ECE 351-51

April 26, 2022

Lab 12

Submitted By :
Kennedy Beach

Contents

1 Introduction

The purpose of this lab was to analyze a noisy signal, then filter out the desired frequencies between 1.8 kHz to 2 kHz by isolating those frequencies and designing a filter. Bode plots that fit the following criteria were also derived:

- The position measurement information is attenuated by less than -0.3dB
- The low-frequency vibration noise must be attenuated by at least -30dB
- The switching amplifier noise must be attenuated by at least -21dB
- All noise that exists at frequencies greater than 100kHz must be completely attenuated (magnitudes less than 0.05V can be considered completely attenuated for all practical purposes in this situation)

2 Equations

The equations below were sourced from Circuits II material and values for R and C were guess and checked until the bode plots fit the requirements. The inductor value was set initially so the values could be found. A desired frequency of 1.9 kHZ was used since it is between 1.8 kHz and 2 kHz. Once the capacitor value was found, a bandwidth value of 1.1 kHz was used with it to find the resistor value. The bandwidth value had to be adjusted to meet the right attenuation criteria.

Transfer function from our bandpass filter design

$$H(s) = \frac{kBs}{s^2 + Bs + w^2}$$

$$H(s) = \frac{\frac{s}{RC}}{s^2 + \frac{s}{RC} + \frac{1}{LC}}$$

Choose a value for L and solve for C

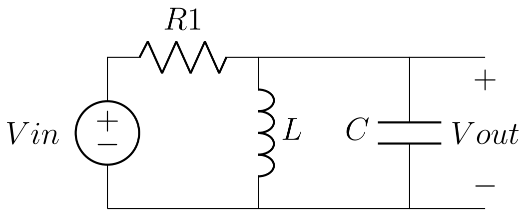
$$W^2 = \frac{1}{LC}$$

L = 200 mH, C = 35.1 nF

Choose a value for the bandwidth and solve for R

$$B = \frac{1}{RC}$$

B = 1.1 kHz, R = 4122 Ohms



3 Methodology

The first part involved plotting the signal from the given csv file. This allowed for a visual analysis of the signal and served as a reference to compare the filtered signal to.

```
fs = 1e6
Ts = 1/fs
t_end = 50e-3
t = np.arange (0,t_end-Ts,Ts)

# load input signal
df = pd.read_csv ('NoisySignal.csv')

t = df ['0'].values
sensor_sig = df ['1'].values
plt.figure(figsize = (10 , 7) )
plt.plot (t, sensor_sig )
plt.grid ()
plt.title ('Noisy Input Signal')
plt.xlabel ('Time [s]')
plt.ylabel ('Amplitude [V]')
plt.show ()
```

Listing 1: NoisySignal.csv plot

Next, the low (1 - 1800 Hz), middle (1800 - 2000 Hz), and high (2000 - 500000 Hz) frequencies were isolated for analysis using the Fast Fourier Transform function that was developed in earlier labs. Given code was used for the stem plots so the plots could be generated quicker.

```
def clean_fft(x,fs):

    N = len( x ) # find the length of the signal
    X_fft = scipy.fftpack.fft ( x ) # perform the fast Fourier
  transform (fft)
    X_fft_shifted = scipy.fftpack.fftshift ( X_fft )     # shift
  zero frequency components
                                                          # to
  the center of the spectrum
```

```

    freq = np.arange ( - N /2 , N /2) * fs / N          # compute the frequencies for the output
                                                          # signal ,
   (fs is the sampling frequency and
                                                          # needs to
   be defined previously in your code
    X_mag = np.abs( X_fft_shifted ) / N # compute the magnitudes of the signal
    X_phi = np.angle ( X_fft_shifted ) # compute the phases of the
   signal
    for i in range(len(X_mag)):
        if np.abs(X_mag[i]) < 1e-10:
            X_phi[i] = 0
    return (freq, X_mag, X_phi)


def make_stem ( ax ,x ,y , color ='k', style ='solid', label ='',
   linewidths =2.5 ,** kwargs) :
    ax.axhline ( x [0] , x [ -1] ,0 , color ='r')
    ax.vlines (x , 0 ,y , color = color , linestyles = style ,
   label = label , linewidths = linewidths )
    ax.set_ylim ([1.05* y . min () , 1.05* y . max () ])

# FFT plots
(freq, X_mag, X_phi) = clean_fft(sensor_sig,fs)

# All frequencies
fig, ax1 = plt.subplots( figsize =(10,7) )
make_stem(ax1,freq,X_mag)
plt.xlim(-10000, 500e3)
plt.grid(True)
plt.xlabel('w (Hz)')
plt.ylabel('H(jw) dB')
plt.title('Total_signal fft')
plt.show()

# Low frequencies
fig, ax1 = plt.subplots( figsize =(10,7) )
make_stem(ax1,freq,X_mag)
plt.grid(True)
plt.xlim(1e0, 1800)
plt.xlabel('w (Hz)')
plt.ylabel('H(jw) dB')
plt.title('Total_signal Low Frequencies')
plt.show()

# High frequencies
fig, ax1 = plt.subplots( figsize =(10,7) )
```

```
make_stem(ax1,freq,X_mag)
plt.grid(True)
plt.xlim(2000, 500e3)
plt.xlabel('w (Hz)')
plt.ylabel('H(jw) dB')
plt.title('Total_signal High Frequencies')
plt.show()


# Middle frequencies
fig, ax1 = plt.subplots( figsize =(10,7) )
make_stem(ax1,freq,X_mag)
plt.grid(True)
plt.xlim(1800, 2000)
plt.xlabel('w (Hz)')
plt.ylabel('H(jw) dB')
plt.title('Total_signal Middle Frequencies')
plt.show()
```

