Project: Guitar Pedal Filter

Ahmed Sabry - 99025207

University of British Columbia - Okanagan

ENGR 362 - Digital Signal Processing I

Dr. Christopher Collier

TA: Nourhan Soliman

April 4, 2022

Overview

In this project a digital guitar compressor pedal is implemented using MATLAB and digital signal processing filters. The signal recording text file is provided with the data sampled at a sampling frequency, f_s, of 48kHz. The recording is of a D Major chord, which consists of four pitches: 146.83 Hz (D3), 220.00 Hz (A3), 293.66 Hz (D4), and 369.99 Hz (F#4). The time-domain and frequency-domain plots of this signal are analyzed to produce a compressed signal with individual frequency amplitudes within 10% of each other.

D Major Chord Recording

The provided text file is a recording of a D Major chord sampled at 48 kHz. Having known the sampling frequency, the time at each element is a function of its index in the data and thus the time at each value can be easily determined using $t_n = T_s * n$. The time-domain plot of the recording is shown in Figure 1.

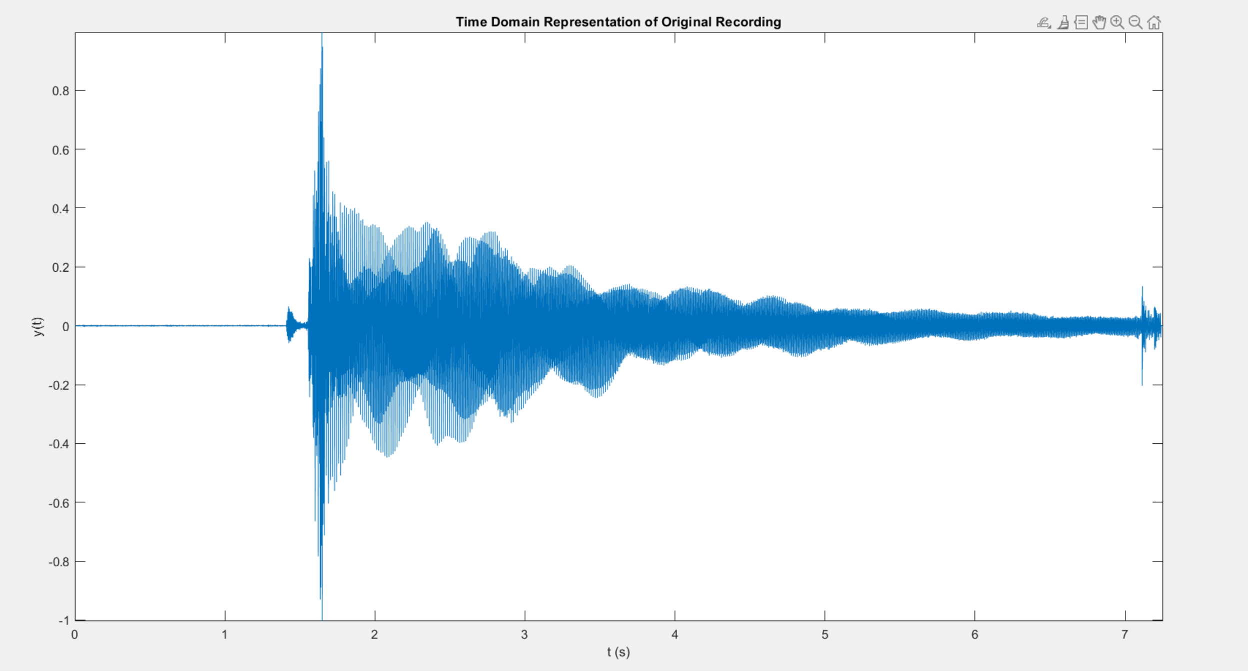


Figure 1. Time Domain Representation of Original Recording

By performing an FFT operation on the time-domain signal, the frequency-domain representation of the input signal can be computed. The D major chord consists of four pitches: 146.83 Hz (D3), 220.00 Hz (A3), 293.66 Hz (D4), and 369.99 Hz (F#4). The frequency-domain plots are shown in Figures 2 and 3. The frequency-domain plots clearly show peaks at these frequencies indicating their prevalence in the recording. Section 7 of the code is used to accurately pinpoint these frequencies, their indices, and the peak amplitudes. The 4 pitches found in the recording are pinpointed to be 146.3 Hz (D3), 220.60 Hz (A3), 294.6 Hz (D4), and 369.5 Hz (F#4).

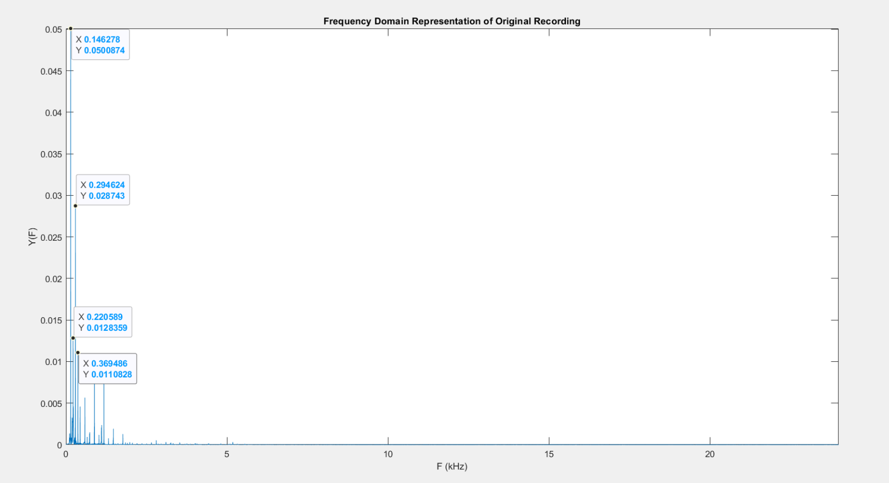


Figure 2. Frequency Domain Representation of Original Recording

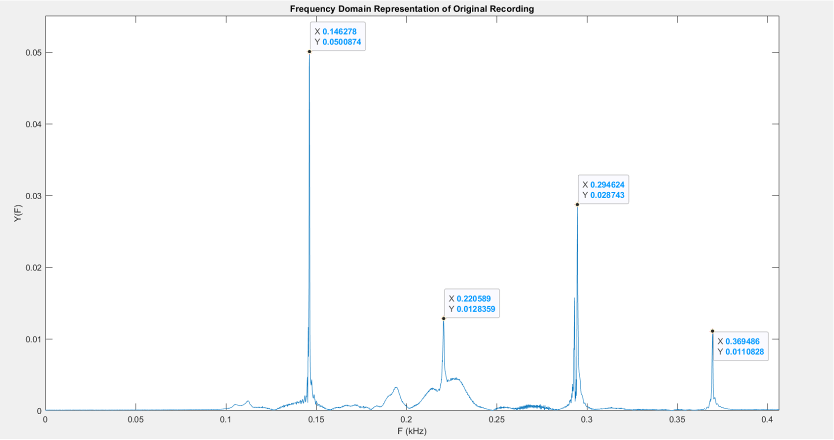


Figure 3. Frequency Domain Plot of Original Recording

Top Level Block Diagram

The processing of the recording is modeled with a block diagram shown below in Figure 4. The input signal is fed into a filter bank that isolates each frequency by means of a bandpass filter. The result is four filtered signals each with the filtered note as the dominant frequency. Each signal is then normalized with respect to its peak amplitude resulting in a signal that has a frequency-domain representation with each peak valued at 1. Finally, all four filtered signals are then added together to construct the compressed signal. The frequency-domain representation of the compressed signal is then verified to have the frequency amplitudes within 10% of each other.

