Getting Rid of Unwanted Noise: Design of a Multi-Band Equalizer

David Armstrong, *Team Member*, Kaan Dincer, *Team Member*

***Abstract*—A problem addressed in this study is that audio clips often have unwanted noises. The goal of this study was to create a multi-band equalizer (EQ) with user control over the gain on each band to filter out these unwanted noises. This EQ was used on three audio signals to remove noisiness or otherwise manipulate the sound. We analyze the effect on the frequency domain, vary the crosstalk between bands, and look at the impulse response of the overall system. The results were a successful processing of the audio signals to our liking. Our design includes three bandpass filters, a lowpass filter, and a highpass filter to attenuate unwanted frequencies and highlight the wanted ones.**

I. Background

A main objective in sound mixing is adjusting frequencies of audio clips to change the clip to the sound engineer's liking. Audio processing is very common and enhances content all over our day to day lives. This is the challenge we faced in this case study: creating a tool, the equalizer, to enhance various audio clips. Our three audio signals used in this case study are a violin playing a G major scale, Ode to Joy, and a signal we made ourselves of a synthesizer playing ascending notes. In the following sections you will see how we implement our EQ to control the frequencies of the filtered signals.

II. Ode To Joy & Violin Signals

A. *EQ Creation*

Our implementation uses three different built-in MATLAB commands to attenuate the different frequency bands. We use the functions `lowpass()`, `bandpass()`, and `highpass()` set to the frequency ranges between our variables labeled cutoff1-cutoff4 to process the signal in parallel. The original filter is sent independently through the lowpass (hZ<`cutoff1`), highpass (hZ>`cutoff4`), and three bandpass (`cutoff1`<hZ<`cutoff2`, etc.) filters. Each filtered signal is scaled by their respective entry in the `gain` vector and then added together to make the composite final result of the EQ. The final component of our filter was a variable called `factor` that spaces the bands apart from each other as the variable increases to minimize crosstalk. Equations (1) and (2) describe the parallel structure of our signal processing:

$$V_{out\_n} = filter(V_{in}, Cutoff \pm factor, fs) \quad (1)$$

$$V_{Out\_Final} = \sum_{n=1}^{5} Gain_n * V_{Out\_n} \quad (2)$$

Our initial attempt for this case study involved scaling Fourier coefficients in the frequency domain to create perfect block filters. This was unsuccessful because we had not considered the side effects that multiplying in the frequency domain convolves the inverse Fourier in the time domain. The implementation using the `__pass()` commands proved to be effective and simpler to analyze.

B. *Bode Plots*

The following plots show the Bode plots for each of the five filters using the following parameters:

```
cutoff1 = 800;
cutoff2 = 1800;
cutoff3 = 3000;
cutoff4 = 6000;
factor = 0;
```

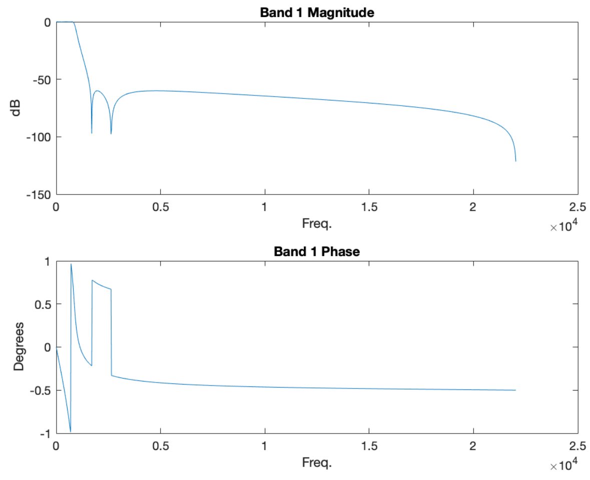

Fig. 1. Bode plot for the lowpass filter with a cutoff frequency of 800hZ.

