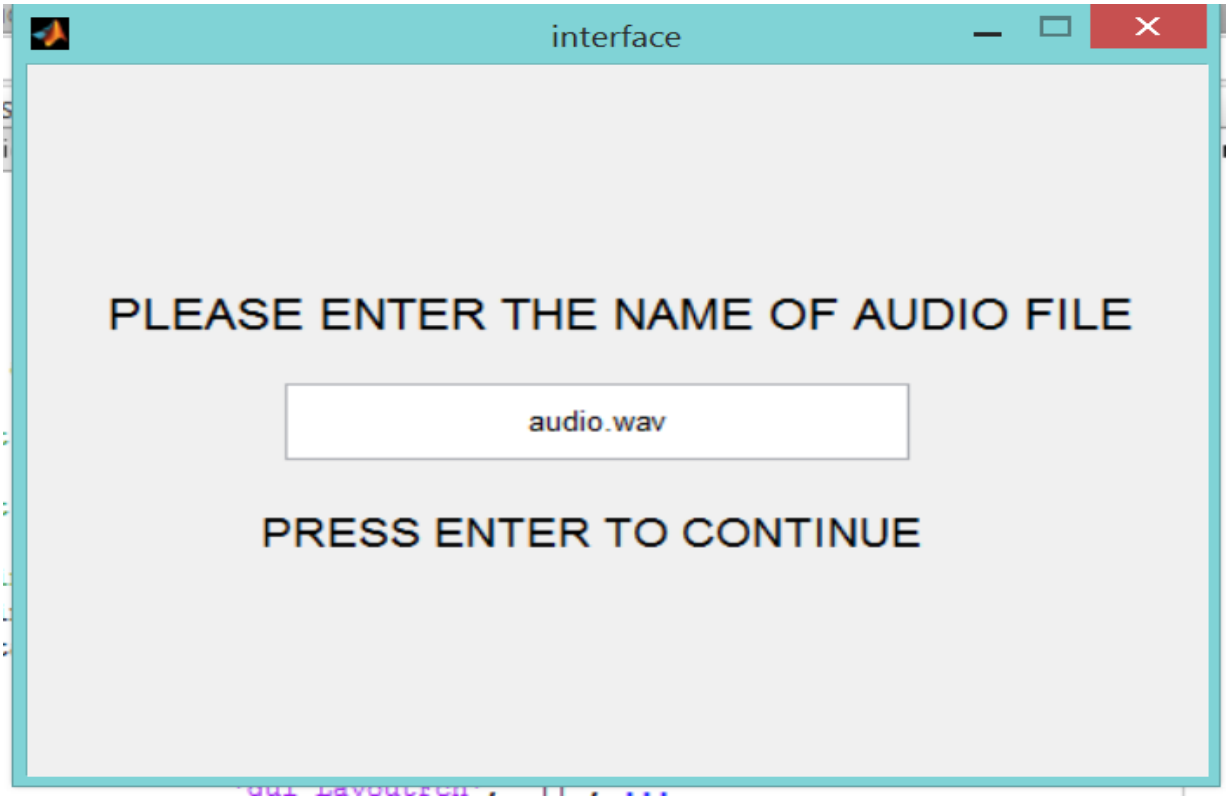
```
t = 0:1/Fs:1; % Time vector of 1 second
nfft = 1024; % Length of FFT
% Take fft, padding with zeros so that length(X)
is equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second half
% FFT is symmetric, throw away second half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
```

```
plot(t,x);
title('Sine Wave Signal');
xlabel('Time (s)');
lbl('A lit d ')
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of a Sine Wave');
xlabel('Frequency (Hz)');
ylabel('Power');
ylabel( Power );
```

