EE2703 Applied Programming Lab - Assignment No 6

Name: Abishek S
Roll Number: EE18B001

March 9, 2020

1 Abstract

The goal of this assignment is the following :

- To characterize and analyze LTI Systems using Laplace Transform.
- To realise low pass filters using RLC circuits.
- To understand the use of scipy.signals toolkit.
- To plot graphs and understand the Laplace Transform.

2 Assignment

2.1 Setting up the Program

Importing the standard libraries

```
import pylab as pl
import numpy as np
import sys
import scipy.signal as sp
import matplotlib
```

We define a utility function for making plotting simpler :

```
def PLOT(x,y,fig_no = 0,label_x = r'$\rightarrow$',label_y = r'$\rightarrow$',
  fn = pl.plot,arg3 = 'b-',title = "Plot",grids = True,
  cmap = matplotlib.cm.jet,label = ''):
        pl.figure(fig_no)
        pl.grid(grids)
        if fn == pl.contourf:
                fn(x,y,arg3,cmap = cmap)
                pl.colorbar()
        else:
                if label == '':
                        fn(x,y,arg3)
                else:
                        fn(x,y,arg3,label = label)
                        pl.legend()
        pl.xlabel(label_x,size = 17)
        pl.ylabel(label_y,size = 17)
        pl.title(title)
```

We also define a utility function for making Bode Plots :

```
def bodeplot(w,s,phi):
    '''Makes Bode Plots'''
    plt.subplot(2,1,1)
    plt.semilogx(w,s)
    plt.xlabel(r'$\omega$',size=17)
    plt.ylabel(r'$|H(j\omega)|$',size =17)
    plt.subplot(2,1,2)
    plt.semilogx(w,phi)
    plt.xlabel(r'$\omega$',size=17)
    plt.ylabel(r'$\angle(H(j\omega))$',size =17)
```

2.2 Single Spring System

2.2.1 Varying the Decay rate of the Input - QNs 1,2

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 all are zero)
The 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

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

We first use numpy polynomials to define the input, zero state response,transfer function and output in Laplace domain.

The output of the system is **Zero Input + Zero State response**, where Zero Input response is Transfer function * F(s).
Once we get the output in Laplace domain, we take it's Inverse Laplace Transform (ILT) 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 rate of 0.5 and 0.05.
We plot the outputs for $0 <= t <= 50$.

```
def input_fn(decay=0.5,cos_term=1.5):
        '''Laplace Tranform of input function'''
        return (np.poly1d([1,decay]),
                np.poly1d([1,2*decay,decay**2 + cos_term**2]))

def transfer_fn(a=1,b=0,c=2.25):
        '''Transfer function of system
        for (a*s^2 + b*s + c)*X(s)'''
        return (np.poly1d([1]),np.poly1d([a,b,c]))

def zero_st(a=1,b=0,c=2.25,xi=0,xii=0):
        '''Zero state Response of system
        for (a*s^2 + b*s + c)*X(s) with x(0) = xi and dx/dt at t = 0 is xii'''
        return (np.poly1d([a*xi,b*(xi+xii)]),np.poly1d([a,b,c]))

def output_fn(F,a=1,b=0,c=2.25,xi=0,xii=0):
        '''Laplace Transform of system output
        where F is input in Laplace domain'''
        ziN,ziD = zero_inp(a,b,c)
        zsN,zsD = zero_st(a,b,c,xi,xii)
        return (np.polyadd(np.polymul(F[0],ziN),np.polymul(F[1],zsN)),
                np.polymul(F[1],ziD))

t = np.linspace(0,50,200)

#Plotting the output (x) for decay rate = 0.5 (Qn1)
t,x1 = sp.impulse(output_fn(input_fn()),None,t)
PLOT(t,x1,1,r'$t\rightarrow$',r'$x(t)\rightarrow$',
        title = 'System response for decay rate = 0.5')
pl.show()

#Plotting the output (x) for decay rate = 0.05 (Qn2)
t,x2 = sp.impulse(output_fn(input_fn(decay=0.05)),None,t)
PLOT(t,x2,2,r'$t\rightarrow$',r'$x(t)\rightarrow$',
        title = 'System response for decay rate = 0.05')
pl.show()
```

