Assignment 6L: The Laplace Equation

Saurav Sachin Kale, EE19B141

April 18, 2021

1 Aim

- Analysis of continuous time linear time invariant (LTI) systems in the Laplace domain
- Exploring SciPy functions for Impulse Response, Convolution, and Bode Plots
- Solving simple linear constant coefficient differential equations with given initial conditions in Laplace Domain

2 Procedure

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

For damping factor $a \in \mathbb{R}$We have been given the function $f(t)$ as follows:

$$f(t) = cos(1.5t)e^{-at}u_0(t)$$

Taking the laplace transform, we obtain the expression

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

Now the given differential equation for the system is

$$\ddot{x} + 2.25x = f(t) \tag{1}$$

With the initial conditions

$$x(0^-) = \dot{x}(0^-) = 0$$

Taking unilateral laplace transform, we get

$$s^2X(s) - sx(0^-) - \dot{x}(0^-) + 2.25X(s) = F(s)$$

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

$$H(s) = \frac{X(s)}{F(s)} = \frac{1}{s^2 + 2.25}$$

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

We now analyse the system for values of a

- We now take the case of $a = 0.5$. We feed the transfer function $X(s)$ into a variable, and use the `sp.impulse()` function to obtain the time domain function $x(t)$, and plot it. We do the same for $a = 0.05$.

```
# define the time vector
t = np.linspace(0, 50, 1000)

# we write the overall laplace domain expression for X(s)
X = sp.lti([1, 0.5], np.polymul([1, 0, 2.25],
        np.polyadd(np.polymul([1, 0.5], [1, 0.5]), [2.25])))

# get the time domain function
t, x = sp.impulse(X, None, t)
```

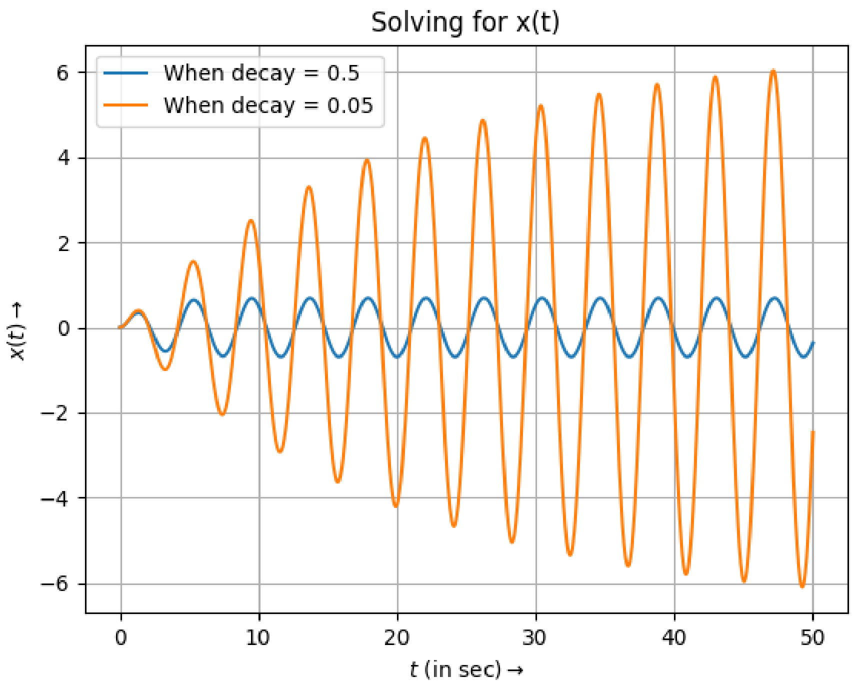


Figure 1: x(t) plotted vs. time for t = 0 to 50 seconds taking $a = 0.5$ and $a = 0.05$

- From the graph we observe that this is the response of a lossess oscillator driven by a decaying sinusoidal force. If the force decays slowly ($a = 0.05$), it imparts more momentum to the oscillator before decaying. Therefore, the response of the oscillator is of higher magnitude, as is seen in the graph.
- We now plot $x(t)$ for varying angular frequency from 1.4 to 1.6 in steps of 0.05 for $a = 0.05$. We get the following graphs.

