Autotuning System For String Instruments

ELL 205 Project Report

Tushar Gujral (2020CS10400)
Shivam Jain (2020CS50626)
Ishaan Govil (2020CS50497)

March 21, 2022

Contents

1 Introduction

The project is aimed at determining the tuning stage (under-tuned / over-tuned) of any string instrument like guitar, violin etc. This is a problem that is frequently encountered by both trained musicians and common people as they need to work with different instruments and even in the case of the same instrument the tension of the guitar string changes subject to the temperature conditions and other factors which lead to length expansion or contraction of strings.
Hence, by developing a model which can predict the over-tuning and under-tuning of a musical instrument we can help the instrumentalists with no longer needing to use their own ears to guess the under-tuning and over-tuning of the instrument which can be misleading.
By tuning a string musical instrument we are trying to get the appropriate tension in the string. The frequency produced by vibrating a string depends on the tension inside the string according to the relation

$$f = \frac{1}{2L}\sqrt{\frac{T}{\mu}}$$

, where f = frequency, L = length, T = tension, and μ = linear mass density of the string.
By rotating the string around the winding of the instrument changes the length of the string by a small amount, say ΔL. This changes the tension force inside the string which is given by the relation

$$Y = \frac{\frac{F}{A}}{\frac{\Delta L}{L}}$$

, where Y = Young's Modulus, F = Tension force inside the string, A = Cross-section area of the string, ΔL = change in length of the string, L = string length.

2 Project Flow

The flow of the project is as follows -

2.1 Plotting the Frequency Spectrum

1 First the input *.wav* or *.mp*3 file (which is the pluking of a single guitar string in our case) is sampled at the sampling frequency Fs which for our project is assumed to be 44100 Hz which is the standard sampling frequency.

2 Then the sampled data which is present in the form of a matrix is used to calculate the Fourier transform of the original music signal.

3 For doing this, we use the **FFT(The Fast Fourier Transform)** algorithm which gives us as output the Fourier transform of the input signal as a $N \times 2$ matrix where N is the number of samples and is equal to $Total\ time\ of\ the\ input\ signal(T) \times\ Sampling\ rate(F_s)$

4 We then calculate the absolute values of the fourier transform and normalize it by dividing it by the number of samples(N).

5 Now since for a real time signal, the frequency spectrum is symmetric around the dc signal, i.e around frequency = 0, hence we are only interested in the positive frequency spectrum of the signal and so we need to convert the double sided signal into a single sided signal.

6 To do this, we take the first N/2 elements of the frequency spectrum matrix so that we only get the positive side of the spectrum since the same information is present in the rest of the N/2 elements of the spectrum matrix as well.

7 We then plot this new modified frequency spectrum of the signal on the y-axis with the corresponding frequency on the x-axis to get an idea of the spectrum of the input signal.

2.2 Finding the Fundamental Frequency

1 Now to identify the fundamental harmonic of the input, we identify the top 3 peaks of the frequency transform. Assuming that the peak for fundamental frequency will be one among them, we find the one with the least frequency among those 3 peaks. This gives the fundamental frequency of the string vibration.

2 Any peak in matlab is a sample such that the sample just before and just after it is much less than the sample itself. This includes all the unnecessary local peak as well that were generated due to signal fluctuations. To get rid of these peaks, we set the peak width equal to 100 samples which removes all the peaks which are the neighbouring local peaks of the actual peak to be identified. To get these peaks we are using the inbuilt $findpeaks()$ function in matlab.

3 The reason for choosing the particular value of 100 samples and to ignore all the peaks that are present within a distance of 100 samples of each other is that assuming that the audio sample provided to us would be of atleast 1 second, meaning that the no. of samples that we would have would be atleast $1 \times F_s = 44100$ samples, and these no of samples are present for frequencies in the range of of $0 - 22050$ (since the maximum signal frequency ω_n is half of the sampling frequency $\omega_s\ or\ F_s$).

So for each frequency we would have approximately two samples corresponding to it. And since we know that the minimum frequency of a guitar string is around 82 Hz so we can safely assume that even an untuned guitar would have a minimum string frequency of around 60 Hz. Now between 2 different harmonic's peaks assuming that the minimum frequency is 60 Hz we would have atleast $60 \times 2 = 120\ samples$ and so we can safely assume that within an interval of 100 samples the different peaks occurring are redundant and correspond to the same harmonic of the fundamental frequency.

