Date received:

SCHOOL OF ENGINEERING

COVER SHEET FOR CONTINUOUSLY ASSESSED WORK

Course Code ……EE3053……

SECTION 1: **Student to complete**

SURNAME/FAMILY NAME: ……OSTERMAN……..

FIRST NAME: ……BRIAN JACK……..

ID Number: ……51337831…….

Date submitted: ……13/11/2015…….

Please:

- *Read the statement on "Cheating" and definition of "Plagiarism" contained over page. The full Code of Practice on Student Discipline, Appendix 5.15 of the Academic Quality Handbook is at: www.abdn.ac.uk/registry/quality/appendices.shtml#section5*

- *attach this Cover Sheet, completed and signed to the work being submitted*

SECTION 2: Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read, understood and will abide by the University statement on cheating and plagiarism defined over the page and that this submitted work is my own and where the work of others is used it is clearly identified and referenced. I understand that the School of Engineering reserves the right to use this submitted work in the detection of plagiarism.

Signed: _____Brian Jack Osterman_____

Date:_____12/11/2015_____

Note: Work submitted for continuous assessment will not be marked without a completed Cover Sheet. Such work will be deemed 'late' until a completed cover Sheet is submitted and will be subject to the published penalty for late submission.

Cheating in any assessment, whether formative or summative, can result in disciplinary action being taken under the University's Code of Practice on Student Discipline. For these purposes "Cheating" includes:

(a) Possession in an examination of material or electronic device which has not been authorised in writing by the relevant Course Co-ordinator. Students whose first language is not English may, however, refer to a dictionary where this is approved by the Head of the School responsible for the examination;

(b) Copying from another student in an examination;

(c) Removing an examination book from an examination room;

(d) Impersonating another candidate in relation to any assessment;

(e) Permitting another person to impersonate oneself in relation to any assessment;

(f) Paying or otherwise rewarding another person for writing or preparing work to be Submitted for assessment;

(g) Colluding with another person in the preparation or submission of work which is to be assessed. This does not apply to collaborative work authorised by the relevant course coordinator.

(h) Plagiarism. Plagiarism is the use, without adequate acknowledgment, of the intellectual work of another person in work submitted for assessment. A student cannot be found to have committed plagiarism where it can be shown that the student has taken all reasonable care to avoid representing the work of others as his or her own.

Filter Design

1 PREFACE

The task in this assignment was to design and simulate a Digital Music System through MATLAB. The system designed needs to be tested by a sound bite of a popular song provided (*Another Brick in the Wall (Part 2) –Pink Floyd*). Provided with two bites from the same section of the song, one a clean bite and the other containing an irritating, high-pitch, single-tone noise due to the recording of it, the assignment called for the cleaning of the noisy signal to better match the clean signal. In short the task was to remove the noise from the noisy signal.
The next step required by the assignment is to find the frequency specifications for a music system's subwoofer, woofer, and tweeter. Using those specifications the audio needed to be split into three components suitable to be played by a particular speaker. [1]

1.1 UNITS

Unit	**Representation**
Hertz	Hz
Decibels	dB
Watts	W
Ohms	Ω

1.2 BACKGROUND CONCEPTS AND THEORIES

1.2.1 Infinite Impulse Response (IIR) Filters

There are two generalized types of filters that can be made. These are the Finite and the Infinite Impulse Response (FIR and IIR respectively) filters. IIR filters do not have their impulse response become exactly zero at any point in time, instead they continue indefinitely. On the other hand, FIR filters have an impulse response *h(t)* that becomes exactly zero at $t > T$ for some finite T. The nature of this is by definition, finite. This project will focus solely on the use of IIR filters for two main reasons. In comparison to FIR filters, the IIR is more compact and as such can reach a specified frequency response faster than an FIR filter. They can also become unstable, unlike FIR filters which are always stable.

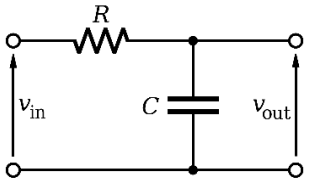

Figure 1: Example of a First Order RC circuit [10]

IIR filters also have several other advantages. They tend to be mathematically simpler, have less time delay and share similarities with analog filters. They also tend to be more admitting

of noises, however whether or not this is a good thing depends solely on the needs of the filter. [5]
It is also important to note that all filter designs are Magnitude only. One the performance of the filter is specified over magnitudes, there is no control over phase. As such it can be desired to reduce and/or compensate the effect of phase change, shift, or delay.

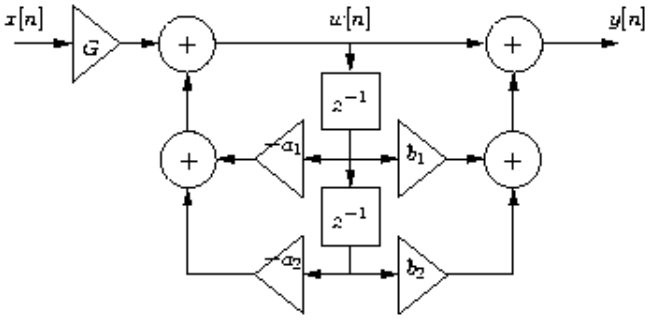

Figure 2: Example of a second order IIR filter implementation [9]

1.2.2 IIR Filter Design

Starting with an analog filter, the design of an IIR filter is pretty straightforward.
Impulse response of an analog filter is $h_a(t)$ and its transfer function is represented by $H_a(s)$.
Next either convert the impulse response from infinite time, $h_a(t)$, to discrete time $h[n]$, or transform the transfer function $H_a(s)$ in the s-plane to its equivalent in the z-transform ($H(z)$).

$$H(z) = H_a(s)|_{s=m(z)}$$

Where $s = m(z)$ is a mapping function from the s-plane to the z-plane. It must have the following properties:

- Mapping from $j\omega$ to the unit circle $|z|= 1$, needs to be one to one and onto, that is each point must correspond with no repeated values in pairs.
- The left-half plane maps inside the unit circle. This helps keep stability.
- $m(z)$ should be a rational function of z, so a rational $H_a(s)$ to rational $H(z)$.

This can be done by finding the poles and zeros of the analog filter's transfer function. The poles of a transfer function are the values of s which cause the denominator to equal zero, thus making the function undefined. The zeros of a function are the values of s for which the numerator is equal to zero, making the function zero.
For example, if the analog filter has transfer function $H_a(s) = \sum_{k=1}^{K} \frac{R_k}{s-p_k}$, then it can be transforms through partial fractions into its equivalent z-transform.

$$H(z) = \sum_{k=1}^{K} \frac{R_k z}{z - e^{p_k T}}$$

1.2.3 Nyquist Frequency

In order to properly display a periodic waveform it is necessary that all of its Fourier series components be present in the sampled signal. To do this the sampling rate r of at *least* twice the *highest* waveform frequency must be used. The Nyquist frequency (or limit) is the highest frequency that can be used at a given sample rate in order to fully reconstruct the signal. [4]

$$f_{Nyquist} = \frac{1}{2} r$$

1.2.4 High and Low Pass Filters

A high-pass filter is one that allows signals above a cutoff frequency, otherwise known as the pass-band, while attenuating signals below the cutoff frequency, which can also be referred to as the stop-band. The output of high-pass filters is directly proportional to the rate of change

of the input signal. Conversely, a low-pass filter is one that attenuates the signals above the cutoff frequency, while allowing signals below that frequency.

1.2.5 Band-Pass and Band-Stop Filters

A band-pass filter is one that allows signals between two cutoff frequencies, one low and one high, while attenuating the signals below and above respective cutoff frequencies. This can be achieved by applying a low pass filter at the high cutoff frequency, and applying a high pass filter at the low cutoff frequency. A band-stop filter is the opposite of a band-pass filter. It allows frequencies below the low cutoff and above the high cutoff while reducing the signals between the two cutoff frequencies. It can be created by applying a low-pass filter at the low cutoff frequency and a high-pass filter at the high cutoff frequency. A narrow band-stop filter can also be referred to as a "notch" filter.

1.2.6 Low Order vs High Order Filters

Filter order is what determines the effectiveness of filters. Low order filters have longer slope for their frequency response curve, while higher order filters have a short slope. In short, the higher the order of the filter, the steeper the slope and closer to the "brick-wall" effect it gets. However, as the filter order increases, so too does its complexity. Greater complexity requires more components and as such higher order filters become increasingly expensive. It is up to the filter designer to select a filter order based on both the specifications required, and the resources provided.

1.2.7 Filter Design Methods

Filters are used to clean up signals, remove noise, decimate signals and discover patterns within and between different signals. When designing filters there are several methods to choose from. These include Butterworth, Chebyshev Type-I, Chebyshev Type-II, and elliptic. The following are shown in the first figure.

