Assignment 7

Lakshya J ee19b035

April 15th, 2021

Introduction

In this assignment we are asked to explore the scipy library and the packages provided by the same. The libraries I imported are as follows:

```python
import scipy.signal as sp
import numpy as np
import matplotlib.pyplot as plt
```

Q1

I begin by declaring the transfer function as given in the problem statement as follows:

```python
num = np.poly1d([1,0.5])
den = np.polymul([1,0,2.25],[1,1,2.5])
H = [num, den]
```

Then I use the *impulse*() function to generate the time varying output for a span of 50 seconds. Here is the plot obtained for the same: Since $x(0) = 0$ and $\dot{x} = 0$ the place transform of $\ddot{x}$ is $s^2X(s)$.
I used *scipy.signal.impulse* as follows:

```python
t, x = sp.impulse(H, None, np.linspace(0,50,501))
```

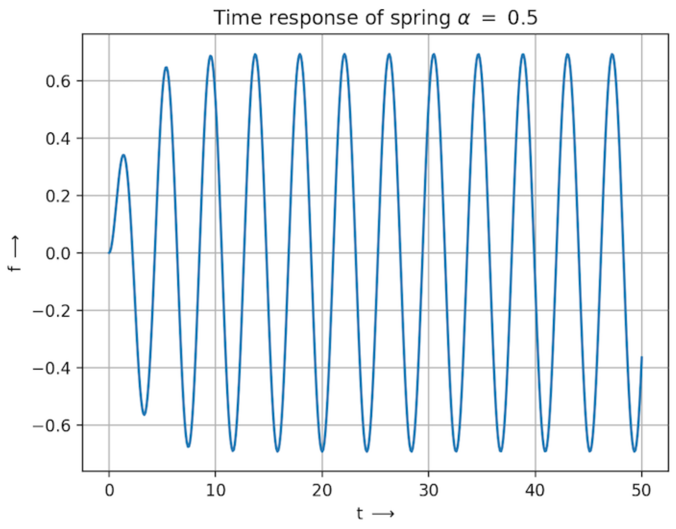

Figure 1: Time response of the spring with decay constant = 0.5

Q2

In this question we are supposed to tackle the same situation with a much smaller decay of the transfer function. Hence the new transfer function will be as follows:

```
num = np.poly1d([1,0.05])
den = np.polymul([1,0,2.25],[1,0.1,2.2525])
H = [num, den]
```

The output I obtained is as follows:

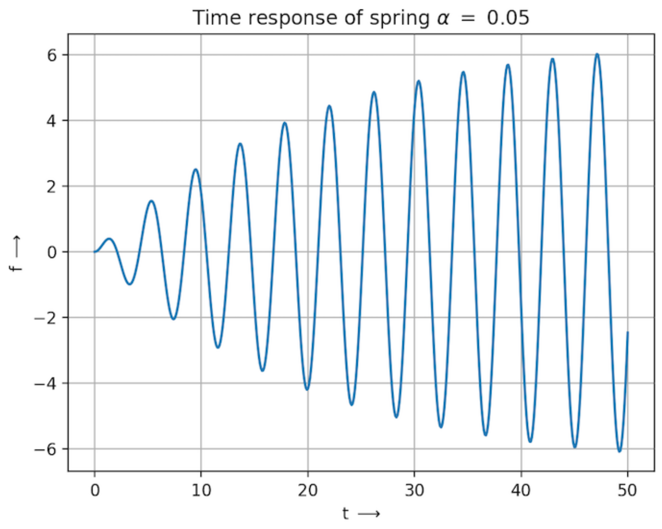

Figure 2: Time response of the spring with decay constant = 0.05

Q3

In this question we are asked to consider a LTI system to simulate the problem by varying the value of the frequency of the cosine from 1.4 to 1.6 keeping the decay $\alpha = 0.05$. Here also I am considering the initial conditions to be zero. The transfer function is simply $\frac{X(s)}{F(s)} = \frac{1}{s^2+2.25}$

```
num = np.poly1d([1])
den = np.poly1d([1,0,2.25])
H = [num, den]
freq = np.arange(1.4, 1.65, 0.05)
t = np.linspace(0, 50, 501)
```

I used *scipy.signal.lsim* to simulate the conolution of the tranfer function with the input time varying function whose frequency increases in steps of 0.05. I used a for loop to generste the multiple plots in a single figure. I plotted the different responses using the code as follows:

```
for i in freq:
  u = np.cos(i*t)*np.exp(-0.05*t)
  t, y, svec = sp.lsim(H, u, t)
  plt.plot(t, y)
```

The output I obtained is as follows: It can be noticed that at $\omega = 1.5$ the amplitude of vibration is the largest as the input function has the same frequency as that of the natural frequency of the spring, which leads us to observe the natural resonance phenomenon. It can be seen that at $\omega = 1.6$ the most number of cycles are completed in the period of 50 seconds.

