EE2703 : Applied Programming Lab
Assignment 6

Rakshit Pandey
EE20B106

March 16, 2022

1. AIM OF THE ASSIGNMENT

In this week's assignment we work with the Linear Time Invariant (LTI) systems. We work with some mechanical cases and in the end we will work with one simple electric RLC filter. The main of this assignment is to work with scipy.signal python library for working with the transfer functions, plotting them and analyzing the different problems.

2. INTRODUCTION TO THE ASSIGNMENT

- We first import the python library scipy.signal as sp and learn to work with it's basic attributes.
- We then begin the analysis of our first LTI system , which is the 1 dimensional spring problem.
- Then, we move analyze the coupled spring problem.
- Finally we will analyze the RLC filter circuit. All the analysis will be done with the help of appropriate curves of function variation as function of time, bode plots and phase plots in the frequency domain.

3. ASSIGNMENT QUESTIONS

Importing important libraries

We first import some important python libraries which will help us in doing the assignment. This is done in python as follows:

```
1  import numpy as np
2  from numpy import *
3  import scipy.signal as sp
4  import matplotlib.pyplot as plt
5  from math import cos,sin,exp
```

1.Question 1

In this problem we were supposed to solve for the time response of a spring satisfying the equation:

$$\ddot{x} + 2.25x = f(t)$$

with x(0) = 0 and $\dot{x}(0) = 0$ for time interval of 0 to 50 seconds. We know that the Laplace transform of $f(t) = cos(\omega t)e^{-\alpha t}u_0 t$ is given by:

$$F(s) = \frac{s + 0.5}{(s + 0.5)^2 + 2.25}$$

For first question, we take ω=1.5 and α=0.5.

If we do computation in Laplace domain then the spring equation gives :

$$X(s) = \frac{s + 0.5}{((s + 0.5)^2 + 2.25)(s^2 + 2.25)}$$

We use the sp.impulse function to compute the inverse Laplace of X(s) to get x(t), then x(t) is plotted against t for 0s < t < 50s. This is executed in python as follows:

Code

```
Num1 = poly1d([1,0.5])
Den1 = polymul([1,0,2.25],[1,1,2.50])
H1 = sp.lti(Num1,Den1)
t1,x1 = sp.impulse(H1,None,linspace(0,50,501))

plt.figure(1)
plt.xlabel("t")
plt.ylabel("x(t)")
plt.title("Time response of a spring for decay = 0.5")
plt.plot(t1,x1)
```

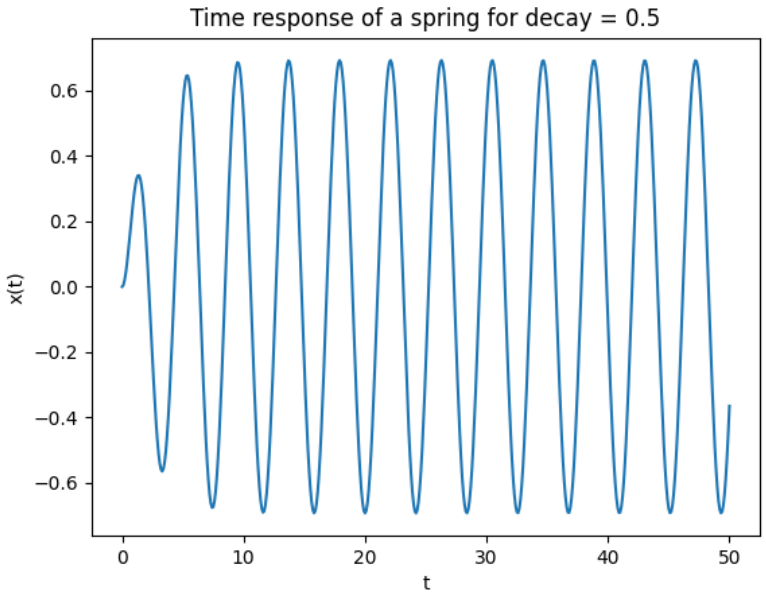

Figure 1: x(t) vs t for decay = 0.5

2. Question 2

We are supposed to solve previous problem but with much smaller decay of 0.05 instead of 0.5.

