Python assignment 1

Kunal Keshav Damame

121901050

1)

1) $$V_{out} = V_{in} \times \frac{X_C}{X_C + X_R}$$

$$X_R = R \qquad X_C = \frac{1}{j\omega C}$$

$$\frac{V_{out}}{V_{in}} = \frac{1}{1 + j\omega RC}$$

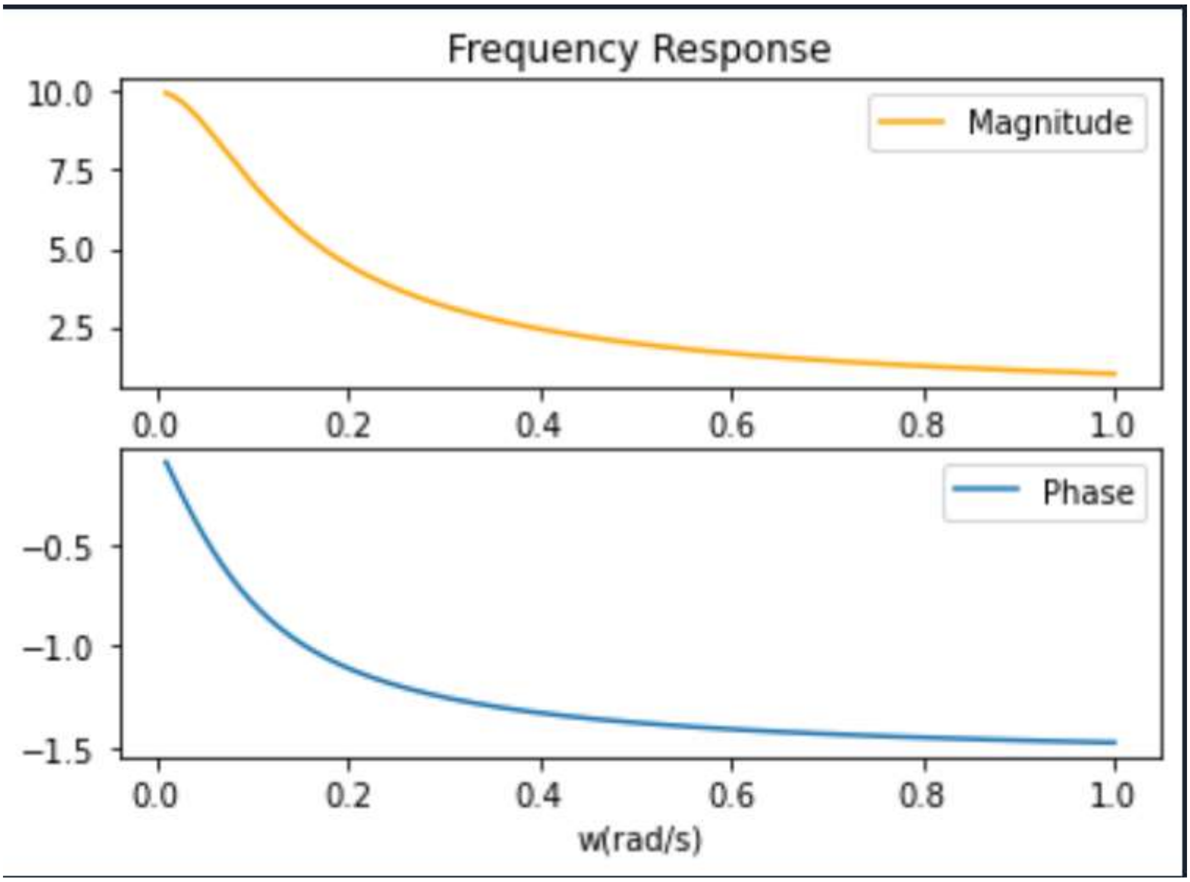