Listing 2: FFT and Plots of signal

Once the signals were plotted, the transfer function was developed. Further explanation is seen in the Equations section above. The Bode plots were then plotted for the low, middle and high frequencies to see the attenuation.

```
# Transfer Function for filter
steps = 1e0
w = np.arange(1e0, fs ,steps)
R = 4122
L = 200e-3
C =35.1e-9

num = [(1/(R*C)), 0]
den = [1, (1/(R*C)), (1/(L*C))]

mag_H = (w/(R*C)) / np.sqrt(( ((1/(L*C)) - (w**2) )**2) + (w/(R*C))
    **2)
db_mag_H = 20*np.log10(mag_H)

phase_H = ((.5*np.pi) - np.arctan( (w/(R*C)) / ((1/(L*C)) - (w**2))
    ))

for i in range(len(w)):
    if ( ( (1/(L*C)) - w[i]**2 ) < 0 ):
        phase_H[i] -= np.pi

# Bode Plots

# All frequencies
```

```
plt.figure(figsize=(10,7))
plt.title('Bode Plot at all Frequencies')
sys = con.TransferFunction(num, den)
_= con.bode(sys, w, Hz=True, dB=True, deg=True, Plot=True)
plt.show()

# Low Frequencies
w = np.arange(1e0, 1800+steps,steps)*2*np.pi
plt.figure(figsize=(10,7))
plt.title('Bode Plot at Low Frequencies')
sys = con.TransferFunction(num, den)
_= con.bode(sys, w, Hz=True, dB=True, deg=True, Plot=True)
plt.show()

# Middle Frequencies
w = np.arange(1800, 2000+steps,steps)*2*np.pi
plt.figure(figsize=(10,7))
plt.title('Bode Plot at Middle Frequencies')
sys = con.TransferFunction(num, den)
_= con.bode(sys, w, Hz=True, dB=True, deg=True, Plot=True)
plt.show()

# High Frequencies
w = np.arange(2000, fs+steps,steps)*2*np.pi
plt.figure(figsize=(10,7))
plt.title('Bode Plot at High Frequencies')
sys = con.TransferFunction(num, den)
_= con.bode(sys, w, Hz=True, dB=True, deg=True, Plot=True)
plt.show()
```

Listing 3: Transfer Function and Bode Plots

The filtered signal was plotted and a FFT was performed to showcase that the frequencies outside of the desired range were filtered out.

```
# Pass through Bandpass RLC circuit

# z-domain equivalent of x
numX, denX = sig.bilinear(num, den, fs)
#Pass through filter
y = sig.lfilter(numX, denX, sensor_sig)

myFigSize = (12,8)
plt.figure(figsize=myFigSize)
plt.plot(t, y)
plt.grid(True)
plt.title('Output Signal y(t) through RLC Filter')
plt.xlabel('t(s)')
plt.ylabel('y(t)')
```

```

# %%
# FFT of filtered signal
clean_sig = y
(freq, X_mag, X_phi) = clean_fft(clean_sig,fs)

# All frequencies
fig, ax1 = plt.subplots( figsize =(10,7) )
make_stem(ax1,freq,X_mag)
plt.xscale('log')
plt.xlim(1e0, fs)
plt.xlabel('w (Hz)')
plt.ylabel('H(jw) dB')
plt.title('Total_signal fft')
plt.show()
```

