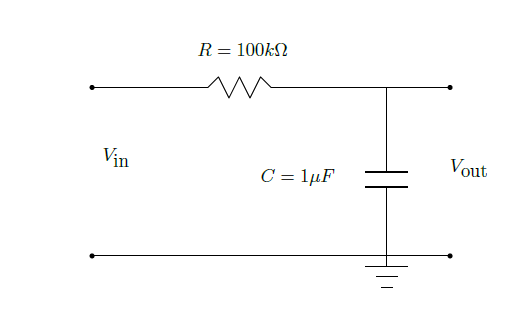
Python assignment 1

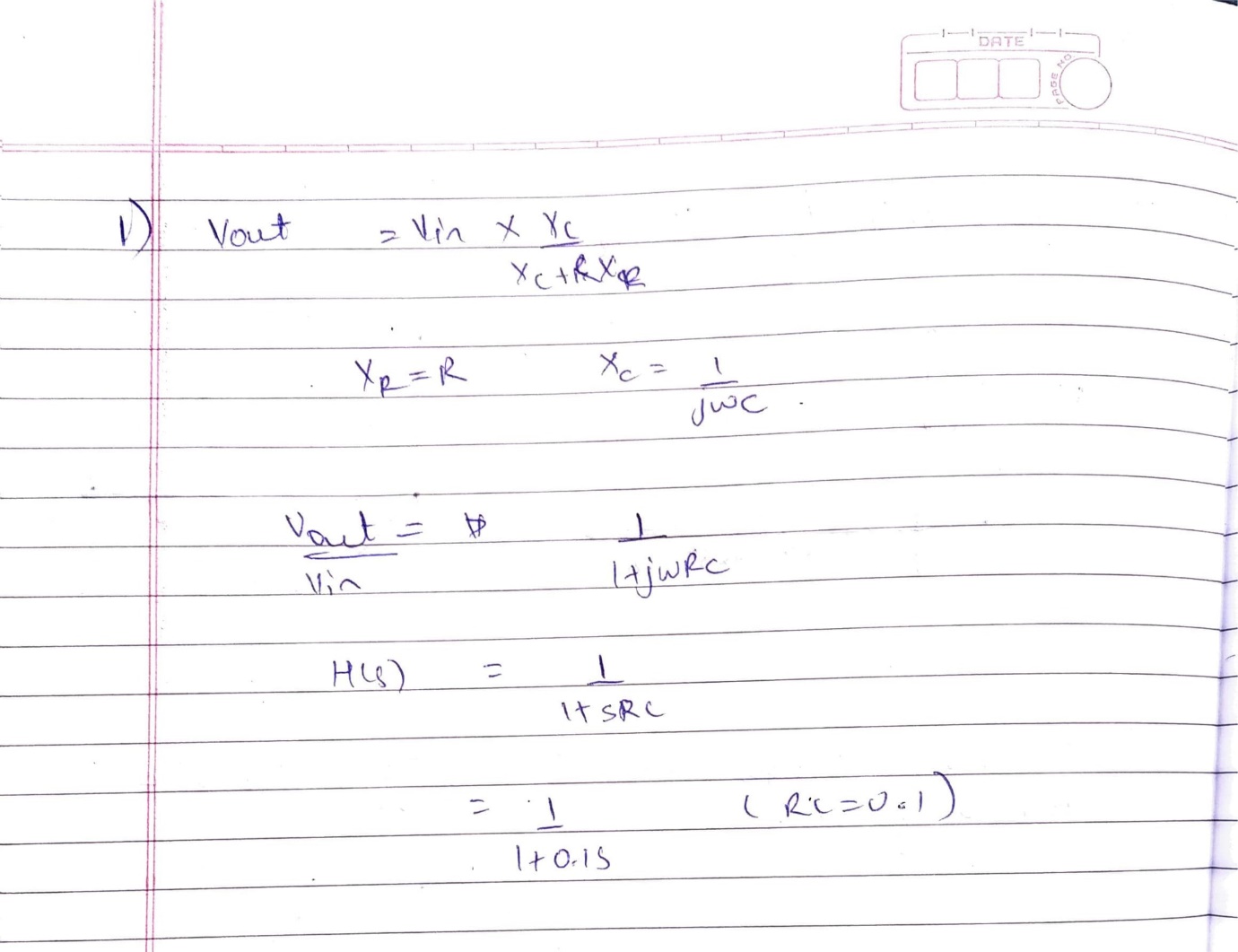
Kunal Keshav Damame

121901050

Q1] AIM:

Plot the magnitude and phase response of the transfer function of the integrator.Also generate Bode magnitude and phase plots of the following circuit.





Code:

import numpy as np

from scipy import signal

import matplotlib.pyplot as plt

#setting the numerator and denominator of the the transfer fucntion

numerator = [1]

denominator = [1,0.1]

#generate the frequency response with w

w,h = signal.freqresp((numerator,denominator))

#seprate the magnitude and phase of frequency response

magnitude = np.abs(h)

angle = np.angle(h)

#plot in subplot 1

plt.subplot(2,1,1)

plt.plot(w,magnitude,color="Orange")

plt.title("Frequency Response")

plt.legend(("Magnitude",),loc="upper right")

#plot in subplot 2

plt.subplot(2,1,2)

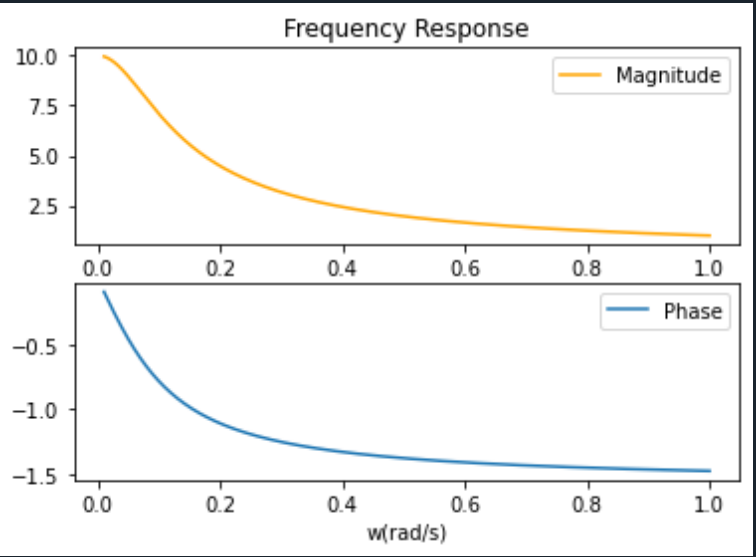
plt.plot(w,angle)

plt.xlabel("w(rad/s)")

plt.legend(("Phase",),loc="upper right")



PLOT



**Bode Plot and code**

from scipy import signal

import matplotlib.pyplot as plt

#setting the numerator and denominator of the the transfer fucntion

numerator = [0.1]

denominator = [1,0.1]

#generate the bode response with w

w,h,angle = signal.bode((numerator,denominator))

#plot the bode response

plt.subplot(2,1,1)

plt.semilogx(w,h,color="Orange")

plt.title("Bode Plot")

plt.legend(("Magnitude",),loc="upper right")

plt.ylabel("dB")