3 Fast Fourier Transform

3.1 Algorithm

Divide and Conquer algorithm is used for the computation of **Fourier Transform**.

1. Divide Fourier Series of length $2m$ into two series of length m each.

$$\begin{bmatrix} a_0 \\ \vdots \\ a_{2m-1} \end{bmatrix} = \frac{1}{2\cdot m} \begin{bmatrix} W_{2m}^0 & \cdots & W_{2m}^0 \\ \vdots & \ddots & \vdots \\ W_{2m}^0 & \cdots & W_{2m}^{(2m-1)^2 \bmod 2m} \end{bmatrix} \begin{bmatrix} x[0] \\ \vdots \\ x[2m-1] \end{bmatrix}$$

where $W_N^x = e^{-j\frac{2\pi}{N}x}$

Even-numbered entries in $x[n]$:

$$\begin{bmatrix} a_0 \\ \vdots \\ a_{2m-2} \end{bmatrix} = \begin{bmatrix} W_{m}^0 & \cdots & W_{m}^0 \\ \vdots & \ddots & \vdots \\ W_{m}^0 & \cdots & W_{m}^{(m-1)^2 \bmod m} \end{bmatrix} \begin{bmatrix} x[0] \\ \vdots \\ x[2m-2] \end{bmatrix}$$

Odd-numbered entries in $x[n]$:

$$\begin{bmatrix} a_1 \\ \vdots \\ a_{2m-1} \end{bmatrix} = \begin{bmatrix} W_{m}^0 & \cdots & W_{m}^0 \\ \vdots & \ddots & \vdots \\ W_{m}^0 & \cdots & W_{m}^{(m-1)^2 \bmod m} \end{bmatrix} \begin{bmatrix} x[1] \\ \vdots \\ x[2m-1] \end{bmatrix}$$

2. Break the original $2m$-point DTFS coefficients a_k into two parts:
$a_k = \frac{1}{2}(d_k + e_k)$
where d_k comes from the even-numbered $x[n]$ and e_k comes from the odd-numbered $x[n]$.

$$\begin{bmatrix} d_0 \\ \vdots \\ d_{2m-1} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} W_{2m}^0 & \cdots & W_{2m}^0 \\ \vdots & \ddots & \vdots \\ W_{2m}^0 & \cdots & W_{2m}^{(2m-1)^2 \bmod 2m} \end{bmatrix} \begin{bmatrix} x[0] \\ \vdots \\ 0 \end{bmatrix} \text{ (Here 0 is used instead of } x[2i+1])$$

$$\begin{bmatrix} e_0 \\ \vdots \\ e_{2m-1} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} W_{2m}^0 & \cdots & W_{2m}^0 \\ \vdots & \ddots & \vdots \\ W_{2m}^0 & \cdots & W_{2m}^{(2m-1)^2 \bmod 2m} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ x[2m-1] \end{bmatrix} \text{ (Here 0 is used instead of } x[2i])$$

3. The m-point DTFS coefficients b_k of the even-numbered $x[n]$:

$$\begin{aligned} \begin{bmatrix} b_0 \\ \vdots \\ b_{m-1} \end{bmatrix} &= \frac{1}{m} \begin{bmatrix} W_{m}^0 & \cdots & W_{m}^0 \\ \vdots & \ddots & \vdots \\ W_{m}^0 & \cdots & W_{m}^{(m-1)^2 \bmod m} \end{bmatrix} \begin{bmatrix} x[0] \\ \vdots \\ x[2m-2] \end{bmatrix} \\ &= \frac{1}{m} \begin{bmatrix} W_{2m}^0 & \cdots & W_{2m}^0 \\ \vdots & \ddots & \vdots \\ W_{2m}^0 & \cdots & W_{2m}^{2\cdot((m-1)^2 \bmod m)} \end{bmatrix} \begin{bmatrix} x[0] \\ \vdots \\ x[2m-2] \end{bmatrix} \end{aligned}$$

