# Digital Guitar Pedal Filter Project

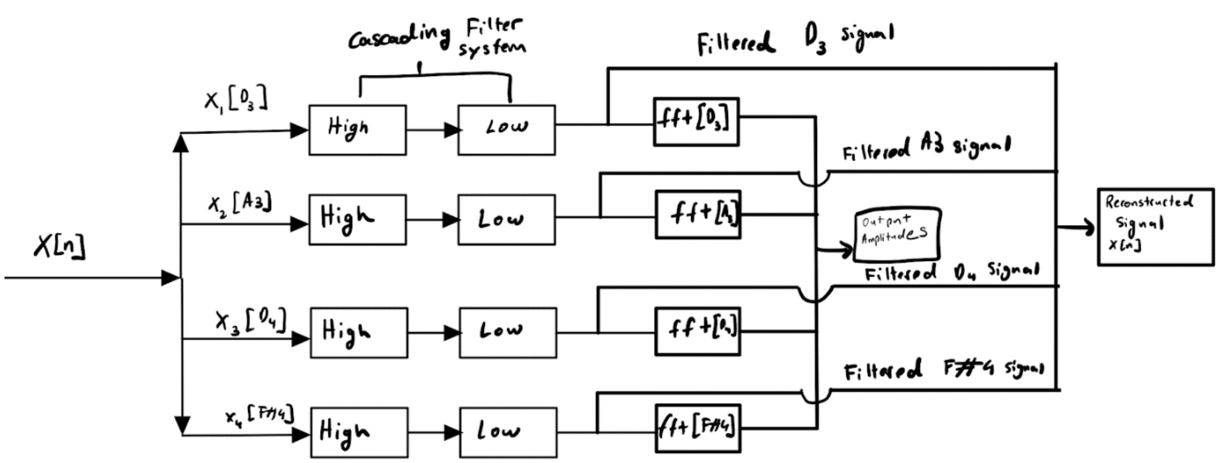
**Anirudh Devarakonda**

**Introduction**

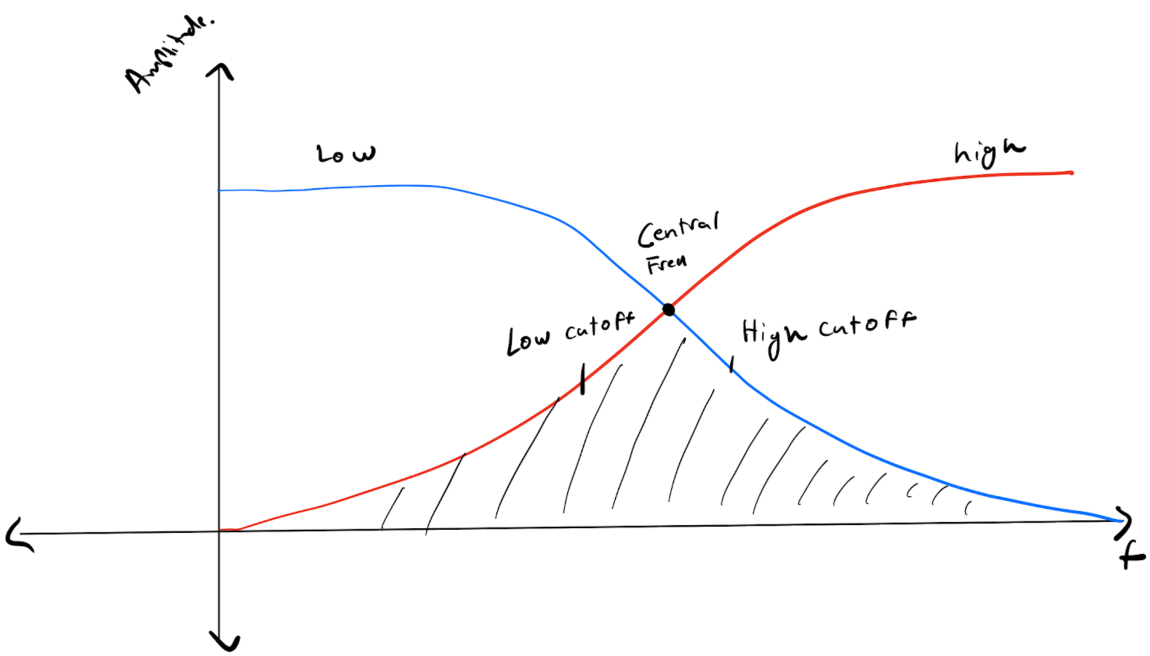
In this project, we are creating a compressor that filters specific frequencies from an input audio signal created from a guitar instrument. Guitar pedals are typically used to filter the sound and frequencies of an electric or acoustic guitar. They are located on the ground, and signal is picked up by a microphone of an electromagnetic pickup, after then it is put through the guitar pedal(s), and then into the amplifier or a speaker.

