ECE 351 - Lab 10

Khoi Nguyen
https://github.com/3khoin

11 November 2021

Contents

1 Introduction

The goal of this lab was to become familiar with Python's set of frequency response tools and use them to construct Bode plots for signals.

2 Equations

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

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

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

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

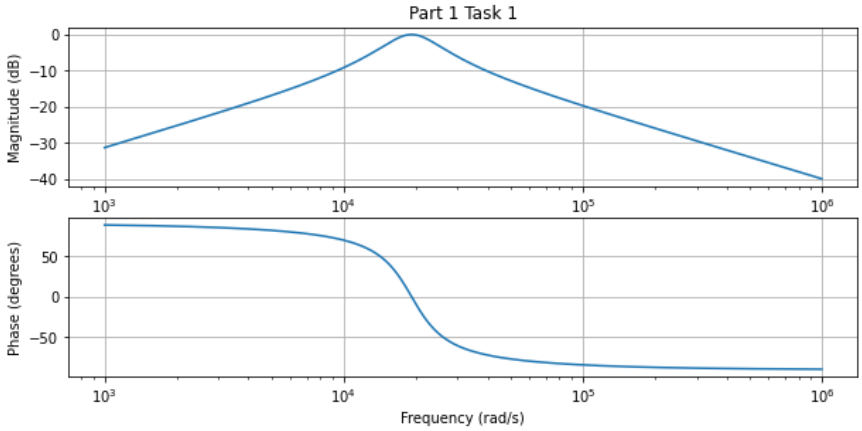
3 Methodology

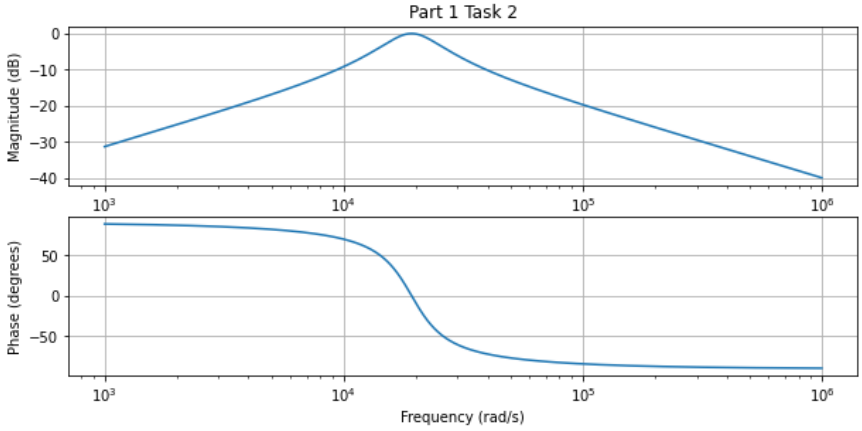
Using the transfer function H(s) described in the Equations section, we derived the magnitude and phase equations for H(s), and, using the matplotlib.pyplot.semilogx() function, plotted the two on a logarithmic scale (the magnitude equation had to be converted into dB). We then verified that our hand-calculated plots were correct with scipy.signal.bode(). We then used the con.bode() function from the "control" package to find the frequency response with respect to Hz.

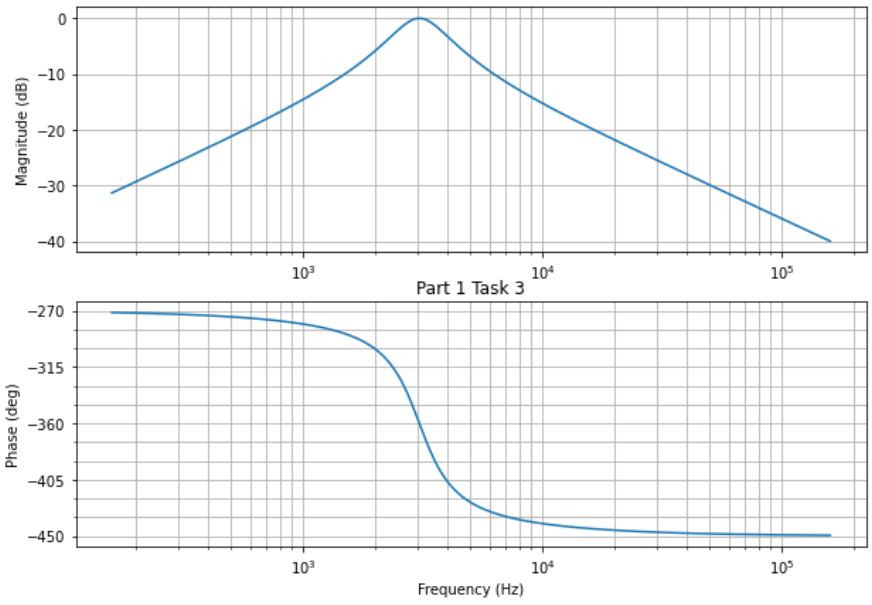
We then plotted the signal x(t) listed in Equations, with a sample frequency of 1e6 to ensure all 3 frequencies were captured, with a domain of $0 \leq t \leq 0.01$ s. We then passed x(t) through H(s) by first converting H(s) into its z-domain equivalent with the scipy.signal.bilinear() function, and then passing the resulting transform as an argument into scipy.signal.lfilter() alongside x(t). The final output y(t) was then plotted with the same domain as the original x(t).

