EE2703: Applied Programming Lab
Assignment 6B
Laplace Tranform

Devaganthan S S
EE19B018

June 9, 2021

0.1 Abstract

This assignment aims to analyze "Linear Time-Invariant Systems" with numerical tools in python.

0.2 Introduction

LTI systems are an important part of Electrical Engineering. Python has a Signals toolbox, which enables engineers to analyze LTI systems efficiently. In this assignment, we take different problems and use the functions such as **sp.lti , sp.impulse , sp.lsim** to solve them.

0.3 Results and Implementation

0.3.1 Spring Mass System:

We apply a time varying (Sinusoidal and Decaying) force to a spring mass system. We need to find position of the mass (x) as function of time. We take the Laplace form of the given differential equation, to find the Laplace Transform of x(t). Using **sp.lti()**, we find x(t). The below code accomplishes the above tasks, and plots x(t) vs time graph

```
def impulseResponsePlot(H,title):
    t,x = sp.impulse(H,None,linspace(0,50,5001)) #Computes x(t)
    plt.plot(t,x)
    plt.title(title)
    plt.xlabel('Time')
    plt.ylabel('x')
    plt.show()

#Q1
p1 = poly1d([1,0.5])
p2 = polymul([1,1,2.5],[1,0,2.25])
H = sp.lti(list(p1),list(p2)) # X(s)
impulseResponsePlot(H,'x(t) Plot for decay = 0.5')
```

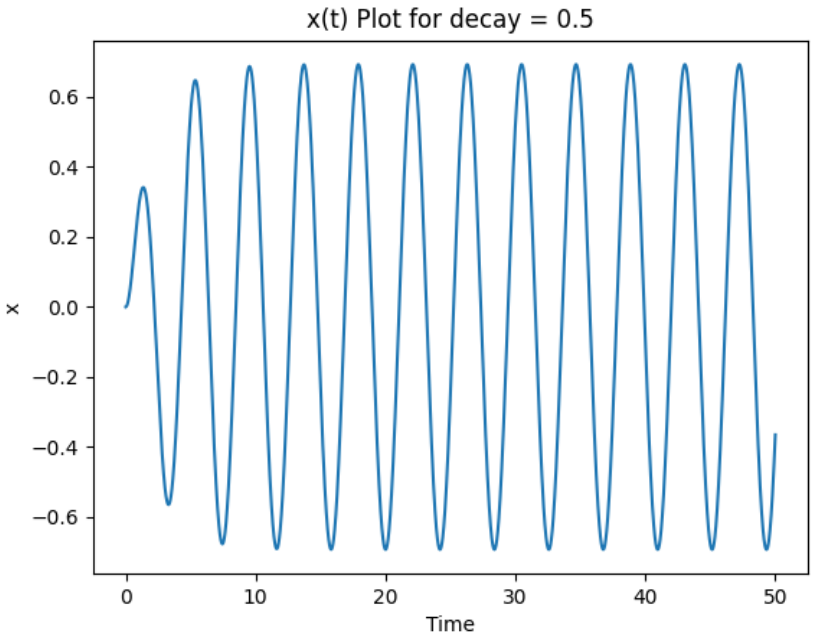

Figure 1: Position of mass as function of time

0.3.2 With a Smaller Decay

We repeat the same as before with a different decay rate. The below code accomplishes the above tasks, and plots the new x(t) vs time graph.

```
#Q2
p1 = poly1d([1,0.05])
p2 = polymul([1,0.1,2.2525],[1,0,2.25])
H = sp.lti(list(p1),list(p2)) #X(s)
impulseResponsePlot(H,'x(t) Plot for decay = 0.05')
```

