Project Report for ECE 351

Lab 12: Filter Design

Isaias Munoz

December 8, 2022

University of Idaho

Contents

1 Introduction

The purpose of this lab is to apply the awesome Python skills developed through the previous labs to filter out a noisy signal within given paramters and present it in a professional manner.

2 Methodology

2.1 Part 1

The problem statement consisted of the student supposedly working for an aircraft company on a positioning control system that controls the position of the landing for their aircraft. this positioning system requires a feedback signal from a position center. The good information frequency is contained within 1.8 kHz to 2 kHz. There is however unwanted frequencies below 1.8khz that we want to get rid of that are due from the building's ventilation system. There is also high frequencies from the switching amplifier to the ground connection. To begin I needed to import the CSV file that contains the noisy signal from an oscilloscope and to plot it.

```
# load input signal
df = pd.read_csv('NoisySignal.csv')
t = df['0']. values
sensor_sig = df['1']. values
#this plots the signal coming in only plot 1 noisy
plt.figure(figsize = (10, 7))
plt.plot(t, sensor_sig)
plt.grid()
plt.title('Noisy Input Signal plot 1')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude [V]')
plt.show()
# your code starts here , good luck
#need to convert this signal first i think
fs = 1/(t[1]-t[0]) #or the 100khz this just calculates it for me
#because it encapsulates all possibilities

def myfft(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)
    for i in range (len(X_mag)):
        if np.abs(X_mag[i])<1e-10:

            X_phi[i]=0
        # else:
            # X_phi = np.angle(X_fft_shifted) # compute the phases of the signal
# ----- End of user defined function ----- #
    return freq,X_mag,X_phi #Need to assign these to a value
freq1,X_mag1,X_phi1=myfft(sensor_sig,fs)#labeling the returns so i can acces them when i go to p
#passing noisy signal to do a fft

1
```

Figure 1: Code for plotting noisy signal and my FFT

As can be seen in Figure 1 the plotting the noisy signal was relatively straightforward and the graph is under the results section. Next I needed to find the values or magnitudes in which the noisy signals occurred. I went about this by doing a Fast Fourier Transform (FFT). This will show me the different frequencies in which these noisy frequencies happen again classifying them in three different regions, low, center, and high frequencies. As seen in figure 1 I used Lab 9 code to do the FFT. Line 77 calls the FFT and returns the frequencies, magnitudes and phase angles. Next I will be able to split the three regions I am interested in and see where the mos occurrences of them ocurre so then I can narrow what to eliminate. Figure 2 has the plot for calling the FFT graphs at specific intervals. Figure 2 shows one example showing the FFT of the whole noisy signal. I did this with frequencies for 0-1.6 kHz and 1.6-2.2 kHz and finally 2.2 - 100 kHz. Again these correspond to the low frequencies, the middle range spectrum which we want to filter everything else but that and the high frequencies.

```
81  def make_stem(ax, x, y, color='k', style='solid', label='', linewidths=2.5, **kwargs):
82      ax.axhline(x[0], x[-1], 0, color='r')
83      ax.vlines(x, 0, y, color=color, linestyles=style, label=label, linewidths=linewidths)
84      ax.set_ylim([1.05*y.min(), 1.05*y.max()])
85  # FFT plots of unfiltered signal
86  # fig, (ax,ax1,ax2,ax3,ax4) = plt.subplots(figsize=(10,7))
87  # this plots the unfiler fft of the signal showing us the frequencies at different values only
88  ##
89  #
90  #plot 2a
91  # make_stem(ax, freq1, X_mag1)
92  # plt.stem(freq1,X_mag1)
93  # plt.grid()
94  # plt.ylabel('|X(f)|')
95  # plt.xlabel('Frequency [Hz]')
96  # plt.title('Noisy Input Signal going through the fft showing all frequencies plt 2a')
97  # plt.show()
98  #Hardest part figuring how to
99  #make these graphsss!!!!!!!!!
100 #this shows the entire band unfiltered mag and phase
101 #-400k to 400k rad/s
102 #using
103 fig , ax = plt . subplots ( figsize =(10 , 7) )
104 make_stem ( ax ,freq1 , X_mag1 )
105 # make_stem ( ax ,freq1 , X_phi1 )
106 plt . title ('Showing the all the band magnitude')
107 plt.grid()
108 plt . xlabel ('Frequency [Hz]')
109 plt . ylabel ('magnittude')
110 plt . show ()
111 fig , ax = plt . subplots ( figsize =(10 , 7) )
112 make_stem ( ax ,freq1 , X_phi1 )
113 plt . title ('Showing the all the band phase')
114 plt.grid()
115 plt . xlabel ('Frequency [Hz]')
116 plt . ylabel ('phase')
117 plt . show ()
```

1

Figure 2: Code for plotting the FFT of noisy signal

Figure 2 shows the function that was provided to make stem graphs easier and was used accordingly. Figure 2 also shows the FFT graph of the noisy signal given. What it does not show is that in order to show specific x-axis ranges I needed and x-limit[] of such and such and that would narrow my window. After seeing the occurrences at certain frequencies I had to design a circuit to combat unwanted frequencies and only keep what is in the middle.

2.2 Part 2

Part 2 had to come up with a circuit to kill unwanted frequencies between a range of frequencies. This means a band-pass filter because There is certain frequencies in the middle I wanna keep and therefore a series RLC band-pass circuit was chosen. Also because the series RLC were on my 212 circuit notes with its appropriate equations. At the end of this section there is scratch work of actually coming up with the appropriate values. Here is a rundown of solving for them. After all we want the range of 1.8 - 2.0 kHz to be seen so I started with that as my corner frequencies. Solving for the middle frequency by doing $W_o = sqrt(W_1 * W_2)$. Then I chose and R of 100 $ohms$ because It is a good round number. I then solved for L by $L = R/B$ where B is the band-with between the corner frequencies in this case 200 Hz and for C by $W_o = 1/sqrt(L * C)$. after obtaining these foundation values I plug them in my code and made it so I could change my corner frequencies if my attenuation's were off.