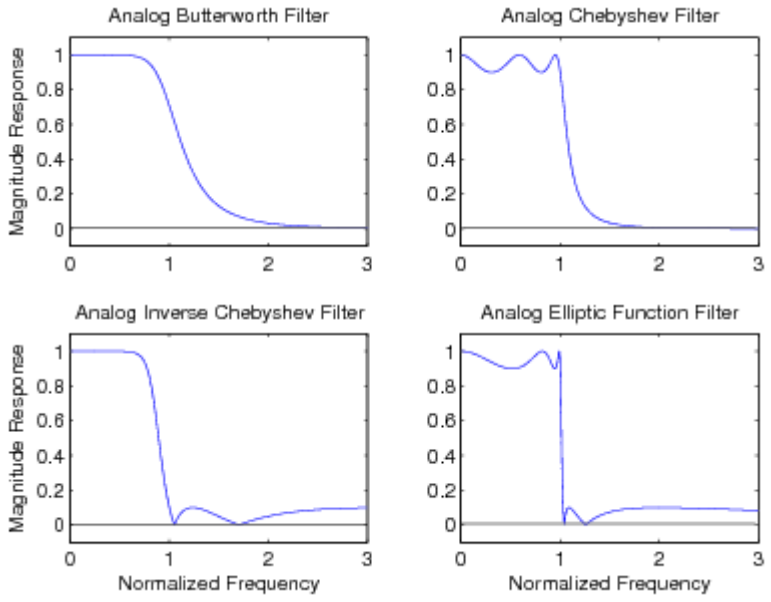

Figure 3: Frequency Responses of the Four Classical Low-pass IIR Filter Approximations [3]

1.2.7.1 *Butterworth Filters*

Butterworth filters are designed to be as smooth as possible. They are the simplest of the four classical IIR filters. The frequency response of a Butterworth filter shows that it is flat in the pass-band (no ripples) and approaches zero in the stop band.

1.2.7.2 *Chebyshev Filters*

Chebyshev filters have steeper roll-off than butterworth filters. They also have ripples in the pass-band (Chebyshev Type-I or just Chebyshev) or ripples in the stop-band (Chebyshev Type-II or Inverse Chebyshev).

1.2.7.3 *Elliptical Filters*

Elliptic filters have ripples in both the pass-band and the stop-band. The size of the ripple in each of the bands can be adjusted independently. It has the fastest transitional gain between the stop-band and the pass-band of any filter of the same order. As the ripple in the pass-band approaches zero, the Elliptic filter becomes a Chebyshev Type-I, and as the ripple in the stop-band approaches zero, the Elliptic filter becomes a Chebyshev Type-II. If both the pass-band ripple and the stop-band ripple approach zero, then the filter becomes a Butterworth filter.

2 ASSIGNMENT

2.1 FILTERING THE SIGNAL

2.1.1 Finding the Noise

The first step in finding the location of the noise in the signal would be to first import the data itself into a program such as OCTAVE or MATLAB. Both of these are interpretive programs for the calculation and evaluation of numerical data. For this project MATLAB is preferable to OCTAVE because of its nature being geared more towards overall mathematical computation and analysis. It is a commonly used program across many disciplines of engineering for its wide variety of functions.

To import data from a file into MATLAB, simply use the *load* function on the desired file. The file provided to use was named 'AssignmentData15.mat':

```
load 'AssignmentData15.mat'; %loads the signal data into MATLAB
workspace
```

This provides three values. "Fs" the frequency at which the sound bites are sampled. "xc", a "clean" sound-bite to compare with the noisy and filtered signals, and "xn", a "noisy" sound-bite containing the irritating frequency to be filtered out as mentioned by the assignment.

Next it can be useful to view the original signal waveforms of both the "clean" and the "noisy" signals in matlab. This can help to ensure that each sound-bite actually occur across the same section of the song. To do this it is necessary to find the sampling rate T_s of the signal and use it to create a time vector. We know that the sampling rate is inversely proportional to the sampling frequency of a signal.

$$T_s = \frac{1}{F_s}$$

To plot the original waveforms of the original signal:

```
t = 0:1/Fs:49452/Fs; % creates a time vector from the provided
sampling frequency
figure(1)
subplot(2,1,1);
plot(t,xc);
title('Signal wave form from Lab3Data.mat');
ylabel('Clean Signal');
xlabel('time');
```

```
subplot(2,1,2);
plot(t,xn);
ylabel('Noisy Signal');
xlabel('time');
```

This provides us with the following figure:

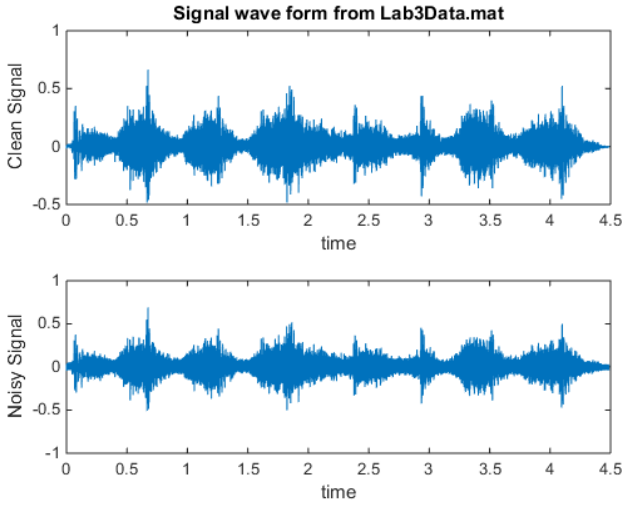

Figure 4

From this it can be seen that the two signals do indeed occur at the same point in the song. This can also be check by using the *sound* function in MATLAB (*sound(xc,Fs) for clean, sound(xn,Fs) for noisy).*

To locate the range in which the noise occurs in the signal, the amplitude spectrum of the sound bites should be mapped. It is preferable to use the single-sided amplitude spectrum, as the waves are duplicated in the negative range.

```
%% FFT of clean signal %%
Lc = length(xc);
NFFTc = 2^nextpow2(Lc);
Xfftc = fft(xc,NFFTc)/Lc;
mag_Xfftc = abs(Xfftc(1:NFFTc/2+1));
fc = Fs/2*linspace(0,1,NFFTc/2+1);
% FFT of noisy signal %
Ln = length(xn);
NFFTn = 2^nextpow2(Ln);
Xfftn = fft(xn,NFFTn)/Ln;
mag_Xfftn = abs(Xfftn(1:NFFTn/2+1));
fn = Fs/2*linspace(0,1,NFFTn/2+1);
```

FFT stands for the Fast Fourier Transform. It is an algorithm used to compute the Discrete Fourier Transform (DFT) of a sequence or signal. Plotting the magnitude of the transform to the frequency vector for its respective transform ("fc" for the clean, "fn" for the noisy).

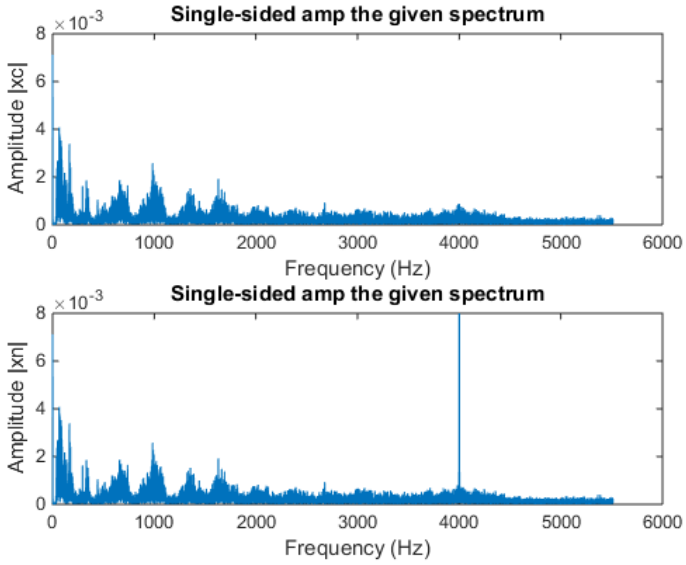

Figure 5

Now the noise from the signal "xn" is clearly visible around the 4000[Hz] mark. Closer inspection on the graphs will show that the frequency still exists in the "clean" signal, and as such it cannot simply be removed from the noisy signal otherwise the integrity of the sound bit will be ruined. In addition, while examining the noise in "xn", it should be noted that the noise actually ranges from just below 4000 Hz and a little over 4000 Hz. The filter designed should accommodate this range as it is not one sole frequency value.