We can observe that:

$$\begin{bmatrix} d_0 \\ \vdots \\ d_{m-1} \\ d_m \\ \vdots \\ d_{2m-1} \end{bmatrix} = \begin{bmatrix} b_0 \\ \vdots \\ b_{m-1} \\ b_0 \\ \vdots \\ b_{m-1} \end{bmatrix}$$

4. The m-point DTFS coefficients c_k of the even-numbered $x[n]$:

$$\begin{aligned}\begin{bmatrix} c_0 \\ \vdots \\ c_{m-1} \end{bmatrix} &= \frac{1}{m}\begin{bmatrix} W_m^0 & \cdots & W_m^0 \\ \vdots & \ddots & \vdots \\ W_m^0 & \cdots & W_m^{(m-1)^2 \mod m} \end{bmatrix}\begin{bmatrix} x[1] \\ \vdots \\ x[2m-1] \end{bmatrix} \\ &= \frac{1}{m}\begin{bmatrix} W_{2m}^0 & \cdots & W_{2m}^0 \\ \vdots & \ddots & \vdots \\ W_{2m}^0 & \cdots & W_{2m}^{2\cdot((m-1)^2 \mod m)} \end{bmatrix}\begin{bmatrix} x[1] \\ \vdots \\ x[2m-1] \end{bmatrix}\end{aligned}$$

We can observe that:

$$\begin{bmatrix} e_0 \\ \vdots \\ e_{m-1} \\ e_m \\ \vdots \\ e_{2m-1} \end{bmatrix} = \begin{bmatrix} W_{2m}^0 c_0 \\ \vdots \\ W_{2m}^{m-1} c_{m-1} \\ W_{2m}^m c_0 \\ \vdots \\ W_{2m}^{2m-1} c_{m-1} \end{bmatrix}$$

5. $\begin{bmatrix} a_0 \\ \vdots \\ a_{2m-1} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} d_0 + e_0 \\ \vdots \\ d_{2m-1} + e_{2m-1} \end{bmatrix}$

3.2 Time Complexity

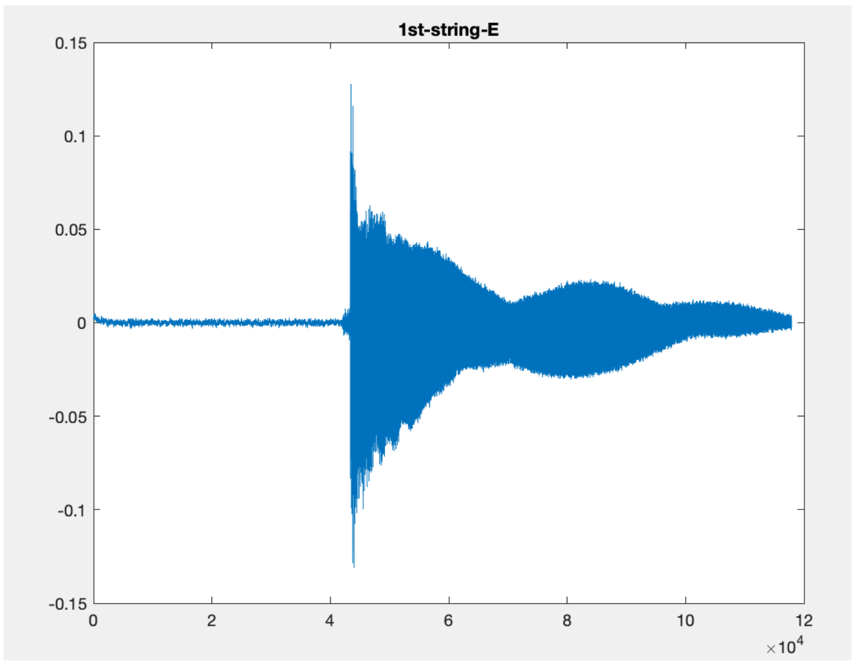
Let $N(n)$ be the number of arithmetic operations to compute n-point FFT (Here $n = 2^k$, $k \geq 0$)

$N(1) = 0$
$N(2) = 2N(1) + 6$
$N(4) = 2N(2) + 12$
$\ldots$
$N(n) = 2N(n/2) + 3 \cdot n$
$\implies N(n) = O(n \log_2 n)$