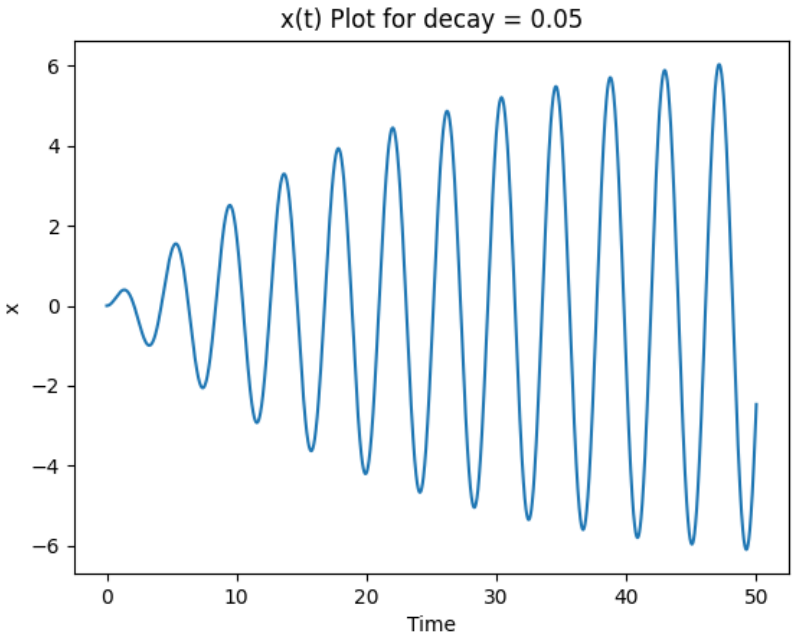

Figure 2: Position of mass as function of time

0.3.3 Varying the Frequency of the Sinusoid:

We consider the problem to be an LTI system with f(t) as the Input and x(t) as the output.We vary the Frequency of the Sinusoid from 1.4 to 1.6 in steps of 0.05, to see how x(t) varies. We use the function **sp.lsim** to compute the output. The amplitude of the output is maximum at its Natural Frequency of 1.5. The below code accomplishes the above tasks and plots the frequency variation of the Ouptut.

```
#Q3
H = sp.lti(1,[1,0,2.25]) #X(s)/F(s)
t = linspace(0,100,5001)
for f in arange(1.4,1.65,0.05):
    u = cos(f*t)*exp(-0.05*t)
    t,y,svec = sp.lsim(H,u,t) #y is the Output
    plt.plot(t,y)
plt.legend(['freq=1.4','freq=1.45','freq=1.5','freq=1.55','freq
    =1.6'])
plt.title('Output Responses for Input of different Frequencies')
plt.xlabel('Time')
plt.ylabel('Output')
plt.show()
```

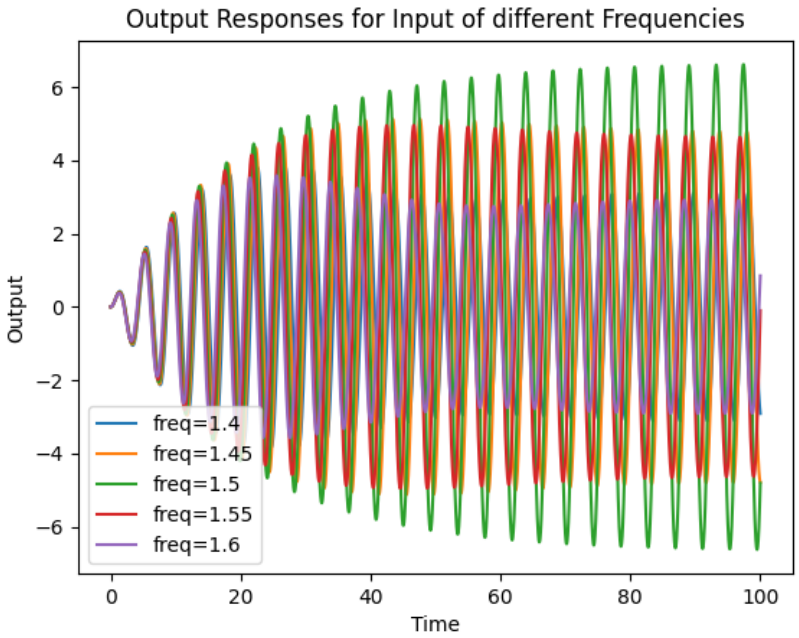

Figure 3

0.3.4 Coupled Spring Problem

From the given Differential Equations and Initial Conditions, we find,

$$X(s) = \frac{s^2 + 2}{s^3 + 3s} \tag{1}$$

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

We find x(t) and y(t) using the function **sp.lsim()**. The below code accomplishes the above tasks and plots x(t) and y(t) as function of time.

```
#Q4
X = sp.lti([1,0,2],[1,0,3,0]) #X(s)
t,x = sp.impulse(X,None,linspace(0,20,5001)) # x is x(t)
plt.plot(t,x,label = 'x(t)')
Y = sp.lti(2,[1,0,3,0]) #Y(s)
t,y = sp.impulse(Y,None,linspace(0,20,5001)) # y is y(t)
plt.plot(t,y,label = 'y(t)',color = 'y')
plt.legend()
plt.xlabel('Time')
plt.ylabel('x(t)/y(t)')
plt.title('Solution of the Coupled Spring Problem')
plt.show()
```