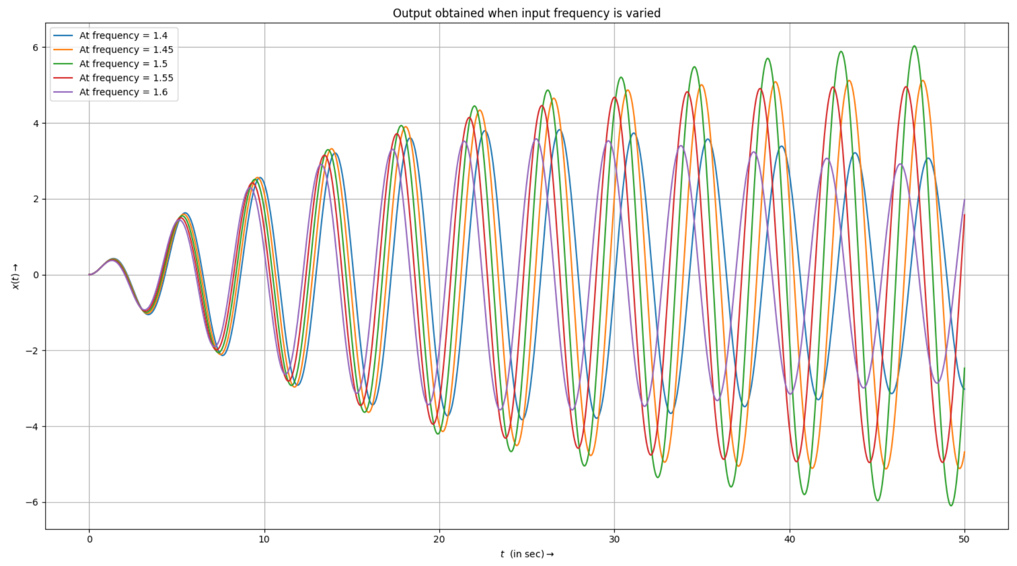


Figure 2: Variation in the output for various frequency values of the input force from t = 0 to 50 seconds

- From the graph we notice that the green curve corresponding to $\omega = 1.5$ has the highest amplitude, and frequencies both above and below 1.5 have lower amplitudes. This suggests that $\omega = 1.5$ is closest to the natural frequency of the system. We see that the frequency response function $H(j\omega)$ has a double pole at $\omega = 1.5$, hence the amplitude rises as we move near it.

2.2 Coupled Spring Problem

Given to us is a pair of coupled linear constant coefficient differential equations

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

$$\ddot{y} + 2(y - x) = 0 \tag{3}$$

The initial conditions being:

$$x(0^-) = 1$$

$$\dot{x}(0^-) = y(0^-) = \dot{y}(0^-) = 0$$

If we take unilateral laplace transform, and apply the initial conditions we get:

$$s^2X(s) - sx(0^-) - \dot{x}(0^-) = Y(s)$$

$$(s^2 + 1)X(s) - s = Y(s) \tag{4}$$

$$s^2Y(s) - sy(0^-) - \dot{y}(0^-) + 2Y(s) = 2X(s)$$

$$(0.5s^2 + 1)Y(s) = X(s) \tag{5}$$

Solving (4) and (5) we obtain the laplace expressions for X and Y:

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

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

- We can now plug these two into our code and use the `sp.impulse()` function to obtain the time domain function $x(t)$ and $y(t)$.

```
# find the transfer function for X and Y
Y = sp.lti([1, 0], np.polyadd(np.polymul([1, 0, 1], [0.5, 0, 1]), [-1]))
X = sp.lti(np.polymul([1, 0], [0.5, 0, 1]),
        np.polyadd(np.polymul([1, 0, 1], [0.5, 0, 1]), [-1]))

# get the time domain function
t, y = sp.impulse(Y, None, t)
t, x = sp.impulse(X, None, t)
```

- Plotting $x(t)$ and $y(t)$ gives the following results:

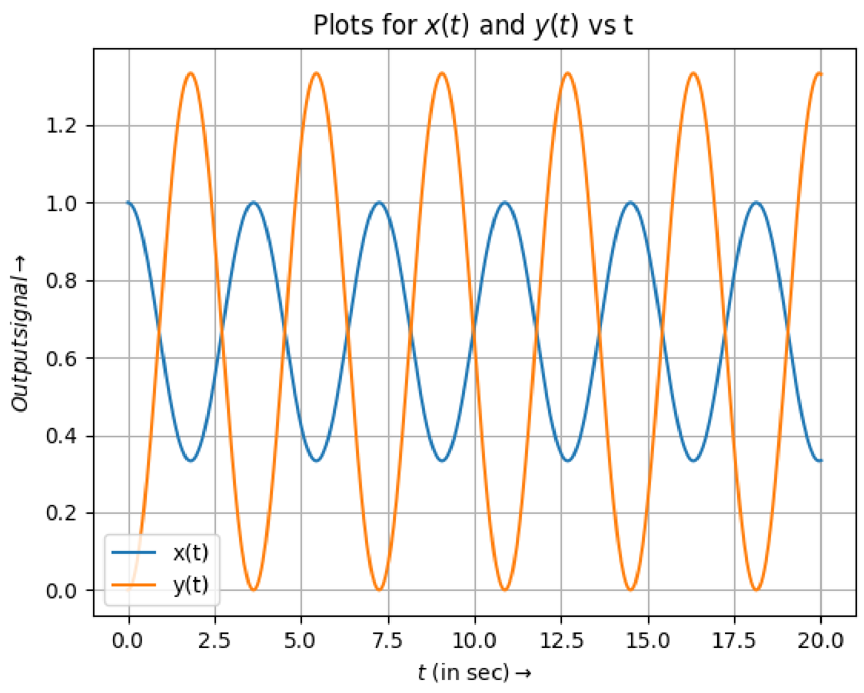
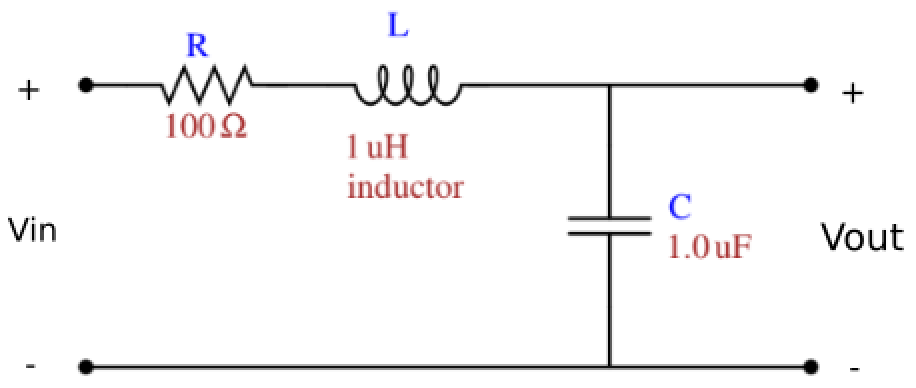


Figure 3: Plot of the signals x(t) and y(t) from 0 to 20 seconds

- This system behaves like an amplifier. Negative feedback is used to make sure that the amplitude of $y(t)$ is twice that of $x(t)$.

2.3 Two port network

We are given a two port network whose schematic is shown below



We have to find the Magnitude and Phase response of this system.
Converting to laplace domain, we obtain the following impedances assuming initial conditions are all zero, since initially, V_{in} is zero according to the given function, and there is a RLC network, so there can be no initial energy stored in inductor or capacitor since it is dissipated by the resistor as heat:

$$Z_R = R$$

$$Z_L = sL$$

$$Z_C = \frac{1}{sC}$$

Now, we use Voltage Divider concept to find the value of V_{out} in terms of V_{in}, and $\frac{V_{out}}{V_{in}}$ is the system transfer function:

$$H(s) = \frac{V_{out}}{V_{in}} = \frac{1}{1 + RCs + LCs^2} \tag{6}$$

$$H(s) = \frac{V_{out}}{V_{in}} = \frac{1}{1 + 10^{-4}s + 10^{-12}s^2}$$

- We now feed this expression into our code and use the `sp.bode()` function to get the magnitude and phase response of this system:

```
#define the transfer function
H = sp.lti([1], [1e-12, 1e-4, 1])

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

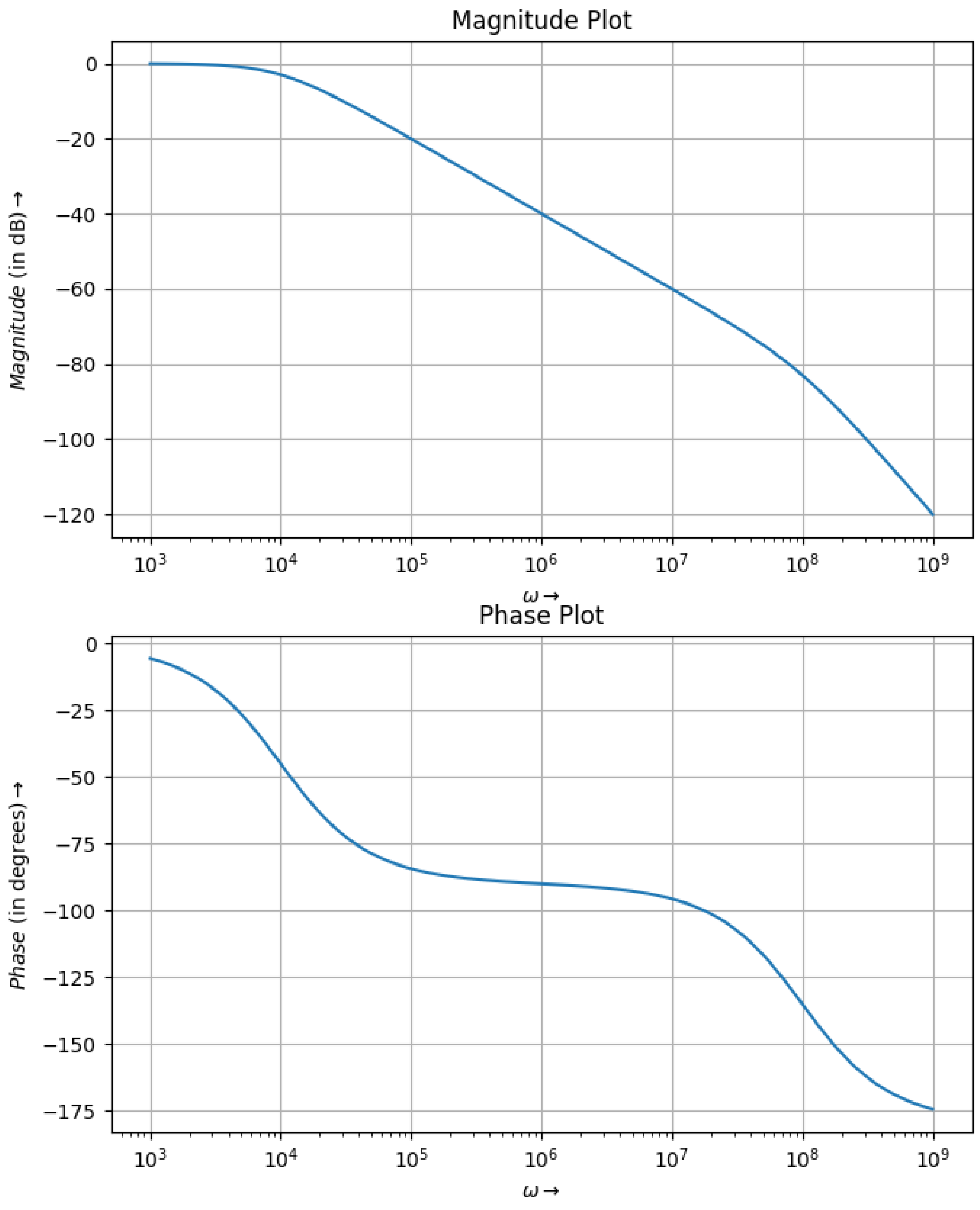


Figure 4: Magnitude and Phase Response

- `w` is frequency ω over which the function is going to be plotted, `S` is the magnitude response, `phi` is the phase response. We now plot them.

- The graph for magnitude response shows two bends where the slope changes. After the first pole slope is roughly -20dB/dec and after the second pole it is -40dB/dec This is expected since a second order transfer function has two poles. Similarly each pole contributes a phase angle of 90^o. We can see the phase moves to -90^oafter the first pole and near -180^0 for the second pole.

- We are now given the input

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

- We first obtain the output by first defining the time step, and then using `sp.lsim()` to calculate both the long term (till $10ms$) and short term (till $30\mu s$) response:

  ```
  #define the transfer function
  H = sp.lti([1], [1e-12, 1e-4, 1])

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

  #define input for short response
  vi = np.cos(1e3*t)-np.cos(1e6*t)

  #perform convolution
  t, vo, svec = sp.lsim(H, vi, t)

  #plotting code omitted, for that check actual code

  # define long term time vector
  t = np.linspace(0, 1e-2, 10000)

  #define input for short response
  vi = np.cos(1e3*t)-np.cos(1e6*t)

  #perform convolution
  t, vo, svec = sp.lsim(H, vi, t)