#plot the phase response

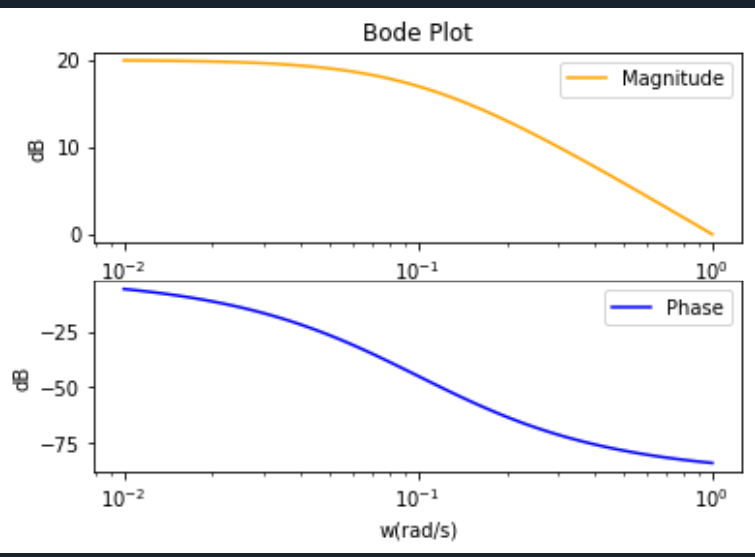
plt.subplot(2,1,2)

plt.semilogx(w,angle,color="blue")

plt.xlabel("w(rad/s)")

plt.legend(("Phase",),loc="upper right")

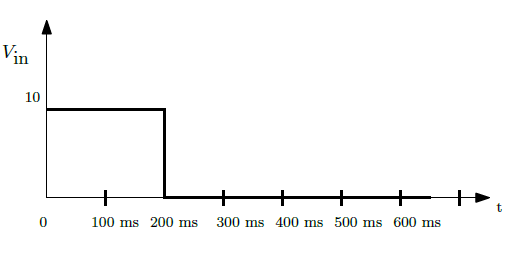
plt.ylabel("dB")



Note: Phase is returned in degrees

Q1] 1]

Plot the output voltage of the circuit for the following input signals



**Pulse Wave**

Code:

import numpy as np

from scipy import signal

import matplotlib.pyplot as plt

#setting the numerator and denominator of the the transfer fucntion

numerator = [0.1,0]

denominator = [0.1,1]

#generate some data points for the time

time = np.linspace(0,1,1000)

inp = np.zeros(1000)

#make the pulse wave

for i in range(1000):

if time[i] <=0.2:

inp[i] = 10

#generate the output voltage

tout,yout,xout = signal.lsim((numerator,denominator),U=inp,T=time)

#plot the voltage

plt.subplot(2, 1,1)

plt.plot(tout,5\*inp+5)

#plt.title("T=10RC")

plt.ylabel("Input V")

plt.subplot(2, 1,2)

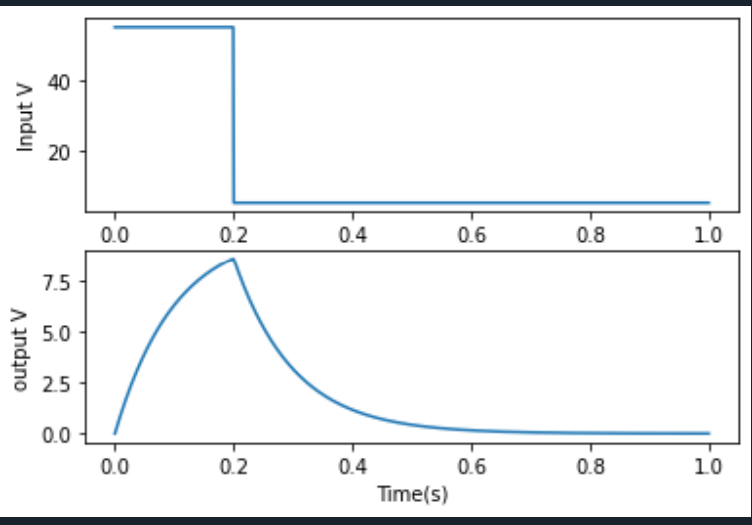
plt.plot(tout,yout)

plt.ylabel("output V")

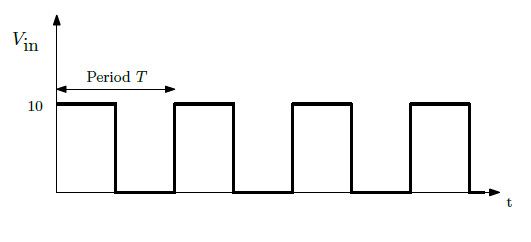
plt.xlabel("Time(s)")

plt.show()

PLOTS:



**Input square wave**

****

Code:

import numpy as np

from scipy import signal

import matplotlib.pyplot as plt

#setting the numerator and denominator of the the transfer fucntion

numerator = [0.1,0]

denominator = [0.1,1]

#generate some data points for the time

time = np.linspace(0,0.1,1000)

#generate the square signal

inp = signal.square(time\*np.pi\*2\*100 )

#generate the output using square input

tout,yout,xout = signal.lsim((numerator,denominator),U=5\*inp+5,T=time)

#plot the graph

plt.subplot(2, 1,1)

plt.plot(tout,5\*inp+5)

plt.title("T=RC")

plt.ylabel("Input V")

plt.subplot(2, 1,2)

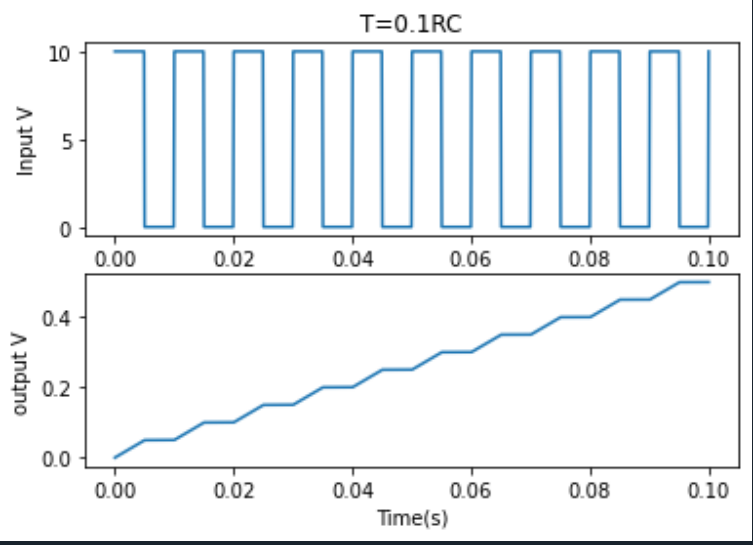
plt.plot(tout,yout)

