ECE 351 - Section #53

Lab #12 Report

Filter Design

Submitted To:
Kate Antonov
University of Idaho
kantonov@uidaho.edu

Submitted By :
Macallyster S. Edmondson
University of Idaho
edmo7033@vandals.uidaho.edu
github.com/mac-edmondson

26^{th} April, 2022

Contents

1 Introduction

The goal of this lab was to take many of the concepts taught throughout this course and apply them to a practical application. More specifically, we used concepts taught throughout this course to filter a positioning feedback signal for an aircraft. The data on the signal is contained in the frequency range of $1.8\,kHz \leq f \leq 2.0\,kHz$. Significant noise exists in the transmitted signal at higher frequencies due to a shared ground of the positioning system and a switching amplifier. Low frequency vibrations also exist on the signal due to a building ventilation system on the receiving end of the signal. A filter was designed to isolate the frequency range of the original signal.

This lab was completed using*Python* through the *Spyder-IDE*. The packages used in the completion of this lab were `numpy` for definitions of mathematical functions, `matplotlib.pyplot` to plot outputs of functions, `scipy.signal` & `control` to compute the frequency response of a digital filter, `scipy.fftpack` to perform Fast Fourier Transform operations, and `pandas` to import signal data from a CSV file.

All code for this lab, including this report, can be found on my Github.

2 Equations

The equations used within this lab are shown in this section. The equations will be referenced by number throughout the rest of the report.

Series RLC Bandpass Circuit Equations:

$$\omega_0 = \sqrt{\frac{1}{LC}} \tag{1}$$

$$\omega_{c1/c2} = \mp\frac{R}{2L} + \sqrt{(\frac{R}{2L})^2 + (\frac{1}{LC})} \tag{2}$$

3 Methodology

The first task accomplished in this lab was importing and viewing the signal given in a CSV. A Fast Fourier Transform was also performed to view the magnitude of noise components at different frequencies. The code to perform this task is given in the below listing.

```
# Get data from CSV into arrays
df = pd.read_csv('NoisySignal.csv')

t = df['0'].values
sensor_sig = df['1'] .values
```

```

step_size = t[1] - t[0]; # Get time step of input signal
print("Step Size is: " + str(step_size))
freq_sample = 1/step_size

# Perfom FFT (Imported from Lab9)
def myFFT(x, fs, clean=False):
    N = len(x)
    X_fft = spfft.fft(x)
    X_fft_shifted = spfft.fftshift(X_fft)

    freq = np.arange(-N/2, N/2)*fs/N
    X_mag = np.abs(X_fft_shifted)/N
    X_phi = np.angle(X_fft_shifted)
    if(clean):
        for i in range(0, len(X_phi)):
            if (X_mag[i] < (1e-10)) :
                X_phi[i] = 0

    return freq, X_mag, X_phi

freq, X_mag, X_phi = myFFT(sensor_sig, freq_sample, clean=False)

plt.figure(figsize = (15, 17))
plt.subplot(3, 1, 1)
plt.plot(t, sensor_sig, "b-")
plt.grid()
plt.ylabel('x(t)')
plt.xlabel('t')
plt.title('FFT of Input Signal, x(t)')
plt.subplot(3, 2, 3)
plt.stem(freq, X_mag, "b-")
plt.grid()
plt.ylabel('|X(f)|')
plt.subplot(3, 2, 4)
plt.stem(freq, X_mag, "b-")
plt.xlim(0, 100e3)

plt.grid()
plt.subplot(3, 2, 5)
plt.stem(freq, X_phi, "b-")
plt.grid()
plt.ylabel('/_X(f)')
plt.xlabel('f [Hz]')
plt.subplot(3, 2, 6)
plt.grid()
plt.stem(freq, X_phi, "b-")
plt.xlim(0, 100e3)

plt.figure(figsize = (15,17))
```

```
plt.subplot(1, 1, 1)
plt.stem(freq, X_mag, "b--")
plt.xticks(np.arange(0, 3e3 + 100, 100))
plt.xlim(0, 3e3)
plt.title("Unfiltered Signal Magnitudes [0 to 3k Hz]")
plt.grid()
plt.ylabel('Mag.')
plt.xlabel('f [Hz]')
```

This code yielded the graphs shown in Figures 1 & 2.

Next, I began designing a filter to isolate the wanted frequency band of $1.8\,kHz \leq f \leq 2\,kHz$. As there is noise on both sides of the needed frequency band, this implies a bandpass filter must be designed. Scratch work for this filter design can be seen in the Attachments section of this report. I started with the center and corner frequency equations (seen in Equations (1) & (2)), and chose a center frequency of $1.9\,kHz$. I then solved for L in terms of C based on Equation (1) and selected a value for L and C that fit the equation and seemed realistic. I then computationally solved for the resistor value for both corner frequencies (Equation (2)), averaged them, and used that as my initial resistor value.

