EE2703-Assignment6-2

EE19B094

April 2021

Abstract

In this assignment, we will look at how to analyze and solve for LTI systems using standard libraries in python .We restrict our analysis to systems with rational polynomial transfer functions. More specifically we consider 3 systems: A forced oscillatory system, A coupled system of Differential Equations and an RLC low pass filter

1 Question 1,2

We are given an equation describing forced oscillatory system(with 0 initial conditions) as:

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

where x is the output and f(t) is the input.

Thus the transfer function of the system is

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

Now, we are given input f(t) as $cos(1.5t)e^{-0.5t}u_0(t)$ i.e.

$$F(s) = \frac{s + 0.5}{(s + 0.5)^2 + 2.25} \tag{3}$$

Thus the output will be X(s) = H(s)F(s)

$$X(s) = \frac{s + 0.5}{((s + 0.5)^2 + 2.25)(s^2 + 2.25)} \tag{4}$$

Now, we use the sp.lit() function to find the plot the time domain output x(t). We also do the same for similar input function but with different decay constant

```
#Q1
def Q_1(freq,decay_constant):
   #Function to get laplace equation for input for given values of
       frequency and decay constant
   def input_laplace(freq,decay_constant):
      num = np.poly1d([1,decay_constant])
      den =
          np.poly1d([1,2*decay_constant,decay_constant*decay_constant+freq*freq])
      return num,den

   #Input function
   input_num,input_den = input_laplace(freq,decay_constant)

   #Output function by multiplying transfer function and input
       function
   output_laplace_num = input_num
```

```
    output_laplace_den = np.polymul(input_den,[1,0,2.25])
    output_laplace = sp.lti(output_laplace_num,output_laplace_den)

    #Get time domain output for time = 0 to 5001
    time,output_time =
        sp.impulse(output_laplace,None,np.linspace(0,50,5001))

    #Plot graph
    g1 =
        General_Plotter("Time$\longrightarrow$","Position$\longrightarrow$","Q1:
        Decay = %s"%decay_constant,[])
    g1.line_plot(time,output_time)


Q_1(1.5,0.5)  #Decay constant = 0.5
Q_1(1.5,0.05) #Decay constant = 0.05
```