In this project, we isolate the audio signal into 4 pitches(D3, A3,D4,F#4). After we isolate the input signal into those 4 frequencies, we then reconstruct the signal by adding those filtered signals, and a reconstructed signal is produced. We pass the signal into 4 Bandpass filters for the 4 frequencies =, and this will be explained in detail further.

## Block Diagram and Flowchart logic



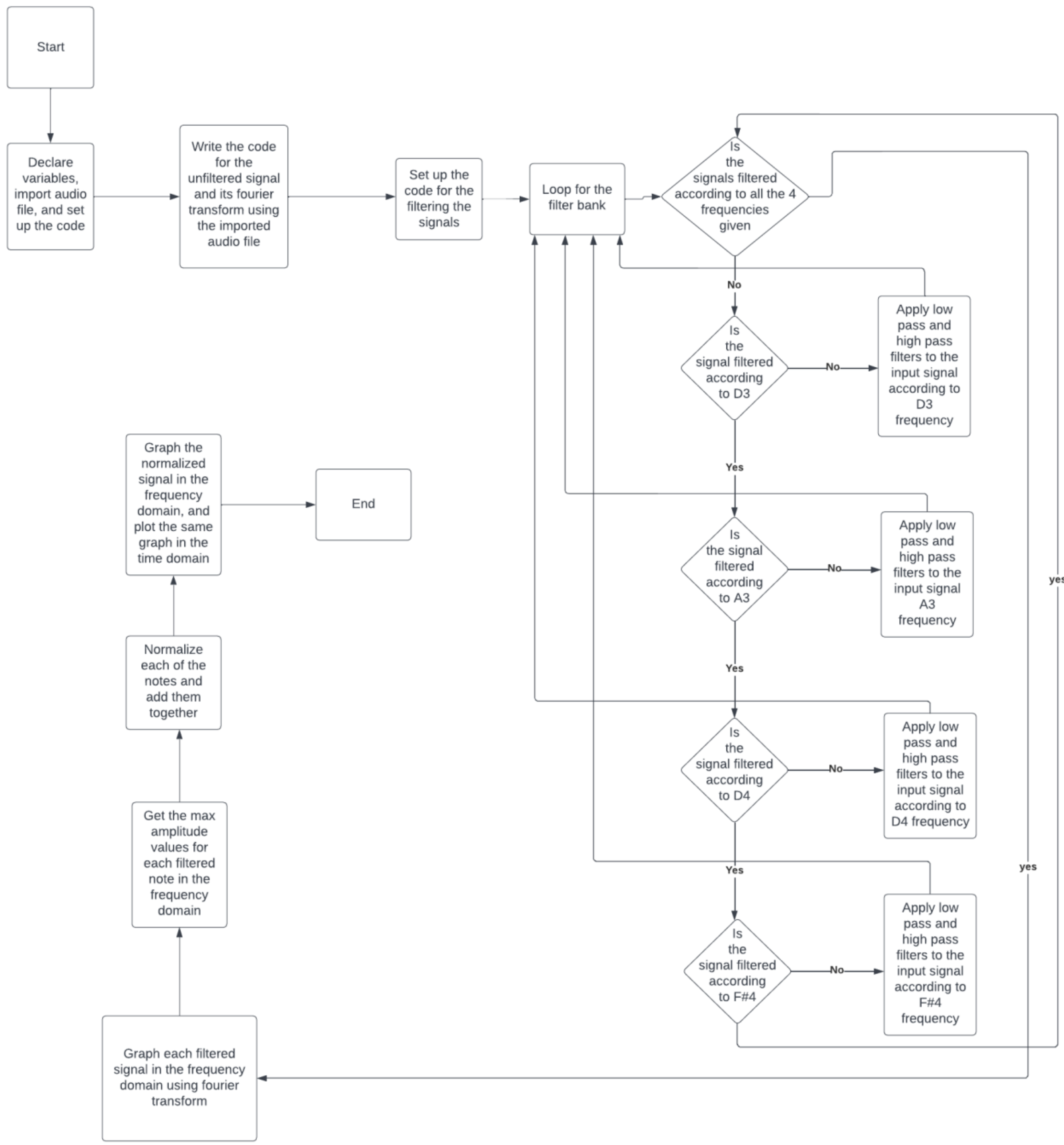
In this block diagram, the input audio signal which is X[n] is passed through a parallel system of bandpass filters. The bandpass filters used in this project is made up of a Chebyshev 1 filter. Within each row of the bandpass filter, is a cascading subsystem of Chebyshev 1 filters that is made up of a high pass cheby1 filter and a cheby1 low pass filter. The reason we setup the bandpass filters as a cascading system can be described by the graph below.