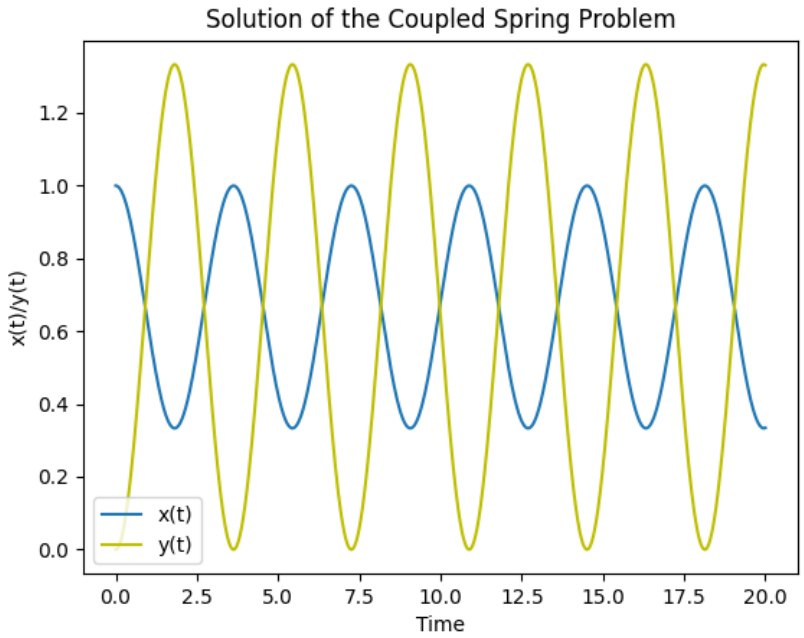

Figure 4

0.3.5 Two Port Network:

The transfer Function of the two port network is,

$$H(s) = \frac{1}{LCs^2 + RCs + 1} \tag{3}$$

We use the function **bode()** and **sp.lti()** to compute values of the magnitude and phase for different frequencies. The below code accomplishes the above tasks and plots the Bode plot for the Transfer Function found.

```
#Q5
R = 100
L = 1e-6
C = 1e-6
H = sp.lti([1],[L*C,C*R,1]) # Transfer Function of the Two Port
    Network
w,S,phi = H.bode() # Computes the Value of Mag and phase
plt.subplot(2,1,1)
plt.title('Bode Plot for the RLC series circuit')
plt.semilogx(w,S)
plt.xlabel('Frequency')
plt.ylabel('Magnitude in Db scale')
plt.subplot(2,1,2)
plt.semilogx(w,phi)
plt.xlabel('Frequency')
plt.ylabel('Phase')
plt.show()
```

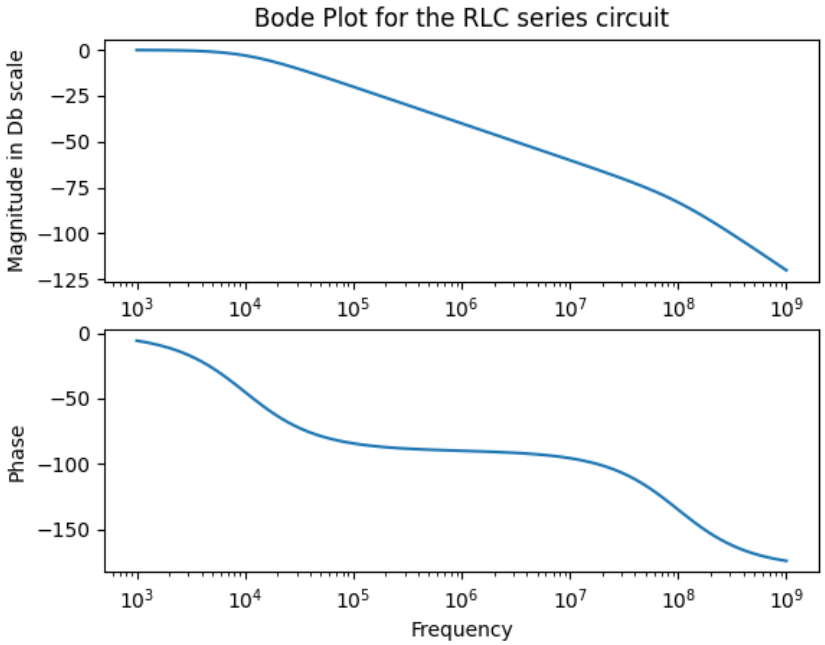

Figure 5

0.3.6 Finding the Output for the given input for the Two Port System:

The Output to the given Input can be found using the **sp.lsim()** function. We plot the output over both small and large time scale to better understand the variations. The below code accomplishes the above tasks.

```
def outputPlotter(H,u,t,title):
    t,y,svec = sp.lsim(H,u,t)
    plt.plot(t,y)
    plt.title(title)
    plt.xlabel('Time')
    plt.ylabel('Output')
    plt.show()

#Q6
t = arange(0,3e-2,1e-7) #Large Time Scale
u = cos((1e3)*t)-cos((1e6)*t)
outputPlotter(H,u,t,'Output plot (Large time scale)')
```

