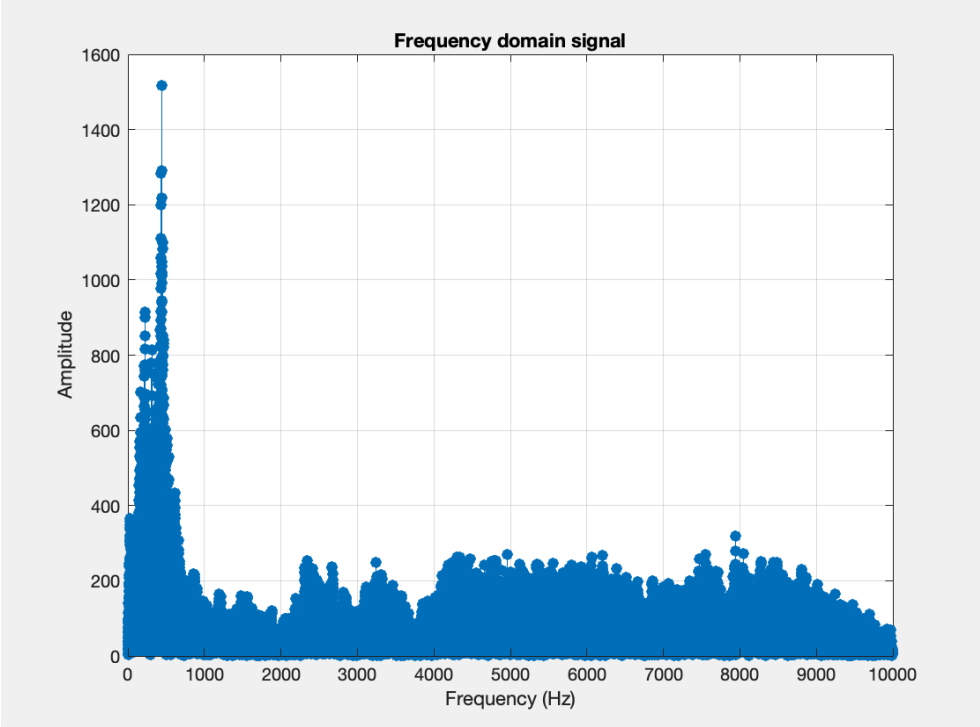
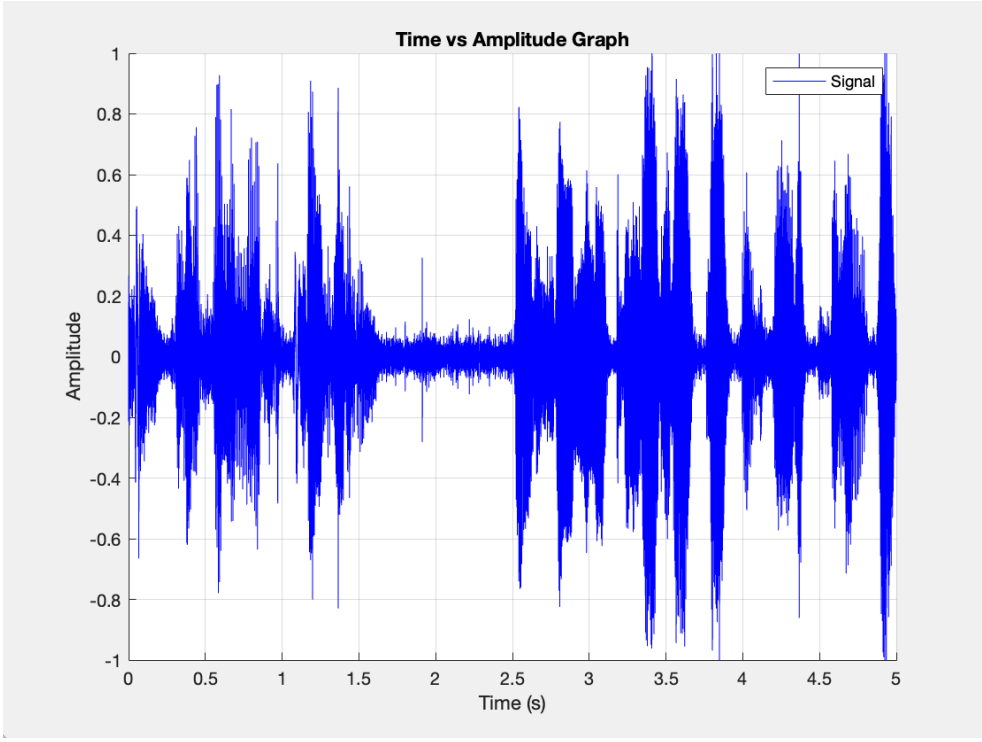
CMPE362

KADIR GOKHAN SEZER
2018400369
HOMEWORK 3

Before starting to work on the signal, I wanted to plot the Frequency Spectrum and amplitude-time graph. It helped me to see the signal visually. The sound became a signal for me after being able to see it instead of hearing it.



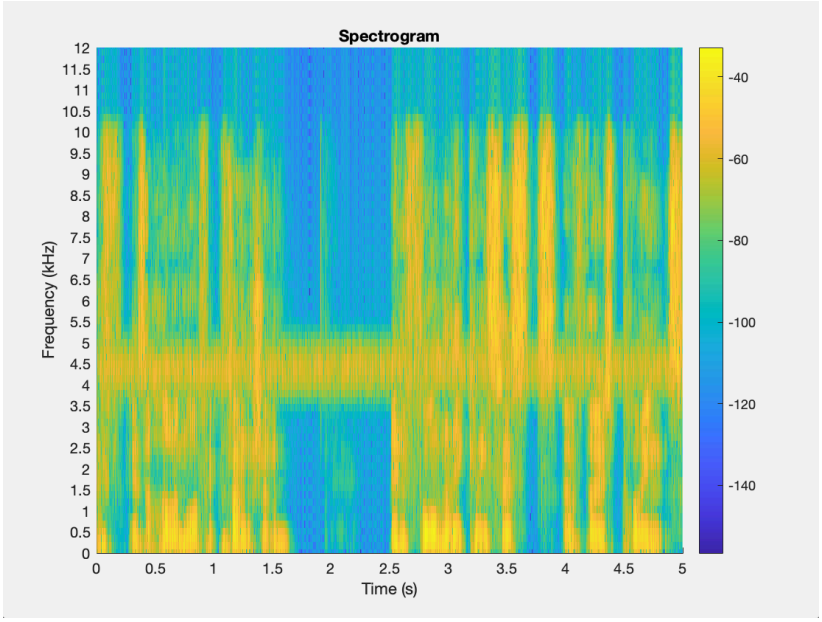


Instruction 1

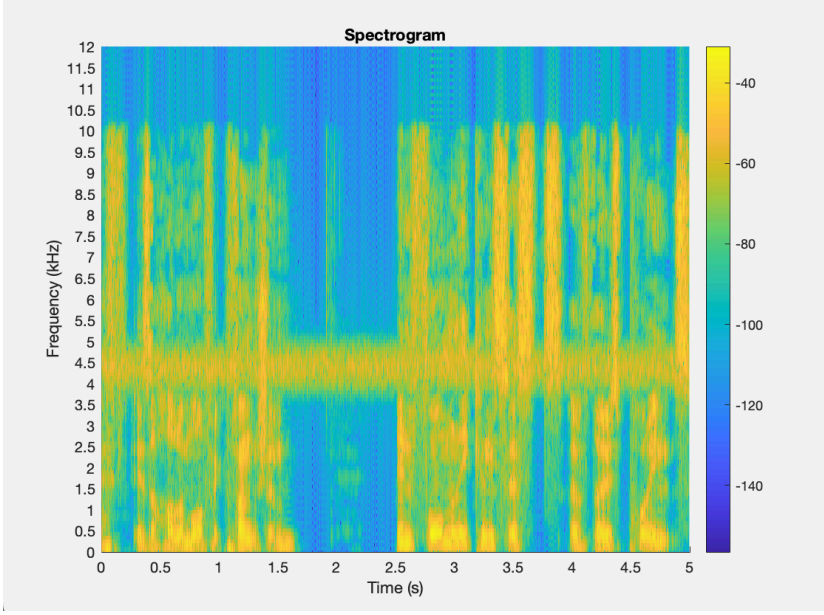
```
[s,f,t] = spectrogram(x,window,noverlap,f,fs)
```

We were told to use this function to plot the required representation. The first topic for me was to decide the parameters to set for this function. I saw some examples on the internet and tried all.