Listing 4: Filtered Signal and FFT

4 Results

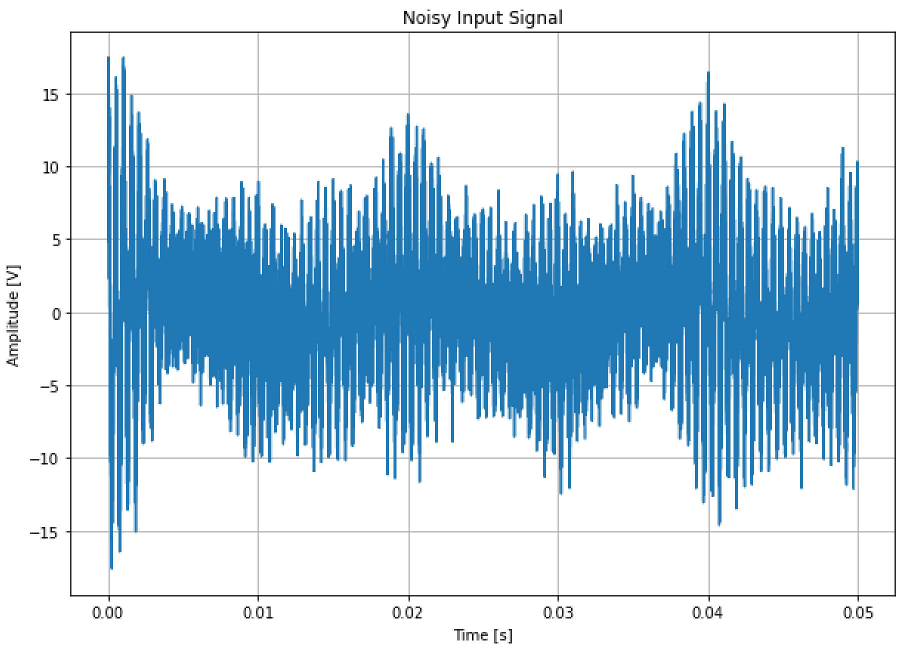


Figure 1: Initial Noisy Signal

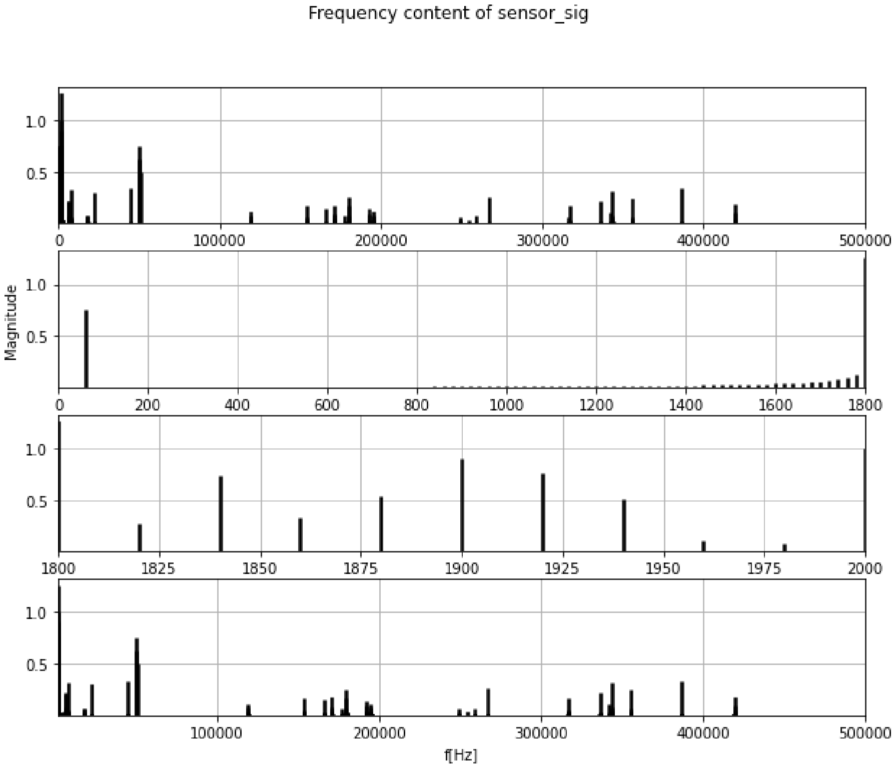


Figure 2: FFT of All Frequencies

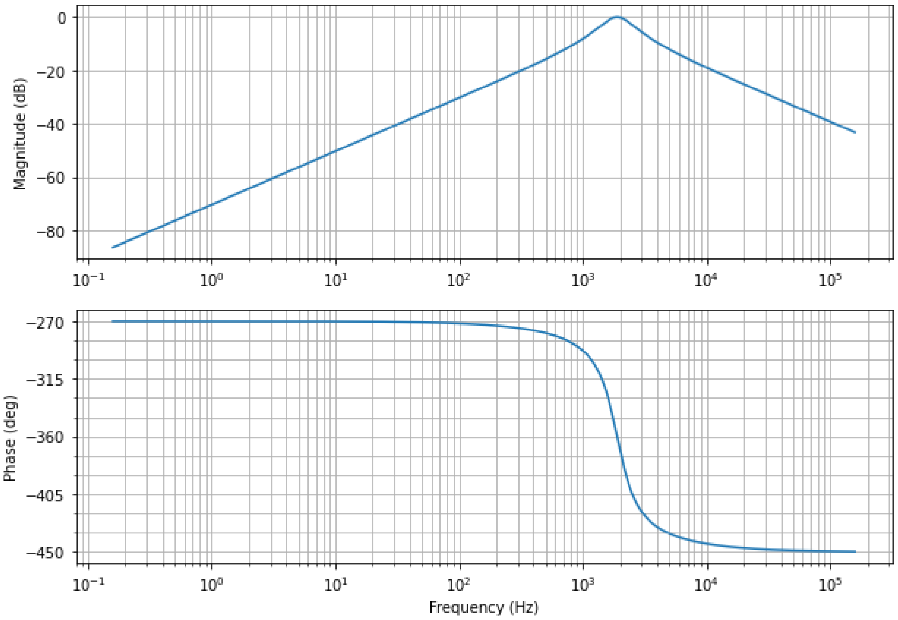


Figure 3: Bode Plot of Entire Signal

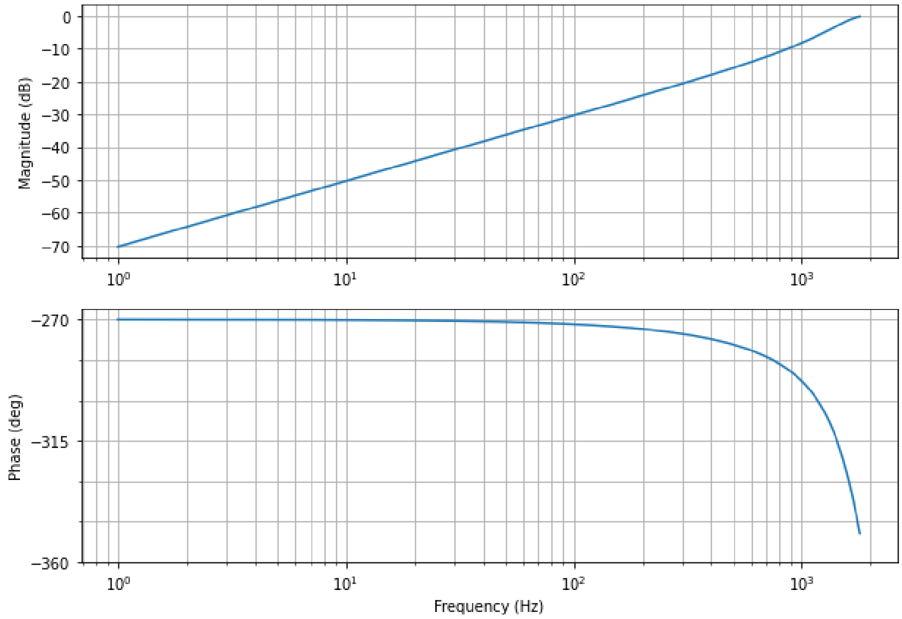


Figure 4: Bode Plot of Low Frequencies

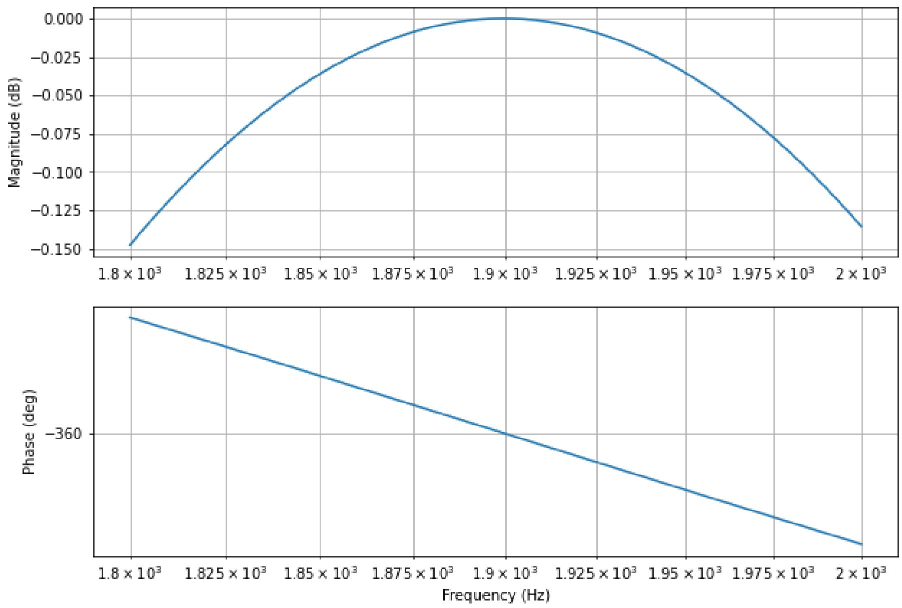