1

```
214
215    wc1=1600*2*np.pi
216    wc2=2200*2*np.pi
217    #if i choose R=100 cause 100 seems cool
218    R=100
219    #B=bandwitch the difference between the corner freq
220    B=wc2-wc1
221    wo=np.sqrt(wc1*wc2)
222    L=R/B
223    C=1/((wo**2)*L)
224    #calculating by hand is this values but im sure
225    #iall need to shift them to adjust bode plot
226    # C = 88.64 * 1e-6
227    # L = 79.4 * 1e-3
228    # R = 100
229    print('These are my values for RLC series circuit')
230    print("R:", R)
231    print("L:", L)
232    print("C:", C)
233    num = [(R/L), 0]
234    den = [1, (R/L), (1/(L*C))]
235    #plotting the bode as is
236    step_size = 100
237    w = np.arange(1, (1e6 )+step_size, step_size)
238    #rreferencing lab 10 in plotting bode
239    #did not meet specifications I crashed the plane
240    #all i need is play with my rlc to filter thats all
241    plt.figure(figsize=(11,7))
242    sys=con.TransferFunction(num,den)
243    _ = con.bode(sys, w, dB= True, Hz = True, deg=True, Plot=True)
244    plt.title('showing the whole bode filtered ')
245    #passing the functions
246    thing1,thing2=sig.bilinear(num,den,fs)
247    final1=sig.lfilter(thing1,thing2,sensor_sig)
248    plt . figure ( figsize = (11 , 7) )
249    plt . plot (t, final1)
250    plt . ylabel ('Y output')
251    plt . xlabel ('seconds')
252    plt . title (' filtering the given function x(t) ')
253    plt . grid ()
```

Figure 3: Code for RLC values and filtering the transfer function

I also needed the transfer function of my series RLC because using Lab 10 code I would pass my transfer function coefficient's to $sig.bilinear()$ and then $sig.lfilter$. These library functions will pass my noisy signal through the RLC circuit using the RLC values I calculated and I should get the frequencies I wanted. The transfer function looks like so;

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

Inputting the coefficients only and passing it to the two functions to filter I obtain a final signal that should be filter which is what line 246 and 247 in Figure 3 is doing. Figure 3 also shows the code to graph the filter signals as well as the bode plot on lines 241-244. The graphs are found under the result section.

2.3 Part 3

Part 3 was to generate bode plots to confirm the appropriate attenuation's where the middle frequencies is attenuated by less then -.3 dB and the low frequencies by at least -30 dB and finally the high frequencies bu at least -21 dB. In order to achieve this special attention to your RLC components should be taken and your corner frequencies. I ended p switching my corner frequencies to obtain my attenuation's within the requirements.

```
277  #time to generate the bode plots
278
279  step_size = 1
280  w = np.arange(1e3, 1e6+step_size, step_size)
281
282  #rreferencing lab 10 in plotting
283  # plt.figure(figsize=(11,7))
284  # sys=con.TransferFunction(num,den)
285  # _ = con.bode(sys, w, dB= True, Hz = True, deg=True, Plot=True)
286
287  plt.figure ( figsize = (10 , 7) )
288  _ = con.bode ( sys , np.arange(1, 1600 +step_size, step_size)*2*np.pi , dB = True , Hz = True ,
289  plt.title('0 to 1600 to ')
290
291
292  plt.figure ( figsize = (10 , 7) )
293  _ = con.bode ( sys , np.arange(1600, 2200 +step_size, step_size)*2*np.pi , dB = True , Hz = True
294  plt.title('1600 to 2200 to ')
295
296  plt.figure ( figsize = (10 , 7) )
297  _ = con.bode ( sys , np.arange(2200, 1e6 +step_size, step_size)*2*np.pi , dB = True , Hz = True
298  plt.title('2200 to 1e6 to ')
299
300
301  ##now doing the fft of filter signal
302  #should see a difference if chose correct rlc
303  #parameters
304
```

Figure 4: Code for plotting Bode plots and validating results

Figure 4 shows calling the bode plots at the different frequencies ranges to see their attenuation's and the graphs are under the result section. Like stated earlier choosing different corner frequencies changes your attenuation's and so on.

2.4 Part 4

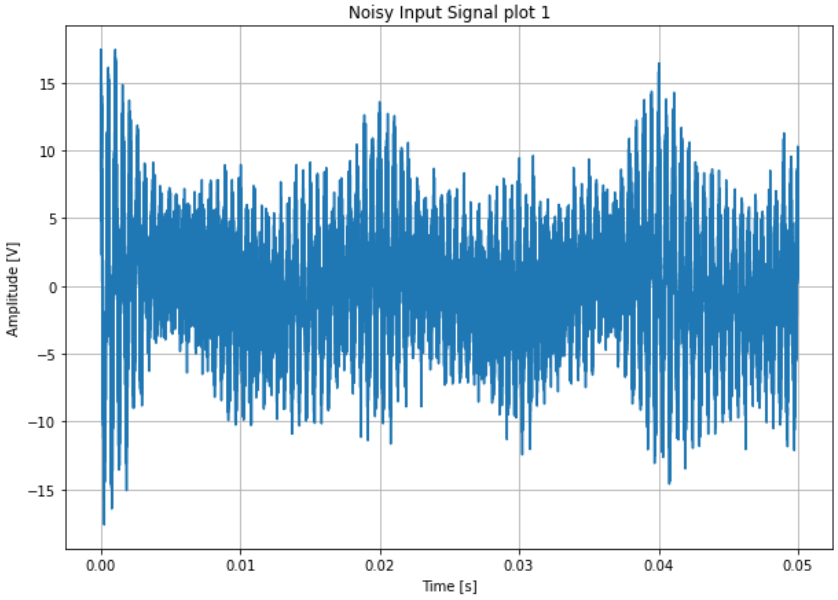
Finally I graphed the stem plots for the filtered signal which is just repeating Figure 2 plotting style except this time instead of the noisy signal it should be the clean signal that came out after putting it through the circuit. Figure 5 shows the code for the graphing section of the "clean" filter signal. If this works then I still have a job otherwise I'm jobless. The FFT graphs are shown in the result section.

```
#repeating the graphs of the fft before the filter
freq2,X_mag2,X_phi2=myfft(final1,fs)#labeling the returns so i can acces them when i go to plot
fig , ax = plt . subplots ( figsize =(10 , 7) )
make_stem ( ax ,freq2 , X_mag2 )
# make_stem ( ax ,freq1 , X_phi1 )

plt . title ('Showing the all the band magnitude filtered')
plt.grid()
plt . xlabel ('Frequency [Hz]')
plt . ylabel ('magnittude')
plt . show ()

fig , ax = plt . subplots ( figsize =(10 , 7) )
make_stem ( ax ,freq2 , X_phi2 )
plt . title ('Showing the all the band phase filtered')
plt.grid()
plt . xlabel ('Frequency [Hz]')
plt . ylabel ('phase')
plt . show ()
#repeating the above to show specific intervals
#showing 0 to 1800 magnitude and phase
fig , ax = plt . subplots ( figsize =(10 , 7) )
make_stem ( ax ,freq2 , X_mag2 )
plt . title ('Showing the  band magnitude 0-1600 filtered')
plt.grid()
plt.xlim(0, 1600)
plt . xlabel ('Frequency [Hz]')
plt . ylabel ('magnittude')
plt . show ()

fig , ax = plt . subplots ( figsize =(10 , 7) )
make_stem ( ax ,freq2 , X_phi2)
plt . title ('Showing the  the band phase0-1600 filtered')
plt.grid()
plt.xlim(0, 1600)
plt . xlabel ('Frequency [Hz]')
plt . ylabel ('phase')
plt . show ()

1
```