  #plotting code omitted, for that check actual code
  ```

- We now obtain the following plots

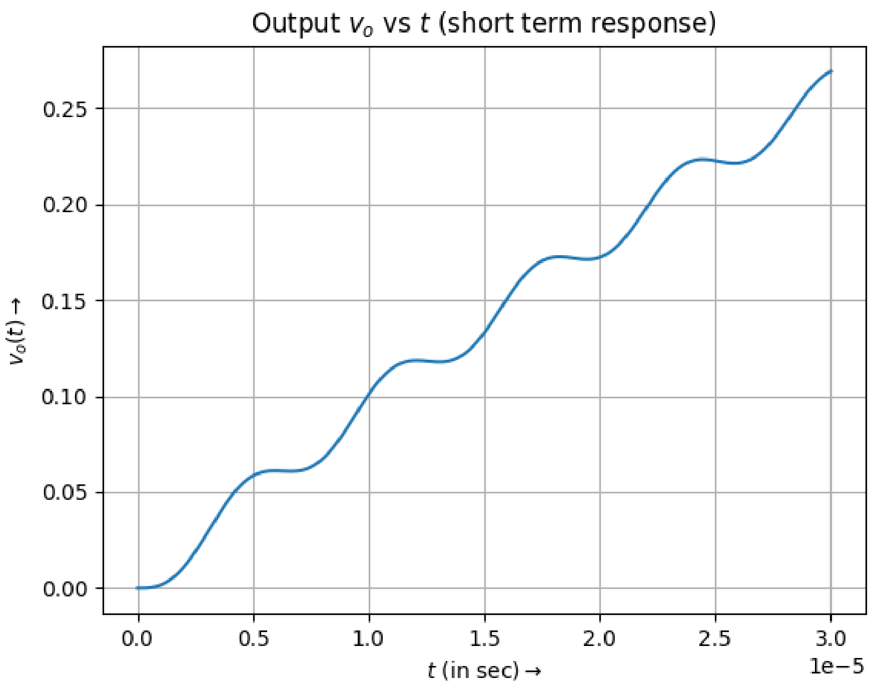


Figure 5: Short term response ($0 < t < 30\mu s$)

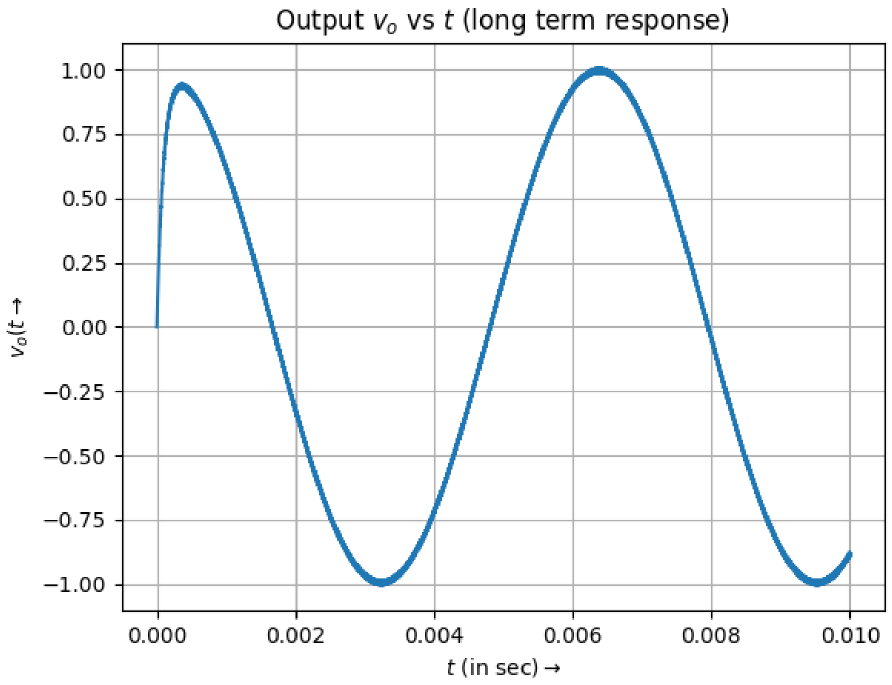


Figure 6: Long term respose ($0 < t < 10ms$)

- We can see that the short term response has some small amount of sinusoidal noise. But the long term response appears to be a sinusoid of frequency approximately $10^3 rad/s$.
- This is expected, since we saw in the magnitude plot that for frequencies beyond $10^4 rad/s$ the system attenuates the response quite appreciably. In fact at $10^6 rad/s$ the signal will be attenuated by as much as 40dB, which explains the very small noise of higher frequency in the output. This magnitude however is negligible compared to the main frequency of $10^3 rad/s$.
- This system behaves like a lowpass filter with a 3dB bandwith of $10^4 rad/s$.

3 Conclusions

- Laplace domain analysis was used to analyse several physical and electrical systems.
- SciPy's signal toolbox functions were used for calculating the impulse response, bode plot and convolution of continuous time LTI systems. Linear Constant Coefficient Differential Equations with known initial conditions were solved in Laplace domain using the properties of Unilateral Laplace Transform.
- A pole in a bode plot contributes a slope of -20dB/dec in magnitude plot and a phase shift of -90^o in phase plot.
- Near a pole the output becomes more amplified and comes closer to instability.
- The concept behind an RLC lowpass filter was used to determine the cutoff frequency and predict the output.