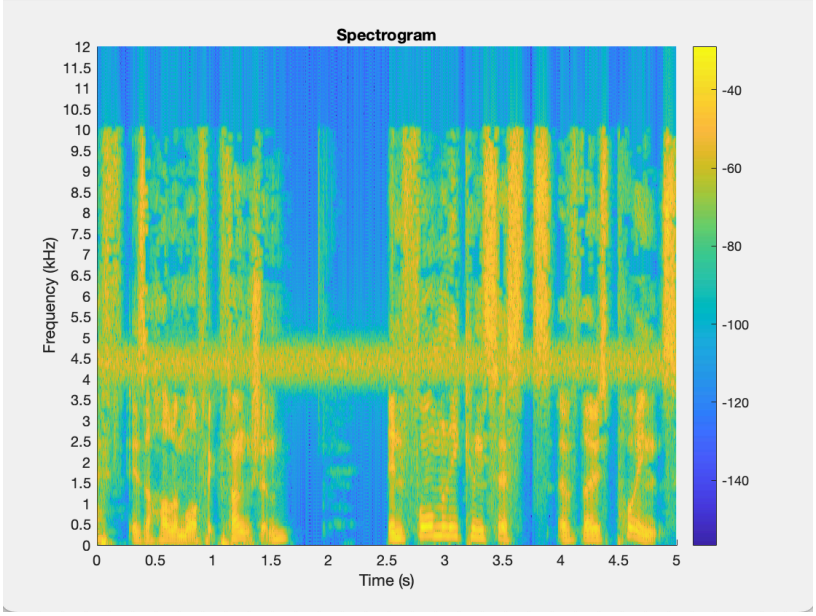
`For window=128, noverlap=100, f=256`



`For window=256, noverlap=200, f=512`

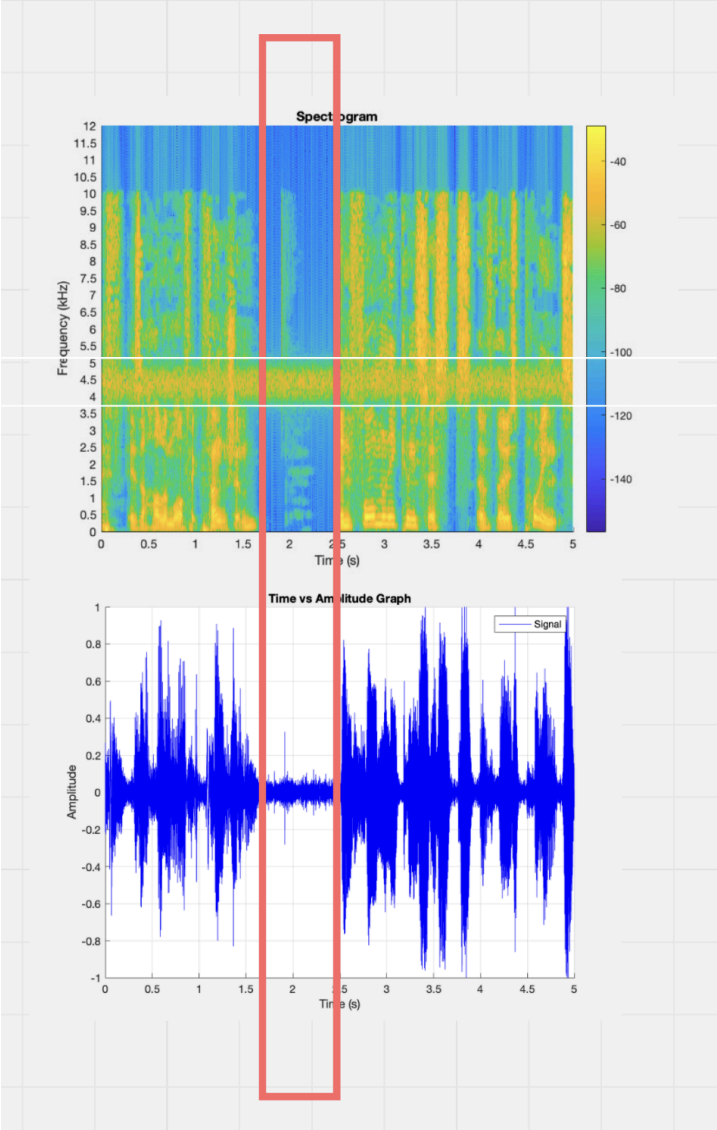


`For window=512, noverlap=400, f=1024`



As you can see, the spectrogram gains more resolution as we give a larger window size. However, the behaviour of the signal keeps its shape. Therefore, we can go with any of them, but I chose the window=512 one.

Since there is no need to see the frequencies above 12, I used `ylim([0 12])`.
To see more resolution on the y line, I used `yticks(0:1/2:12)`.



From the picture on the left, we can see the red window, which is silent. There is no sound from the instructor in the audio, and we can confirm this by listening as well. However, even though the instructor is not speaking, there is still some amplitude within that window range. It seems we have some noise during that period.

The noise may last from the beginning to the end, but since the instructor is speaking at certain times, we cannot hear the noise during those moments. The key here is to check the signal during the periods when only noise is present.

The time-amplitude graph helps us identify when the instructor is not speaking, while the spectrogram, thankfully, shows us which frequencies have amplitude within that small window.

The next step was to find the frequency of the noise, which becomes very simple after examining the picture on the left, right? This is the skill that the CMPE362 course has taught us.

Between 1.5 and 2 seconds, the instructor is not speaking, but there is a small amplitude present. The frequency of that noise is between 4kHz and 5 kHz. It can easily be seen within the white window.

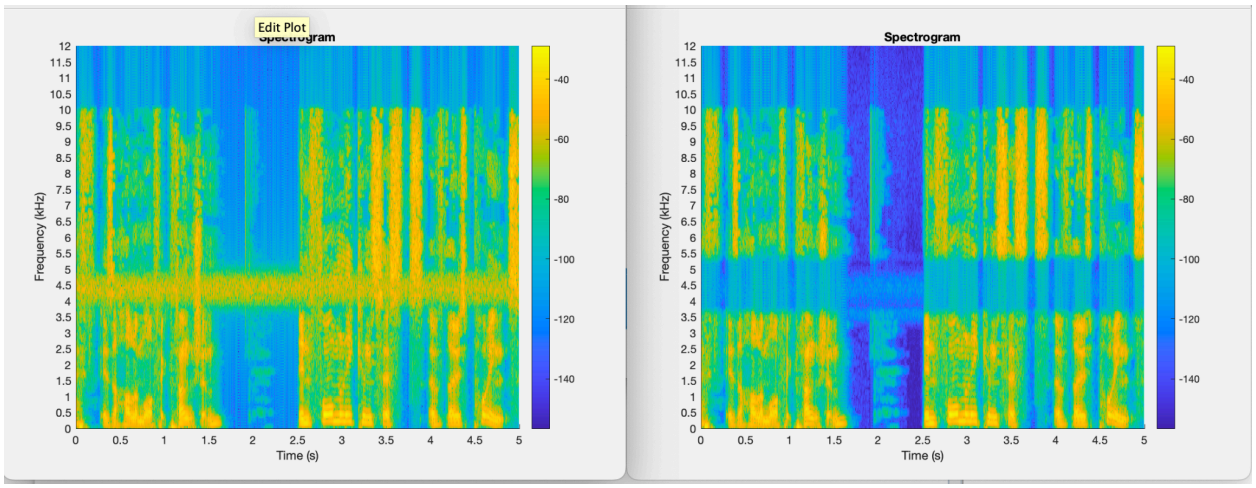
Instruction 2

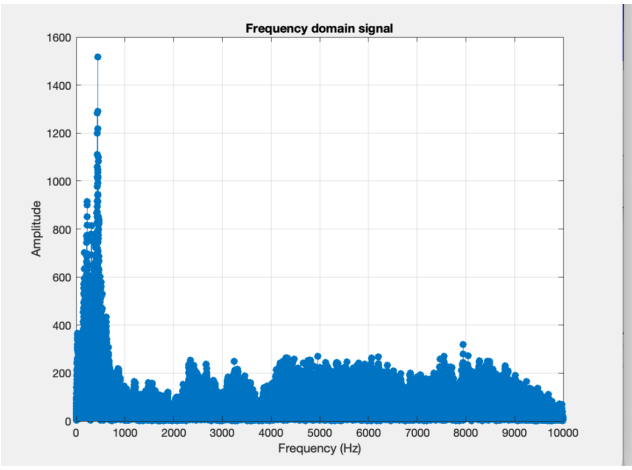
Design a 256th-order FIR bandstop filter using the fir1() function. Use ($f_1 - 500$, $f_2 + 500$) as cutoff frequencies. If your estimated frequency range (f_1, f_2) is sufficiently accurate, this filter should significantly reduce the noise (almost inaudible).

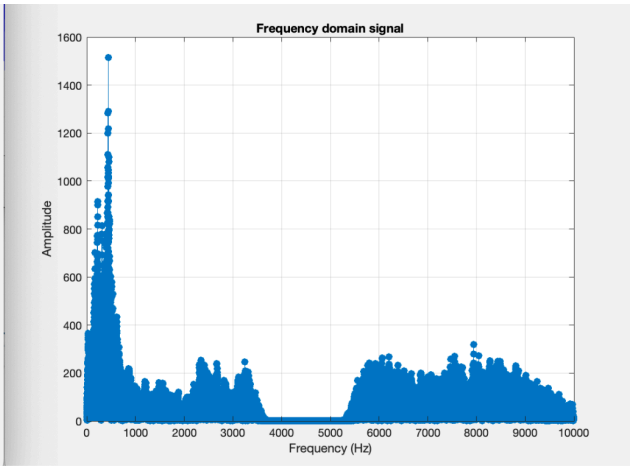
From Instruction 1, we have found f1 = 4 KHz and f2 = 5 KHz. The next step is just to make the calculations.

