Project Report for ECE 351

Lab 12: Final Project

Skyler Corrigan

December 8, 2022

ECE351 Code Repository:
$https://github.com/ElfinPeach/ECE351\_Code.git$

ECE351 Report Repository:
$https://github.com/ElfinPeach/ECE351\_Report.git$

Contents

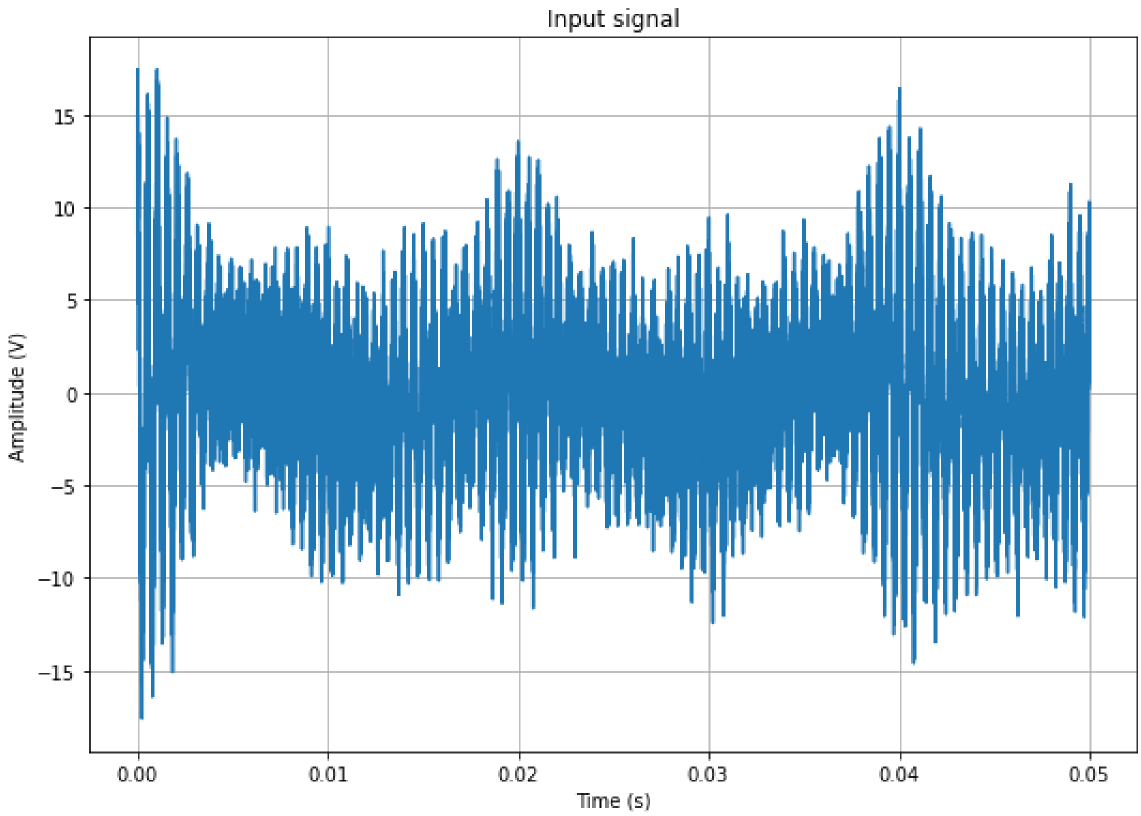
1 Objective

For this Project, a signal was given that had a lot of noise in it. The objective of this Project is to take that signal and filter out the unwanted frequencies and identify the magnitudes of the desired frequencies. For this lab, the desired frequencies were between 1.8 and 2 kHz.

2 Unfiltered Signal

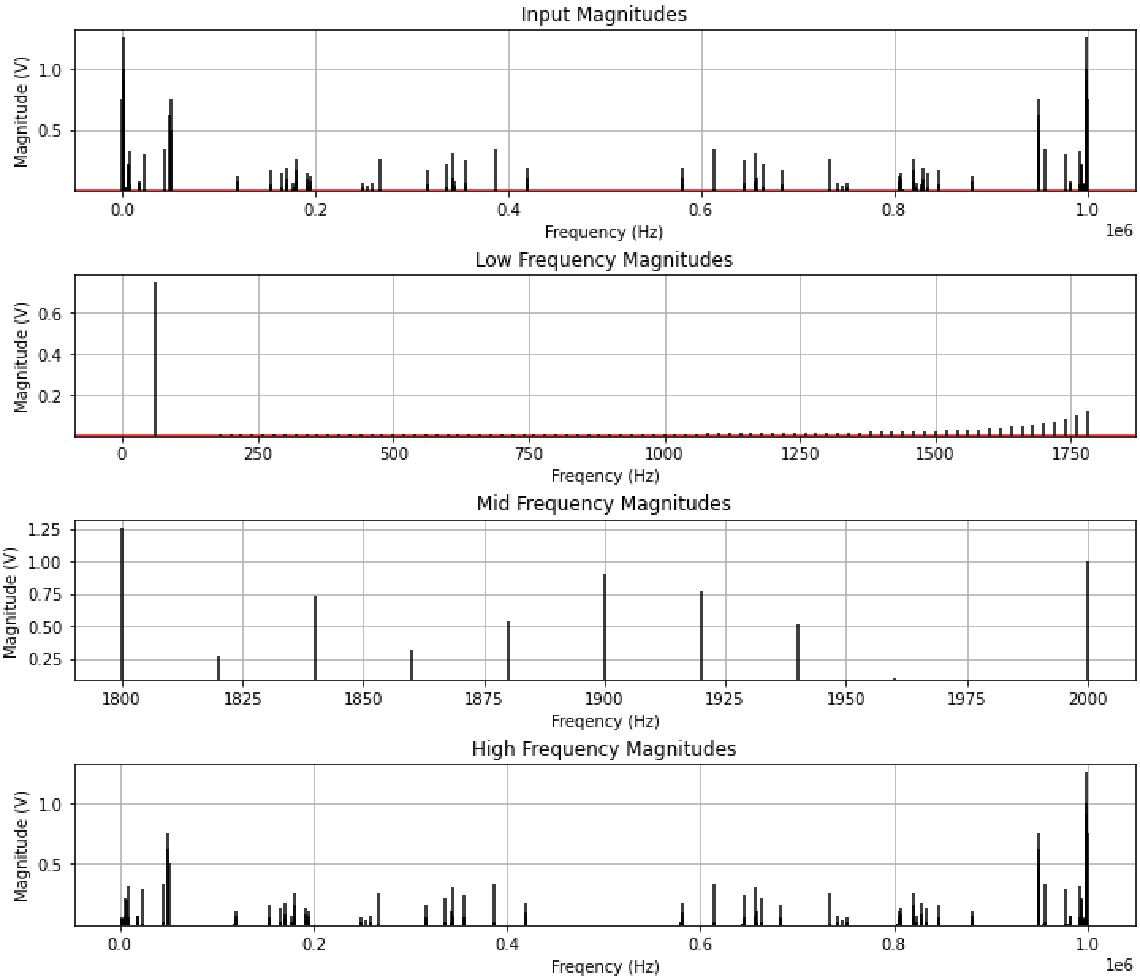
The image below shows the unfiltered Signal.

