PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Ft. Road, BSK III Stage, Bengaluru – 560 085

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Title: Image Processing and Data Visualization Using MATLAB		
Course code: -UE19CS257B		
Semester: 4th Sem	**Branch: CSE**	**Team Id: 32**
SRN: PES1UG19CS068	**Name: Ankita.V**	
SRN: PES1UG19CS072	**Name: Anurag Khanra**	

PROJECT REPORT

Problem Statement:

Digital tuning of an acoustic guitar.

Objectives:

The objective of this paper is to understand the critical parameters that need to be addressed while designing a digital guitar tuner.

Description:

- A guitar consists of 6 strings namely e, B, G, D, A, E.
- The tuner helps a guitarist tune his guitar to desired notes, the guitarist can play a string and observe the fundamental frequency of note played and the computer determines whether or not it is tuned.
- A user-friendly program is developed using MATLAB to allow any user to easily tune his guitar using the developed program easily.
- The program can be easily changed to cater to other instruments if their frequencies are known.

- Compared to tuning by ear, where a certain amount of guess work is involved in deciding how much to tighten/loosen a string, the MATLAB based tuner is able to give accurate instructions so that tuning can be achieved more quickly.

Explanation of Code

- The tuner is designed using the filtering, measuring and analyzing capabilities of MATLAB.
- When a certain note is played on a guitar it consists of various harmonies and frequencies. The fundamental frequency has to be recovered from the note played.
- The guitar tuner mainly receives user's input from the computer 's built-in microphone.
- The string is plucked once and the sound is sampled into an array variable, the unwanted frequencies are removed by converting it into a domain using the Fast Fourier transform.
- Fundamental frequency for instruments like guitar is the maximum frequency, therefore we simply utilize the max function to locate the maximum peak value from the entire frequency response and extract it as the user's note frequency.
- We also cannot increase the sampling time greatly as the lot of noise might get picked up with the guitar thus, we limit the time to 2 seconds.
- The fundamental frequency of the string is taken as the frequency bin corresponding to the maximum magnitude in the Fast Fourier Transform (FFT) as seen in the graph with a clear peak.
- The frequency of the played note is compared with the standard value that is desired for that particular note (i.e., the chosen string).
- The required instruction (whether to increase or decrease the frequency) is then mentioned with the percentage error in a while loop until the string is tuned or the user may choose to exit.

New Concept Learnt

- Fast Fourier Transform
  -converts time domain into frequency domain
- Audio Recording and Processing
  -records audio as an amplitude vs time and then we use FTT to convert it ti frequency

- Plotting graphs
  -spectrogram is generated, visual representation of the frequencies as it varies with time

Learning Outcome:

With the help of this project, we have expanded our knowledge in MATLAB and are now able to program scripts and functions using the MATLAB development environment. In addition to this we have also learnt about the Fast Fourier Function(FFT), plotting various graphs and the critical parameters to be understood while digitally tuning a guitar.

Code:

```
clearvars;

x=1;

ghighE=1318.1505;
gB=987.7669;
gG=783.9911;
gD=587.3297;
gA=440;
glowE=329.6277;

guitar=[ghighE,gB,gG,gD,gA,glowE];

while x==1

    x=isempty (input ('Press enter to start program or press zero to quit. '));

    if x==1

        guitar_string = input ('Which string are you adjusting?
                                \n1=e\n2=B\n3=G\n4=D\n5=A\n6=E\n');
        required_freq = guitar(guitar_string);

         y = isempty(input('Press enter to record input signal
                           or 0 to start over. '));

        while y==1

            Fs=44100;

            RecordObject = audiorecorder(Fs,16,1,-1);  %#ok<*TNMLP>
            record(RecordObject,2);
            pause(3);

            input = getaudiodata(RecordObject,'double');

            disp('This is how the input signal sounds. ');
            sound (input,Fs);

            inputFFT = fft(input)/size(input,1);
            K=0:1:(Fs/2-1);
```

