ASSIGNMENT 6: THE LAPLACE TRANSFORM

Nithin Babu [EE18B021]

March 8, 2020

Abstract

The goal of this assignment is the following:

- To analyze LTI Systems using Laplace Transform.
- To see how RLC systems can be used as a low pass filter .
- To understand how to use the scipy signals toolkit.
- To plot graphs to understand the Laplace Transform.

1 Finding the Laplace Tranforms

Single Spring System

We use the Laplace transform to solve a simple spring system. The system is characterized by the given differential equation.

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + 2.25x = f(t)$$

(Initial Conditions are all zero) whose Laplace transform is of the form

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

The input signal is of the form $f(t) = \cos(\omega t)\exp(-at)u(t)$, where a is the decay factor and ω is the frequency of the cosine.
The Laplace Transform of the input signal is

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

First we define these function using numpy polynomials and multiply to get the output laplace transform. Finally we take the ILT of the function using sp.impulse to get the time domain sequences and we plot these. We do this for ω=1.5 (natural frequency of the system), and decay of 0.5 and 0.05.

The following python code snippet will solve for $x(t)$ for a decay of 0.5 and 0.05:

```
# The python code snippet for a decay of 0.5 (Q.1)
p11 = poly1d([1,0.5])
p21 = polymul([1,1,2.5],[1,0,2.25])
X1 = sp.lti(p11,p21)
t1,x1 = sp.impulse(X1,None,linspace(0,50,500))

# The python code snippet for a decay of 0.05 (Q.2)
p12 = poly1d([1,0.05])
p22 = polymul([1,0.1,2.2525],[1,0,2.25])
X2 = sp.lti(p12,p22)
t2,x2 = sp.impulse(X2,None,linspace(0,50,500))

# The plot x(t) vs t for Q.1
figure(0)
plot(t1,x1)
title("The solution x(t) for Q.1")
xlabel(r'$t\rightarrow$')
ylabel(r'$x(t)\rightarrow$')
grid(True)

# The plot of x(t) vs t for Q.2
figure(1)
plot(t2,x2)
title("The solution x(t) for Q.2")
xlabel(r'$t\rightarrow$')
ylabel(r'$x(t)\rightarrow$')
grid(True)

show()
```

The plots of $x(t)$ in both cases are as shown below:

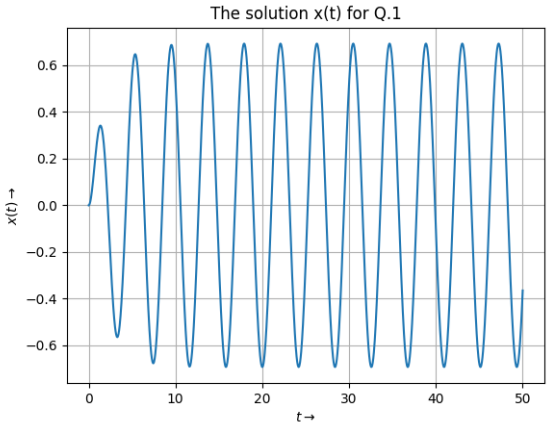


Figure 1: $x(t)$ for a decay of 0.5

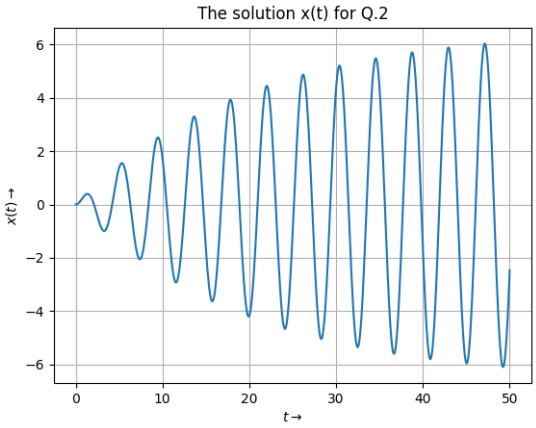


Figure 2: $x(t)$ for a decay of 0.05

We observe that the osciallation amplitude settles to a fixed value in both the cases. We observe it takes longer to settle with decay being less. We also observe that the amplitude increases to a much larger amount in the case with lower decay.