Code

```
Num2 = poly1d([1,0.05])
Den2 = polymul([1,0,2.25],[1,0.1,2.2525])
H2 = sp.lti(Num2,Den2)
t2,x2 = sp.impulse(H2,None,linspace(0,50,501))

plt.figure(2)
plt.xlabel("t")
plt.ylabel("x(t)")
plt.title("Time response of spring for decay = 0.05")
plt.plot(t2,x2)
```

We obtain the following curve:

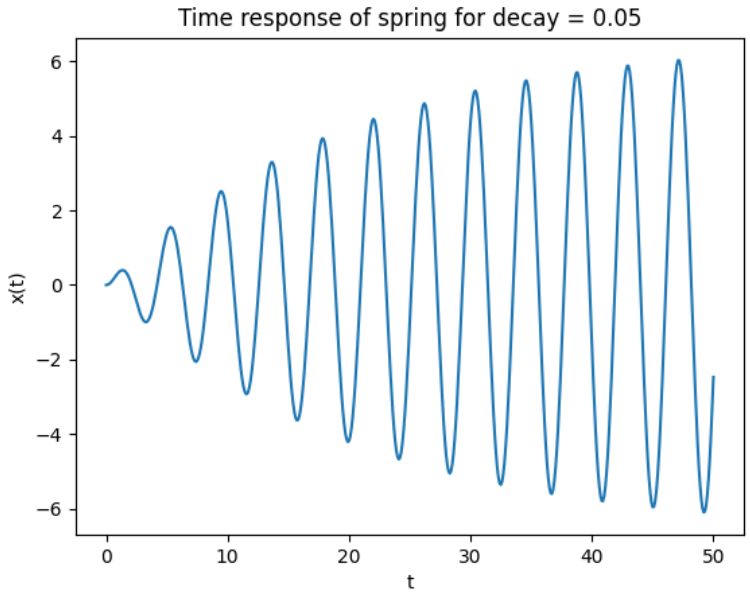

Figure 2: x(t) vs t for decay = 0.05

3. Question 3

In this question we are required to generate the plot of x(t) vs t for 5 different values of ω from 1.40 to 1.60 in the steps of 0.05, while keeping the value of α as 0.05. Here, instead of using sp.impulse we will use sp.lsim to simulate the problem. This is executed in python as follows:

```
    freq = 1.40
    for i in range(5):
      f= []
      F_num = poly1d([1,0.05])
      F_den = polymul([1,0.1,freq**2+0.0025],[1,0,2.25])
      H = sp.lti(F_num,F_den)
      t = linspace(0,100,1001)
      for i in range(len(t)):
        f.append(cos(freq*t[i])*exp(-0.05*t[i]))
      t,x,vec = sp.lsim(H,f,t)
      plt.figure()
      plt.xlabel("t")
      plt.ylabel("x(t)")
      plt.title("Time response of a spring for frequency = {}".format(freq))
      freq += 0.05
      plt.plot(t,x)
      plt.show()
```

