Lab 4: Digital Filtering and Music Equalizer

University of Washington, Electrical Engineering

EE 341

14 August 2018

|  |  |  |
| --- | --- | --- |
| Kalvin Hallesy 1750416 | Irina Golub 1775424 | Chenglong Li 1429842 |

# 

# 1. Introduction

The purpose of this lab is to become acquainted with different types of digital filters, how they are implemented, and how they can be applied. In Part 2, we found that a difference equation digital filter provided a more desirable result than the same impulse response filter. In Part 3, we observed the differences between a low-pass FIR filter and a low-pass IIR Butterworth filter. We found that for data, the FIR filter was a better choice, for functions, the Butterworth filter was a better choice, and for music signals, there was no difference between the results after filtering with both filters.

# 2. Digital Filtering

A filter can be implemented using the difference equation and MATLAB’s ‘filter’ function, or using the filter impulse response and MATLAB’s ‘conv’ function. We will observe the result of both methods on a vector of data containing Microsoft stock price over 4 years.

## 2.1 Implementation

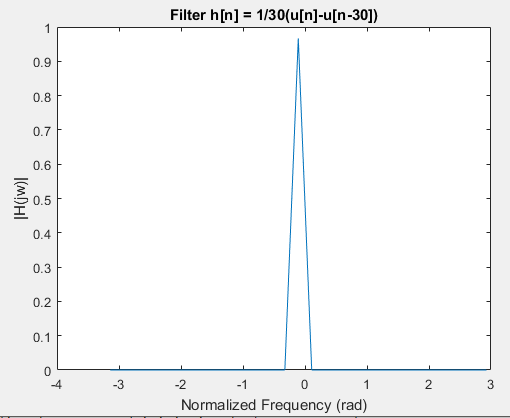
The stock price information was loaded from the .txt file using the ‘importdata’ command and plotted.

The difference equation of the filter we are using is y[n] = 1/30(x[n]+x[n1]+...+x[n-29]. The ‘filter function takes inputs of the a and b coefficients, as well as the function to be filtered. From the above y[n] equation, we can see that a0=30 but all other a values are 0. The ‘a’ vector is created with the first value of 30, horizontally concatenated with 29 zeros using MATLAB’s ‘zeros’ function. Next, we can see that all ‘b’ coefficients are 1 because there is no other multiplier in front of each x[n-k] value. The ‘b’ vector is initialized as a 30-value ‘ones’ vector. The ‘b’ vector, ‘a’ vector, and the stock information is input to the ‘filter’ function, and the output is the stock information with the filter applied. This result is plotted on the same graph as the unfiltered result.

The impulse response of our filter is h[n]=1/30(u[n]-u[n-30]). This filter is applied to the stock information data by using MATLAB’s ‘conv’ function.

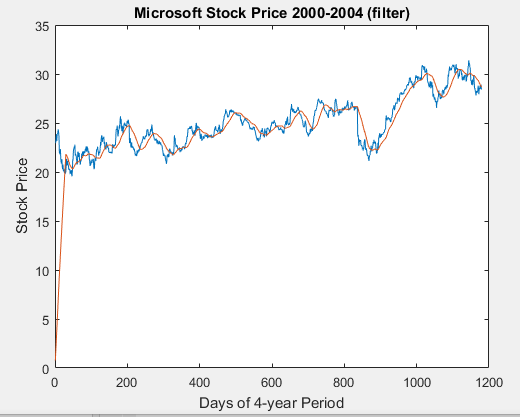
## 2.2 Results

Fig. 1 shows the plot of the magnitude of the shifted fft of h[n]. This plot has a gain of zero at high frequency values, so this function corresponds to a low-pass filter.



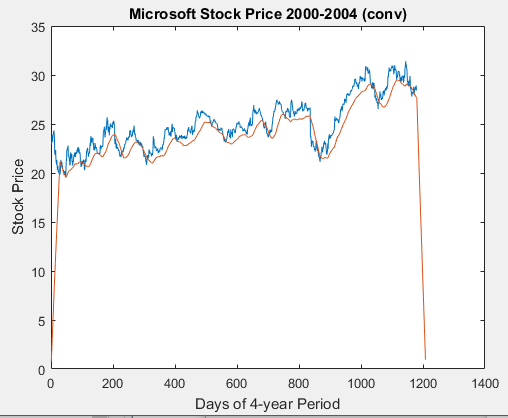
**Fig. 1: DTFT magnitude of h[n] plotted vs. normalized frequency in radians.**

Fig. 2 shows the original stock price information plotted in blue, with the filtered stock information plotted in red. The filter y[n] was applied using the ‘filter’ function.