Figure 5: Code for plotting Bode plots and validating results

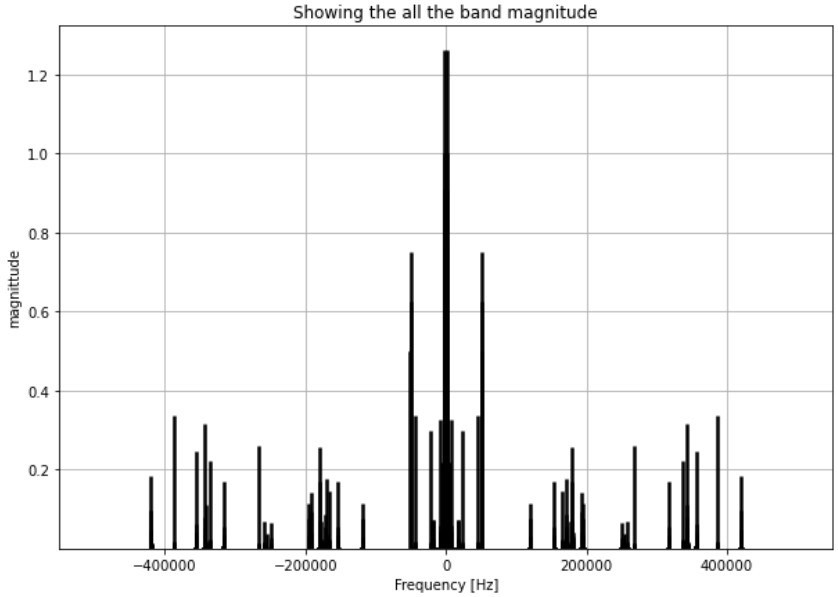
3 Results

3.1 Part 1



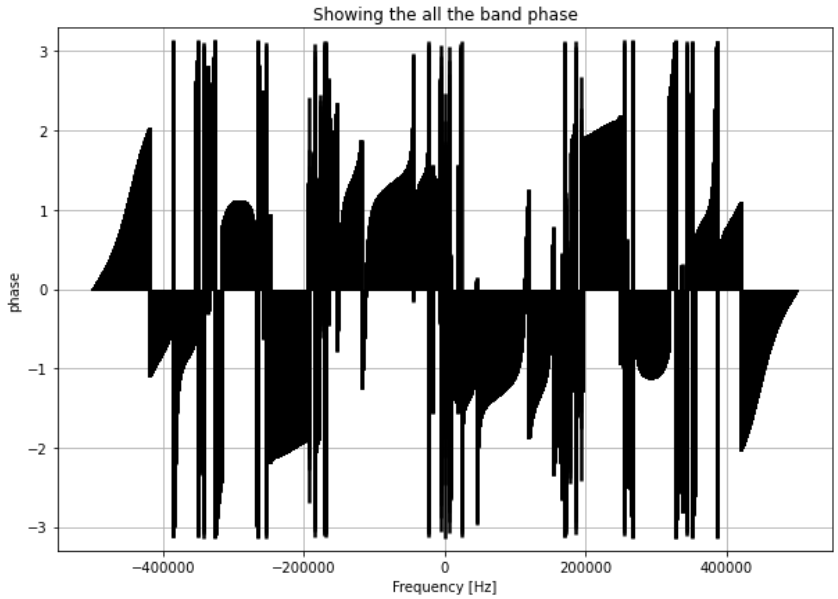
1

Figure 6: Plotting the noisy signal



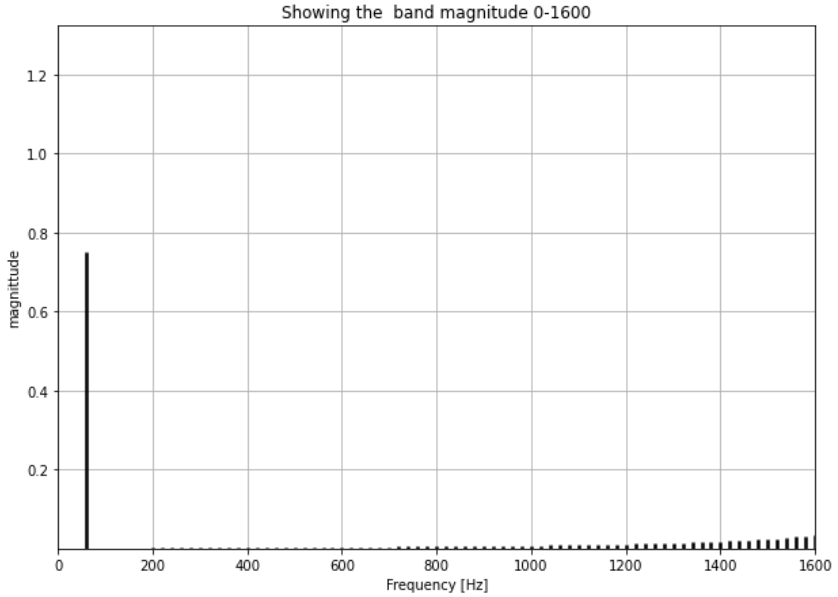
1

Figure 7: FFT of the noisy signal all frequencies magnitudes



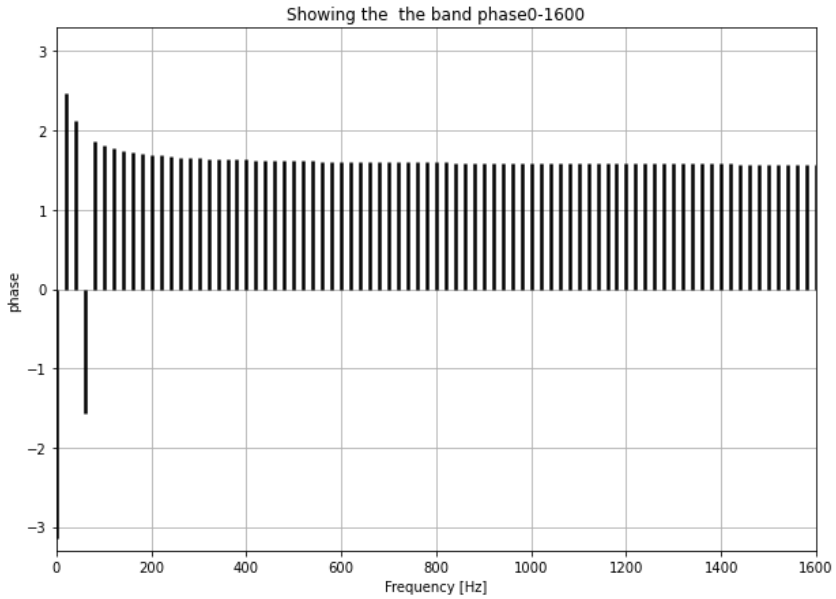
