CPX 2 Report

ECE 434: Digital Signal Processing

November 5, 2024

C1C Ben Cometto

Contents

List of Figures

List of Tables

Abstract

Computer Exercise 2 (CPX 2) required processing 5 signals in a variety of manners to produce certain objectives. Digital signal processing using various FIR and IIR filters enabled these objectives to be met with low order filters. The signals, including test tones, ECG data, and jammed audio, provided a wide range of applications for digital signal processing. Overall, CPX2 was a great exercise in solving real-world problems using digital filter techniques.

1 Introduction

Computer Exercise 2 (CPX 2) for ECE 434, Digital Signal Processing, involved processing 5 signals. The five signals are:

- Signal x_1: Two tones, one louder and one softer
- Signal x_8: A message containing the secrets of sunshine extraction from cucumbers, but important information is jammed
- Signal x_3: An electrocardiogram (ECG) that is contaminated with a power line artifact
- Signal x_4: An unknown audio signal that is jammed
- Signal x_7: A series of 10 test tones that must be modified

The processing goals for each signal are:

- Signal x_1: Make the louder tone softer and the softer tone louder
- Signal x_8: Reveal the jammed information
- Signal x_3: Remove the power line artifact
- Signal x_4: Reveal the unknown audio
- Signal x_7: Correctly modify the 10 test tones

I filtered each signal to enact the signal's processing goal. All aspects of the filter, including using an infinite impulse response (IIR) versus a finite impulse response (FIR) design method, order, and other parameters, have been carefully considered, and are reported in each signal's respective section. For access to the Matlab `.mlx` files, source signal data, filter designer session `.fda` file, filter taps `.bin` files, and output `.wav` sound files, see the project's GitHub repository at `https://github.com/dbcometto/ece434_cpx2`.

2 Signal x_1

Signal x_1 contains two tones. The louder tone (Tone 1) is at approximately 1539 Hz , and the quieter (Tone 2) at 2000. The tones can be seen in the signal's frequency spectrum in Figure 1. The unfiltered audio can be heard with the `x1_unfiltered.wav` file in the GitHub repository.

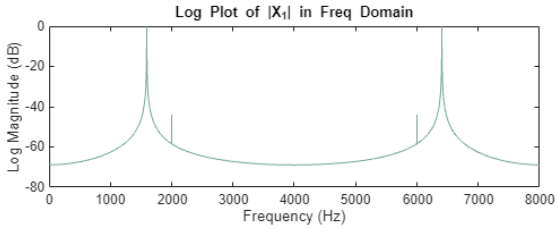

Figure 1: Signal x_1 Frequency Spectrum

In Figure 2 and Figure 3, the signal can be seen in the time domain. The signal looks as expected for two sinusoids being present, although the effect of Tone 2 is minimal.

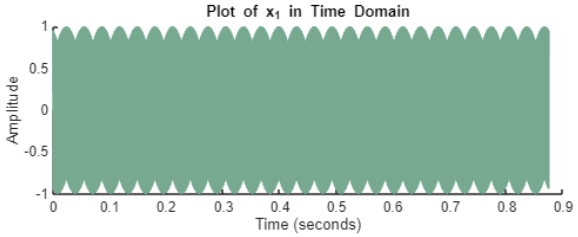

Figure 2: Signal x_1 by Time

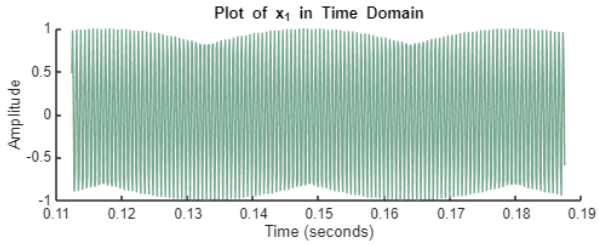

Figure 3: Signal x_1 by Time

The processing goal for Signal x_1 was to attenuate the tone of higher magnitude (Tone 1) so that the magnitude of Tone 1 was more than 30 dB less than the magnitude of the tone of lower magnitude (Tone 2).

A highpass filter was chosen for the first method, as that was the simplest option. Because preserving linear phase is not important, IIR was chosen. Additionally, the elliptic method was chosen as it yields the best magnitude response for a given order. After several iterations, Filter 1 with the parameters in Table 1 was designed.

Order	Fstop	Fpass	Astop	Apass
7	1600 Hz	2000 Hz	73 dB	1 dB

Table 1: Parameters for Filter 1 for Signal x_1

Applying this filter to the signal yielded the response in Figure 4. The horizontal dark green line marks $-30\,\text{dB}$, with the peak of Tone 2 at $0\,\text{dB}$. Because the magnitude of Tone 1 is more than $30\,\text{dB}$ less than the magnitude of Tone 2, Filter 1 meets the specifications at order 7. The signal's response can be heard with the `x1_filtered7.wav` file in the GitHub repository.