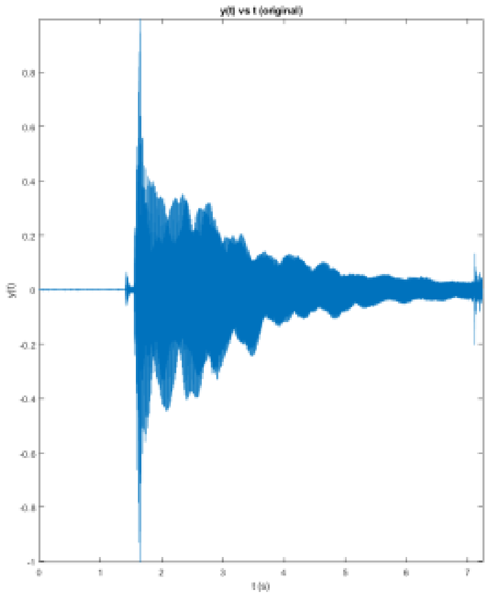

As you can see in the graph above, the bandpass filter can be made using a cascaded high pass and low pass. The low cutoff is for the high pass, and high cut off is for the low pass filter. The range of frequencies that lie between the cutoff frequencies created by the cascaded filters will be the frequencies of interest for our filters. The cascade of high pass and low pass filters is can be considered as convolution of the two filtered signals , and the low cutoff and high cutoff will be the range of the bandwidth for the bandpass filter.  Once we have isolated the input signal through the 4 rows of cascaded subsystems, we can then reconstruct our input signal based by adding the filtered signals based on the Fs that we sampled with.

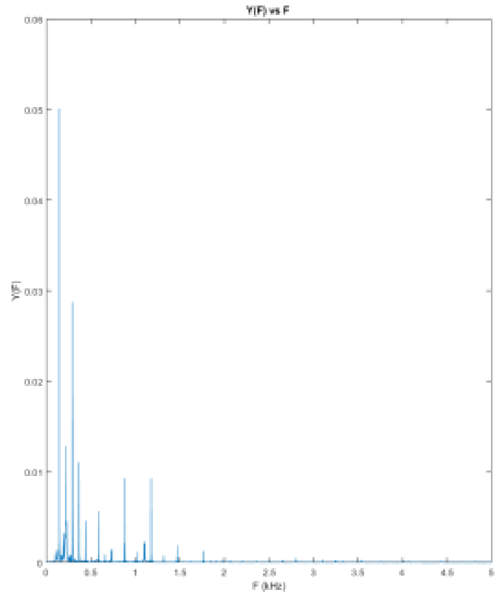
Based on the output filtered signals we can then take the Fourier transform of the signals to analyze the signal in the frequency domain. We need to record the amplitudes of the sum of all the filtered signals in the frequency domain so we can provide the graph of the final compressed(normalized) signal in the frequency domain.

