Project Report for ECE 351

Lab 10: Frequency Response

Isaias Munoz

November 6, 2022

University of Idaho

Contents

1 Introduction

The purpose of this lab is to become familiar with frequency response tool and Bode plots using Python.

2 Methodology

2.1 Part 1

The first task that was assigned before the Lab began and that was to solve the below transfer function for the magnitude and phase by hand.

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

$$|H(j\omega)| = \frac{\frac{\omega}{RC}}{\sqrt{(\frac{1}{LC} - \omega^2)^2 + (\frac{\omega}{RC})^2}}$$

$$\angle H(j\omega) = 90 - tan^{-1}(\frac{\frac{\omega}{RC}}{\frac{1}{LC} - \omega^2})$$

The solving was done by inserting jw into the s domain and the rest is algebra. After doing so we needed to plot the expression of the magnitude and phase into python. As can be seen from the figure 1 below for the code. The expression above was typed in python and assigned to *ymag* then it was converted to a dB scaled on line 49. To plot this results *plt.semilogx* was used instead of a regular *plt.plot* because we were plotting in logarithmic axis. A similar approach to the phase expression except it needed some adjustments to when it came to solving for the angle. A for loop is implemented to match my results. After doing so I plotted both expressions and plots can be foudn under the results section.

```
steps = .5
w = np.arange(1e3, 1e6 + steps, steps)

r=1000
l=.027
c=.0000001
#trying to make a function to add all omegas but didnt work out
# def magfunc(w,r,c,l):
#     y = np.zeros(len(w))
#     for i in range(len(w)):
#         y+=20*np.log10(  (w/r*c)/  np.sqrt( ( 1 / (l * c) - w**2)**2
#     return y
#magn=magfunc(w,r,l,c)
#just wrote the simple equation to plot
ymag= w/(r*c)/  np.sqrt( ( 1 / (l * c) - w**2)**2 +   (w/(r*c))**2 )

ymagnit=20*np.log10(ymag)

yphase=90-np.arctan( w/(r*c) / (1/(l*c) -(w**2) ) )  * (180/np.pi)

for i in range(len(yphase)):
    if  (yphase[i] > 90):
        yphase[i] = yphase[i] - 180
    # else:
    #     yphase[i] = yphase[i]*(180/np.pi)

plt . figure ( figsize = (12 , 8) )
plt . semilogx (w, ymagnit )
plt . ylabel ('Y output dB')
plt . xlabel ('w rads/second')
plt . title ('magnitude of transfer function task 1 ')
plt . grid ()


plt . figure ( figsize = (12 , 8) )
plt . semilogx (w, yphase )
plt . ylabel ('Y output degrees')
plt . xlabel ('w rads/second')
plt . title ('phase angle of transfer function task 1')
```

Figure 1: Code for magntitude and phase of trasnfer function

After the first task was completed I moved onto using in house called functions to plot the magnitude and phase frequency response of the transfer function above and compare that with figure 1 code results. They should be identical because we are doing the same exact thing again.

```
77  #plotting using scipy.signal.bode
78
79
80  numer=[0,1/(r*c),0]
81  denom=[1,1/(r*c), 1/(l*c) ]
82
83
84  # ymagnit2=20*np.log10(denom)
85
86  warray, marray,parray=sig.bode((numer,denom),w)
87
88
89  plt . figure ( figsize = (12 , 8) )
90  plt . semilogx (warray, marray )
91  plt . ylabel ('Y output dB')
92  plt . xlabel ('w rads/second')
93  plt . title ('magnitude of transfer function using .signal.bode task 2')
94  plt . grid ()
95
96
97  plt . figure ( figsize = (12 , 8) )
98  plt . semilogx (warray, parray )
99  plt . ylabel ('Y output degrees')
00  plt . xlabel ('w rads/second')
01  plt . title ('phase angle using .signal.bode task 2 ')
02  plt . grid ()
1 03
```

Figure 2: Code for magntitude and phase of trasnfer function using *sig.bode*

As can be seen in figure 2 I needed and array with the coefficient of the expressions of magnitude and phase and passed it to *sig.bode* along with the x-axis and it will generate the magnitude and phase functions and I can plot. SO the in house function generates three things as can be seen in line 86. I then plot it using *plt.semilogx* and the plots can be found under the result section below.