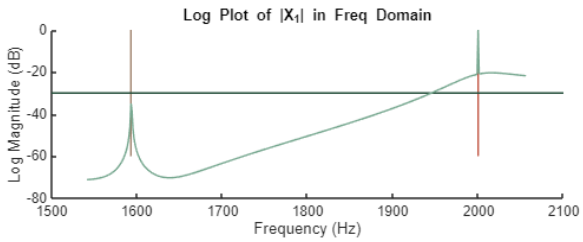

Figure 4: Response of Signal x_1 to Filter 1, Order 7

In order to reduce the order, a notching filter's poles and zeros were modified. An order 2 filter, with a zero on Tone 1's frequency, was unable to reduce the magnitude enough. Thus, Filter 2, an order 4 filter with the pole zero plot in Figure 5 was developed.

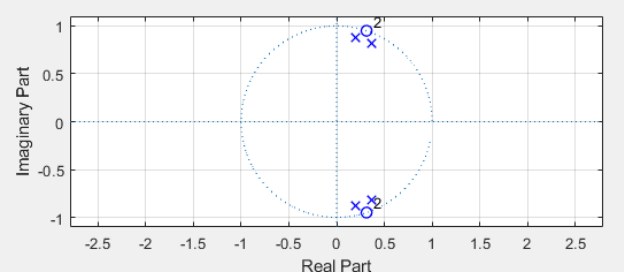

Figure 5: Pole Zero Plot of Filter 2 for Signal x_1

The locations of the poles and zeros are recorded in Table 2.

Order		Zeros	Pole	Pole
4	Angle	$\pm 1.2519\,\text{rad}$	$\pm 1.1519\,\text{rad}$	$\pm 1.3519\,\text{rad}$
	Magnitude	1	0.9	0.9

Table 2: Parameters for Filter 2 for Signal x_1

The signal response of Filter 2 can be seen in Figure 6. The signal response can be heard in the `x1_filtered4.wav` file in the GitHub repository.

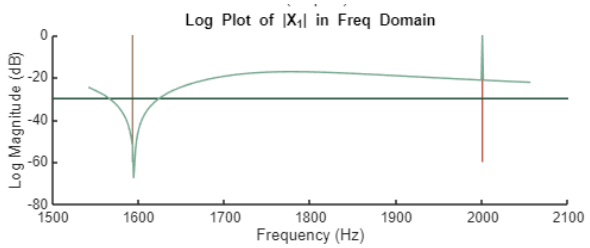

Figure 6: Response of Signal x_1 to Filter 2, Order 4

Because of the sharpness of the notching filter, the response looks different than expected for two tones. Additionally, the sound is not identical to the first response. Despite these factors, the majority of Tone 1's magnitude is below the $-30\,\text{dB}$ line, and thus Filter 2 meets the specifications with order 4.

The response of an order 2 notching filter can be seen in Figure 7. The filter is not able to bring Tone 1's magnitude below the $-30\,\text{dB}$ specification.

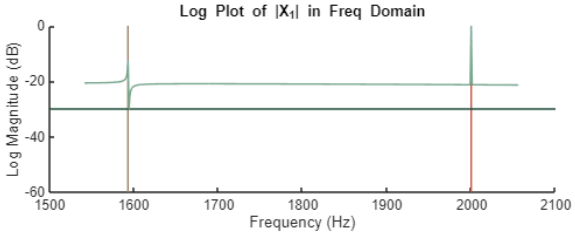

Figure 7: Response of Signal x_1 to an Order 2 Notching Filter

3 Signal x_8

Signal x_8 contains an audio signal. For most of the signal's duration, as can be seen in Figure 8, the audio is undisturbed. However, at about the 8 second mark, narrowband jamming with 3 tones (plus a "tone" at DC) begins. The tones can be clearly seen in the signal's frequency spectrum in Figure 9. The unfiltered audio can be found in the `x8_unfiltered.wav` file in the GitHub repository.

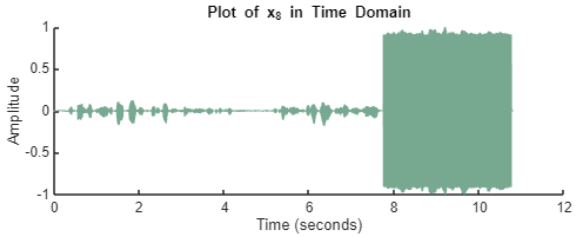

Figure 8: Signal x_8 over Time

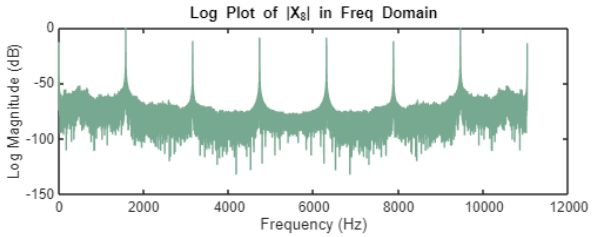

Figure 9: Frequency Spectrum of Signal x_8

Focusing on the portion of the frequency spectrum including the tones, it can be seen that the tones are at approximately 1573 Hz , 3151 Hz , and at 4726 Hz (and at DC).