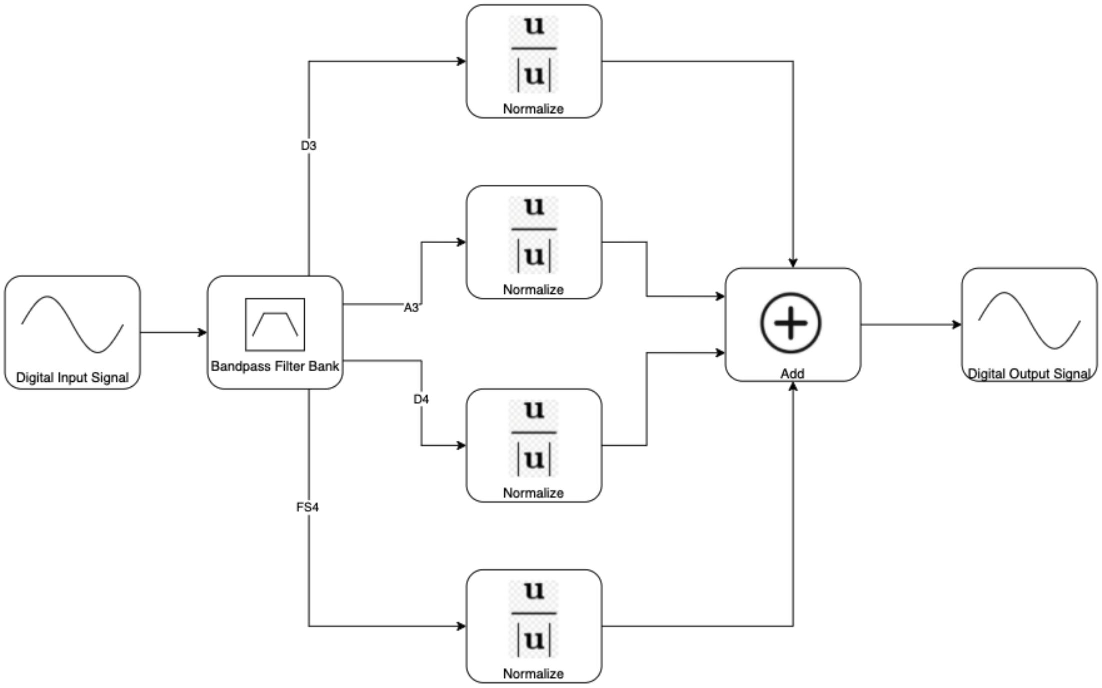


Figure 4. Top-Level Block Diagram

Filter Bank

The filter bank is created by designing 4 bandpass filters that allow only the intended frequency to pass, therefore, there is one bandpass filter per pitch. Each bandpass filter is created by cascading a Type 1 Chebychev lowpass filter in series with a Type 1 Chebychev highpass filter. The order, bandpass ripple, and frequency window are parameters of each filter. Figure 5 shows the block diagram demonstrating the composition of the filter bank.

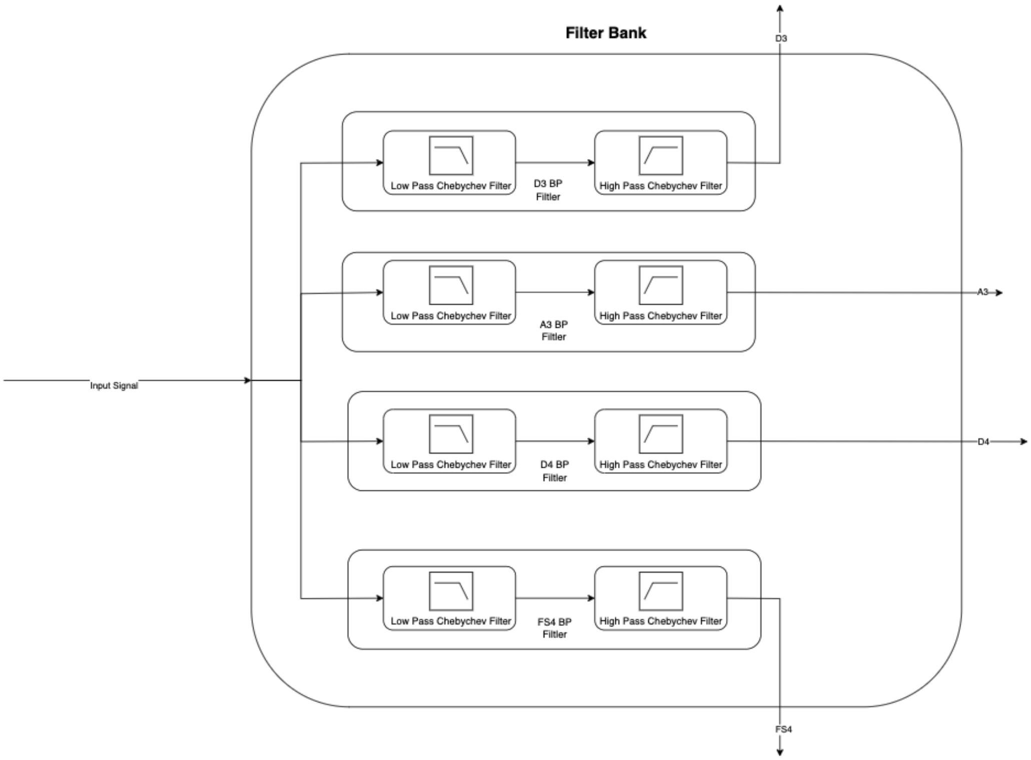

Filter Bank
D3
Low Pass Chebychev Filter
D3 BP Filtler
High Pass Chebychev Filter
Low Pass Chebychev Filter
A3 BP Filtler
High Pass Chebychev Filter
A3
Input Signal
Low Pass Chebychev Filter
D4 BP Filtler
High Pass Chebychev Filter
D4
Low Pass Chebychev Filter
FS4 BP Filtler
High Pass Chebychev Filter
FS4

Figure 5. Filter Bank Block Diagram

Each filter will create a transfer function corresponding to its frequency response behavior. By cascading the lowpass filter in series with the high pass filter, the **series(A,B)** MATLAB command is utilized to calculate the equivalent bandpass filter frequency response.

Series cascaded filters have an overall transfer function that is the product of the individual transfer functions as shown in Figure 6[1].

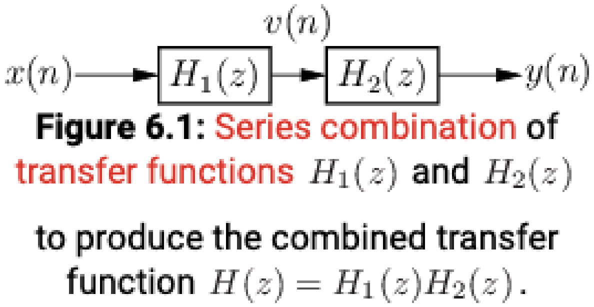


Figure 6.1: Series combination of transfer functions $H_1(z)$ and $H_2(z)$ to produce the combined transfer function $H(z) = H_1(z)H_2(z)$.

Figure 6. Series Transfer Functions Combination

The filters are a function of the order, bandpass ripple, and frequency window. Parameters used to create one filter will not necessarily produce optimal filtering behavior for other filters. The optimal filter parameters are determined by trial and error and are shown below in Table 1. The parameters are selected according to the isolation of the passband frequency. **Filter parameters are determined to be optimal if they result in a normalized DFT graph of the filtered signal with only one spike at the passband frequency.** The frequency response plots shown in Figures 7 - 10 were generated using the optimal parameters. Section 8 of the code plots the frequency response of each filter.

	D3 (146.83 Hz)	**A3 (220.00 Hz)**	**D4 (293.66 Hz)**	**F#4 (369.99 Hz)**
$Filter\ Order\ (n)$	3	5	3	4
$Passband\ Ripple\ (Rp)$	2	2	5	8
Δf (%)	2	0.01	1	1