Figure 1: Unfiltered Signal



After passing the signal through an FFT, the frequencies and magnitudes can be seen in the figure below. There are four graphs on it: All frequencies, frequencies under 1.8 kHz, frequencies between 1.8 and 2 kHz, and frequencies above 2 kHz.

Figure 2: Unfiltered Frequencies



3 Analog Filter Design

To start the design, an RLC circuit was chosen. The reason for this is if the output is measured across the resistor, the resulting filter would be a Bandpass filter, which is the type of filter that is needed for this Project.

The bandwidth needed for this project is $B = 800Hz$ with a center frequency of $\omega_c = 1.9kHz$. By choosing a resistance of $R = 10\Omega$, the rest of the parameters can be found with the following equations:

$$L = \frac{R}{B}$$

$$C = \frac{B}{\omega_c^2 * R}$$

Using a TI Inspire Student Software, the results for C and L can be seen in the screen capture below (note: C was recalculated separately in the same software to get the more accurate result of $C = 3.527\mu F$).

Figure 3: Parameters Calculations

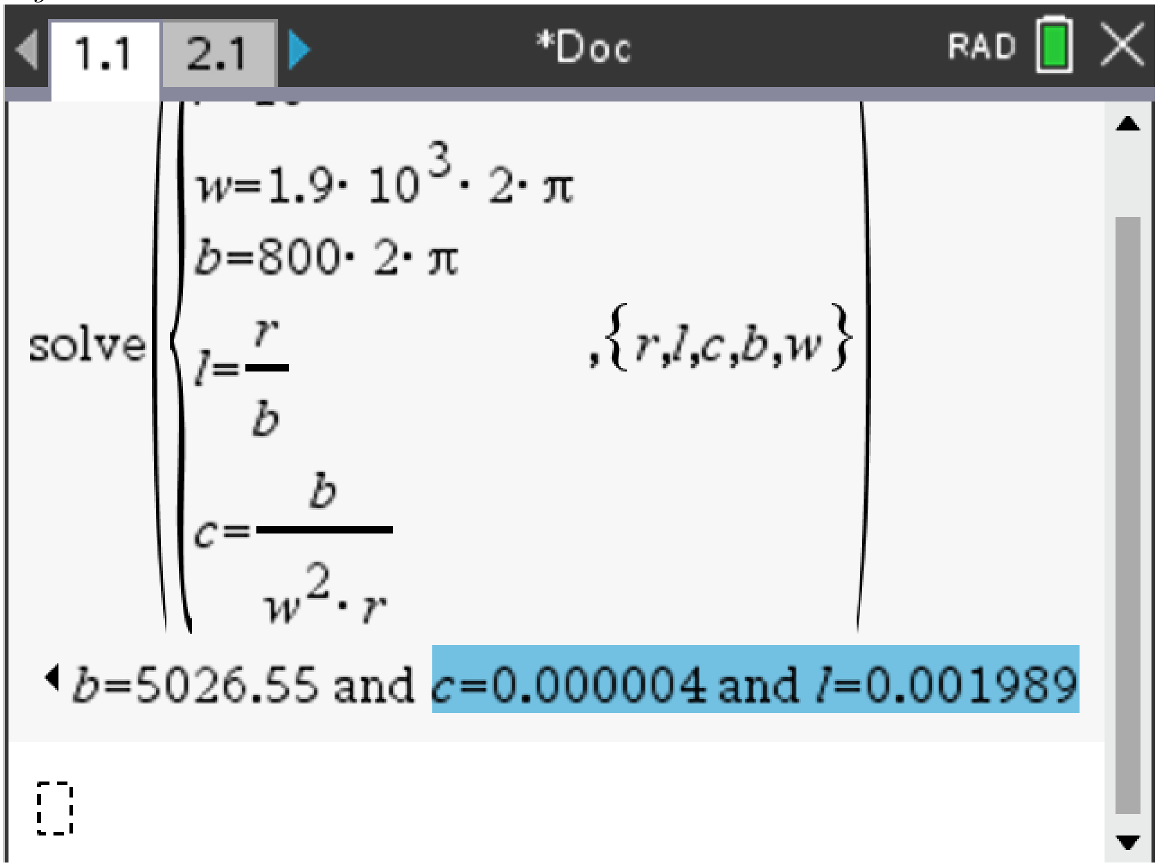
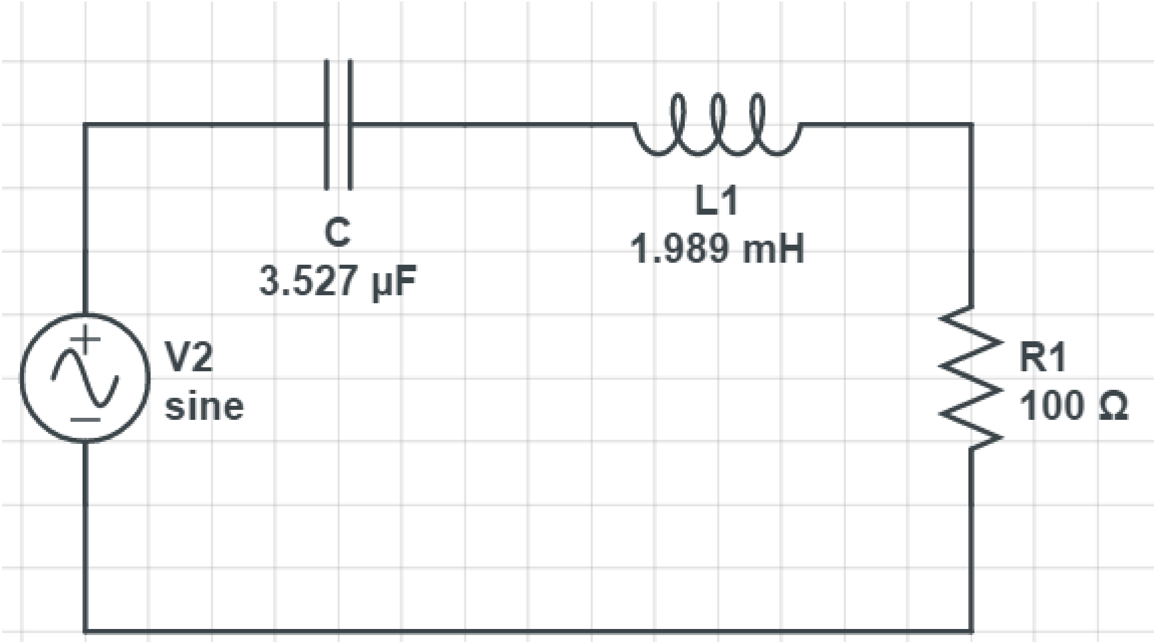