4 Results

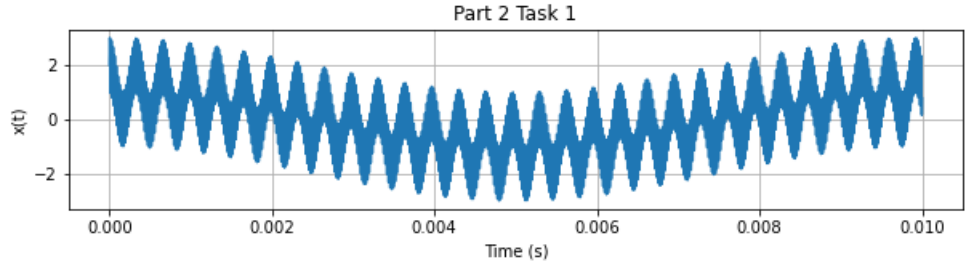
The magnitude and phase plots of the transfer function are listed below, with the third plot below being the magnitude and phase plot with the domain in Hz frequency rather than rad/s.

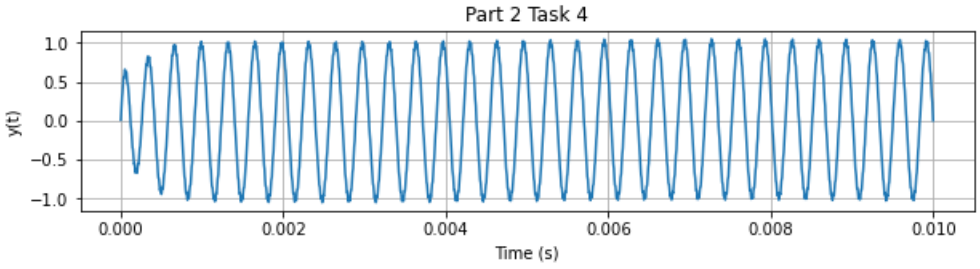

Part 1 Task 1
Magnitude (dB)
0
−10
−20
−30
−40
10^3
10^4
10^5
10^6
Phase (degrees)
50
0
−50
Frequency (rad/s)


Part 1 Task 2
Magnitude (dB)
0
−10
−20
−30
−40
10^3
10^4
10^5
10^6
Phase (degrees)
50
0
−50
Frequency (rad/s)


Magnitude (dB)
0
−10
−20
−30
−40
10^3
10^4
10^5
Part 1 Task 3
Phase (deg)
−270
−315
−360
−405
−450
10^3
10^4
10^5
Frequency (Hz)

x(t) is plotted below, as well as y(t) which passes x(t) through H(s).





5 Error Analysis

One error that we were challenged to work through showed up when we initially attempted to plot the phase for H(s); after a certain frequency, the resulting phases would be shifted π radians (180 degrees) from their intended location. To remedy this, we had to modify the phase function to subtract π from the final result if it was above the threshold of $\frac{\pi}{2}$.

6 Questions

1. The Part 2 filtered output makes sense when we look at the magnitude portion of the Part 1, Task 3 Bode plot; we can see that it is passing most the frequency of 3e3 Hz. Since the time domain plotted is 0.01 s, since $\frac{30}{3e3} = 0.01$, there should be about 30 periods in the Part 2, Task 4 plot (which there are).

2. The scipy.signal.bilinear() function maps a continuous time signal to a discrete one. This is necessary since Python plots use discrete values. The scipy.signal.lfilter() function passes an input signal through a given filter, which should be made discrete.

3. Using a different sampling frequency in scipy.signal.bilinear() than was used for the time-domains signal results in either a signal that is too noisy or one that has filtered too many frequencies out.

4. The lab tasks, expectations, and deliverables were very clear.

7 Conclusion

This lab introduced the methods with which to construct Bode plots and analyze Bode plots and signals in the frequency domain. With these tools, we were able to further understand how signal filtering operates.