To explain the logic behind the process in detail, below is a flowchart that shows every step and decision taken in the process of calculating and plotting the compression output signal from the input signal.

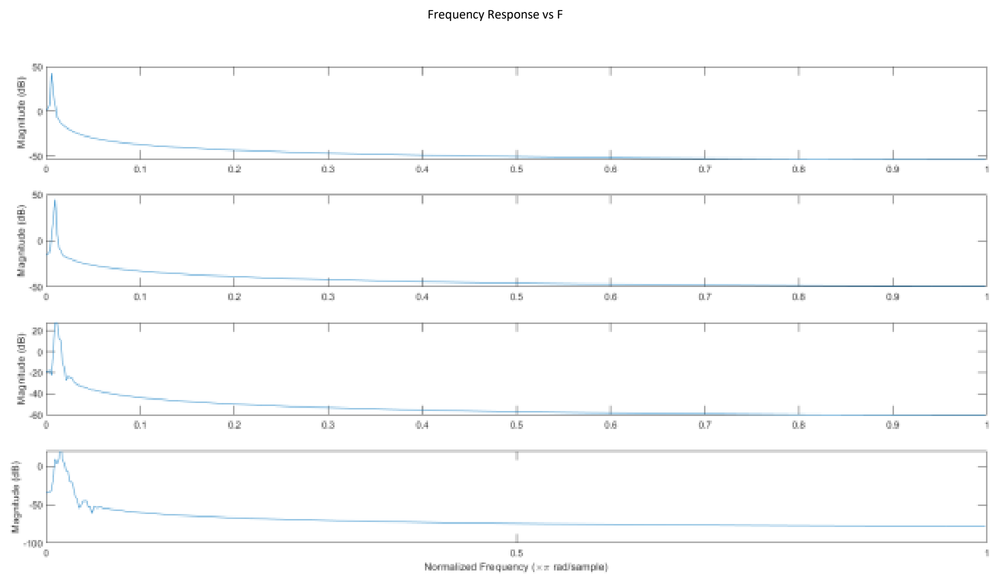

Start
Declare variables, import audio file, and set up the code
Write the code for the unfiltered signal and its fourier transform using the imported audio file
Set up the code for the filtering the signals
Loop for the filter bank
Is the signals filtered according to all the 4 frequencies given
No
Is the signal filtered according to D3
No
Apply low pass and high pass filters to the input signal according to D3 frequency
Yes
Is the signal filtered according to A3
No
Apply low pass and high pass filters to the input signal A3 frequency
Yes
Is the signal filtered according to D4
No
Apply low pass and high pass filters to the input signal according to D4 frequency
Yes
Is the signal filtered according to F#4
No
Apply low pass and high pass filters to the input signal according to F#4 frequency
yes
yes
Graph each filtered signal in the frequency domain using fourier transform
Get the max amplitude values for each filtered note in the frequency domain
Normalize each of the notes and add them together
Graph the normalized signal in the frequency domain, and plot the same graph in the time domain
End

**Analysis of figures**




These two figures represent the unfiltered audio signal in the time domain and the frequency domain respectively. From the time domain signal, we can analyze that the amplitude of the signal reduces as time progresses. The strength of the signal decreases as time increase, thus the volume of the sound decreases. From the FFT of the audio signal, we can observe the max density of the frequencies are located near the origin. We can also infer that the maximum frequencies present in the audio signal is from 0 kHz to 1 kHz. The unfiltered frequency diagram shows all the frequencies present in the audio signal and gives us a better picture to navigate through the frequencies we want to isolate.