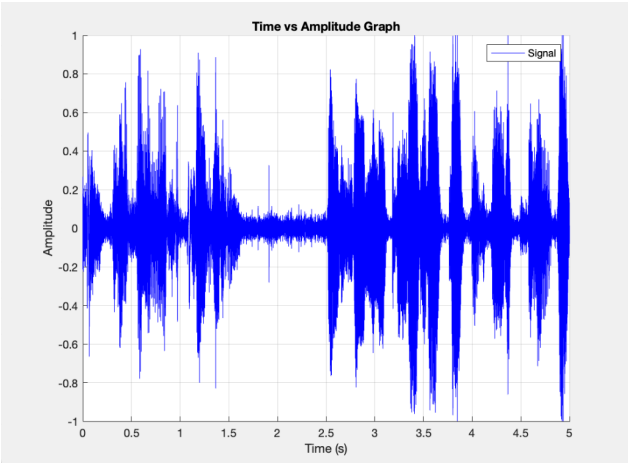
```
f1 = 4000;
f2 = 5000;
low_cutoff  = (f1 - 500) / (fs/2);
high_cutoff = (f2 + 500) / (fs/2);

b           = fir1(256, [low_cutoff high_cutoff], 'stop');
y_filtered  = filter(b, 1, y);
```

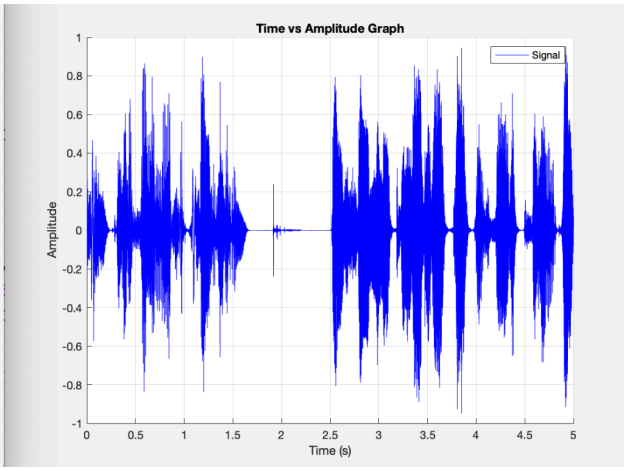
Before and after the filtering.




Frequency domain signal
Amplitude
Frequency (Hz)
0
200
400
600
800
1000
1200
1400
1600
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Frequency domain signal
Amplitude
Frequency (Hz)
0
200
400
600
800
1000
1200
1400
1600
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Time vs Amplitude Graph
Signal
Amplitude
Time (s)
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Time vs Amplitude Graph
Signal
Amplitude
Time (s)
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Instruction 3

Butterworth

For the Butterworth filter, I need another parameter, which is the order. I chose 6 by reading the official documentation.

```
function y_butter = ButterworthFilter(low_cutoff, high_cutoff, order, y)
    [b_butter, a_butter] = butter(order, [low_cutoff high_cutoff], 'stop');
    y_butter             = filter(b_butter, a_butter, y);
end
```

Chebyshev Type I

```
function y_cheby1 = Chebyshev1Filter(low_cutoff, high_cutoff, order, y, Rp)
    [b_cheby1, a_cheby1] = cheby1(order, Rp, [low_cutoff high_cutoff], 'stop');
    y_cheby1              = filter(b_cheby1, a_cheby1, y);
end
```

Elliptic Filter

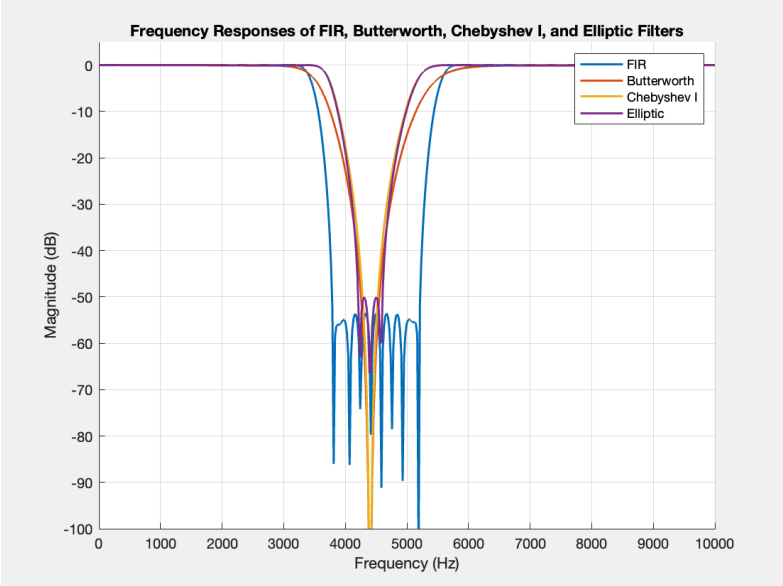
For the Rs parameter, I chose to use 50, which is written on the official page.

```
function y_ellip = EllipticFilter(low_cutoff, high_cutoff, order, y, Rp, Rs)
    [b_ellip, a_ellip]   = ellip(order, Rp, Rs, [low_cutoff high_cutoff], 'stop');
    y_ellip              = filter(b_ellip, a_ellip, y);
end
```

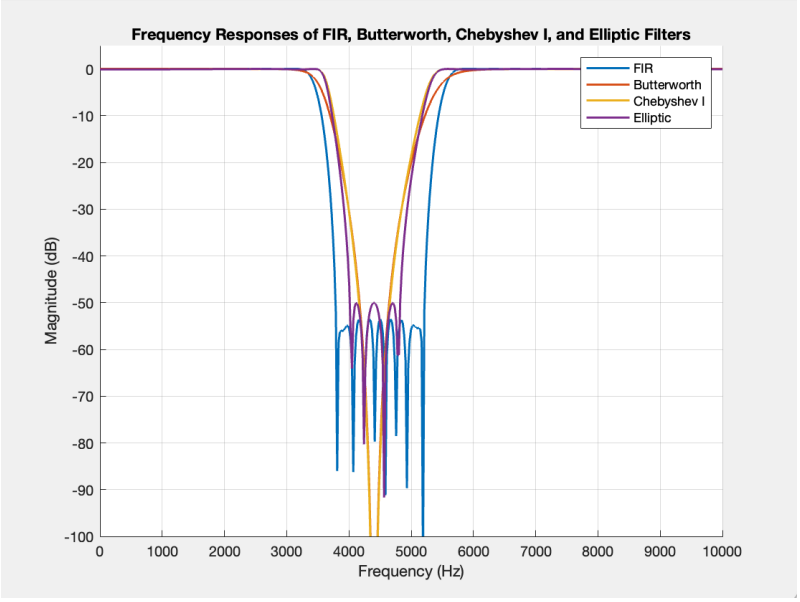
Instruction 4

In this section, we need to find an n-value to change the behaviour of the filter mechanism. FIR does not depend on the n-value, so it will not change accordingly.
As the description of the project says, "*Here the term 'similar performance' is not very well defined and subjective",* the n-values will be chosen accordingly.