2.1.2 Selecting Filter Type

When filtering out the noise, there are several options to take. We can use any one of the four classical filters (Butterworth, Chebyshev Type-I, Chebyshev Type-II, and Elliptical). As explained in the preface each type has a different effect on the filter. The design of filters necessitates mention of complexity and cost. The more complex a filter is, the more it will cost to develop. However as filters become more complex, they become more accurate and produce better and sharper results. As such it is important to decide whether you want a smooth frequency response or a rippling frequency response.

The best way to remove this signal would be to use a "notch" filter, also known as a narrow band-stop filter. Notch filters can be designed by using any of the four classical filter approximations.

From examining the amplitude spectrum of the noisy signal, as well as that of the clean signal provided you see that there is quite a bit of ripple near the point in which noise is displayed. There are several options for implementing a band-stop filter. Each of the different options will provide different results in the final filter. Although the differences between different classes of filter (Butterworth, Chebyshev, Elliptical) will exist in the resulting signal waveform, the audible differences may be insignificant and as such choice mostly comes down to user preference as well as design limitations.

In this assignment, an Elliptical filter will be used to clean the noisy sound bite provided. An Elliptical filter was chosen from the fact that it includes ripple in both the pass-band and the stop-band. This is important because it will create variation in the power of each frequency it dampens and passes. Since instrumental audio is sporadic by nature, this will help keep the signal sounding more natural in its recording as opposed to other filter types which just attenuate signals. The nature of the filter type allows users to tailor the effect of the ripple

exhibited in both the pass-band and the stop-band and as such gives good control on the overall effect of the filter.

2.1.3 Filtering the Noise

Designing any type of filter can require some complex mathematics. Luckily MATLAB can do it all, and even has the calculations for the classical filters stored as functions. For the Elliptical filter, MATLAB has the algorithm stored as

```
[b,a] = ellip(n,Rp,Rs,Wp,ftype)
```

Where *n* is the filter order, *Rp* is the peak-to-peak pass-band ripple, *Rs* is the peak-to-peak stop-band ripple, *Wp* is the pass-band edge frequency, and *ftype* is the type of filter (low-pass, high-pass, band-pass, band-stop). [6]

Examining the noisy signal closely reveals that the noise starts around 3999 Hz and ends around 4001 Hz at first glance. For the initial design, these are the cutoff frequencies that will be chosen to implement the filter. Once the filter is applied, go back and slowly increase or decrease the values in order to get the filter as accurate as possible. The same is said for the values for *n, Rp, and Rs*.

Wp is the pass-band edge frequency, which is where the band-stop filter starts and stops. This will be entered as a vector from the lowest value to the highest value. Since filter functions in MATLAB use normalized frequency, the values chosen for pass-band edge frequency need to be converted.

$$f_{Normalized} = \frac{2f}{f_s}$$

Where f is the specified frequency, and f_s is the sampling frequency.

When entering filter order, it should be noted that for a band-stop filter, the number chosen for *n* is actually half of the order number. For this filter, a 6th order elliptical filter input minimizes its effect on frequencies outside its ranges while still providing the desired effect. Higher order filters be more expensive to use, but their accuracy is worth the extra cost in this case. After toying with values the best options (or near best) for filter accuracy for this particular case are found and the filter implementation is performed.

```
cut1 = 3999/(Fs/2);
cut2 = 4000.28/(Fs/2);
[b,a] = ellip(3,2,23.1,[cut1 cut2],'stop');
freqz(b,a) %frequency response
y = filtfilt(b,a,xn);
```

From the function *freqz* the frequency response of the filter designed is shown.

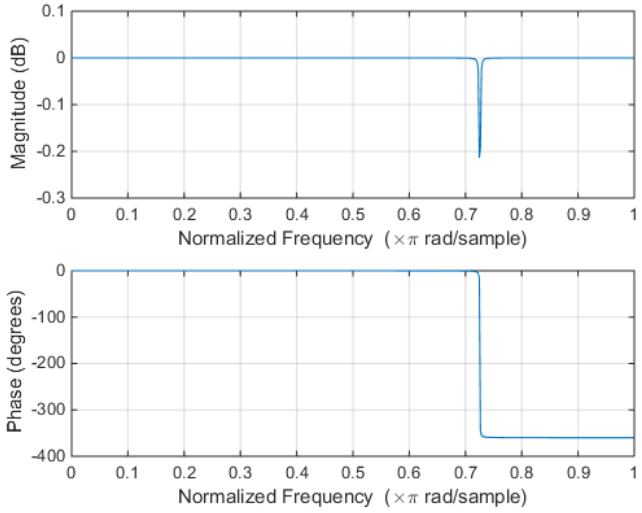

Figure 6: Frequency Response of the Elliptical band-stop filter

Since IIR filters introduce phase delay to the signal, the MATLAB function *filtfilt* is used instead of the basic filtering function *filter*. The *filtfilt* function attempts to compensate for the

phase delay by applying the input data in both directions. This is done by filtering the data forward, then reversing the filtering sequence and running it back through the filter again. According to MATLAB documentation the results of this is zero-phase distortion, a transfer function which equals the squared magnitude of the original filter transfer function, and double the filter order of the original filter specified by *b* and *a*. [7]
Understand that this application of the 6^{th} order filter returns a 12^{th} order filter, as the 6^{th} order filter is applied twice. The higher the order of filters, the more accurate they become. However that accuracy comes with a cost. It is possible to get the signal relatively accurate on lower filter orders if need be, however since there is no current restriction on cost for this assignment it does not hurt to make the filter as accurate as possible. Even if that means an expensive high-order filter.
Now to compare the filtered signal with the original and noisy. Since the Fourier transforms of the clean and noisy signals are already done, just perform the same calculations on the new filtered signal.

```
Lstop = length(y);
NFFTstop = 2^nextpow2(Lstop);
Xfftstop = fft(y,NFFTstop)/Lstop;
mag_Xfftstop = abs(Xfftstop(1:NFFTstop/2+1));
ystop = Fs/2*linspace(0,1,NFFTstop/2+1);
```

Using this the amplitude spectrum of the three signals can now be compared.

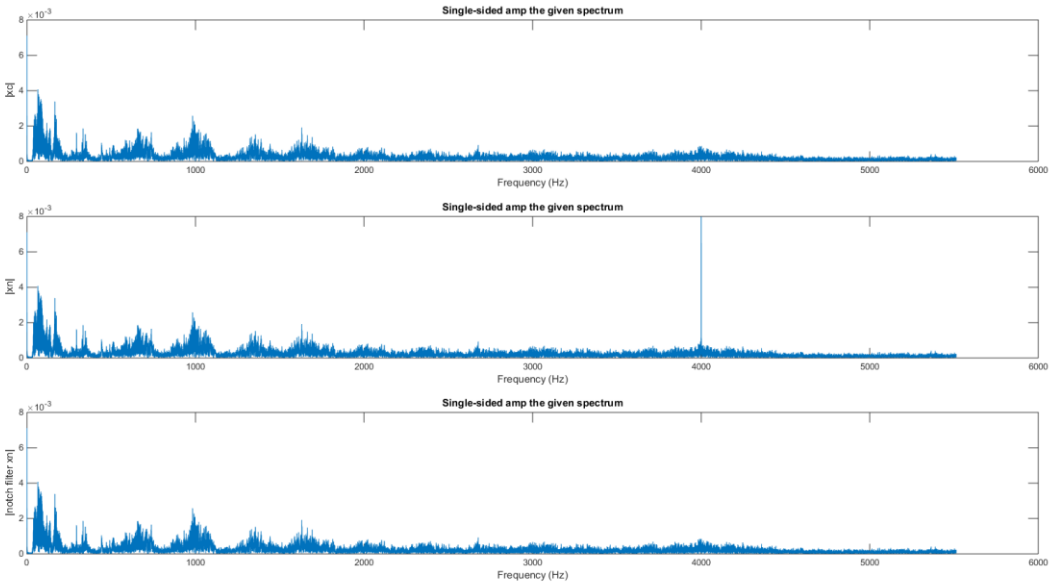

Figure 7: Amplitude Spectrum Comparison between the three signals

As shown by this the noise has certainly been filtered out and the filtered signal appears to be almost identical to the clean signal. The accuracy of the filtered signal can be better shown by plotting the clean signal over the filtered one, and seeing where the outlying conditions are.