Table 1. Optimal Filter Parameters

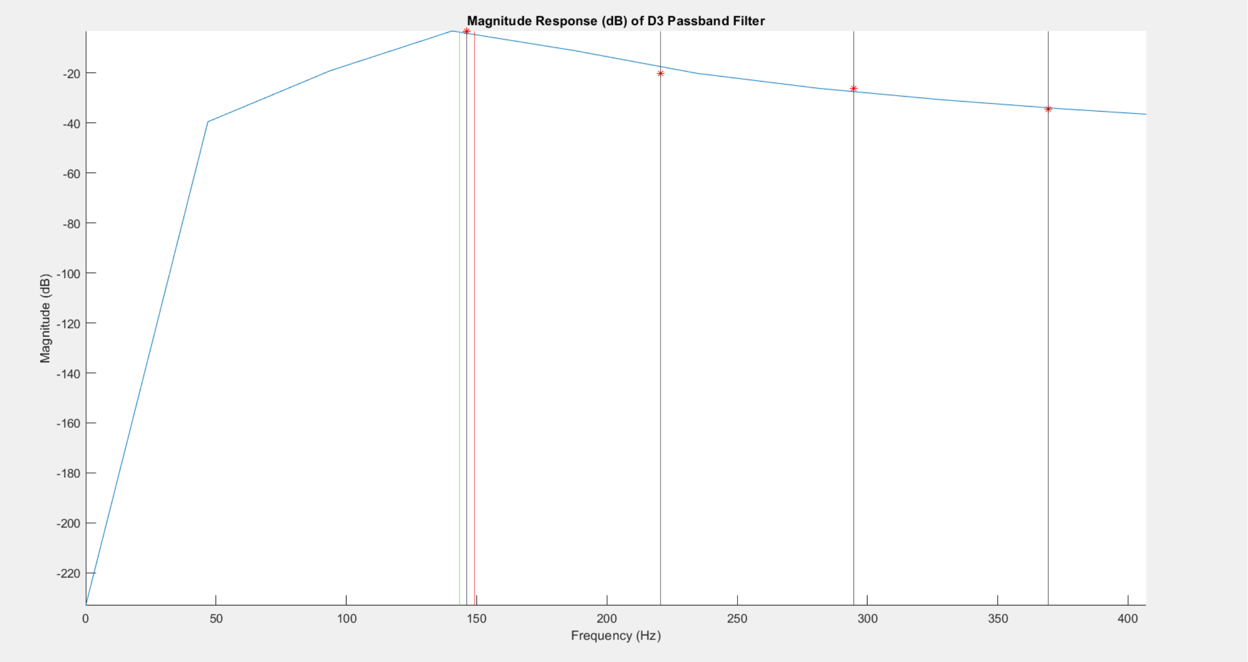


Figure 7. Frequency Response of D3 Bandpass Filter

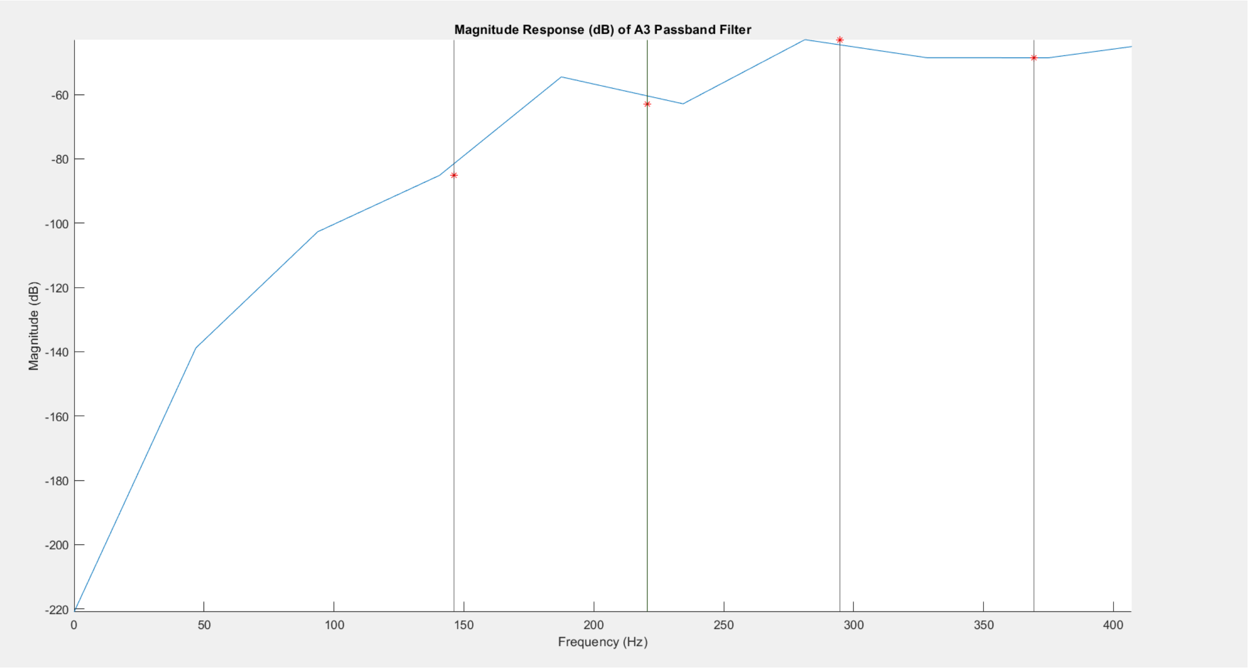


Figure 8. Frequency Response of A3 Bandpass Filter

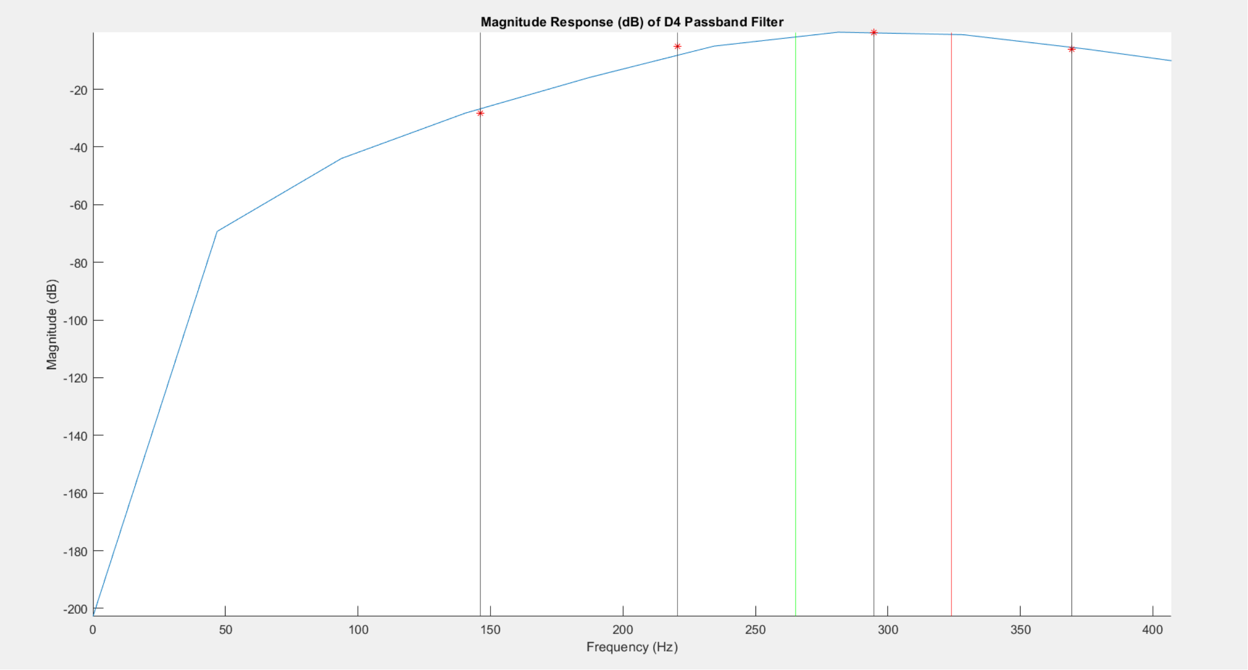


Figure 9. Frequency Response of D4 Bandpass Filter

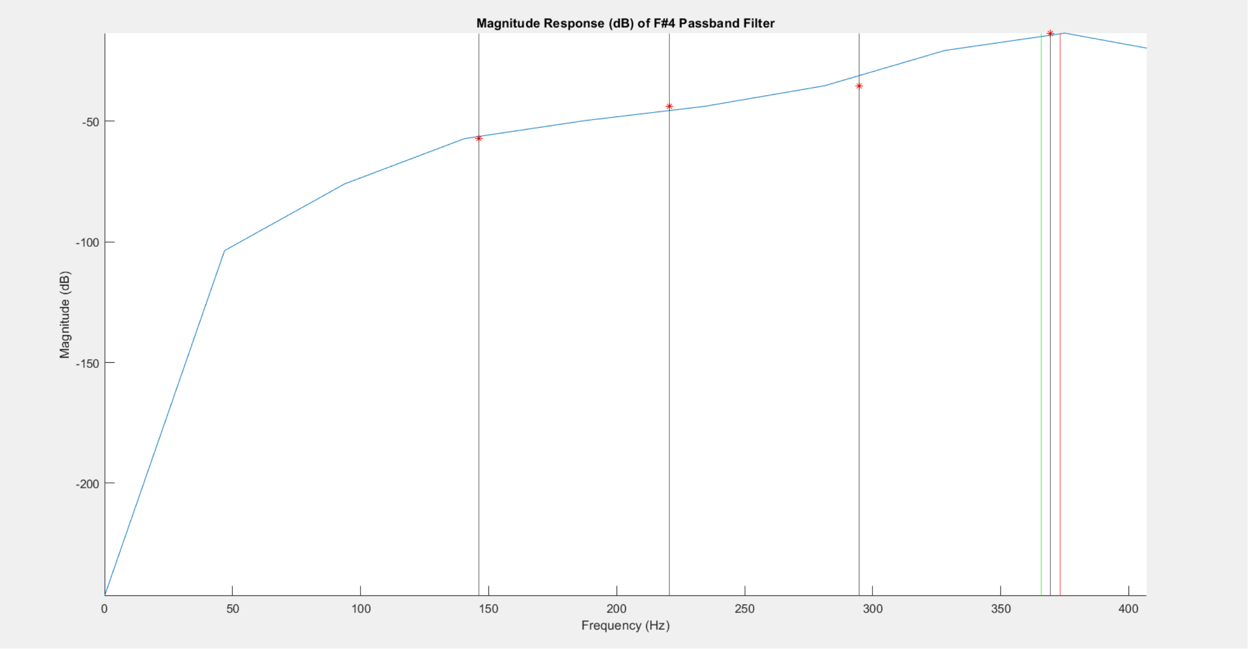


Figure 10. Frequency Response of F#4 Bandpass Filter

Filtered Signals

Section 9 of the code then creates the filters with the optimized parameters and creates the array of output signals, i.e. a matrix, to be further processed. Figures 11 - 19 show the time-domain and frequency-domain representations of the filtered signals.