We obtain the following 5 curves for 5 different values of ω

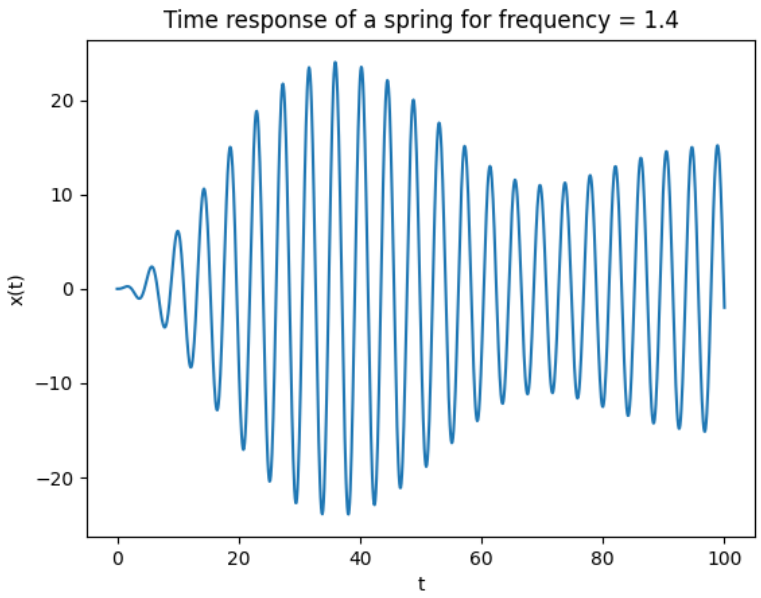

Figure 3: x(t) vs t for ω=1.40

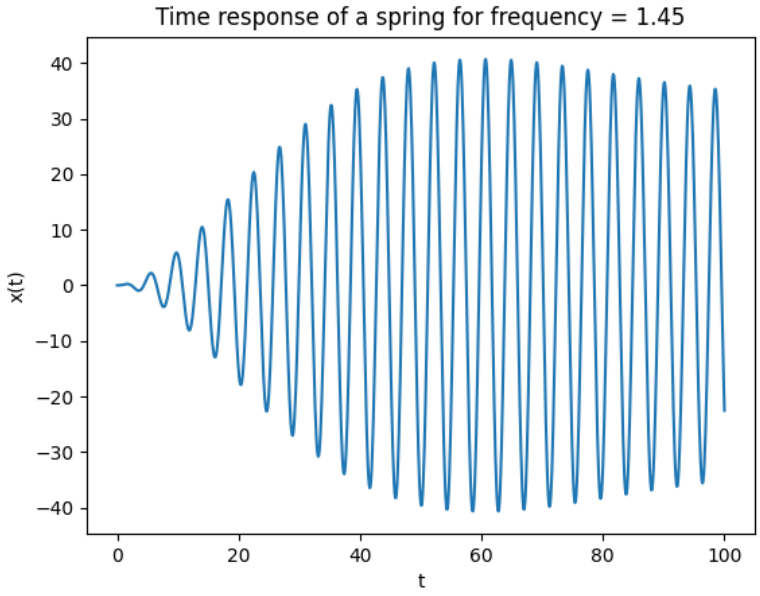

Figure 4: x(t) vs t for ω=1.45

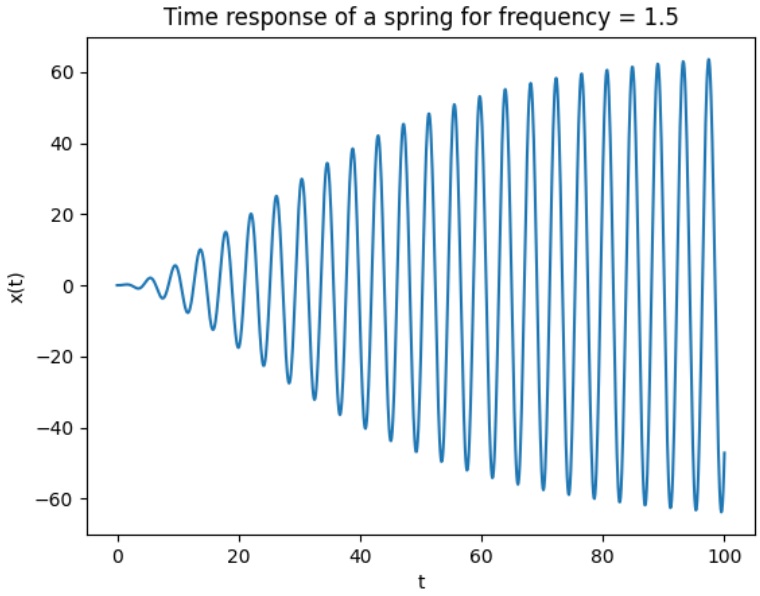

Figure 5: x(t) vs t for ω=1.50