**Fig. 2: Stock Price (blue) and filtered stock price (red) using the ‘filter’ function and y[n].**

Fig. 3 shows the original stock price information plotted in blue, with the filtered stock information plotted in red. The filter h[n] was applied using the ‘conv’ function.



**Fig. 3: Stock Price (blue) and filtered stock price (red) using the ‘conv’ function and ‘h[n]’.**

The most noticeable difference between these two sets of filtered data is that the ‘conv’ function outputs values of magnitude typically lower than the magnitude of the input. The ‘filter’ function outputs a function with magnitude more similar to the input function.

For the ‘conv’ function to output a function with more accurate magnitude, I would multiply the ‘conv’ output by 30/29 (in a general case, N/N-1) to increase the magnitude of the filtered output by a scaled amount.

# 3. Filter Characteristics

Finite Impulse Response (FIR) filters and Infinite Impulse Response (IIR) filters are implemented differently and can filter a signal using different methods.

## 3.1 Implementation

To create a low-pass FIR filter with order 10 and cutoff frequency of 0.3pi, we used the ‘fir1’ function with inputs of 10 and 0.3pi. The output of the function is the FIR filter coefficients.

To create a low-pass IIR Butterworth filter with order 10 and cutoff frequency 0.3pi, we used the ‘butter’ function with inputs of 10 and 0.3pi. The outputs of the ‘butter’ function are the ‘a’ and ‘b’ filter coefficients.

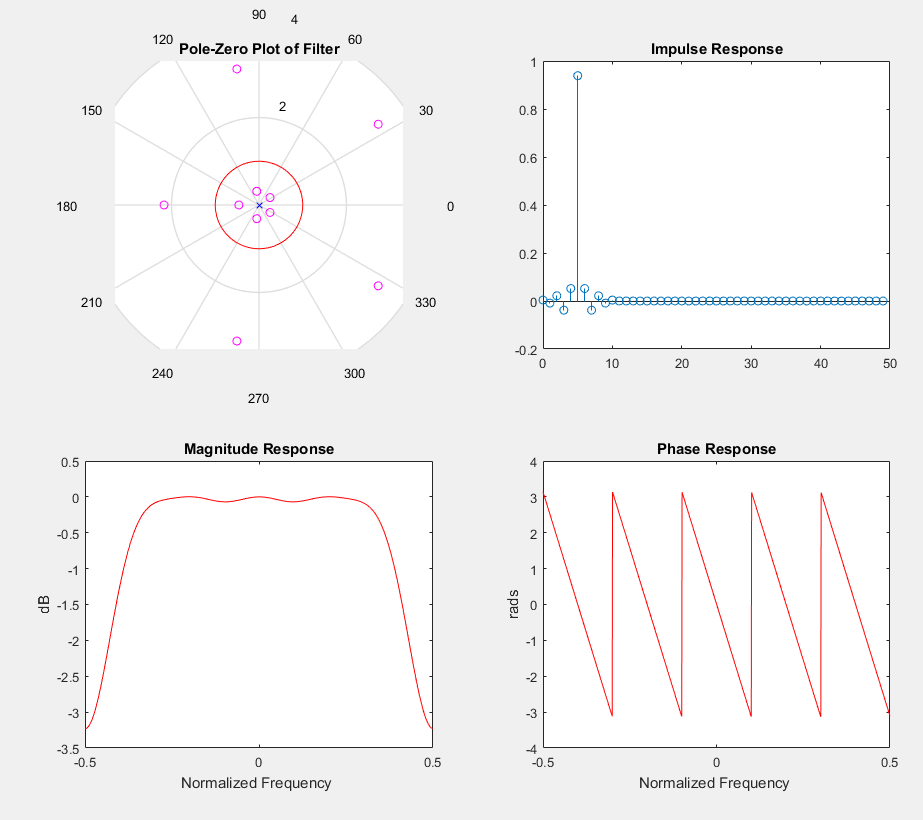
These two filters were both applied to a signal using the ‘filter’ function. In the case of the FIR filter, the ‘b’ input is the ‘fir1’ output vector and the ‘a’ input is 1. In the case of the Butterworth filter, the ‘a’ and ‘b’ inputs are given as the output of the ‘butter’ function. The filters are applied to three different signals:

1. The stock market data from Part 2.
2. A pulse of length twenty: x[n] = u[n]-u[n-20] with total length n=60.
3. A music file.