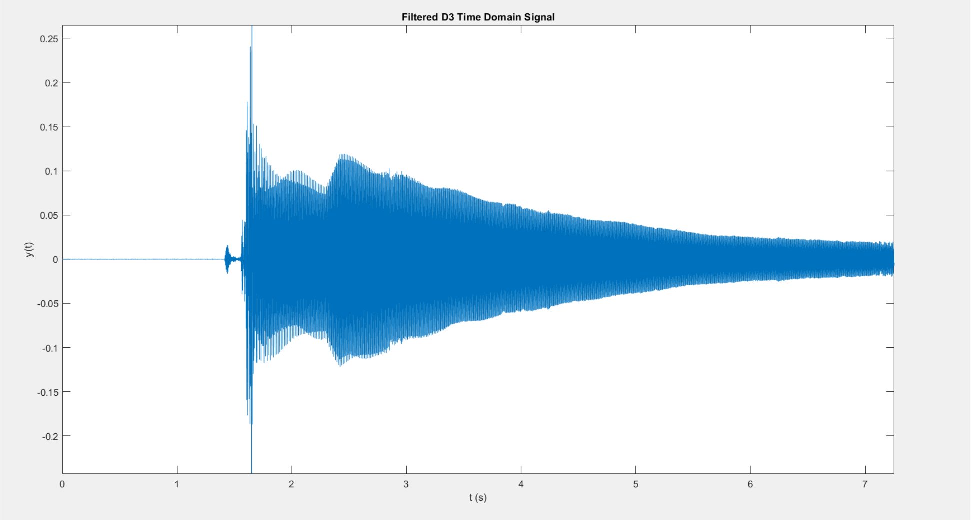


Figure 11. Filtered D3 Time Domain Signal

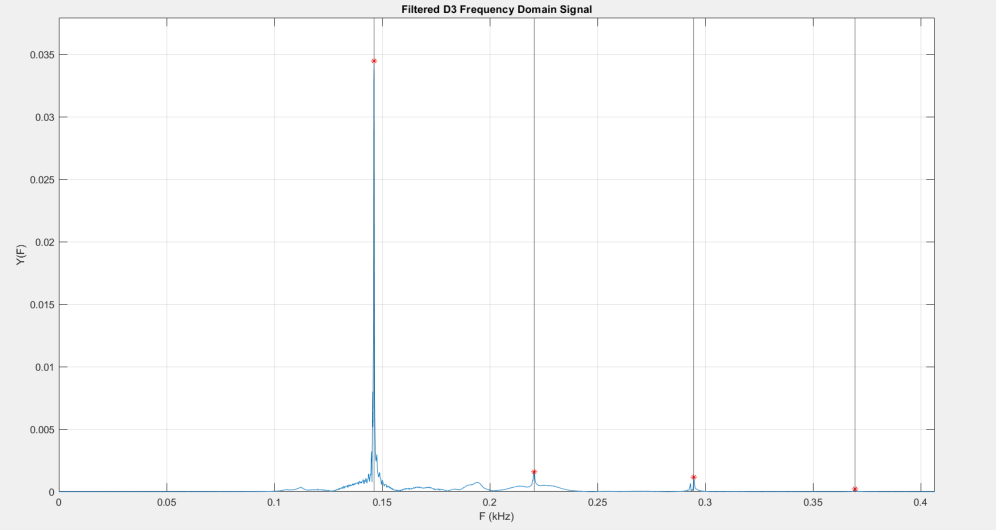


Figure 12. Filtered D3 Frequency Domain Signal

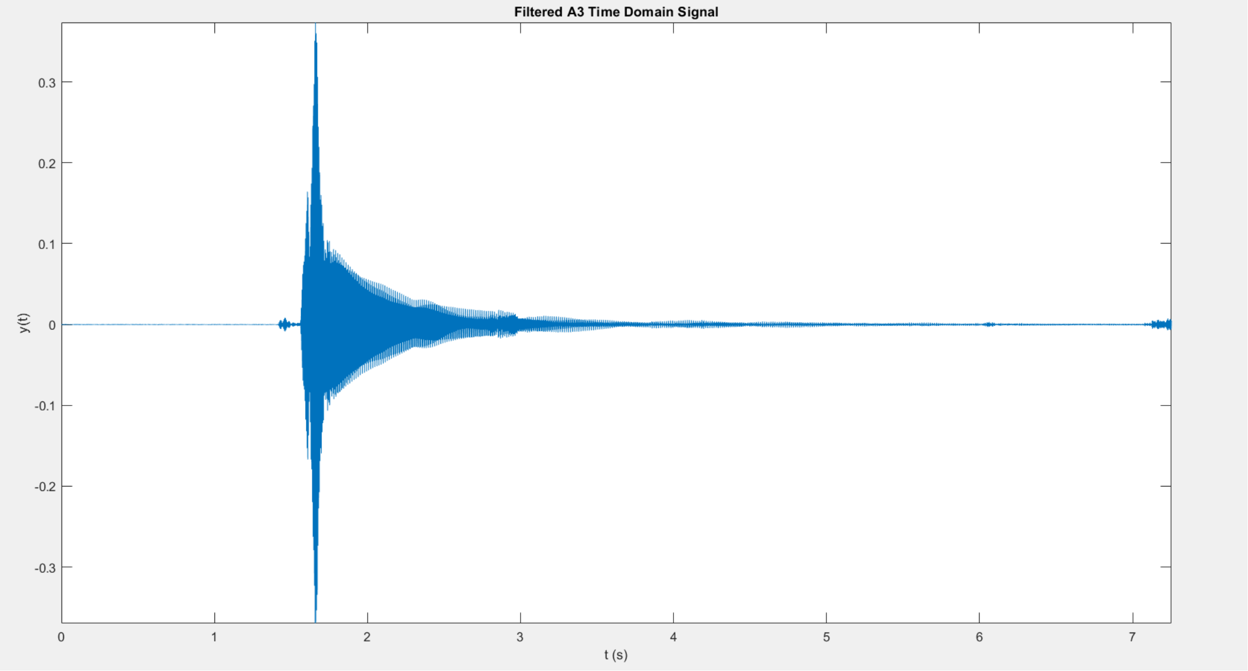


Figure 13. Filtered A3 Time Domain Signal

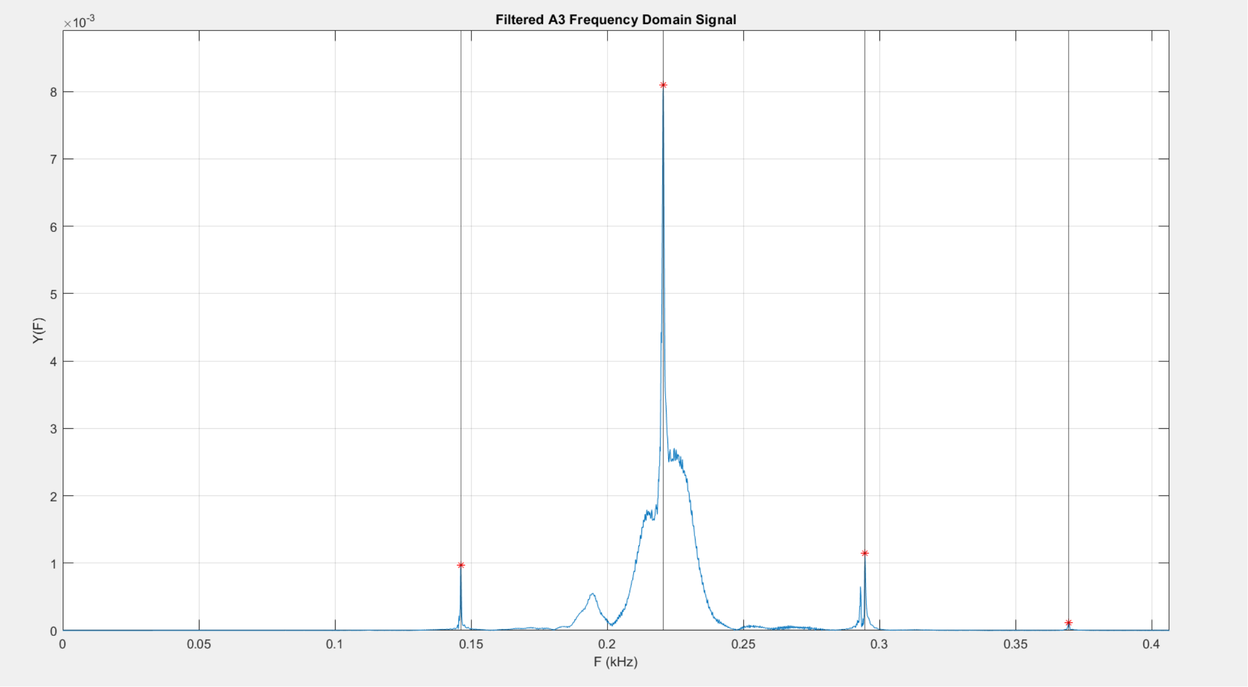


Figure 14. Filtered A3 Frequency Domain Signal

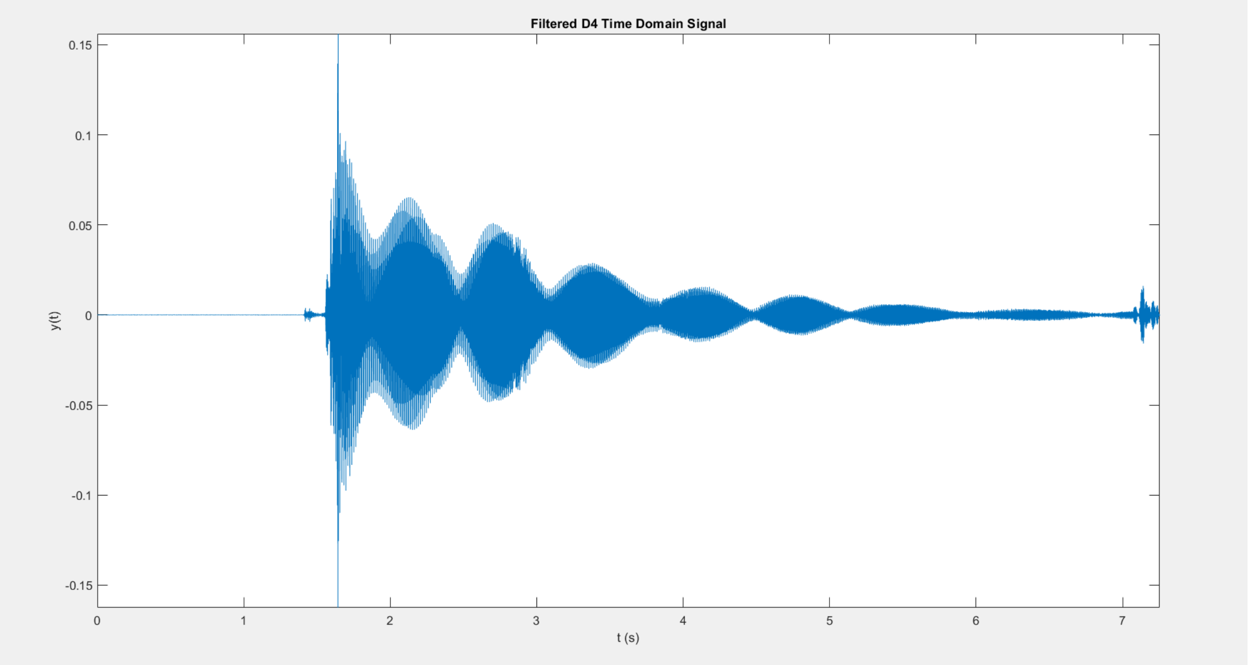


Figure 15. Filtered D4 Time Domain Signal

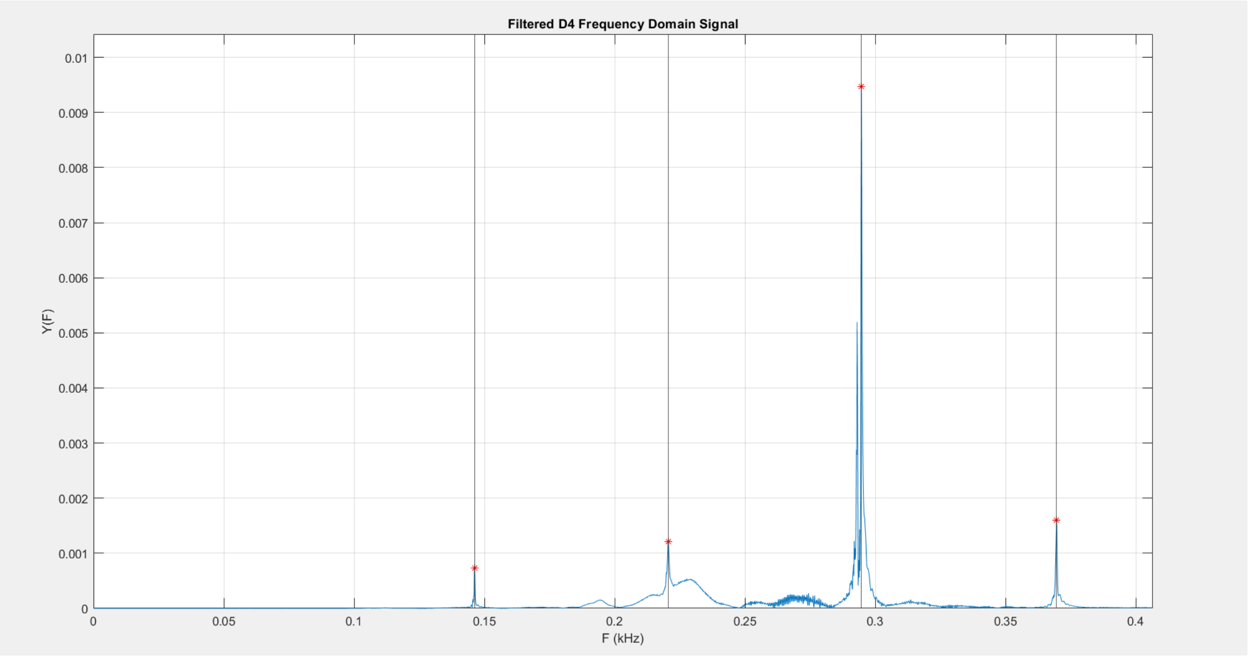