Once some values were decided upon, I created a Bode plot to verify my results. I found that I could greatly reduce my resistor value and eventually came up with the filter seen in Figure 3. Multiple Bode plots of the filter's frequency response are seen in Figures 4, 6, & 5. Each bode plot shows different frequency ranges of the filter and that the filter meets the specifications given for the lab. Code for generating the plots is seen in the following listing.

```
C = 3.5 * 1e-6
L = 2.01 * 1e-3
R = 9.4

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

sys = con.TransferFunction(num, den)
syssig = spsig.lti(num, den)

step_size = 1e2
w = np.arange(1, (1e6 * 2*np.pi)+step_size, step_size)
bodeW, bodeMag, bodePhase = spsig.bode(syssig, w)

plt.figure(figsize=(11, 15))
plt.subplot(2, 1, 1)
plt.semilogx(bodeW/(2*np.pi), bodeMag)
plt.grid(True, which='both', ls='-')
plt.ylabel('Magnitude (dB)')
plt.yticks(np.arange(-90, 10, 10))
```

```
plt.title('Magnitude and Phase Bode Plot [spsig.bode]')
plt.xlabel('freq (rad/s)')
plt.show()

plt.figure(figsize=(10, 11))
plt.ylim(0, 10)
_ = con.bode(sys, omega=None, dB = True, Hz = True, deg = True, Plot =
    True)
plt.xlim(1.8e3, 2e3)
plt.subplot(2, 1, 1)
plt.ylim(-.5, 0)
plt.title('Magnitude and Phase Bode Plot [con.bode] (Hz)')
plt.show()

plt.figure(figsize=(10, 11))
_ = con.bode(sys, omega=None, dB = True, Hz = True, deg = True, Plot =
    True)
plt.xlim(0, 1e6)
plt.subplot(2, 1, 1)
plt.title('Magnitude and Phase Bode Plot [con.bode] (Hz)')
plt.show()
```

Lastly, I filtered the Noisy input signal using the filter designed. A FFT was also performed to verify the filter was properly attenuating all unwanted noise to an appropriate magnitude. The code implementing this filtering process is shown in the below listing. The output of this code can be seen in Figures 7 & 8.

```
step_size = t[1] - t[0];
fs = 1/step_size

z, p= spsig.bilinear(num, den, fs=fs)
filtered_sig = spsig.lfilter(z, p, sensor_sig)

freq, X_mag, X_phi = myFFT(filtered_sig, freq_sample, clean=False)

plt.figure(figsize = (15, 17))
plt.subplot(3, 1, 1)
plt.plot(t, filtered_sig, "b-")
plt.grid()
plt.ylabel('x(t)')
plt.xlabel('t')
plt.title('FFT of Input Signal, x(t)')
plt.subplot(3, 2, 3)
plt.stem(freq, X_mag, "b-")
plt.grid()
plt.ylabel('|X(f)|')
plt.subplot(3, 2, 4)
plt.stem(freq, X_mag, "b-")
plt.xlim(0, 100e3)

```

```
plt.grid()
plt.subplot(3, 2, 5)
plt.stem(freq, X_phi, "b-")
plt.grid()
plt.ylabel('/_X(f)')
plt.xlabel('f [Hz]')
plt.subplot(3, 2, 6)
plt.stem(freq, X_phi, "b-")
plt.xlim(0, 100e3)

plt.grid()
plt.xlabel('f [Hz]')

plt.figure(figsize = (15,17))
plt.subplot(1, 1, 1)
plt.stem(freq, X_mag, "b-")
# plt.grid(True, which='both', ls='-')
plt.xticks(np.arange(0, 3e3 + 100, 100))
plt.xlim(0, 3e3)
plt.title("Filtered Signal Magnitudes [0 to 3k Hz]")
plt.grid()
plt.ylabel('Mag.')
plt.xlabel('f [Hz]')
```

4 Results

The results of this lab are very straightforward. The implementation of all functions worked as expected and the results are as expected. More analysis of theory and results is discussed in the Methodology section of this report.

The deliverables of this lab are seen in all figures given below.

Looking at the unfiltered data (Figures 1 & 2), it can be seen there is significant noise below 100 Hz and above 3000 Hz. When comparing that to the data after it was run through the filter (Figures 7 & 8), all noise was attenuated to the point it is almost unnoticeable in the magnitude plots. Additionally, it can be seen that the magnitudes of the frequencies in the range 1.8 $kHz \leq f \leq 2.0\,kHz$ are attenuated by no less than -0.3 dB (Figure 6). All of this makes a very nice nice looking output signal seen in Figure 7 which could then be processed further to extract data.