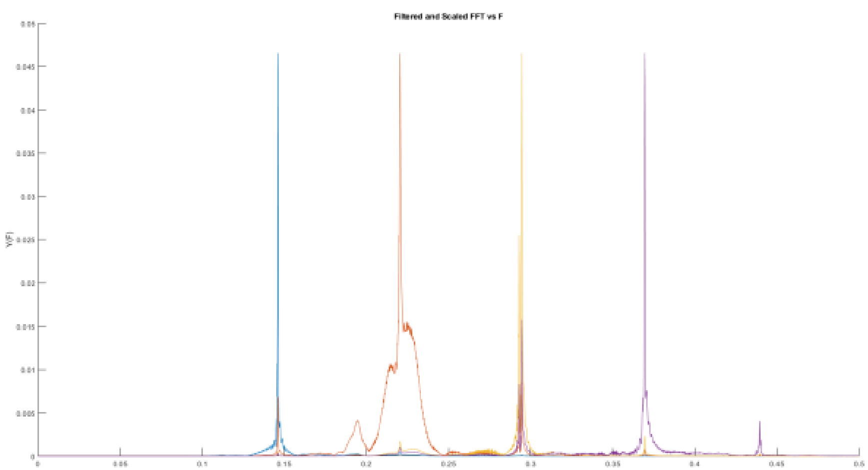



This image is the series of frequency response plots for each of the filters. Each graph represents the frequency response for each of the isolated signal through the bandpass filter. The graphs shows that only a specific frequency (D3,A3,D4,F#4) is isolated from the input signal. The bandpass filter uses a cascaded high pass low pass filter to extract the content and displays the magnitude corresponding to the that frequency of interest. The order of the graphs ranges from (D3,A3,D4,F#4). D3 has the largest amplitude , where it will be used as the max peak for the scaling factor. Below is the record of the amplitudes of the filtered signals,

```
>> max_amplitude

max_amplitude =

    0.0466    0.0114    0.0250    0.0095

>> norm_factor

norm_factor =

    1.0000    4.0676    1.8609    4.9070
```

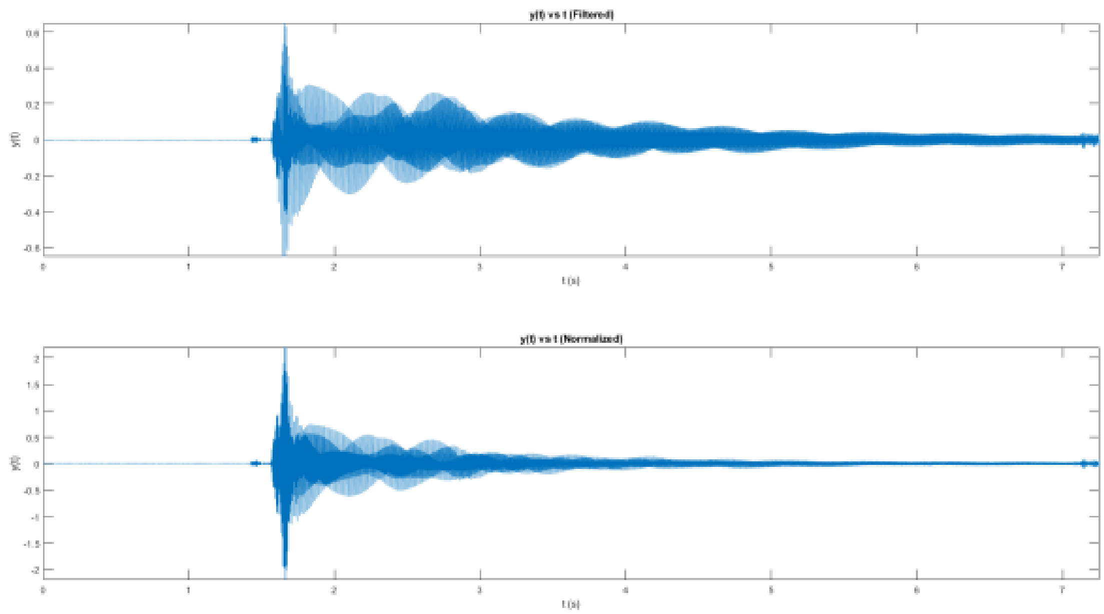
The max amplitude is 0.0466 whereas 0.0095 is the smallest. Therefore, since D3 filtered frequency has the largest amplitude it will be used as in the normalizing factor for the other values of the amplitudes. We must use the max peak amplitude because we are trying to compress the signal, for all the values of the amplitudes are within 10% within each other.

