GUITAR TUNING USING MATLAB

by

VIGHNESH M 18BEC1223
SUSHIL BALA 18BEC1227
PRAMOD K 18BEC1257
ANDREW JOHN J 18BEC1278

A project report submitted to

Dr. Sathiya Narayanan

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfilment of the requirements for the course of

ECE1004 – SIGNALS AND SYSTEMS

in

B.Tech. ELECTRONICS AND COMMUNICATION ENGINEERING

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

VIT, CHENNAI CAMPUS

Vandalur-Kelambakkam Road

Chennai – 600127

November 2019

BONAFIDE CERTIFICATE

Certified that this project report entitled "**GUITAR TUNING USING MATLAB"** is a bonafide work of **SUSHIL BALA(18BEC1227), VIGHNESH M(18BEC1223), PRAMOD K (18BEC1257)** and **ANDREW JOHN J (18BEC1278)** who carried out the Project work under my supervision and guidance.

Dr. Sathiya Narayanan

Professor

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai-600127.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Sathiya Narayanan,** Professor, School of Electronics Engineering, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. A. Sivasubramanian,** Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her unstinting support.

We express our thanks to our Programme Chair **Dr. Vetrivelan.P** and Co-Chair **Dr. D. Thiripurasundari** for their support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

SUSHIL .B **VIGHNESH.M** **ANDREW** **PRAMOD**

INDEX

1.What Is a Guitar Tuner?

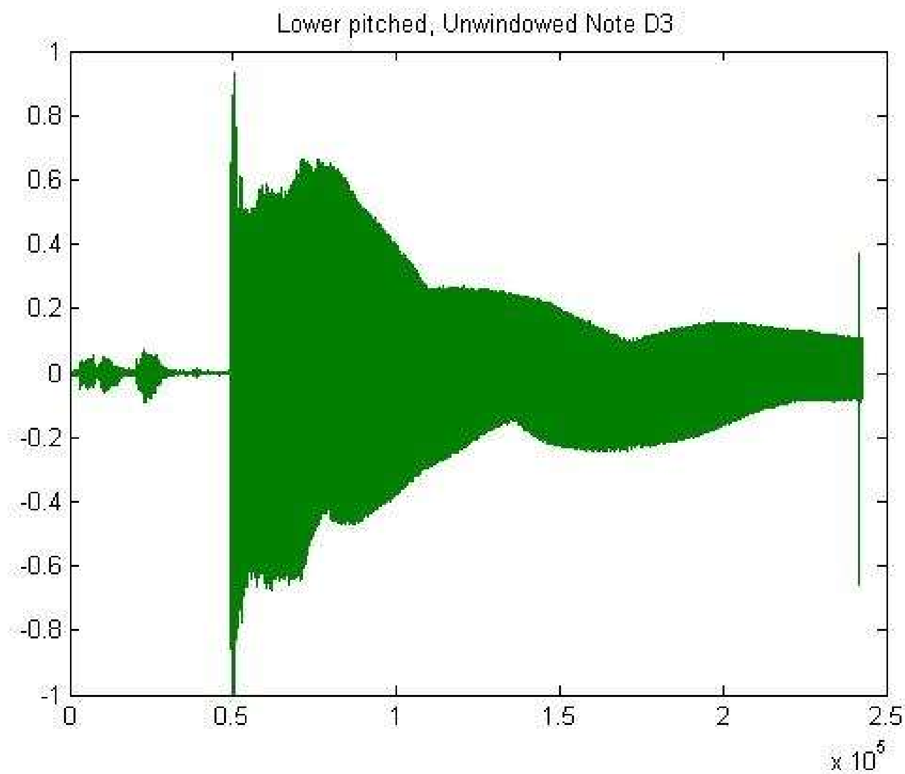
A guitar tuner is a device that measures the frequencies produced by vibrating strings on an electric guitar or an acoustic guitar. It then aligns those measurements with notes in a scale. If the frequencies match a particular note, the tuner will display the name of that note on an LED display.

There are also bass tuners specially made for bass guitars and string basses, but in a pinch, a guitar tuner will often work for both guitars and bass instruments.