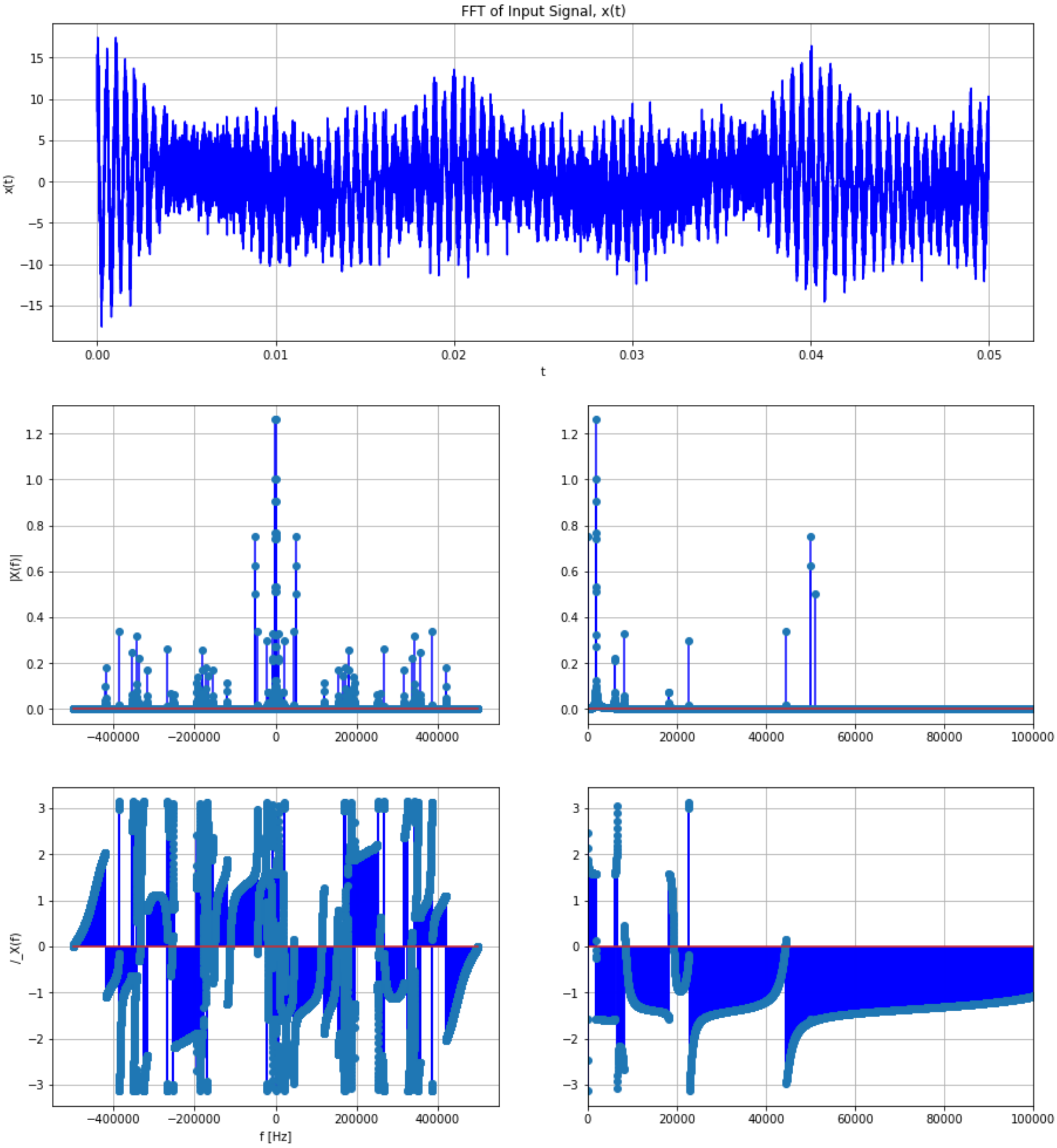

Figure 1: Noisy Input Signal + FFT