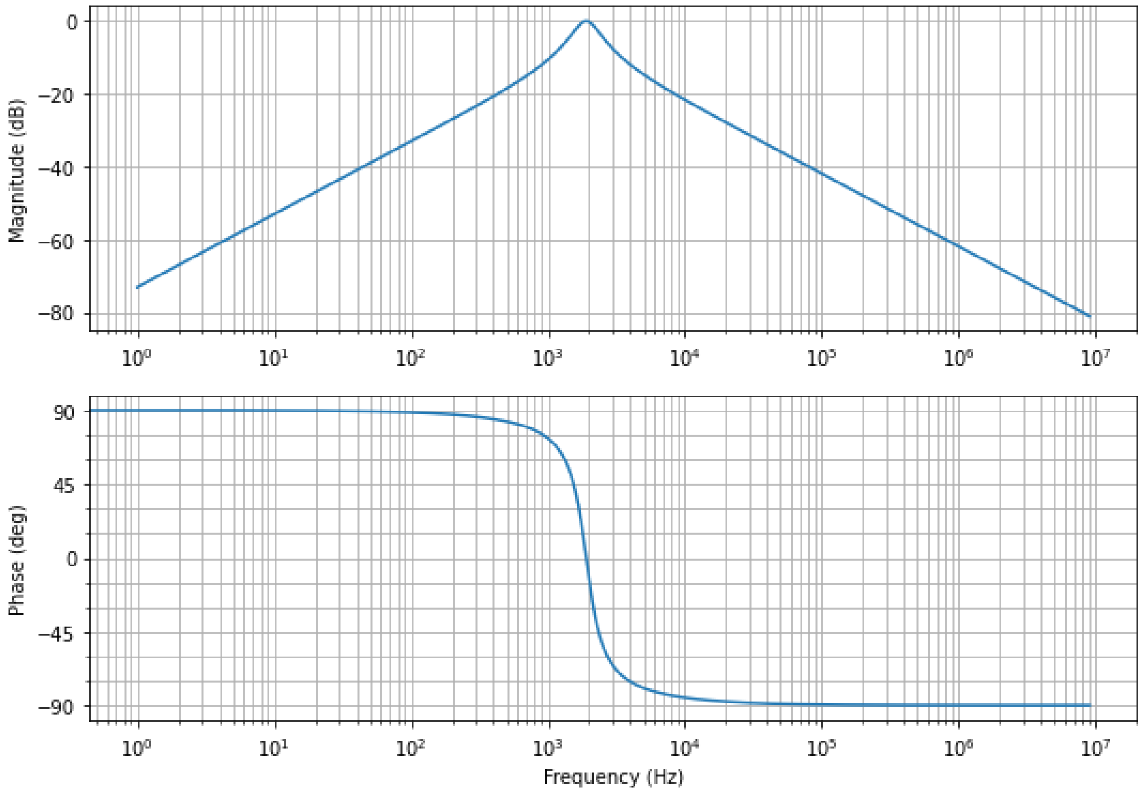
Figure 4: Circuit



4 Filter Verification

To validate the filter the following Bode plot was constructed:

Figure 5: All Frequencies Bode Plot



Looking at both the low (¡1.8 kHz) and high (¿2 kHz) frequency sides, we can see that the low frequency side is attenuated by at least -30.