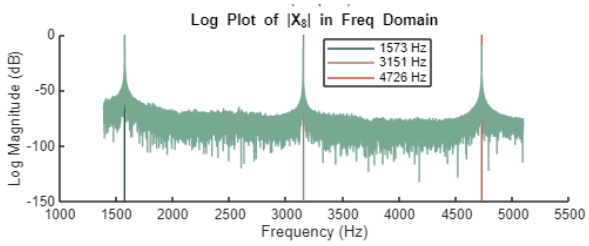

Figure 10: Frequency Spectrum of Signal x_8 near Interfering Tones

Up to the 8 second mark, we are able to hear the message, "He has been eight years upon a project for extracting sunbeams out of cucumbers... He told me, he did not doubt, that, in eight years more—." The rest of the message (perhaps the secrets for extracting sunbeams from cucumbers?) has been jammed. In order to recover the message, we must remove the interfering tones.

Again, a notching filter is indicated, as specific frequencies must be stopped. This notching filter was built by placing poles and zeros. Because we do not need to preserve linear phase, we can use poles (and thus create an IIR filter). Converting the tone frequencies to normalized angular frequencies gave the locations of the zeros. The poles were offset slightly in both angle and magnitude to maximize the notching effect while

minimizing the effect on the surrounding frequencies. For Filter 1, six zeros were used, one for each tone. The pole zero plot can be seen in Figure 11.

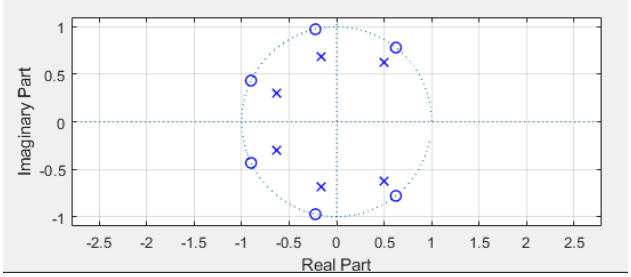

Figure 11: Pole Zero Plot of Filter 1 for Signal x_8

The locations of the poles and zeros are compiled in Table 3.

Order		Zero/Pole	Zero/Pole	Zero/Pole
6	Angle	± 0.8969 rad	± 1.7962 rad	± 2.6931 rad
	Magnitude	1/0.8	1/0.8	1/0.8

Table 3: Parameters for Filter 1 for Signal x_8

Applying this filter to the signal results in the removal of the majority of the interference. However, the tones are still present, albeit at a diminished volume. The signal response can be seen in Figure 12. The audio information is clearly able to be heard, as can be verified in the `x8_filter7.wav` file in the GitHub repository.

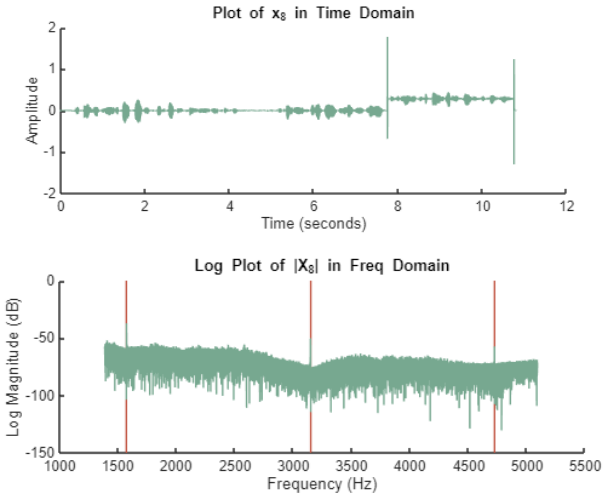

Figure 12: Response of Signal x_8 to Filter 1, Order 6

To fully remove the tones, the filter can be modified to order 12, placing two zeros at each tone's frequency. This change results in Filter 2. The pole zero plot for Filter 2 can be seen in Figure 13.

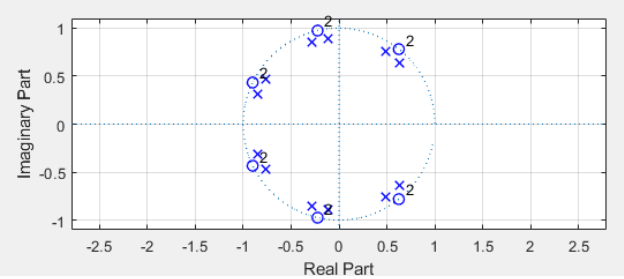

Figure 13: Pole Zero Plot of Filter 2 for Signal x_8

The locations of the zeros are the same as in Filter 1. The poles' magnitude were increased to 0.9, and they are placed ± 0.1 rad from the angle of their corresponding zero.

The signal response to Filter 2 can be seen in Figure 14. Now, the tones have been completely removed, but there is a DC shift and two pops can be heard during the jammed portion of the signal. The output audio can be heard in the `x8_filter12.wav` file in the GitHub repository. It is not necessary to remove these artifacts, as they do not interfere with the content of the signal.