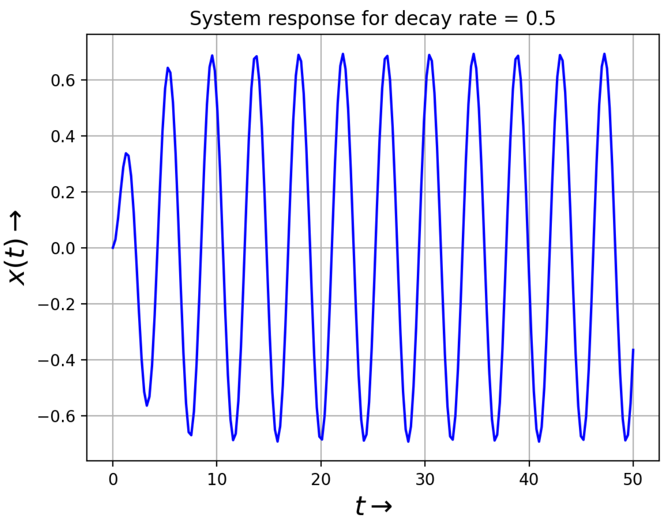


Figure 1: System Output for decay rate = 0.5

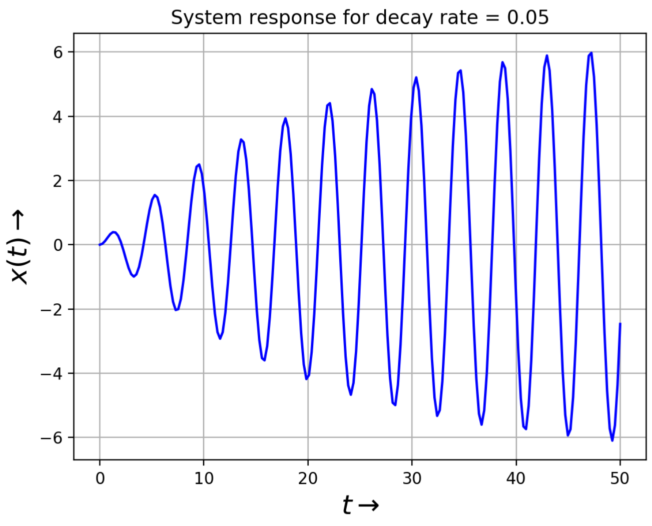


Figure 2: System Output for decay rate = 0.05

We observe that the **oscillation amplitude settles to a fixed value** in both the cases.
We observe it **takes longer to settle when the decay rate is less**.
We also note that the **amplitude increases to a much larger amount in the case with lower decay rate**.
At zero decay rate the amplitude increases to infinity, and at high decay rate the output reaches the maximum amplitude almost instantaneously.

2.2.2 Varying the frequency of the input - QN 3

We first define a **input_td** function which gives the input values in time domain corresponding to the time values passed to it.

```
def input_td(t,decay=0.5,cos_term=1.5):
        '''Return list of input values in time domain
                corresponding to the t values given'''
        return (np.cos(f*t)*np.exp(-1*decay*t))
```

We vary the frequency of the cosine term in input and analyse it's effect.
We plot the outputs for $0 <= t <= 100$.
We also construct the bode plot of the transfer function to understand the results in a better way.

```
t = np.linspace(0,100,300)
H = transfer_fn()

#We use the following 5 colours for the lines :
#Black Green Red Cyan and Magenta
for f,col in zip(np.arange(1.4,1.61,0.05),['k','g','r','c','m']):
        u = input_td(t,0.05,f)
        t,y,svec = sp.lsim(H,u,t)
        PLOT(t,y,3,r'$t$',r'$x(t)$',
                title = 'System responses for different frequencies
                with decay rate = 0.05',arg3 = col+'-',label = 'freq '+str(f))

pl.show()

#We make the Bode plot of the transfer function to understand it better
w,s,phi = sp.lti(H[0],H[1]).bode()
pl.figure(4)
pl.title('Bode plot of Transfer function of single spring system')
bodeplot(w,s,phi)
pl.show()
```