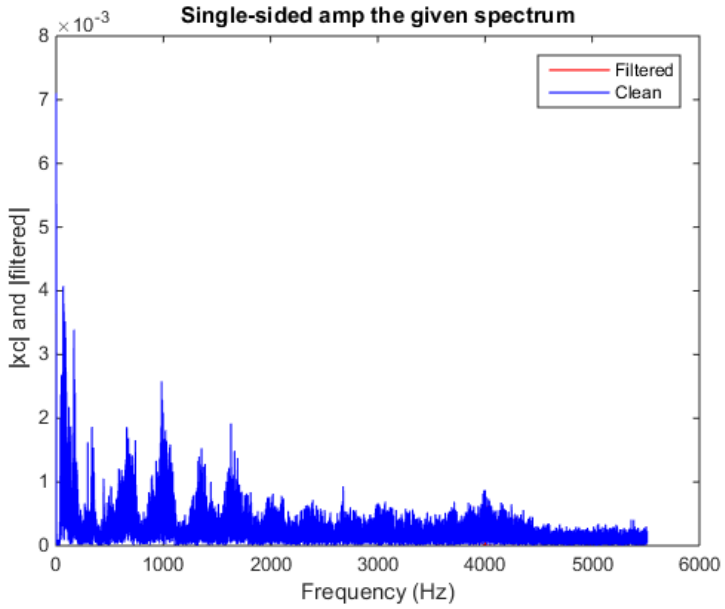

Figure 8: Comparison of Filtered and Clean signals

Taking a closer look at the outlying point in the comparison:

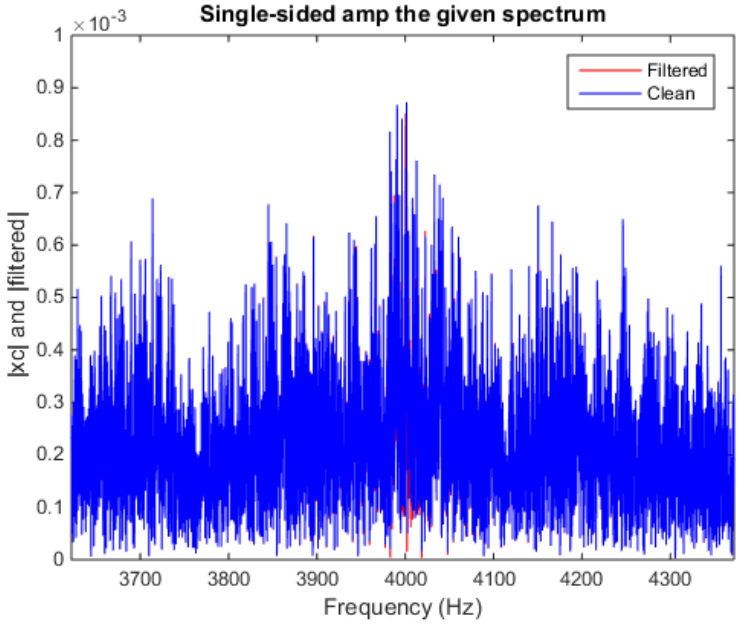

Figure 9: Filter comparison overlap

Where the filtered signal is given in red, and the clean filter is given in blue. Figure 7 shows that the only point where there is a difference between the two signals is near where the filter was applied. The difference between the two signals is so miniscule that it will not make an audible difference in the sound. Again to check the cleanliness of this signal the *sound* function can also be used and should show that they sound exactly the same within the accuracy of human hearing.

2.2 Sound System Simulation

Now that the noise from the sound bite has been cleaned, the next task is to simulate a sound system. Find a sound system to your liking and look up its user manual. The user manual should contain a variety of design specifications for the owner to read and understand. The set of speakers chosen for this are the Elite SP-EFS73 Dolby Atmos® enabled Elite® Concentric Floorstanding Speakers, designed by Andrew Jones. These speakers run quite

expensive, sitting at a suggested price of $699.99. However from the marketing and renown of the particular brand one would imagine that their sound quality is very good. Their expense also matches with the filter used, as one would imagine a 12th order filter can get complex and expensive.
To do the simulation, the signal needs to first be separated into its three ranges; low, mid, and high. This is done using the system specs for crossover frequency in addition to the systems minimum and maximum frequencies. Then the three separate ranges will be recombined to provide a new signal that displays what the sound system outputs as a whole when all three ranges are played at the same time.

2.2.1 Sound System Specifications

The notable specifications for this project for the SP-EFS73 speaker given by the specification sheet are as follows:

- Frequency Range…… 38 Hz – 20 kHz
- Nominal Impedance…… 4 Ohms
- Sensitivity (2.83 V)…… 86 dB
- Maximum Input Power……. 140 W
- Crossover Frequency…….. 260 Hz, 2.6 kHz [2]

The two specifications that matter most for the basic simulation of this sound system is the frequency range, and crossover frequency. The crossover frequency is the point at which sound being played changes between the speakers. 260 Hz is the point in which sound output switches from the subwoofer to the woofer, and 2.6 kHz is the point at which sound output switches from the woofer to the tweeter.

2.2.2 Separating the Signal

As with the selection of the filter for the cleaning of the noisy signal, filter classes and types must also be chosen for the separation of signals to simulate the sound system. The three components being subwoofer, woofer and tweeter. The subwoofer plays the low frequency range: from 38 Hz to 260 Hz. The woofer plays the middle frequency range: from 260 Hz to 2.6 kHz. The tweeter plays the high frequency range: from 2.6 kHz to 20 kHz.
Since the nature of the crossover frequencies is such that no frequency should be audible above or below that point (dependent on the range for the filter to pass), it makes sense to use a filter class with no ripple in the stop band. This is to maximize the attenuation of the stop-band. Recall that the Chebyshev Type-I filter has pass-band ripple, which as explained during the initial filtering stage is helpful in maintaining variance in the audio, while it has a steep cutoff and a maximally flat stop-band. That meaning there is no ripple in the stop-band. The filter class used below will be Chebyshev. The Chebyshev filters have a built in MATLAB function to design the filter.

```
[b,a] = cheby1(n,Rp,Wp,ftype)
```

Where *cheby1* is the function for Chebyshev Type-I, *n* is the filter order, *Rp* is the pass-band ripple, *Wp* is the pass-band edge frequency and *ftype* is the type of filter desired.
Also note how each speaker in the system has a defined range. This means that it has a lower end and a higher end. As such a band-pass filter, one that allows audio within its range, should be used.
Firstly the frequency range and crossover frequency values will be converted from Hertz to normalized frequency.

$$f_{Normalized} = \frac{2f}{f_s}$$

This gives us the following values:

- Low End normalized frequency = 0.0069 (π x rad/sample)
- Crossover 1 normalized frequency = 0.0472 (π x rad/sample)

- Crossover 2 normalized frequency = 0.4717 (π x rad/sample)
- High End normalized frequency = 3.6281 (π x rad/sample)

2.2.2.1 Subwoofer

From the specification sheet the subwoofer needs to be set to play frequencies in the 38 Hz to 260 Hz (low end to crossover 1) range. This is done using the Chebyshev Type-I band-pass filter. For this filter the same order number, and pass-band ripple strength will be used, just to stay consistent.
Applying the values to the *cheby1* function, and implementing the filter:

```
[d,c] = cheby1(3,2,[lowend cross1],'bandpass');
freqz(d,c);

sub = filter(d,c,y);
```

Note that the *freqz* function displays the frequency response of the filter.

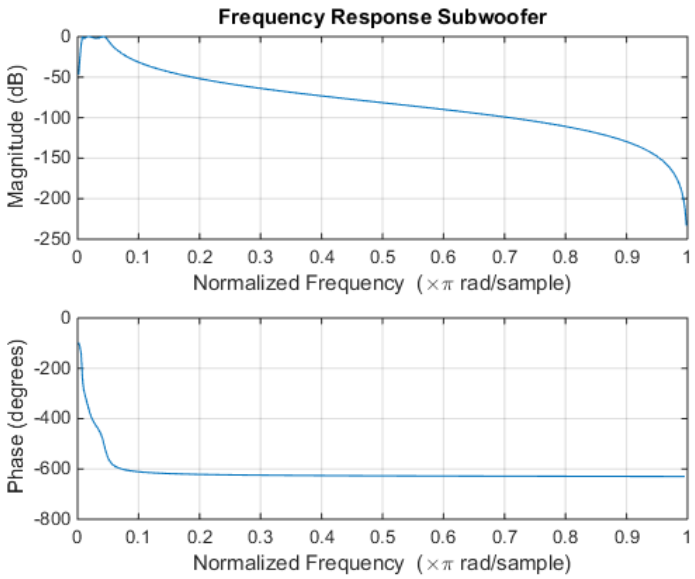

Figure 10: Frequency response of the Subwoofer filter

It can be seen in the frequency response that the filter is working as expected. Meaning that at before the low end frequency of 38 Hz or 0.0069 (π x rad/sample), all the frequencies are attenuated. During the passband, i.e. between the low end frequency and the first crossover frequency, the filter is allowing frequencies through with a ripple of 2 dB. After the first crossover frequency the frequencies are attenuated throughout the rest of the signal.
Again *filtfilt* is used because of its phase delay correction. Otherwise the delay/change in phase that is evidenced in the frequency response would impact the output of the signal. In fact *filtfilt* will be used for each of the filters implemented in this section. The reduction and/or elimination of phase shift and delay is important in keeping the signal as close to original as possible.