Chapter 7

Screenshots of Project

7.1 AUDIO INPUT



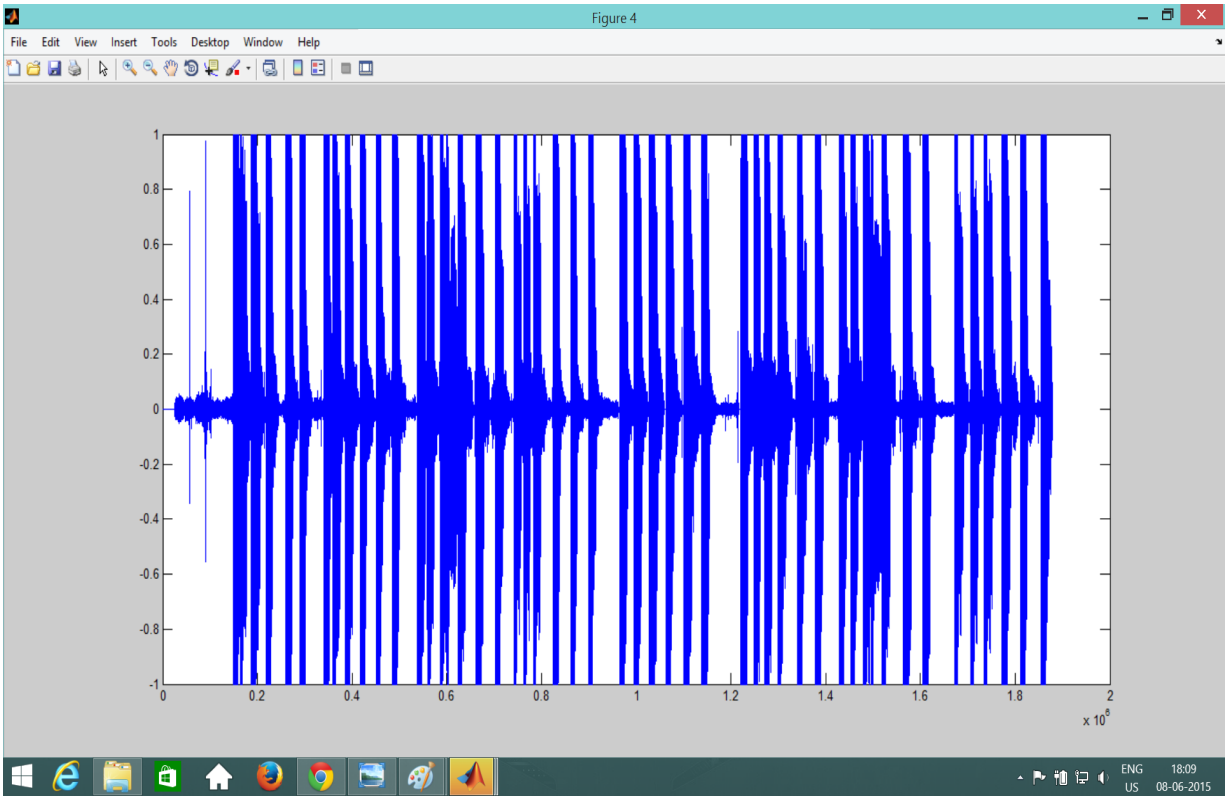


Figure 4
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
x 10^5
ENG US 18:09 08-06-2015