$$H(s) = \frac{1}{1 + sRC}$$

$$= \frac{1}{1 + 0.1s} \qquad (RC = 0.1)$$

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")
```

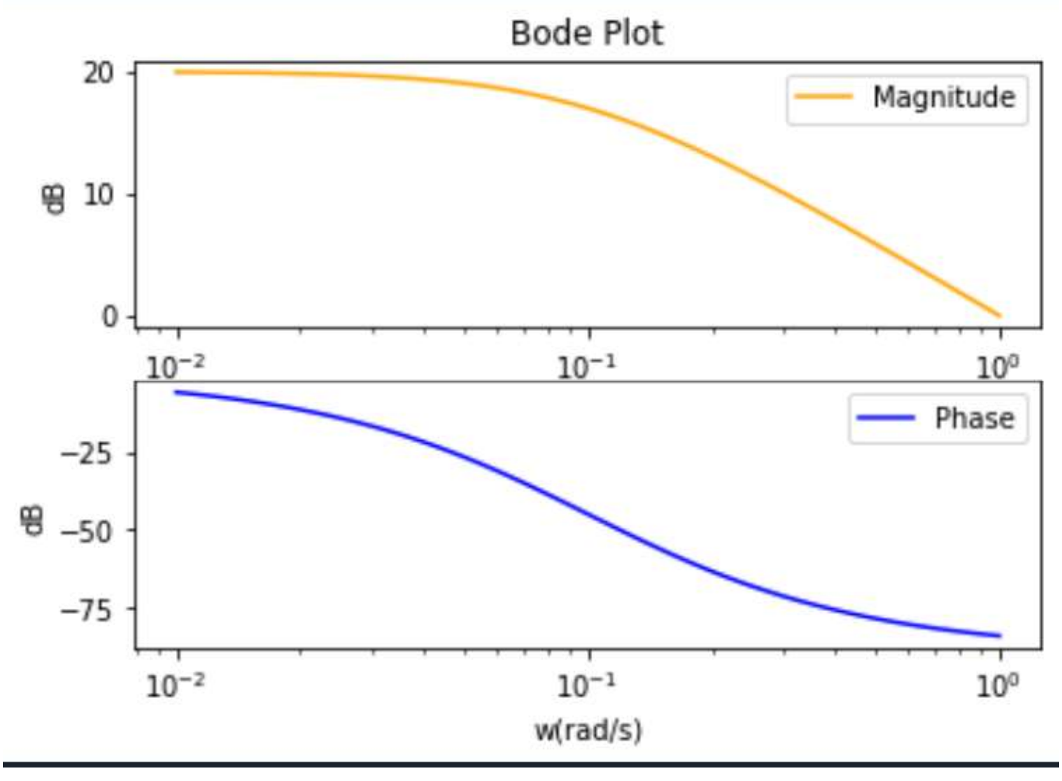
PLOT



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")
```