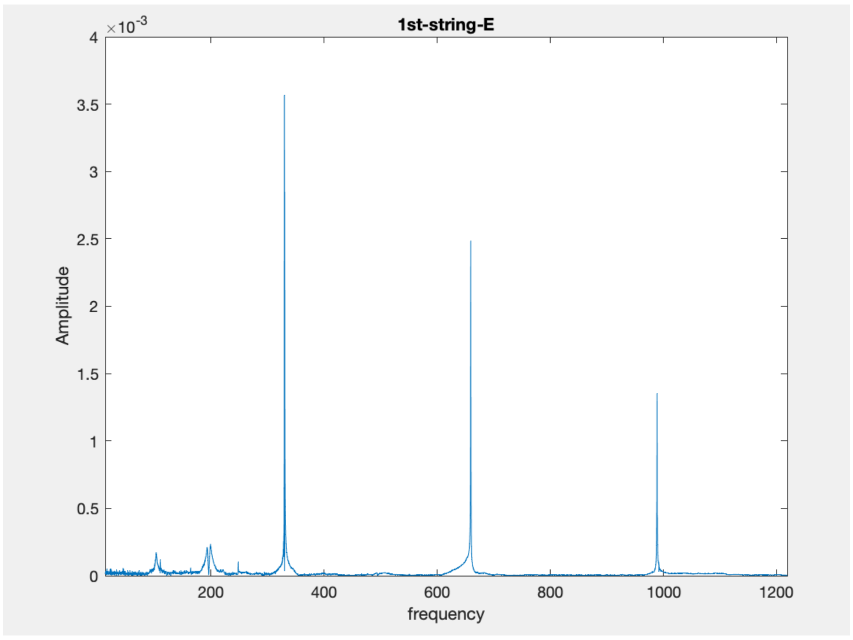
Thus, time-complexity is reduced significantly by using FFT instead of conventional matrix multiplication.