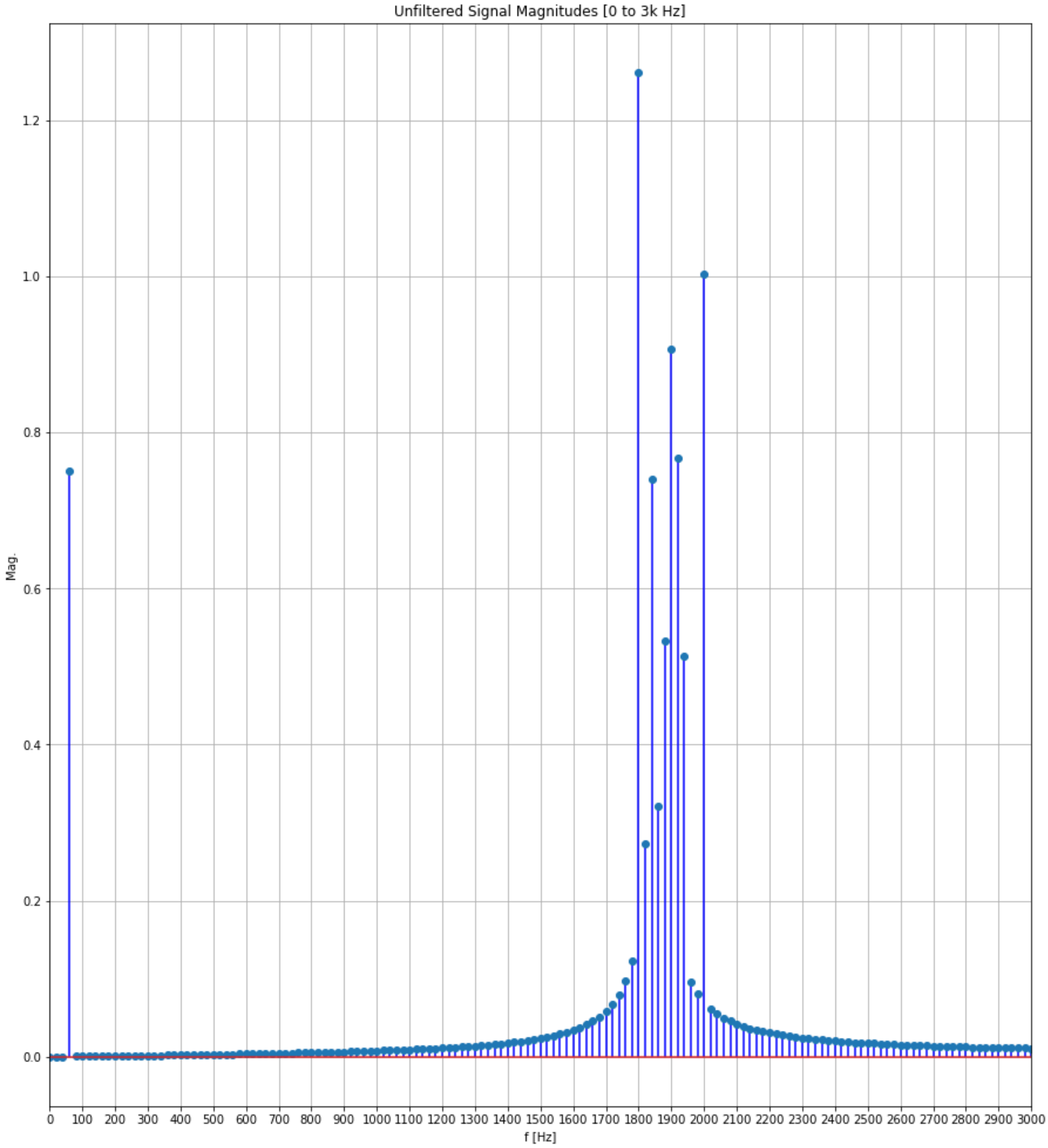

Figure 2: Noisy Input Signal Magnitudes (Reduced Freq. Range)