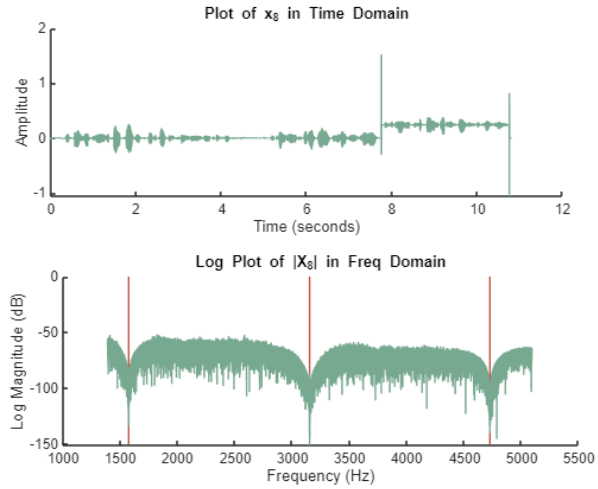

Figure 14: Response of Signal x_8 to Filter 2, order 12

However, in order to maximally enhance the audio and remove these final artifacts, zeros and corresponding poles are added near DC, resulting in Filter 3. The response to Filter 3 can be seen in Figure 15

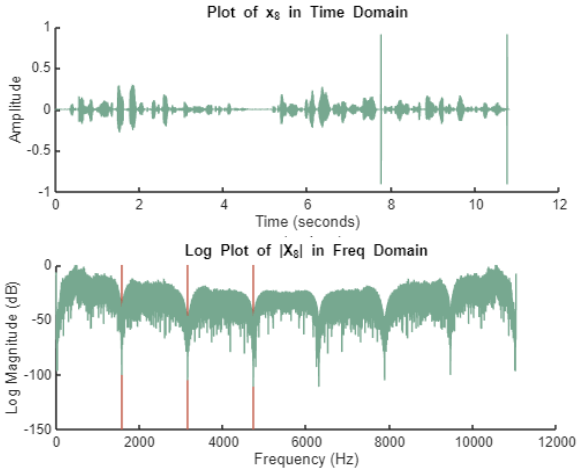

Figure 15: Response of Signal x_8 to Filter 3

Additionally, the pops are manually removed from the audio by setting the corresponding portions of the time signal to zero. A final, completely clean output can be seen in Figure 16, and heard in the `x8_filterNoPop.wav` file in the GitHub repository.

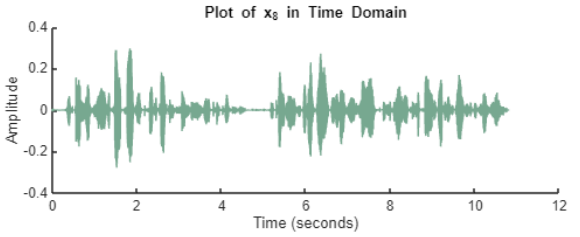

Figure 16: Signal x_8 after Filter 3 and Editing

As can be heard following the application of any of the 3 filters, the complete message is, "He has been eight years upon a project for extracting sunbeams out of cucumbers... He told me, he did not doubt, that, in eight years more, he should be able to supply the governor's gardens with sunshine." Originally, I misheard several of the words. However, a Google search revealed that it is an abridged quote from *Gulliver's Travels*, which enabled me to correct the words that I heard incorrectly.

4 Signal x_3

Signal x_3 contains data from an electrocardiogram (ECG) that was contaminated with an artifact from a power line. Important to note, though, is that the ECG was recorded in Europe, and that European power lines oscillate at 50 Hz , rather than the American 60 Hz . The contaminated data can be seen in Figure 17.

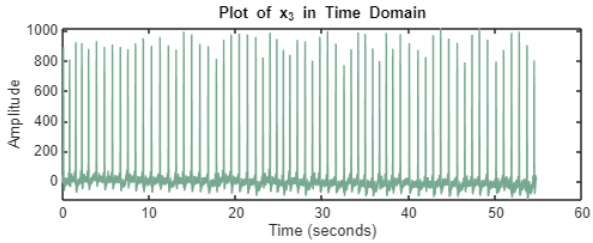

Figure 17: Signal x_3 over Time

Focusing in on the first 15 seconds, as is done in Figure 18, yields a clearer picture of the contamination.

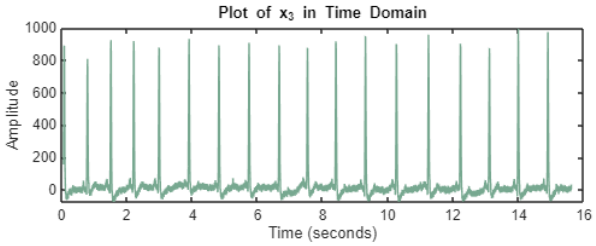

Figure 18: Signal x_3 over the First 15 Seconds

Additionally, looking at the signal's frequency spectrum in Figure 19 reveals the large magnitude spike around 50 Hz . This is the contamination that must be removed.