For the last task in part 1 we want the frequency response with respect to Hz not rads/sec. We also could want a specific frequency we want to look at. This part of the code was provided and all we really had to do is pass it the numerator and denominator coefficients. Then it would generate a nice bode plot for us. The code is Figure 3 below..

```
plt.figure(figsize=(11,7))
sys=con.TransferFunction(numer,denom)
_ = con.bode(sys, w, dB= True, Hz = True, deg=True, Plot=True)
```

Figure 3: Code for plotting a bode plot nicely

2.2 Part 2

Part 2 was to filter a signal given below, and to pass it through the RLC circuit. In order to do that it needed to be converted into it's z-domain equivalent using $scipy.signal.bilinear()$ and then finally $scipy.signal, lfilter()$. Below is the code in doing so.

$$X(t) = cos(2\pi 100t) + cos(2\pi 3024t) + sin(2\pi 50000t)$$

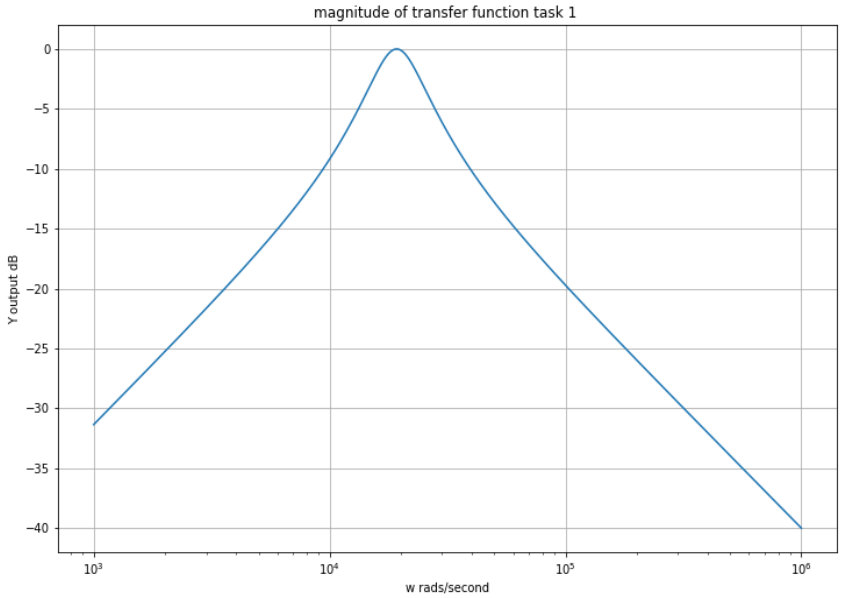
```
freq=100000 *np.pi
newsteps = 1/freq
t1 = np.arange(0, 0.01 + newsteps, newsteps)
funcx = np.cos(2*np.pi*100*t1) + np.cos(2*np.pi*3024*t1) + np.sin(2*np.pi*50000*t1)
plt . figure ( figsize = (10 , 6) )
plt . plot (t1, funcx )
plt . ylabel ('Y output')
plt . xlabel ('seconds')
plt . title ('Graphing x(t)')
plt . grid ()
# warray, marray,parray=sig.bode((numer,denom),w)
thing1,thing2=sig.bilinear(numer,denom,freq)
#spits
final1=sig.lfilter(thing1,thing2,funcx)
plt . figure ( figsize = (11 , 7) )
plt . plot (t1, final1)
plt . ylabel ('Y output')
plt . xlabel ('seconds')
plt . title ('Part 2 filtering the given function x(t) ')
plt . grid ()
```

Figure 4: Code for filtering x(t) signal

Setting the sampling frequency high enough to capture all three frequencies of the signal and a step size of 1/frequency was used. Bilinear generated to parameters which were used in the lfilter function along with the signal. Both plots before the filter and after the filtering are shown under the result sections.