1

Figure 8: FFT of the noisy signal all frequencies phase



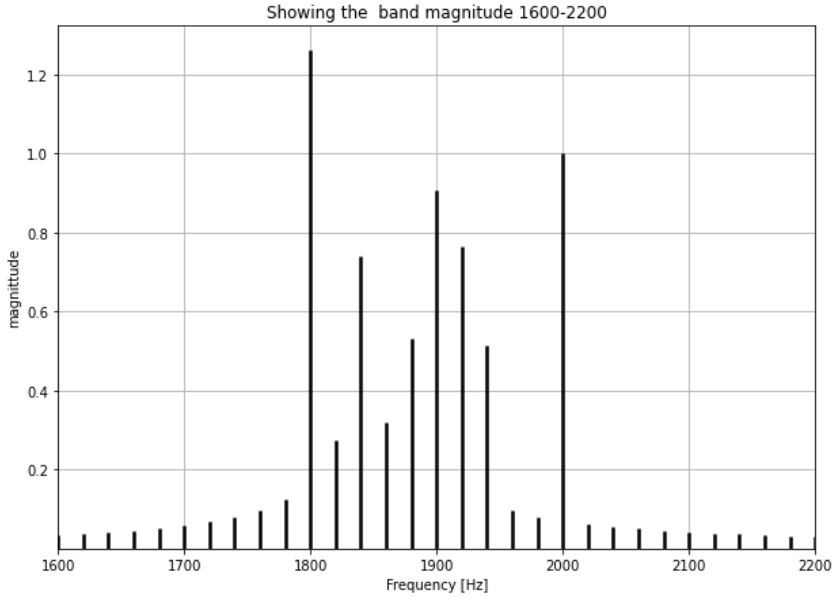
1

Figure 9: FFT of the noisy signal magnitudes 0 - 1600 Hz



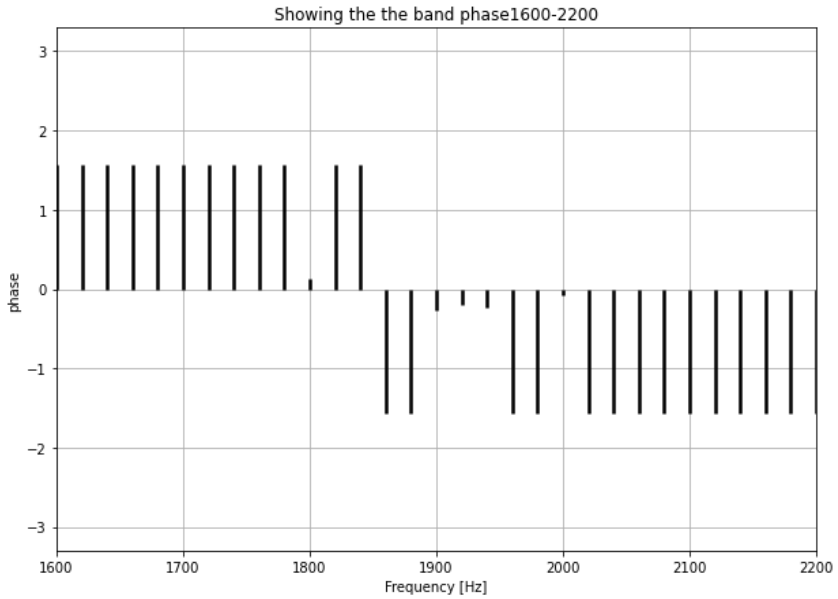
1

Figure 10: FFT of the noisy signal phases 0 - 1600 Hz



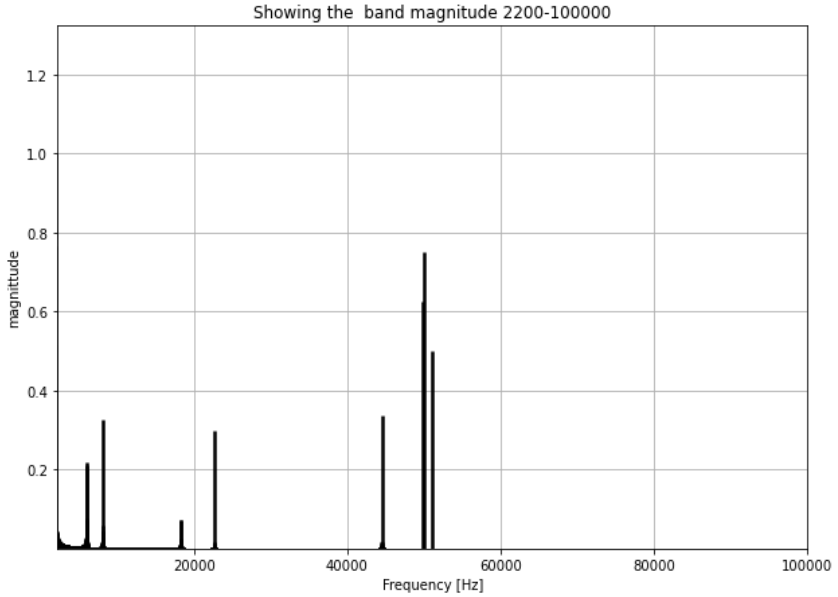
1

Figure 11: FFT of the noisy signal magnitudes 1600-2200 Hz