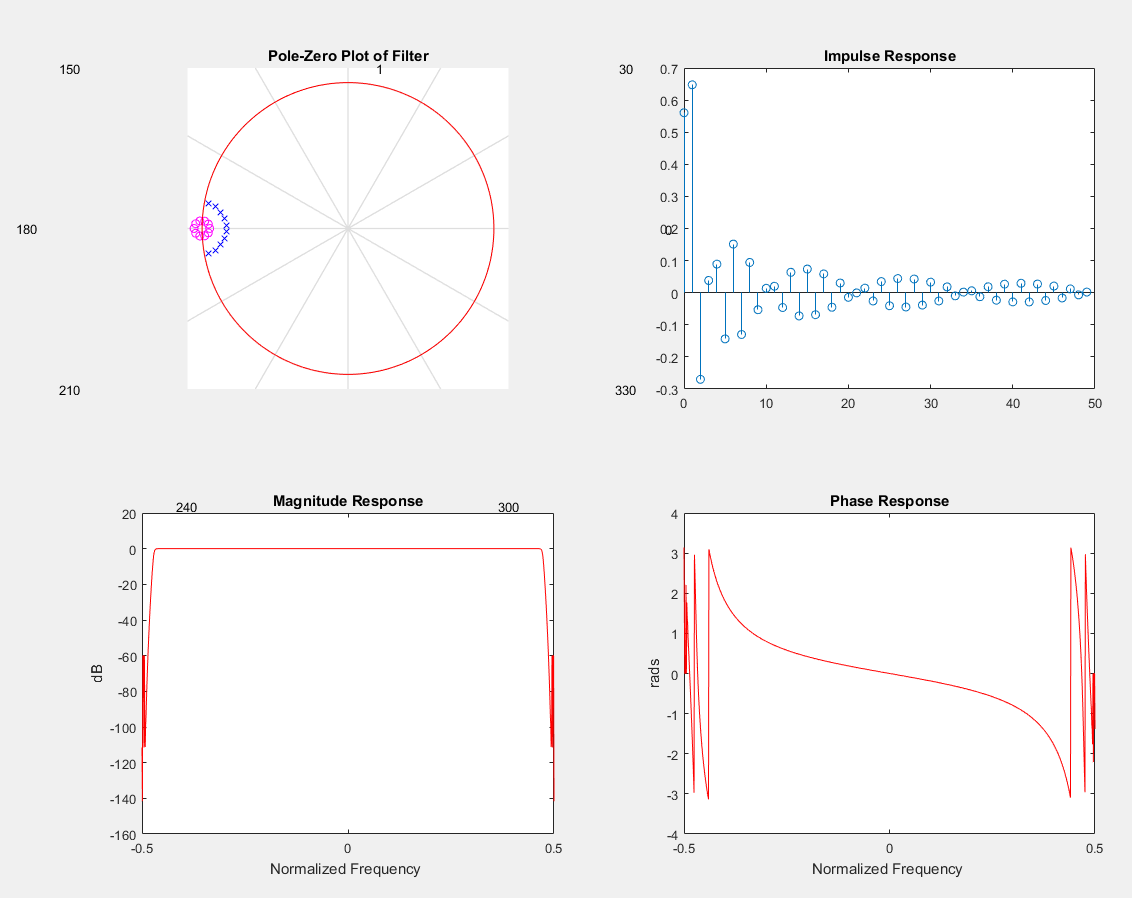
X[n] was created as a vector of 19 ones horizontally concatenated with a vector of 40 zeros. The music file was read using the ‘audioread’ function, and then adding the columns together to form one music channel. Each input signal and both filter outputs were plotted.

## 3.2 Results

Fig. 4 shows the ‘frevalz01’ function analysis of the low-pass FIR filter, and Fig. 5 shows the ‘frevals01’ function analysis of the low-pass Butterworth filter.