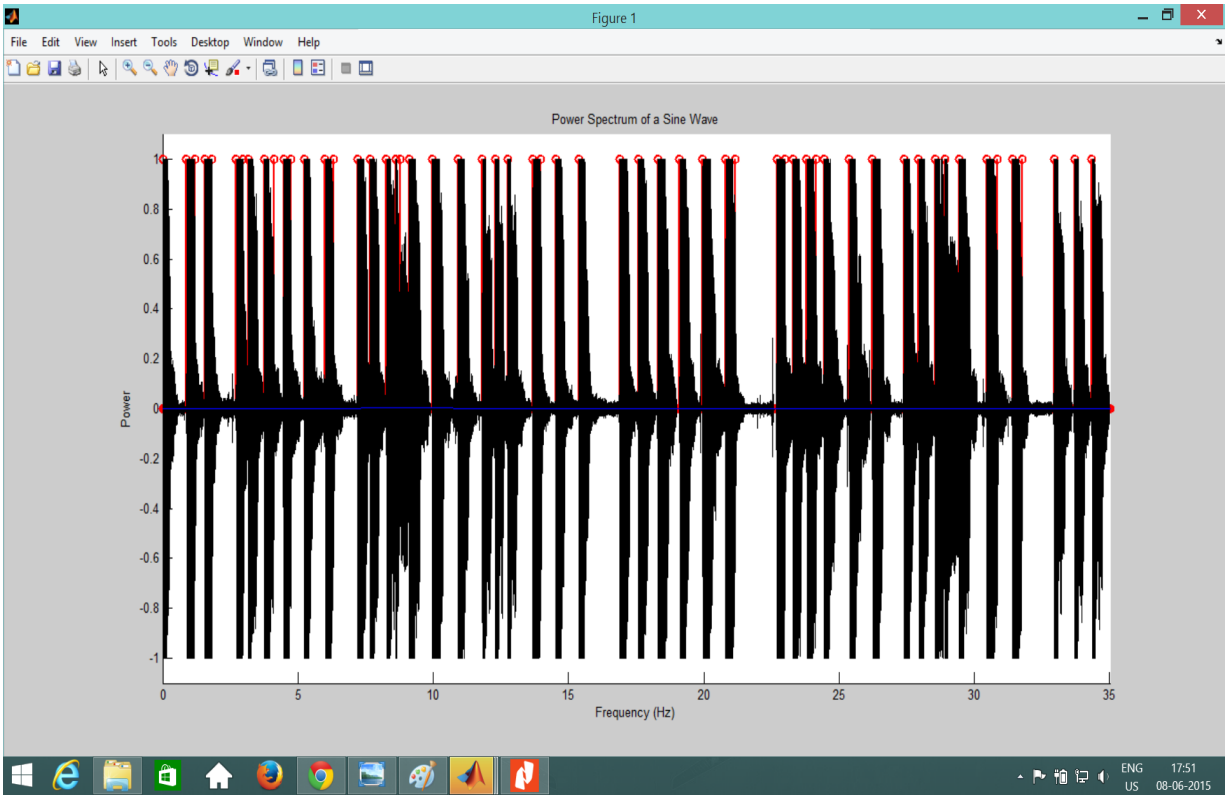
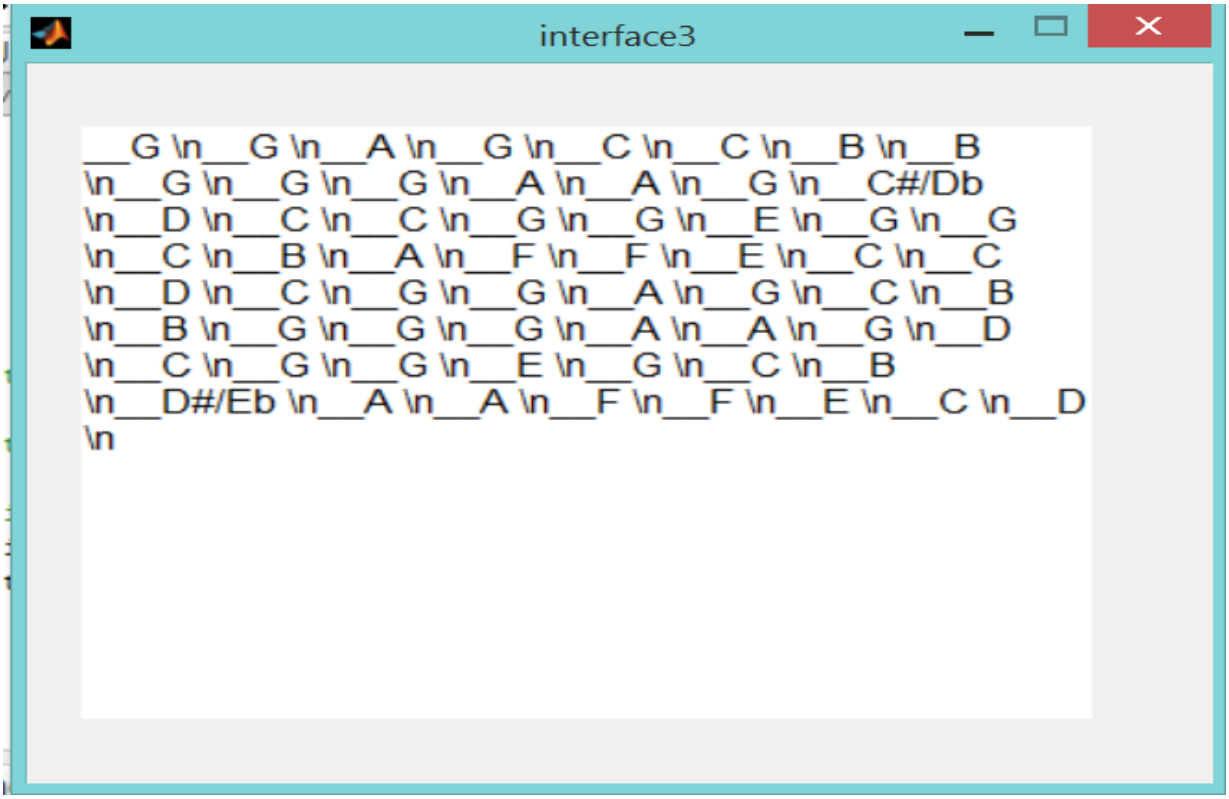