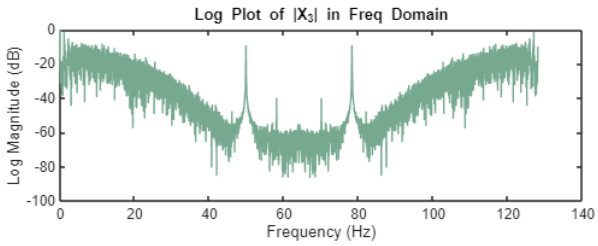

Figure 19: Frequency Spectrum of Signal x_3

Before designing a filter to remove the contamination, it will be helpful make some measurements. As can be seen in the markings in Figure 20, we see that the peak of the interference is at -10 dB , while the average if the data is around -55 db near 50 Hz . This requires a filter to decrease the spike's magnitude by about -45 dB . Additionally, we can note that the peak is at around 49.8 Hz , and impacts the range of about 48.6 Hz to 51 Hz .

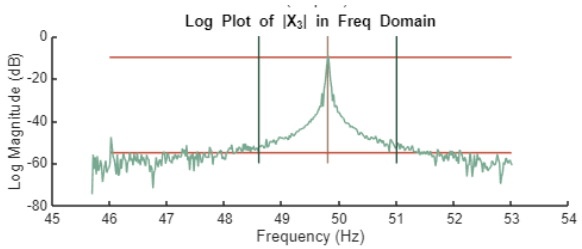

Figure 20: Frequency Spectrum of Signal x_3 focused on Interference

In this signal, preserving linear phase is important to not disrupt important information. This requires the use of an FIR filter. Additionally, we will be using a bandstop filter to reduce the magnitude of the interference. Iterating from the starting point given by the measurements conducted earlier, Filter 1 was developed. The frequency response of Filter 1 can be seen in Figure 21.

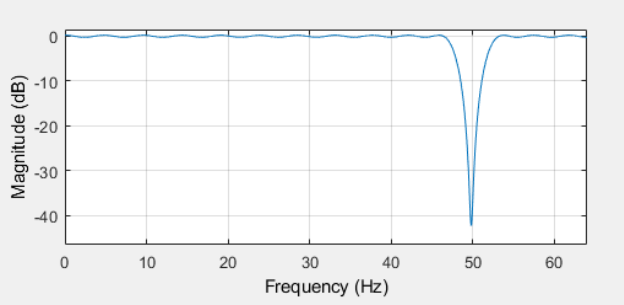

Figure 21: Frequency Response of Filter 1 for Signal x_3

The parameters defining Filter 1 are compiled in Table 4. Typically, the equiripple method will produce the best magnitude response, but in this case, the generalized equiripple method produced a lower order. The passband ripple was set low so that the information in the data was preserved.

Order	Fpass1	Fstop1	Fstop2	Fpass2	Apass1	Astop	Apass2
56	46.6 Hz	49.72 Hz	49.88 Hz	53 Hz	0.5 dB	40 dB	0.5 dB

Table 4: Parameters for Filter 1 for Signal x_1

Applying this filter yields the response in Figure 22.

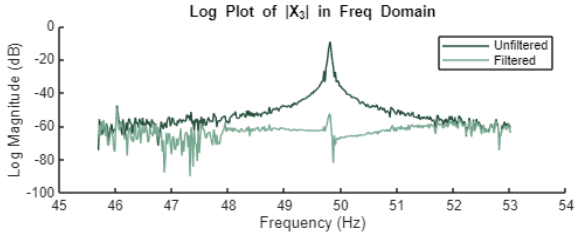

Figure 22: Response of Signal x_3 to Filter 1, Order 56

Notice that the original data is in dark green, while the filtered data is in light green. It can be seen that this

filter removes the interfering spike near 50 Hz while preserving the rest of the data, with minor adjustments. In this case, the higher order of 56 is justified, as it better preserves the information found within the signal. Across the spectrum, Filter 1 removes the interference without adjusting the signal content, as can be seen in Figure 23.

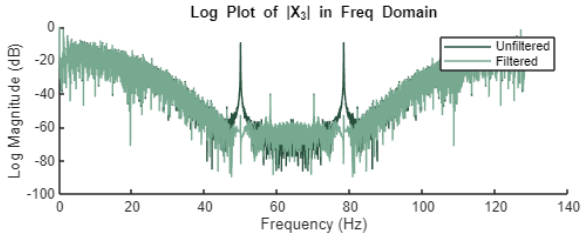

Figure 23: Entire Response of Signal x_3 to Filter 1, Order 56

Focusing in on a few heart cycles in Figure 24, we can see that the filter removed most of the contamination resulting from the power line, while preserving what appears to be the information in the data. This would need to be confirmed by a medical professional.