**Fig. 4: Analysis of the FIR low-pass filter using ‘frevalz01’.**

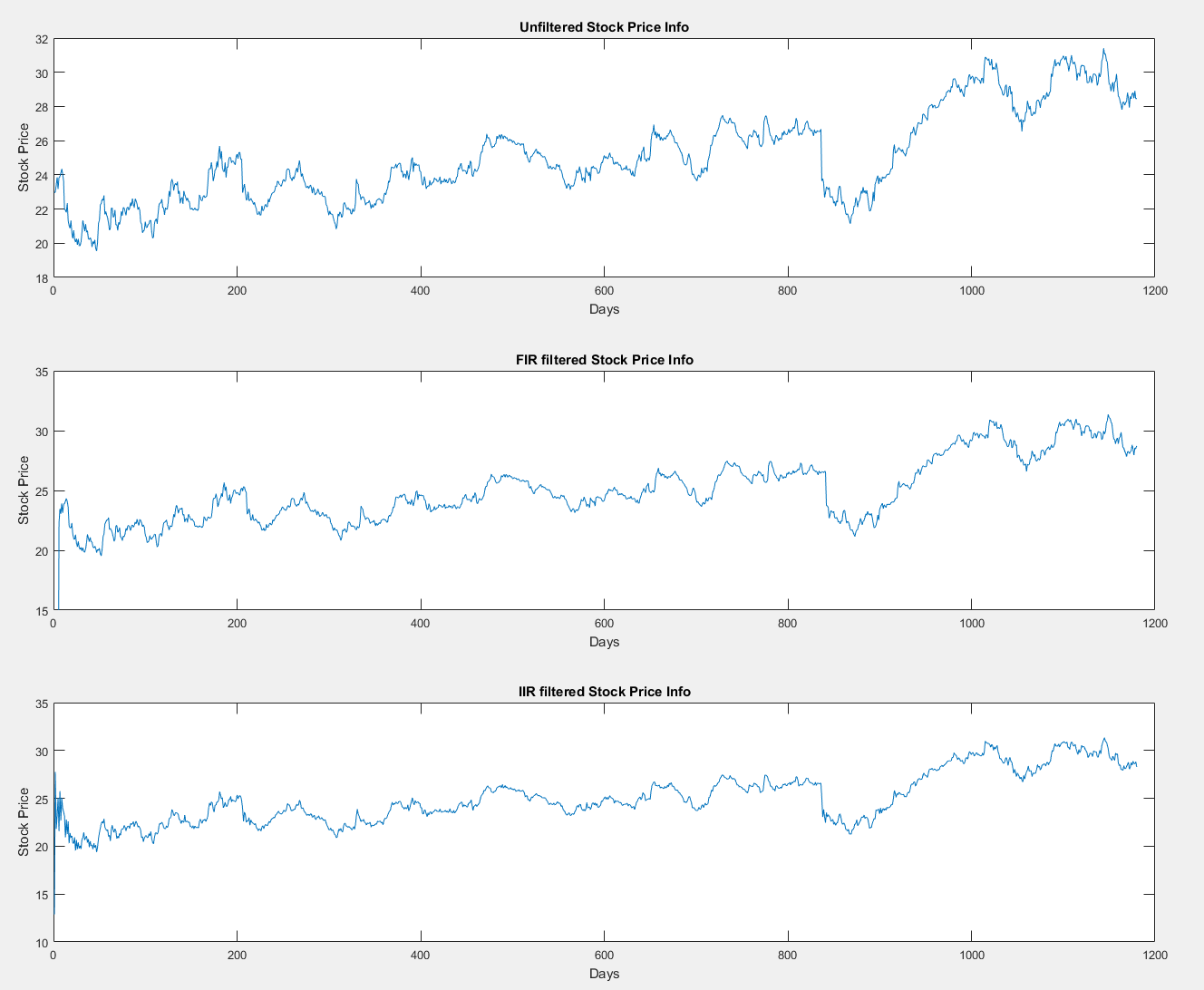
****

**Fig. 5: Analysis of the Butterworth low-pass filter using ‘frevalz01’.**

The magnitude response of each filter shows that the Butterworth filter has a much steeper cutoff than the FIR filter. In an ideal filter, the slope of the magnitude at the cutoff frequencies is infinite - the Butterworth filter achieves an effect closer to this, so the Butterworth filter is closer to an ideal filter than the FIR filter. Because of this, we predicted that a signal filtered with the low-pass Butterworth filter will appear smoother than one filtered with a low-pass FIR filter.