Figure 1
File Edit View Insert Tools Desktop Window Help
Power Spectrum of a Sine Wave
Power
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0 5 10 15 20 25 30 35
Frequency (Hz)
ENG US 17:51 08-06-2015


interface2
SELECT ONE OF THE
FOLLOWING
GRAPHS
NOTES
PLAY DETECTED NOTES


interface3
__G \n__G \n__A \n__G \n__C \n__C \n__B \n__B
\n__G \n__G \n__G \n__A \n__A \n__G \n__C#/Db
\n__D \n__C \n__C \n__G \n__G \n__E \n__G \n__G
\n__C \n__B \n__A \n__F \n__F \n__E \n__C \n__C
\n__D \n__C \n__G \n__G \n__A \n__G \n__C \n__B
\n__B \n__G \n__G \n__G \n__A \n__A \n__G \n__D
\n__C \n__G \n__G \n__E \n__G \n__C \n__B
\n__D#/Eb \n__A \n__A \n__F \n__F \n__E \n__C \n__D
\n

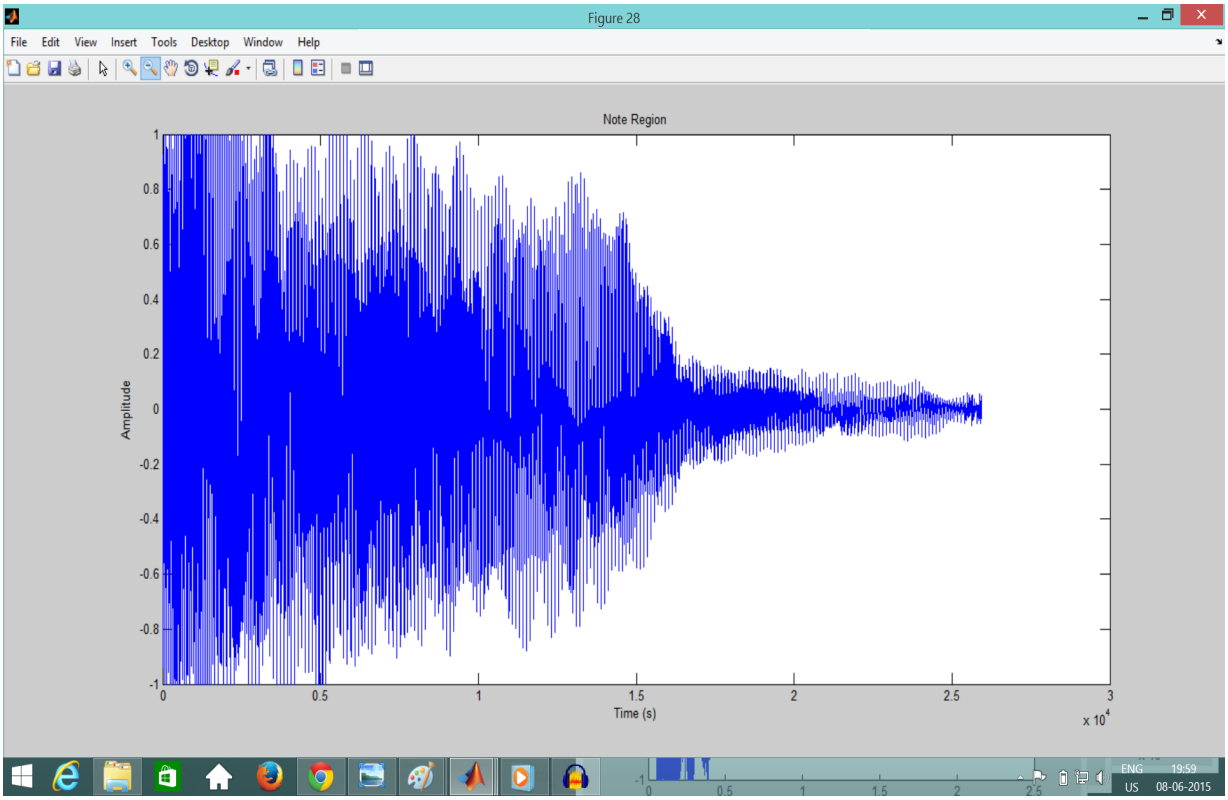


Figure 28
File Edit View Insert Tools Desktop Window Help
Note Region
Amplitude
Time (s)
x 10^4