2. Detailed DSP Techniques

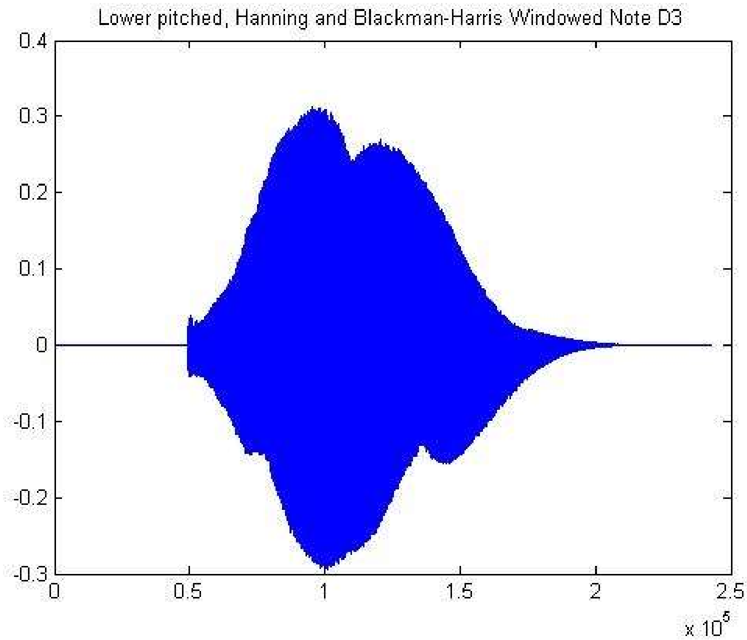
1. Collected seven (7) data sets for each string for analysis and real-time signal for the actual signal; analyzed signals include:

a. In-tune string
b. Lowest, lower, low pitch
c. Highest, higher, high pitch

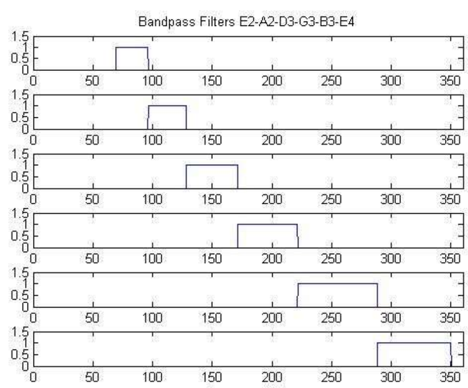


2. Windowed the signal using two windows: (tested multiple other combinations of windows, and these two offered a better result for amplitude distinction and narrower peaks - for clearer analysis)

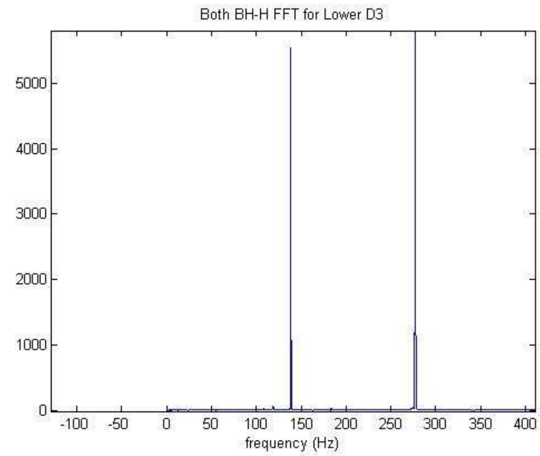
.
a. Blackman-Harris 4-term
b. Hanning



3. With the information of different fundamental frequencies from http://en.wikipedia.org/wiki/Piano_key_frequencies , determined a boundary by taking the average of two notes' fundamental frequency and let that be their bound.
   For the lowest and highest pitched strings (E2, E4 respectively), a lower and upper bound, respectively for E2 and E4, were chosen assuming the individual trying to tune their guitars don't have their strings too loose or too tight.
4. Manually implemented a bandpass filter using ones and zeros with the information of each note's bounds.



5. Performed the Fast-Fourier Transform (FFT) on the given signal, with the main focus on the very first peak of each plot.

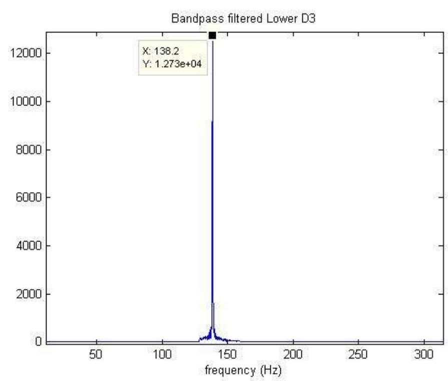


6. Search for the first peak in the FFT plot that corresponds to pitch/fundamental frequency with the use of the implemented bandpass filters. The algorithm goes as follows:

. Use the bandpass filter for the lowest pitched string (E2) first

a. Order: E2 -> A2 -> D3 -> G3 -> B3 -> E4

b. Multiply the FFT data with the bandpass filter



c. A threshold for that certain string is set (amplitude 500 for E2, and 1000 for the rest). If the maximum of the result of part (b) exceeds the threshold, then recognize the signal as the note/string that corresponds to the bandpass filter being used

d. If the maximum of the result from part (b) doesn't meet the threshold condition, then move on to the next note and use its corresponding bandpass filter. Then go back to part (b).

e. If part (d) doesn't apply, take the location of the peak amplitude in that range and that is the fundamental frequency of the recorded signal.

f. Determine if this location is above or below the corresponding note's fundamental frequency. If it is above, command to tune down; below, command to tune up.

g. The condition for being in tune is given a padding of around 1.5 Hz. If the tuning is within 1.5 Hz of the in-tune fundamental frequency of the note, then it is considered in-tune.