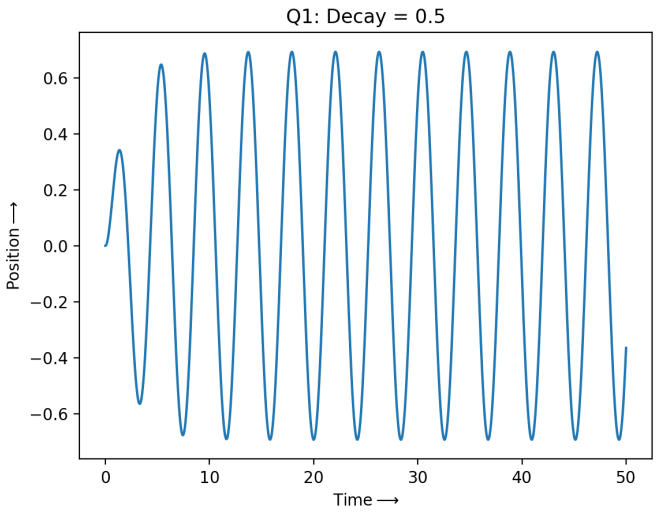

Figure 1: System Response with Decay = 0.5

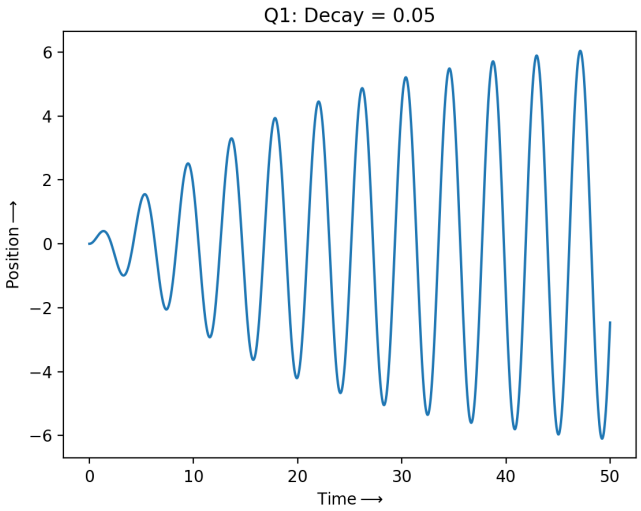

Figure 2: System Response with Decay = 0.05

For different values of decay constant, system gives similar final output except the time taken to reach steady state changes.

2 Question 3

We now simulate the above equation but with varying frequencies this time.

```
#Q3
def Q_3():
   #Helper function which returns input function values at given
       time stamps
   def input_time(freq,decay_constant,time):
      cos = np.cos(freq*time)
      exp =
          np.multiply(np.exp(-decay_constant*time),np.heaviside(time,0.5))
      return np.multiply(cos,exp)

   #Looping through different values of frequency and plotting the
       output time domain simulation
   for freq in np.arange(1.4,1.6,0.05):
      transfer_function = sp.lti([1],[1,0,2.25])
      time_stamps = np.linspace(0,100,1000)

      #Simulating
      _,response,_ =
          sp.lsim(transfer_function,input_time(freq,0.05,time_stamps),time_stamps)
      g1 =
          General_Plotter("Time$\longrightarrow$","Position$\longrightarrow$","Q3:
```

```
        Frequency = %s"%freq,[])
    g1.line_plot(time_stamps,response)

Q_3()
```