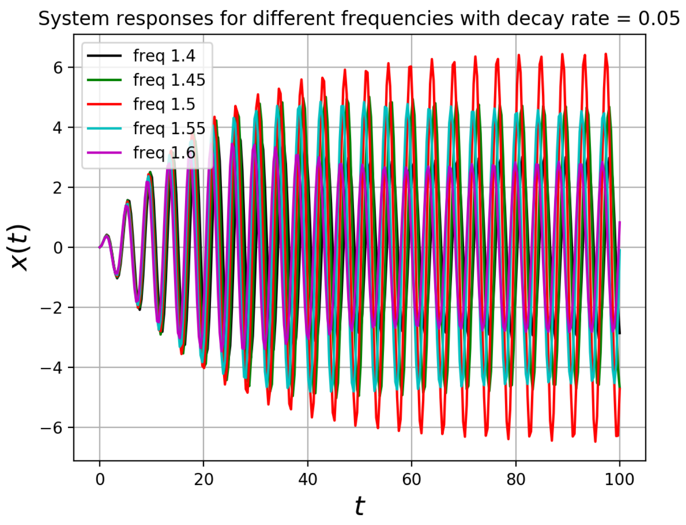


Figure 3: System Output for different frequencies at decay rate = 0.05

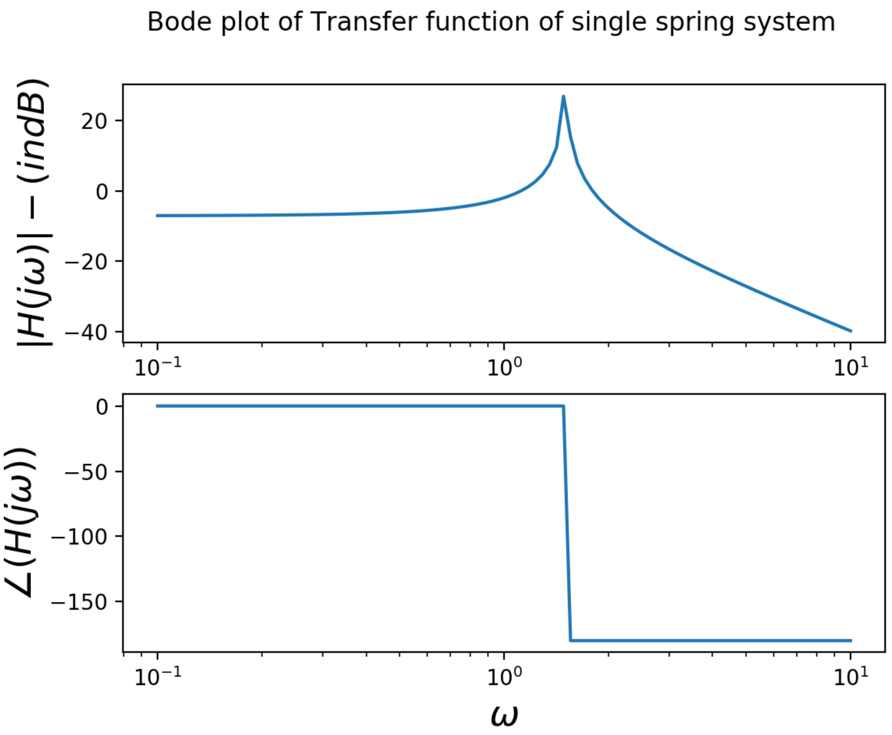


Figure 4: Bode Plot of Transfer function (both magnitude and phase)

When the **input frequency is equal to the natural frequency the output amplitude is maximum**.
In the other cases the output amplitude is less than that. This phenomenon can be attributed to **Resonance**.
We can also clearly see from the bode plot that there is a maximum at the natural frequency, and there is a second order pole there, as the phase shift is $-180\,\text{deg}$.

2.3 Coupled Spring System - QN 4

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 conditions for x (some are given, others can be obtained by substituting in the given equations) :

$$x(0) = 1$$
$$x^{(1)}(0) = 0$$
$$x^{(2)}(0) = -1$$
$$x^{(3)}(0) = 0$$

We substitute 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 y, 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 for $0 <= t <= 20$ in one graph.

```
t = np.linspace(0,20,200)
#Transfer function for x
X = sp.lti([1,0,2],[1,0,3,0])

#Transfer function for y
Y = sp.lti([2],[1,0,3,0])

t,x = sp.impulse(X,None,t)
t,y = sp.impulse(Y,None,t)
```

```
#Plotting x(t) and y(t) in a single graph
PLOT(t,x,5,'t','f(t)',title = 'Plot of x(t) and y(t) in coupled spring system',
        label = 'x(t)')
PLOT(t,y,5,'t','f(t)',title = 'Plot of x(t) and y(t) in coupled spring system',
        arg3 = 'g-',label = 'y(t)')
pl.show()
```