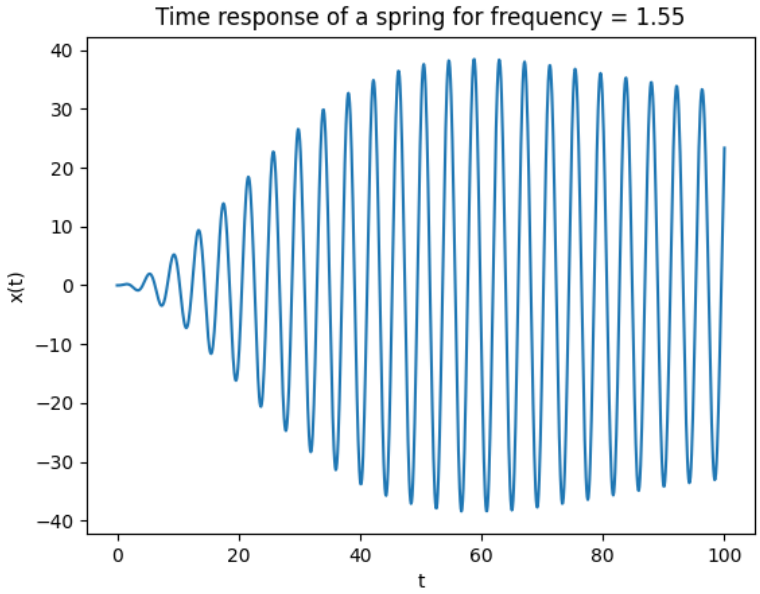

Figure 6: x(t) vs t for ω=1.55

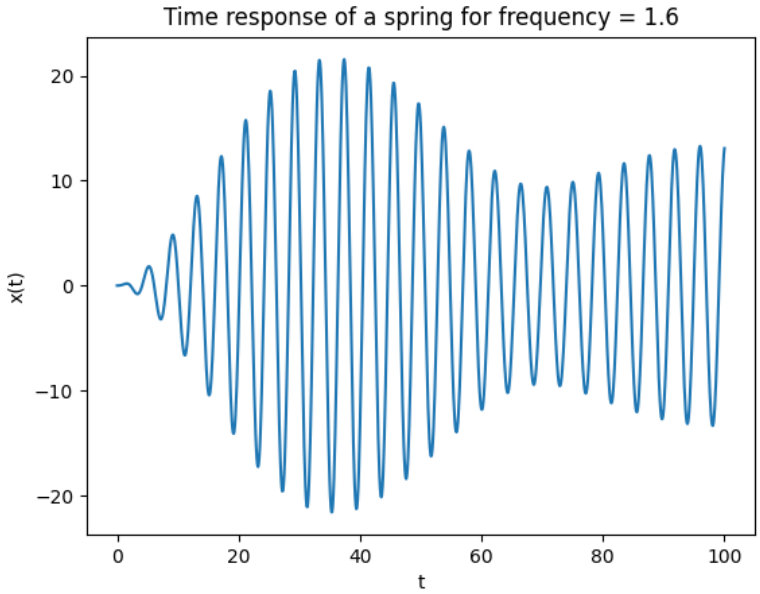

Figure 7: x(t) vs t for ω=1.60

The frequency $\omega = 1.50$ is the natural frequency of the system, hence at this frequency the resonance occurs and we see the maximum amplitude.