Figure 6: Low Frequency Bode Plot

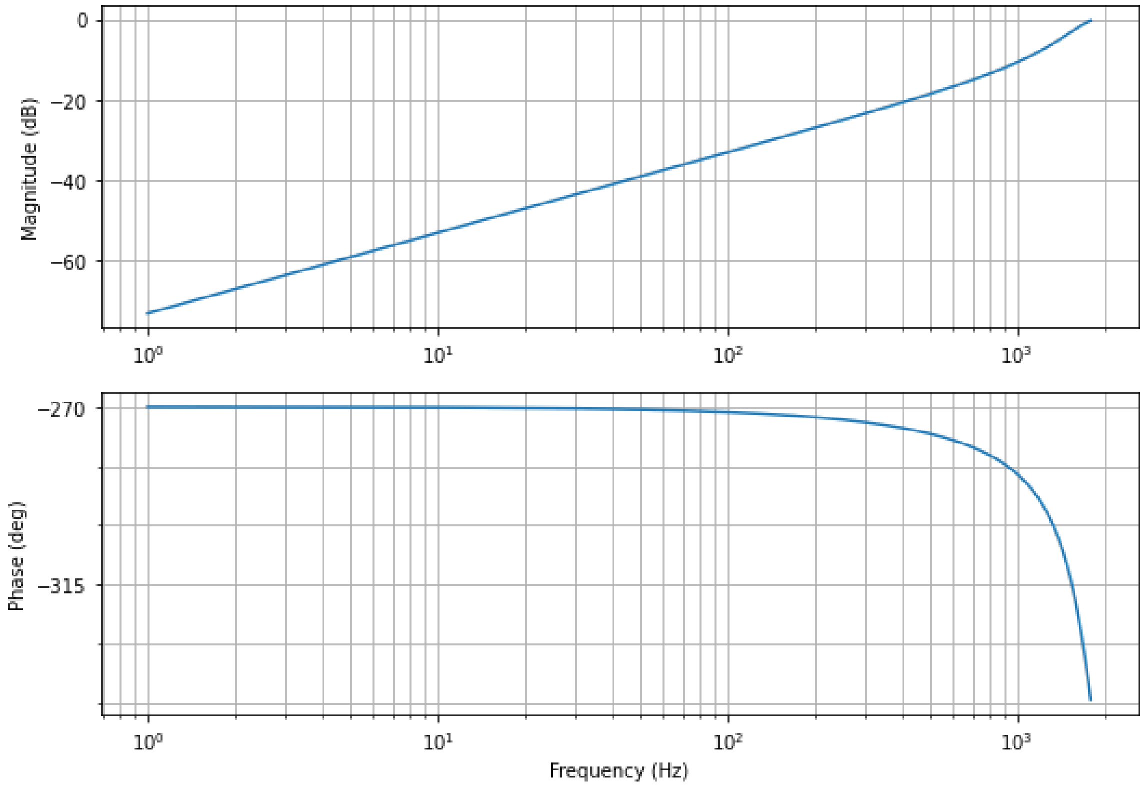
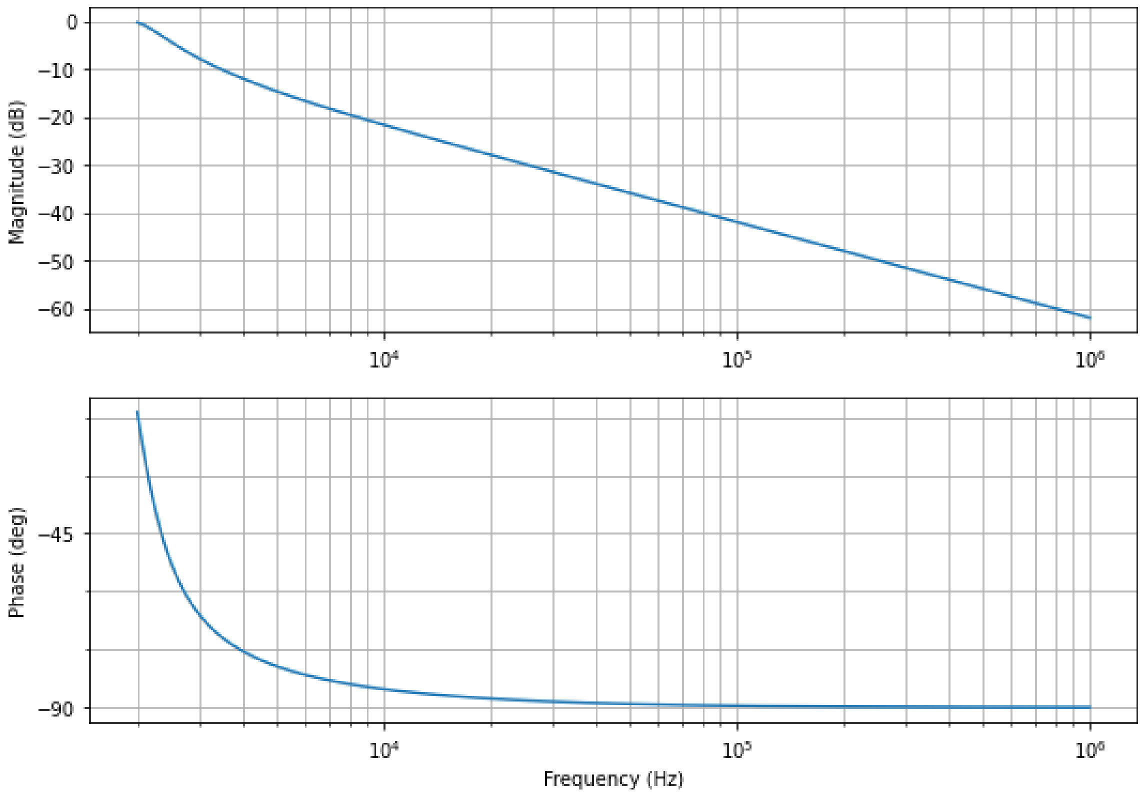
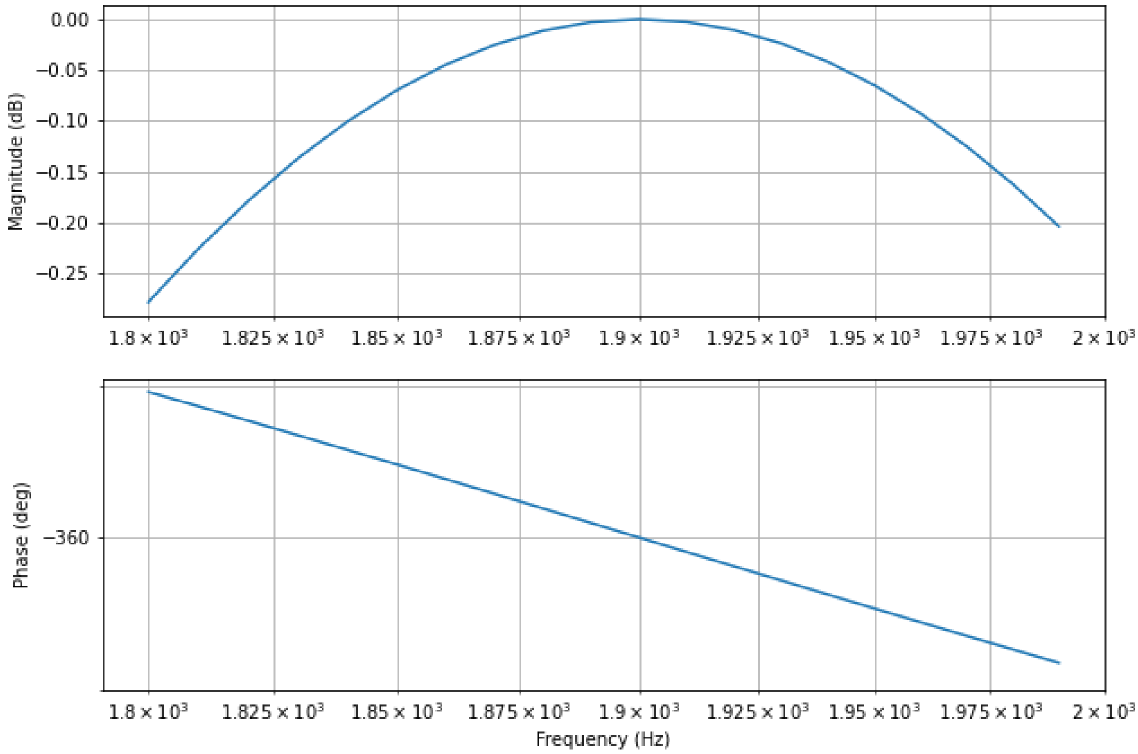