Figure 5: Bode Plot of Middle Frequencies

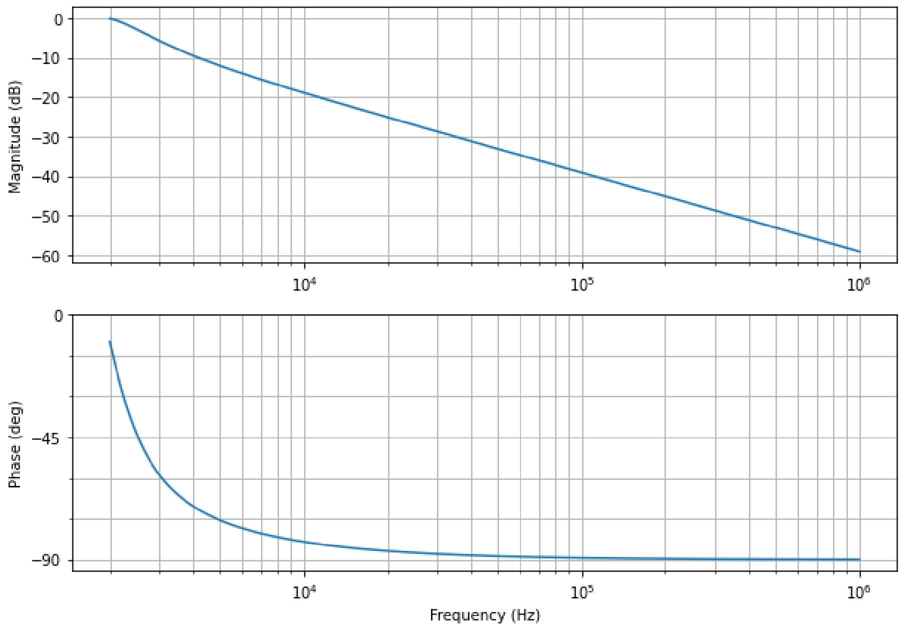


Figure 6: Bode Plot of High Frequencies

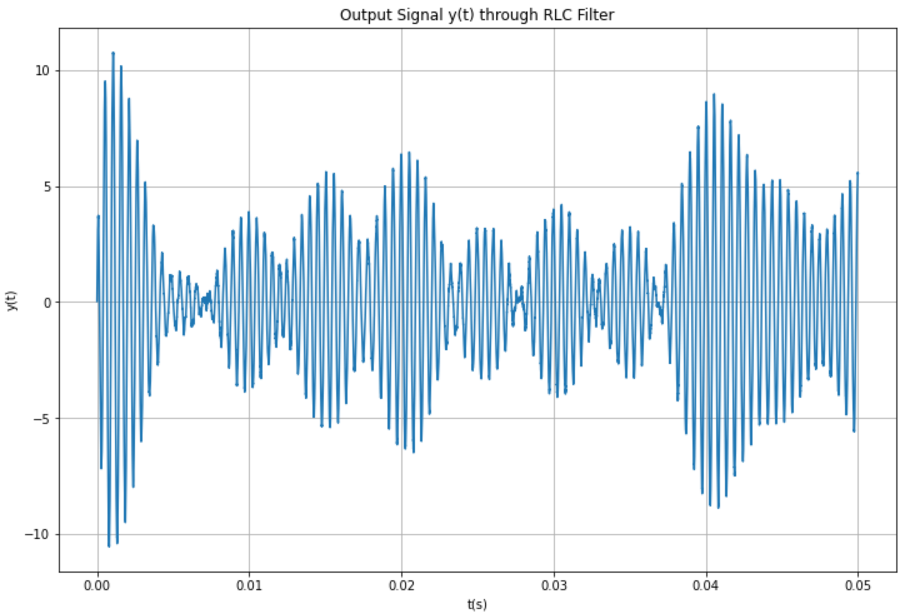


Figure 7: Plot of Filtered Output Signal

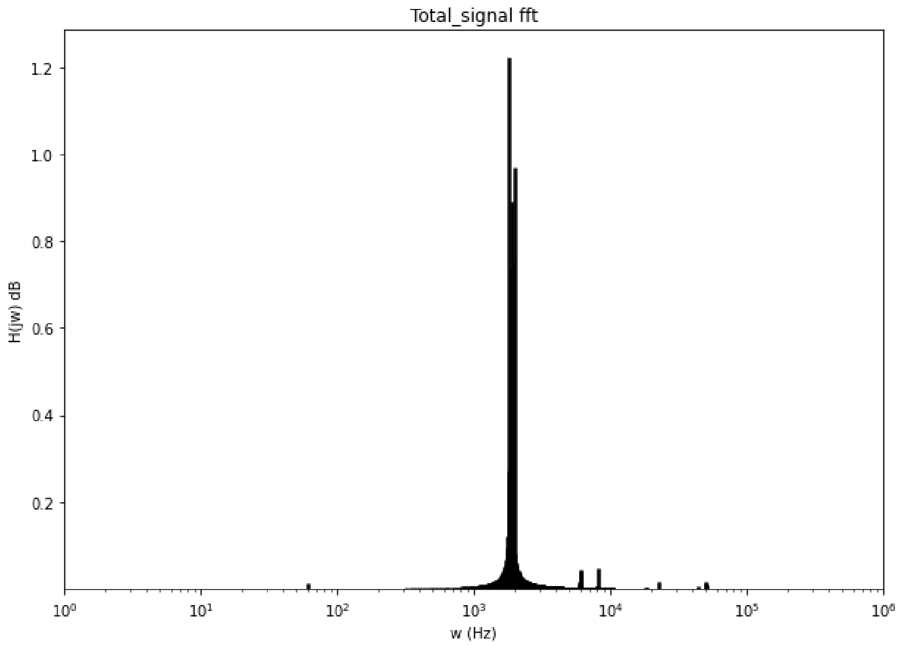


Figure 8: FFT of Filtered Signal

5 Questions

1. Earlier this semester, you were asked what you personally wanted to get out of taking this course. Do you feel like that personal goal was met? Why or why not?

I wanted to become more proficient in Python and learn more about applications of Signals and Systems. This course definitely helped with that since every assignment involved Python and learning more about various libraries and their applications. As for real world applications, the final assignment put that into perceptive when we were tasked with designing our own filter for a position sensor for an aircraft.

6 Conclusion

This final project delved deeper into the development process by having us analyze a signal and designing a filter based off of it. The entire Bode plot looks inelegant, but the indiviudal Bode plots show the right attenuation for the specified areas. The Python and LaTeX code are seen in `https://github.com/Eniac618/ECE351_Code` and `https://github.com/Eniac618/ECE351_Reports` respectively.