Assignment 6B: The Laplace Transform

J.Phani Jayanth - EE19B026

April 28, 2021

Abstract

This week's Python Assignment involves the following:

- Solving Linear Differential Equations in Laplace Domain (s-domain)
- Analysis of Linear Time Invariant (LTI) Systems
- Using the Signals Toolbox from *scipy* library

LTI systems are what Electrical Engineers spend most of their time thinking about - linear circuit analysis or communication channels for example. In this assignment we will mostly use mechanical examples, but will move on to circuits in the next assignment.

Time Response of a Spring

Time response of the Spring system is obtained by solving the following equation:

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

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

Laplace Transform of f(t) is given by:

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

We first obtain the Laplace Transform of x(t) i.e., X(s) as:

$$X(s) = \frac{F(s)}{s^2 + 2.25} = \frac{s + 0.5}{(s^2 + s + 2.5)(s^2 + 2.25)}$$

We then apply Inverse Laplace Transform on X(s) to obtain x(t):

$$\mathcal{L}^{-1}X(s) = x(t)$$

We use the function *sp.lti()* to define X(s) and then use the *sp.impulse()* function to find the ILT. The resulting signal x(t) thus obtained will be the time response of the spring with damping constant(ζ) = 0.5. The response is plotted against time as shown in Figure 1.

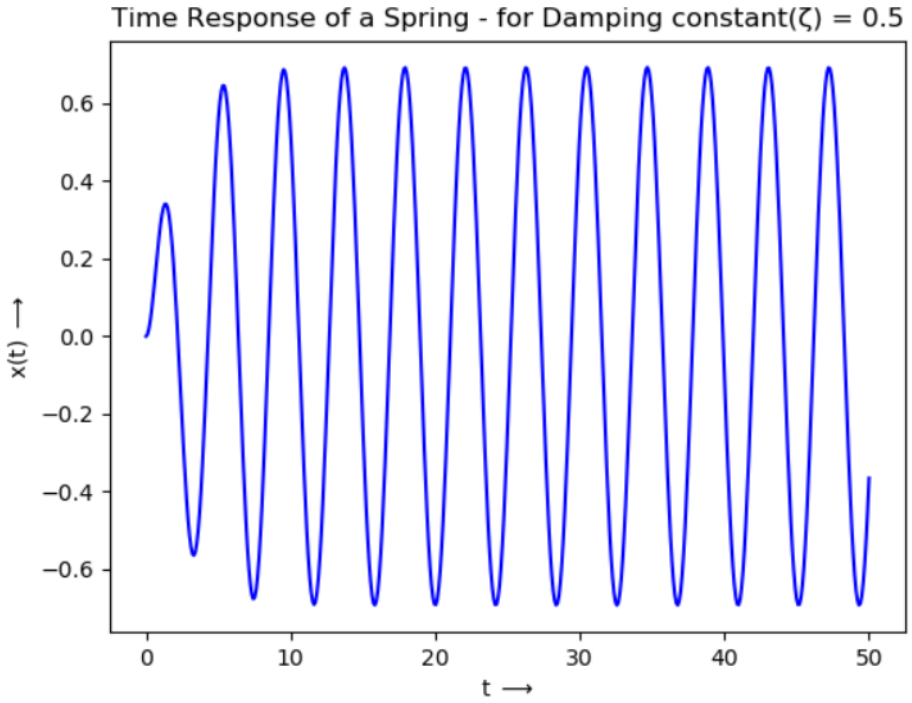

Figure 1: Time Response of the Spring System for Damping Constant(ζ) = 0.5

We repeat the above process for a lower damping constant ($\zeta = 0.05$) i.e., we now compute the time response with $f(t) = cos(1.5t)e^{-0.05t}u(t)$. The time response plot obtained for this is shown in Figure 2.