Figure 7: High Frequency Bode Plot



Observing the desired frequencies, we can see that it is attenuated by less than -0.3 dB.

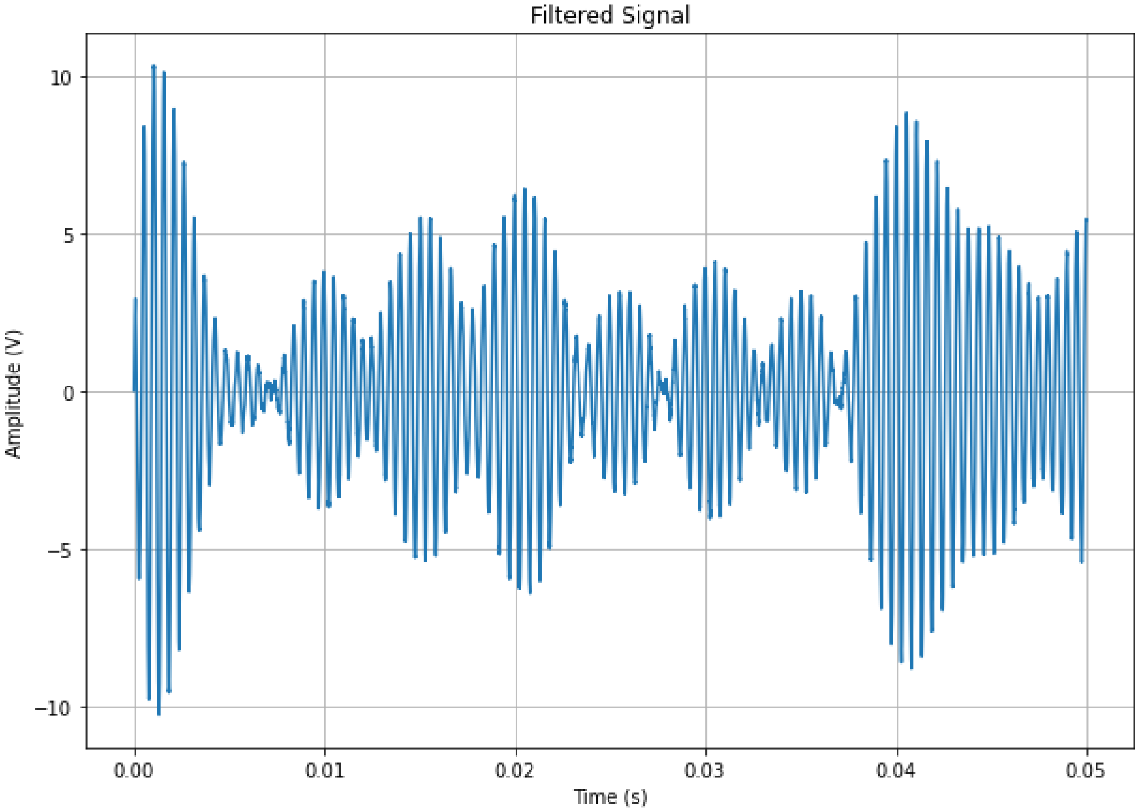
Figure 8: Desired Frequency Bode Plot



5 Filtering the Signal

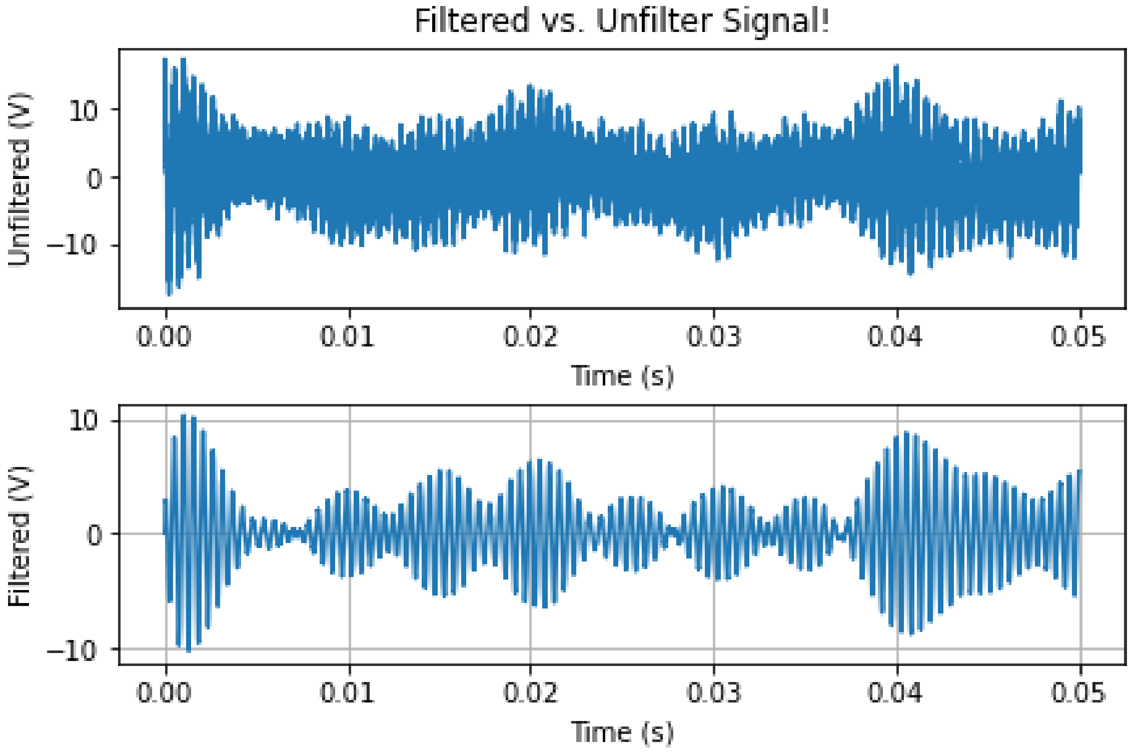