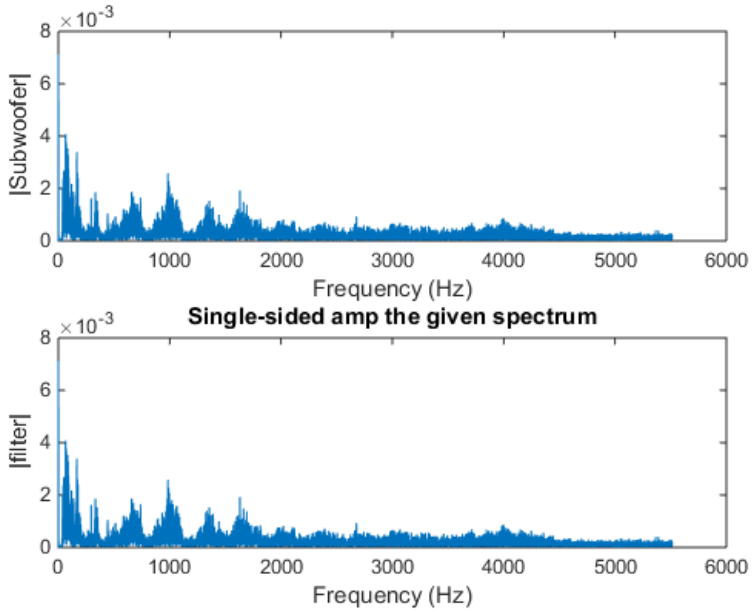

Figure 11: Single Sided Amplitude spectrum comparison between Subwoofer and Filtered Signals

As shown there is no noticeable difference between the two graphs other than the label on the y-axis. Instead, simply plot the filtered component to the time vector made earlier and the difference between the two can be seen very clearly.

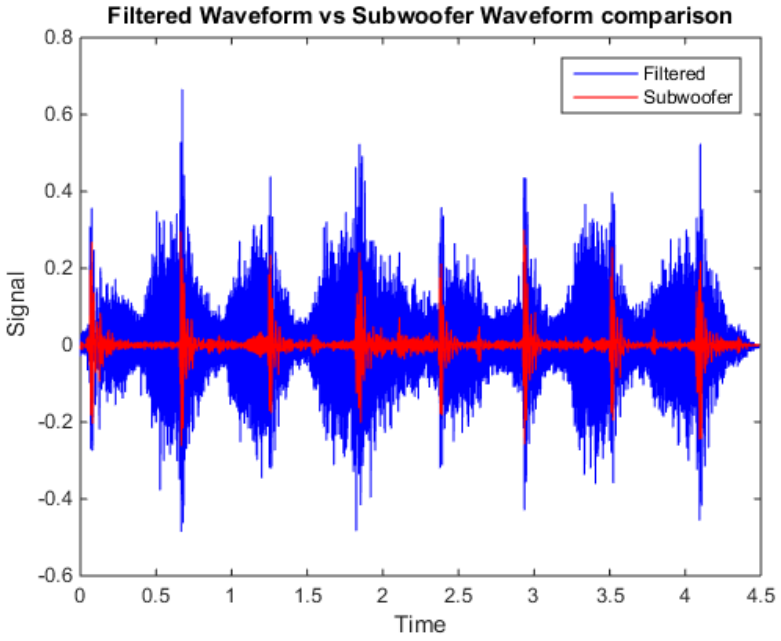

Figure 12: Filtered vs Subwoofer Signal Comparison

As is clearly evidenced by the comparison of the waveforms, with the filtered signal in blue and the Subwoofer's signal in red, the filter properly reduces all signals not within the specified range of the subwoofer.

2.2.2.2 Woofer

This filter is done exactly the same as the subwoofer. It is a Chebyshev Type-I band-pass filter, and will use the same settings as the one implemented for the subwoofer. The only difference between the two will be the pass-band frequencies, which will be the two crossover frequencies.

```
[f,e] = cheby1(3,2,[cross1 cross2],'bandpass');
woof = filtfilt(f,e,y);
freqz(f,e)
```

The frequency response shows details on how the filter works when implemented.

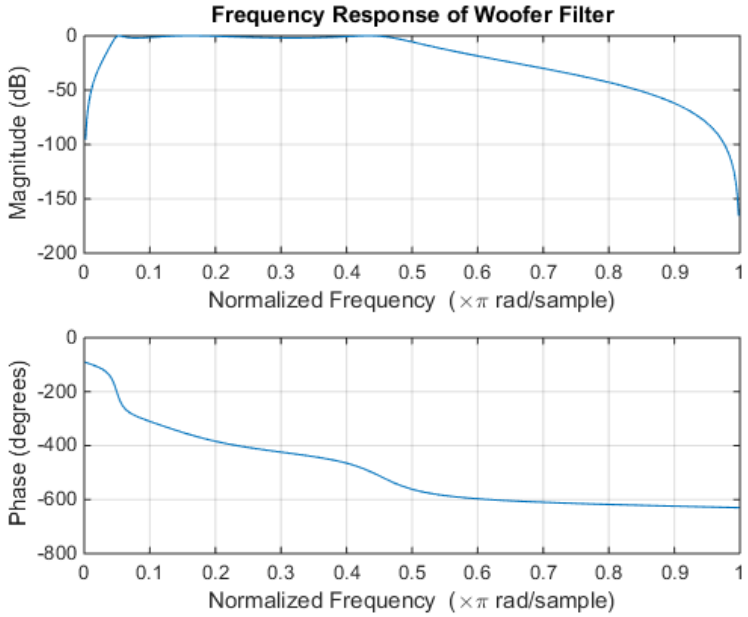

Figure 13: Frequency Response for Woofer filter

As shown in the magnitude of the frequency response, the filter allows frequencies within the selected pass-band (between the crossover frequencies), and attenuates frequencies outside of the pass-band. Additionally the phase graph shows that there is a shift by the filter being applied on the signal, however, this shift is neutralized from the use of *filtfilt* in applying the filter to the signal.
The effect of the filter can be easily seen on the graph of the waveform for the signal.

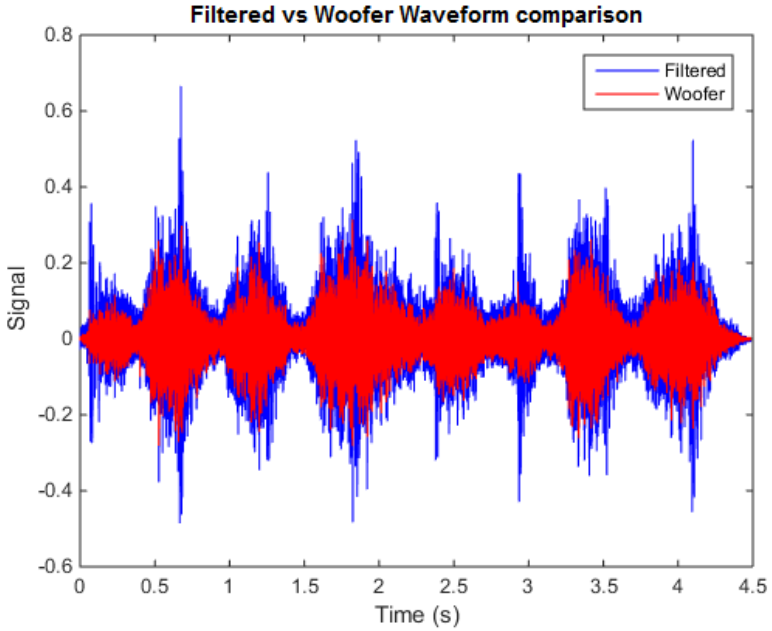

Figure 14: Comparison between Filtered signal waveform and Woofer signal Waveform

This comparison shows the effect of the filter for the Woofer on the overall signal much easier than the frequency response does. It shows that while the woofer is able to loudly play most of the frequencies in the signal, the very high, and very low frequencies are out of its range and therefore attenuated.