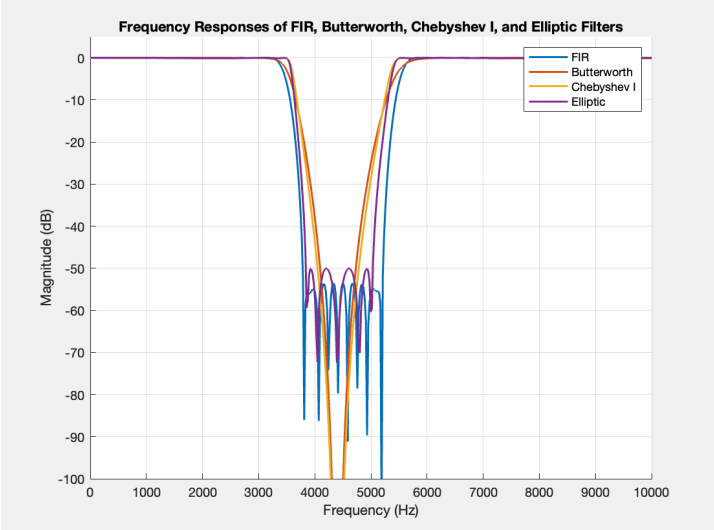
N=3, Frequency Responses of FIR



N=4, Frequency Responses of FIR

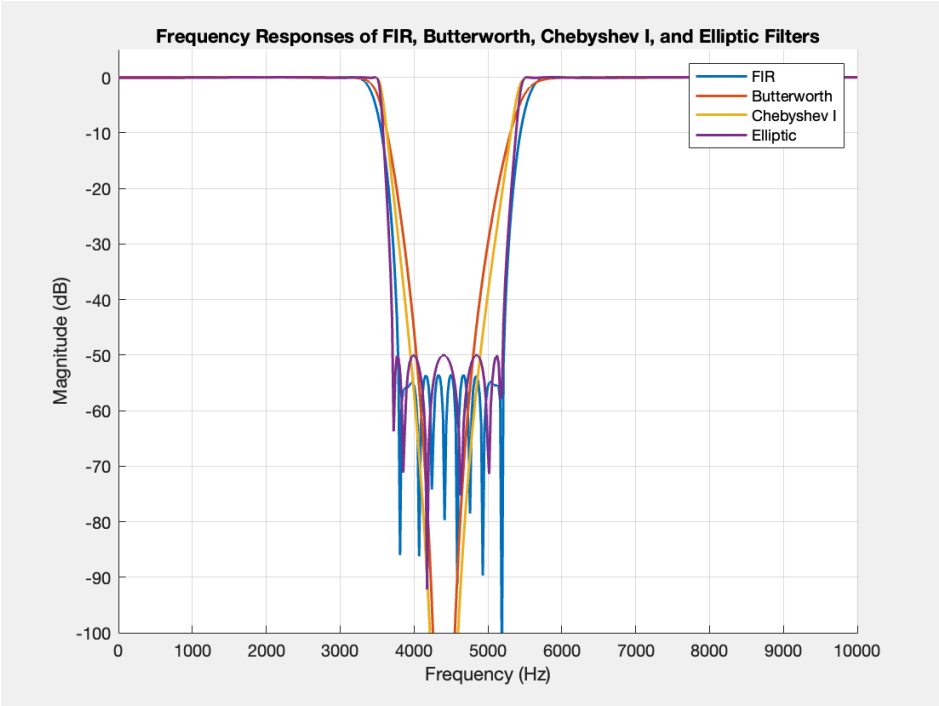


N=5, Frequency Responses of FIR

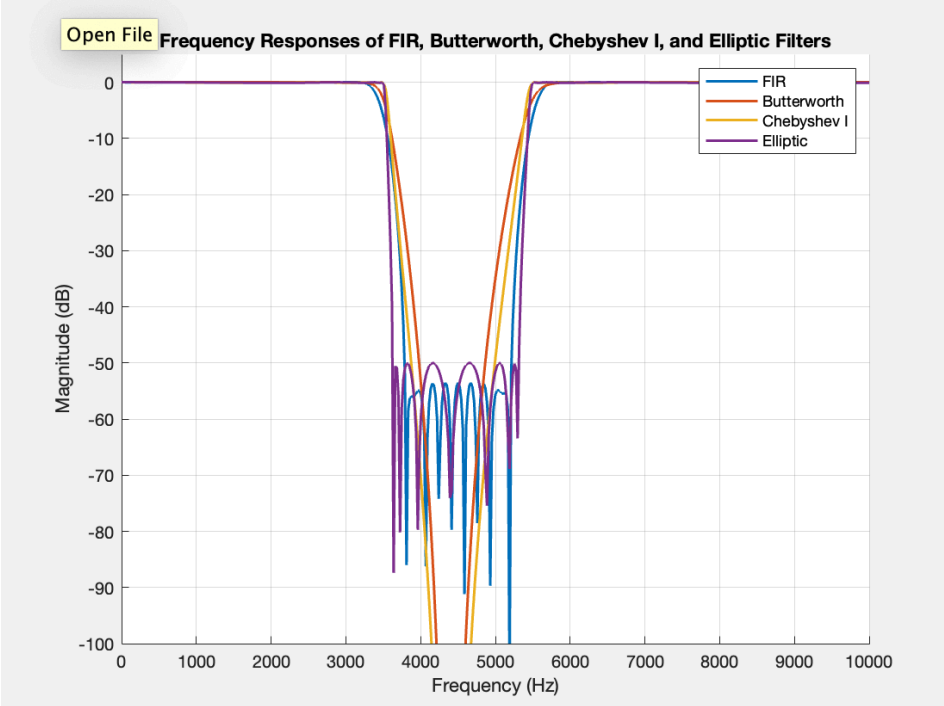


For elliptic, n=5 can be chosen since the other n values would seem inappropriate for this filtering mechanism.

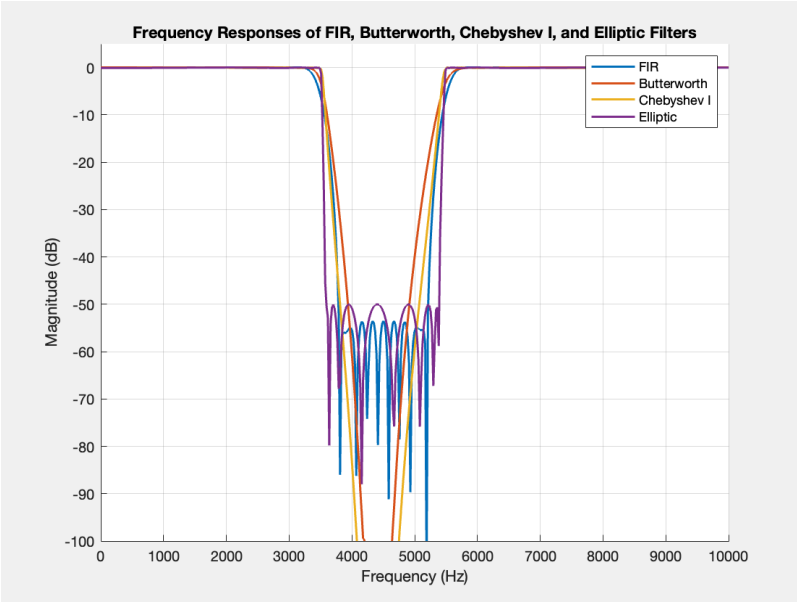
N=6, Frequency Responses of FIR



N=7, Frequency Responses of FIR

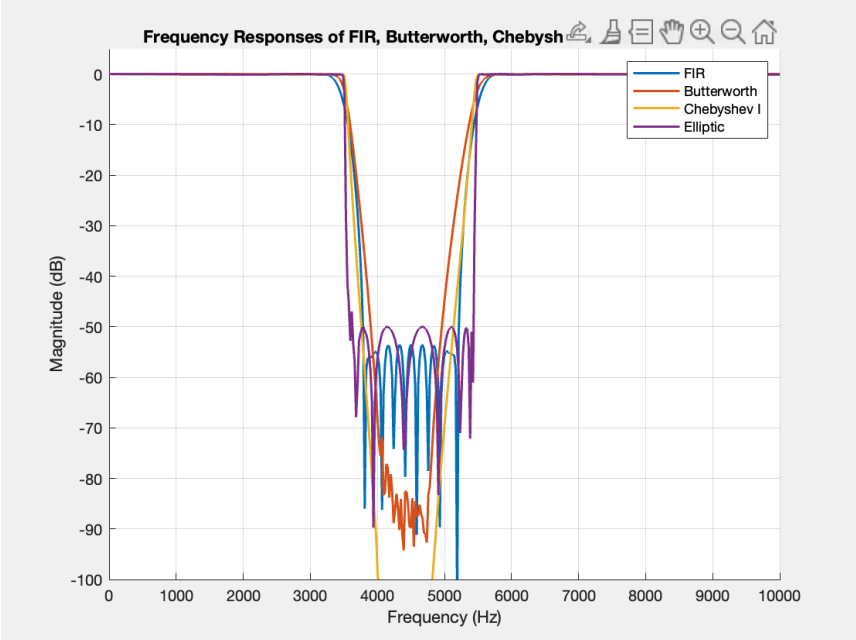


N=8, Frequency Responses of FIR

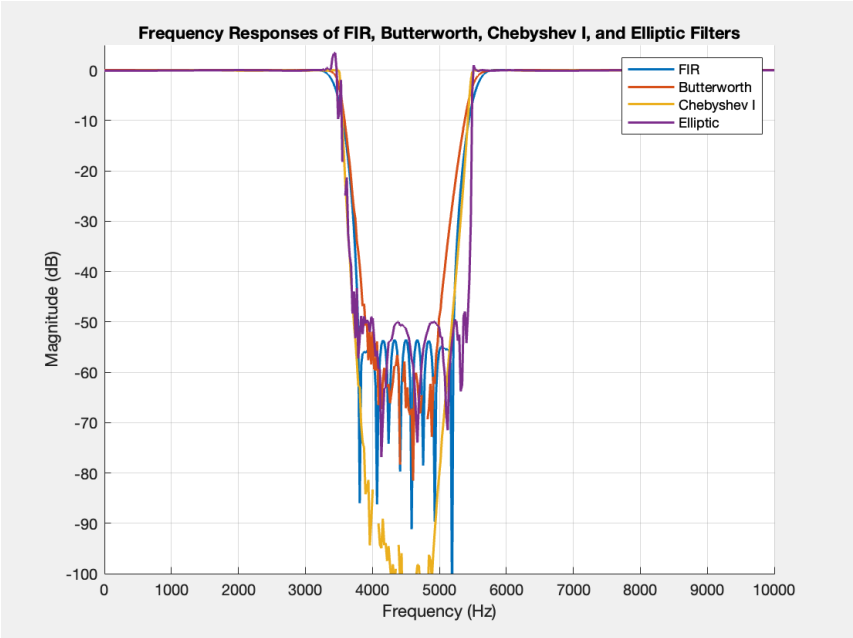


For Chebyshev Type 1, I chose n=8 since n>8 seems to stop more values than required, but it still depends on the application.

N=9, Frequency Responses of FIR



N=10, Frequency Responses of FIR

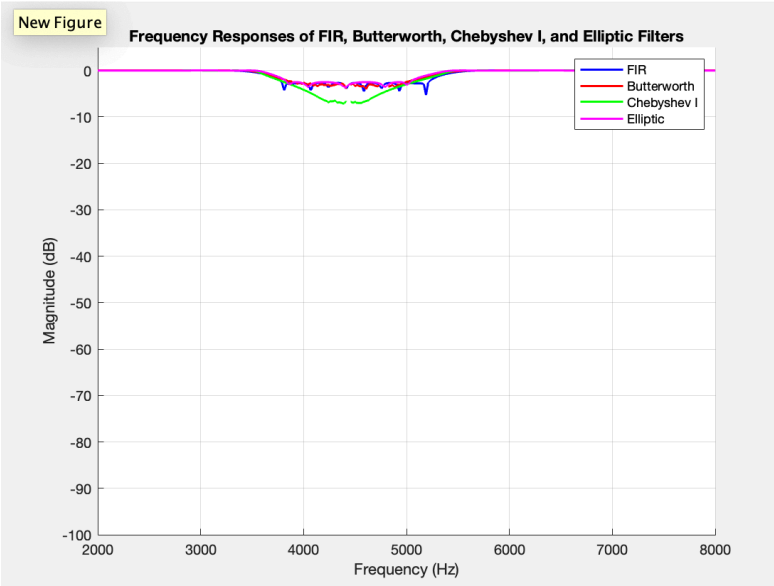


For Butterworth, I also chose n=10. The values n<10 and n>10 do not seem okay; n=10 would work.