4 Sample Audios

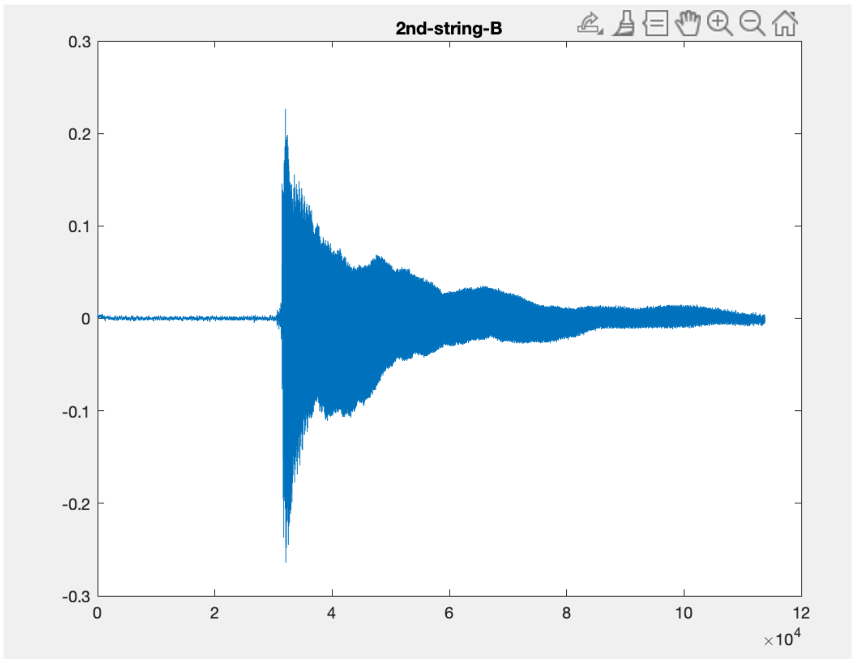
4.1 1st-string-E



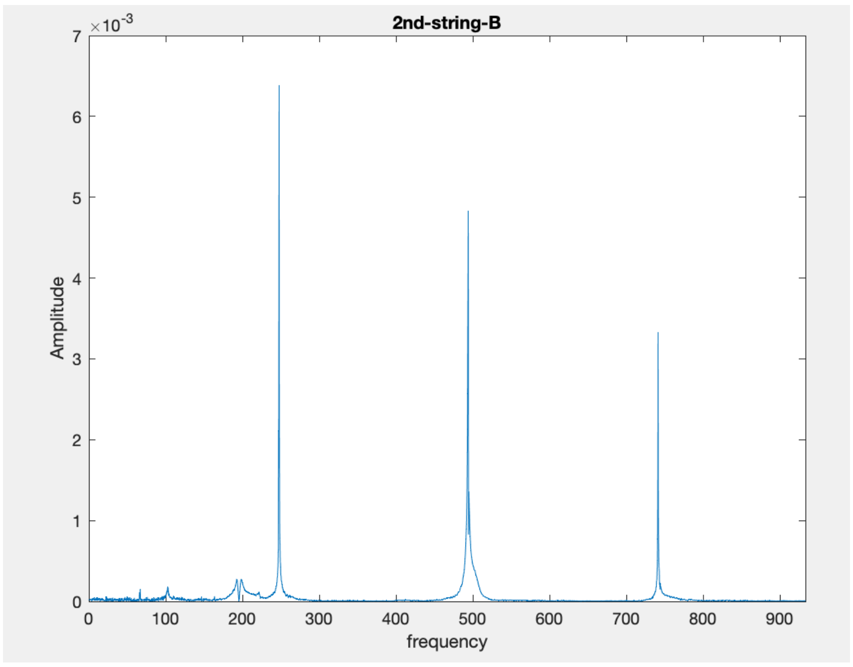
$$Fundamental\ Frequency = 329.68 \text{ Hz}$$