The results are the following,

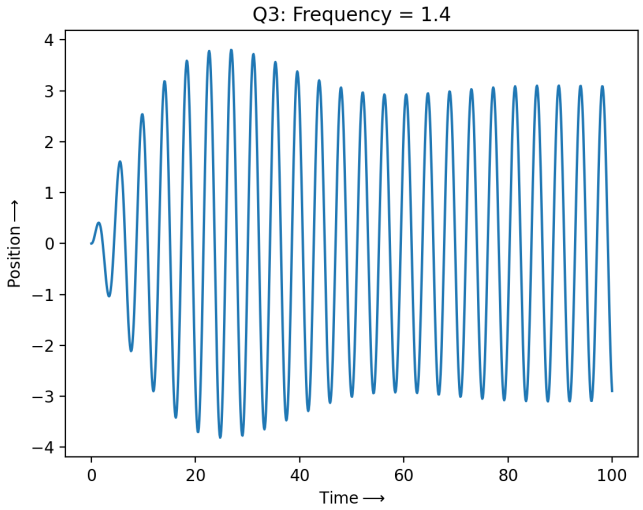

Figure 3: System Response with frequency = 1.4

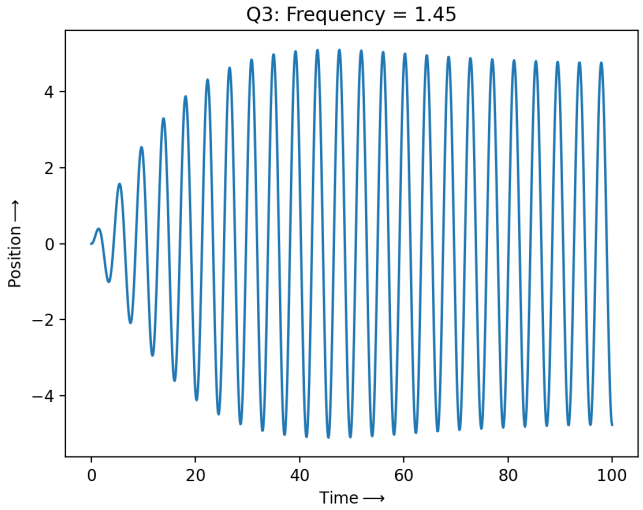

Figure 4: System Response with frequency = 1.45

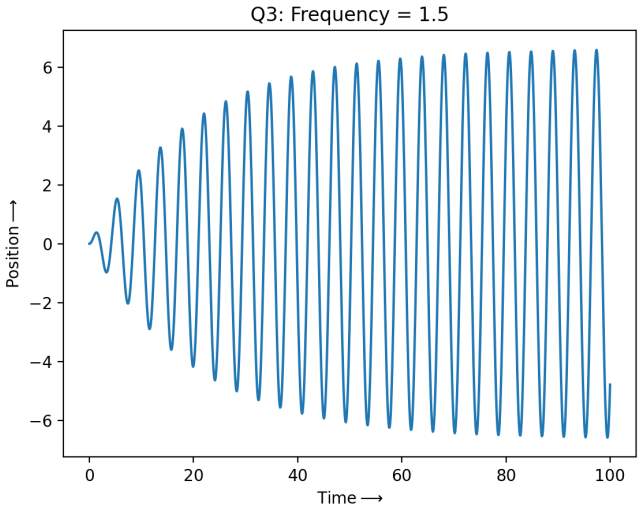

Figure 5: System Response with frequency = 1.5

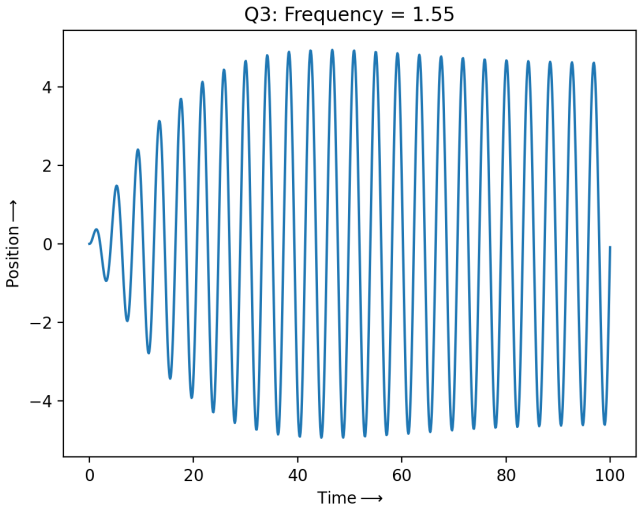

Figure 6: System Response with frequency = 1.55

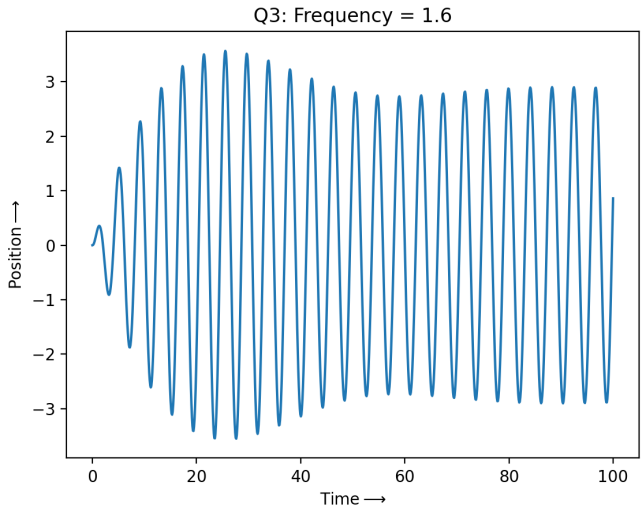

Figure 7: System Response with frequency = 1.6

We notice that the value of amplitude is maximum for frequency = 1.5, which is true because it is also the natural frequency of our system.