3. Theory

Window Method is used to obtain Finite Impulse Response (FIR) from systems that are noncausal and infinitely long. Filter design begins with desired frequency response represented as

$$H_d\,(e^{jw}) = \sum_{n=-\infty}^{\infty} h_d[n]e^{-jwn} \qquad (1)$$

where is the corresponding impulse response sequence, which can be expressed as

$$h_d[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{jw})e^{jwn}dw \qquad (2)$$

Equation 1 is like Fourier series representation of periodic frequency response Hd (ejw), with $h_d[n]$ acting as the coefficients.[1]

Truncating $h_d[n]$ is another way to obtain a causal FIR filter.

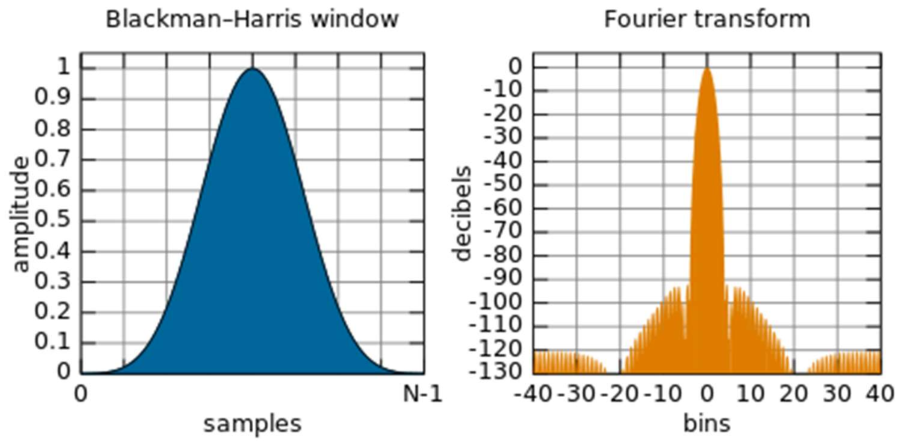
$$h[n] = \begin{cases} h_d[n], & 0 \le n \le M \\ 0 & otherwise \end{cases}$$

therefore, h[n] is equal to the product of the desired impulse response and a finite-duration "window"[1]

$$h[n] = h_d[n]w[n]$$

-Blackman-Harris window- returns an n-point, minimum 4-term Blackman-Harris window. The FFT of the Blackman-Harris window has sidelobes significantly lower in amplitude than the amplitude of the main lobe.[5]

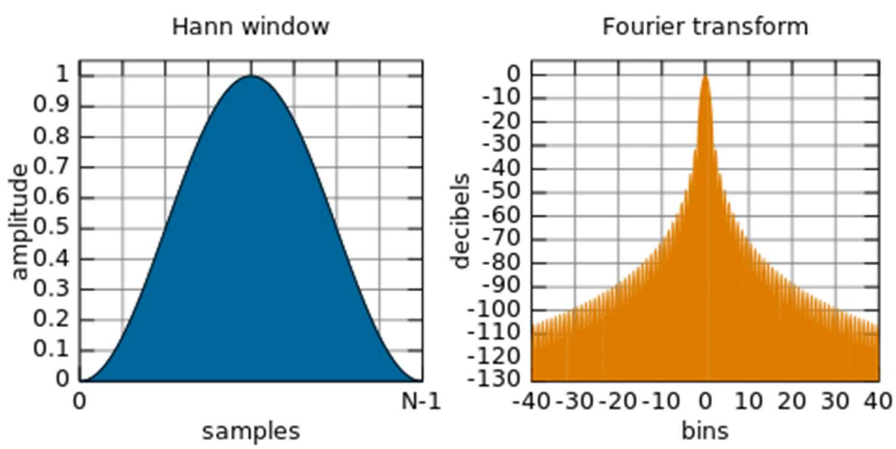
$$W(n) = 0.35875 - 0.48829\left[\cos\left(\frac{2\pi n}{M-1}\right)\right] + 0.14128\left[\cos\left(\frac{4\pi n}{M-1}\right)\right] - 0.01168\left[\cos\left(\frac{6\pi n}{M-1}\right)\right],$$
$$0 \le n \le M$$