4. Question 4

In this question we were supposed to solve for the given coupled spring problem:

$$\ddot{x} + (x - y) = 0$$

$$\ddot{y} + 2(y - x) = 0$$

with the given initial conditions:

$$x(0) = 1, \dot{x}(0) = y(0) = \dot{y}(0) = 0$$

If we solve the the given system of two equations in Laplace domain with the given initial conditions, then we get X(s) and Y(s), as follows:

$$X(s) = \frac{s^2 + 2}{s^3 + 3s}$$

$$Y(s) = \frac{2}{s^3 + 3s}$$

Now, we are supposed to obtain x(t) and y(t) and plot them for 0<t<20s, In order to inverse Laplace transforms for X(s) and Y(s) we use sp.impulse, this is executed in python as follows:

```
num3 = poly1d([1,0,2])
den3 = poly1d([1,0,3,0])
H3 = sp.lti(num3,den3)
t3,x3 = sp.impulse(H3,None,linspace(0,20,201))

num4 = poly1d([2])
den4 = poly1d([1,0,3,0])
H4 = sp.lti(num4,den4)
t4,y4 = sp.impulse(H4,None,linspace(0,20,201))

plt.figure(8)
plt.xlabel("t")
plt.ylabel("x(t) and y(t)")
plt.title("Coupled spring oscillation")
plt.plot(t3,x3,label="x(t)")
plt.plot(t4,y4,label="y(t)")
plt.legend(loc="upper right")
```

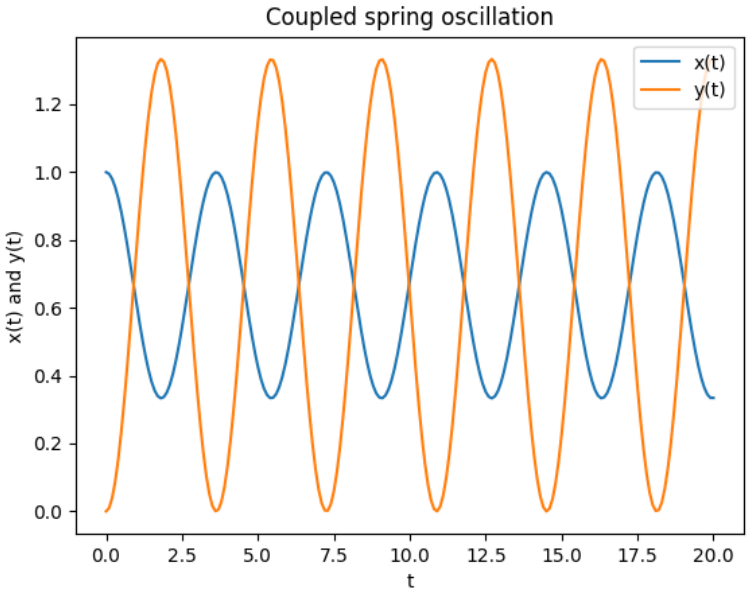

Figure 8: coupled spring

5. Question 5

In this question we are supposed to find the magnitude and phase plot of the steady state transfer funciton for the given RLC low pass filter:

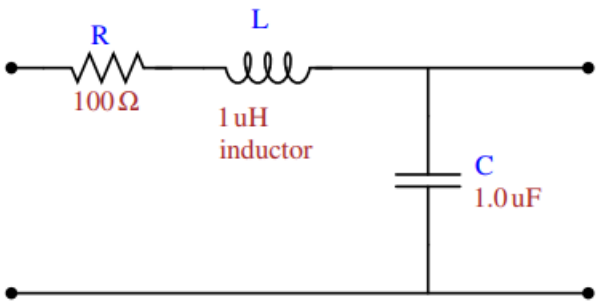

Figure 9: Low pass RLC filter

This is achieved in python as follows:

```
1    R = 100
2    L = 1e-6
3    C = 1e-6
4    s_2 = L*C
5    s_1 = R*C
```

```
6
7       num5 = poly1d([1])
8       den5 = poly1d([s_2,s_1,1])
9       H5 = sp.lti(num5,den5)
10      w,S,phi = H5.bode()
11
12      t_30 = linspace(0,30e-6,10000)
13
14      plt.figure(9)
15      plt.xlabel("Frequency")
16      plt.ylabel("Magnitude of transfer function")
17      plt.title("Magnitude Response")
18      plt.semilogx(w,S)
19      plt.grid()
20
21      plt.figure(10)
22      plt.xlabel("Frequency")
23      plt.ylabel("Phase of transfer function")
24      plt.title("Phase Response")
25      plt.semilogx(w,phi)
26      plt.grid()
```