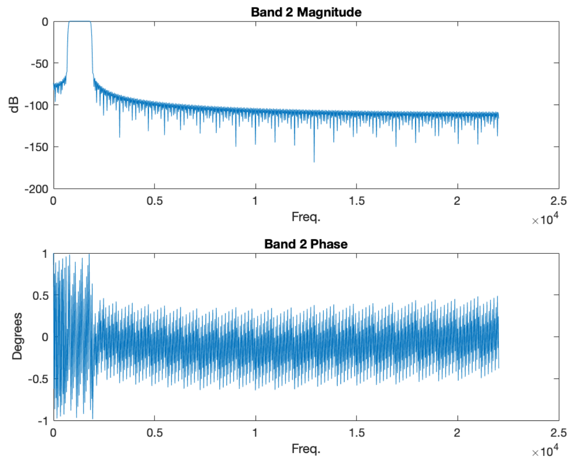

Fig. 2. Bode plot for the first bandpass filter from cutoff1 (800) to cutoff2 (1800).

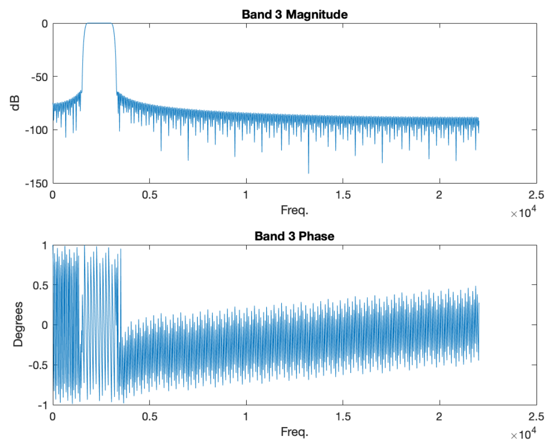

Fig. 3. Bode plot for the second bandpass filter from cutoff2 (1800) to cutoff3 (3000).

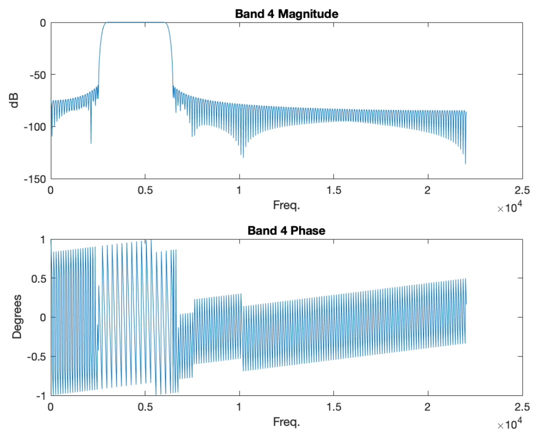

Fig. 4. Bode plot for the third bandpass filter from cutoff3 (3000) to cutoff4 (6000).

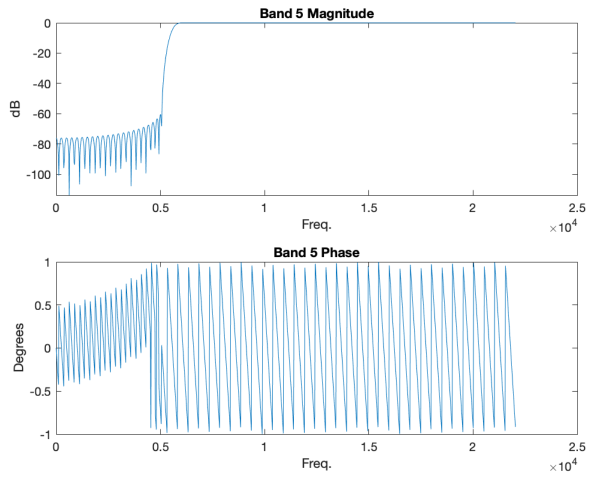

Fig. 5. Bode plot for the highpass signal with a cutoff of 6000hZ.

These values were chosen because most of the noise in the two provided signals appeared in the 0-800hZ range, so we attenuated the lowest two bands of frequencies and left the rest at their original volume. The crosstalk variable "factor" was left at zero as a baseline and the cutoffs 2 through 4 were gradually spaced out across the standard audible frequency range, similar to an industry-standard equalizer.

The Bode plots were formed by finding the frequency response of each band using the `freqz()` command and then calculating the magnitude or phase of the resulting frequency response vector.