Niemitalo, Olli. Window function and frequency response - Blackman

-Hanning Window- Associated with Julius von Hann, an Austrian meteorologist. Window length is M+1. The FFT of the Hanning window has sidelobes significantly lower in amplitude than the amplitude of the main lobe.

$$w[n] = \begin{cases} 0.5 - 0.5\left(\cos\left(\frac{2\pi n}{M}\right)\right), & 0 \le n \le M \\ 0, & otherwise \end{cases}$$



Niemitalo, Olli. Window function and frequency response - Hann
https://en.wikipedia.org/wiki/File:Window_function_and_frequency_response_-_Hann.svg

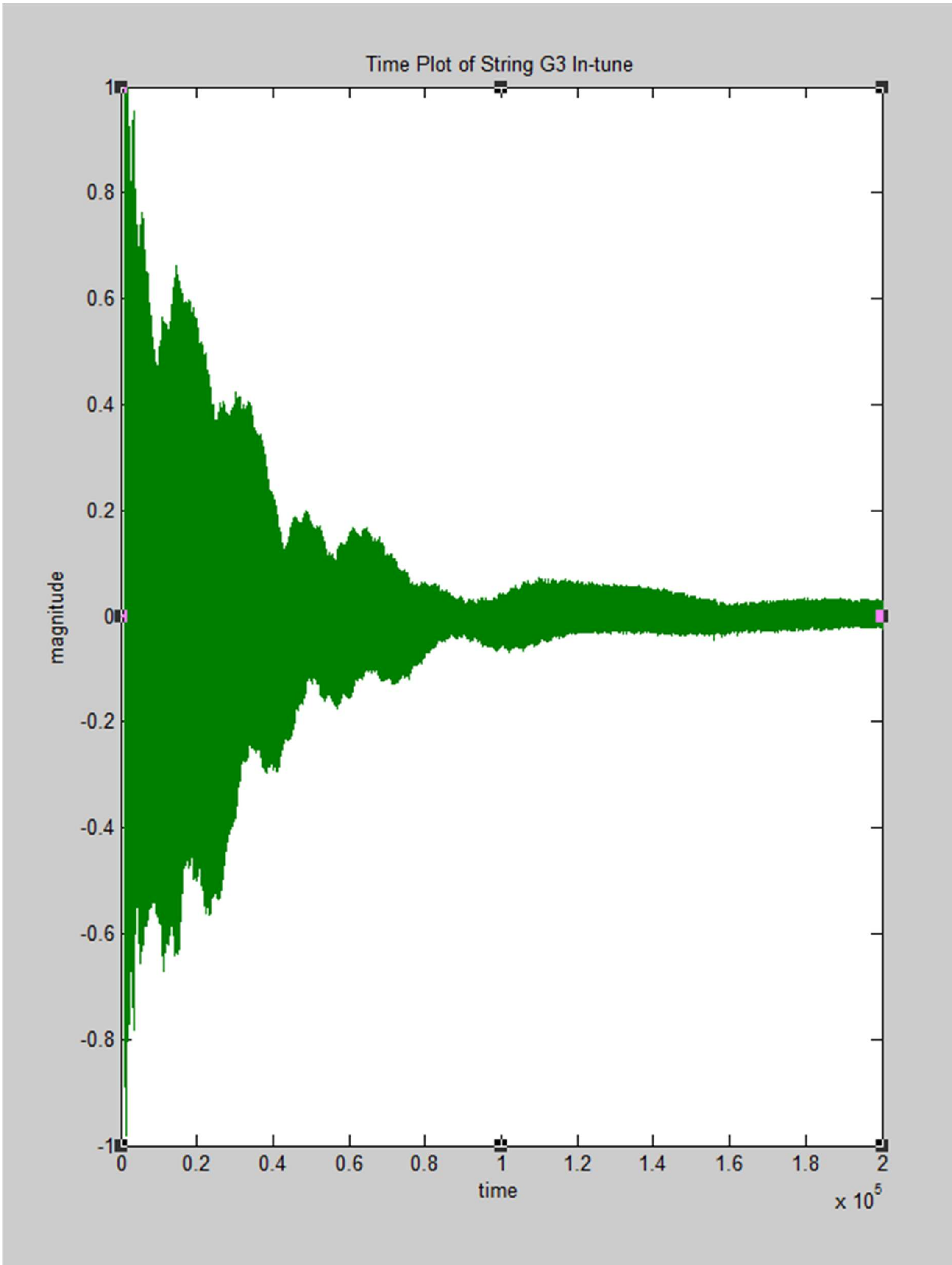
FFT - Fast Fourier Transform is an algorithm to compute the Discrete Fourier Transform (DFT) of a finite-duration sequence. The DFT is a transform between discrete time and discrete frequency. The DFT is obtained by decomposing a sequence of values into components of different frequencies [