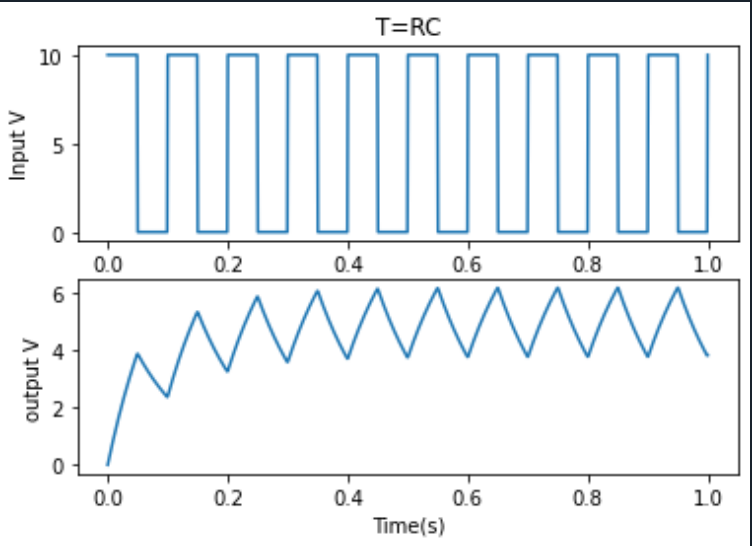
plt.ylabel("output V")

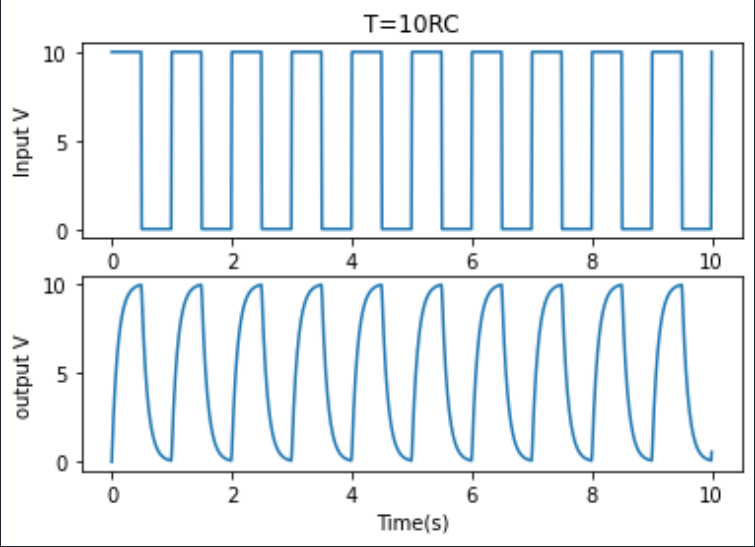
plt.xlabel("Time(s)")

plt.show()

Plots

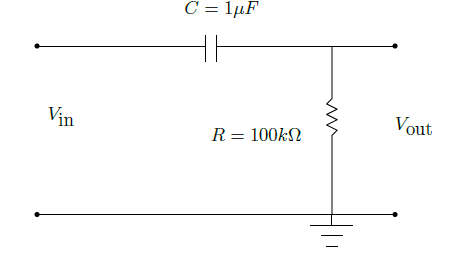


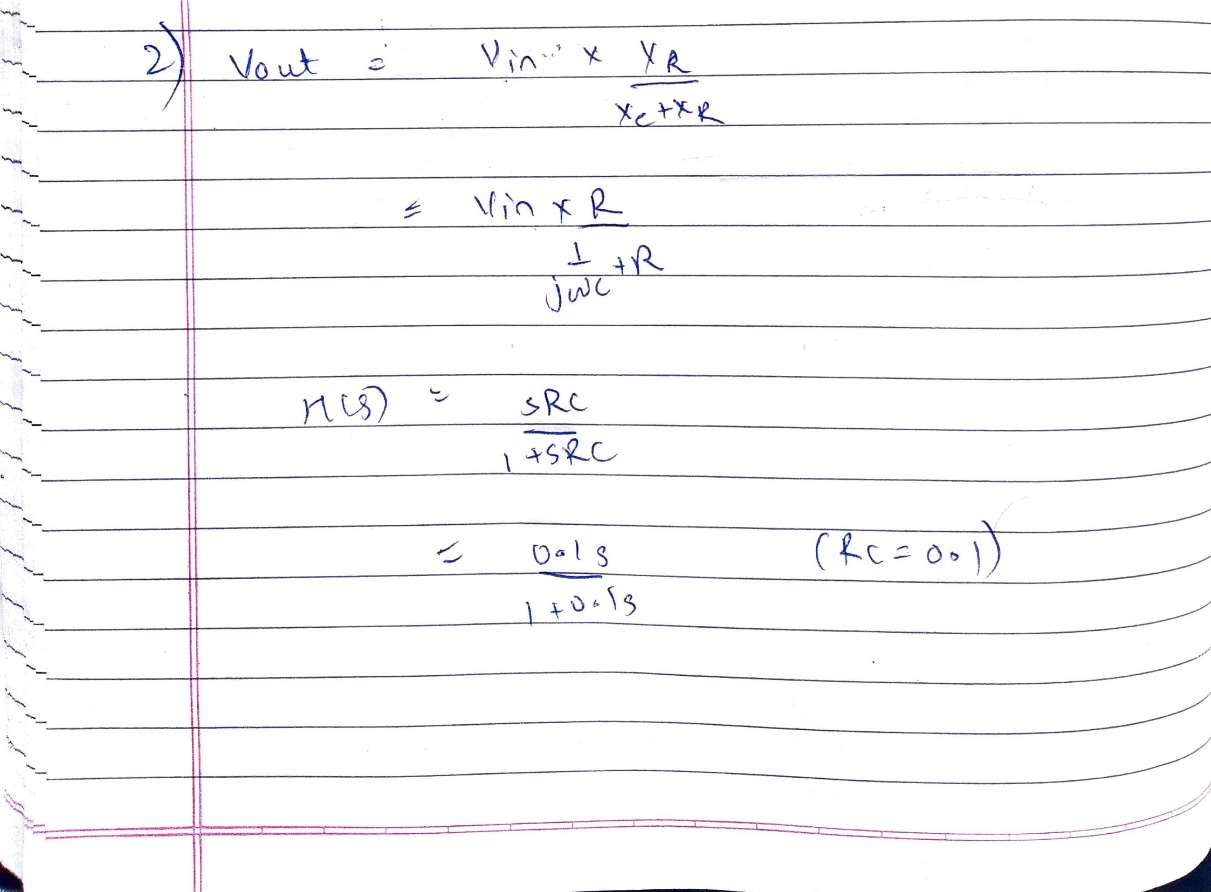




Q 2] AIM:

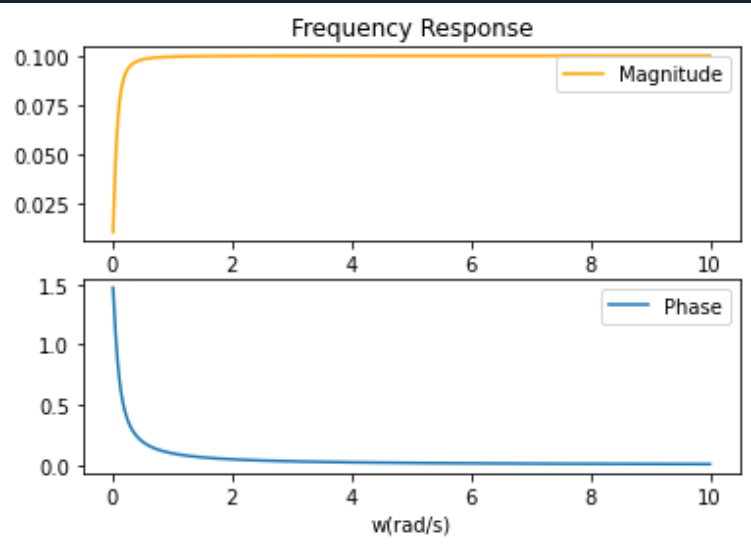
Plot the magnitude and phase response of the transfer function of the differentiator. Also generate Bode magnitude and phase plots of the following circuit.