3 Question 4

We now consider a coupled Differential system

$$\ddot{x} + (x - y) = 0 \tag{5}$$

and

$$\ddot{y} + 2(y - x) = 0 \tag{6}$$

with the initial conditions: $\dot{x}(0) = 0, \dot{y}(0) = 0, x(0) = 1, y(0) = 0$. Taking Laplace Transform and solving for $X(s)$ and $Y(s)$, We get:

$$X(s) = \frac{s^2 + 2}{s^3 + 3s} \tag{7}$$

$$Y(s) = \frac{2}{s^3 + 3s} \tag{8}$$

Thus in laplace domain both equations are uncoupled. We find the corresponding equations in time domain

```
#Q4
def Q_4():
   #Laplace function for x and y obtained by decopling the equations
   laplace_function_x =
       sp.lti(np.poly1d([1,0,2]),np.poly1d([1,0,3,0]))
```

```
    laplace_function_y = sp.lti(np.poly1d([2]),np.poly1d([1,0,3,0]))
    time_stamps = np.linspace(0,20,1000)

    #Obtaining the output in time domain for each laplace function
    response_x = sp.impulse(laplace_function_x,None,time_stamps)
    response_y = sp.impulse(laplace_function_y,None,time_stamps)

    g1 =
        General_Plotter("Time$\longrightarrow$","X$\longrightarrow$","Q4:
        X vs time",[])
    g1.line_plot(response_x[0],response_x[1])

    g1 =
        General_Plotter("Time$\longrightarrow$","Y$\longrightarrow$","Q4:
        Y vs time",[])
    g1.line_plot(response_y[0],response_y[1])

Q_4()
```

The corresponding graphs are,

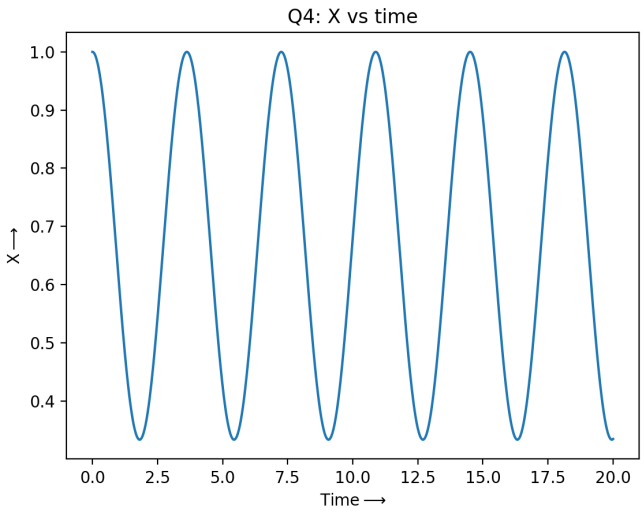

Figure 8: Value of X vs Time

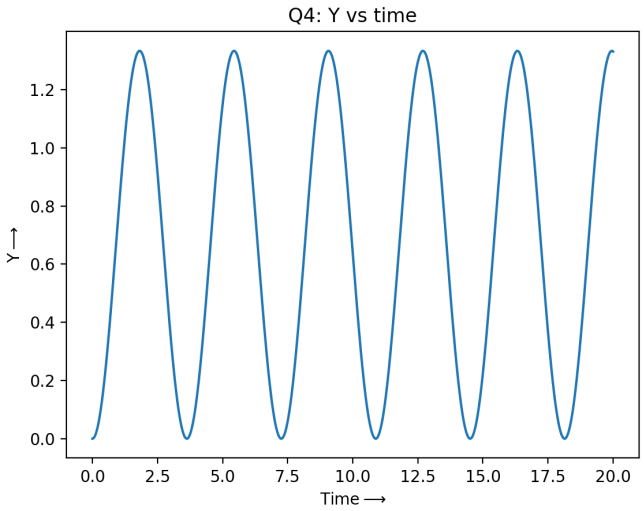

Figure 9: Value of Y vs Time

We notice that the outputs of this system are 2 sinusoids which are 90° out of phase. This system can be realized physically by creating an undamped single spring double mass system where x and y are position values of the blocks.