After passing the signal through the filter, this is the resulting filtered signal:

Figure 9: Filtered Signal



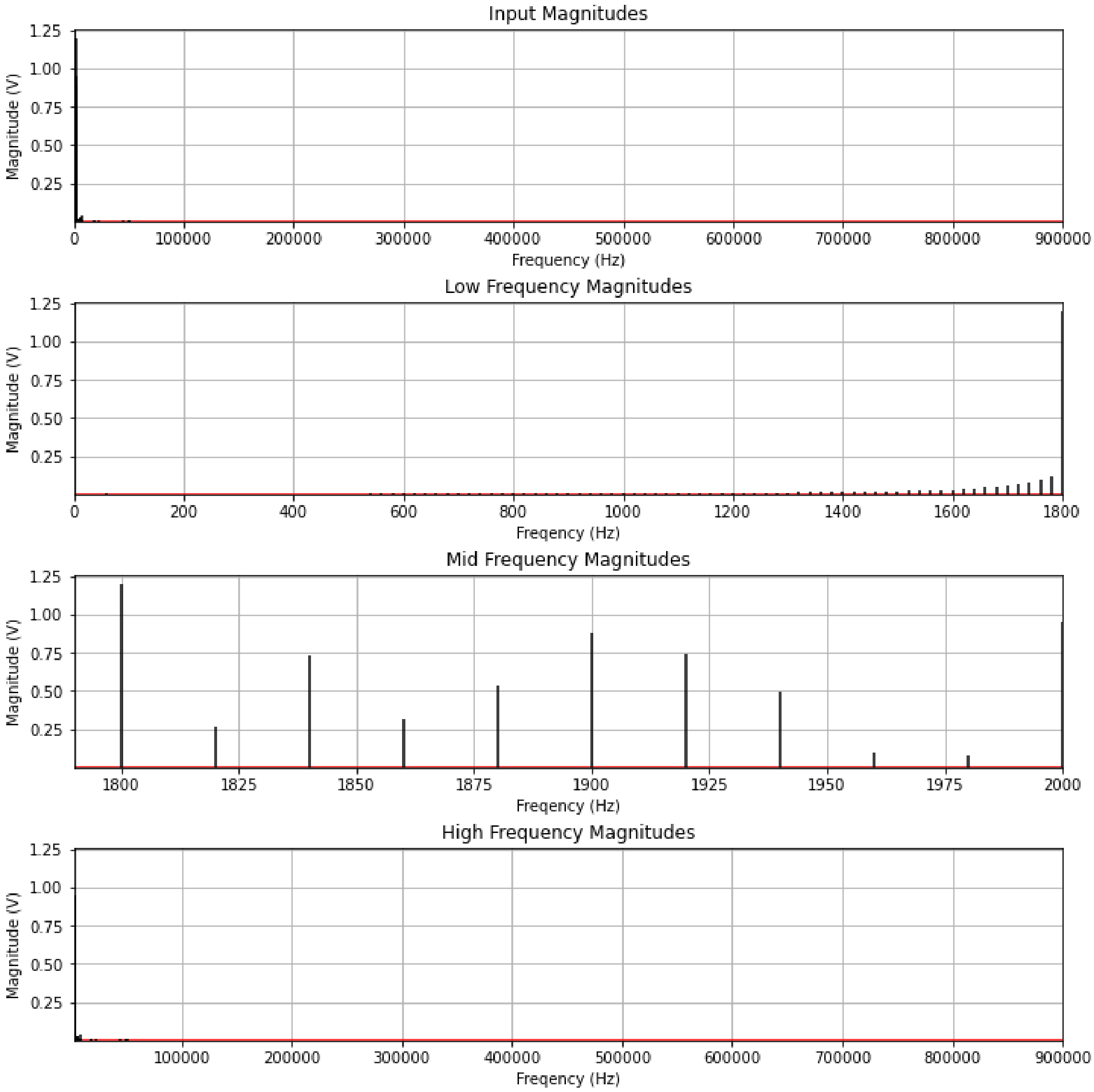
This can be then compared to the original signal, showing that the filter worked correctly.