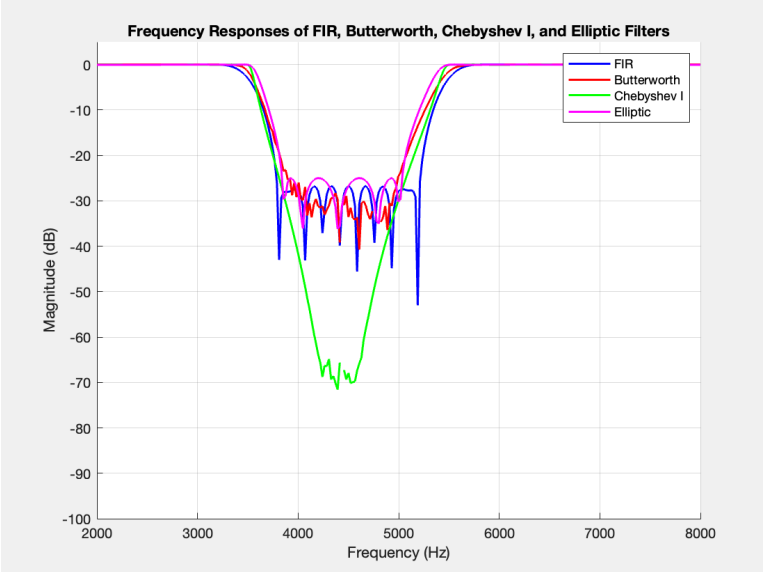
Final view

After choosing n=10 for Butterworth and n=8 for Chebyshev Type I, and n=5 for Elliptic Filtering, the final view is below.