2.2.2.3 Tweeter

The tweeter plays the high frequency range for the sound system. These are the frequencies ranging from 2.6 kHz all the way up to 20 kHz. It is commonly stated that the range of

human hearing is from 20 Hz up to 20 kHz. [8] This makes our speakers play the upper limit of human hearing. This can be taken into account in the design of the filter as humans will not, save for a select few with extraordinary hearing under strict testing conditions, hear above this upper limit. As such it makes the filter design easier as a high-pass filter can be implemented instead of a band-pass. This wasn't done for the design of the subwoofer filter as humans can hear as low as 20 Hz, and the speaker's range only drops as low as 38 Hz.
A Chebyshev Type-I filter is still used however, and sticking with the consistency of the other two filters it will remain a 6^{th} order filter and keep a pass-band ripple of 2 dB.

```
[h,g] = cheby1(6,2,cross2,'high');
tweet = filtfilt(h,g,y);
freqz(h,g)
```

As before the details of the filter are shown by its frequency response. That is its Magnitude and Phase changes.

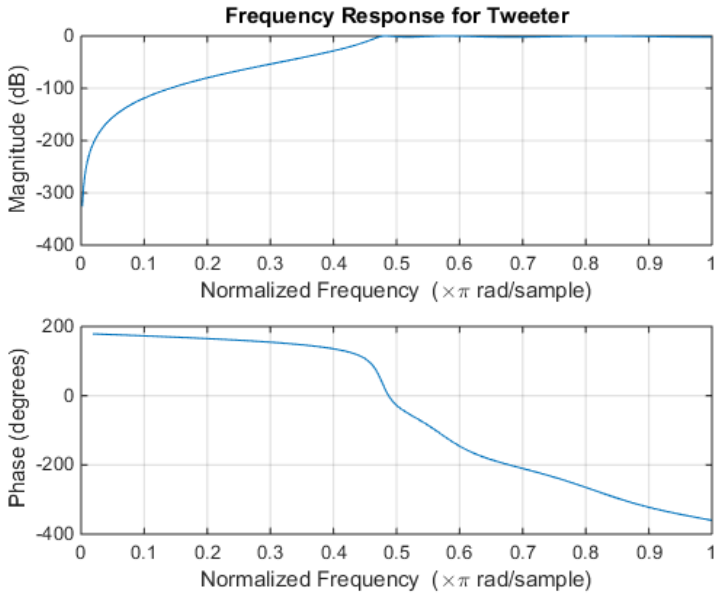

Figure 15: Frequency Response for Tweeter Filter

The frequency response shows what is expected from the magnitude changes of the filter designed, and is one of the best indicators that it was implemented correctly. Attenuating everything below the second cutoff frequency, while passing all above it with a slight ripple. As expected from Chebyshev Type-I filters, it has ripple in the pass-band and attenuation in the stop-band. As with before, although the frequency response shows that there is phase shift/change/delay, by using the *filtfilt* function the distortion is counteracted.
Plotting the signal waveform over time can better show the implementation of the filter in comparison with the original signal.

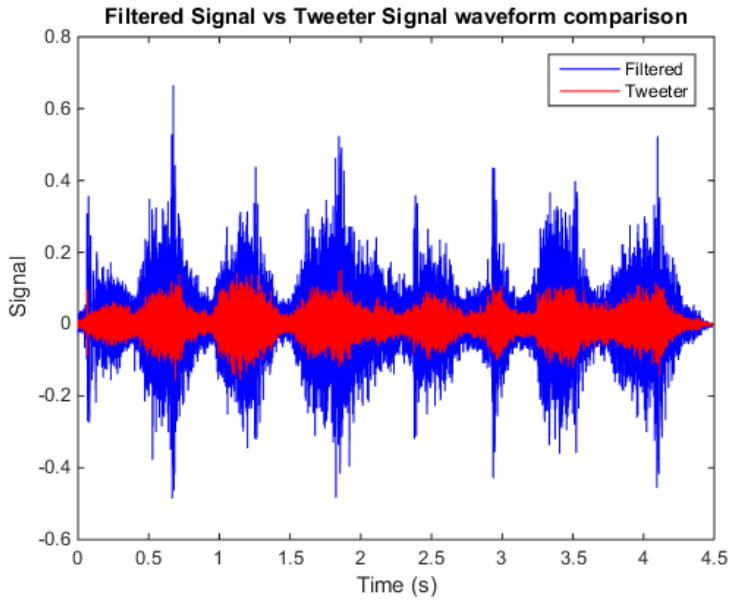

Figure 16: Waveform comparison between filtered signal and tweeter signal

This waveform comparison shows how the filter affects the sound of the original signal over time. It plays only a portion of the frequencies. By listening to the signal using the *sound* function in matlab, only the high frequencies should be heard.

2.2.3 Recombining the Separated Signals

To finish the simulation of the speaker system the three separated and filtered signals need to be recombined. Using a full comparison waveform, the effect of combining the separate parts can be compared with the originally filtered signal before separation.

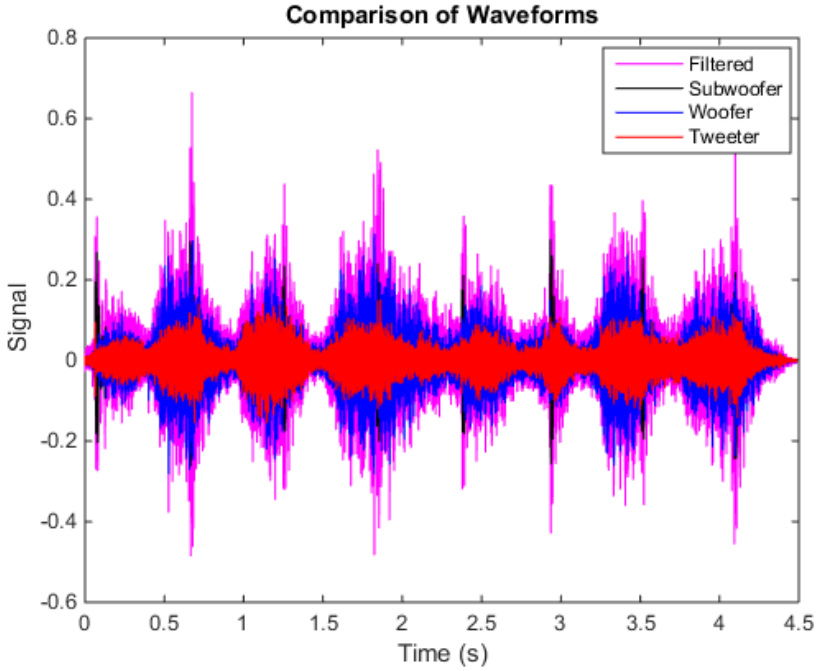

Figure 17: Full waveform comparisons

It is shown here that simply combining the waveforms will produce a dampened version of the filtered signal. This dampening is a downside to filtering, but the cleanliness provided by filtering noise and the crisp sound produced by the dedicated speakers is worth the dampening as it can be compensated for by simply turning up the volume. However in other applications of filtering it may be preferable to reduce the effect of damping as much as possible.

Recombining the separated signals in this simulation is as simple as it sounds, and is accomplished by simply finding the sum of the three separated signals.

```
system = sub+woof+tweet;
```

Now that the signals are combined and the speaker system simulation is complete, view the new waveform comparison for the sake of gauging how much dampening has been experienced.

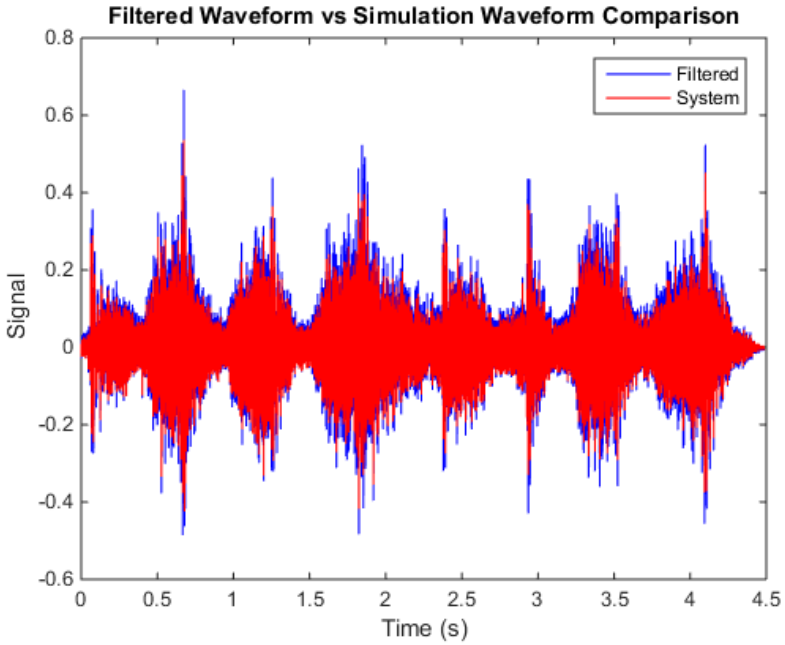

Figure 18: Waveform comparison between the Filtered Waveform, and the Sound System Waveform