Figure 16. Filtered D4 Frequency Domain Signal

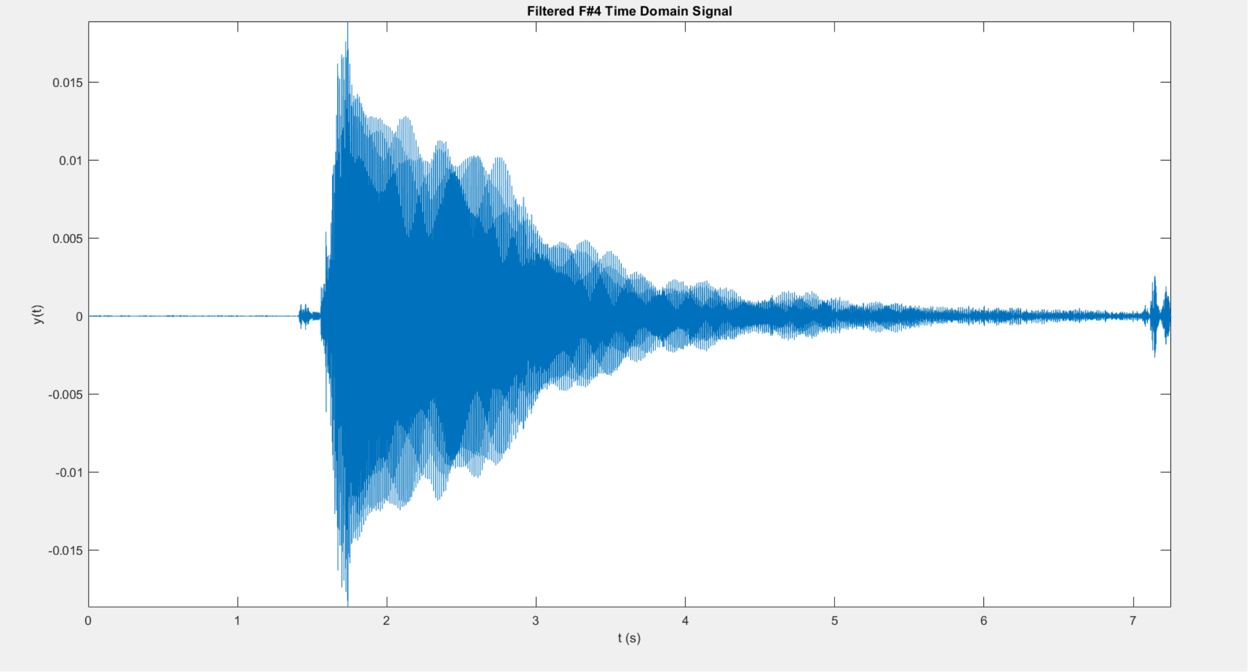


Figure 17. Filtered F#4 Time Domain Signal

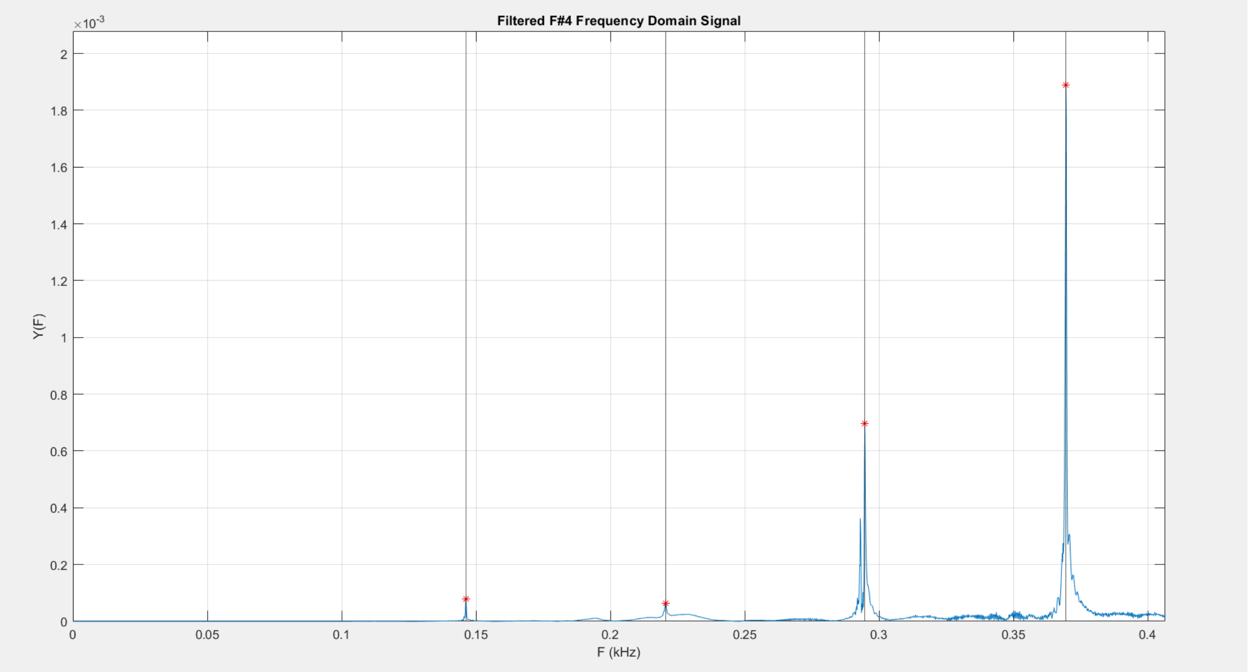
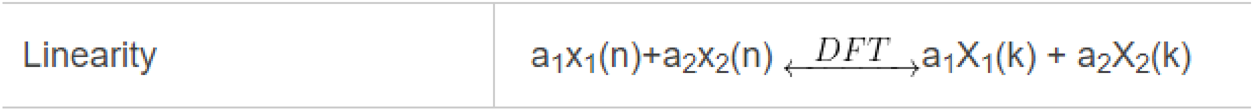


Figure 18. Filtered F#4 Frequency Domain Signal

Normalization

Section 10 of the code then normalizes each filtered signal with the amplitude of the filtered frequency. The DFT is linear, therefore, normalizing with the amplitude of the filtered frequencies will proportionally normalize the time domain signal as shown in Figure 19. The magnitudes of the filtered frequencies are shown in Table 2. Each filtered signal is then normalized with respect to its filtered frequency magnitude. Figures 20 - 23 show the normalized frequency-domain plots.



Linearity	$a_1x_1(n)+a_2x_2(n) \xleftrightarrow{DFT} a_1X_1(k) + a_2X_2(k)$

Figure 19. Linearity Property of the DFT[2]

Frequency	**D3 (146.83 Hz)**	**A3 (220.00 Hz)**	**D4 (293.66 Hz)**	**F#4 (369.99 Hz)**
Amplitude	0.0501	0.0128	0.00287	0.0101