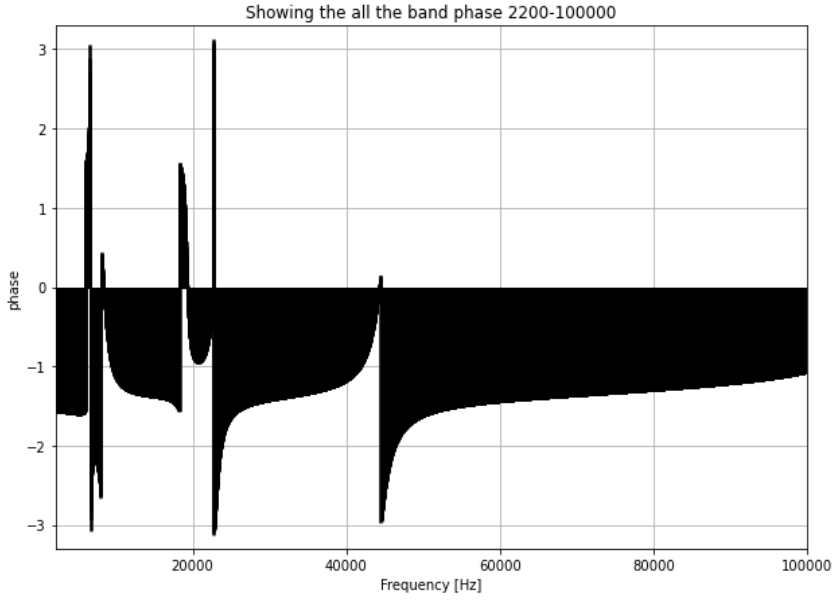
1

Figure 12: FFT of the noisy signal phase 1600-2200 Hz



1

Figure 13: FFT of the noisy signal magnitude 2200 - 100000 Hz



1

Figure 14: FFT of the noisy signal phase 2200 - 100000 Hz

3.2 Part 2

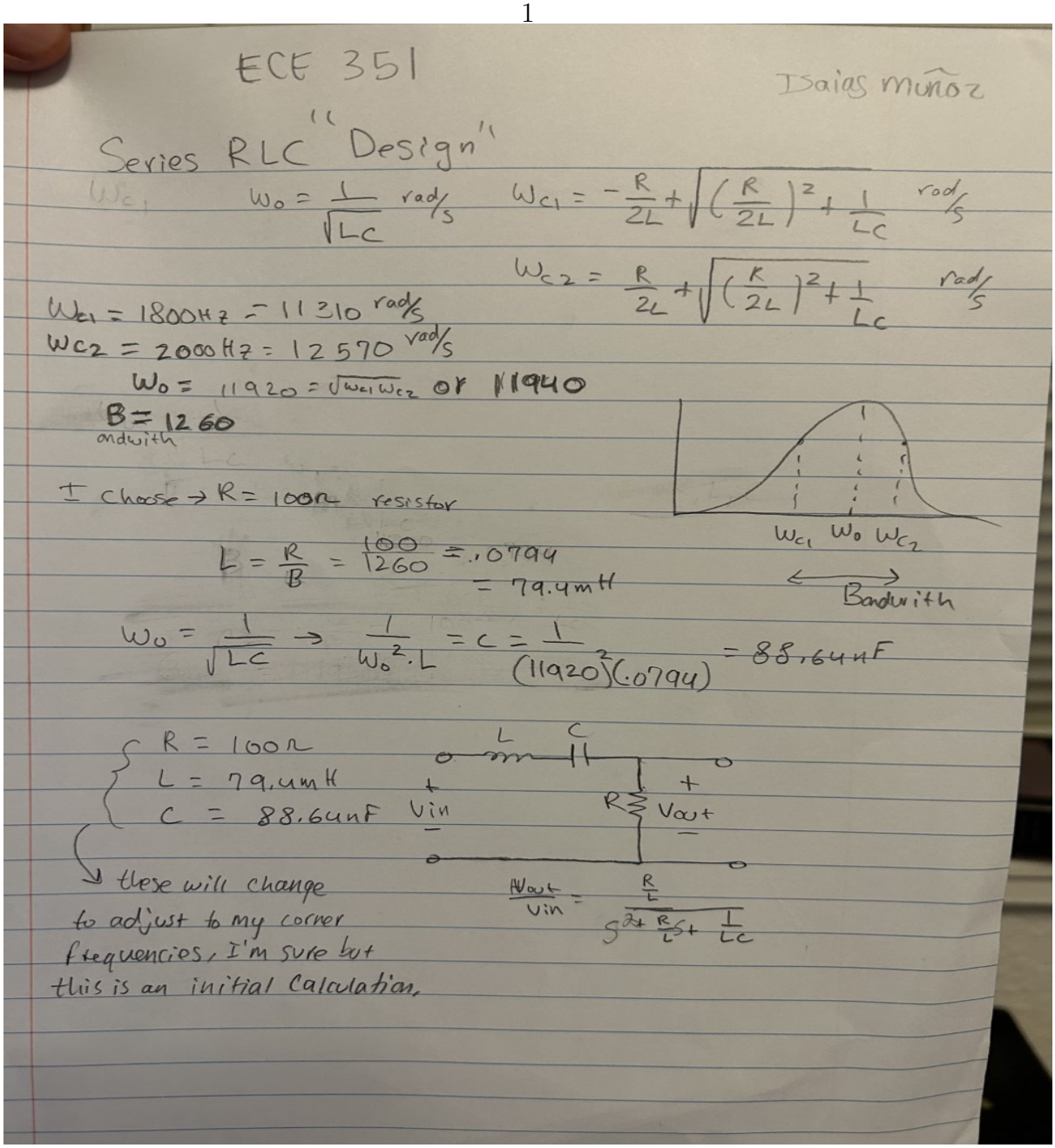
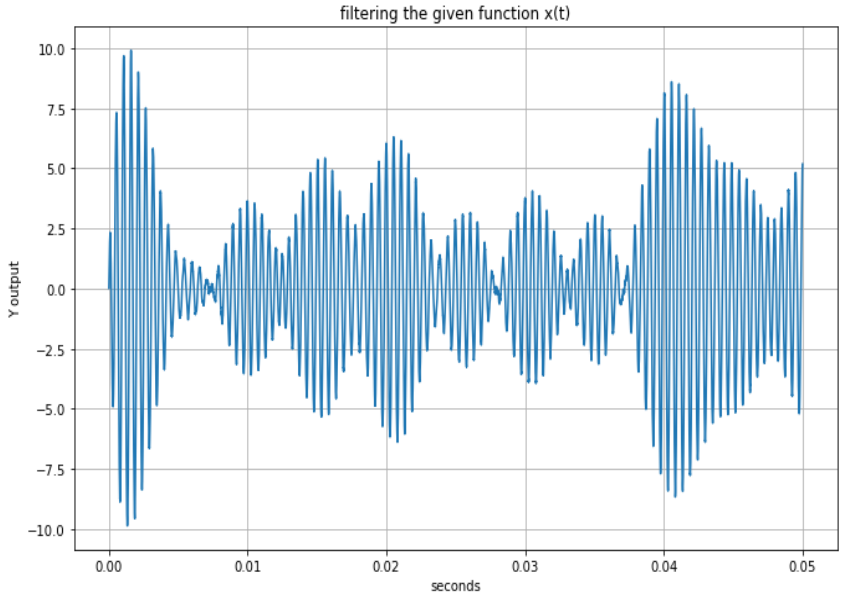