To calculate and plot the compressed signal in the frequency domain, there are a few steps that must be taken to perform this operation. Firstly, we must calculate the Fourier transform for each of the filtered signals, and then calculate the discrete Fourier transform for those signals. Normalizing the signals in the time domain will not be normalized in the frequency domain. If a signal is not normalized and you take the fft of it, its amplitudes are 1/2 not one when the fft is taken in the time domain. Amplitudes in time domain are not related to amplitudes in the frequency domain, thus the fft must be done first before plotting for the compressed normalized signal. The normalization factor is calculated by dividing the max amplitude for each filtered signal by the largest amplitude among the 4 signals. The filtered signals' Fourier transform were then multiplied by the normalization factor one by one, and the graph shown below is the output.




This graph showcases the sum of amplitudes of each of the filtered signal in the frequency domain. The steps above were taken precisely to get an output that is within 10% of each other.

The figure above is in the frequency domain, and we plot the same graph in the time domain highlighting the compressed signal of the 4 isolated frequencies in the time domain. Below is the graph of the filtered signal shown in the time domain as control, and the compressed filtered signal is also in the time domain. As you can see the normalized signal has a larger pitch and amplitude than the unscaled therefore the amplitude is higher, and the signal is evened out. A compressor's role is to increase the sounds of low volume sounds, and decrease sounds of high volume, therefore giving the audio a more filtered and evened out tone. When you play something very quietly, a compressor can boost the output to make it more audible. If we compare the two signals to the original unfiltered signal, we can see that the filtered and normalized signal in the time domain is way smoother and less rough, which means there is less noise and the sound has a more even pitch to it, thus making it more audible.




**Justification for the Filter Parameters and Compression specifications**

There are three parameters that were used for this project. The order of the Chebyshev filter, the ripple frequency of the filter, and the delta cutoff for the high and low pass cutoff frequency from the central frequency. The reason why I used the specific parameters of 3 for the delta cutoff, 6 for the order of the filter, and 1 for the ripple frequency is mostly due to trial and testing. These values were a result of several iterations of trial and testing to see which set of values for these specific parameters were the best to yield the most optimal compression signal from the 4 isolated frequencies. These filters have a high Q factor because they have a narrow passband of allowed frequencies for each filtered signal. Each filter in each row will only allow a certain range of frequency close to the central frequency(ie D3, A3,..etc). The high Q factor for each of the bandpass filters ensures that only a narrow range of frequencies including the cutoffs will be allowed to pass through.

A narrowband filter used for the bandpass in this project limits the bandwidth of the output signal to the band allocated for the transmission.

The design meets the specifications for compression. Compression in a guitar pedal mean that it makes less audible noises more audible and enhances certain frequencies that is dull in amplitude. If you can see the graph above, that shows the uncompressed filtered signal in the time domain vs the compressed filtered signal in the time domain, you can see the compressed signal has a larger amplitude than the uncompressed signal. If you also analyze the signal in the frequency domain, all the amplitudes are within 10% of each other thus satisfying that design specification, with each normalized amplitude being around 0.045.

## Conclusion

Guitar compressors are wonderful equipment that can make less audible tones more prominent and enhances the quality for music an artist produces. This project showcases how valuable the study and analysis of signals are and how versatile it is that it can be applied a variety of fields.