Figure 10: Unfiltered vs. Filtered Signal



After passing the filtered signal through a FFT, the following image shows the frequencies and magnitudes. The breakdown is the same as *Figure 2*, and clearly shows that the low and high frequencies are all but eliminated. Furthermore, anything over 100 kHz has completely disappeared.

Figure 11: Filtered Frequencies



6 Questions

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?

Excuse me for a second whilst I look up what I wrote :)
What I wrote lol: "A passing grade :)". I think I'm a bit of a smartalic.
I got more out of this course than I intended. While I still don't like coding all too much, I can see the usefulness of some of it.

7 Appendix A: Code

The following is the code I used for this project.

Lab12 Code

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
import scipy.fftpack as fft
import pandas as pd

# The following package is not included with the Anaconda distribution
# and needed to be installed separately. The control package also has issue
# working on macs and a PC or a linux distribution is needed
import control as con

#---------------------------Function Stuff---------------------------

def make_stem(ax ,x,y,color='k',style='solid',label='',linewidths =1.5 ,**
    ax.axhline(x[0],x[-1],0, color='r')
    ax.vlines(x, 0 ,y, color=color , linestyles=style , label=label , linew
    ax.set_ylim ([1.05 * min(y), 1.05 * max(y)])
    return ax

# FFT function, ripped (and modified) from lab9
def FFT(X, fs):

    # Length of input array
    n = len(X)

    # Preform fast fourier transorm
    X_fft = fft.fft(X)

    """not used
    X_fft_shift = fft.fftshift(X_fft)
    """
    # Calculate frequnecies for output. fs is sampling frequency
    freqX = np.arange(0, n) * fs / n

    # Calculate magnatude and phase
    magX = np.abs(X_fft)/n
    angX = np.angle(X_fft)

    # Clean up the phase array a bit
```

```
    for i in range(len(angX)):
        if ( magX[i] < 1e-10):
            angX[i] = 0

    # return values
    return freqX, magX, angX

#——————————Time to start. This is taking too long...————————————————————

    # first off... import the thing
fp = pd.read_csv('NoisySignal.csv')

# define variables
t = np.array(fp['0'])
signal = np.array(fp['1'])

    # Completely unfiltered input signal! it looks great.

plt.figure(figsize = (10, 7))
plt.plot(t, signal)
plt.grid()
plt.title("Input signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude (V)")
plt.show()

    # Sampling frequency
s = 1e6

    # initiate arrays for spliting up frequency stuff
# freq < 1.8e3
lowfreq = []
lowmag = []

# 1.8e3 < freq < 2e3
midfreq = []
midmag = []

# freq > 2e3
highfreq = []
highmag = []

    # shove that signal into the FFT!
freqX, magX, angX = FFT(signal,s)
```

```
for i in range(len(freqX)):

    if (freqX[i] < 1.8e3):
        lowfreq.append(freqX[i])
        lowmag.append(magX[i])

    if ((freqX[i] <= 2e3) and (freqX[i] >= 1.8e3)):
        midfreq.append(freqX[i])
        midmag.append(magX[i])

    if (freqX[i] > 2e3):
        highfreq.append(freqX[i])
        highmag.append(magX[i])


# FFT plotting
fig = plt.figure(figsize = (10,10), constrained_layout = True)

# Magnitueds
FFTMag = plt.subplot2grid((5,1), (0,0))
FFTMag = make_stem(FFTMag, freqX, magX)
# plotting FFTmag
FFTMag.set_title("Input Magnitudes")
FFTMag.set_xlabel("Frequency (Hz)")
FFTMag.set_ylabel("Magnitude (V)")
FFTMag.grid()

# zoom in on sections
# low freq. zoom
FFTlowfreq = plt.subplot2grid((5,1), (1,0))
FFTlowfreq = make_stem(FFTlowfreq, lowfreq, lowmag)
# plotting stuff
FFTlowfreq.set_title("Low Frequency Magnitudes")
FFTlowfreq.set_xlabel("Freqency (Hz)")
FFTlowfreq.set_ylabel("Magnitude (V)")
FFTlowfreq.grid()

# mid freq. zoom
FFTmidfreq = plt.subplot2grid((5,1), (2,0))
FFTmidfreq = make_stem(FFTmidfreq, midfreq, midmag)
# plotting stuff
FFTmidfreq.set_title("Mid Frequency Magnitudes")
FFTmidfreq.set_xlabel("Freqency (Hz)")
FFTmidfreq.set_ylabel("Magnitude (V)")
FFTmidfreq.grid()
```

```
# high freq. zoom
FFThighfreq = plt.subplot2grid((5,1), (3,0))
FFThighfreq = make_stem(FFThighfreq, highfreq, highmag)
# plotting stuff
FFThighfreq.set_title("High Frequency Magnitudes")
FFThighfreq.set_xlabel("Freqency (Hz)")
FFThighfreq.set_ylabel("Magnitude (V)")
FFThighfreq.grid()

plt.show()