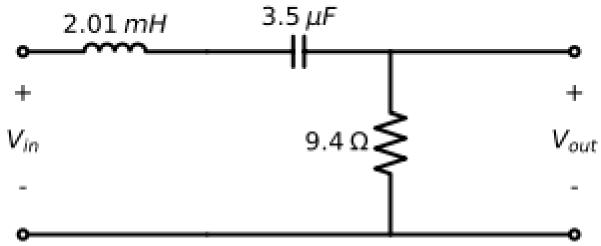

Figure 3: Filter Design

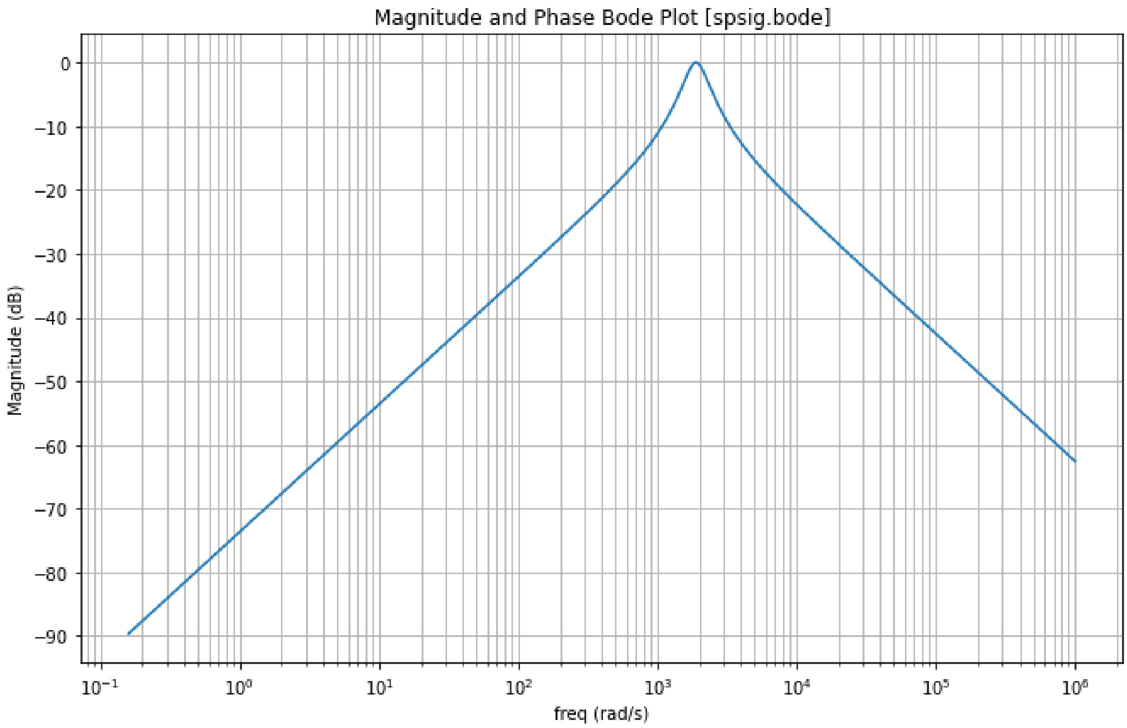

Figure 4: Filter Full Range Magnitude Bode Plot

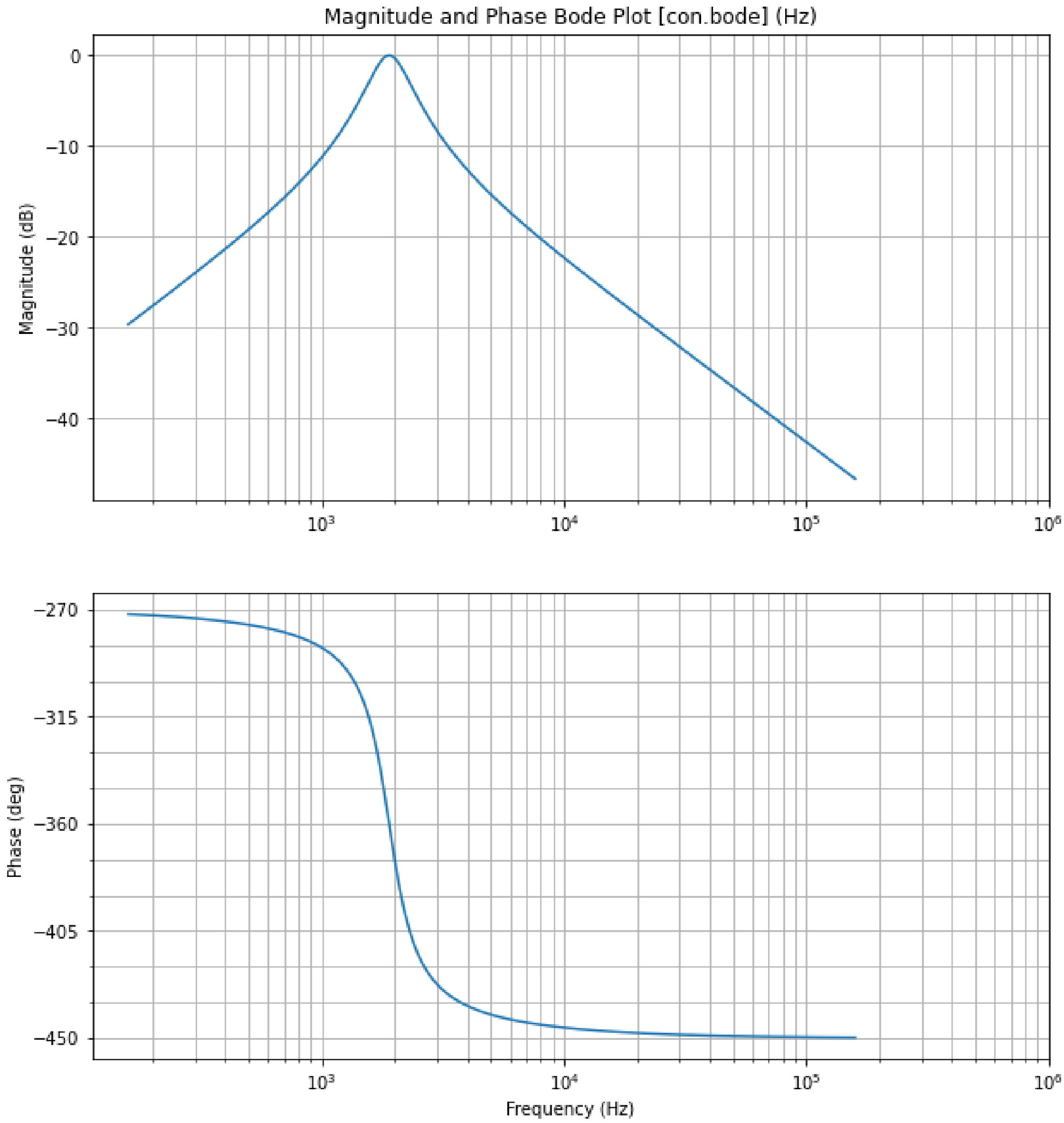

Figure 5: Filter Partial Range Magnitude + Phase Bode Plot

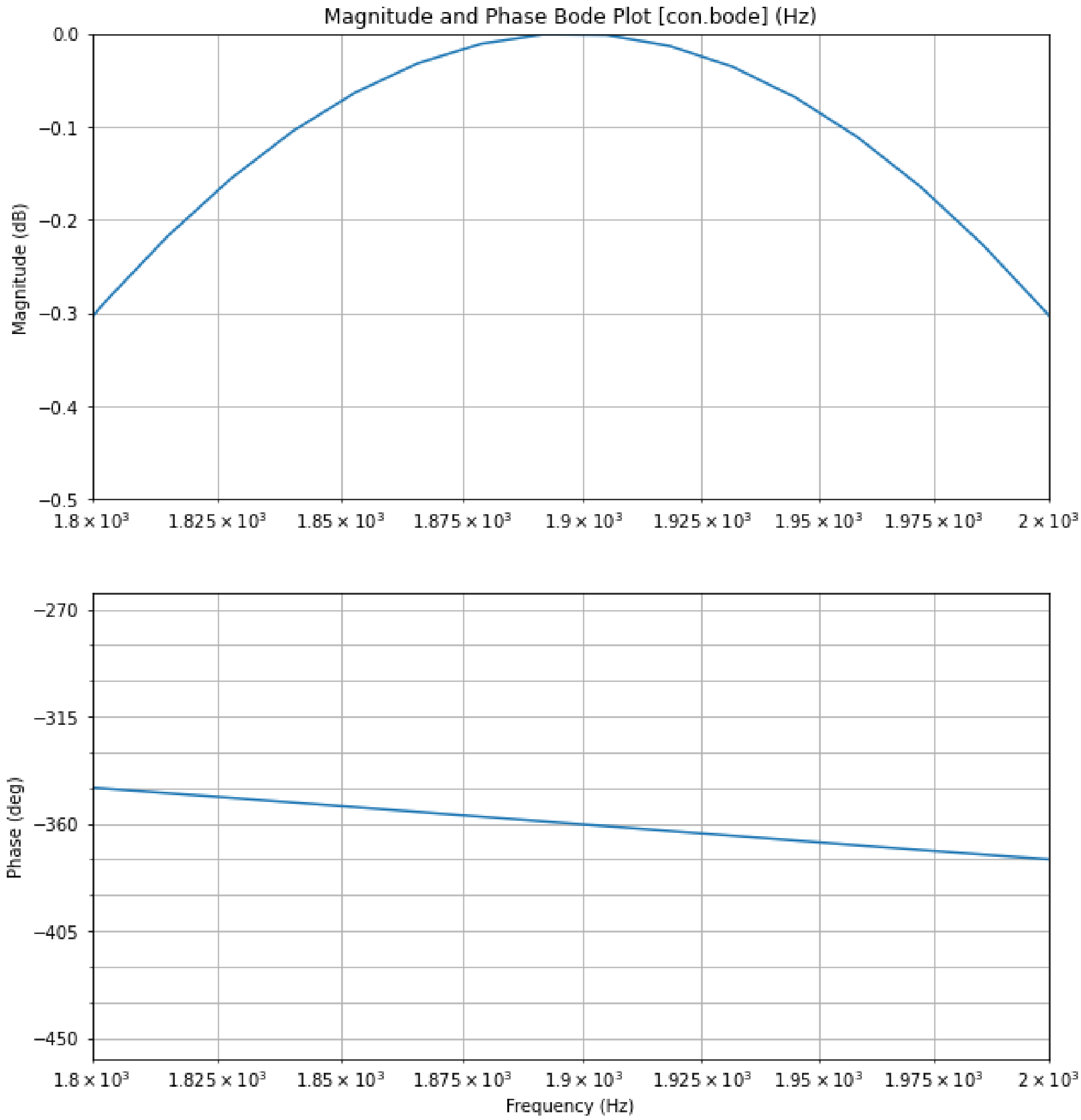

Figure 6: Filter Input Range Magnitude + Phase Bode Plot

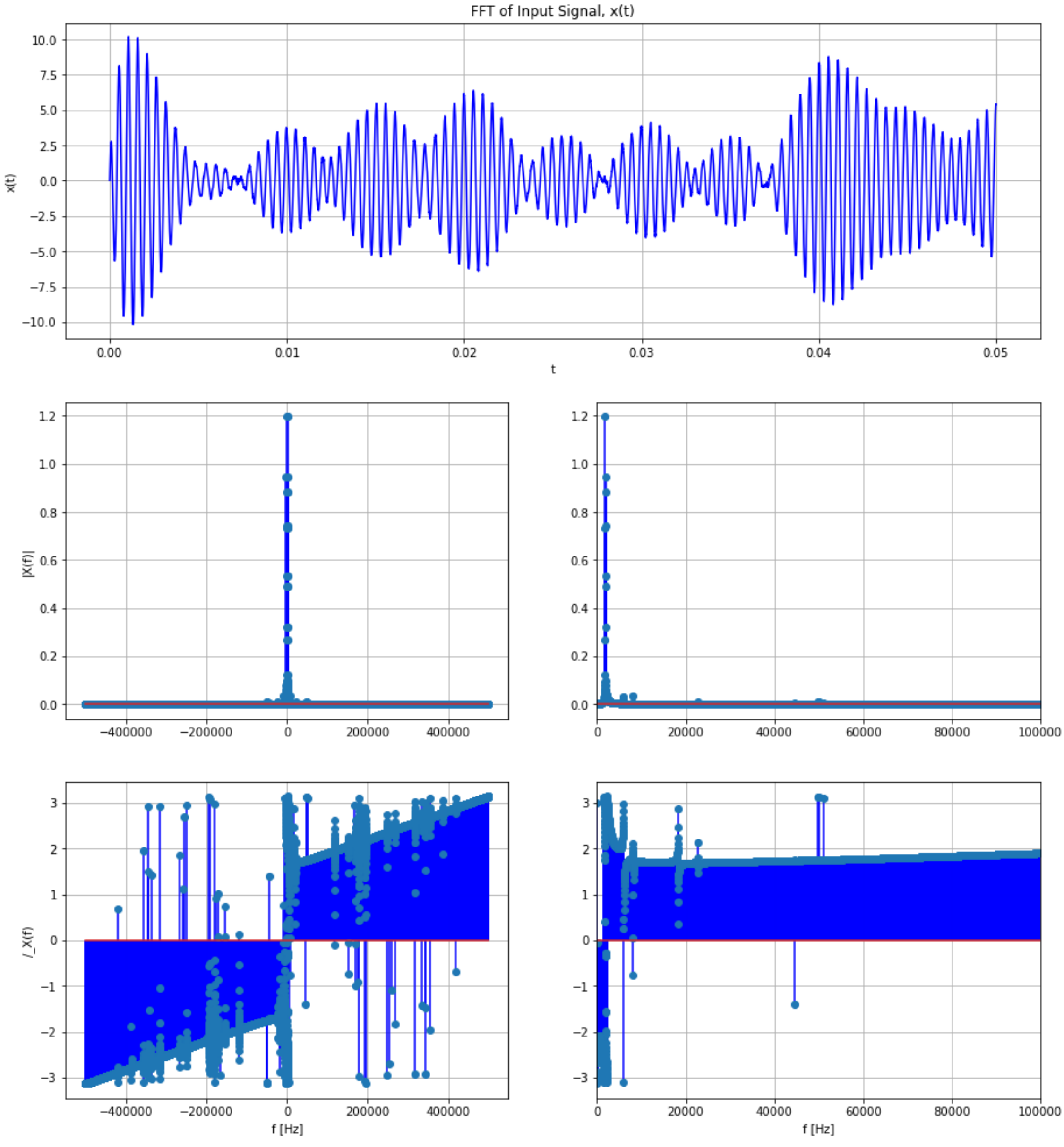

Figure 7: Filtered Input Signal + FFT

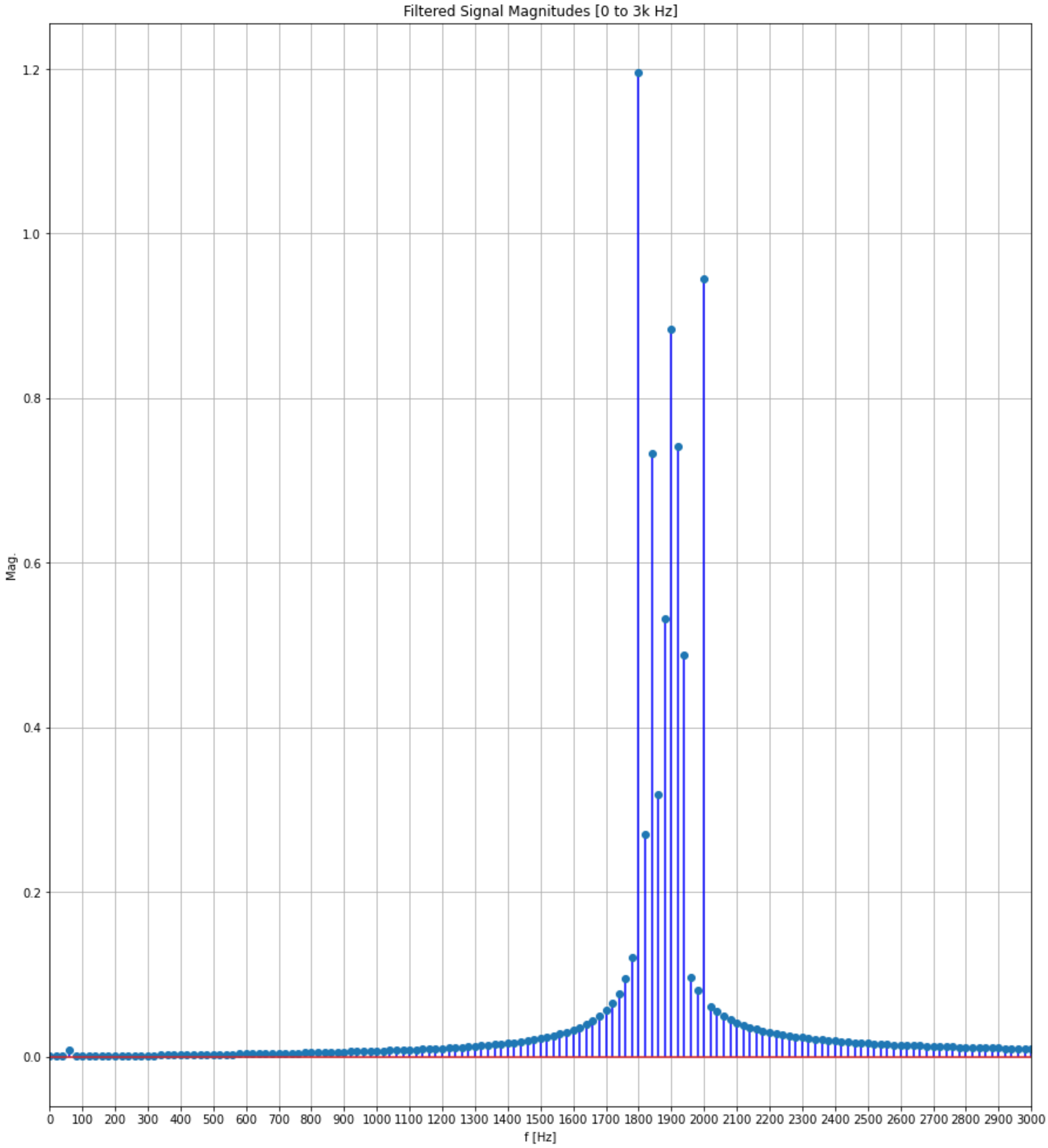

Figure 8: Filtered Input Signal Magnitudes (Reduced Freq. Range)

5 Error Analysis