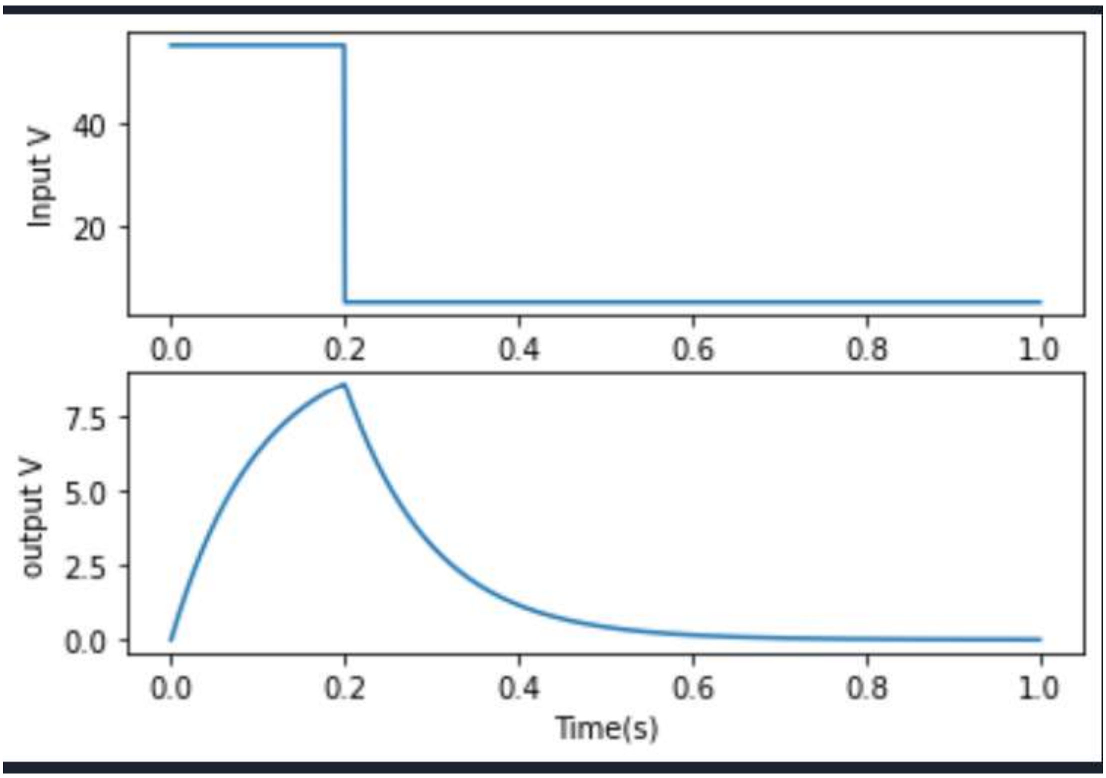
Phase is returned in degrees

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()