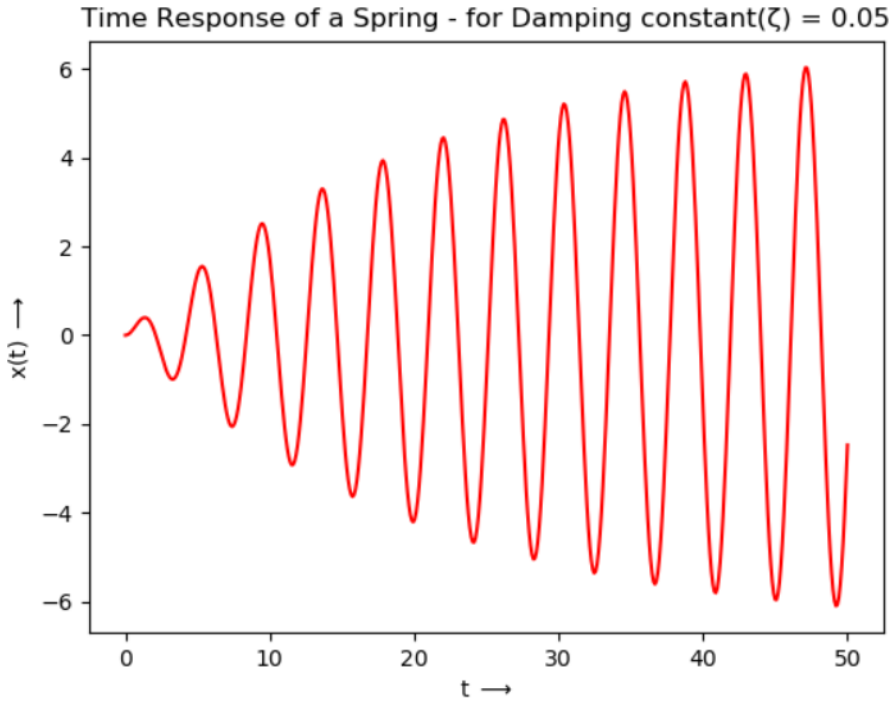

Figure 2: Time Response of the Spring System for Damping Constant(ζ) = 0.05

The code for this is as follows:

```
# Laplace Transform of x(t): X(s) - Damping = 0.5
X = sp.lti(poly1d([1,0.5]),polymul(poly1d([1,1,2.50]),poly1d([1,0,2.25])))
# Inverse laplace Transform of X(s)
t,x = sp.impulse(X,None,linspace(0,50,500))
# Plotting the time response for the spring satisfying x'' + 2.25x = f(t)
plot(t,x,'b')
title("Time Response of a Spring - for Damping constant(\u03B6) = 0.5")
ylabel('x(t) $\longrightarrow$')
xlabel('t $\longrightarrow$')
show()
# Laplace Transform of x2(t): X2(s) - Damping = 0.05
X2 = sp.lti(poly1d([1,0.05]),polymul(poly1d([1,0.1,2.2525]),poly1d([1,0,2.25])))
# Inverse Laplace Transform of X2(s)
t2,x2 = sp.impulse(X2,None,linspace(0,50,500))
# Plotting
plot(t2,x2,'r')
title("Time Response of a Spring - for Damping constant(\u03B6) = 0.05")
ylabel('x(t) $\longrightarrow$')
xlabel('t $\longrightarrow$')
show()
# Plotting the above graphs in the same plots
plot(t,x,'b',label='\u03B6 = 0.5')
plot(t2,x2,'r',label='\u03B6 = 0.05')
title("Time Responses for Damping constant(\u03B6) = 0.5 and 0.05")
ylabel('x(t) $\longrightarrow$')
xlabel('t $\longrightarrow$')
legend()
show()
```