4 Question 5

Here, we plot the bode plot of the transfer function of the low-pass filter circuit given in the question.

$$H(s) = \frac{1}{LCs^2 + RCs + 1} \tag{9}$$

```
#Q5
def Q_5():
   #Transfer function for the system
   transfer_function =
       sp.lti(np.poly1d([1000000]),np.poly1d([0.000001,100,1000000]))

   #Obtaining and plotting the bode plot of the transfer function
   freq,magnitude,phase = transfer_function.bode()

   plt.subplot(2,1,1)
   plt.semilogx(freq,magnitude)
   plt.ylabel(r'$|H(s)|$')
   plt.subplot(2,1,2)
   plt.semilogx(freq,phase)
   plt.ylabel(r'$\angle(H(s))$')
```

```
    plt.suptitle("Q5: Bode plot of two port network")
    plt.show()
    return transfer_function

transfer_function = Q_5()
```

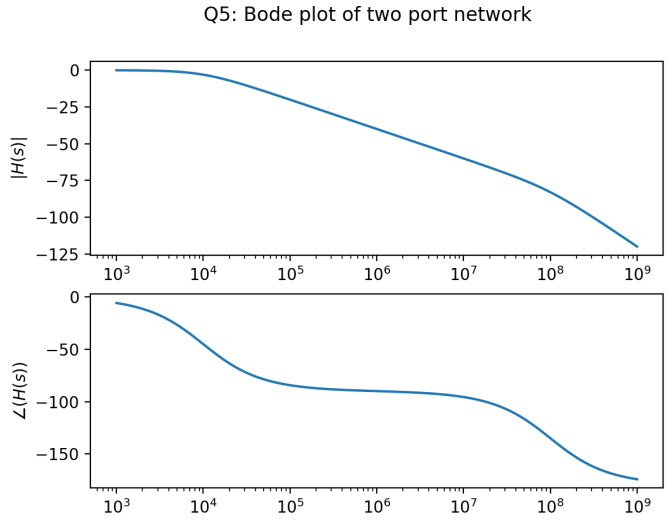

Figure 10: Bode Plots For RLC Low pass filter

5 Question 6

Now we find output voltage for a given input voltage $V_i(t)$,

$$V_i(t) = (cos(10^3 t) - cos(10^6 t))u(t) \tag{10}$$

We plot them for two time ranges i.e. for $0 < t < 30\mu s$ and $0 < t < 10ms$

```
#Q6
def Q_6(transfer_function):
    #Simulating and plotting Q5 for given input for time uptill 30us
    time = np.linspace(0,30*0.000001,1000)
    vi =
        np.multiply(np.cos(1000*time)-np.cos(1000000*time),np.heaviside(time,0.5))
    _,output_time,_ = sp.lsim(transfer_function,vi,time)
    g1 =
        General_Plotter("Time$\longrightarrow$","Voltage$\longrightarrow$","Q6:
        Voltage uptill 30$\mu$s",[])
    g1.line_plot(time,output_time)
```

```
#Simulating and plotting Q5 for given input for time uptill 1ms
time = np.linspace(0,10*0.001,100000)
vi =
    np.multiply(np.cos(1000*time)-np.cos(1000000*time),np.heaviside(time,0.5))
_,output_time,_ = sp.lsim(transfer_function,vi,time)
g1 =
    General_Plotter("Time$\longrightarrow$","Voltage$\longrightarrow$","Q6:
    Voltage uptill 10ms",[])
g1.line_plot(time,output_time)

Q_6(transfer_function)
```