Table 2. Filtered Frequency Amplitude

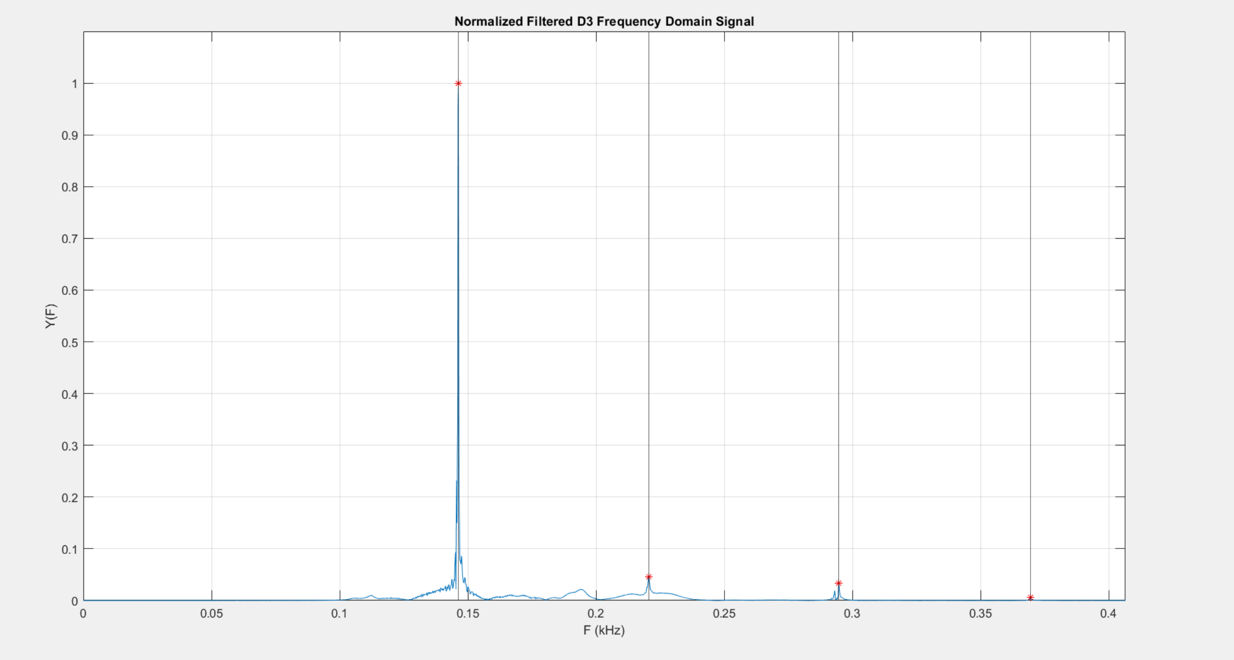


Figure 20. Normalized Filtered D3 Frequency Domain Signal

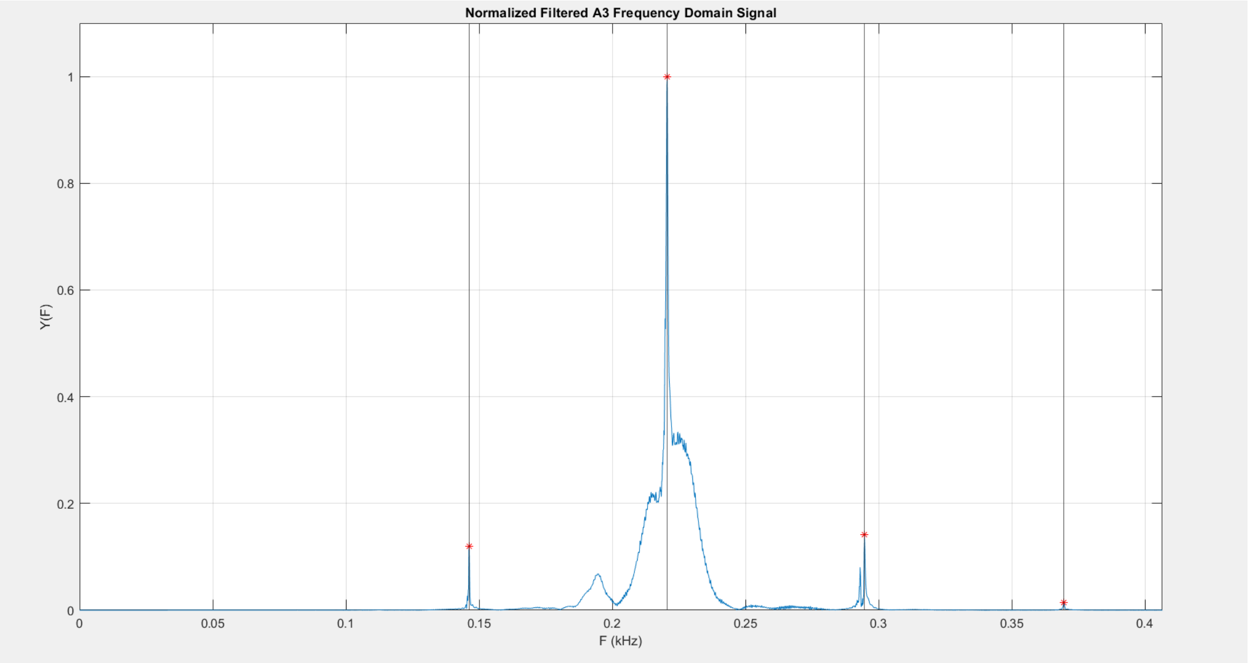


Figure 21. Normalized Filtered A3 Frequency Domain Signal

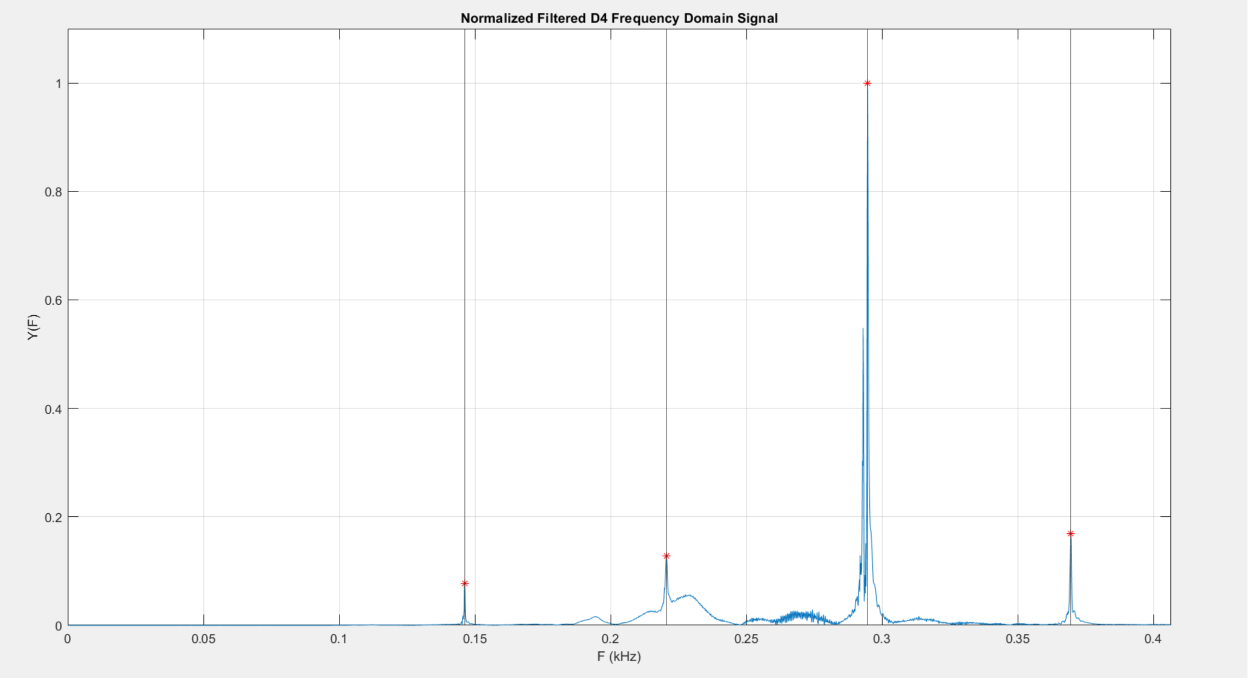


Figure 22. Normalized Filtered D4 Frequency Domain Signal

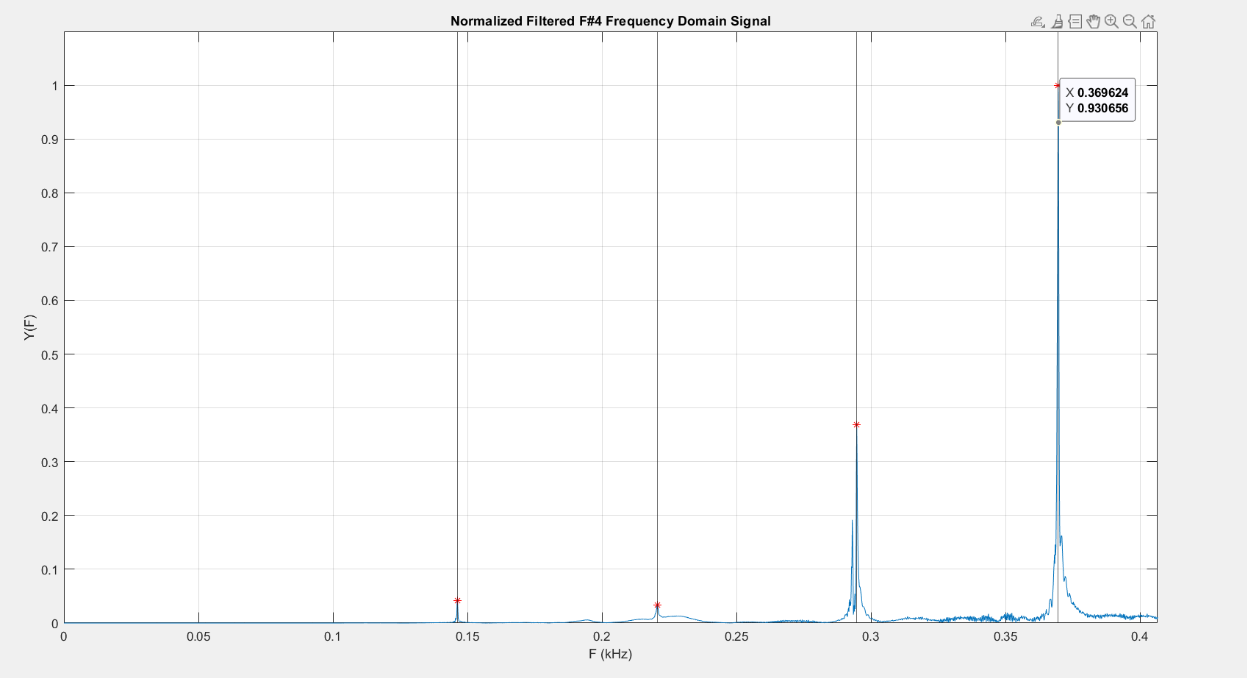


Figure 23. Normalized Filtered F#4 Frequency Domain Signal

Reconstruction

Finally, the normalized signals are then added together to construct the compressed and scaled by the maximum filtered frequency amplitude to construct the compressed signal. Due to the linearity property of the DFT, the compressed signal can be created by adding either the time domain representations of the normalized signals or adding the frequency-domain representations of the normalized signals. Section 11 of the code adds the frequency domain signals and performs an IDFT operation to produce the time-domain representation of the compressed signal. Section 12 of the code adds the time-domain signals and performs a DFT operation to produce the frequency-domain representation of the compressed signal. Figure 24 shows the time-domain plot of the compressed signal. Figures 25 and 26 show the frequency-domain plot of the compressed signal.