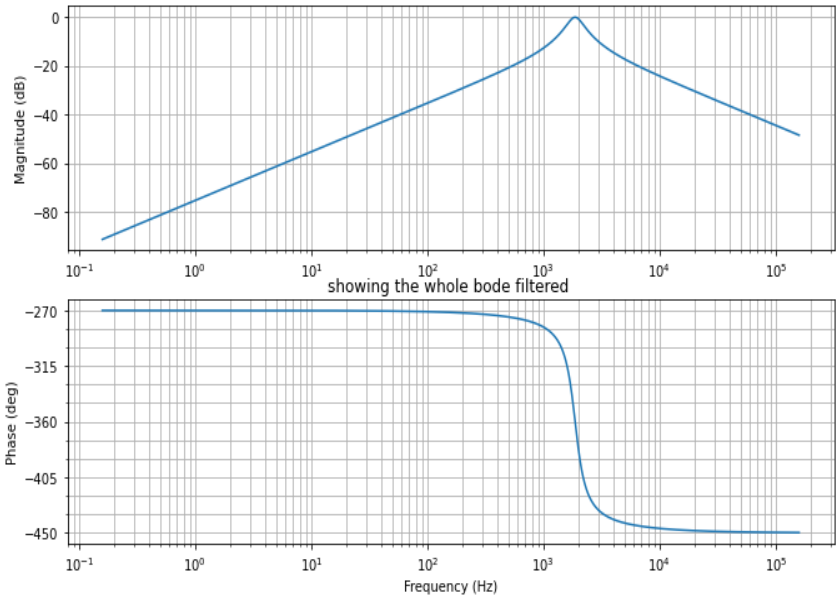
Figure 15: Series RLC figuring out base components

3.3 Part 3



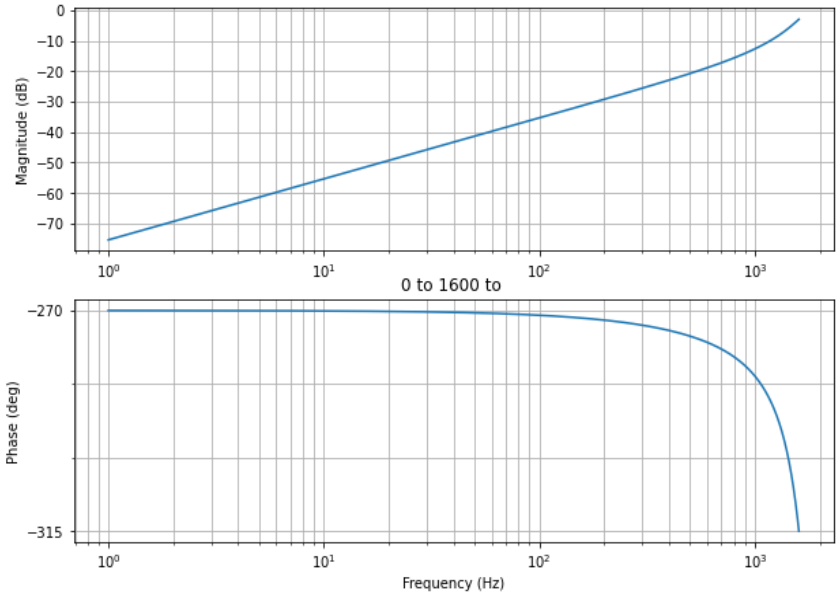
1

Figure 16: Filter signal after passing it through the RLC circuit



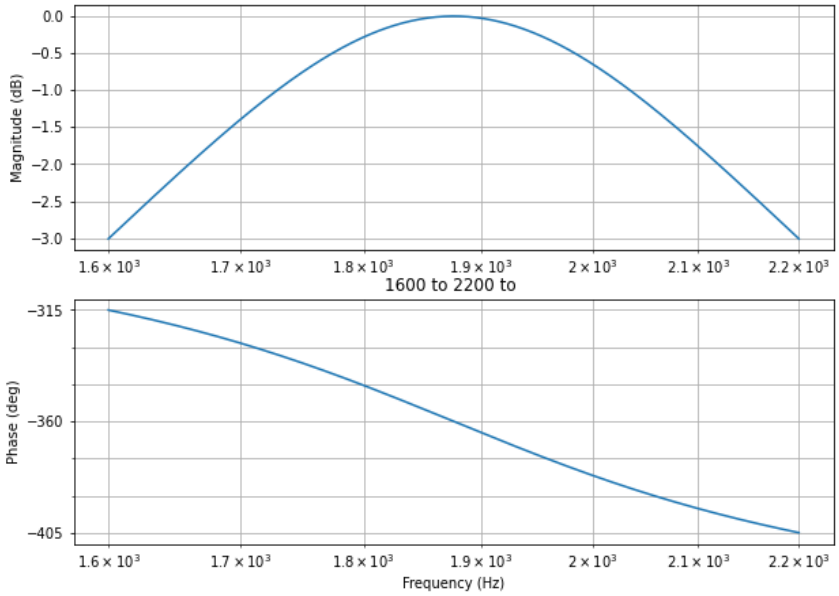
1

Figure 17: Bode plot showing all frequencies magnitude and phase after filtering it



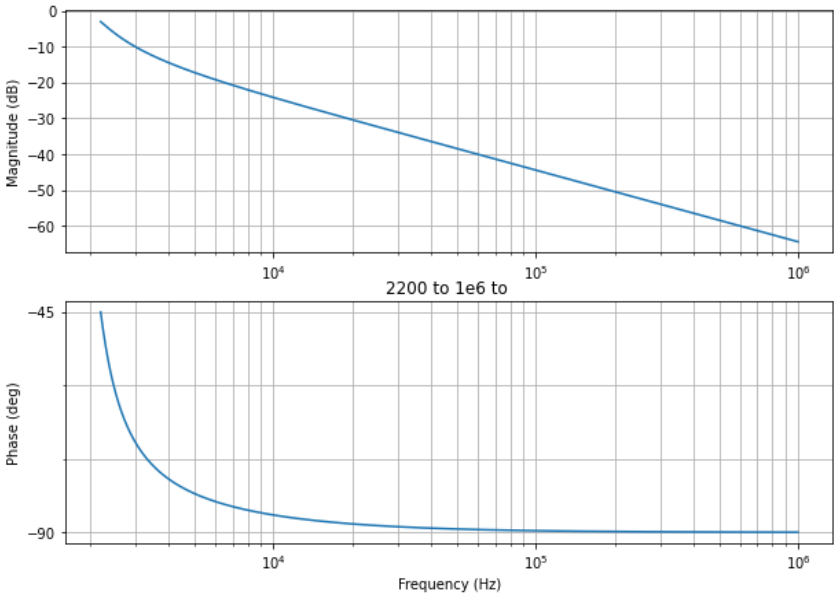
1

Figure 18: Bode plot magnitude and phase after filtering 0 - 1600 Hz



1

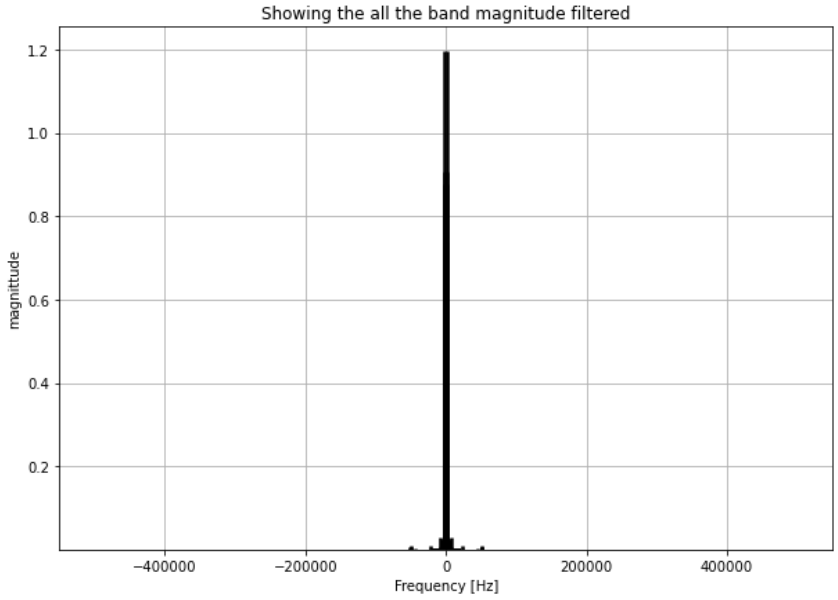
Figure 19: Bode plot magnitude and phase after filtering 1600 - 2200 Hz