2 Varying the frequency of the input

We vary the frequency of the cosine and see what affect it has on the output. The following python code snippet can be used to calculate $x(t)$ for different frequencies:

```
H = sp.lti([1],[1,0,2.25])
for w in arange(1.4,1.6,0.05):
        t = linspace(0,50,500)
        f = cos(w*t)*exp(-0.05*t)
        t,x,svec = sp.lsim(H,f,t)

# The plot of x(t) for various frequencies vs time.
        figure(2)
        plot(t,x,label='w = ' + str(w))
        title("x(t) for different frequencies")
        xlabel(r'$t\rightarrow$')
        ylabel(r'$x(t)\rightarrow$')
        legend(loc = 'upper left')
        grid(True)

show()
```

The plot showing $x(t)$ for different frequencies are as shown below:

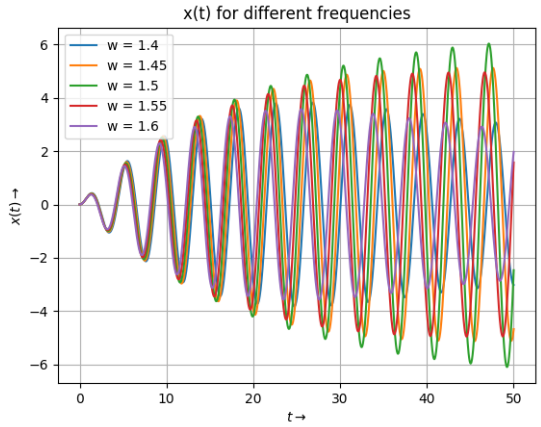


Figure 3: $x(t)$ for different frequencies

When the input frequency is at the natural frequency, the output amplitude is maximum. In the other cases the output amplitude decreases. This phenomenon is known as resonance.

3 Coupled Spring Problem

In this problem we have two differential equations and two variables to solve for. The equations are

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + (x - y) = 0$$

$$\frac{\mathrm{d}^2y}{\mathrm{d}t^2} + 2(y - x) = 0$$

With initial condition as x(0) = 1 We substitute for y in the second equation from the first, and we get a fourth order differential equation in terms of x. Simplifying this and substituting to find the y equation, we get.

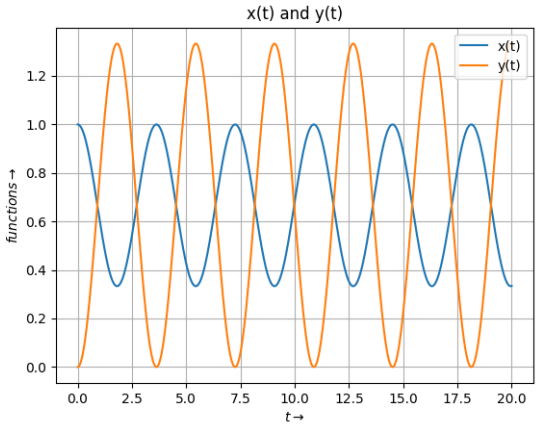
$$X(s) = \frac{s^2 + 2}{s^3 + 3s}$$

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

We can take the ILT of these two expressions to find the time domain expressions for x(t) and y(t). We plot these in 1 graph. The python code snippet for finding $x(t)$ and $y(t)$ are as follows:

```
# The python code snippet for Q.4
t4 = linspace(0,20,500)
X4 = sp.lti([1,0,2],[1,0,3,0])
Y4 = sp.lti([2],[1,0,3,0])
t4,x4 = sp.impulse(X4,None,t4)
t4,y4 = sp.impulse(Y4,None,t4)

# The plot of x(t) and y(t) vs t for Q.4
figure(3)
plot(t4,x4,label='x(t)')
plot(t4,y4,label='y(t)')
title("x(t) and y(t)")
xlabel(r'$t\rightarrow$')
ylabel(r'$functions\rightarrow$')
legend(loc = 'upper right')
grid(True)

show()
```

The plot of $x(t)$ and $y(t)$ are as shown below:



We observe that the amplitude of y is greater than x. The phase of the two are opposite. The offsets are the same for both the expressions. This models two masses attached to the ends of an ideal spring.

4 RLC Filter

We now consider the case of an RLC Filter with the transfer function as shown.

