EE2703 : Applied Programming Lab
Assignment 6

Andapally Snehitha
EE20B008

March 27, 2022

Aim:

Most general systems around us can be modeled as Linear Time-Invariant systems, and are widely used in the field of Electrical Engineering. The objective of this assignment is to:

1. Analyse continuous time LTI systems in Laplace domain.
2. Solve Linear Constant Coefficient Differential Equations (LCCDE) in Laplace domain using signal toolbox in scipy library.
3. Explore functions in Time and Laplace domains.

Part-I

The time response of a lossless spring system is given by:

$$x(t) + 2.25x(t) = f(t)$$

where $f(t)$ is the forced input on the spring system.
Suppose if the forced input $f(t)$ is a decaying sinusoidal force as given by:

$$f(t) = cos(1.5t)e^{-0.5t}u(t)$$

In Laplace domain:

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

with $x(0) = 0$ and $x = 0$ input conditions. This corresponds to:

$$F(s) = s^2X(s) + 2.25X(s)$$

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

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

When the damping coefficient is 0.05

$$f(t) = cos(1.5t)e^{-0.05t}u(t)$$

$$F(s) = \frac{s + 0.05}{(s + 0.05)^5 + 2.25}$$

$$X(s) = \frac{s + 0.05}{(s^2 + 2.25)((s + 0.05)^2 + 2.25)}$$

Now, we use Scipy's *impulse*() function to get the time-domain form of $X(s)$. The python code for the same is as follows:

```
# defining function for showing the impulse response plot def ForcedOscillation(a,n):
  t = np.linspace(0,50,1000)      #Time vector going from 0 to 50 seconds
  # F(s) = (s + a)/((s+a)^2 + 2.25)
  X = sp.lti([1,a],np.polymul([1,0,2.25], np.polyadd(np.polymul([1,a],[1,a]),[2.25])))     #Laplace domain expression for X(s)
  t, x = sp.impulse(X, None, t)   # Time domain function values
    figure(n)   #code for plotting the figure
  title("x(t) time domain")
  ylabel("$x(t)\\rightarrow$")
  xlabel("$t$ (in sec)$\\rightarrow$")
  plot(t, x, label = "Decay = " + str(a))
  grid(True)
  legend()
  show()
```

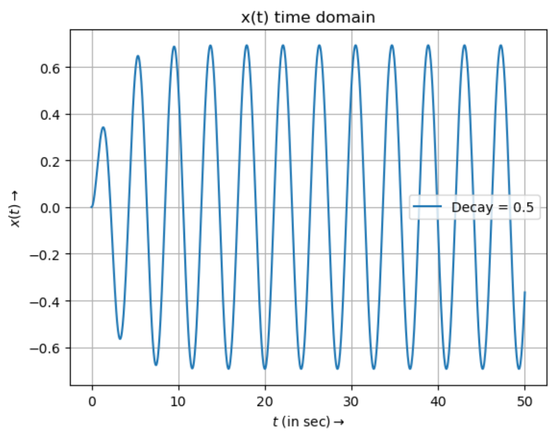

Figure 1: X(t) in time domain for decay coefficient 0.5

We can see that the output of the system when the decay coefficient = 0.05 has higher amplitude - since the energy supplied by the forced input in case of decay coefficient = 0.05 will be higher compared to decay coefficient

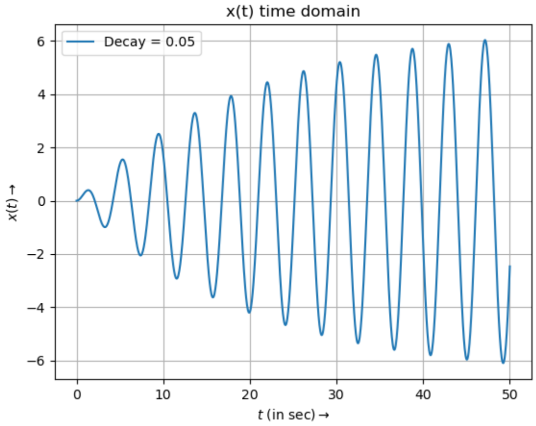

Figure 2: X(t) in time domain for decay coefficient 0.05

= 0.5. The is because the output with decay coefficient 0.05 decays slower compared to the case when decay coefficient is 0.5.