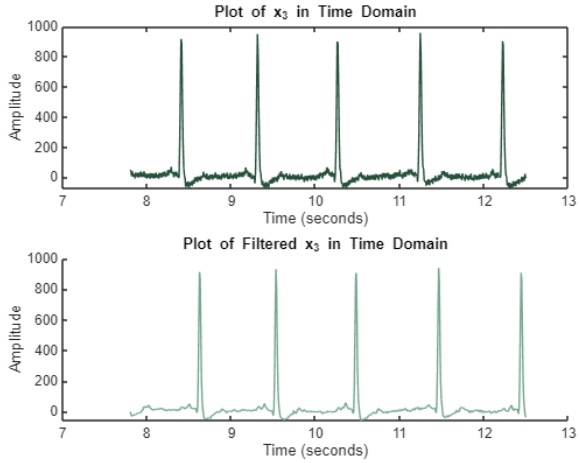

Figure 24: Signal x_3 after Filter 1, Order 56, in Time

With the interference removed, we are able to analyze the heart rhythm recorded in the ECG. First, we see that there are 35 R waves present in the first 30 seconds, as seen in Figure 25, which allows us to estimate a heart rate of 70 BPM.

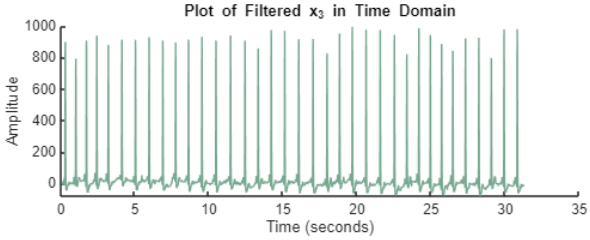

Figure 25: Signal x_3 over the First 30 Seconds

Similarly, we count 35 R waves in the last 30 seconds, as seen in Figure 26, again allowing us to estimate a heart rate of 70 BPM.

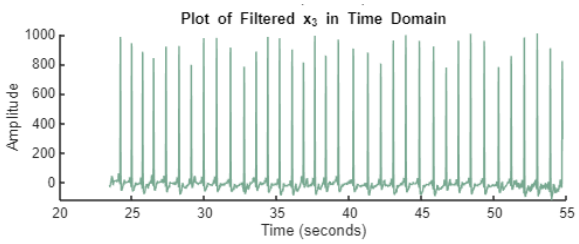

Figure 26: Signal x_3 over the Last 30 Seconds

Taking a closer look at Figure 24, we can see that there does appear to be a P wave, a very small Q wave, a large and irregular R wave, an S wave, and a T wave. However, the rhythm between R waves seems irregular, and the P and T waves are barely present. Additionally, the amplitude of the R waves is irregular. This, in many ways, appears similar to the rhythm of atrial fibrillation. However, this would need to be confirmed by a medical professional, as the interference from the power line could be corrupting the information in the data.

5 Signal x_4

Signal x_4 is an audio signal. However, it has been jammed with a wideband jammer, producing the data seen in Figure 27. This is clearly unintelligible, and the unfiltered audio can be found in the `x4_unfiltered.wav` file in the GitHub repository.

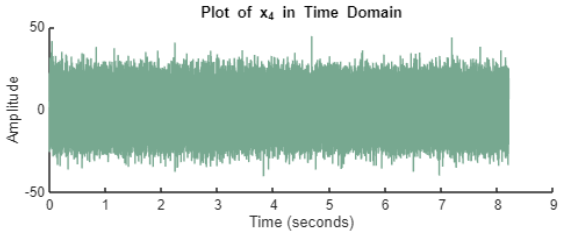

Figure 27: Signal x_4 over Time

The frequency spectrum in Figure 28 similarly shows that a large amount of noise has been added to the signal.

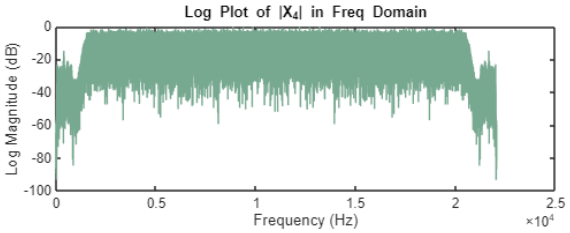

Figure 28: Frequency Spectrum of Signal x_4

However, notice that the noise appears to begin just before 1000 Hz , as can be seen in Figure 29. This will enable us to extract the majority of the information in the original audio signal, as a large portion of the speech range of frequencies has been preserved.

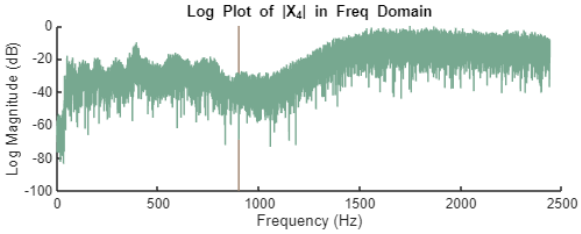

Figure 29: Frequency Spectrum of Signal x_4, focused on Audible Frequencies

In order to filter out the noise that is overpowering the audio, we will use a low pass filter. Also, because preserving linear phase is important, we will use an FIR filter. First, we will try to decipher the message, allowing our order to be higher. After several iterations, we arrive at the design for Filter 1. The parameters for Filter 1 can be seen in Table 5. The equiripple method yields the best magnitude response for a given order.