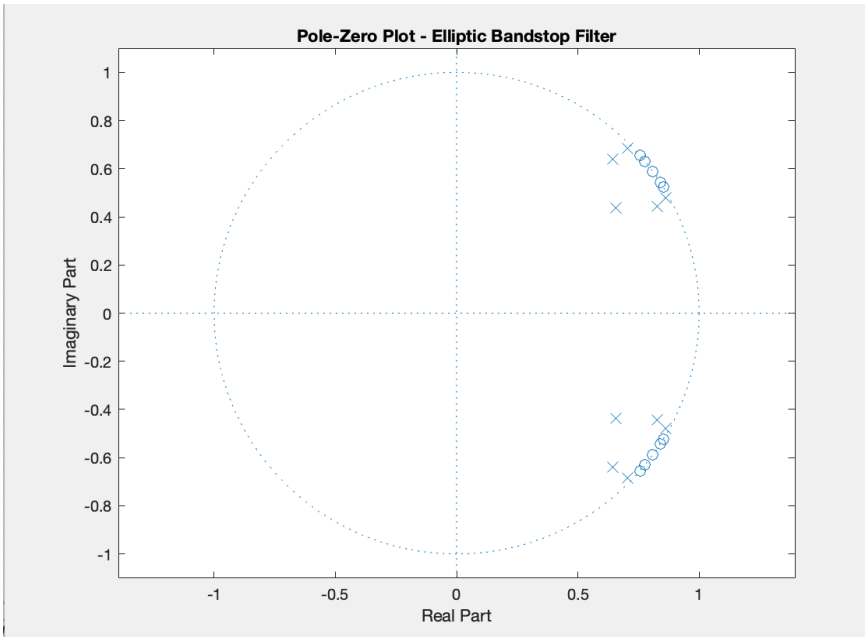
Using `1 * log10(abs(H_fir))`



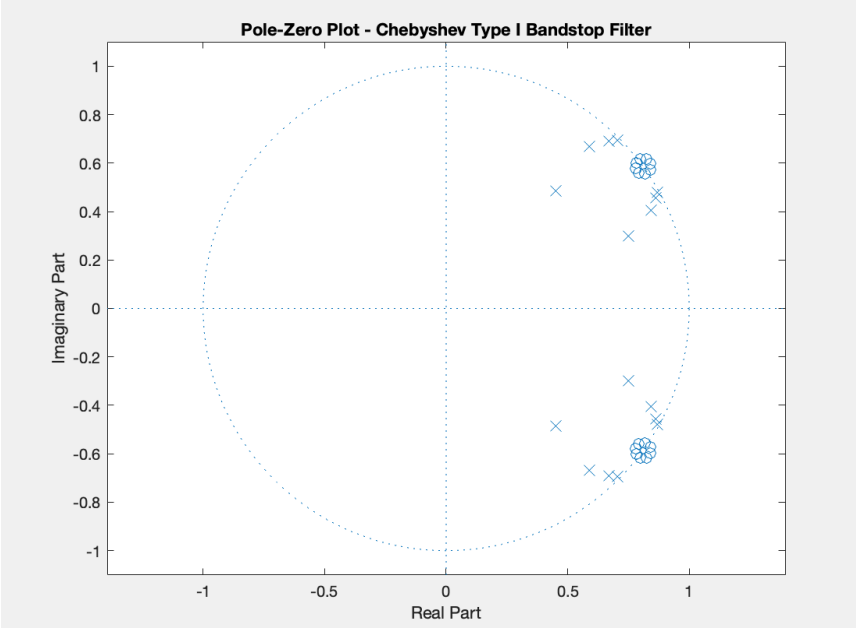
Using `10 * log10(abs(H_fir))`



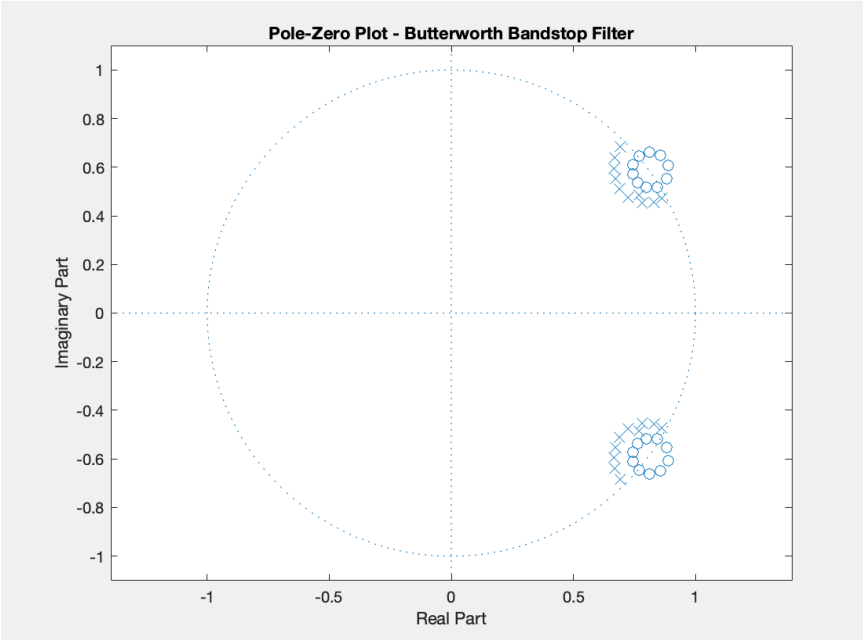
Pole-Zero Elliptic



Pole-Zero Chebyshev Type 1

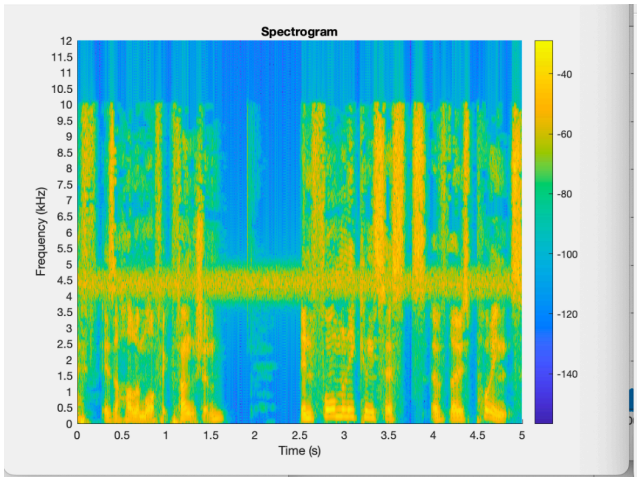


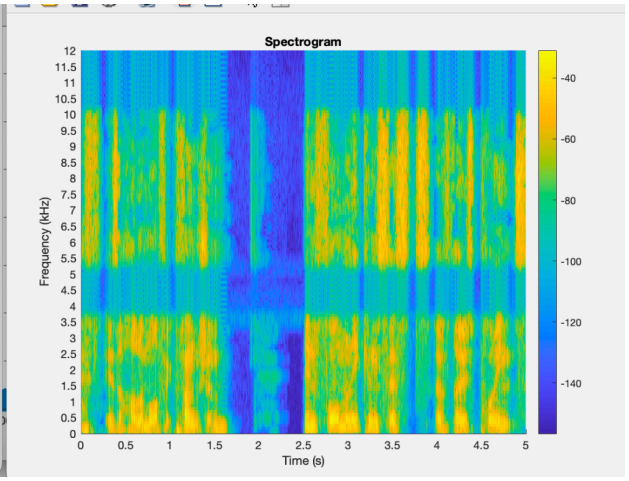
Pole-Zero Butterworth

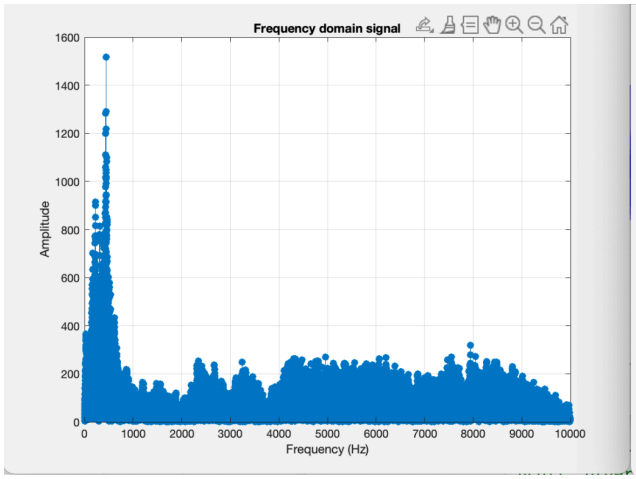


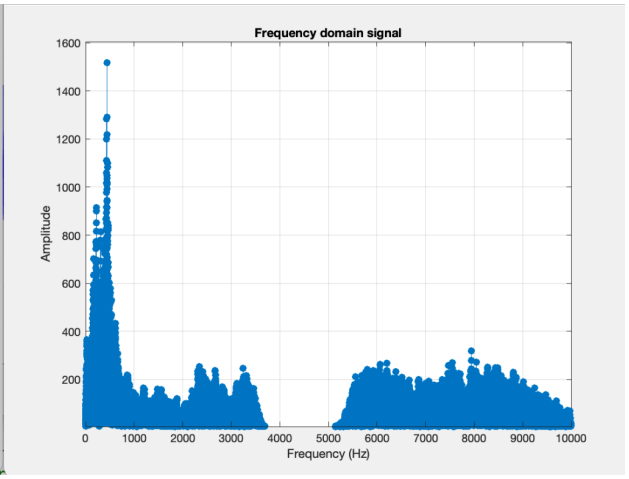
Instruction 5

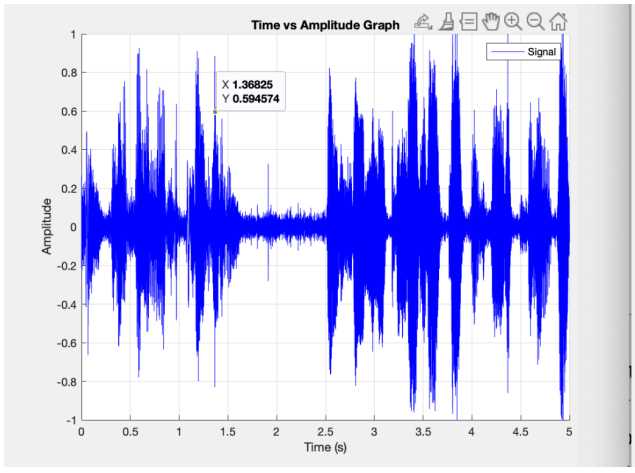
Butterworth Filt. Results

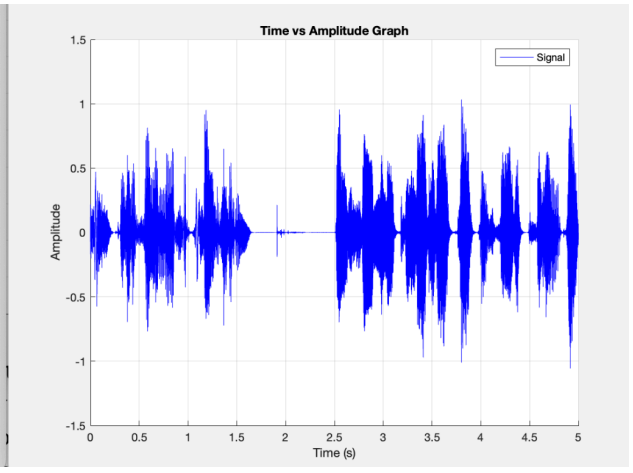




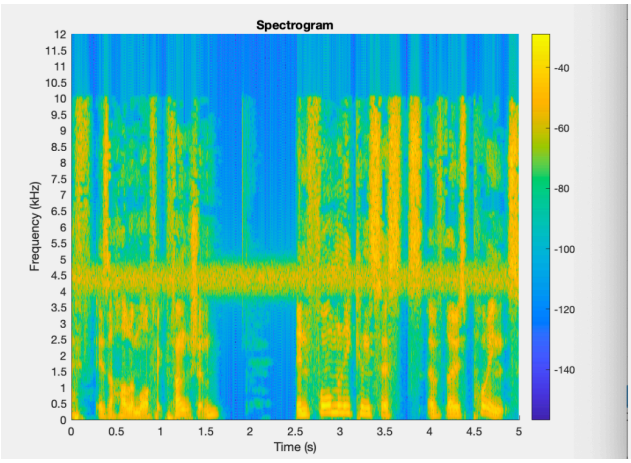


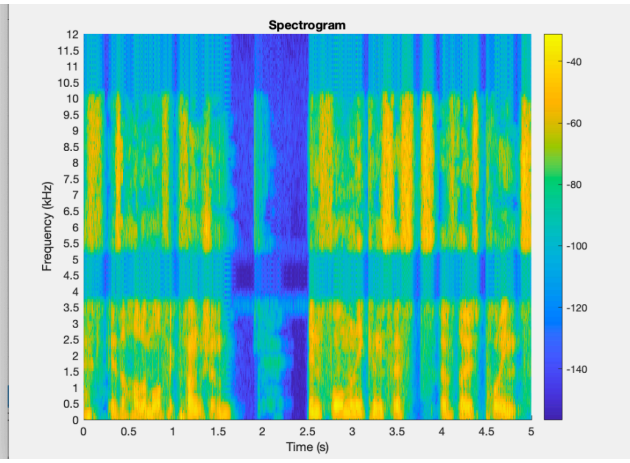


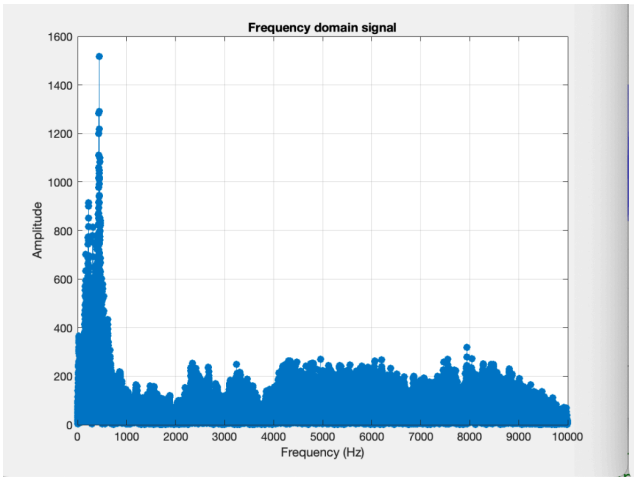


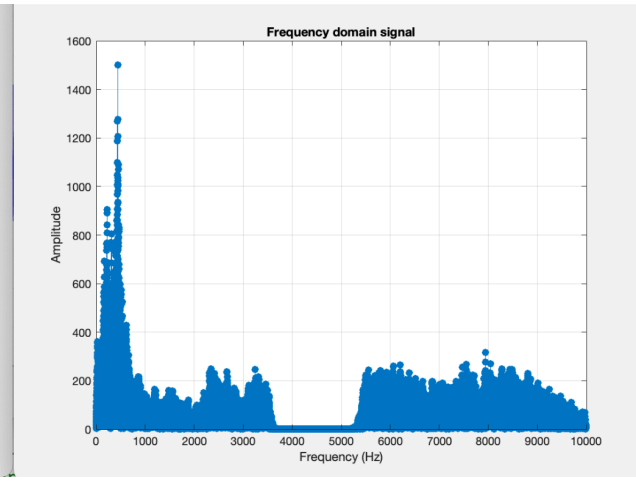


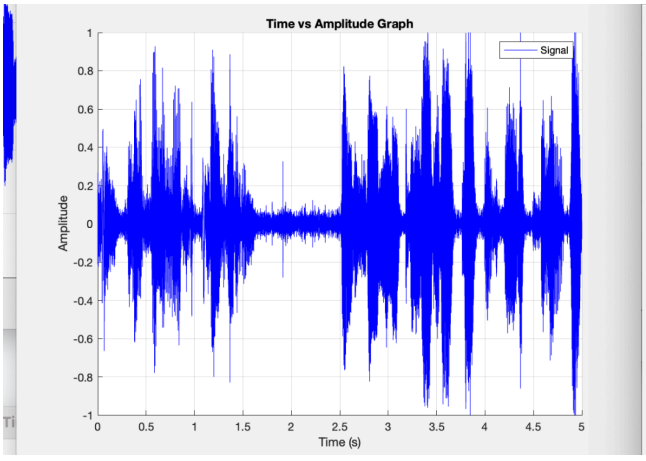
Chebyshev Filt. Results

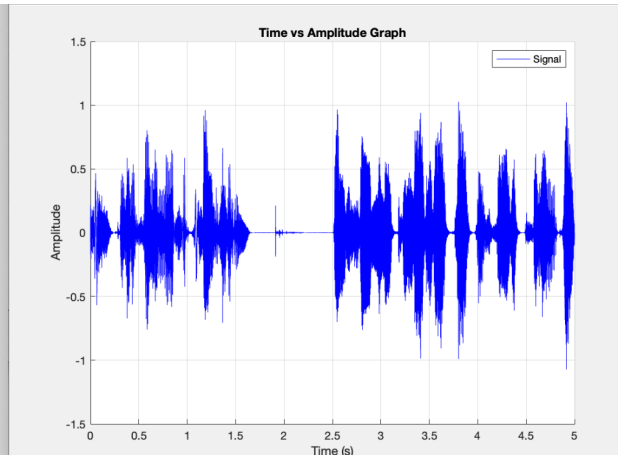




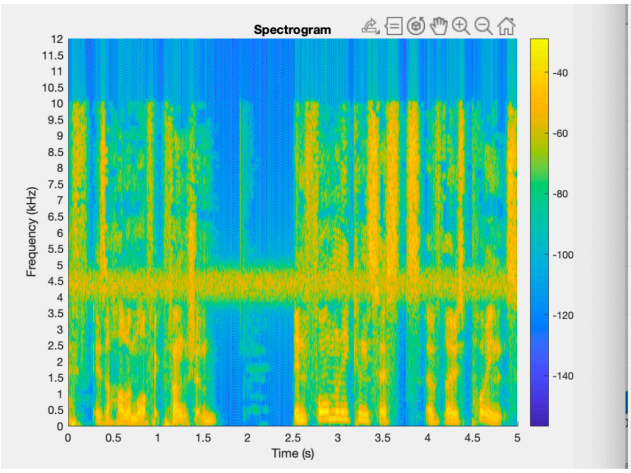


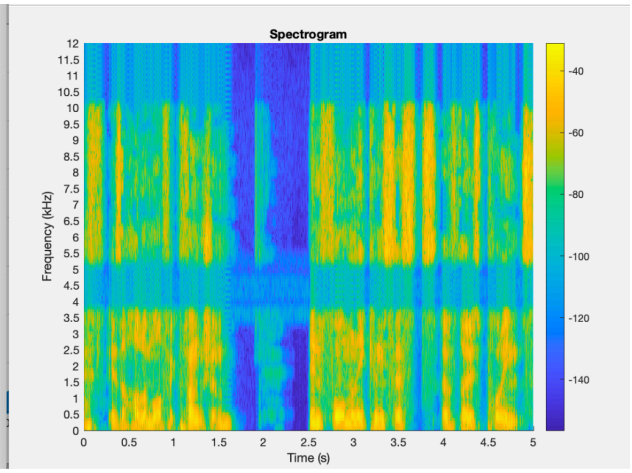