No sources of error were seen throughout this lab.

6 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 feel strongly that the personal goal I set at the beginning of this semester was met. I have built my skills in using Python to perform post-processing on signals and general mathematics, etc. I have used Python as an aid to verify results of homework questions throughout this course which has been very useful and I think it shows how my goal has been accomplished.

7 Conclusion

In conclusion, I feel this lab was very successful. I feel this lab was a great conglomeration of many concepts taught throughout the course and it was fun to solve a problem on my own with little hand holding from the lab handout. All in all, I am very satisfied with what this lab has taught me and feel it was an excellent use of time.

8 Attachments

1. Filter design scratch-work

Mag @ w_c is $\approx -3db = .707$

as for range 1.8 KHz to 2 KHz attenuation must be no more than -0.3 dB. Thus, cut-off frequencies must be selected to fit this range.

Choose cut-off of 1.2 KHz & 2.6 KHz.

$w_{c1} = 1.2$ KHz ; $w_{c2} = 2.6$ KHz

$w_0 = 1.9 \cdot 2\pi \cdot 10^3 \frac{rad}{s}$

$$w_0 = \sqrt{\frac{1}{LC}} \Rightarrow LC = \frac{1}{w_0^2} = 7.017 \text{ nHF}$$

$$LC = 277 \cdot 10^{-9} \Rightarrow L = \frac{7.017 \cdot 10^{-9}}{C}$$ } Plot and select good values for both

select $C = 3.5$ µF $\Rightarrow L = 2.01$ mH

$$w_c = \pm\frac{R}{2L} + \sqrt{\left(\frac{R}{2L}\right)^2 + \left(\frac{1}{LC}\right)}$$

$\Rightarrow R \approx 22.74\ \Omega$ for w_{c1}
$R \approx 15.34\ \Omega$ for w_{c2} } Best R found was 9.4

Thus R will likely be somewhere in that range.

$2.77 \cdot 10^{-7} = 277 \cdot 10^{-9}$

$\downarrow R \Rightarrow$ implies tighter corner frequencies