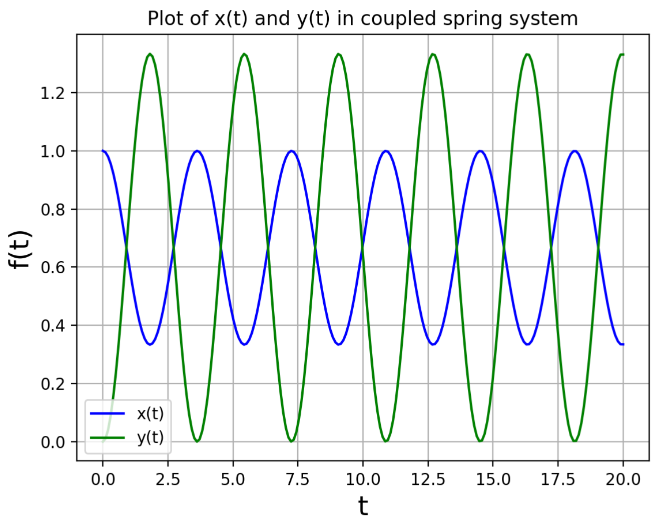


Figure 5: x(t) and y(t) in coupled spring system

We observe that the **amplitude of y is greater than x**.
They are **opposite in phase**.
They oscillate about the same mean point.
This models two masses attached to the ends of an ideal spring.

2.4 RLC Filter - QNs 5,6

We now consider an RLC Filter with transfer function calculated as :

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

(Initial Conditions all zero)
We define a function to return the Transfer function of the RLC circuit in **np.poly1d** form :

```
def RLCtf():
        '''Transfer function of the RLC circuit given'''
        return (np.poly1d([1]),np.poly1d([1e-12,1e-4,1]))
```

We then make the Bode plot of the Transfer function.

```
#Defining time vector appropriately from 0 to 10msec with 1e-7 time steps
        to cpature the fast variation
t = np.arange(0,10e-3,1e-7)
H = RLCtf()

w,s,phi = sp.lti(H[0],H[1]).bode()
pl.figure(4)
bodeplot(w,s,phi)
pl.suptitle('Bode plot of Transfer function of RLC filter')
pl.show()
```

From the Bode plot, it is clear that the RLC System is a **second order Low Pass Filter**.
The input is of the form

$$x(t) = \cos\left(10^3 t\right) + \cos\left(10^6 t\right)$$

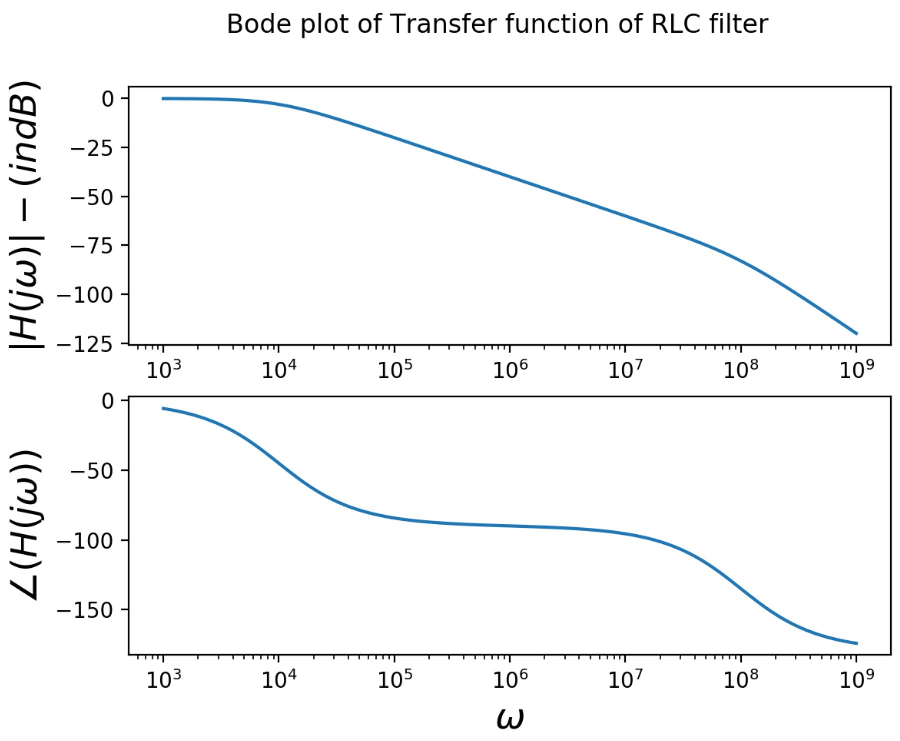
which is the superposition of two sinusoids with low and high frequencies of 10^3 and 10^6 respectively.
We define another function that returns the input to the RLC circuit :

```
def RLCinp():
        '''Input to the RLC circuit'''
        return (np.cos(1e3*t)-np.cos(1e6*t))
```