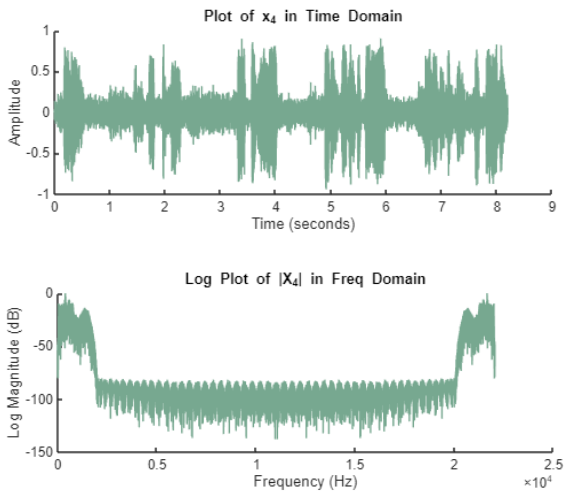

Figure 30: Response of Signal x_4 to Filter 1, Order 56

Order	Fpass	Fstop	Apass	Astop
56	800 Hz	2000 Hz	1 dB	100 dB

Table 5: Parameters for Filter 1 for Signal x_4

Applying the filter to the signal yields the following time and frequency responses in Figure 30.

This removes all of the noise, but also removes portions of the speech information. Despite this, we are still able to interpret the information. At first, I was only able to hear, "huh, what is that the light... crush your enemies... it is given before you... and walalalalalanight." Putting this into a google search brought me to a Reddit page quoting Conan the Barbarian. I followed this lead, and came across a YouTube video that contains the full quote, "Conan, what is best in life? To crush your enemies, see them driven before you, and to hear the lamentations of their women." While still distorted, it is easy to hear the words once they are known. This audio can be heard in the `x4_goodAudio.wav` file in the GitHub repository.

Pushing this idea to its maximum, progressively adjusting the parameters enabled lowering the order of the filter to order 5. The specifications for Filter 2 can be seen in Table 6.

Order	Fpass	Fstop	Apass	Astop
5	100 Hz	3300 Hz	4 dB	39 dB

Table 6: Parameters for Filter 2 for Signal x_4

Applying Filter 2 to the signal yields the time and frequency response in Figure 31.

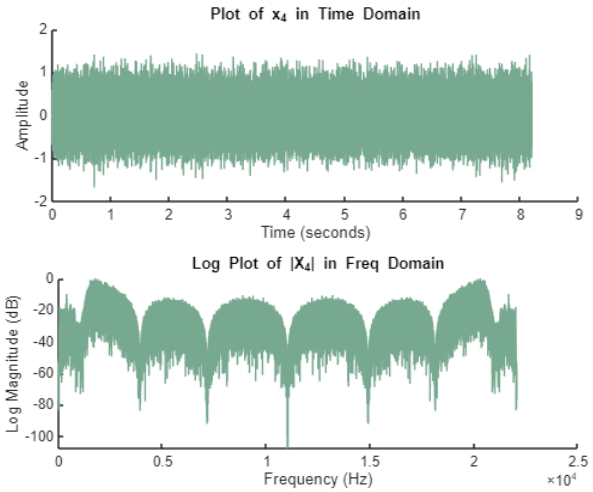

Figure 31: Response of Signal x_4 to Filter 2, Order 5

This yields quiet audio with a lot of hissing noise on top. However, it is still easy to make out the words. In many ways, the audio sounds better: it had less of the audio information removed. The audio is just less uncovered. This response can be heard in the `x4_barelyAudible.wav` file in the GitHub repository.

6 Signal x_7

Finally, Signal x_7 consists of 10 test tones, each separated by 1000 Hz. The processing goal of this signal is to make the magnitudes of the test tones rest within 1 dB of the brown line in Figure 32.

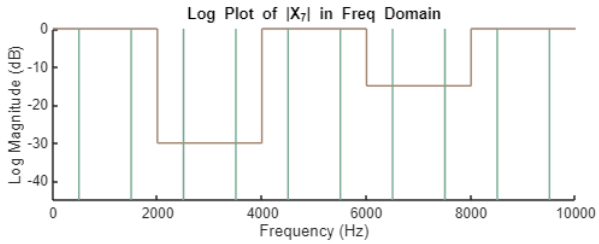

Figure 32: Frequency Spectrum of Signal x_4 with Desired Magnitude Line

In order to create the necessary filter, we will use the `firpm` function, which implements the Remez algorithm to interpolate given points in a way that minimizes the maximum error. This interpolation then is converted into a filter.

In order to make the best use of this feature, we will use the frequencies of the test tones coupled with the desired magnitudes as the points given to the `firpm` function. This can be seen in Figure 33, where the red x's represent the points given to the `firpm` function. After trying every order working up from order 3, Matlab's minimum filter order, order 50 was determined to be the first to meet these requirements. Filter 1's frequency response can be seen in blue.