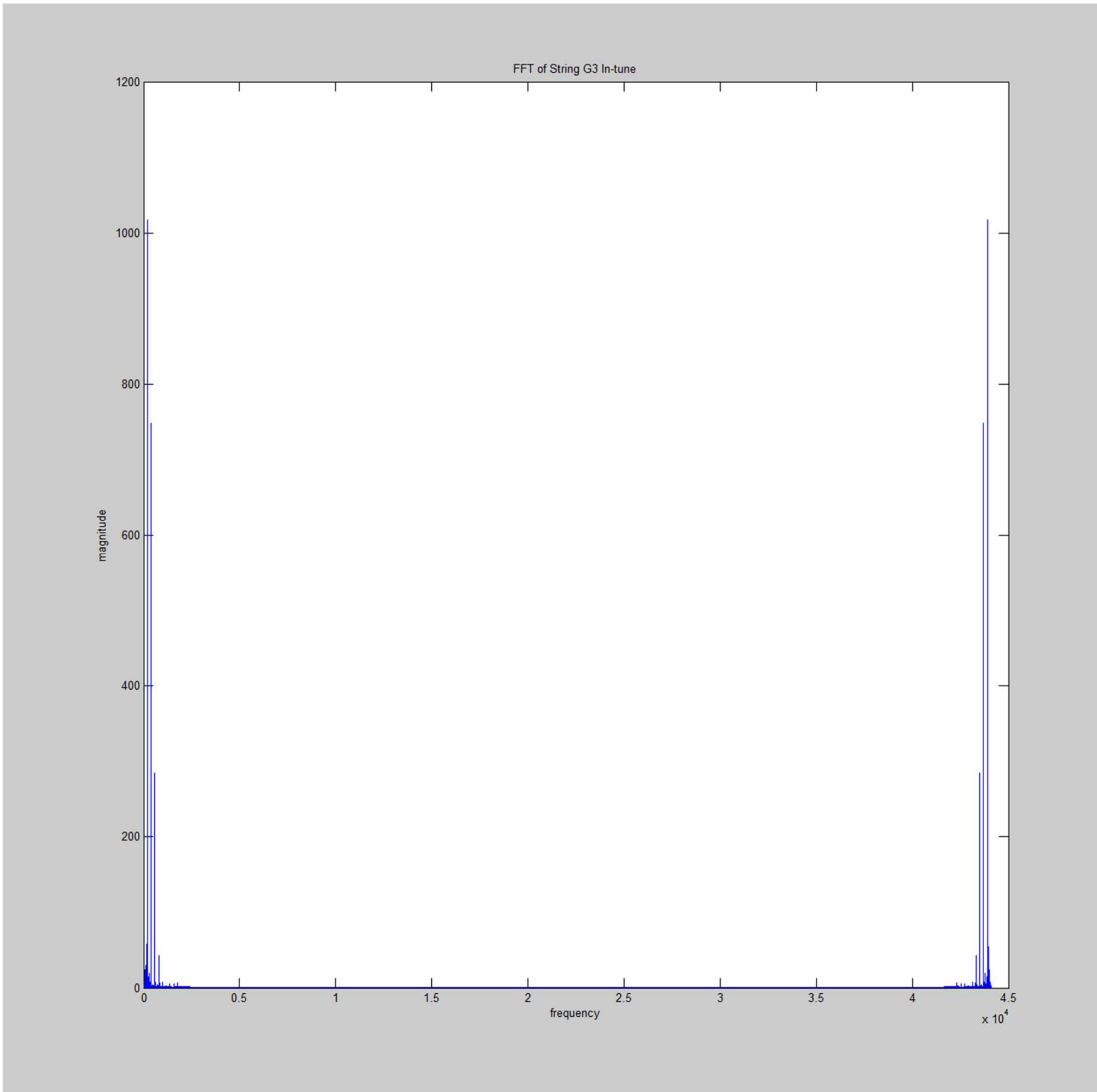
$$DFT\ Analysis\ Equation:\ X[k] = \sum_{k=0}^{N-1} x[n] W_N{}^{kn}, \quad 0 \le k \le N-1$$

$$DFT\ Synthesis\ Equation:\ x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N{}^{-kn}, \quad 0 \le k \le N-1$$