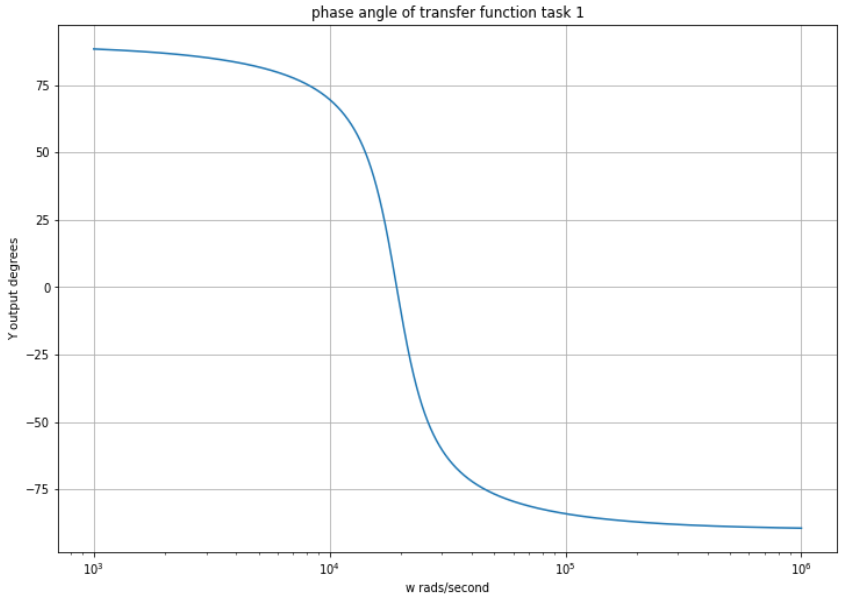
3 Results

3.1 Part 1



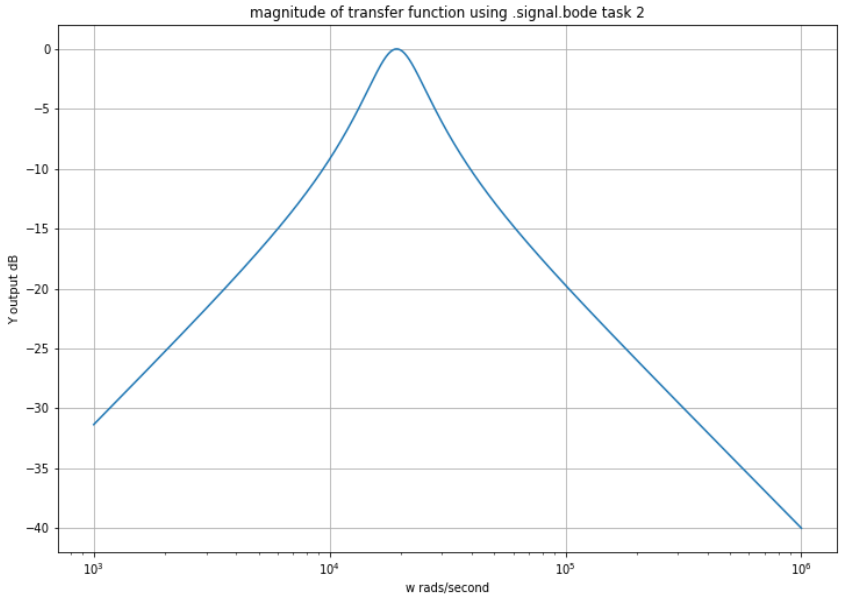
1

Figure 5: Part 1 Task 1



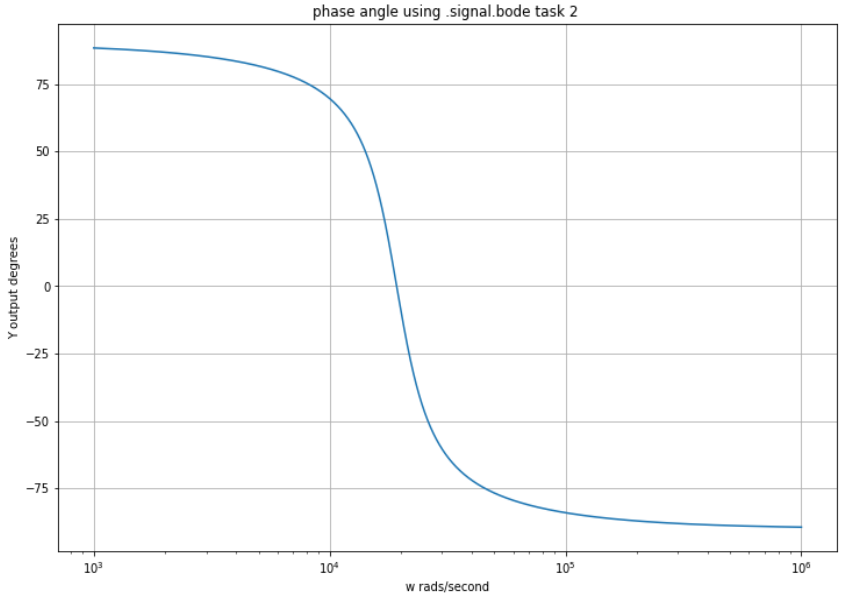
1

Figure 6: Part 1 Task 1



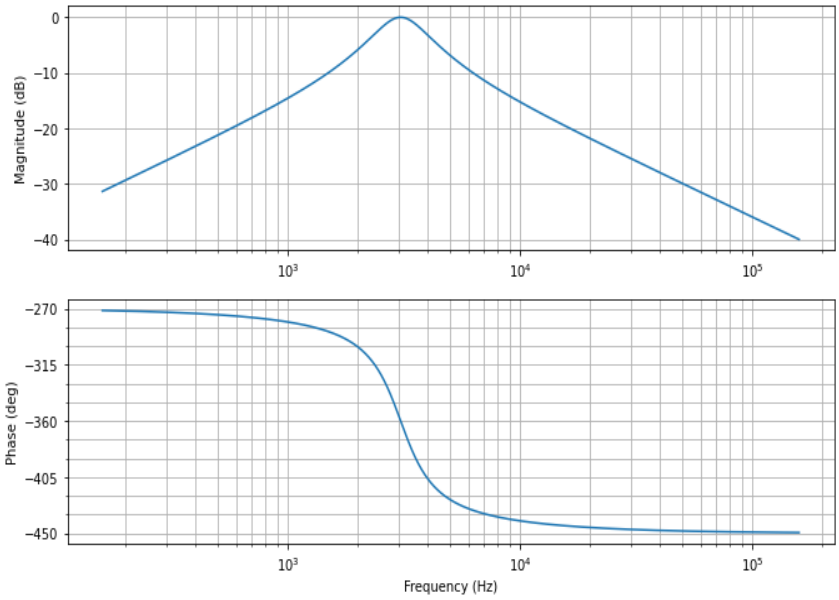
1

Figure 7: Part 1 Task 2



1

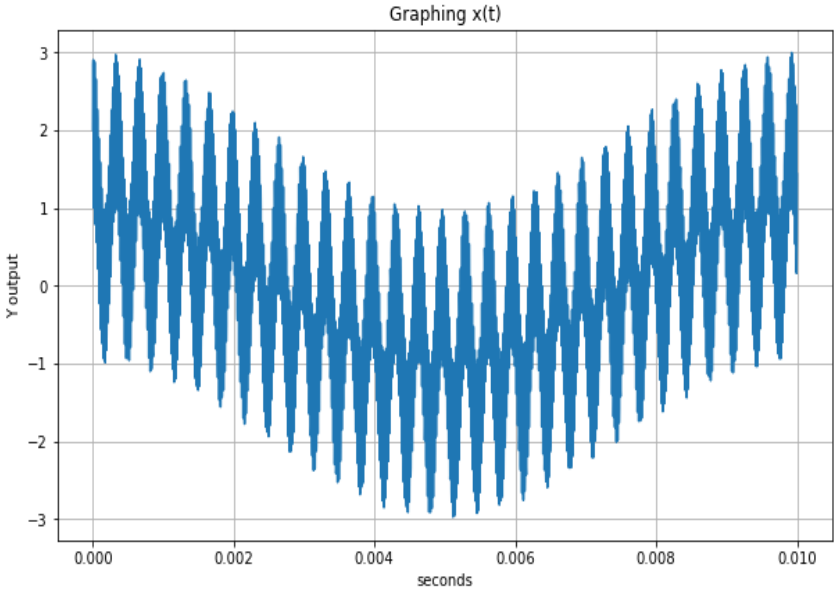
Figure 8: Part 1 Task 2



1

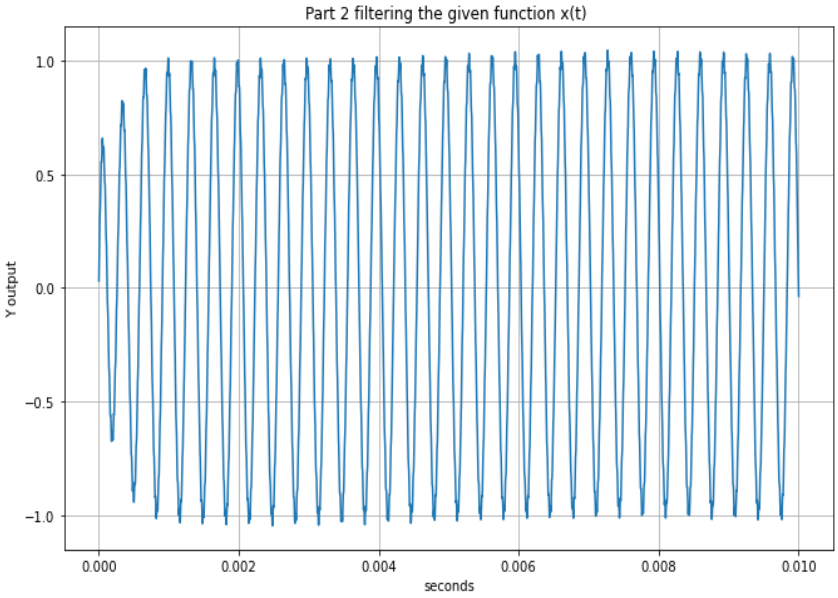
Figure 9: Part 1 Task 3

3.2 Part 2



1

Figure 10: Part 2 task 1



1

Figure 11: Part 2 Task 4

4 Questions

1. Explain how the filter and filtered output in Part 2 makes sense given the Bode plots from Part 1. Discuss how the filter modifies specific frequency bands, in Hz.
The filter is cutting all those higher outputs and only narrowing from -1 to 1 in height which result in what we see in the bode plot for the output after filtering the signal.

2. Discuss the purpose and workings of scipy.signal.bilinear() and scipy.signal.lfilter().
One converts the numerator and denominator along with the axis to a z domain and the other actually does the filtering.

3. What happens if you use a different sampling frequency in scipy.signal.bilinear() than you used for the time-domain signal?
You would see different samples and not the ones you actually are itnersted in.

4. Leave any feedback on the clarity of lab tasks, expectations, and deliverables.
It was straightforward I think.