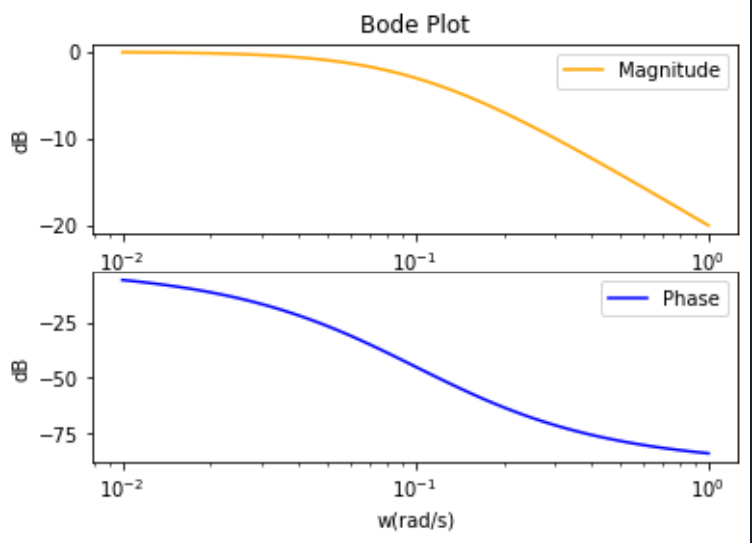




As all codes are same for q1 and q2 , the code is not given . the only difference is denominator is [0.1,0] instead of [1]

Plots:

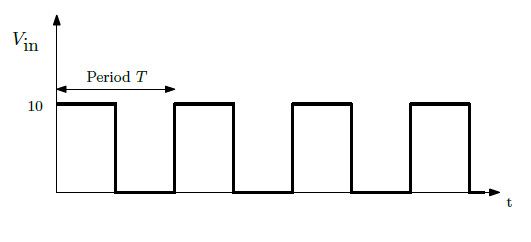


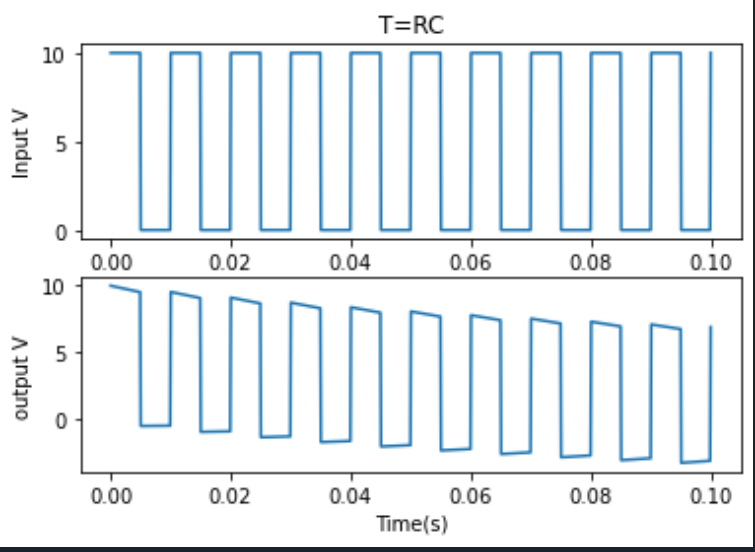


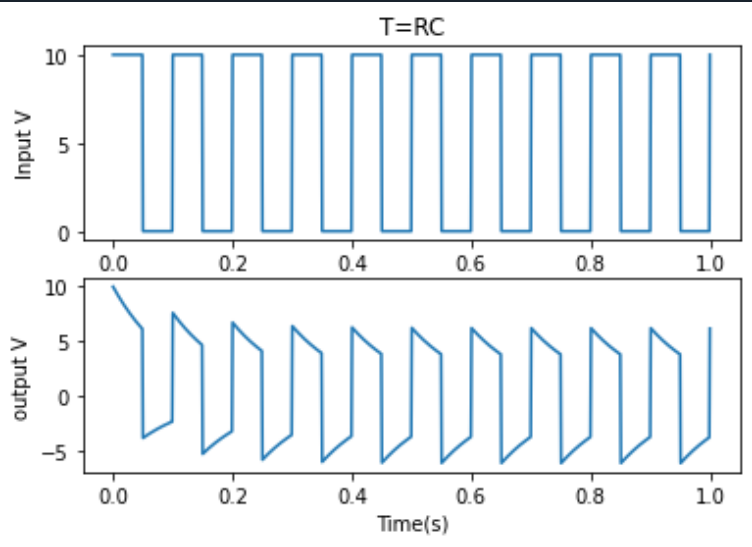
Q2]1]

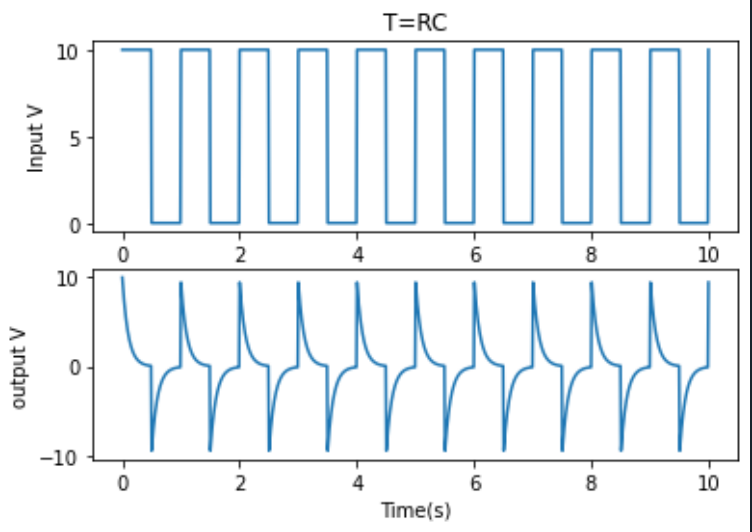
Plot the output voltage of the circuit for the following input signals