MATLAB computes the FFT based on a library called FFTW. The library uses the Cooley-Tukey algorithm. FFT algorithms have two classes: decimation in time, and decimation in frequency. The Cooley-Tukey FFT algorithm rearranges the input elements in reversed order bit, next builds the output transform in which is the decimation in time. The main idea is to deconstruct a transform of length N onto two transforms of length N/2 using the identity

$$\begin{aligned} \sum_{n=0}^{N-1} a_n\, e^{-\frac{2\pi k}{N}} &= \sum_{n=0}^{\frac{N}{2}-1} a_{2n}\, e^{-\frac{(2n)2j\pi k}{N}} + \sum_{n=0}^{\frac{N}{2}-1} a_{2n}\, e^{-\frac{(2n+1)2j\pi k}{N}} \\ &= \sum_{n=0}^{\frac{N}{2}-1} a_n^{even}\, e^{-\frac{2nj\pi k}{\frac{N}{2}}} + e^{-\frac{2\pi jk}{N}} \sum_{n=0}^{\frac{N}{2}-1} a_n^{odd}\, e^{-\frac{2nj\pi k}{\frac{N}{2}}} \end{aligned}$$


Time Plot of String G3 In-tune
magnitude
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
time
x 10^5



Time plot of string G3 in-tune and FFT of string G3 in-tune.

PROGRAM CODE:

```
%Guitar Tuner Program
%Sushil Bala
% Vighnesh M
% Pramod K
% Andrew John

% close all;
clear all;
clc;
z=input('Enter the file name');
tic;
while toc<1

[notesig,fs] = audioread(z.string);

n = length(notesig)-1;
f = 0:fs/n:fs;
bh = blackmanharris(n+1); %Blackman-Harris Window
h = hanning(n+1); %Hanning Window
%Windowing
notesig = notesig(:,1).*bh.*h;
%plot(notesig);title('Incoming Signal');

%fundamental frequency
E2f = 82.4069;
A2f = 110;
D3f = 146.832;
G3f = 195.998;
B3f = 246.942;
```

```
E4f = 329.628;

%bounds for frequency filters
E2A2 = mean([E2f A2f]);
A2D3 = mean([A2f D3f]);
D3G3 = mean([D3f G3f]);
G3B3 = mean([G3f B3f]);
B3E4 = mean([B3f E4f]);

%frequency filters
E2filter = (1.*(f<E2A2).*(f>69))';
A2filter = (1.*(f>E2A2).*(f<A2D3))';
D3filter = (1.*(f>A2D3).*(f<D3G3))';
G3filter = (1.*(f>D3G3).*(f<G3B3))';
B3filter = (1.*(f>G3B3).*(f<B3E4))';
E4filter = (1.*(f>B3E4).*(f<350))';

%FFT
y = abs(fft(notesig));
sigFFT = y(:,1); %extract one column
if(max(E2filter.*sigFFT)> 500)
disp('E2')
filtSig = E2filter.*sigFFT;
[fundAmp, sample_loc] = max(filtSig);
if(sample_loc*fs/n < E2f-1.5)
disp('Tune up');
disp('Difference:');
disp(E2f-1.5-(sample_loc*fs/n));
elseif(E2f+1.5 < sample_loc*fs/n)
disp('Tune down');
disp('Difference:');
disp((sample_loc*fs/n)-E2f+1.5);
```

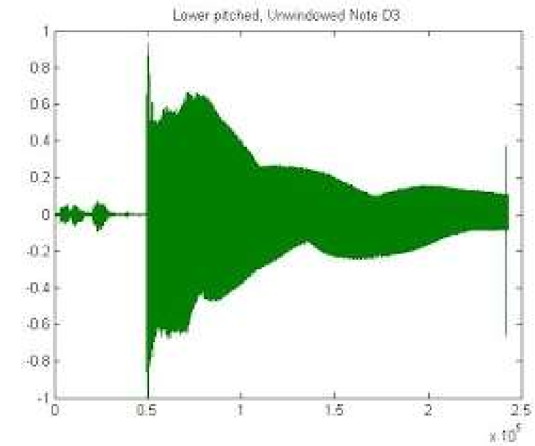
```
else
disp('Nice, In Tune Bro');
end
elseif(max(A2filter.*sigFFT)> 1000)
disp('A2');
filtSig = A2filter.*sigFFT;
[fundAmp, sample_loc] = max(filtSig);
if(sample_loc*fs/n < A2f-1.5)
disp('Tune up');
disp('Difference:');
disp(A2f-1.5-(sample_loc*fs/n));
elseif(A2f+1.5 < sample_loc*fs/n)
disp('Tune down');
disp('Difference:');
disp((sample_loc*fs/n)-A2f+1.5);
else
disp('Nice, In Tune Bro');
end
elseif(max(D3filter.*sigFFT)> 1000)
disp('D3');
filtSig = D3filter.*sigFFT;
[fundAmp, sample_loc] = max(filtSig);
if(sample_loc*fs/n < D3f-1.5)
disp('Tune up');
disp('Difference:');
disp(D3f-1.5-(sample_loc*fs/n));
elseif(D3f+1.5 < sample_loc*fs/n)
disp('Tune down');
disp('Difference:');
disp((sample_loc*fs/n)-D3f+1.5);
else
disp('Nice, In Tune Bro');
```