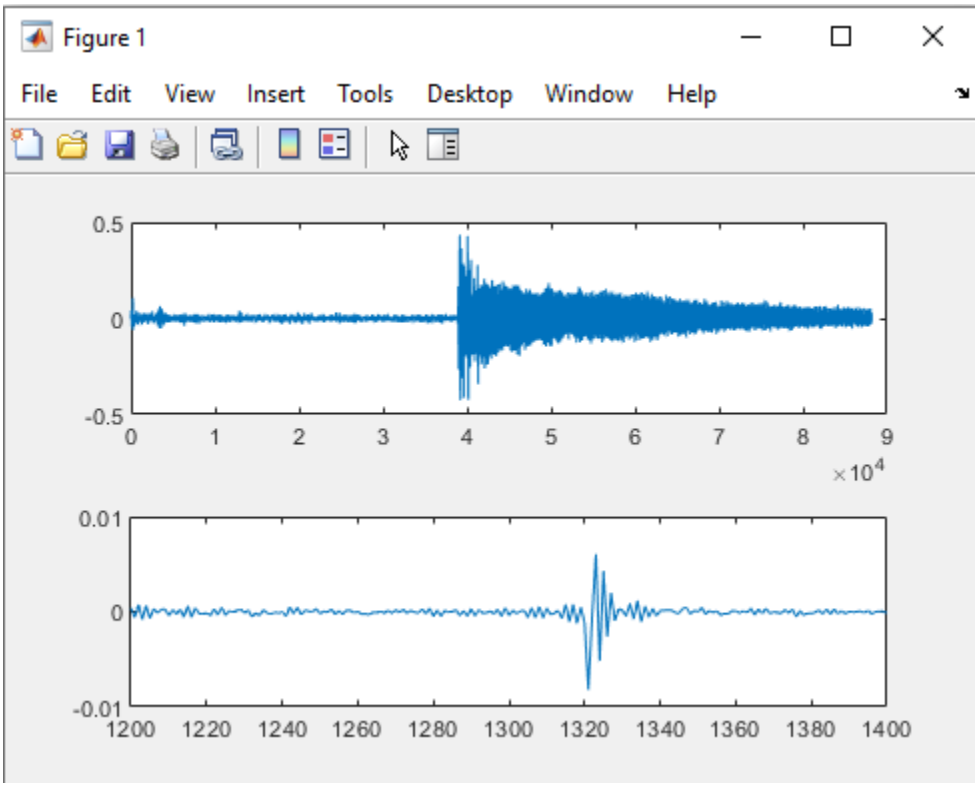
```
if guitar_string==6
    for i=400:size(inputFFT,1)
        inputFFT(i,1) = 0;
    end
else
    for i=1500:size(inputFFT,1)
        inputFFT(i,1) = 0;
    end
end

subplot(2,1,1);
plot(input);
subplot(2,1,2);
plot(K,2*real(inputFFT(1:Fs/2)));

if guitar_string==6
    axis([ 200 500 -0.01 0.01])
    for i=400:size(inputFFT,1)
        inputFFT(i,1) = 0;
    end

elseif guitar_string==5
    axis([ 400 550 -0.01 0.01])
    for i=550:size(inputFFT,1)
        inputFFT(i,1) = 0;
    end

elseif guitar_string==4
    axis([ 450 700 -0.01 0.01])
    for i=700:size(inputFFT,1)
        inputFFT(i,1) = 0;
    end

elseif guitar_string==3
    axis([ 650 850 -0.01 0.01])
    for i=900:size(inputFFT,1)
        inputFFT(i,1) = 0;
    end

elseif guitar_string==2
    axis([ 800 1100 -0.01 0.01])
    for i=1200:size(inputFFT,1)
        inputFFT(i,1) = 0;
```

```
                end

            elseif guitar_string==1
                axis([ 1200 1400 -0.01 0.01])
                for i=1500:size(inputFFT,1)
                    inputFFT(i,1) = 0;
                end
            end


        current_freq = K(find(inputFFT == max(inputFFT)));
        z=((current_freq-required_freq)/required_freq)*100;

        fprintf('Required Frequency is %d Hz.\n', required_freq);

        if (1.0015*required_freq) > current_freq &&
                            (0.9985*required_freq) < current_freq
                fprintf('You are in tune.\nThe frequency of the input
                         signal is %d Hz.\n', current_freq);
                fprintf('Percent Error %d%% \n',z)

        elseif required_freq > current_freq
                fprintf('Input frequency should be increased.\nThe frequency
                         of the input signal is %d Hz.\n', current_freq);
                fprintf('Percent Error %d%% \n',z)

        elseif required_freq < current_freq
                fprintf('Input frequency should be decreased.\nThe frequency
                         of the input signal is %d Hz.\n', current_freq);
                fprintf('Percent Error %d%% \n',z)
        end

        y = isempty(input('Press enter if you want to rerecord the same
                           string or any number to start over.\n'));

            if y==1
                continue;
            else
                x=1;
            end
        end
    end
end
```