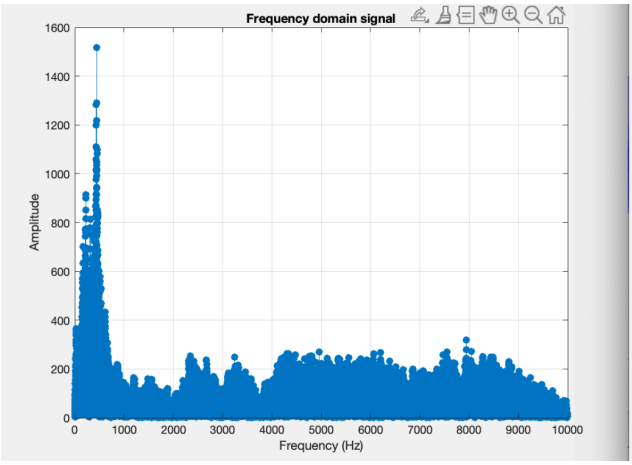


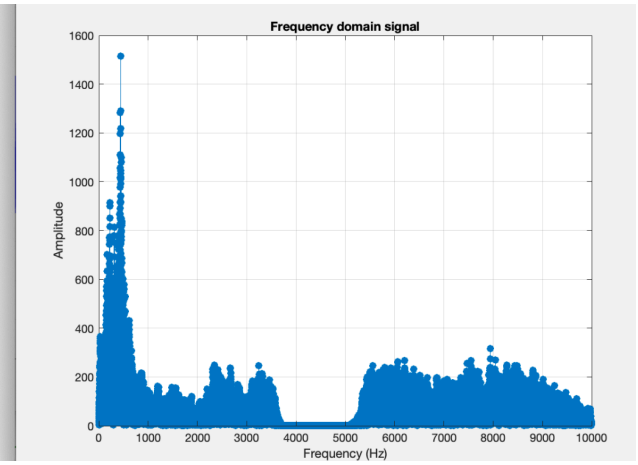


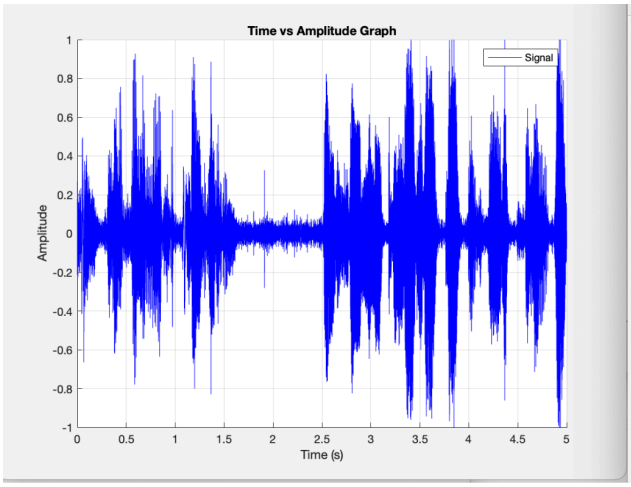
Elliptic Filter Results

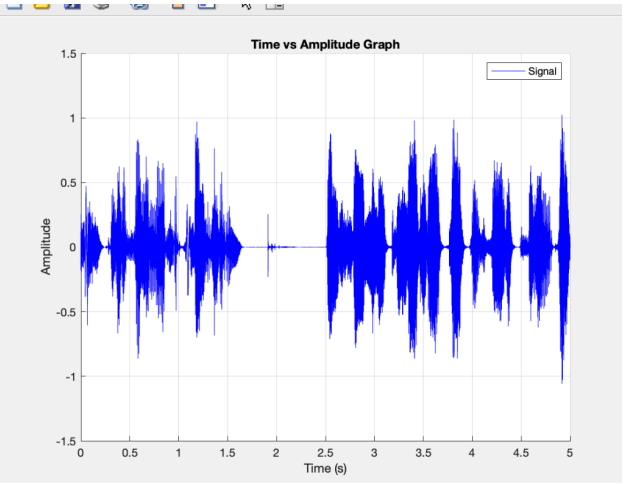






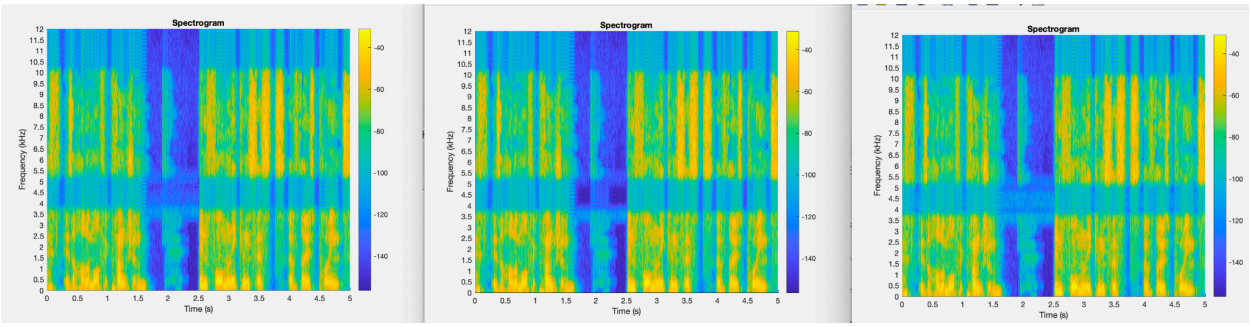


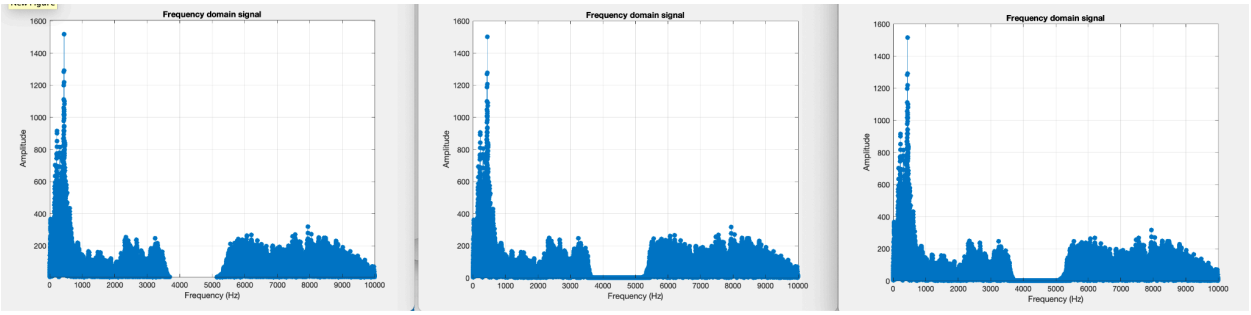


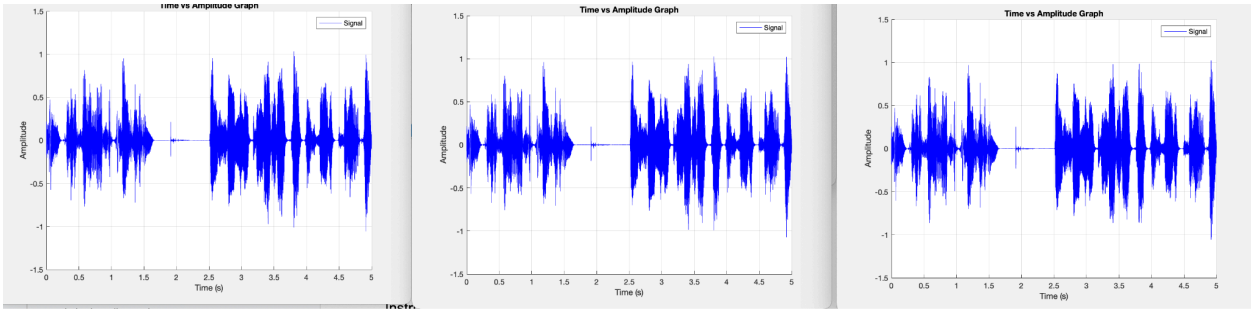


Comparison of the filtering mechanisms

Butterworth - Chebyshev - Elliptic







Instruction 6

For this instruction, I tried to increase the n values one by one and listen to the sound.

Butterworth

```
Until order=12, everything was okay.
sound(ButterworthFilter(lcoff, hcoff, 12, y), fs);
   For order=12, the sound came back as it was.
sound(ButterworthFilter(lcoff, hcoff, 13, y), fs);
   For order=13, the sound was not recognizable.
```

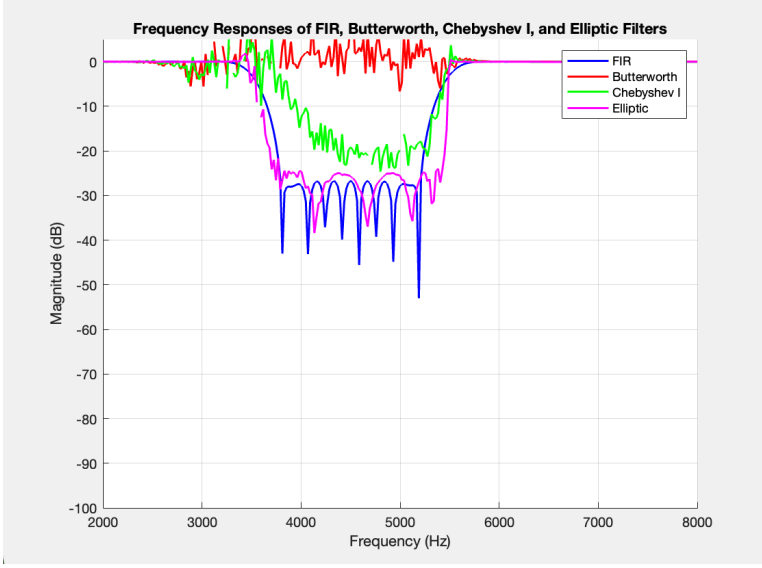
Chebyshev Type 1

```
Until order=11, everything was okay.
sound(Chebyshev1Filter(lcoff, hcoff, 12, y), fs);