**Square wave**

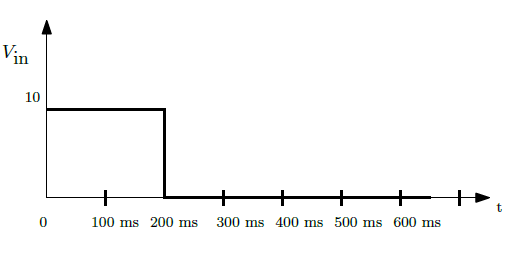
****



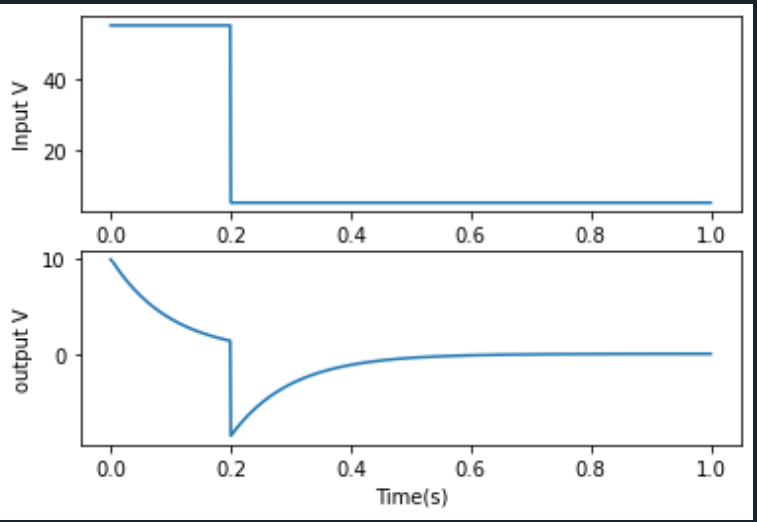




**Q2]2] Input Pulse**

****

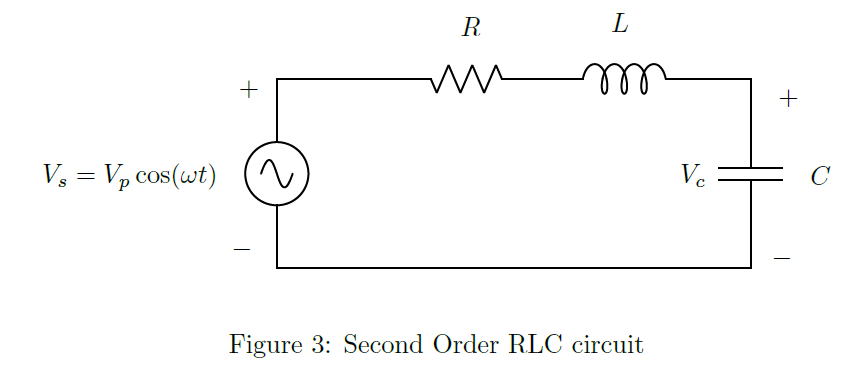
Plots:



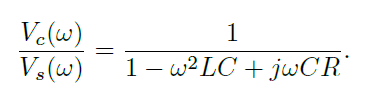
Q3]

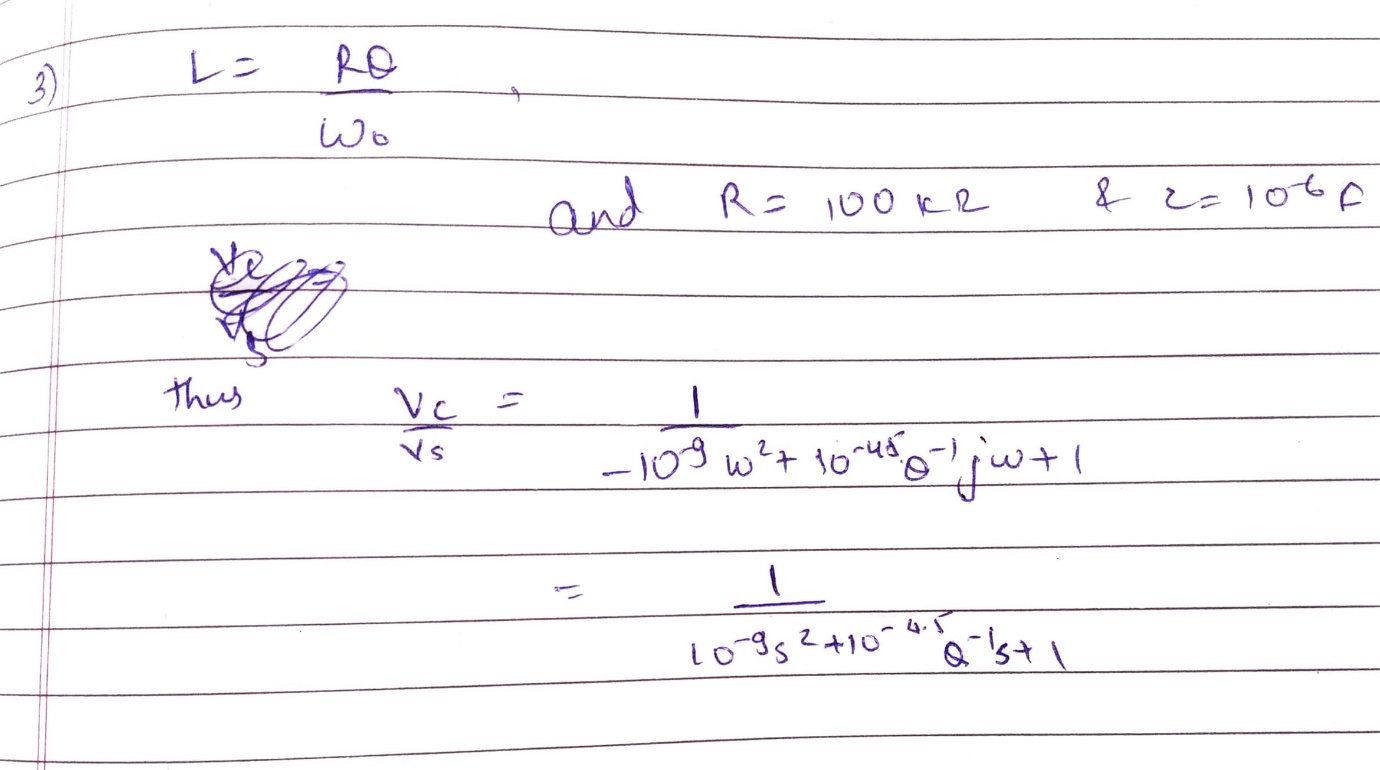
**AIM**:

Choose different values of Q for the following circuit and plot the response for underdamped , overdamped and critically damped condition.



