Assignment No 6-2, The Laplace Transform

Pranav Phatak, EE19B105

April 17, 2021

Objective

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

Spring response to forced input system

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

```
#Question 1
def input_signal(freq,decay):
    """Transfer function of the given system"""

    n = np.poly1d([1,decay])
    d = n*n+freq**2
    return n,d

def general_transfer_fcn(wn=1.5,zeta=0,gain=1/2.25):
    """General transfer function"""

    n = np.poly1d([wn**2*gain])
    d = np.poly1d([1,2*wn*zeta,wn**2])
    return n,d

def LTI(decay,freq=1.5):
    """Find the response to the given system to a decaying cosine."""

    input_numerator, input_denominator = input_signal(freq,decay=decay)
    transfer_numerator, transfer_denominator = general_transfer_fcn()

    output_numerator,output_denominator = input_numerator*transfer_numerator, input_
    out_s = sp.lti(output_numerator.coeffs, output_denominator.coeffs)

    t = np.linspace(0,50,1000)

    return sp.impulse(out_s,None,t)

X1,Y1 = LTI(0.5)    #Decay constant is 0.5
p1=General_Plotter(r"$t$",r"$x$",r"Q1: System response with decay of $0.5$")
p1.general_plot(X1,Y1)

#Question 2
X2,Y2 = LTI(0.05)   #Decay constant is 0.05
p2=General_Plotter(r"$t$",r"$x$",r"Q2: System response with decay of $0.05$")
p2.general_plot(X2,Y2)
```

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

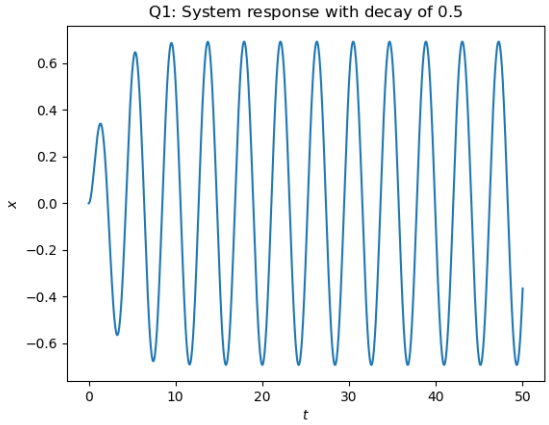

Figure 1: System response with decay 0.5

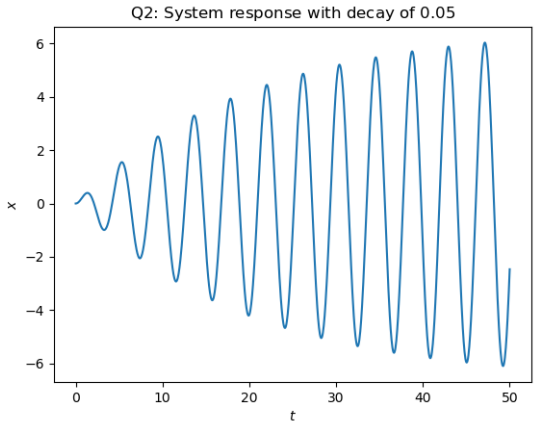

Figure 2: System response with decay 0.05

Varying frequencies

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

```
#Question 3
def input_time(t,decay=0.5,freq=1.5):
    """Exponentially decaying cosine function."""

    u_t = 1*(t>0)
    return np.cos(freq*t)*np.exp(-decay*t) * u_t

time_stamps = np.linspace(0,100,1000)
transfer_function = general_transfer_fcn(wn=1.5,zeta=0,gain=1/2.25)
```

```
outputs = []

#Looping through different values of frequency and plotting the output time domain s
for freq in np.arange(1.4,1.6,0.05):
    #Simulating
    t,response,_ = sp.lsim(transfer_function,input_time(time_stamps,0.05,freq),time_
    outputs.append(response)

p3 = General_Plotter(r"$t$",r"$x$",r"Q3: System responses with variation of input fr
p3.general_plot(t,np.array(outputs).T,["Freq = ${:.2f}$".format(f) for f in np.arang

w,S,phi=sp.lti(*transfer_function).bode()

p4 = General_Plotter("Frequency in rad/s (log)","Magnitude in dB","Q3: Magnitude plo
p4.semilogx(w,S)

p5 = General_Plotter("Frequency in rad/s (log)","Phase in degrees","Q3: Phase plot")
p5.semilogx(w,phi)
```