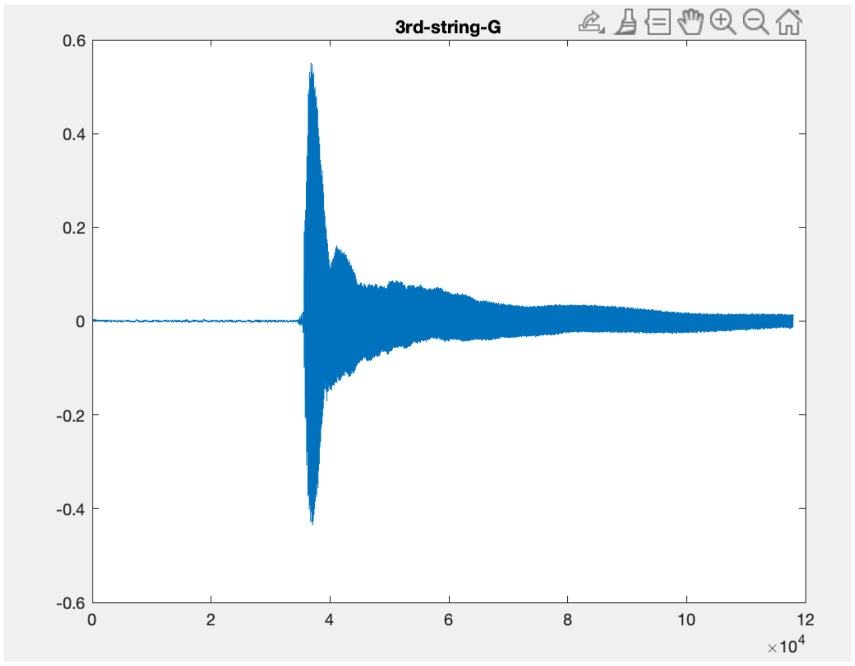
4.2 2nd-string-B



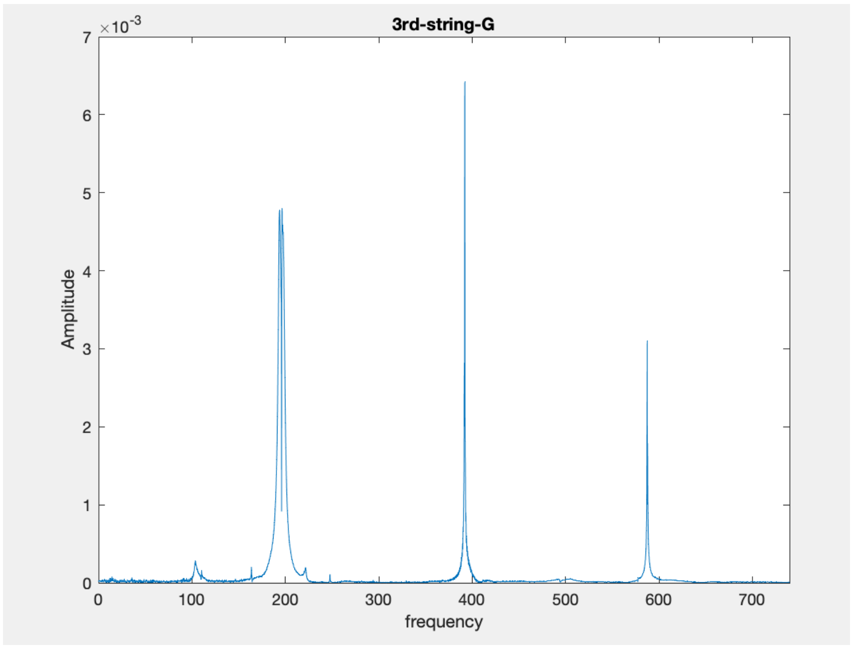
$$Fundamental\ Frequency = 247.97\text{ Hz}$$



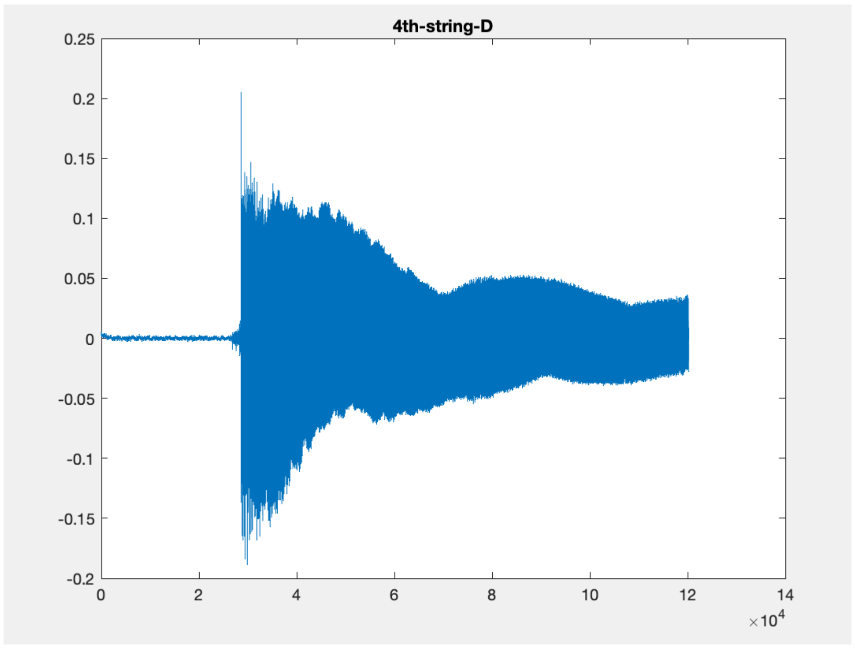
4.3 3rd-string-G



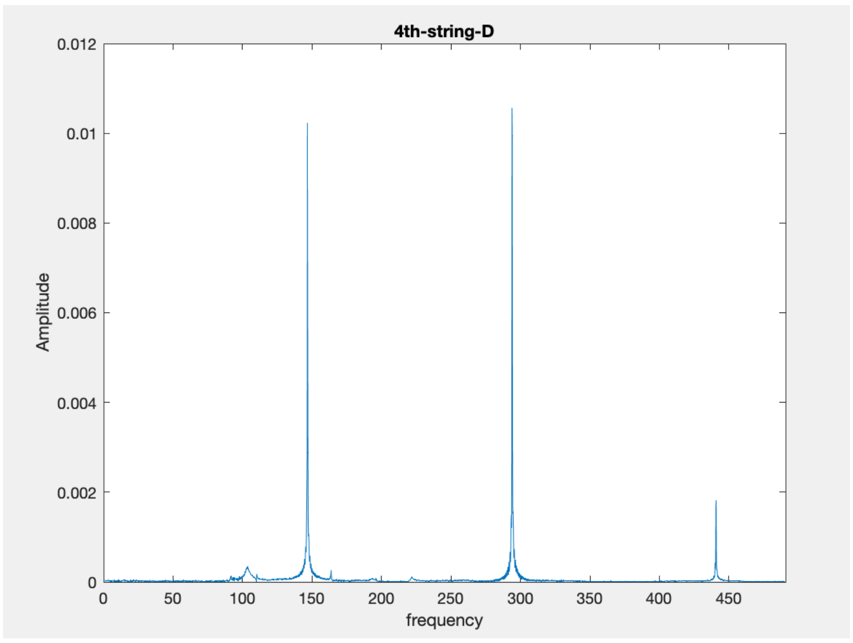
$$Fundamental\ Frequency = 196.72 \text{ Hz}$$