C. *Impulse Response*

The impulse response for each filter can be seen in the following plot. The FFT of the impulse response shows us the effect on the frequency domain of each filter on its own. The sum of these signals is what the impulse response would be for the entire EQ.

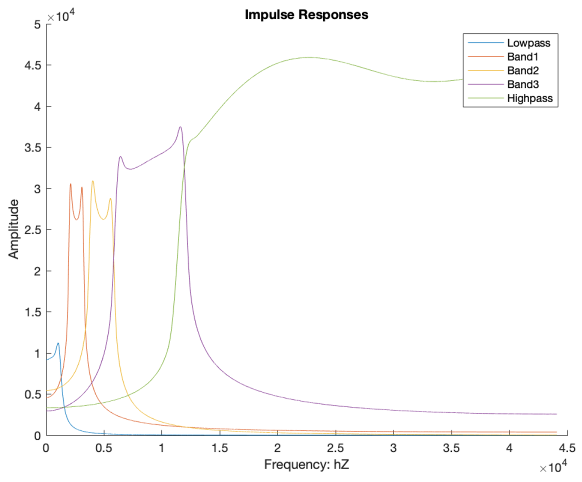

Fig. 6. Impulse response for each of the five filters.

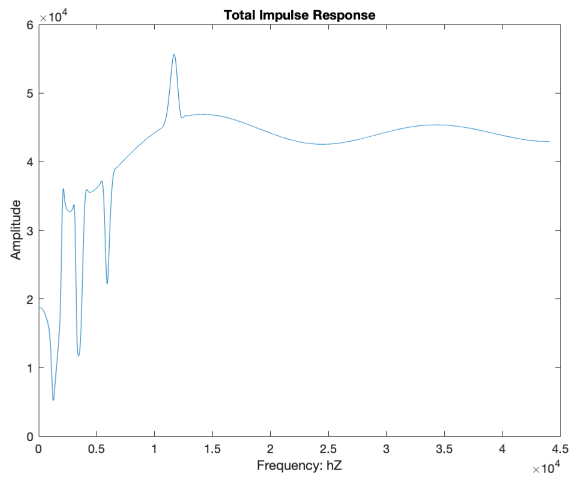

Fig. 7. Impulse response for each of the five filters added together.

We can see from these plots that a signal sent through each filter will be boosted by default, since almost the entire frequency range sits above an amplitude of 1. This should be taken into consideration when the user of this equalizer is choosing their gain values.

D. *Ode to Joy Spectrogram*

We know that our EQ is working as intended by listening to our audio, but also by inspecting the spectrogram of the audio signal before and after going through the EQ.

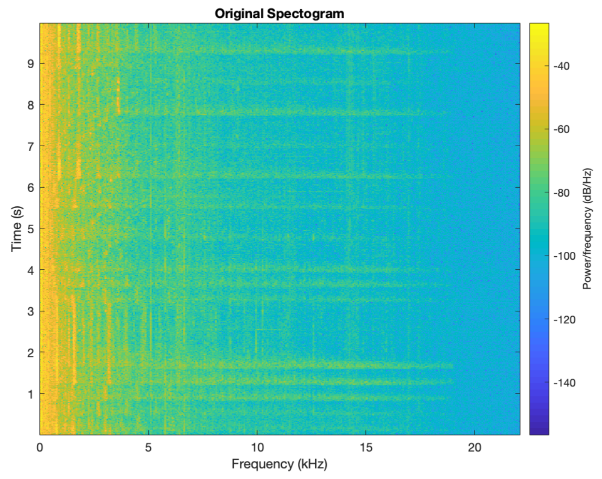

Fig. 8. The original spectrogram for the clip of "Ode to Joy," including noisiness at low frequencies.