The results are the following,

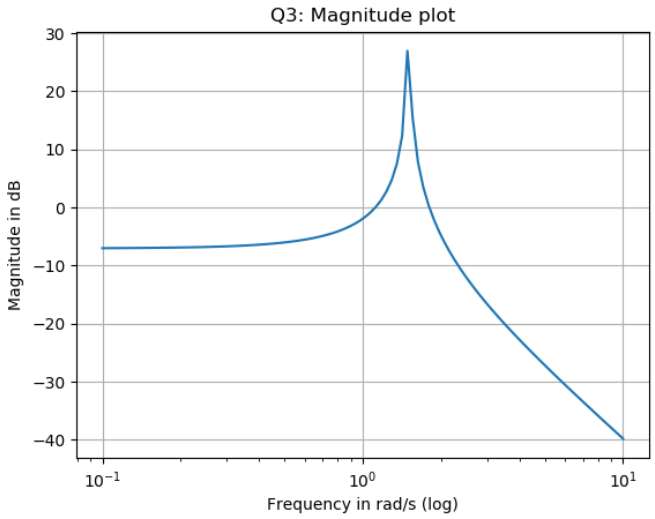

Figure 3: Magnitude plot

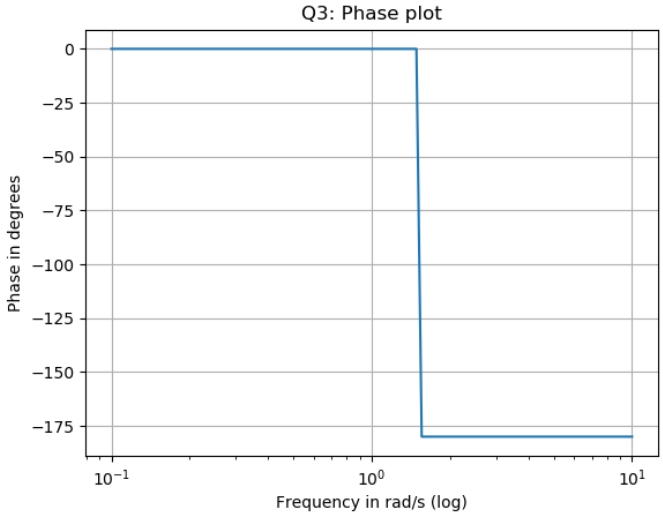

Figure 4: Phase plot

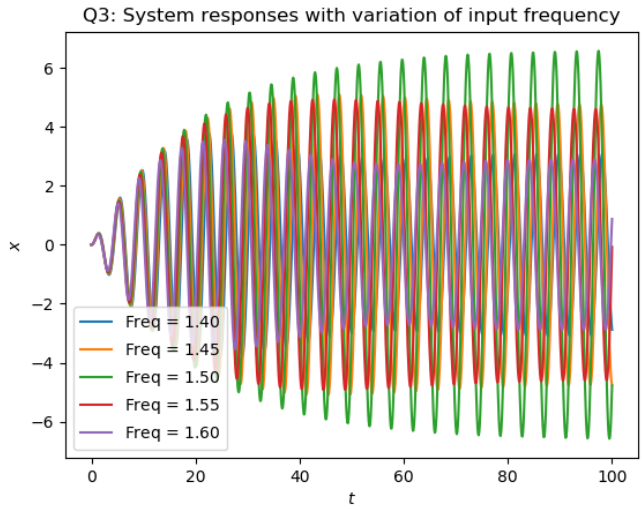

Figure 5: System response for different input frequencies

Coupled Spring System

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

```
#Question 4
def coupled_spring():
    #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)

    p6 = General_Plotter("Time","X","Q4: X vs time")
    p6.general_plot(response_x[0],response_x[1])

    p7 = General_Plotter("Time","Y","Q4: Y vs time")
    p7.general_plot(response_y[0],response_y[1])

    p8 = General_Plotter("Time","Displacement","Q4: Responses of coupled system")
    p8.general_plot(t,np.array([response_x[1],response_y[1]]).T,legend_txt=[r"$x(t)$

coupled_spring()
```