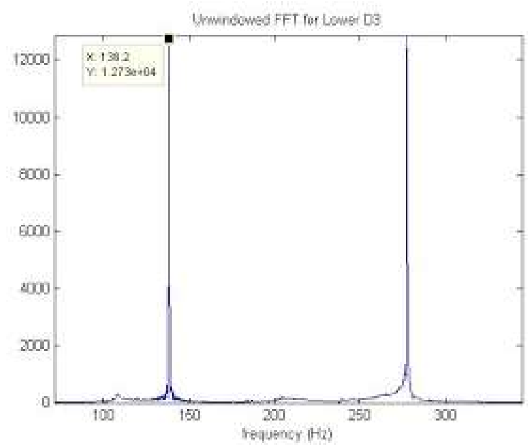
```
end
elseif(max(G3filter.*sigFFT)> 1000)
disp('G3');
filtSig = G3filter.*sigFFT;
[fundAmp, sample_loc] = max(filtSig);
if(sample_loc*fs/n < G3f-1.5)
disp('Tune up');
disp('Difference:');
disp(G3f-1.5-(sample_loc*fs/n));
elseif(G3f+1.5 < sample_loc*fs/n)
disp('Tune down');
disp('Difference:');
disp((sample_loc*fs/n)-G3f+1.5);
else
disp('Nice, In Tune Bro');
end
elseif(max(B3filter.*sigFFT)> 1000)
disp('B3');
filtSig = B3filter.*sigFFT;
[fundAmp, sample_loc] = max(filtSig);
if(sample_loc*fs/n < B3f-1.5)
disp('Tune up');
disp('Difference:');
disp(B3f-1.5-(sample_loc*fs/n));
elseif(B3f+1.5 < sample_loc*fs/n)
disp('Tune down');
disp('Difference:');
disp((sample_loc*fs/n)-B3f+1.5);
else
disp('Nice, In Tune Bro');
end
elseif(max(E4filter.*sigFFT)> 1000)
```

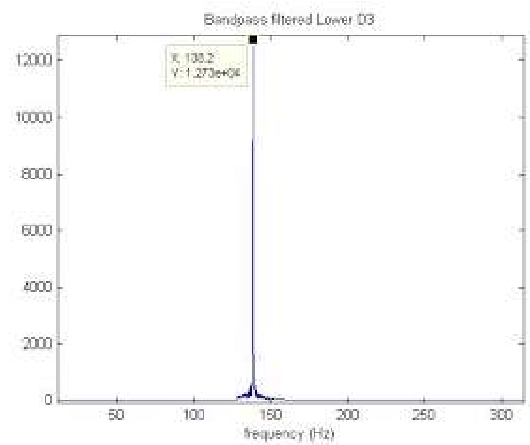
```
disp('E4');
filtSig = E4filter.*sigFFT;
[fundAmp, sample_loc] = max(filtSig);
if(sample_loc*fs/n < E4f-1.5)
disp('Tune up');
disp('Difference:');
disp(E4f-1.5-(sample_loc*fs/n));
elseif(E4f+1.5 < sample_loc*fs/n)
disp('Tune down');
disp('Difference:');
disp((sample_loc*fs/n)-E4f+1.5);
else
disp('Nice, In Tune Bro');
end
end
end
```

Results

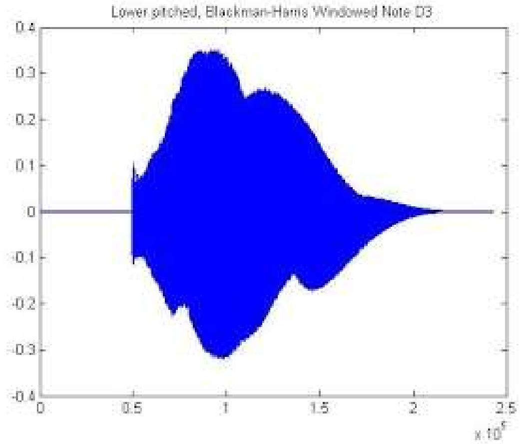
Lower pitched unwindowed D3 signal; Signal, FFT, Filtered FFT