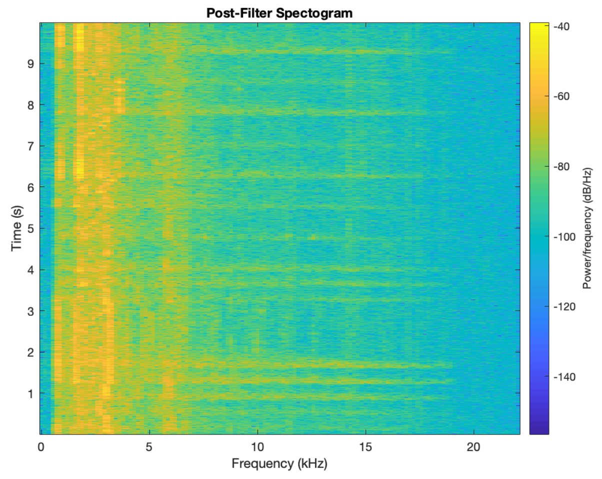

Fig. 9. The post-EQ spectrogram of "Ode to Joy" after attenuating the lowest frequencies, which unfortunately includes some musical information.

All that was needed to remove the noisiness from the "Ode to Joy" clip was bands one and two, which makes this filter amount to an overly complicated lowpass filter for this case. However, the point of this case study was to create an EQ that can easily adapt to the user's desires, which holds true for this code. The violin clip playing the G major scale also has some unneeded information at higher frequencies that we could remove. By adjusting our gain variable to [.001 .2 1 .2 .001], we create a filter that isolates the frequencies that the violin is creating to produce a crisper sound file.

E. Violin FFT

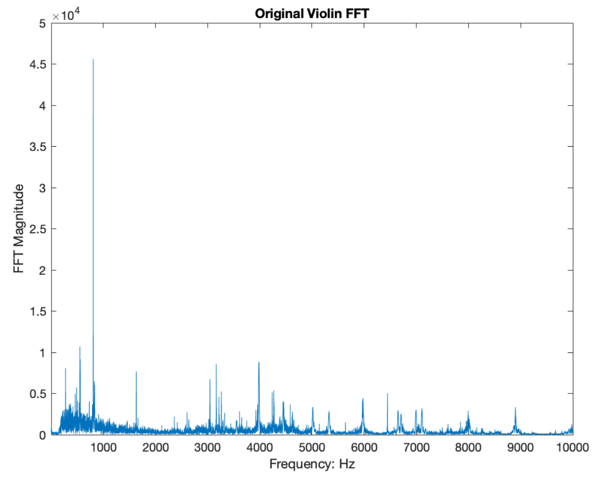

Fig. 10. Original FFT plot of the violin playing the G-major scale.

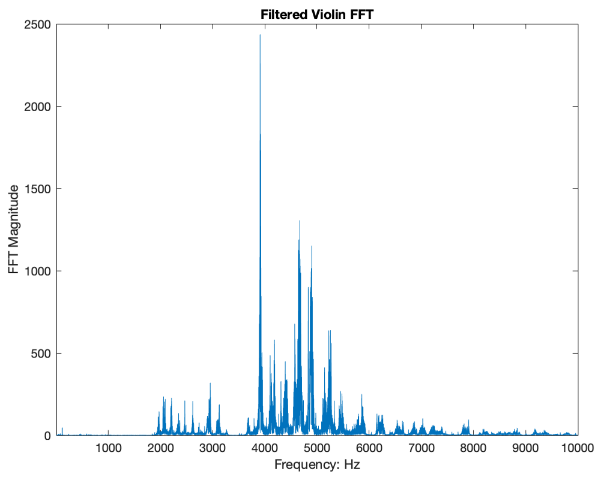

Fig. 11. After trial and error, the violin clip sounded subjectively the best when frequencies below 1800 hZ were attenuated as well as frequencies above 3000.

We can see that our filter attenuated the low and high frequencies, and even boosted the middle frequencies because of crosstalk.

F. Crosstalk

The crosstalk between the filters is the amount that one filter affects another filter's frequency band. We tackled this issue in two ways. One issue that we saw was that the lowpass has a very large effect on the first bandpass filter's frequency range. To address this, we added a steepness factor to the bandpass's code that minimizes its effects outside of the range from 0 to cutoff1. The second measure we took was implementing the crosstalk variable `factor` to separate the bands' overlap.