It is clear here that there definitely exists some dampening. Whether or not this amount of dampening is acceptable depends largely on personal opinion. Now obviously since the clean signal is provided it would be foolish to not compare this system against it, and that will provide a much better and more accurate depiction of the dampening of the signal.
To do this the waveform over time can be used:

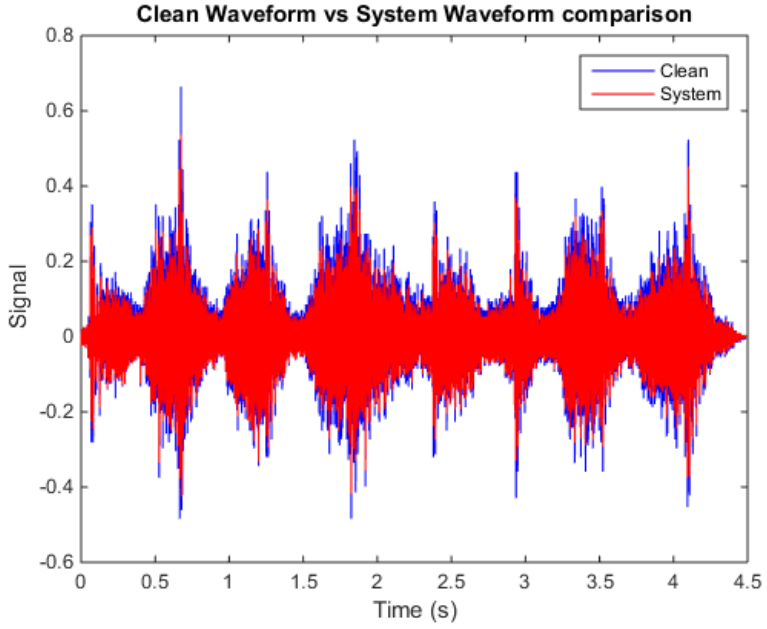

Figure 19: Waveform comparison between the Clean Signal and the simulated System Signal

Since the filtered signal was so accurate by comparing the waveform for the simulated system was near redundant if not for sheer purpose of observing any potential difference between the clean and the filtered signals. Any difference is not easily discernable.

For a more accurate comparison the single-sided amplitude spectrum will be able to provide breakdown on the frequencies and their strength in comparison to each other.

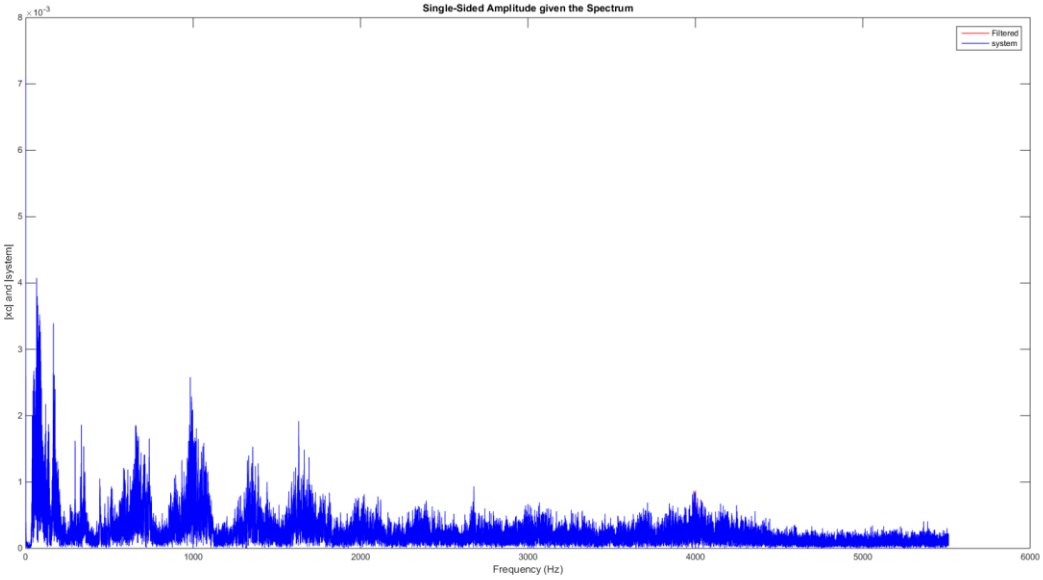

Figure 20:Single Sided Amplitude Spectrum Comparison Between Clean Signal and Simulated Speaker System Signal

In the single-sided amplitude spectrum comparison it is clearly shown that there is no significant dampening exhibited by the simulated system. The only point in which there appears to be noticeable damping is where the elliptical stop-band filter was placed.

3 CONCLUSIONS

3.1 ALTERNATIVE METHODS

Alternatively to what has been done, a simpler way for the user to conduct the filtering shown above through MATLAB would be to just use Butterworth filters. While these filters do not provide the ripple shown, and thus it may compromise the integrity of the original signal, not much is required detail-wise to construct the programming for these. This is because since the filter has no ripple in either the pass-band or the stop-band, it does not require specifications for the power of the ripples as both the Chebyshevs and the Elliptical filters call for these design requirements.
Implementations of a Butterworth type notch filter can be done using the following MATLAB Code:

```
[b,a] = butter(order,[cutoff1 cutoff2],'stop');
freqz(b,a)
y = filter(b,a,xn);
```

Where “cutoff1” is the lower cutoff frequency, “cutoff2” is the higher cutoff frequency, and “order” is the filter order.
Or one can simply use the MATLAB function for a second order infinite impulse response notch filter:

```
wo = cutoff/(Fs/2);  bw = wo/Q;

[b,a] = iirnotch(wo,bw);

fvtool(b,a);
```

Where “cutoff” is the cutoff frequency, Fs is the sampling frequency, and “bw” is the bandwidth.
The notch filter can also be designed using Chebyshev Type-I and Chebyshev Type-2 filters. This is done within matlab by using the *cheby1* and *cheby2* functions:

```
[b,a] = cheby1(order,Rp,Wp,'stop');
freqz(b,a)
y = filter(b,a,xn)
```

Where “order” is the filter order, “Rp” is the peak-to-peak pass-band ripple, and “Wp” is the pass-band edge frequency. For a Chebyshev Type-II filter just substitute “cheby1” with “cheby2”.

3.1.1 Developing a Custom Filter apart from built in MATLAB filter functions.

Alternatively to using the prescribed MATLAB functions for various filters (*ellip, butter, cheby1, etc.*), MAtLAB also has a function called fdesign, which allows users to custom tailor filters to their needs beyond just the few options available in the built in functions. Using fdesign everything about the filter can be controlled. The distance between stop-bands and pass-bands, the intensity of ripples and attenuation in both stop and pass-bands. In the case of a band type filter (band-pass/band-stop), it allows the user to specify attenuation in both ends of the stop or pass bands.
These functions can be useful is the needs of design for simulation are very specific, however for most cases using the prescribed MATLAB functions are good enough. [11]

3.2 CONCLUSION

As evidenced there can be a great deal to ponder in the development of filters, regardless of their application. The main advantage of using a program such as MATLAB is that it has the capability to compute all the complex mathematics required to develop specific filters. All it requires is a few input variables from the user that pertain to the location and effectiveness of the filter and the program is able to handle the rest from there.
There is no better way to gain a greater understanding and appreciation for filters and the effort put into developing sound systems like having a hands on experience in developing them. Although this was simply a computerized simulation it forces the user to try and think about the pros and cons of various filter types and pick one that best applies to the needs required by the problem. Hands on projects also require at least a decent amount of background research, in which further solidifies the users knowledge on the subject itself. They help to reinforce what is learned through the practical application which necessitates a combination of both abstract and concrete reasoning. That is, reasoning through surface level literal observation, and reasoning through facts, and specific examples.
While the different filter classes all have their advantages and disadvantages, it is up to the user to determine which fits best with what is required to solve the problem. Since any of the filter classes, whether it be butterworth, chebyshev, or elliptical, can be used to develop filters for clean noise or separating signals it is important for the user to think about how the different ones effect the filtered section along with the overall signal.