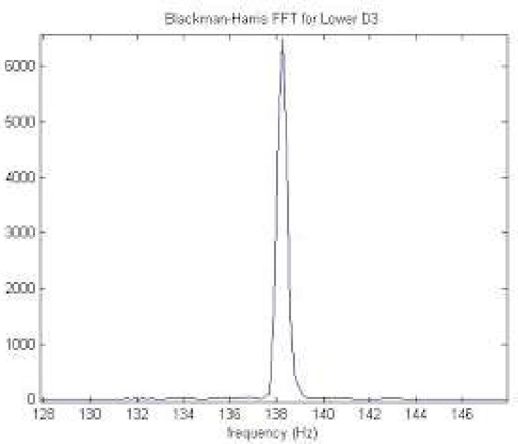




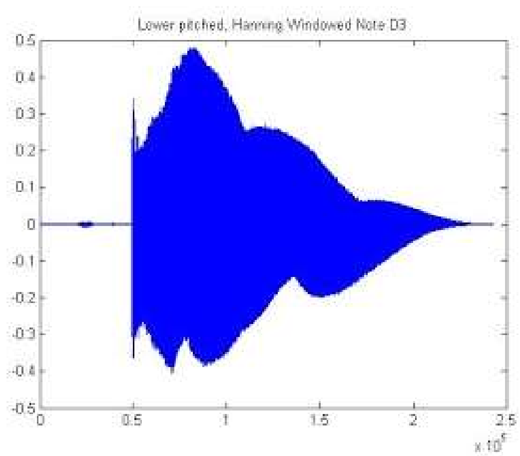


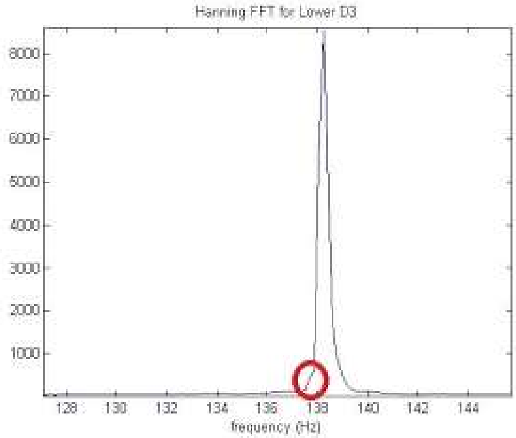
Lower pitched Blackman-Harris windowed D3 signal; Signal, FFT





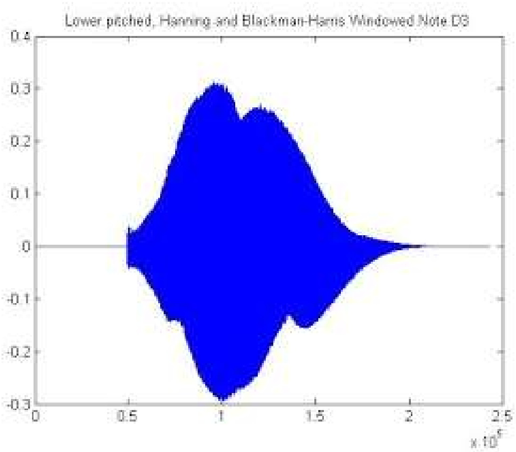
Lower pitched Hanning windowed D3 signal; Signal, FFT

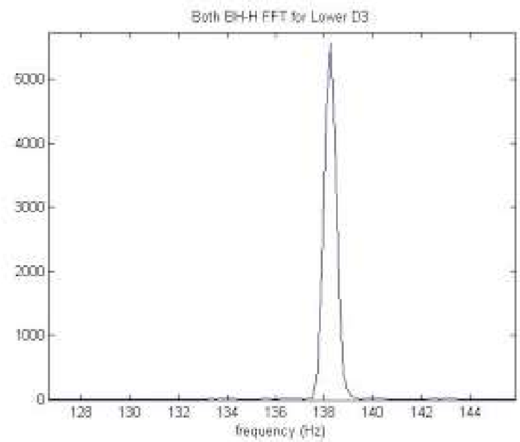




****Looking at the images for the Blackman-Harris and Hanning windowed signals, the Blackman-Harris shows a smoother peak. The red circled spot isn't showing for the Blackman-Harris, but the peak is a little wider. For the circled portion, there are usually peaks present on either side of the peaks for the other signals but the Blackman-Harris lowers those amplitudes. These were all done in trial and error.**

Lower pitched Blackman-Harris and Hanning windowed D3 signal; Signal, FFT




Both BH-H FFT for Lower D3
5000
4000
3000
2000
1000
0
128
130
132
134
136
138
140
142
144
frequency (Hz)

References:

a. Signals and Systems (English, Paperback, Barry Van Veen Simon Haykin)

b. MATLAB: An Introduction with Applications, 6th Edition: An Introduction with Applications by Amos Gilat

c. https://www.google.com/

d. https://www.wikipedia.org/