Transfer function





**Frequency response and bode plot combined**

Code:

import numpy as np

from scipy import signal

import matplotlib.pyplot as plt

#setting the numerator and denominator of the the transfer fucntions

numerator = [1]

denominator=[10\*\*-9,(10\*\*-4.5)/(2\*\*(-0.5)),1]

#generate the frequency response and the bode plot

w,h = signal.freqresp((numerator,denominator))

w\_bode , y\_bode , x\_bode = signal.bode((numerator,denominator),w=w)

#plot the frequency response

plt.figure()

plt.plot(w,np.angle(h),w,np.abs(h))

plt.title("frquency response")

plt.legend(("Phase","magnitude"),loc="upper right")

plt.xlabel("w(rad/s)")

#plot the bode plot vs frequency response

plt.figure()

plt.plot(w,np.abs(h),w\_bode,y\_bode)

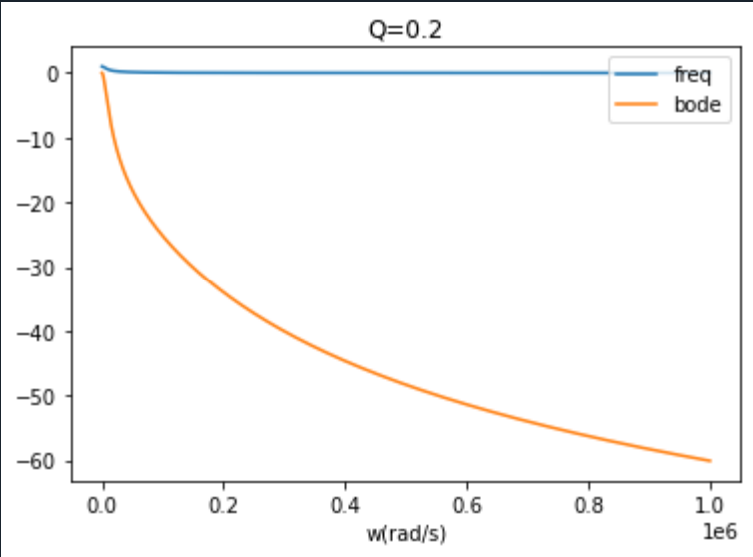
plt.legend(("freq","bode"),loc="upper right")

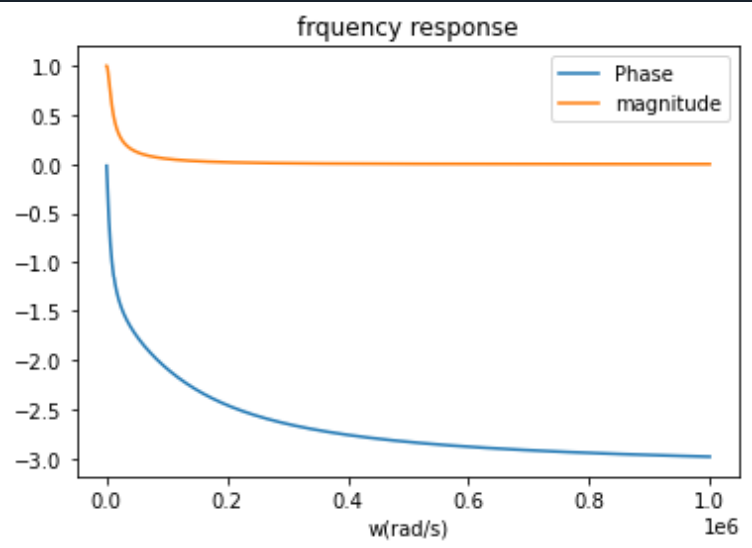
plt.xlabel("w(rad/s)")

plt.title("Q=-0.5")

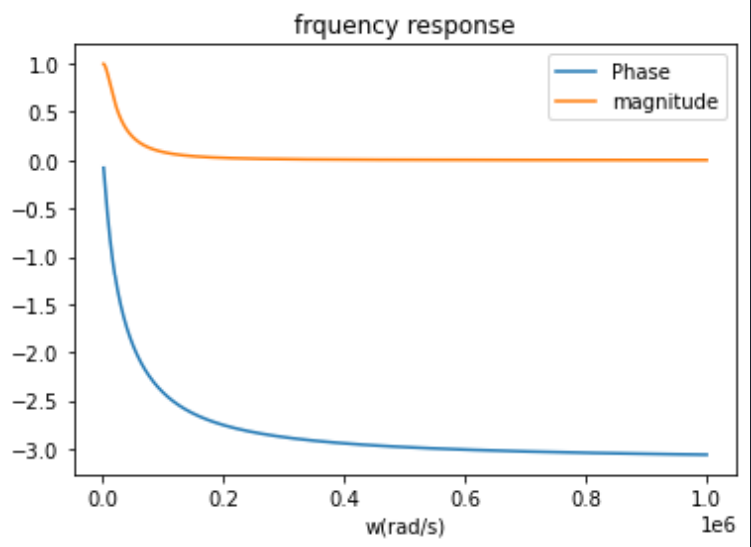
\*The codes for all other values of q are same but only diff is value of q is changed \*