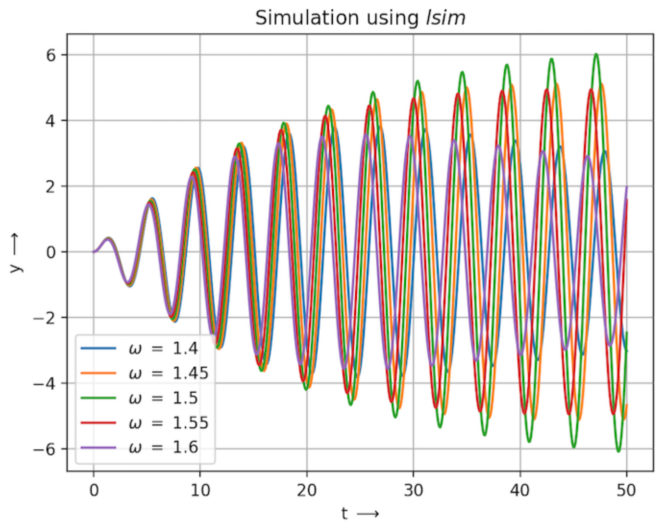

Figure 3: Plots with varying frequencies

Q4

In this section we attempt to solve the coupled spring problem. On solving the equation with substituting the initial conditions as required, here are the results I obtained for the Laplace transforms of x and y.

$$Laplace(x) = X(s) = \frac{s^2+2}{s(s^2+3)}$$
$$Laplace(y) = Y(s) = \frac{2}{s(s^2+3)}$$

On solving these equations for their time evolution in the given time span, here are the results I obtained.

```
# Solving for x vs time in laplace domain
numX = np.poly1d([1,0,2])
denX = np.poly1d([1,0,3,0])
X = [numX, denX]
t, x = sp.impulse(X, None, np.linspace(0,20,501))
# Solving for y vs time in laplace domain
numY = np.poly1d([2])
denY = np.poly1d([1,0,3,0])
Y = [numY, denY]
t, y = sp.impulse(Y, None, np.linspace(0,20,501))
```

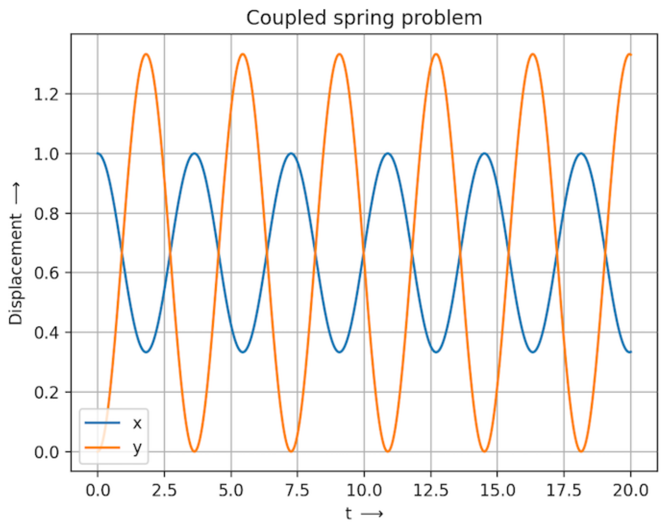

Figure 4: Plots with varying frequencies

Q5

I now solved for the steady state transfer function for the series RLC circuit given in the assignment. This is the tranfer function I obtained for the same: $H(s) = \frac{v_o(s}{v_i(s)} = \frac{1}{s^2LC+sRC+1}$
I declare the given values for the components and applied the *scipy.signal.lti* over the given transfer function and *scipy.signal.bode* function later helps me find the variation in magnitude and phase of the given transfer function.

```
L = 1e-6
C = 1e-6
R = 100
H = sp.lti([1],[L*C, R*C, 1])
w, S, phi = sp.bode(H)
```

The result obtained is as follows:

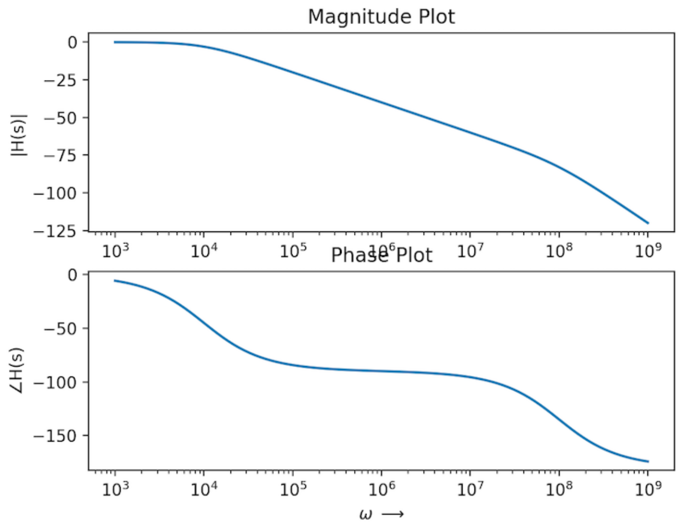

Figure 5: Magnitude and phase plots

Q6

Lastly we are supposed to consider an input function given as $v_i(t) = cos(10^3 t)u(t) - cos(10^6 t)u(t)$. The code used to achieve the same is as follows:

```
t = np.linspace(0, 0.02, 1000000)
vi = np.cos((10**3)*t) - np.cos((10**6)*t)
t, y, svec = sp.lsim(H, vi, t)
```

As can be seen from the code I simulate the same for 20 ms by taking 10^6 samples of the same. The plot obtained is as follows:

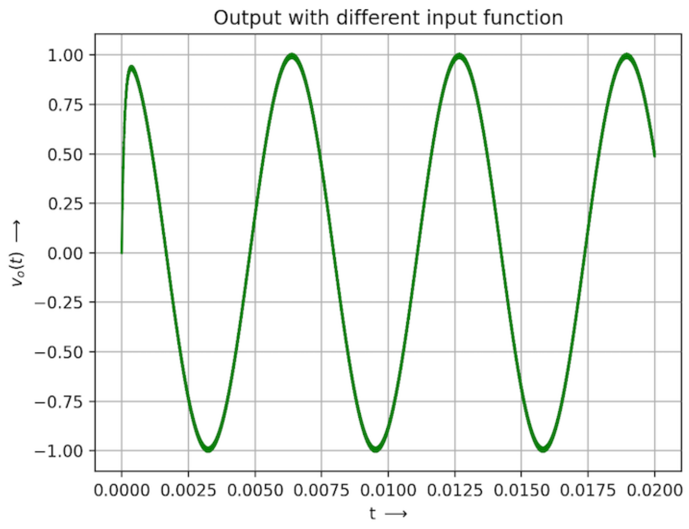

Figure 6: Two port network output for different input function

As seen from the figure it can be noticed that at the beginning there is a sudden increase in the voltage from 0V almost reaching 1V. Since the current through the inductor is zero initially, the inductor would not allow a sudden surge in current. To counter act the effect, the voltage observed at the other end is 1V. Hence the sudden increase in the value of the output voltage.
As seen from the bode plot, the filter only allows signals less than 10^4 rad/s. Hence the high frequency component of the input signal is not observed across the output. The input and output voltages of the same frequency are almost in phase as observed from the phase frequency plot of the transfer function. Another peculiarity is observed when the graph is zoomed in as shown in the figure below.

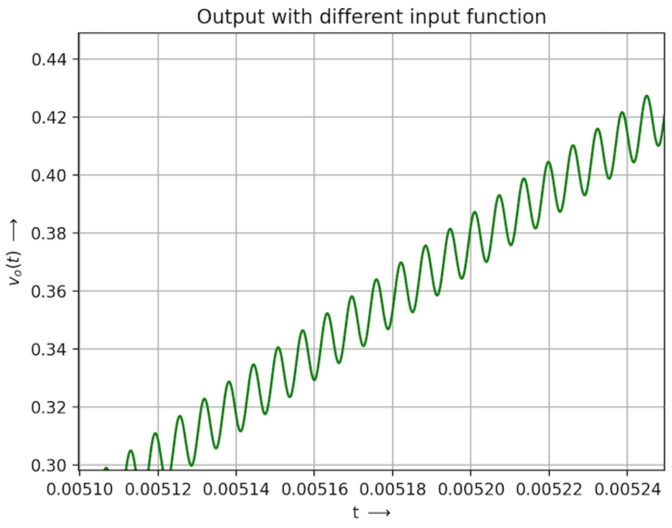

Figure 7: Zoom in of the two port output plot

Clearly it can be seen that the high frequency component, though attenuated had been overlaid over the low frequency part of the input which has been passed through the low pass filter.

Conclusion

From this assignment I understood how some of Python's libraries can be used to advantage by an electrical engineer to get insightful inputs on various real life applications.