#-----------------Filter information-----------------
# A lot of the bode stuff in this section is modified from lab10

bandwidth = 800 * 2 * np.pi # Hz converted to rad/s
centerfreq = 1.9e3 * 2 * np.pi # Hz converted to rad/s

R = 10
L = 1.989e-3
C = 3.527e-6

numerator = [0, R/L, 0]
denominator = [1, R/L, 1/(L*C)]

# find w for stuff
# step size, start, and stop
step = 1
start = 0
stop = 9e6
w = np.arange(start, stop, step)

# transfer function
Hs = con.TransferFunction(numerator, denominator)

# entire bode plot
plt.figure(figsize = (10, 7))
plt.title("Entire Bode")
ang,mag,phase = con.bode(Hs, w * 2 * np.pi, dB=True, Hz=True, deg=True, plo

# low bode plot
plt.figure(figsize = (10, 7))
plt.title("Low Freqencies")
ang,mag,phase = con.bode(Hs, np.arange(1, 1.8e3, 10) * 2 * np.pi, dB=True,
```

```
# desired frequencies
plt.figure(figsize = (10, 7))
plt.title("Desired Frequencies")
ang,mag,phase = con.bode(Hs, np.arange(1.8e3, 2e3, 10) * 2 * np.pi, dB=True

# high frequencies
plt.figure(figsize = (10, 7))
plt.title("High Freqencies")
ang,mag,phase = con.bode(Hs, np.arange(2e3, 1e6, 10) * 2 * np.pi, dB=True,

# time to filter this thang!
# but first, z-transform
Znum, Zden = sig.bilinear(numerator,denominator,s)

# now, filter it :)
signalFiltered = sig.lfilter(Znum, Zden, signal)

# now it's been filter, now it must be plotted!
plt.figure(figsize = (10,7))
plt.title("Filtered Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude (V)")
plt.plot(t,signalFiltered)
plt.grid()
plt.show()

#Comparison!
plt.figure(figsize=(20,10))
plt.figure(constrained_layout=True)
plt.subplot(2,1,1)
plt.title("Filtered vs. Unfilter Signal!")
plt.xlabel("Time (s)")
plt.ylabel("Unfiltered (V)")
plt.plot(t,signal)

plt.subplot(2,1,2)
plt.plot(t, signalFiltered)
plt.grid()
plt.xlabel("Time (s)")
plt.ylabel("Filtered (V)")
# note: Resulting graph shows a much smoother signal that will be shoved in

# Shove that filtered signal into a FFT!
filtfreq, filtmag, filtang = FFT(signalFiltered, s)
```

```
"""
# this wasn't working for some reason...
# I had the wrong sampling freqency or something so it wasn't doing stuff I
# wanted. I'm too lazy to put thi back in :)
    # initiate arrays for spliting up frequency stuff
# freq < 1.8e3
lowfreqfilt = []
lowmagfilt = []

# 1.8e3 < freq < 2e3
midfreqfilt = []
midmagfilt = []

# freq > 2e3
highfreqfilt = []
highmagfilt = []

for i in range(len(freqX)):

    if (filtfreq[i] < 1.8e3):
        lowfreqfilt.append(filtfreq[i])
        lowmagfilt.append(filtmag[i])

    if ((filtfreq[i] <= 2e3) and (filtfreq[i] >= 1.8e3)):
        midfreqfilt.append(filtfreq[i])
        midmagfilt.append(filtmag[i])

    if (filtfreq[i] > 2e3):
        highfreqfilt.append(filtfreq[i])
        highmagfilt.append(filtmag[i])
"""

# FFT plotting
fig2 = plt.figure(figsize = (10,10), constrained_layout = True)

# Magnitueds
FFTMagfilt = plt.subplot2grid((4,1), (0,0))
FFTMagfilt = make_stem(FFTMagfilt, filtfreq, filtmag)
# plotting FFTmag
FFTMagfilt.set_title("Input Magnitudes")
FFTMagfilt.set_xlabel("Frequency (Hz)")
FFTMagfilt.set_ylabel("Magnitude (V)")
FFTMagfilt.set_xlim(0, 9e5)
FFTMagfilt.grid()
```

```
# zoom in on sections
# low freq. zoom
FFTlowfreqfilt = plt.subplot2grid((4,1), (1,0))
FFTlowfreqfilt = make_stem(FFTlowfreqfilt, filtfreq, filtmag)
# plotting stuff
FFTlowfreqfilt.set_title("Low Frequency Magnitudes")
FFTlowfreqfilt.set_xlabel("Freqency (Hz)")
FFTlowfreqfilt.set_ylabel("Magnitude (V)")
FFTlowfreqfilt.set_xlim(0, 1.8e3)
FFTlowfreqfilt.grid()

# mid freq. zoom
FFTmidfreqfilt = plt.subplot2grid((4,1), (2,0))
FFTmidfreqfilt = make_stem(FFTmidfreqfilt, filtfreq, filtmag)
# plotting stuff
FFTmidfreqfilt.set_title("Mid Frequency Magnitudes")
FFTmidfreqfilt.set_xlabel("Freqency (Hz)")
FFTmidfreqfilt.set_ylabel("Magnitude (V)")
FFTmidfreqfilt.set_xlim(1.79e3, 2e3)
FFTmidfreqfilt.grid()

# high freq. zoom
FFThighfreqfilt = plt.subplot2grid((4,1), (3,0))
FFThighfreqfilt = make_stem(FFThighfreqfilt, filtfreq, filtmag)
# plotting stuff
FFThighfreqfilt.set_title("High Frequency Magnitudes")
FFThighfreqfilt.set_xlabel("Freqency (Hz)")
FFThighfreqfilt.set_ylabel("Magnitude (V)")
FFThighfreqfilt.set_xlim(2.1e3, 9e5)
FFThighfreqfilt.grid()

plt.show()
```