We use **sp.lsim** to find the output of the filter, which will be convolution between input and ILT of the Transfer function. We plot the output from 0 to 30μs (to capture the **Fast time** part) as well as from 0 to 10ms (to capture the **Slow time** part).

```
#Solving for the RLC filter output
u = RLCinp()
t,x,svec = sp.lsim(H,u,t)

#Plotting the slow time and fast time outputs separately
PLOT(t,x,6,'t','V(t)',title = 'Plot of Output voltage of
        RLC circuit - Slow time')
pl.show()
PLOT(t[:300],x[:300],7,'t','V(t)',title = 'Plot of Output voltage of
        RLC circuit - Fast time')
pl.show()
```



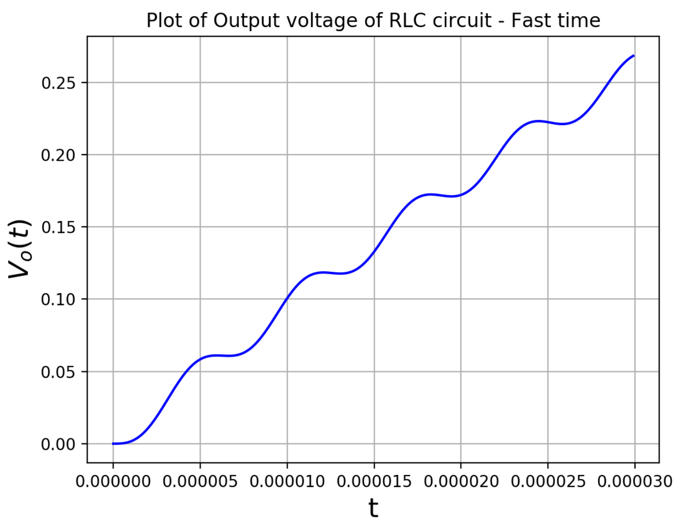


Figure 6: RLC Circuit - Plot of Output Voltage - Fast Time

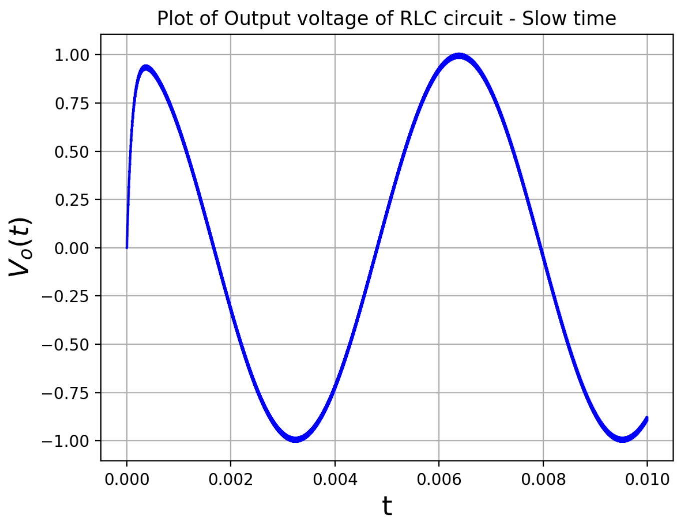


Figure 7: RLC Circuit - Plot of Output Voltage - Slow Time

The fast time plot shows that the **capacitor is charging up** to meet the input amplitude.
The high frequency component can be seen as a **ripple** in the fast time plot. This component is highly attenuated and hence not visible in the slow time plot.
In the slow time plot, we notice that the low frequency component passes almost unchanged, the amplitude is nearly 1. The reason for this is that $\omega = 10^3\frac{rad}{s}$ is well within the 3-dB bandwidth ($\omega_{3dB} \approx 10^4\frac{rad}{s}$) of the system. This reiterates the fact that this is a low pass filter with bandwidth $\omega_{3dB} \approx 10^4\frac{rad}{s}$.

3 Conclusions

- We modelled the spring system and RLC circuit as LTI systems and analysed them using Laplace Transforms.
- We observed how a RLC circuit can behave as a Low Pass Filter.
- We used the scipy signals toolkit to construct Bode Plots and calculate time domain response and convolutions.
- We plotted graphs and Bode plots to understand the behaviour of those LTI systems.