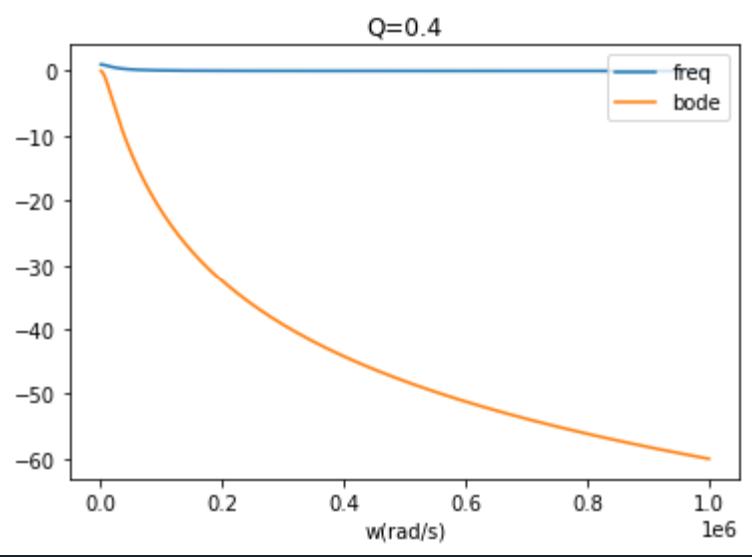
**Overdamped**



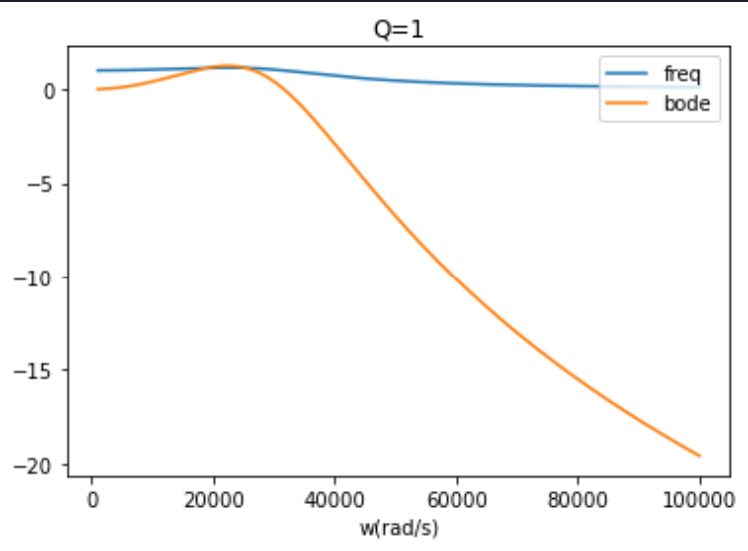


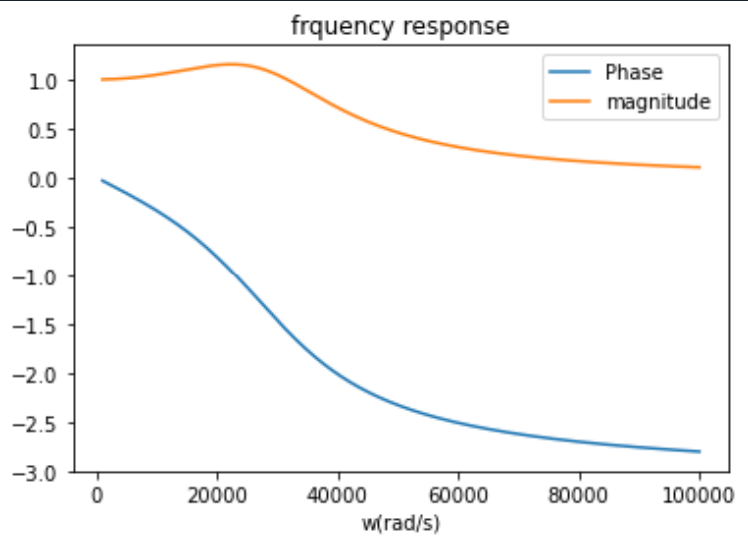
Q=0.4



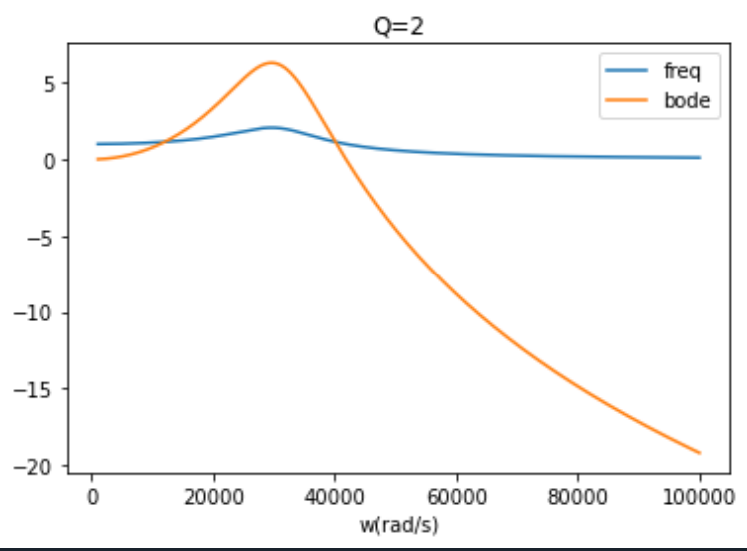


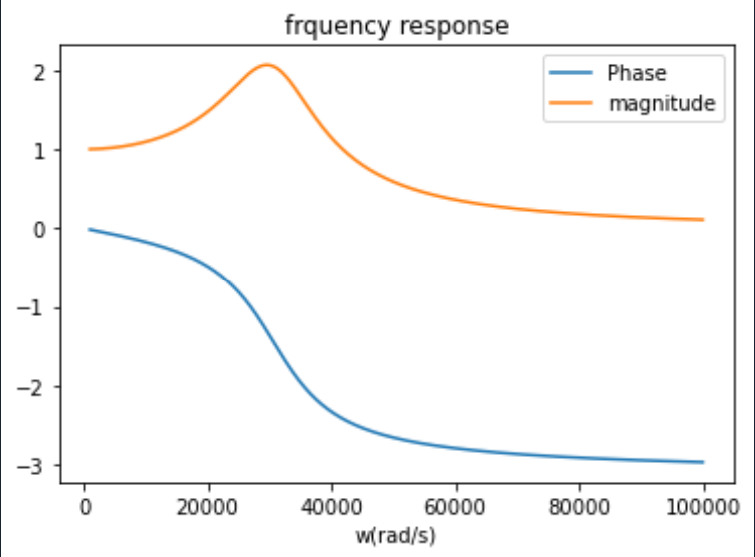
**Underdamped** Q=1



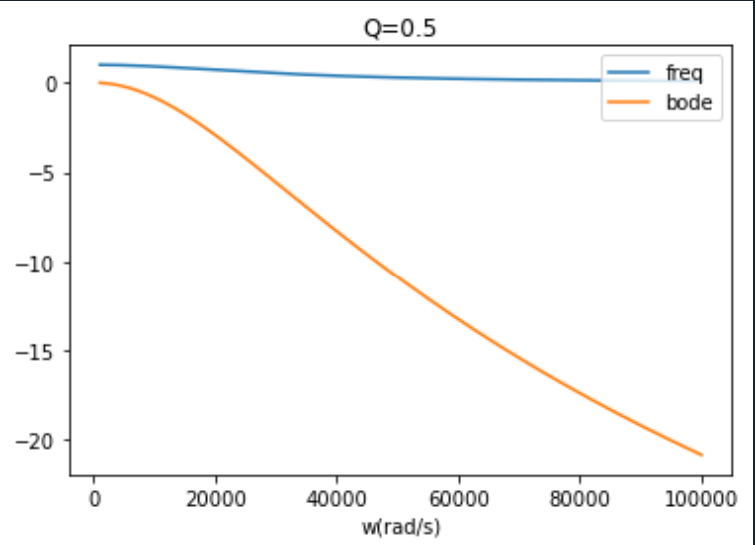


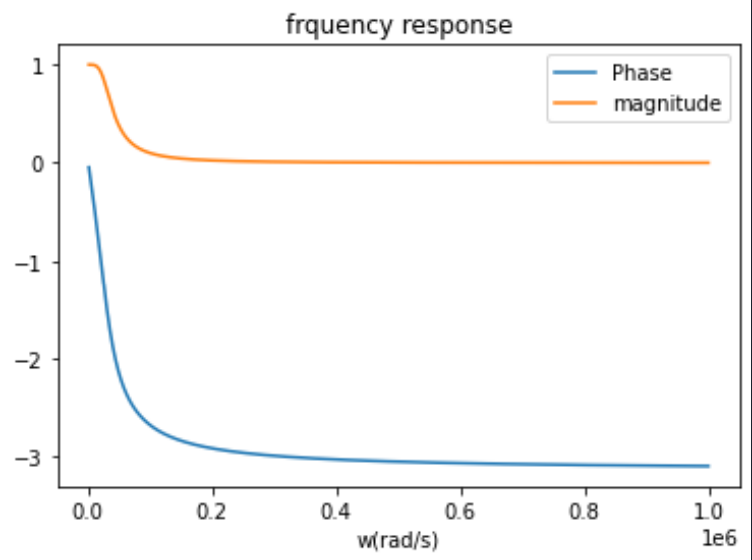
Q = 2





**Critically Damped**





Q3] 2]

AIM:.

step response of the above circuit for the cases mentioned earlier.