If the decay coefficient = 0, then the forced input with oscillation frequency $\omega = 1.5$ resonates with the natural frequency of the system, and hence will blow up the output.
Now, we plot the output for various frequencies:

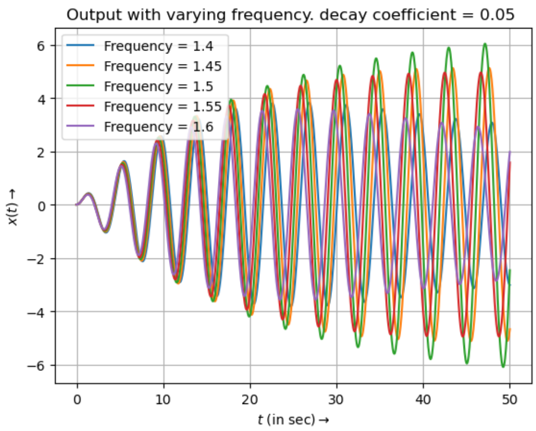

Figure 3: X(t) for various frequencies

We can see clearly that the output corresponding to frequency of 1.5 has the maximum amplitude.

Part-II

We try to solve the following coupled spring problem with:

$$x + (x - y) = 0$$

$$y + 2(y - x) = 0$$

with initial conditions $x(0) = 1, x(0) = y(0) = y(0) = 0$.
Solving further:

$$s^2X(s) - sx(0^-) - x(0^-) = Y(s)$$

$$s^2Y(s) - sy(0-) - y(0-) + 2Y(s) = X(s)$$

Substituting and solving further, we arrive at

$$X(s) = \frac{(0.5s^2+1)s}{(s^2+1)(0.5s^2+1) - 1}$$

$$Y(s) = \frac{s}{(s^2+1)(0.5s^2+1) - 1}$$

The Python code for plotting $x(t)$ and $y(t)$ is as follows:

```python
t = np.linspace(0, 20, 1000)     #Time vector going from 0 to 20 seconds

# Transfer functions for x and y
X = sp.lti(np.polymul([1, 0], [0.5, 0, 1]), np.polyadd(np.polymul([1, 0, 1], [0.5, 0, 1]), [-1])) Y = sp.lti([1, 0], np.polyadd(np.polymul([1, 0, 1], [0.5, 0, 1]), [-1]))

#Time domain function values of X(s) and Y(s)
t, x = sp.impulse(X, None, t)
t, y = sp.impulse(Y, None, t)

#Plotting functions
figure(4)
plot(t, x, label="x(t)")
plot(t, y, label="y(t)")
title("x(t) & y(t) in time domain")
ylabel("$Signal\\rightarrow$")
xlabel("$t$ (in sec)$\\rightarrow$")
legend()
grid(True)
show()
```

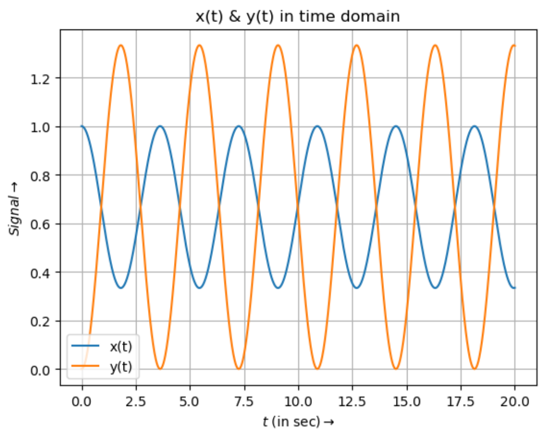

Figure 4: x(t) and y(t) verses time

Part-III

Our objective is to find the Magnitude and Phase response of the Transfer function of the RLC circuit given below:

Figure_f5.png

On doing simple circuit analysis, we can see that the transfer function is:

$$H(s) = \frac{1}{1 + RCs + LCs^2} = \frac{1}{1 + 10^{-4}s + 10^{-12}s^2}$$

The Magnitude and Phase plots can be obtained as follows:

```
# Transfer functions for H(s)
H = sp.lti([1], [1e-12, 1e-4, 1])

#Obtaining bode plot
w, S, phi = H.bode()
```

There are two major bending sections in both graphs indicating the two poles of the system.

The magnitude plot remains constant until it sees a pole after which it decreases at $-20dB/decade$. After encountering the second pole, its slope goes down to $-40dB/decade$. There will be a 90^ophase drop at each pole.