3.3 FULL MATLAB CODE

```
load 'AssignmentData15.mat'; %loads the signal data into matlab workspace
t = 0:1/Fs:49452/Fs; % creates a time vector from the provided sampling
frequency

figure(1)
subplot(2,1,1);
plot(t,xc);
title('Signal wave form from Lab3Data.mat');
```

```
ylabel('Clean Signal');
xlabel('time');
subplot(2,1,2);
plot(t,xn);
ylabel('Noisy Signal');
xlabel('time');
%% FFT of clean signal %%
Lc = length(xc);
NFFTc = 2^nextpow2(Lc);
Xfftc = fft(xc,NFFTc)/Lc;
mag_Xfftc = abs(Xfftc(1:NFFTc/2+1));
fc = Fs/2*linspace(0,1,NFFTc/2+1);
% FFT of noisy signal %
Ln = length(xn);
NFFTn = 2^nextpow2(Ln);
Xfftn = fft(xn,NFFTn)/Ln;
mag_Xfftn = abs(Xfftn(1:NFFTn/2+1));
fn = Fs/2*linspace(0,1,NFFTn/2+1);

% Single-sided amp plot
figure(2)
subplot(2,1,1);
plot(fc,mag_Xfftc);
title('Single-sided amp the given spectrum');
ylabel('Amplitude |xc|');
xlabel('Frequency (Hz)');
subplot(2,1,2);
plot(fn,mag_Xfftn);
axis([0 6000 0 8e-3])
title('Single-sided amp the given spectrum');
ylabel('Amplitude |xn|');
xlabel('Frequency (Hz)');
%% Filter Noisy Signal %%
cut1 = 3999/(Fs/2); %lower end frequency of stop band
cut2 = 4000.28/(Fs/2); %higher end frequency of stop band

[b,a] = ellip(3,2,23.1,[cut1 cut2],'stop');
freqz(b,a) %frequency response of filter

y = filtfilt(b,a,xn);
%FFT of filter:
Lstop = length(y);
NFFTstop = 2^nextpow2(Lstop);
Xfftstop = fft(y,NFFTstop)/Lstop;
mag_Xfftstop = abs(Xfftstop(1:NFFTstop/2+1));
ystop = Fs/2*linspace(0,1,NFFTstop/2+1);

%Single-sided amp plot
figure(3)
subplot(3,1,1);
plot(fc,mag_Xfftc);
title('Single-sided amp the given spectrum');
ylabel('|xc|');
xlabel('Frequency (Hz)');
subplot(3,1,2);
plot(fn,mag_Xfftn);
axis([0 6000 0 8e-3])
title('Single-sided amp the given spectrum');
ylabel('|xn|');
xlabel('Frequency (Hz)');
subplot(3,1,3);
plot(ystop,mag_Xfftstop);
axis([0 6000 0 8e-3])
```

```
title('Single-sided amp the given spectrum');
ylabel('|filtered xn|');
xlabel('Frequency (Hz)');
%% Singled sided amplitude spectrum comparing filtered and clean signals
figure(4)
plot(ystop,mag_Xfftstop,'r');
axis([0 6000 0 8e-3])
title('Single-sided amp the given spectrum');
hold on
plot(fc,mag_Xfftc,'b');
title('Single-sided amp the given spectrum');
ylabel('|xc| and |filtered|');
xlabel('Frequency (Hz)');
%% Sound System Specs %%
lowend = 38/(Fs/2); %Hz
cross1 = 260/(Fs/2); %Hz
cross2 = 2600/(Fs/2); %Hz
highend = 20000/(Fs/2); %Hz
%% Subwoofer %%
[d,c] = cheby1(3,2,[lowend cross1],'bandpass');
freqz(d,c); %frequency response of cheby1 bandpass for subwoofer

sub = filtfilt(d,c,y);
%calculating single-sided fft of subwoofer
Lsub = length(sub);
NFFTsub = 2^nextpow2(Lsub);
Xfftsub = fft(y,NFFTsub)/Lsub;
mag_Xfftsub = abs(Xfftsub(1:NFFTsub/2+1));
ysub = Fs/2*linspace(0,1,NFFTsub/2+1);

figure(5) %single sided amp spectrum of subwoofer to filtered
subplot(2,1,1);
plot(ysub,mag_Xfftsub);
ylabel('|Subwoofer|');
xlabel('Frequency (Hz)');
subplot(2,1,2);
plot(ystop,mag_Xfftstop);
title('Single-sided amp the given spectrum');
ylabel('|filter|');
xlabel('Frequency (Hz)');

figure(6) %waveform comparison subwoofer and filtered
plot(t,y,'b');
hold on
plot(t,sub,'r');
xlabel('Time');
ylabel('Signal')
%% Woofer %%
[f,e] = cheby1(3,2,[cross1 cross2],'bandpass');
woof = filtfilt(f,e,y);
freqz(f,e) %frequency response of cheby1 bandpass

figure(7)
plot(t,y,'b'); %waveform comparison woofer and filtered
xlabel('Time (s)');
ylabel('Signal');
hold on
plot(t,woof,'r');
%% Tweeter %%
[h,g] = cheby1(6,2,cross2,'high');
tweet = filtfilt(h,g,y);
freqz(h,g) %frequency response of cheby1 filter
```

```
figure(8) %waveform comparison tweet and filtered
plot(t,y,'b');
xlabel('Time (s)');
ylabel('Signal');
hold on
plot(t,tweet,'r');
%% Comparing the three Speakers %%
figure(9)
plot(t,y,'m');
title('Comparison of Waveforms');
xlabel('Time (s)');
ylabel('Signal');
hold on
plot(t,sub,'k');
hold on
plot(t,woof,'b');
hold on
plot(t,tweet,'r');
%Find the sum%
system = sub+woof+tweet;

figure(10)
plot(t,y,'b')
title('Filtered Waveform vs Simulation Waveform Comparison');
xlabel('Time (s)');
ylabel('Signal');
hold on
plot(t,system,'r');
%% Compare with clean %%
figure(11)
plot(t,xc,'b');
title('Clean Waveform vs System Waveform comparison');
xlabel('Time (s)');
ylabel('Signal');
hold on
plot(t,system,'r');
%% SS Amp spectrum b/w clean and system %%
Lsystem = length(system);
NFFTsystem = 2^nextpow2(Lsystem);
Xfftsystem = fft(y,NFFTsystem)/Lsystem;
mag_Xfftsystem = abs(Xfftsystem(1:NFFTsystem/2+1));
ysystem = Fs/2*linspace(0,1,NFFTsystem/2+1);

figure(12)
plot(fc,mag_Xfftc,'r');
title('Single-Sided Amplitude given the Spectrum');
ylabel('|xc| and |system|');
xlabel('Frequency (Hz)');
hold on
plot(ysystem,mag_Xfftsystem,'b');
```

3.4 References

- [1] Aphale, Dr. S (2015). *Term Assignment*. Signals, Systems and Signal Processing EE3053 Course Handout. Aberdeen, UK: University of Aberdeen.
- [2] Jones, Andrew. *SP-EFS73 Dolby Atmos Enabled Elite Concentric Floorstanding Speakers Designed By Andrew Jones*. 1st ed. Pioneer & Onkyo USA Corporation, 2015. Web. 9 Nov. 2015.
- [3] Burrus, C. Sidney. *Digital Signal Processing and Digital Filter Design (Draft)*. OpenStax CNX. Nov 18, 2012 http://cnx.org/contents/31edd076-92d1-4406-b71e-1db8e4560237@6.2.
- [4] Mathworld.wolfram.com, (2015). *Nyquist Frequency – from Wolfram MathWorld*. [online] Available at: http://mathworld.wolfram.com/NyquistFrequency.html [Accessed 10 Nov. 2015].
- [5] Hasan, Md. Mehedi, Md. Rafiqul Islam, and Mursalin Sayeed. 'Design And Implementation Of Butterworth, Chebyshev-I And Elliptic Filter For Speech Signal Analysis'. International Journal of Computer Applications 98.7 (2015): 12-18. Web. 3 Nov. 2015.
- [6] Uk.mathworks.com,. *'Elliptic Filter Design - MATLAB Ellip'*. N.p., 2015. Web. 6 Nov. 2015.
- [7] Uk.mathworks.com,. *'Zero-phase Digital Filtering - MATLAB filtfilt'*. N.p., 2015. Web. 9 Nov. 2015.
- [8] Rosen, Stuart, and Peter Howell. *Signals And Systems For Speech And Hearing*. 2nd ed. London: Academic Press, 2011. Print.
- [9] Courses.engr.illinois.edu,. *'Lab 3: IIR Filters Overview — ECE420 |Today| Documentation'*. N.p., 2015. Web. 12 Nov. 2015.
- [10] Wikipedia,. 'First Order RC Circuit.Svg'. N.p., 2015. Web. 12 Nov. 2015.
    - Note: Used solely for image.
- [11] Uk.mathworks.com,. *'Design A Filter Using Fdesign - MATLAB & Simulink'*. N.p., 2015. Web. 12 Nov. 2015.