1

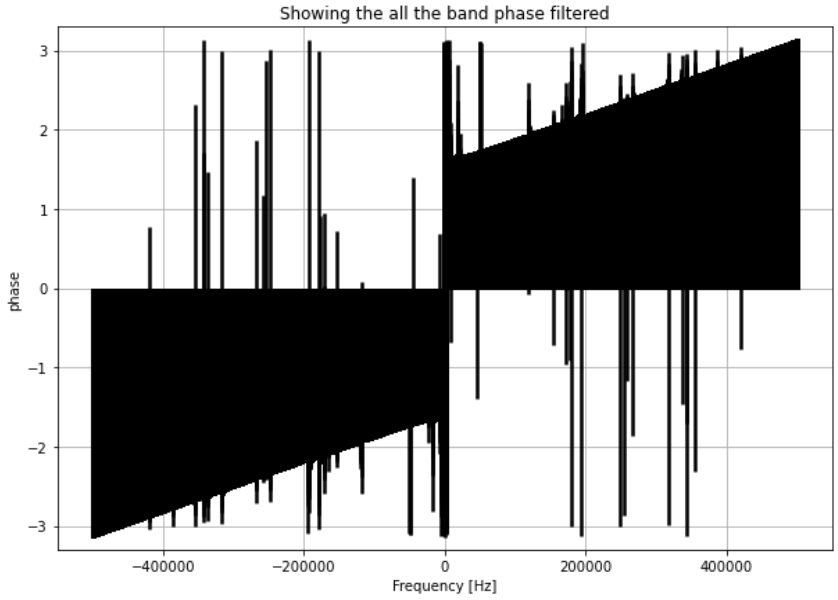
Figure 20: Bode plot magnitude and phase after filtering noisy signal phases 2200 - 100000 Hz