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

(Initial Conditions are all zero) The input is of the form

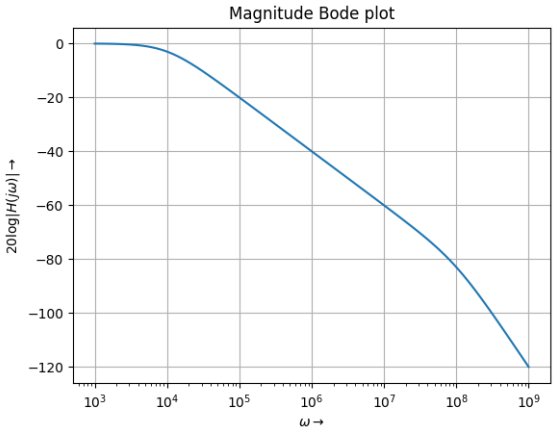
$$x(t) = \cos(10^3 t) - \cos(10^6 t)$$

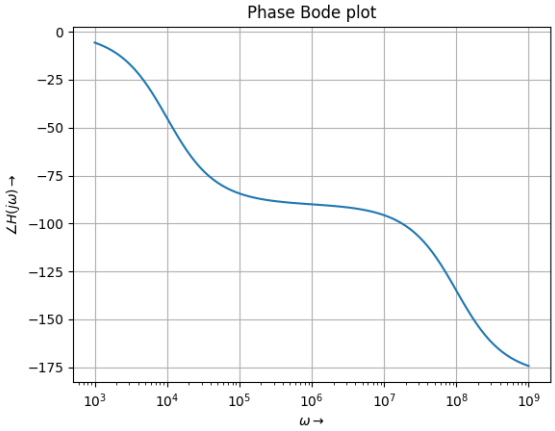
which is basically the superposition of two sinusoids with low and high frequencies. First we plot the bode plot of the transfer function. Then we use sp.lsim to find the output of the filter to the input system. We plot the output from 0 to 30μs (Small time interval) as well as from 0 to 25ms (Large time interval).

```
# The python code snippet for Q.5
temp = poly1d([1e-12,1e-4,1])
H5 = sp.lti([1],temp)
w,S,phi = H5.bode()

# The python code snippet for Q.6
t6 = arange(0,25e-3,1e-7)
vi = cos(1e3*t6) - cos(1e6*t6)
t6,vo,svec = sp.lsim(H5,vi,t6)
```

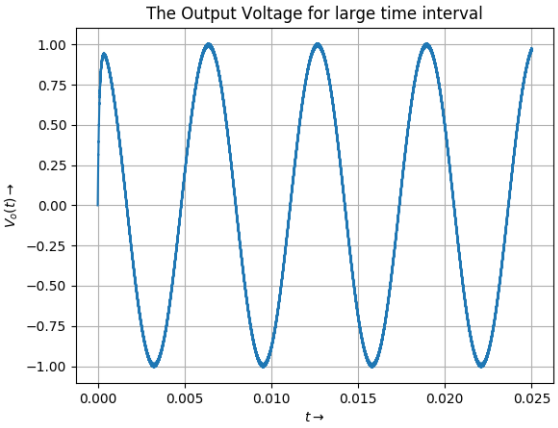
```
# The magnitude bode plot for Q.5
figure(4)
semilogx(w,S)
title("Magnitude Bode plot")
xlabel(r'$\omega\rightarrow$')
ylabel(r'$20\log|H(j\omega)|\rightarrow$')
grid(True)

# The phase bode plot for Q.5
figure(5)
semilogx(w,phi)
title("Phase Bode plot")
xlabel(r'$\omega\rightarrow$')
ylabel(r'$\angle H(j\omega)\rightarrow$')
grid(True)

# The plot of Vo(t) vs t for large time interval.
figure(6)
plot(t6,vo)
title("The Output Voltage for large time interval")
xlabel(r'$t\rightarrow$')
ylabel(r'$V_o(t)\rightarrow$')
grid(True)

# The plot of Vo(t) vs t for small time interval.
figure(7)
plot(t6[0:300],vo[0:300])
title("The Output Voltage for small time interval")
xlabel(r'$t\rightarrow$')
ylabel(r'$V_o(t)\rightarrow$')
grid(True)

show()
```

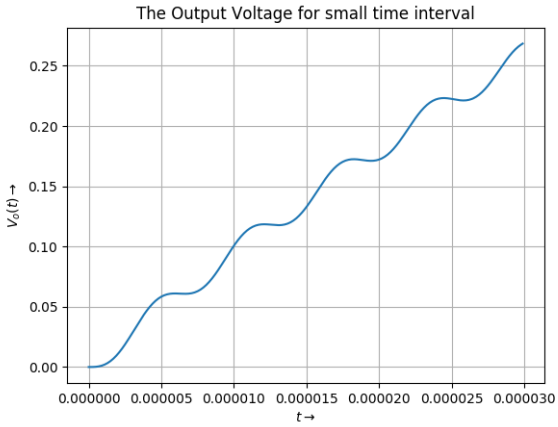
The Bode plots for $H(s)$ is as shown:





The plot of $V_o(t)$ for large and small time inervals are as shown:





From the Bode plot, it is clear that the RLC System is a second order low pass filter. The high frequency component can be seen as a ripple in the small time interval plot. This component is highly attenuated and hence not visible in the large time interval plot. In the large time interval plot, we see that the low frequency component passes almost unchanged, the amplitude is almost 1. The reason is that the $\omega = 10^3 \frac{rad}{s}$ is well within the 3-dB bandwidth ($\omega_{3dB} = 10^4 \frac{rad}{s}$) of the system. Clearly this reiterates the fact that this is a low pass filter with bandwidth $\omega_{3dB} = 10^4 \frac{rad}{s}$.

5 Conclusions

- We analyzed LTI Systems using Laplace Transform.
- We saw a low pass filter constructed from an RLC circuit.
- We used the scipy signals toolkit to calculate the time domain response and the Bode Plot.
- We plotted graphs to understand the above