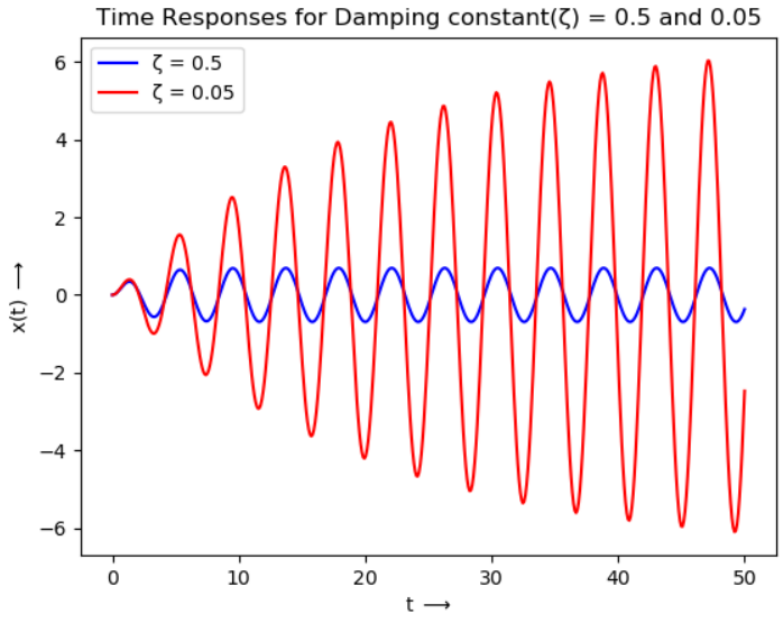

Figure 3: Time Responses of the Spring System for Damping Constants $\zeta = 0.5$ and $\zeta = 0.05$

Obtaining System Responses for varying Frequency

We define the transfer function of the system as:

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

We use the function *sp.lsim()* to find the output by convolution of input signal f(t) and impusle respose of the system h(t) = $\mathcal{L}^{-1}H(s)$. We consider f(t) as follows:

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

where ω is varied from 1.4 to 1.6 in steps of 0.05. The outputs corresponding to these different frequencies are then plotted against time as shown from Figure 4 to Figure 9.