Part-IV

Now, a input signal of V_{in} is given, where V_{in} is given by:

$$V_{in}(t) = cos(10^3t)u(t) - cos(10^6t)u(t)$$

$V_{out}$of the system is obtained by using the following Python code:

```
def LCRresponse(t,n):
  #LCR response
  Vin = np.cos(1e3*t)-np.cos(1e6*t)   # Input signal
```

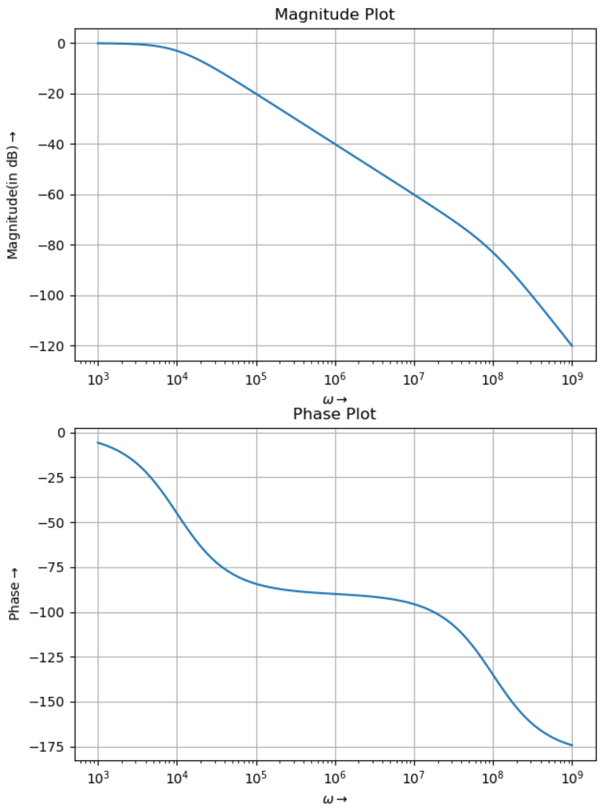

Figure 5: Magnitude & Phase Plots of $H(s)$

```
    t, Vout, svec = sp.lsim(H, Vin, t)  # Convolution for initial response

    #Plotting functions
    figure(n)
    plot(t, Vout)
    title("$V_{out}$ vs $t$ (Initial response)")
    xlabel("$t$ (in sec)$\\rightarrow$")
    ylabel("$V_{out}\\rightarrow$")
    grid(True)
    show()

#Intitial short term response
t = np.linspace(0, 30e-6, 10000)
LCRresponse(t,6)
```

```
#Long term response
t = np.linspace(0, 1e-2, 10000)
LCRresponse(t,7)
```

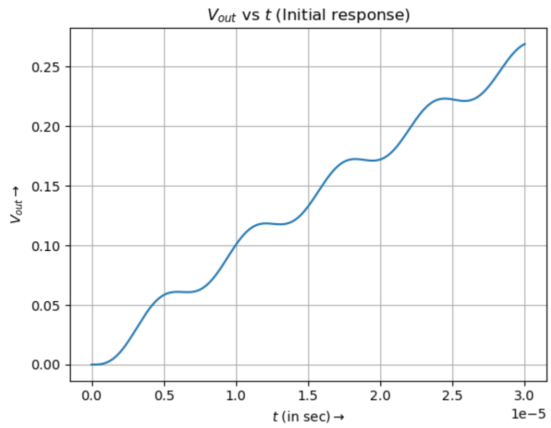

Figure 6: V_{out}vs t (transient/initial)

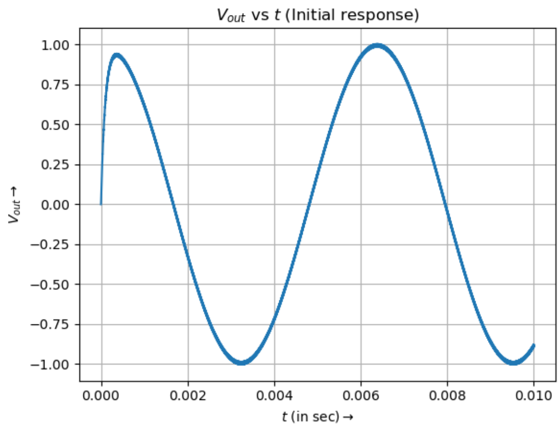

Figure 7: V_{out} vs t (steady state)

From the graphs, we can see that the output of the system is almost a sinusoidal signal with frequency 10^3 rad/s. This is because the system is a Low pass Filter and higher frequencies can be seen as a noise.

Conclusion

1. We explored solving Laplace Equation of various LTI systems, such as spring system, coupled spring problem, and a RLC network, using scipy's signal toolbox.
2. We observed that when the forced input operates at a frequency close to the natural frequency it blows up the output.
3. We also found how a RLC circuit can be used as a low pass filter.