Output Screenshots:

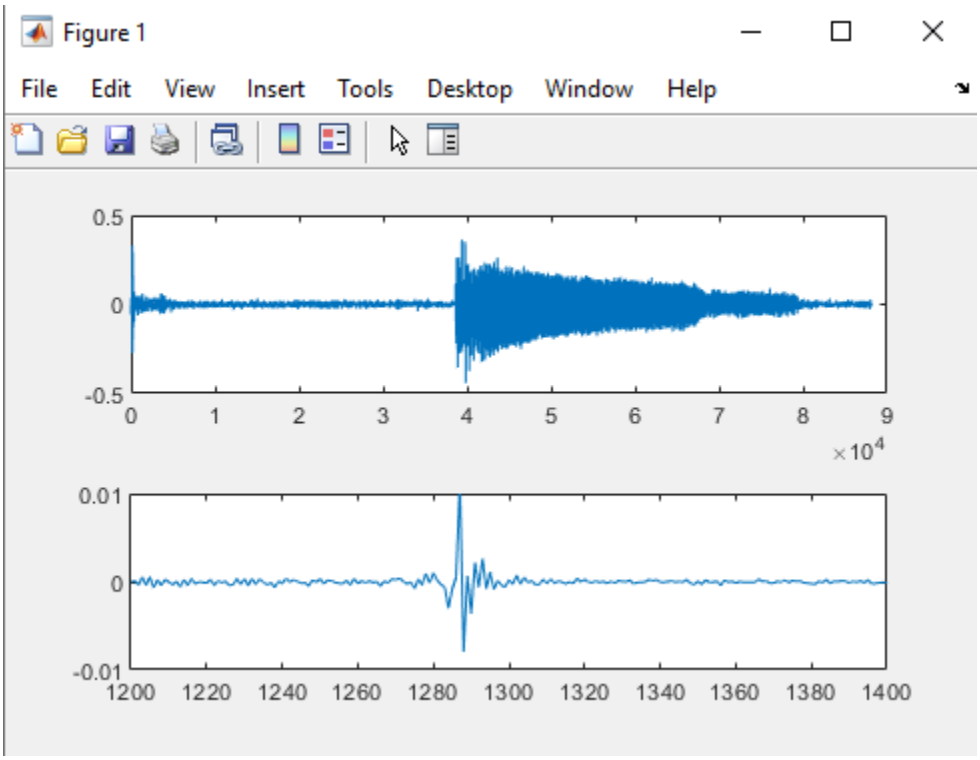
Above required Frequency

```
>> Guitar_Tuner
Press enter to start program or press zero to quit.
Which string are you adjusting?
1=e
2=B
3=G
4=D
5=A
6=E
1
Press enter to record input signal or 0 to start over.
This is how the input signal sounds.
Required Frequency is 6.590752e+02 Hz.
Input frequency should be decreased.
The frequency of the input signal is 662 Hz.
Percent Error 4.437657e-01%
```



Below required Frequency

```
>> Guitar_Tuner
Press enter to start program or press zero to quit.
Which string are you adjusting?
1=e
2=B
3=G
4=D
5=A
6=E
1
Press enter to record input signal or 0 to start over.
This is how the input signal sounds.
Required Frequency is 6.590752e+02 Hz.
Input frequency should be increased.
The frequency of the input signal is 644 Hz.
Percent Error -2.287334e+00%
```



Guitar in tune:

```
>> Guitar_Tuner
Press enter to start program or press zero to quit.
Which string are you adjusting?
1=e
2=B
3=G
4=D
5=A
6=E
1
Press enter to record input signal or 0 to start over.
This is how the input signal sounds.
Required Frequency is 6.590752e+02 Hz.
You are in tune.
The frequency of the input signal is 660 Hz.
Percent Error 1.403102e-01%
```