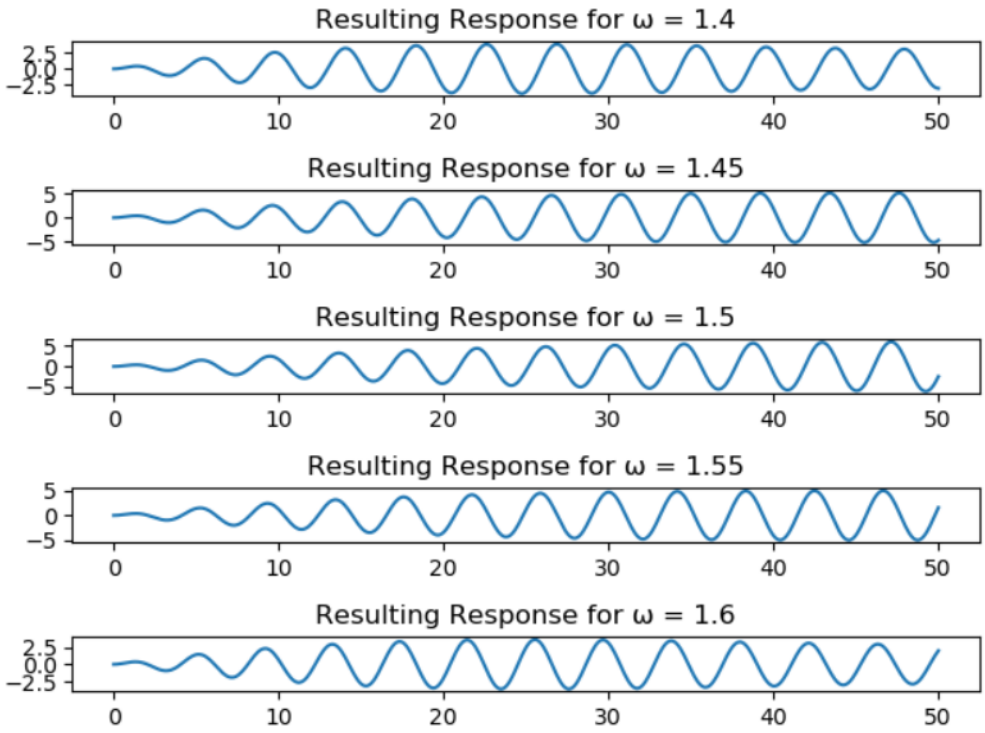

Figure 4: Time Responses for frequencies ranging from 1.4 to 1.6 as subplots

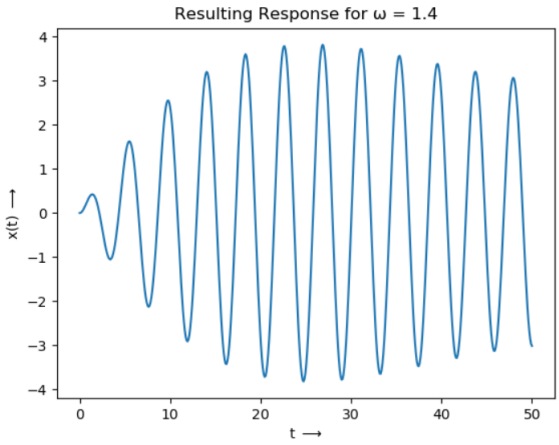

Figure 5: Time Response for $\omega = 1.4$

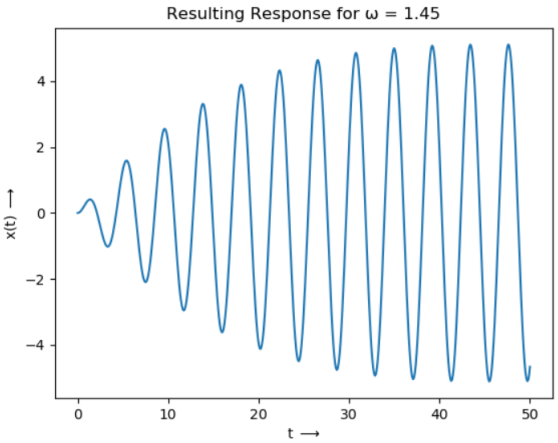

Figure 6: Time Response for $\omega = 1.45$

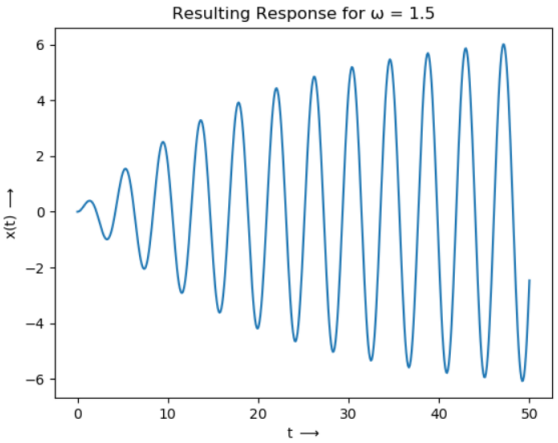

Figure 7: Time Response for $\omega = 1.5$

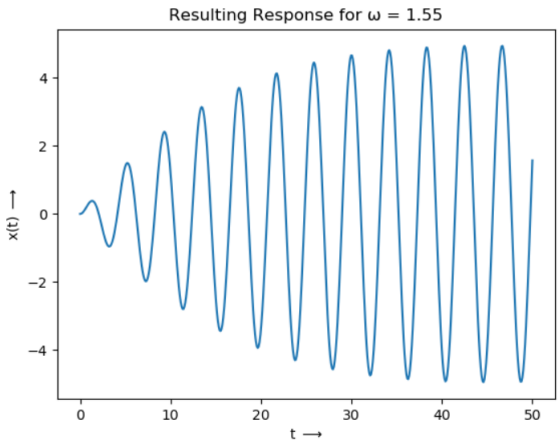

Figure 8: Time Response for $\omega = 1.5$

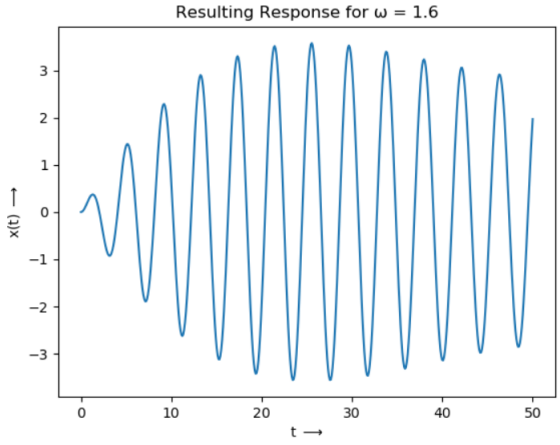

Figure 9: Time Response for $\omega = 1.5$

From the above plots, it is clear that the magnitude (or) amplitude of the output is the highest for $\omega = 1.5$. For frequencies above and below this, the amplitude is reduced. This means that the system has a resonant frequency $\omega_o = 1.5$, and hence the output has the maximum amplitude here. The code is as follows:

```
t = linspace(0,50,500)
u = np.cos(1.5*t)*np.exp(-0.05*t)
# Obtaining the Transfer Function H(s) = X(s)/F(s)
H = sp.lti(1,poly1d([1,0,2.25]))
# Defining frequency range
freq = np.arange(1.4,1.65,0.05)
# Defining f(t)
def f(w,t):
    return np.cos(w*t)*np.exp(-0.05*t)*np.heaviside(t,1)
```

The following *for* loop runs through the frequencies from 1.4 to 1.6 and plots the corresponding Time response plots.

```
for w in freq:
    u = f(w,t)
    # Obtaining the output x(t)
    t,x3,svec = sp.lsim(H,u,t)
    # For Subplots
    if w == 1.40:
        plt.subplot(5,1,1)