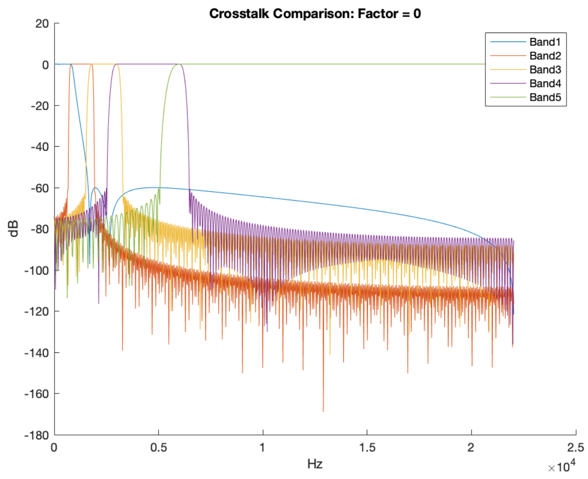

Fig. 12. When the crosstalk factor is 0, the bands are right next to each other and the slopes overlap and boost the adjacent frequencies of the signal.

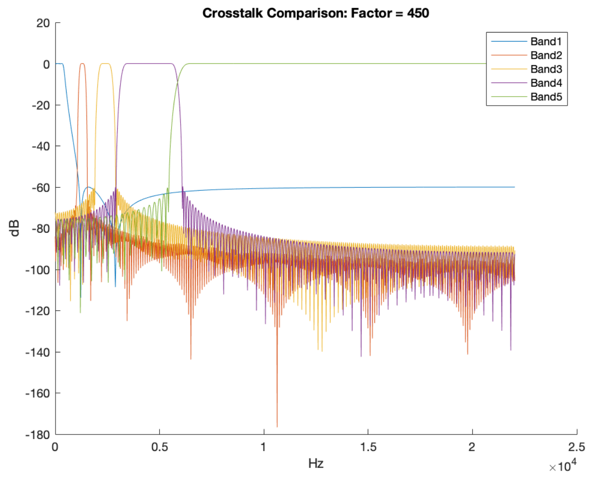

Fig. 13. When the crosstalk factor is raised to 450, the bands are very tight, so they have less of an effect on the other frequency ranges. This is seen mostly in the top-left quadrant of Fig. 12 and Fig. 13.

The ideal crosstalk factor would cause the sum a filtered signal with a gain of [1 1 1 1 1] come out of the EQ the same way it came in. However, our implementation stops at these two measures as an adequate method to have control over the crosstalk.

III. Original Signal

A. Decision

We decided to create our own audio signal in Apple's Logic Pro X to apply to the EQ we created in MATLAB. Logic calls the synthesizer we used "Soft Synth Bells." We chose this sound because it is an imperfect sinewave, so it is interesting to see but at the same time is still creates clear notes for the EQ to filter. The sound itself is an almost random mix of whole notes and half notes in an ascending fashion. This combination of sound timbre and composition turns our waveform into a sort of crude frequency response plot. This signal is sent through our EQ with the same cutoff values but a cutoff factor of 300 and a gain of [1 1 1 .01 1].

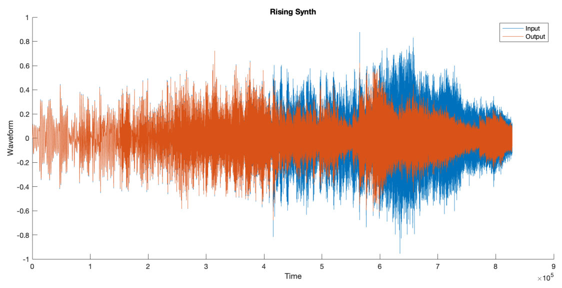

Fig. 1. Waveform of the filtered audio on top of the original audio. The attenuation of higher frequencies can be seen as the volume only decreases for the second half of the recording (the higher pitched notes).

The above figure shows that the notes are severely compressed once the sound starts filling the frequencies above cutoff3. Even though the note being played at that time is nowhere near that high of a frequency (3000), the timbre and harmonics of that note are much higher than the fundamental, causing the second half of the audio signal to be compressed.

IV. Conclusion

In this case study, we have created a multi-band equalizer that can read any waveform and adjust its frequencies with precision. The user of this equalizer can change the gain of each band, change the cutoff value between bands, change the space between bands, and the steepness of the lowpass, all just by changing some variables in the code. The plots shown in this report give an insightful look at the filters' true effect: the imperfections that occur in the slopes of the frequency responses and the crosstalk between bands.