   For order=12, the sound has some unwanted sounds, but it is
   recognizable.
sound(Chebyshev1Filter(lcoff, hcoff, 13, y), fs);
   For order=13, the sound was not recognizable.
```

Elliptic

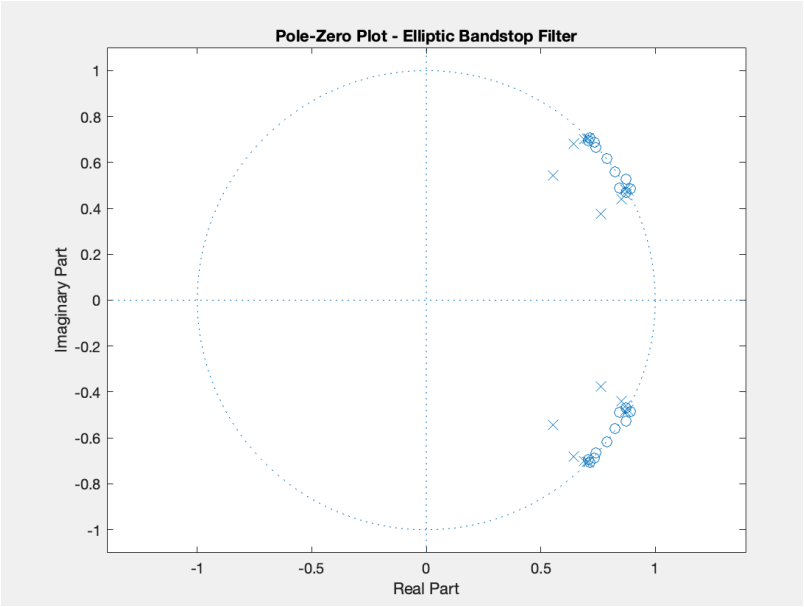
```
Until order=9, everything was okay.
sound(EllipticFilter(lcoff, hcoff, 10, y), fs);
   For order=10, the sound was not recognizable.
```

N=13 for Butterworth
N=13 for Chebyshev Type 1
N=10 for Elliptic

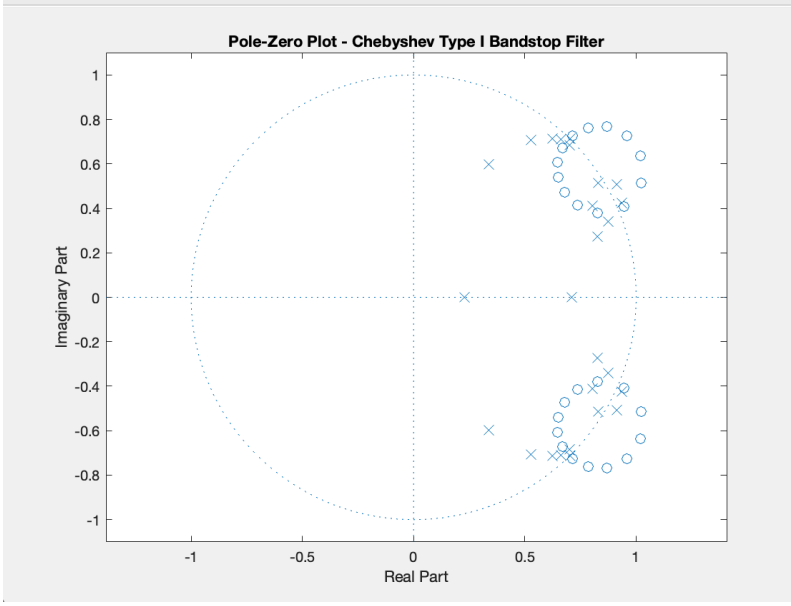


As you can see, instead of stopping some frequencies, it amplifies more. So, from the graph above, we can observe that this is corrupting the signal.

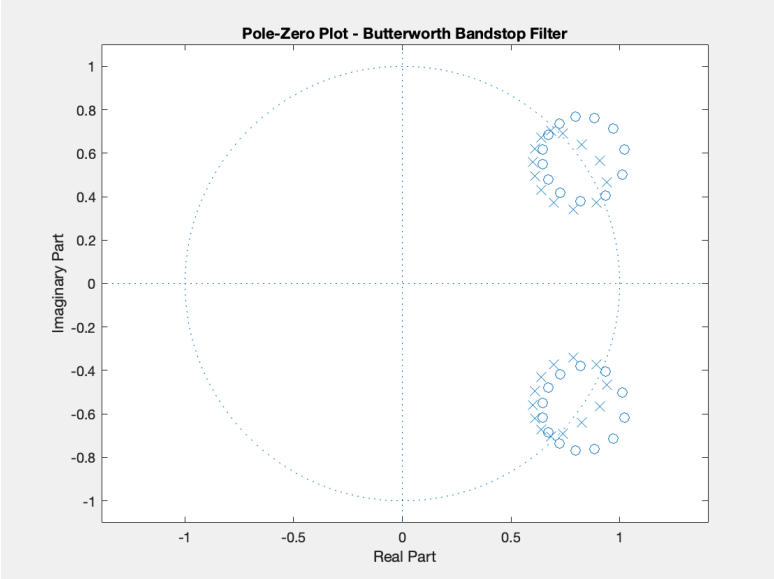
Pole-Zero Elliptic



Pole-Zero Chebyshev Type 1



Pole-Zero Butterworth



Resources:
https://www.mathworks.com/help/signal/ref/spectrogram.html
https://miro.com for drawing the pictures.
https://chatgpt.com/ for grammar fixes.