        plt.title(f'Resulting Response for \u03C9 = {w}')
        plt.plot(t,x3)
    if w == 1.45:
        plt.subplot(5,1,2)
        plt.title(f'Resulting Response for \u03C9 = {w}')
        plt.plot(t,x3)
    if w == 1.50:
        plt.subplot(5,1,3)
        plt.title(f'Resulting Response for \u03C9 = {w}')
        plt.plot(t,x3)
    if w == 1.55:
        plt.subplot(5,1,4)
        plt.title(f'Resulting Response for \u03C9 = {w}')
        plt.plot(t,x3)
    if w == 1.60:
        plt.subplot(5,1,5)
        plt.title(f'Resulting Response for \u03C9 = {w}')
        plt.plot(t,x3)
plt.tight_layout()
plt.show()

# For plotting individual plots
for w in freq:
    u = f(w,t)
    t,x3,svec = sp.lsim(H,u,t)
    plot(t,x3)
    title(f'Resulting Response for \u03C9 = {w}')
    ylabel('x(t) $\longrightarrow$')
    xlabel('t $\longrightarrow$')
    show()
```

Coupled Spring System

We solve the following time-domain differential equations:

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

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

From the initial conditions x(0) = 1, $\dot{x}(0)$ = y(0) = $\dot{y}(0) = 0$, we get:

$$\ddot{x} = s^2X(s) - s$$

$$\ddot{y} = s^2Y(s)$$

Substituting these in the Linear Differential equations, we obtain X(s) and Y(s):

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

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

Now, we use the function *sp.impulse()* to obtain x(t) and y(t) from X(s) and Y(s). We then plot them as shown in Figure 10.