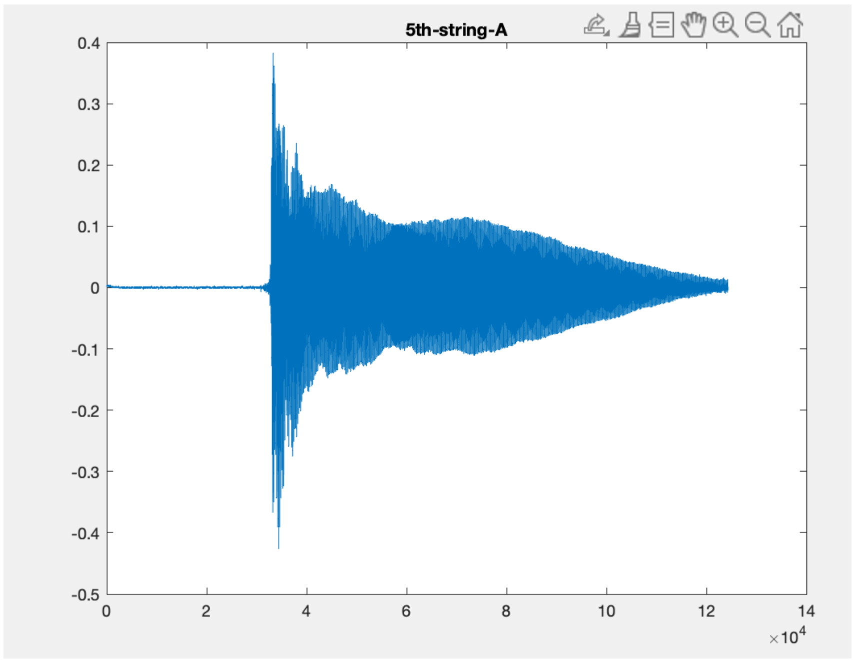
4.4 4th-string-D



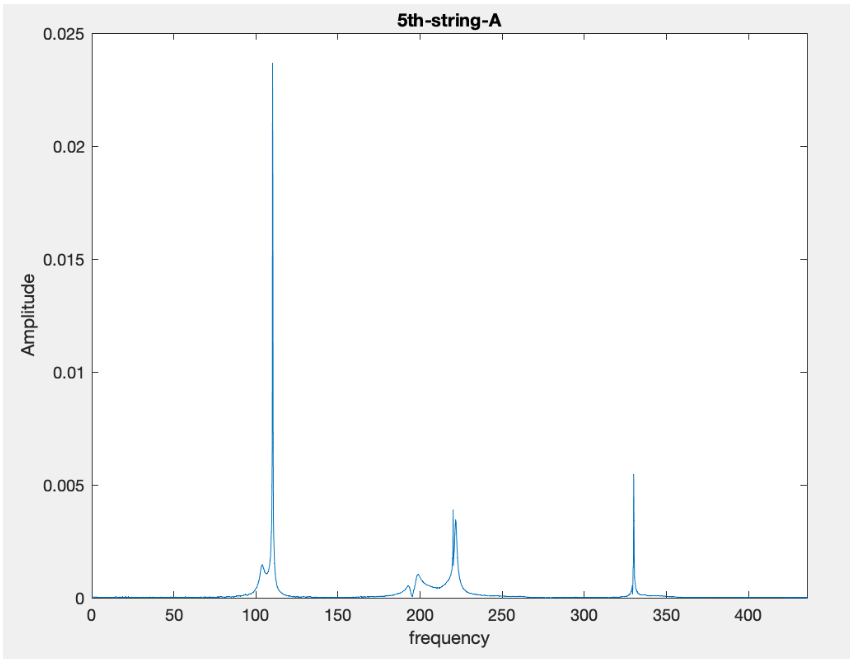
$$Fundamental\ Frequency = 146.88 \text{ Hz}$$