Code

from scipy import signal

import numpy as np

import matplotlib.pyplot as plt

#setting the numerator and denominator of the the transfer fucntions

numerator = [1]

denominator\_1 = [10\*\*-9,(10\*\*-4.5)/0.2,1]

denominator\_2 = [10\*\*-9,(10\*\*-4.5)/0.4,1]

denominator\_3 = [10\*\*-9,(10\*\*-4.5)/0.5,1]

denominator\_4 = [10\*\*-9,(10\*\*-4.5)/1,1]

denominator\_5 = [10\*\*-9,(10\*\*-4.5)/2,1]

denominator\_6 = [10\*\*-9,(10\*\*-4.5)/2\*\*-0.5,1]

#getting the step response of the functions

t,s1 = signal.step((numerator,denominator\_1),T=np.linspace(0,0.0025,1000))

t,s2 = signal.step((numerator,denominator\_2),T=np.linspace(0,0.0025,1000))

t,s3 = signal.step((numerator,denominator\_3),T=np.linspace(0,0.0025,1000))

t,s4 = signal.step((numerator,denominator\_4),T=np.linspace(0,0.0025,1000))

t,s5 = signal.step((numerator,denominator\_5),T=np.linspace(0,0.0025,1000))

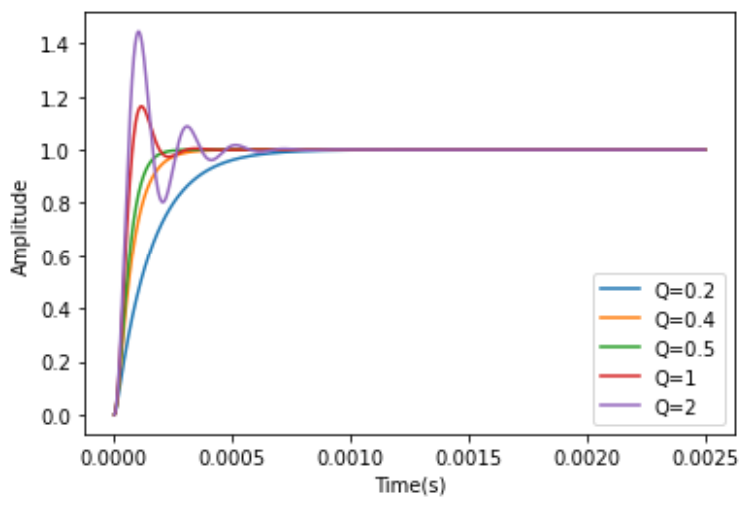
t,s6 = signal.step((numerator,denominator\_6),T=np.linspace(0,0.0025,1000))

#ploting the transient response

plt.plot(t,s1,t,s2,t,s3,t,s4,t,s5)

plt.xlabel("Time(s)")

plt.ylabel("Amplitude")



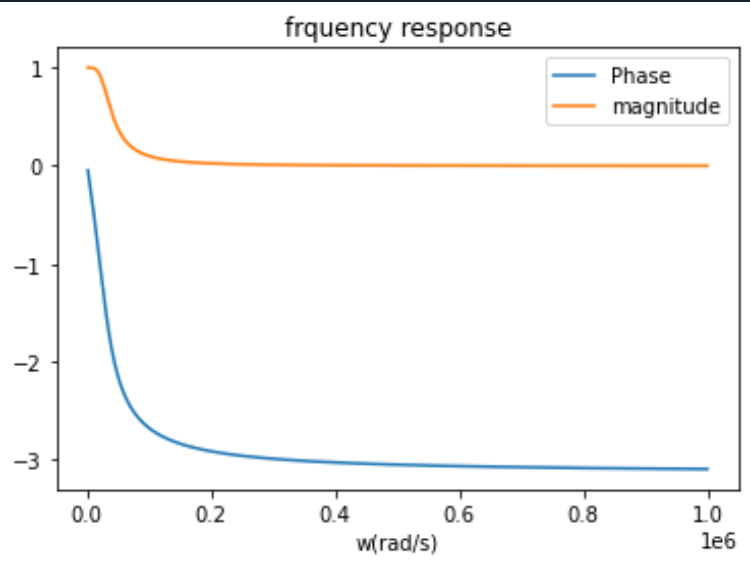
3] 3.]

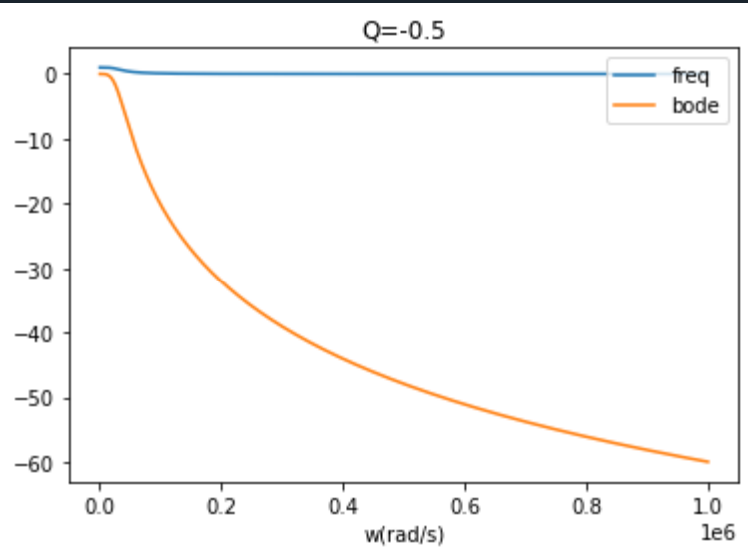
AIM:

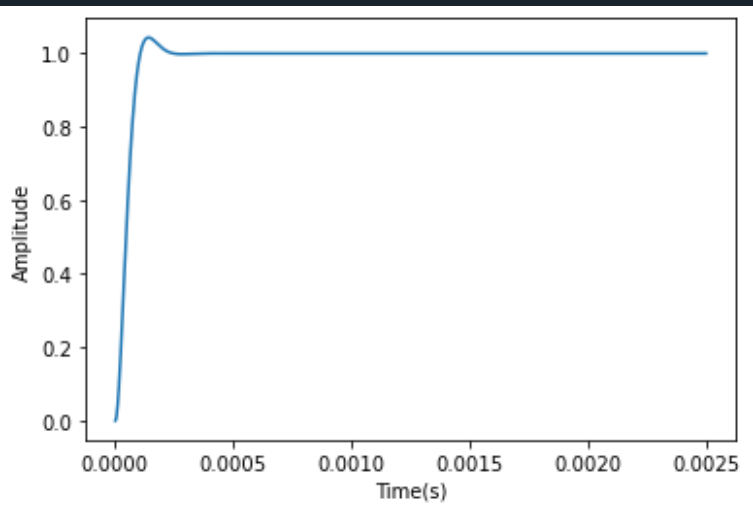
Now repeat the above analysis for q = 1/sqrt(2)

Note-

For the codes are not shown as they are same as the above parts except the change in q







The shape of frequency response starts to change after q=1/sqrt(2) as there is peak for all q greater than this value.