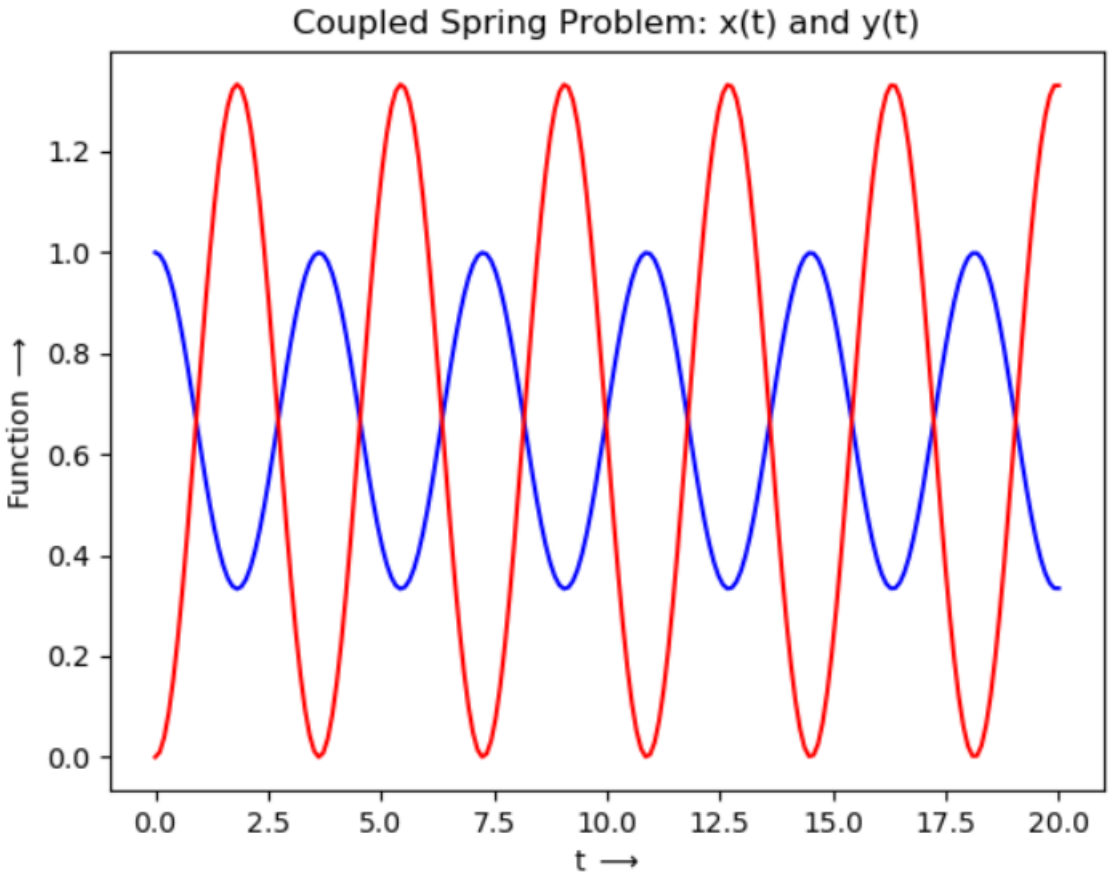

Figure 10: Time Response Plots for Coupled Spring System

Note: x(t) is represented in *blue* and y(t) in *red*.

The code is as follows:

```
# For x(t):
X = sp.lti([1,0,3,0,2,0],[1,0,4,0,3,0,0])
# Inverse Laplace Transform
t,x = sp.impulse(X,None,np.linspace(0,20,200))
# For y(t):
Y = sp.lti([2,0],[1,0,3,0,0])
# Inverse Laplace Transform
t,y = sp.impulse(Y,None,np.linspace(0,20,200))
# Plotting
plot(t,x,'b',label = 'x(t)')
plot(t,y,'r',label = 'y(t)')
title('Coupled Spring Problem: x(t) and y(t)')
ylabel('Function $\longrightarrow$')
xlabel('t $\longrightarrow$')
show()
```

Steady State Transfer Function of Two-Port Network

We know that, for the given LCR Circuit, the transfer function is given as follows:

$$H(s) = \frac{V_o(s)}{V_i(s)} = \frac{1}{s^2LC + sRC + 1}$$

From the transfer function H(s), we obtain the Bode Magnitude Plot and Phase Plot by using *H.bode()*. The Plots are shown in Figure 11.