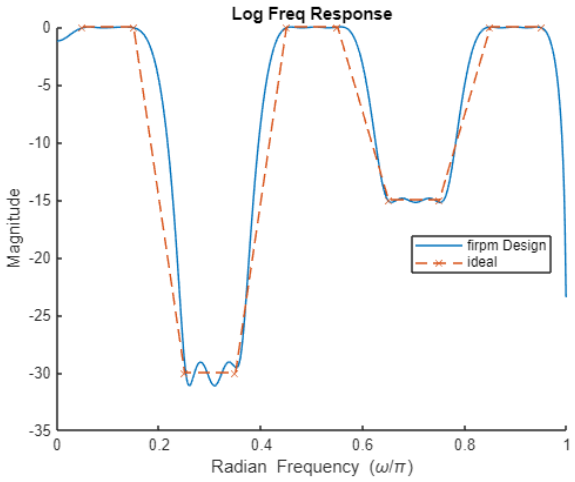

Figure 33: Frequency Response of Filter 1 for Signal x_7

These points are listed in Table 7.

Normalized Angular Frequency	Magnitude
0.05	1.0
0.15	1.0
0.25	0.031623
0.35	0.031623
0.45	1.0
0.55	1.0
0.65	0.17783
0.75	0.17783
0.85	1.0
0.95	1.0

Table 7: Parameters for Filter 1 for Signal x_7

Applying Filter 1 to Signal x_7 yields the frequency response in Figure 34, where it can be seen that all tones are very close to the desired magnitude line.

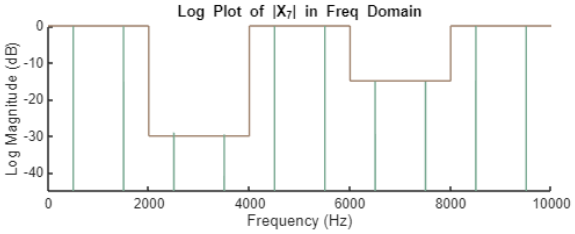

Figure 34: Response of Signal x_4 to Filter 1, Order 50

The errors at each tone are compiled in Table 8

Frequency (Hz)	Error
500	0.06762
1500	0.06377
2500	0.93816
3500	0.39825
4500	0.06582
5500	0.00000
6500	0.10854
7500	0.22556
8500	0.00107
9500	0.06390

Table 8: Errors for Tones in Signal x_7

From this, we can see that the maximum error is 0.93816, and so every tone is within the 1 dB tolerance that is required. We can also calculate that the mean error is 0.1932 dB.

However, notice that the Remez algorithm minimizes the maximum absolute difference between the polynomial and the function, but does not minimize the maximum absolute error at each of the given points. Thus, `firpm` minimizes the error off the of the dashed red line in Figure 33, but does not ensure that the polynomial passes through every point. In this application, because we are primarily interested in the values at the 10 test tones, if we instead used a method such as Lagrange polynomial interpolation or cubic spline polynomial interpolation, we could get the required error with a much lower order.

7 Conclusion

For each of the five signals provided for Computer Exercise 2 (CPX 2), the processing goal was met with a filter of minimum order. In summary, in Signal x_1, the louder tone was attenuated so that the quieter tone was more than 30 dB greater in magnitude. In Signal x_8, the secrets of extracting sunshine from cucumbers was revelaed after an attempted jamming. In Signal x_3, interference from a power line in an electrocardiogram (ECG) was removed. In Signal x_4, the secret to a good life according to Conan the Barbarian was revealed after another attempted jamming. In Signal x_7, ten test tones were modified to be within 1 dB of a specified magnitude.

For the filters for Signals x_1 and x_8, preserving linear phase was not important, and thus infinite impulse response (IIR) filters were used because they have a better magnitude response for a given order. For Signals x_3 and x_4, preserving linear phase was important, and thus finite impulse response (FIR) filters were used. For details on each filter, see the signal's respective section.

Throughout this experience, I gained experience working with digital filters, in both design and application. Additionally, I built upon my "signal hunting" skills that had been built in CPX 1 in order to understand the content of the signals, a necessary prerequisite to processing them appropriately. I also gained more experience with audio processing in Matlab, and learned how to export to `.wav` files. Additionally, I gained more experience in LaTeX, which is always helpful.

For access to the Matlab `.mlx` files, source signal data, filter designer session `.fda` file, filter taps `.bin` files, and output `.wav` sound files, see the project's GitHub repository at `https://github.com/dbcometto/ece434_cpx2`.

Documentation

I did all my own work. I had various conversations with classmates, including C1C Csicsila and C1C Chen. However, no changes were made based on those conversations. Additionally, I used Google (`https://brainly.com/question/2233369`) to find the cucumber quote and YouTube (`https://www.youtube.com/watch?v=V30tyaXv6EI`) to find the Conan the Barbarian quote. I used various resources, including overleaf.com, for Latex help. I also used various ECG interpretation resources, including this article, several YouTube videos, and several Google images of atrial fibrillation. I also used the Mathworks website for Matlab help, for various syntax issues, and additionally for the `upsample` function, in an attempt to play the ECG data (until I learned that real ECGs use sonification to play the data, which makes way more sense). I also used my notes from Math 342 Numerical Analysis and the Remez algorithm Wikipedia page.