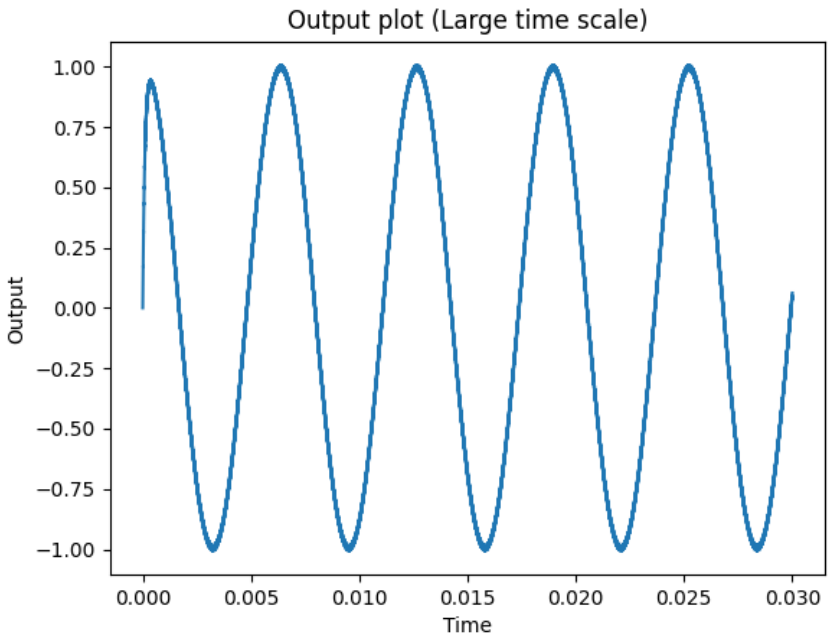

Figure 6

```
t = arange(0,30e-6,1e-7) #Large Time Scale
u = cos((1e3)*t)-cos((1e6)*t)
outputPlotter(H,u,t,'Output Plot (Small time scale)')
```

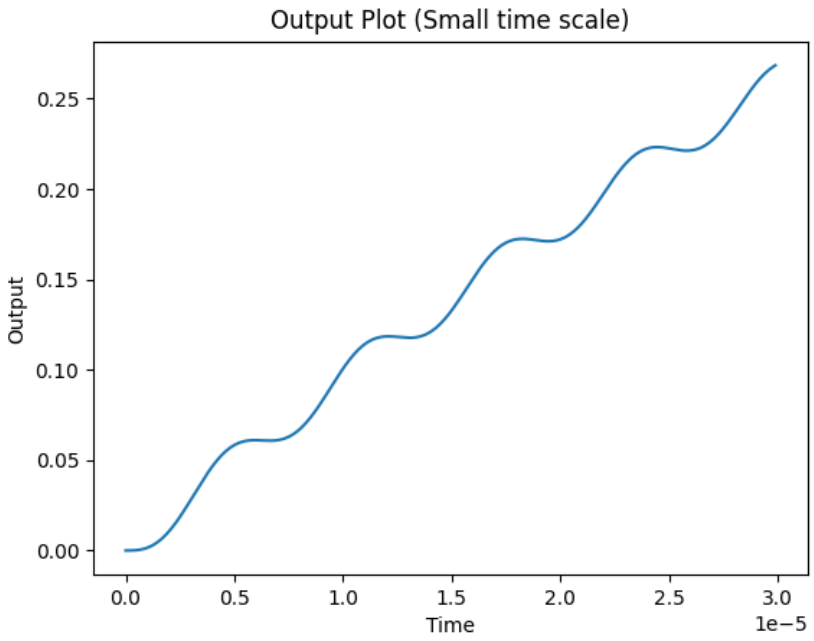

Figure 7

0.4 Conclusion

We have Analyzed "Linear Time-Invariant systems" with Numerical tools in Python. We learnt how to use functions like sp.lti(), sp.impulse(), sp.lsim().