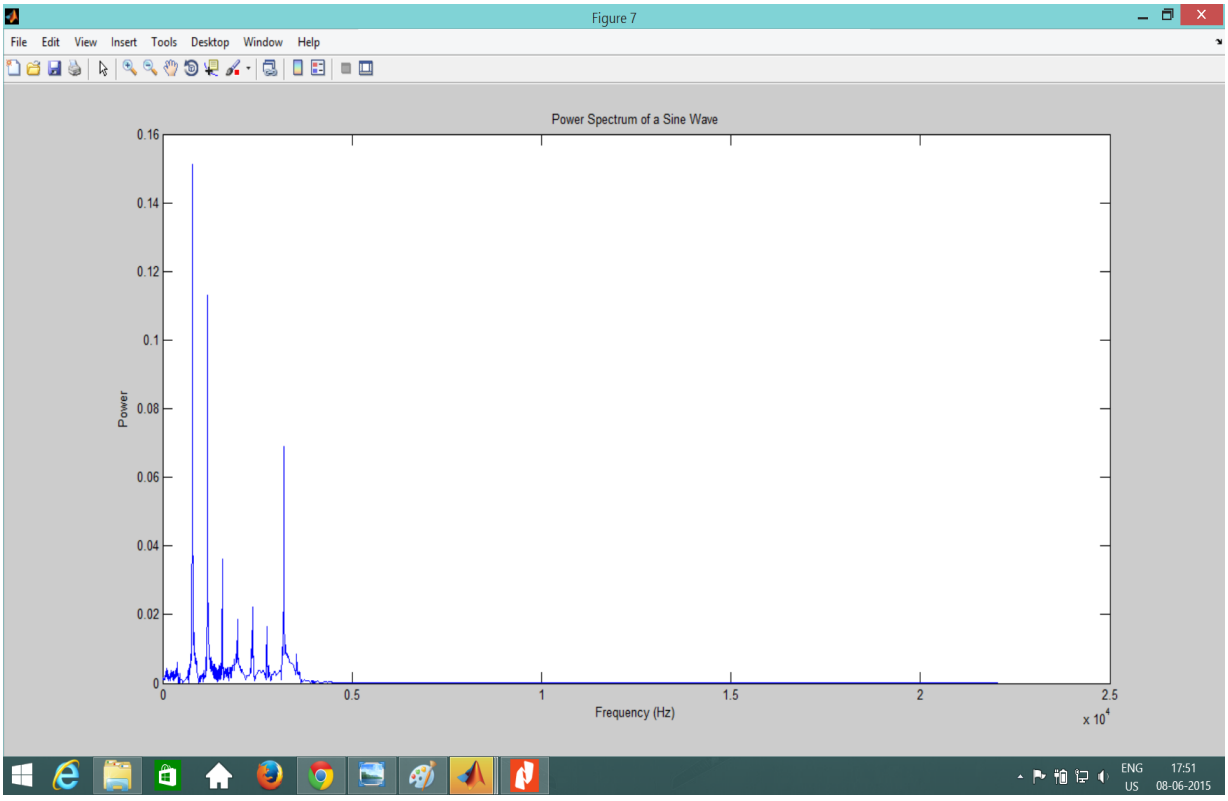


Figure 7
File Edit View Insert Tools Desktop Window Help
Power Spectrum of a Sine Wave
Power
Frequency (Hz)
x 10^4

Chapter 8

Conclusion and Future Scope

8.1 Conclusion

In our project, we designed and implemented an effective and user-friendly frequency estimation system with Fourier Analysis. The target users of the system are not only the people practicing music,but also professional musicians who cannot waste their time figuring out the notes of an audio sample.

8.2 Future Scope

There is still much room for future development that would enhance the system and increase its usage value. The following items are some suggestions:

- Advanced Note Detection:

  There are lot of ways we to improve and customise note detection.Most of them use variations in intensity,which is not the right way because strictly speaking a note is said to change when the frequency of the signal changes.It is not easy to keep track of change in frequency because the change is gradual and hence it is an existing challenge.Moreover the frequency estimation for calculating note detection requires note detection in a crude sense which is paves way for development in this area.

- Non Periodic Signal analysis:

  The process is relatively simple if the signal were sinusoidal or periodic. But the real life musical notes or vocals are approximately periodic and the frequency itself changes with time because a sample may contain more than one

note and that is how music is played.While Fourier Analysis is a nice solution to this problem ,it is not sufficient.Theoretically it may be sufficient but its high level implementation is not as there is resolution and run time limit.There is scope to overcome latter by designing algorithms especially for the purpose of frequency estimation and not focusing on phase detection.

- Multiple Notes at a time:

  Our project assumes that only a single note is played at a time.But that is not it.We can develop it further by using Fourier Analysis again.There are existing algorithms which can isolate multiple notes.After splitting the audio sample into individual notes we can apply our own techniques to find the frequency.

References

[1] https://www.sharelatex.com/learn/Hyperlinks

[2] http://en.wikipedia.org/wiki/Piano_key_frequencies

[3] http://www.tutorialspoint.com/matlab/

[4] http://in.mathworks.com/help/matlab/math/discrete-fourier-transform-dft.html

[5] http://in.mathworks.com/videos/creating-a-gui-with-guide-68979.html