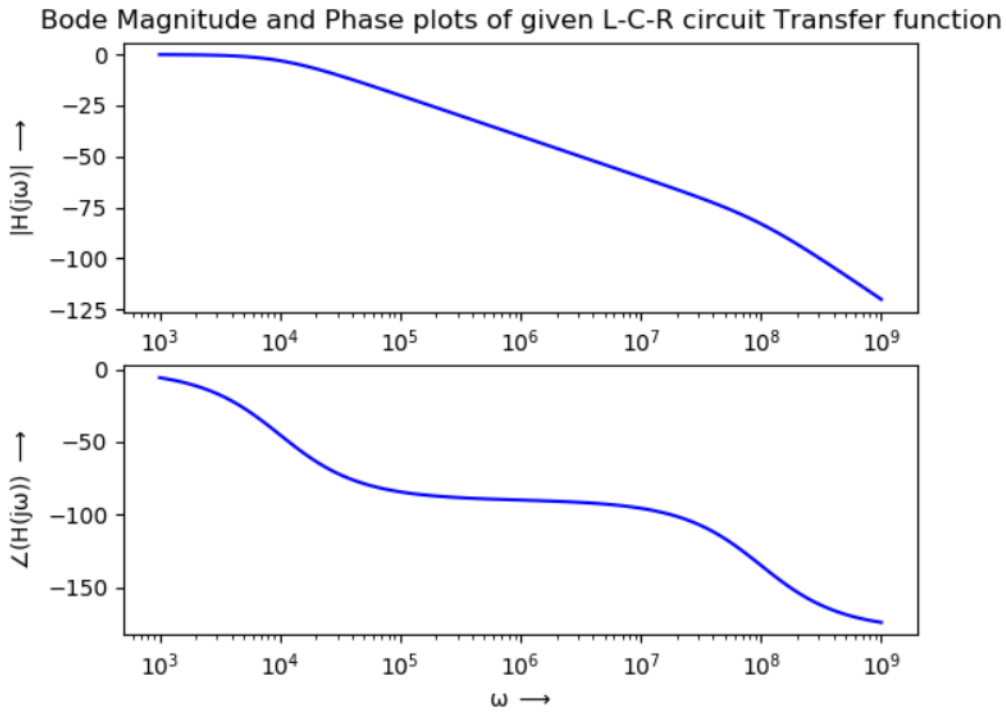

Figure 11: Bode Magnitude Plot and Phase Plot for Transfer Function of LCR Circuit

The code is as follows:

```
R = 100
L = 1e-6
C = 1e-6
# Transfer Function:
H2 = sp.lti(1,poly1d([L*C,R*C,1]))
# Obtaining Phase and Magnitude
w,S,phi = H2.bode()
# Plotting Bode Magnitude Plot and Phase Plot
plt.subplot(2,1,1)
plt.semilogx(w,S,'b')
plt.ylabel('|H(j\u03C9)| $\longrightarrow$')
plt.xlabel('\u03C9 $\longrightarrow$')
plt.title('Bode Magnitude and Phase plots of given L-C-R circuit Transfer
    function')
plt.subplot(2,1,2)
plt.semilogx(w,phi,'b')
plt.ylabel('\u2220(H(j\u03C9)) $\longrightarrow$')
plt.xlabel('\u03C9 $\longrightarrow$')
plt.show()
```

Time-Domain Output Voltage for the LCR Circuit

We have to find the output voltage ($v_o(t)$ from the LCR circuit, for the following input:

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

Output signal is obtained by convolution of the Input Signal and the Impulse response of the system i.e., $v_o(t) = v_i(t) * h(t)$. We achieve this by using the function *sp.lsim()*.

We plot the output voltage for the first 30μs to have a closer look at how the output behaves initially. We can observe from this plot (Figure 12) that the transient voltage increases rapidly in a small amount of time. This is because at t=0, the inductor behaves as a short and the capacitor behaves as an open-line.So, all the increase in input voltage increases the voltage across the capacitor (output voltage). We can see that the higher frequency dominates the response, which corresponds to the ripples observed in the transient.(Figure 12)

The Steady state response can be seen from the 10ms plot. Here, we can see that that the output is similar to a $10^3 rad/s$ sinusoid. This is because the gain for $\omega = 10^6$ is orders of dB less than the gain of $\omega = 10^3$. This means that at the steady state, the latter frequency component will dominate. Therefore, the Steady state output is obtained as shown in Figure 13.