The phase response of each filter shows that the FIR filter has a periodic phase response of linear components, while the phase response of the Butterworth filter has a phase response that is smooth at low frequencies but erratic for high frequencies. Thus, we can expect the time-shifting effect of the FIR filter to be steady, but for the Butterworth filter, the effect will be different on low-frequency and high-frequency components.

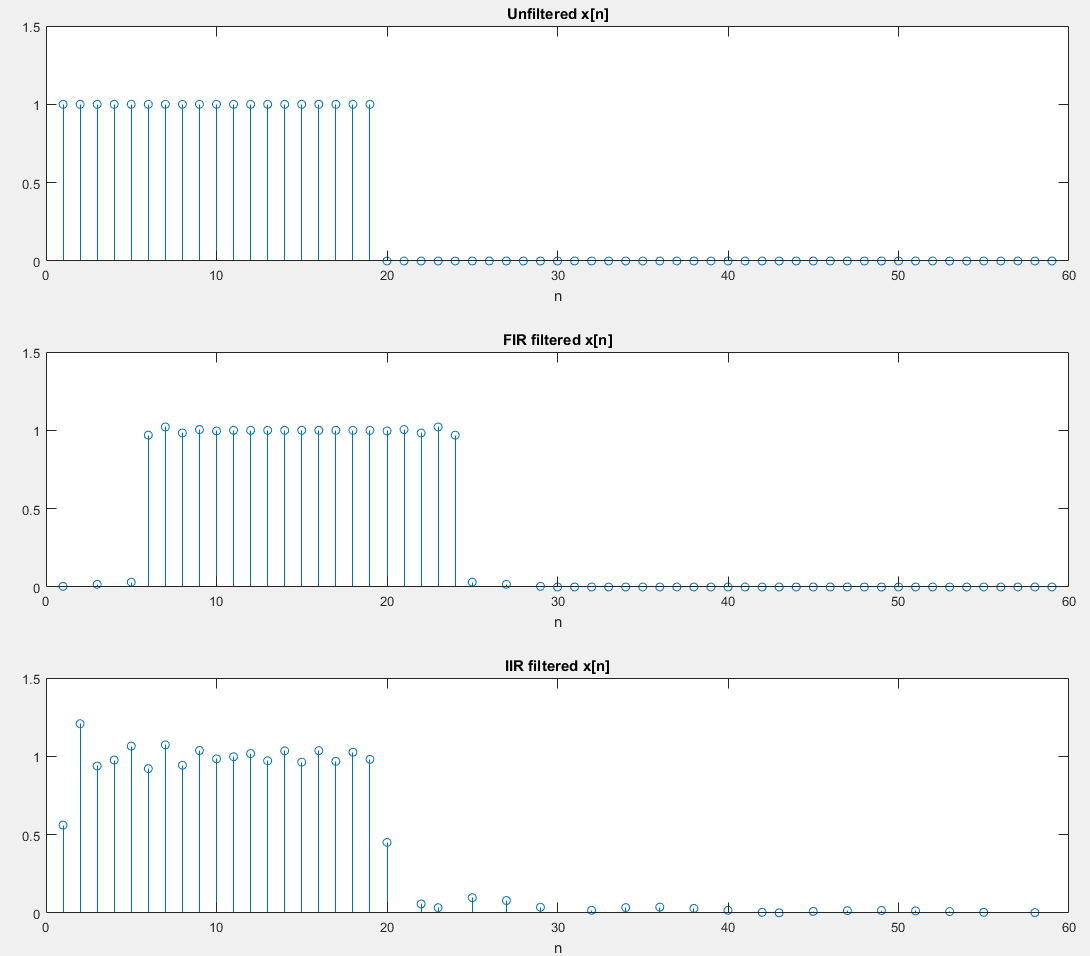
Fig. 6 shows the plot of the stock price information unfiltered, passed through an FIR filter, and passed through a Butterworth filter.



**Fig. 6: Original Stock Price Plot (top), after FIR filter (center), and after Butterworth filter (bottom).**

There is not much of a difference between the two filter outputs, except that the Butterworth IIR filter causes the output to behave undesirably at the beginning of the signal. In this case, the FIR filter is a better low-pass filter method.