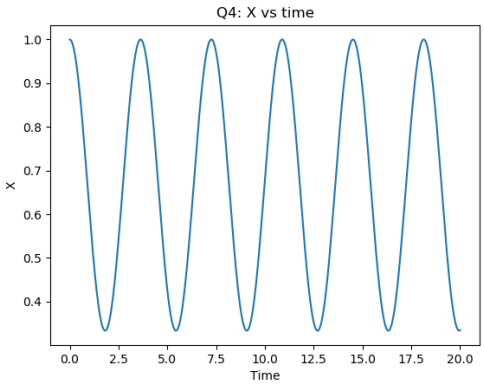

Figure 6: X vs time

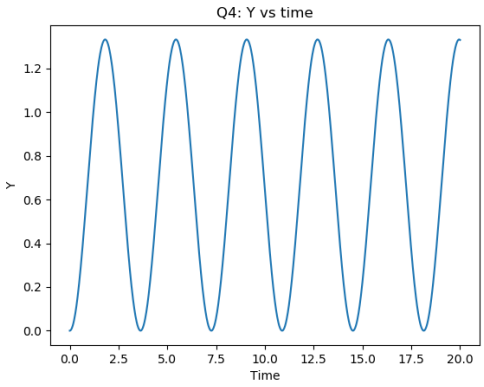

Figure 7: Y vs time

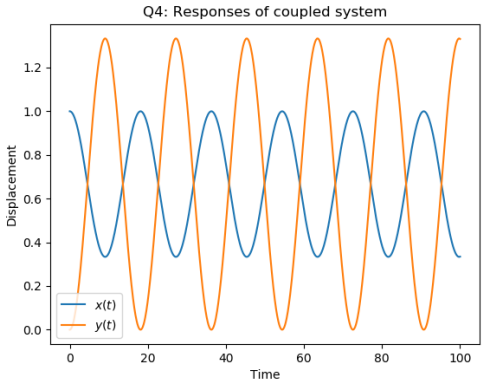

Figure 8: X,Y on same graph for comparison vs time

RLC Low Pass Filter

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

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

```
#Question 5
def two_port():
# Find the transfer function of the given circuit
    R = 100
    L = 1e-6
    C = 1e-6

    wn = 1/np.sqrt(L*C) # natural frequency
    Q = 1/R * np.sqrt(L/C) # quality factor
    zeta = 1/(2*Q) # damping constant

# transfer function
    n,d = general_transfer_fcn(gain=1,wn=wn,zeta=zeta)

    # make system
    transfer_function = sp.lti(n,d)

    # get bode plots
    w,S,phi=transfer_function.bode()

    p9 = General_Plotter("Frequency in rad/s (log)","Magnitude in dB","Q5: Magnitude
    p9.semilogx(w,S)

    p10 = General_Plotter("Frequency in rad/s (log)","Phase in degrees","Q5: Phase p
    p10.semilogx(w,phi)

    return transfer_function

transfer_function = two_port()
```