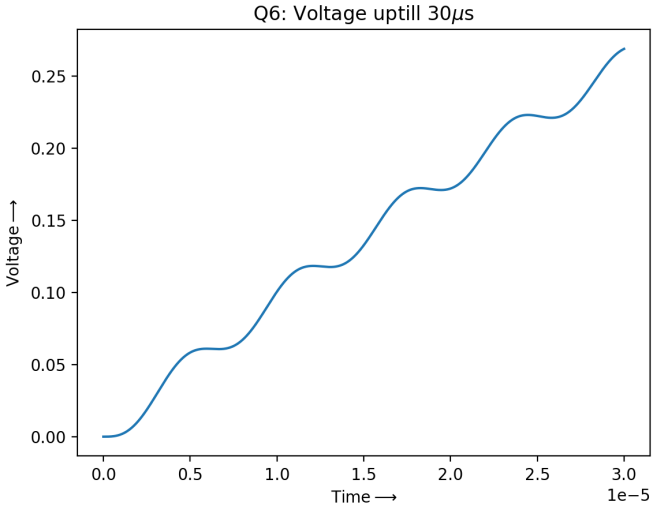

Figure 11: System response for t<30us

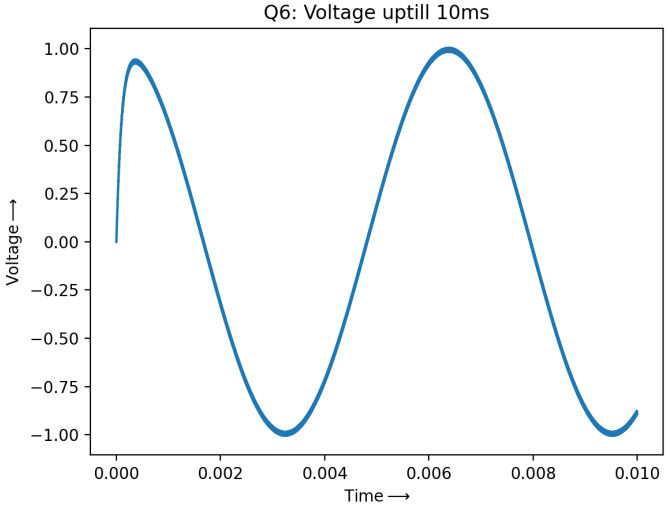

Figure 12: System response for t<10ms

The oscillations in the curve in small time range such as for the $30\mu s$ curve are due to the high frequency component of the input.

However as, the system is low-pass filter, it will allow low frequencies as 10^3 to pass, however frequencies such as 10^6 will get damped.

We can see that amplitude of high frequencies have dropped to almost 0.02 on zooming.

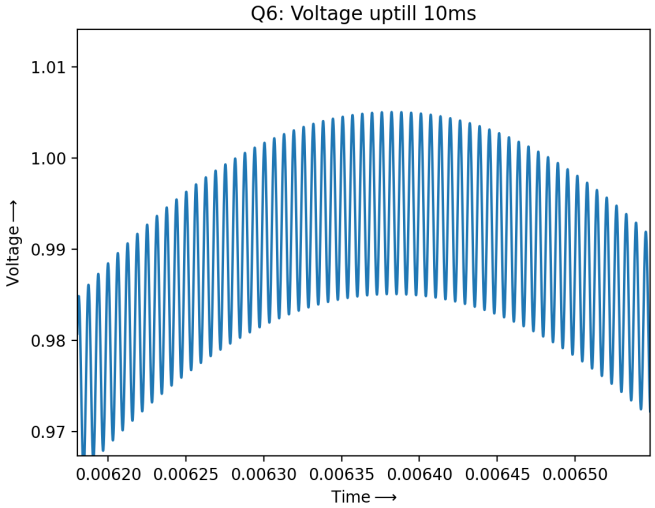

Figure 13: Zoomed System response for t<10us

6 Conclusion

We used the scipy.signal library to solve circuits and equations in laplace domain including forced response of simple spring body system, a coupled spring problem and a low-pass filter circuit was analysed and output signal found for a mixed frequency input signal.