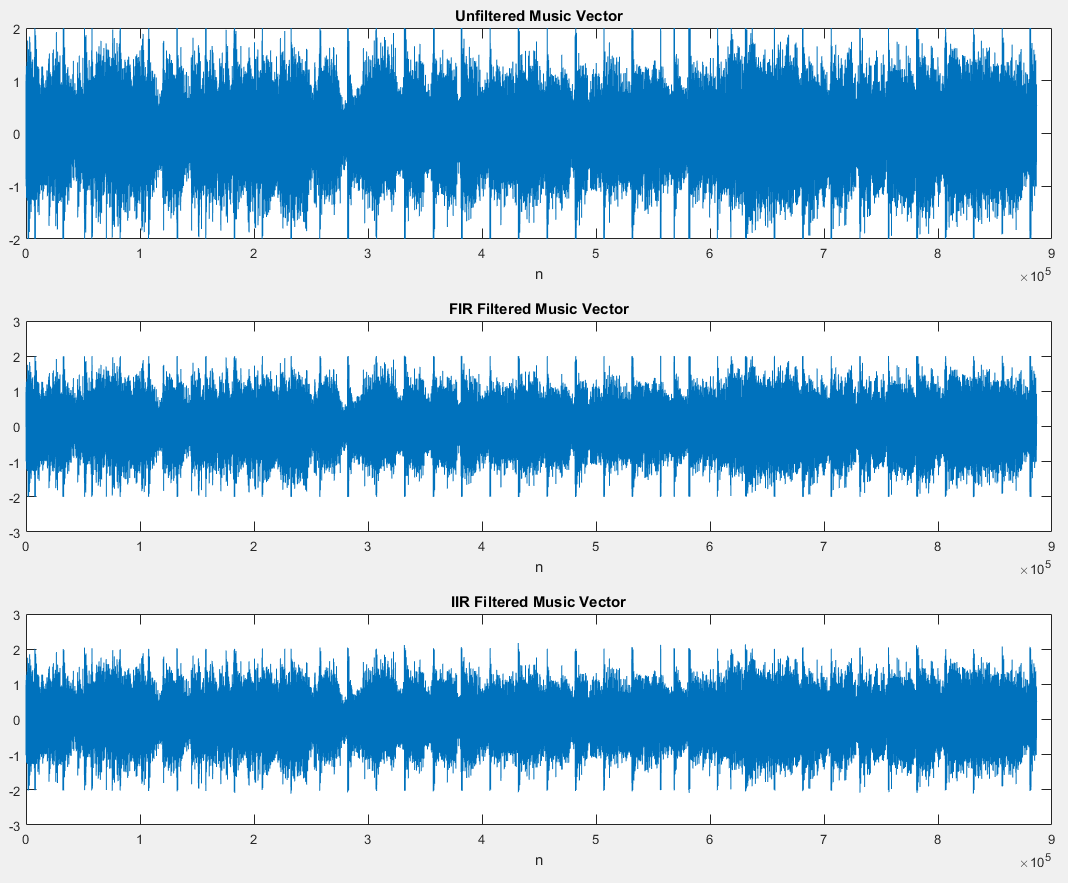
Fig. 7 shows the pulse x[n] unfiltered, passed through an FIR filter, and passed through a Butterworth filter.



**Fig. 7: Original x[n] (top), after FIR filter (center), and after Butterworth filter (bottom).**

The FIR filter time-shifted the signal by n = 4 to the right. The Butterworth filter caused ripples in the signal output. In the time-domain, a low-pass filter should smooth out sharp transitions in magnitude, so these results show that the Butterworth filter is a better choice for filtering x[n].

Fig. 8 shows the music vector unfiltered, passed through an FIR filter, and passed through a Butterworth filter.



**Fig. 8: Original music vector (top),music vector after FIR filter (center), and music vector after Butterworth filter (bottom).**

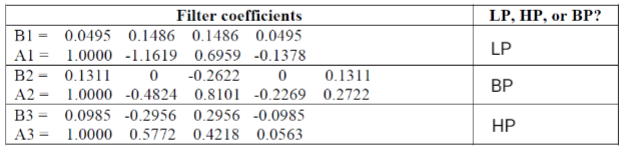
In this case, there is not much visible nor audible difference between the two filters application to the music vector, and we can conclude that either filtering method would be suitable for music filtering.