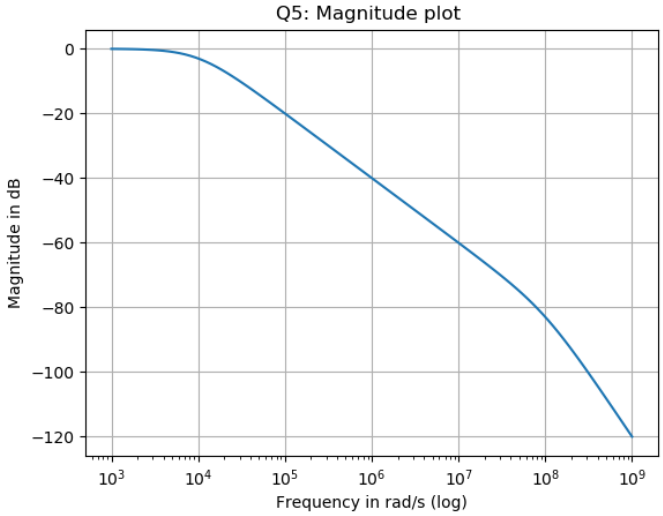

Figure 9: Magnitude Plot

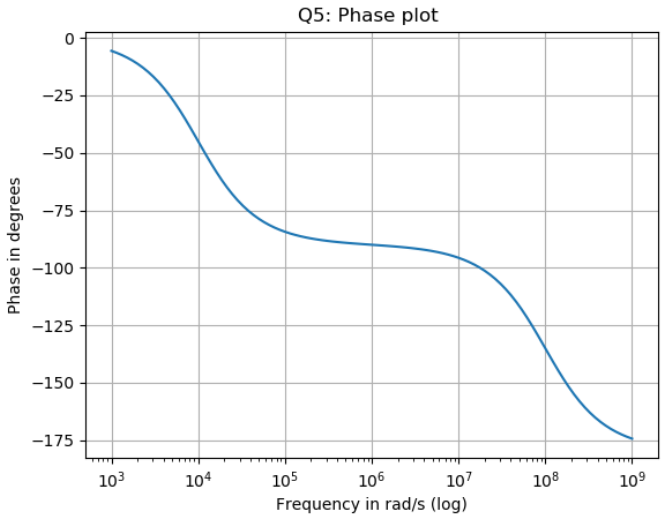

Figure 10: Phase plot

Multiple Frequencies in Input Voltage for LPF

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 $0 < t < 10ms$

```
#Question 6
```

```
def multiple_frequencies(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)
p11 = General_Plotter("Time$\longrightarrow$","Voltage$\longrightarrow$","Q6: Voltag
p11.general_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)
p12 = General_Plotter("Time$\longrightarrow$","Voltage$\longrightarrow$","Q6: Voltag
p12.general_plot(time,output_time)

multiple_frequencies(transfer_function)
```

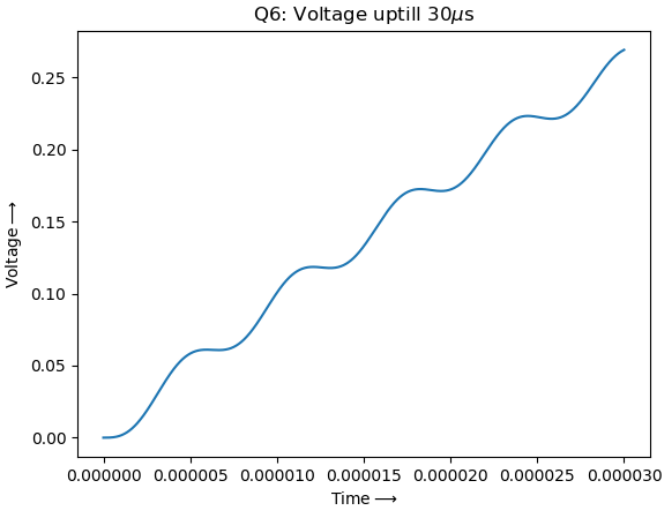

Figure 11: Voltage uptill 30 μs

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.

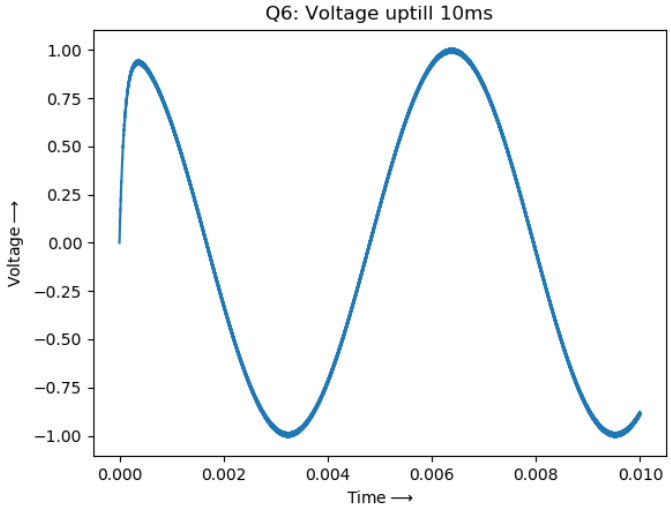

Figure 12: Voltage uptill 10ms

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.