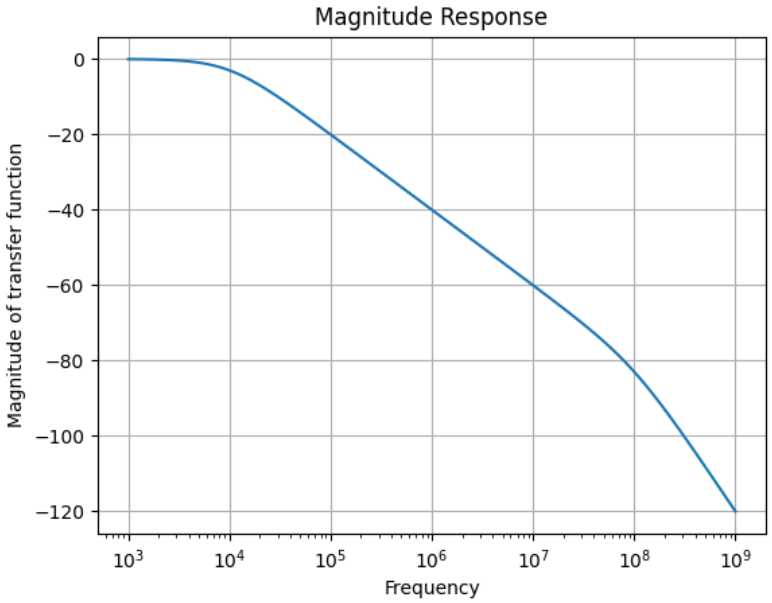

Figure 10: Magnitude response of transfer function

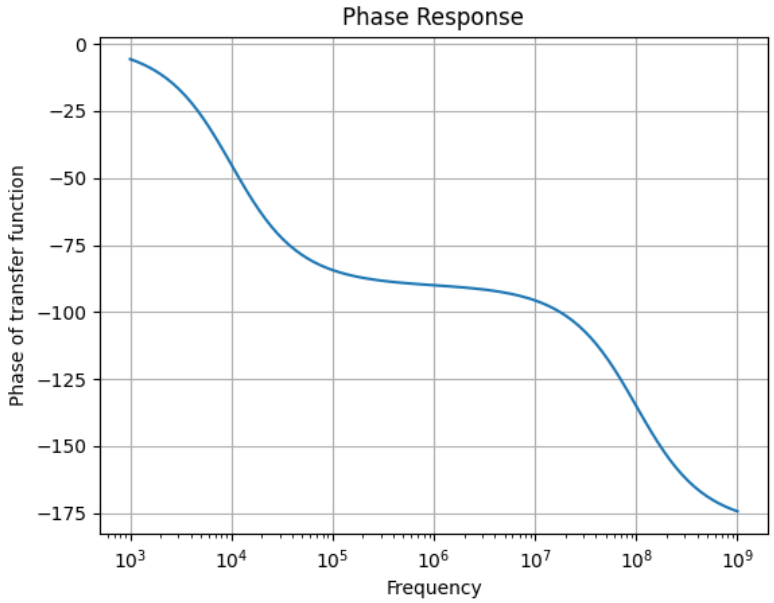

Figure 11: Phase response of transfer function

6. Question 6

Let us consider that the applied input voltage is $V_i(t)$, given by:

$$V_i(t) = cos(10^3 t)u(t) - cos(10^6 t)u(t)$$

Now we will obtain the output voltage V_0 function by doing the convolution of V_i(t) with the impulse response of system h(t). This is done for two cases, one the plot of V_0(t) vs t for 0<t<30μs. The other case is when the time is taken from 0 to 30ms. We do this in python with the help of sp.lsim function as shown:

```
t_30m = np.linspace(0,30e-3,10000)

Vi = []
Vi2 = []
for i in range(len(t_30)):
  Vi.append(cos(1e3*t_30[i]) - cos(1e6*t_30[i]))

for i in range(len(t_30m)):
  Vi2.append(cos(1e3*t_30m[i]) - cos(1e6*t_30m[i]))

t5,V0,vec2 = sp.lsim(H5,Vi,t_30)

t6,V02,vec3 = sp.lsim(H5,Vi2,t_30m)

plt.figure(11)
plt.xlabel("t")
plt.ylabel("$V_{0}(t)$")
```

```
    plt.title("Output signal for 0 < t < 30$\mu$s")
    plt.plot(t5,V0)
    plt.grid()

    plt.figure(12)
    plt.xlabel("t")
    plt.ylabel("$V_{0}(t)$")
    plt.title("Output signal for 0 < t < 30ms")
    plt.plot(t6,V02)
    plt.grid()
```

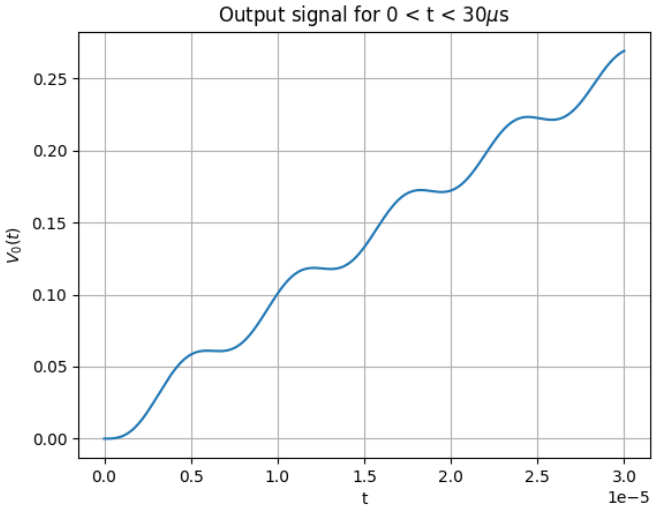

Figure 12: The plot of output voltage vs t for 0<t<30μs

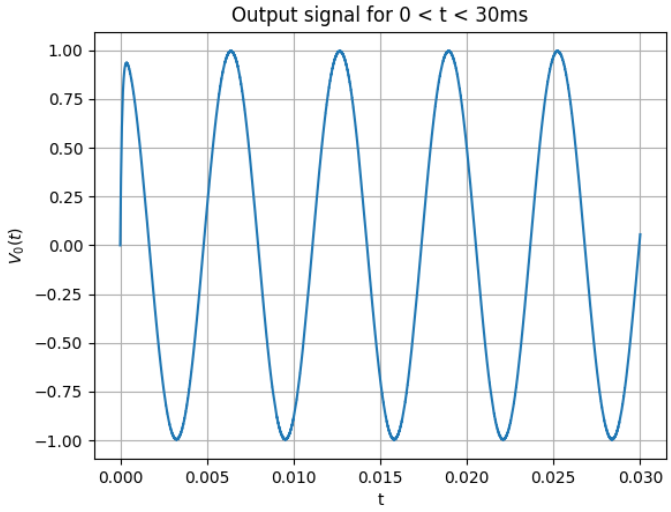

Figure 13: The plot of output voltage vs t for 0<t<30ms

We can clearly see that the given system is an RLC low pass filter of second order. When we do small time interval plot the high frequency components can be seen as the ripples in the small

time interval. These components are highly attenuated hence they don't appear in the large time interval plot.Also we can clearly see that in large time interval plot the low frequency plots almost pass unaltered hence further consolidating the fact that the given RLC circuit is the low pass filter circuit.

4.CONCULUSIONS

The analysis of LTI systems (both mechanical and electrical) was done with the help of scipy signals toolkit. Specifically we analyzed forced oscillatory system (single spring and coupled spring problems) and electric RLC low pass filter. We made Bode plots to analyze the transfer functions, both magnitude as well as phase of the transfer function was realized with the help of scipy.signal library.