3.4 Part 4



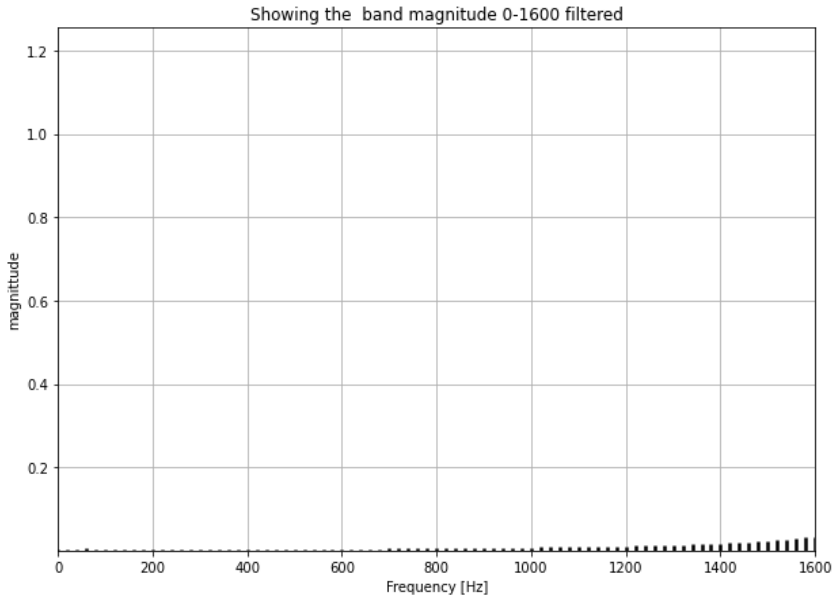
1

Figure 21: FFT of the filtered signal all frequencies magnitudes



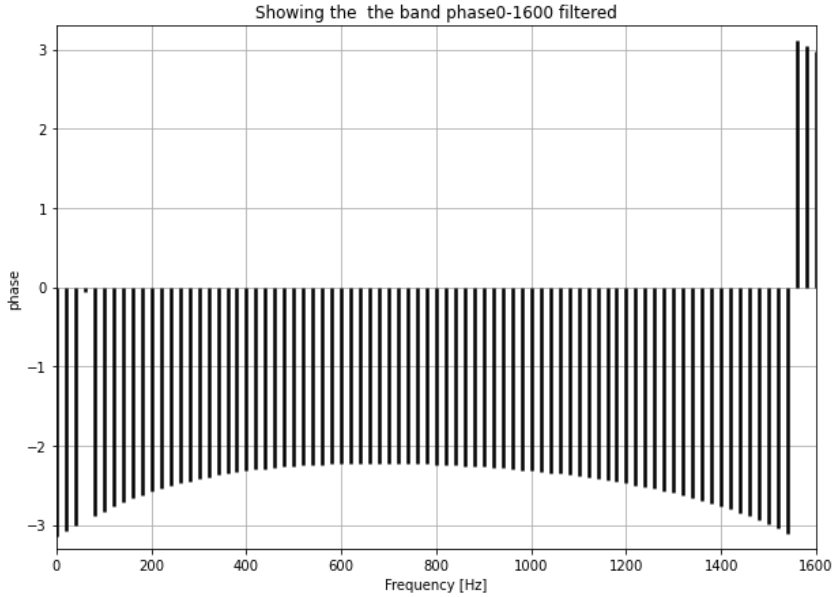
1

Figure 22: FFT of the filtered signal all frequencies phase



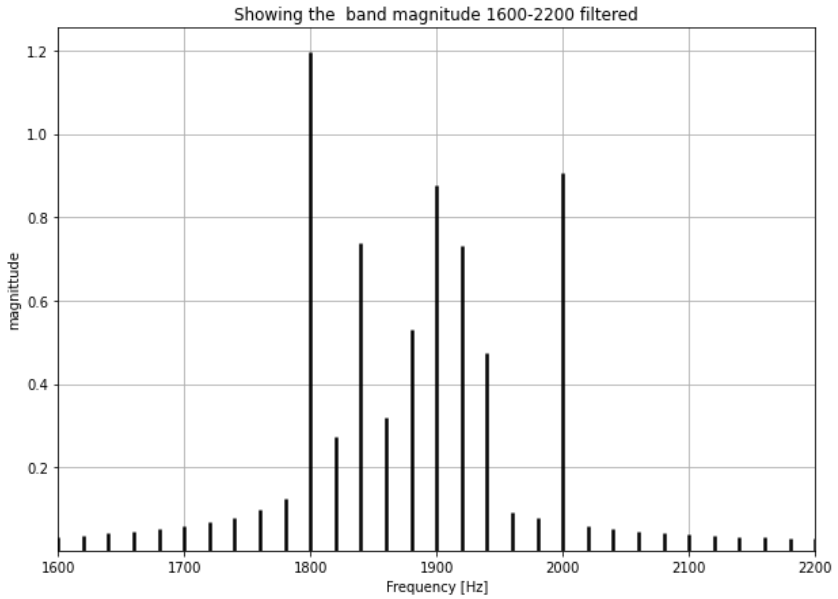
1

Figure 23: FFT of the filtered signal magnitudes 0 - 1600 Hz



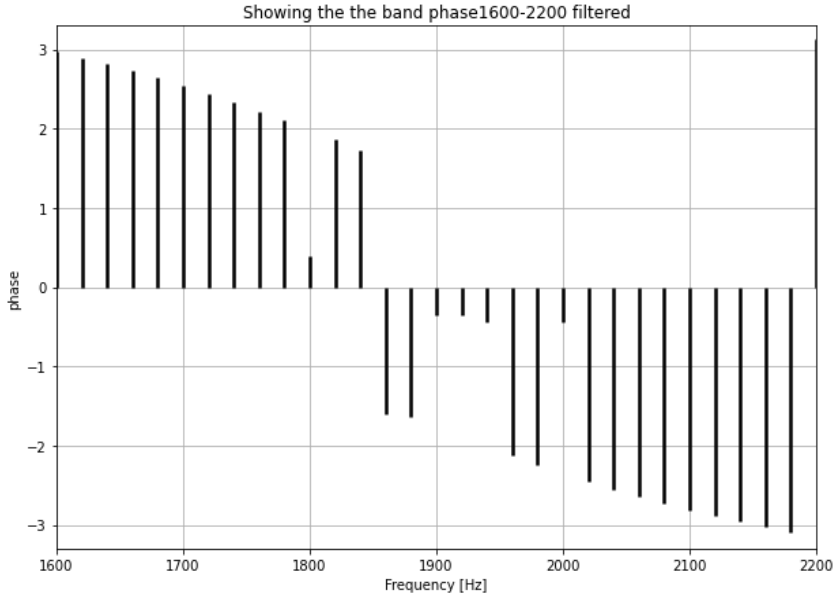
1

Figure 24: FFT of the filtered signal phases 0 - 1600 Hz



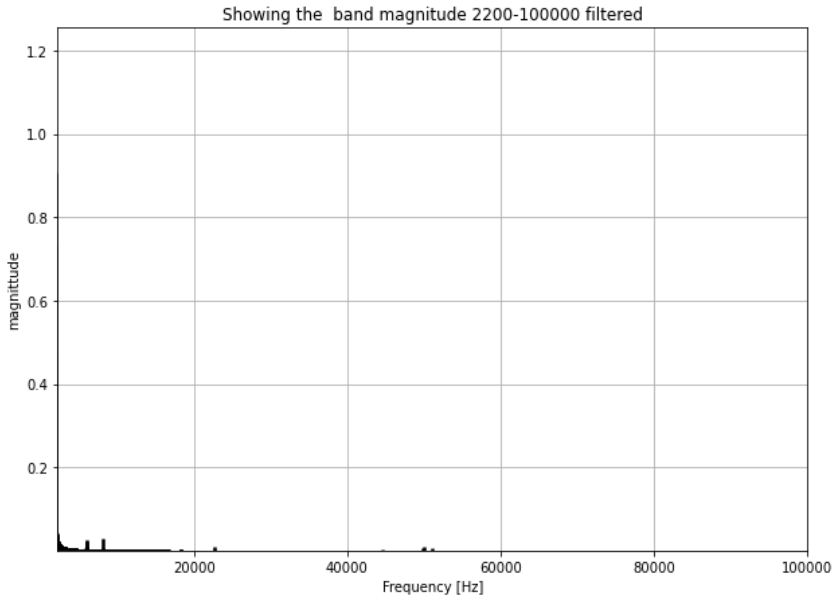
1

Figure 25: FFT of the filtered signal magnitudes 1600-2200 Hz



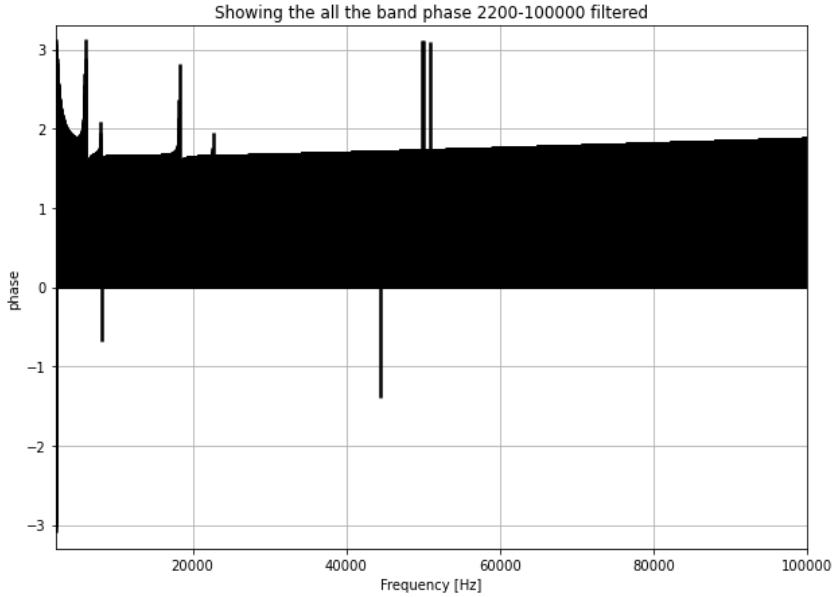
1

Figure 26: FFT of the filtered signal phase 1600-2200 Hz



1

Figure 27: FFT of the noisy filtered magnitude 2200 - 100000 Hz



1

Figure 28: FFT of the filtered signal phase 2200 - 100000 Hz

4 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 think and hope so, I learned more about python then what I started and wish this would be taught as your introductory class instead of C++ but I feel a little more comfortable then when I satrted so I would say yep.

2. Please fill out the course feedback survey, I will read every word and very much appreciate the feedback. .

3. Good luck in the rest of your education and career!
Thank You, you too!