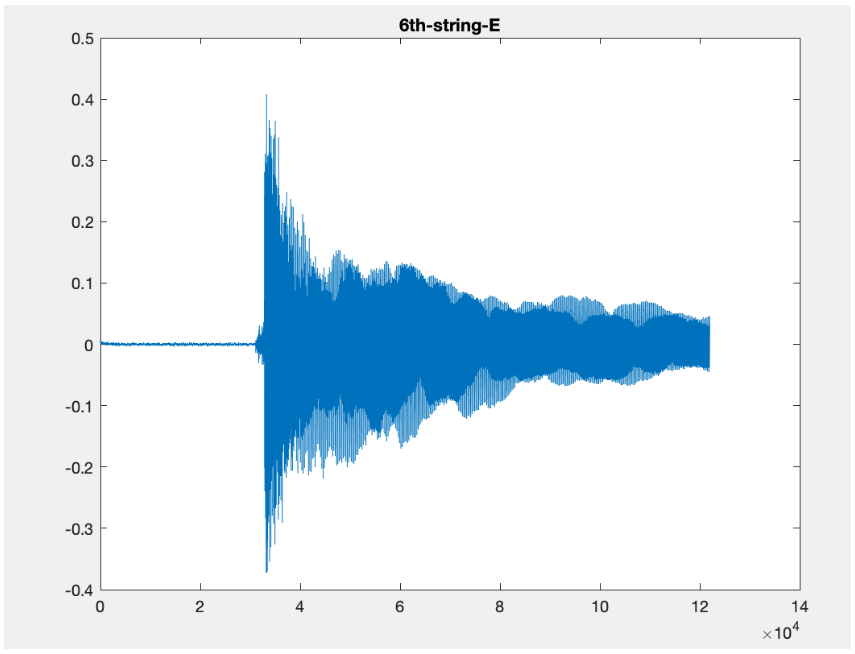
4.5 5th-string-A



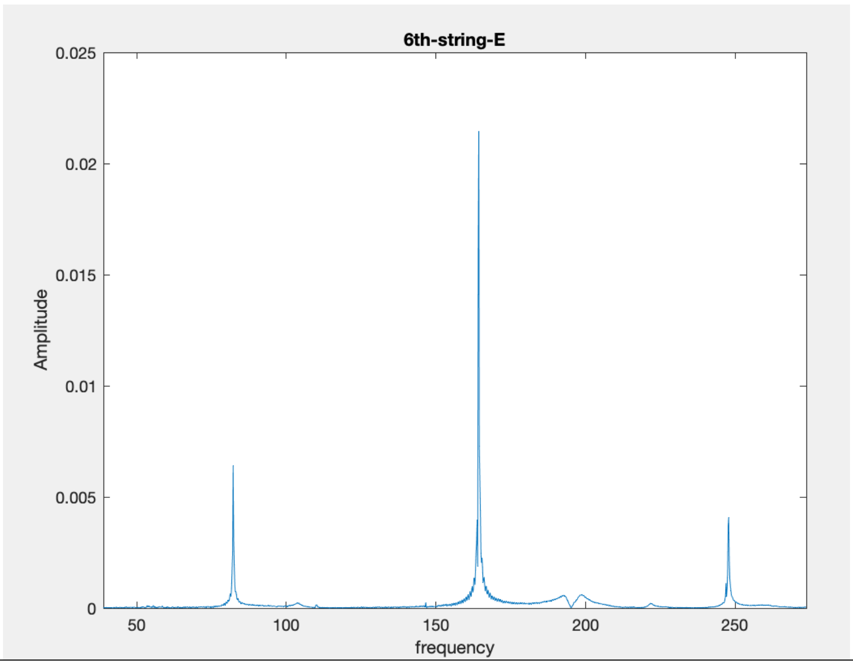
$$Fundamental\ Frequency = 110.29 \text{ Hz}$$



4.6 6th-string-E



$$\textit{Fundamental Frequency} = 82.41 \text{ Hz}$$


6th-string-E
Amplitude
frequency