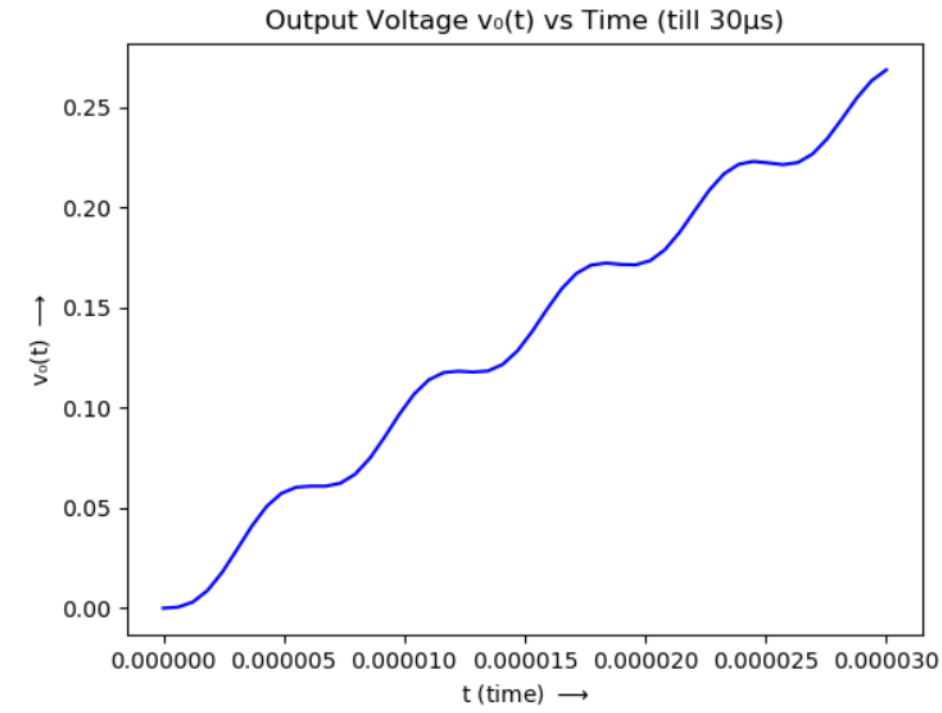

Figure 12: Output signal from $0 < t < 30\mu s$

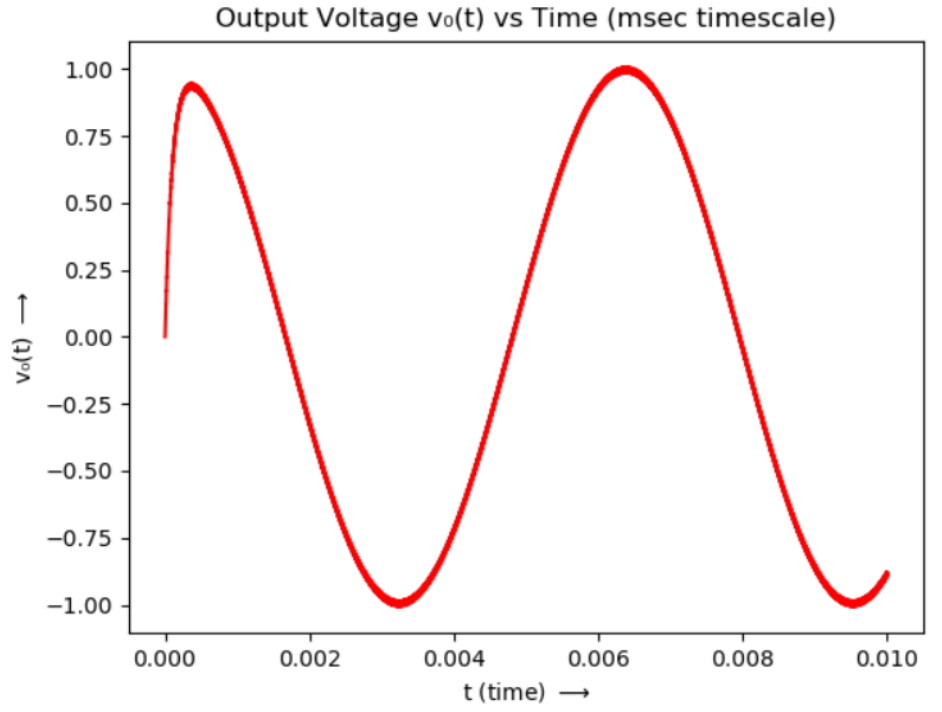

Figure 13: Output Signal at Steady State - msec timescale