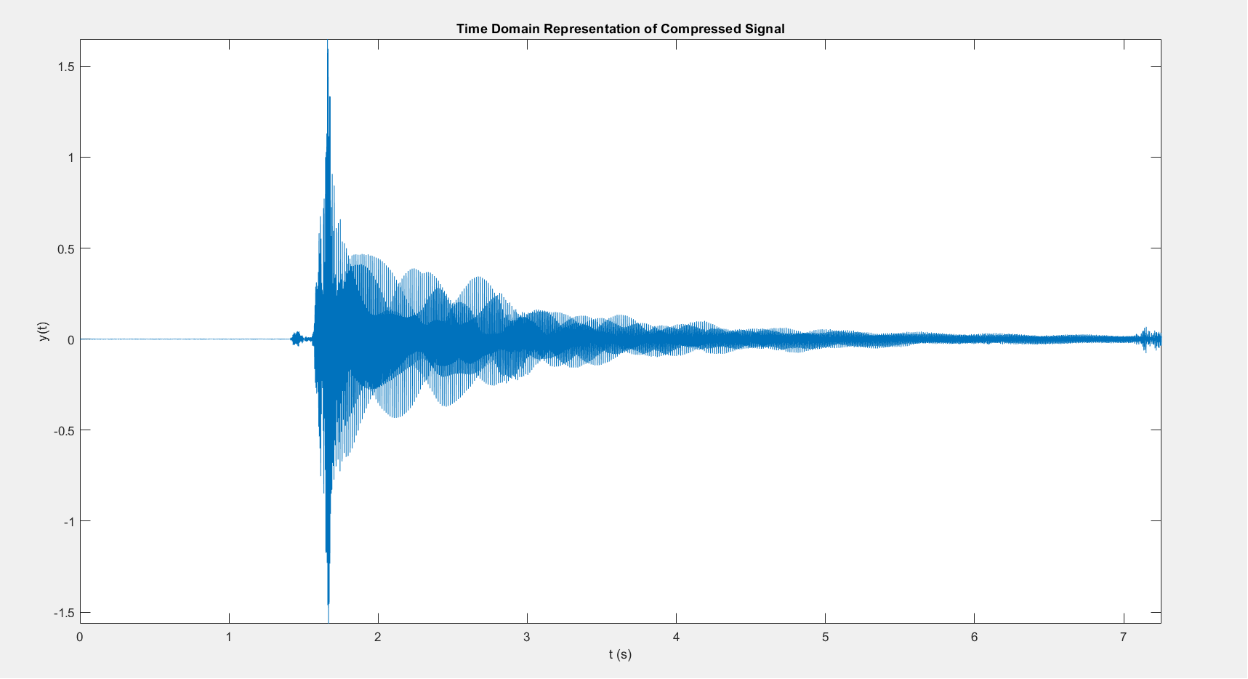


Figure 24. Time Domain Representation of Compressed Signal

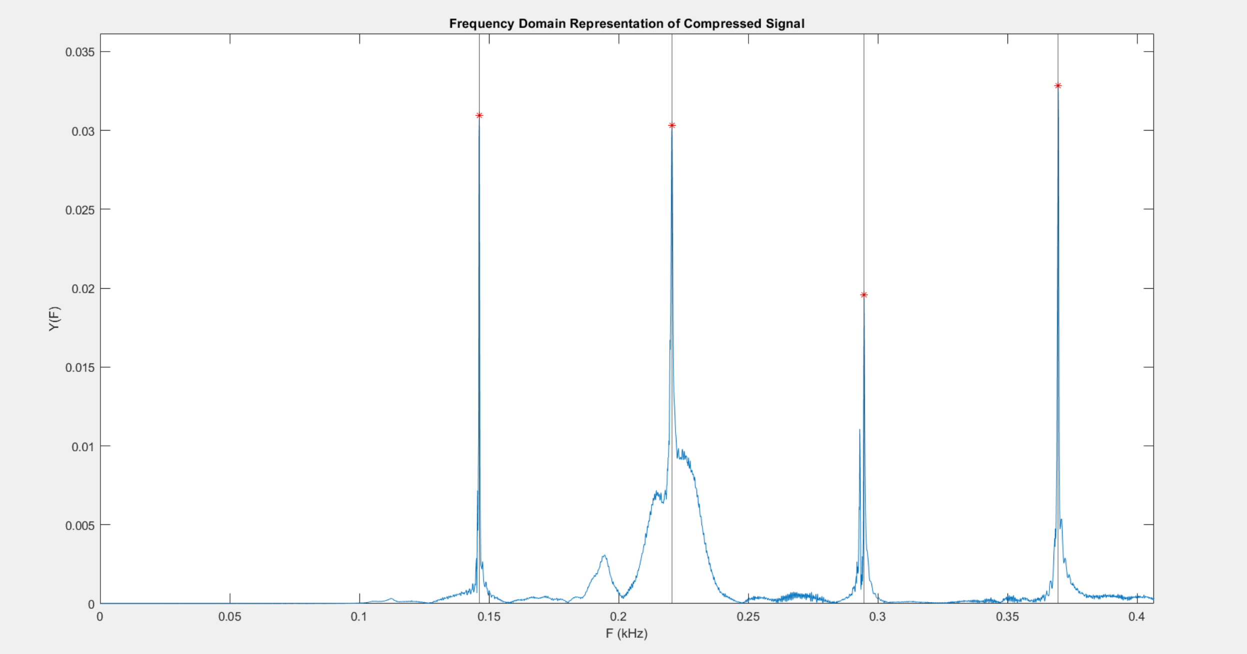


Figure 25. Frequency Domain Representation of Compressed Signal

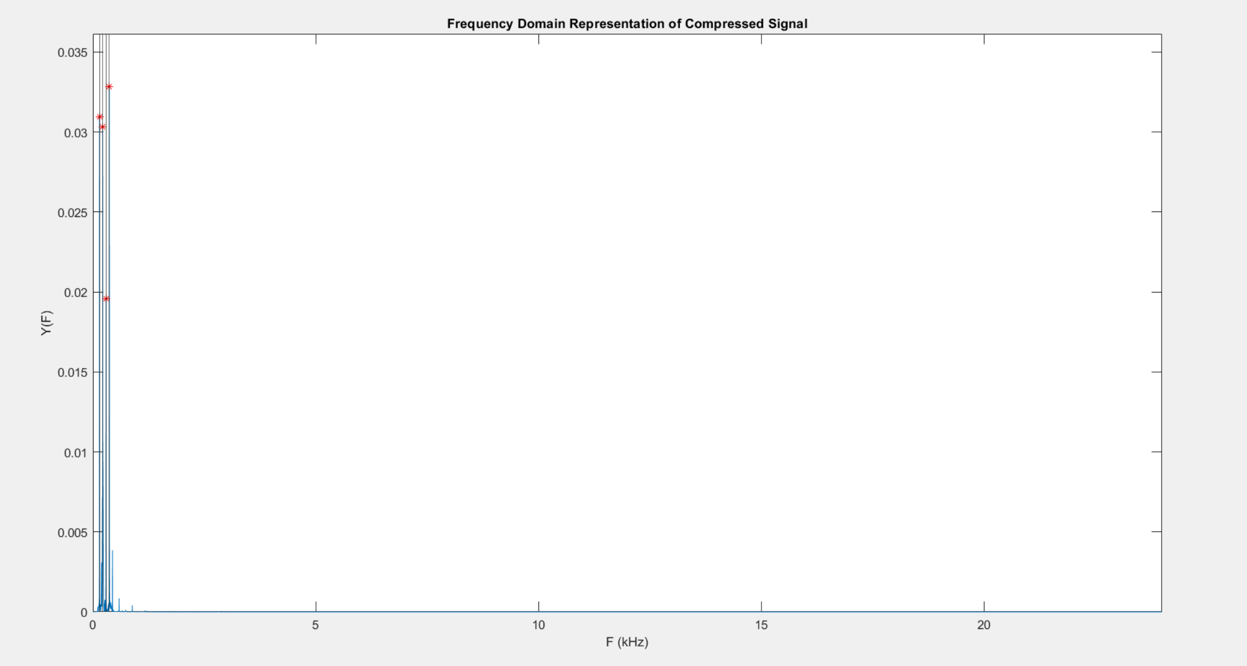


Figure 26. Zoomed Out Frequency Domain Representation of Compressed Signal

Equivalent System Transfer Function

This system was modeled using Simulink and produced the same output. Section 13 of the code stores the Simulink model parameters in the workspace for the Simulink model to use. The Simulink model is shown below in Figure 27. This system has an equivalent transfer function as it is four transfer functions with a gain connected in parallel with an add operation. Section 14 of the code derives the entire system transfer function and its frequency response is shown below in Figure 28. The frequency response clearly shows peaks at the bandpass frequencies of choice.

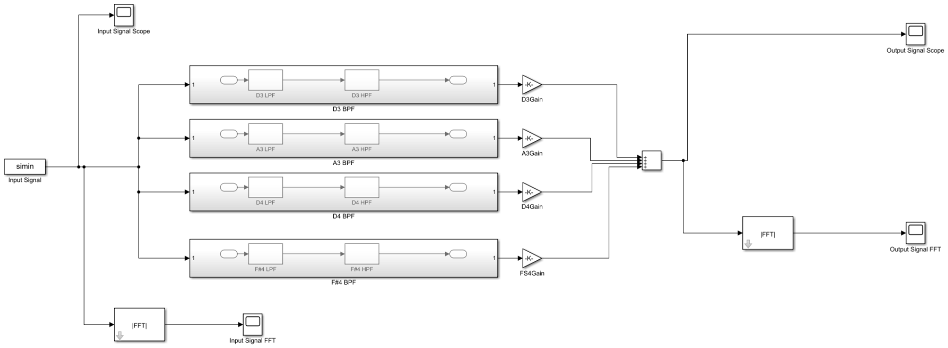


Figure 27. Simulink Model of System

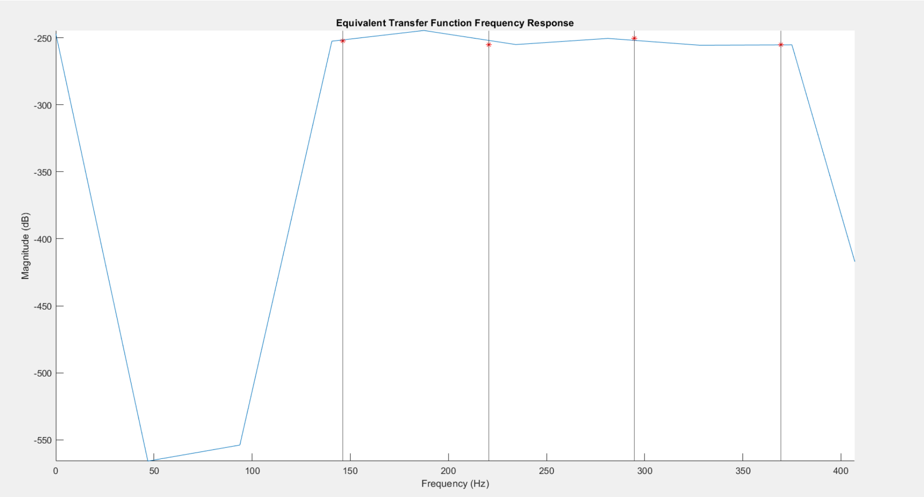


Figure 28. Equivalent System Frequency Response

Conclusion

Although the amplitudes of all of the pitches are not completely within 10% of each other, the resulting signal is more compressed compared to the original signal with the D Major Chord tones being the dominant frequencies of the resultant signal. All the unnecessary information that was present in the original signal is removed, effectively compressing the signal. By tuning the filter parameters more finely, the closer the amplitudes of the pitches become closer to each other. Furthermore, although the bandpass filter parameters do not produce an ideal bandpass filter frequency response that does not attenuate the bandpass frequency, the resulting filtered signals are close to each other as required by the project. All code and models for the project can be found here.

References

[1] https://www.dsprelated.com/freebooks/filters/Series_Parallel_Transfer_Functions.html

[2] https://technobyte.org/properties-discrete-fourier-transform-summary-proofs/