# 4. Music Equalizer

Equalization is a technique used in music to isolate different bands of frequencies from a signal and adjust their individual gains. This is one of the most important steps in getting music to sound good whilst also allowing for personal customizations to the sound such as bass-boosting. In this section, we describe our implementation of a simple three-band equalizer that allows for gain adjustments of low frequencies, mid frequencies, and high frequencies individually. We then discuss the effects of adjusting the gain of different bands on a song.

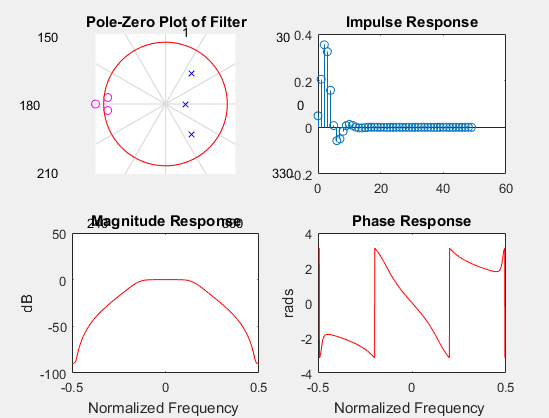
## 4.1 Implementation

To make the three-band equalizer, we first had to establish three different filters to isolate different frequency bands. We used used the coefficients in figure 9 to create a low pass, band pass, and high pass filter.

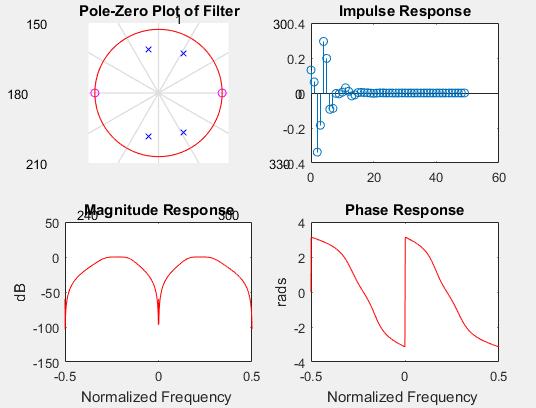


**Fig. 9: Filter Coefficients for 3-band equalizer**

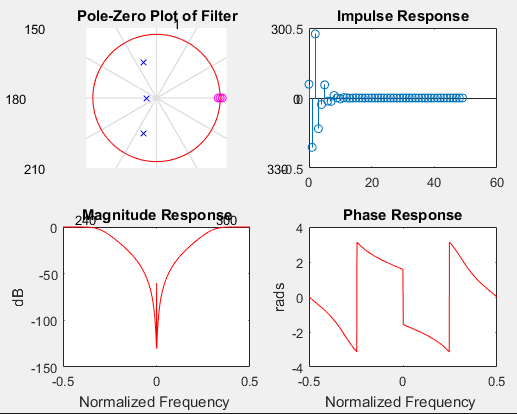
An analysis of each filter with ‘frevalz01’ is shown in the following three figures. Figure 10 shows the low pass analysis, figure 11 shows the bandpass analysis, and figure 12 shows the high pass analysis.



**Fig. 10: Analysis of the low-pass filter**



**Fig. 11: Analysis of the bandpass filter**



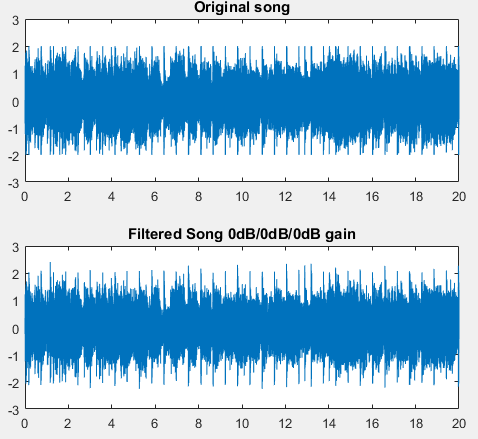
**Fig. 12: Analysis of the high-pass filter**

To turn these filters into an equalizer, we created a function called ‘equalize’ which takes input arguments of the signal and the dB gain for each band. The signal then is then passed through each filter individually to create 3 different filtered versions of the original signal. These filtered versions of the signal are multiplied by the appropriate ratio for the given dB gain input. This ratio is given by the equation . The final equalized signal output is then obtained by adding these three resultant filtered signals.

## 4.2 Results

Our equalizer successfully isolates and applies separate gains to the three frequency bands and produces the expected outputs. The equalizer output was tested in four cases, a gain of 0dB all three bands and a gain of 0dB on each individual band with the other two muted.

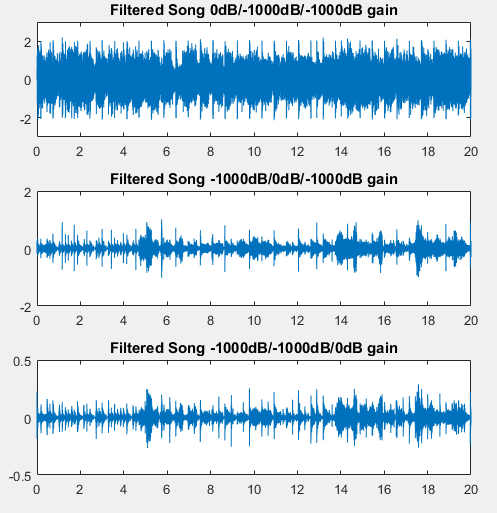
The first test was using a gain of 0dB in all three bands. A perfect equalizer would produce the exact same signal that was put in. A comparison of the original vs equalized waveforms are shown in figure 12:



**Fig. 11: Song signal before and after 0dB-low/0dB-mid/0dB-high equalization**

The waveforms of these two signals are almost identical and sound the same when played through speakers.

The next test was to output only one band at a time. This was achieved by setting the gain of one band to 0dB and the others to -1000dB (to make the wave approximately 0) for each band. A comparison of the three output different output equalization signals is shown in figure 12:



**Fig. 12: Song signal with isolated low, band, and high filters (top to bottom)**

Figure 12 shows that the song is composed mostly of lower frequencies, some middle frequencies, and a very small amount of high frequencies from looking at the amplitude and density of each signal.

The results of playing these waves through speakers are as expected as well. Since the majority of the song is low frequency, playing the isolated low band version of the song sounds largely like the original but with much less clarity and detail since details tend to be carried in higher frequencies. The middle band-passed version sounds very empty in comparison as it lacks the sustenance of the lower frequencies and the detail of the upper frequencies; however, the song can still be made out as if it was being played by a very cheap and small speaker. Lastly, the high band version sounds extremely unpleasant as it mostly contains the quality details to supplement the other parts. As a result, it sounds much like high-frequency screeching to the beat of the song. Thus, we can expect that this equalizer will be able to accurately separate the lower, middle, and upper bands of a music signal and adjust the gain as input by the user.

# 5. Conclusion

Digital filtering is an extremely versatile and important tool to a world where most data is stored digitally. It can be used in all sorts of applications such as obtaining desired information from a group of data or simply improving the sound quality of music without the need of building specific circuits. This project exemplified this by covering the basics of digital filtering and showing its applications in smoothing stock market figures and customizing the way music can sound.

## 5.1 Difficulties

Though we were not required to understand the underlying mathematics/theory of each filter, without this knowledge, it was a bit difficult to understand why certain types of filters behaved in certain ways. However, our experiments still gave us a good idea of how some filters behave and how to filter signals.

## 5.2 Final Thoughts and Discussion

This lab taught us about basic filters and applications for those filters in digital filtering; however there is still much to expand on. For example in section 4’s three band equalizer, the filters can be given more coefficients to become closer to ideal. It would also be much better to implement more than three bands for much more detailed equalization; currently the controllable aspects don’t allow for a high degree of sound customization compared to what an eight-band equalizer could provide.

# Appendix A

In a zip folder submitted with this report, please find the .m-files and .wav-files for the following sections and functions:

* Part 1 Digital Filtering: “Assignment1.m”
* Part 2 Filter Characterization: “Assignment2.m”
* Stock Price Information: “microsoft\_stock.txt”
* Music file: “music.wav”
* Part 3 Equalizer Testing: “Assignment3.m”
* Part 3 Equalier Function: “equalize.m”