```

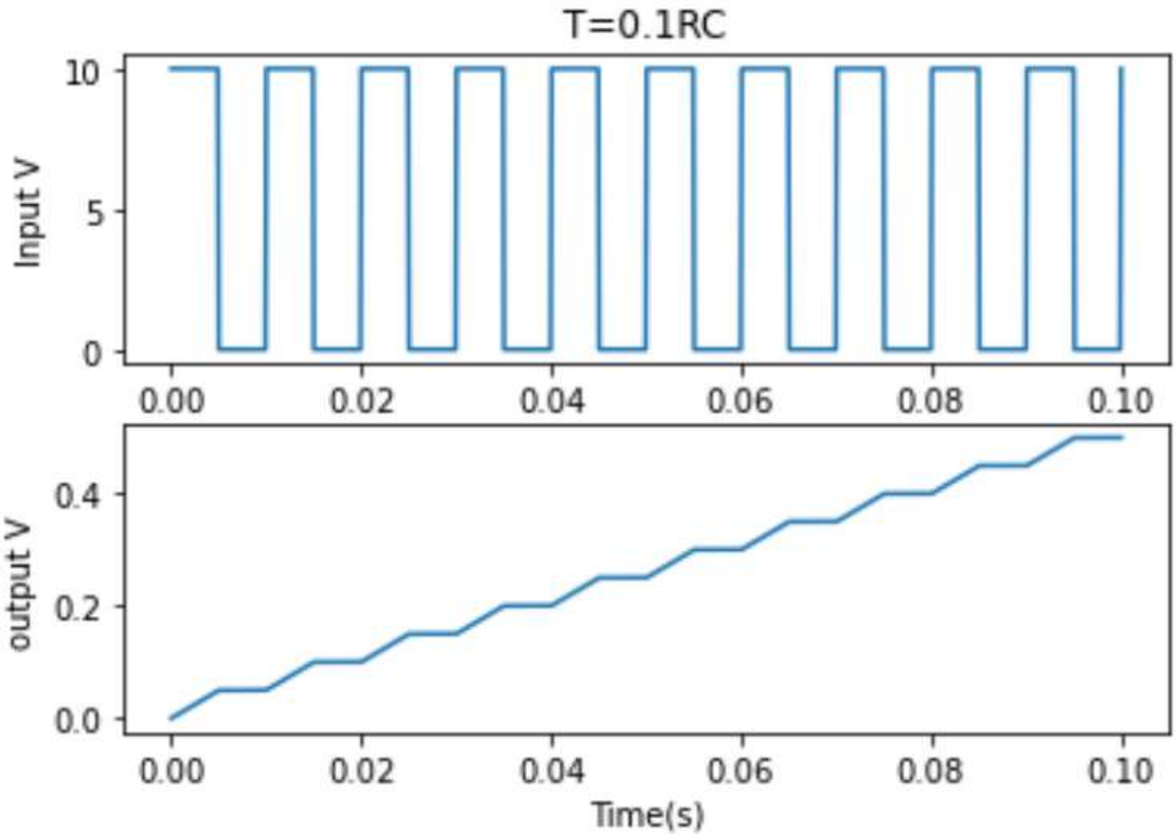


Input square wave

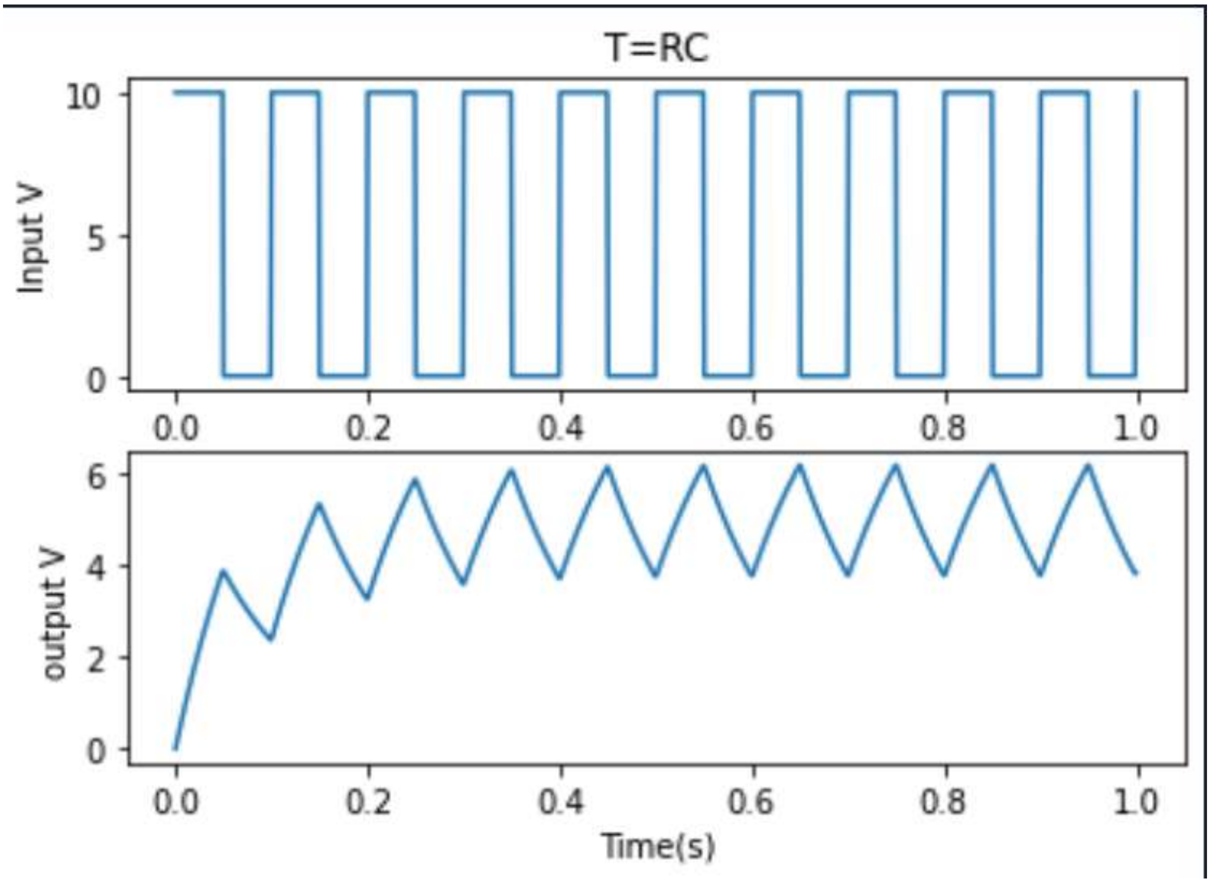
Code:

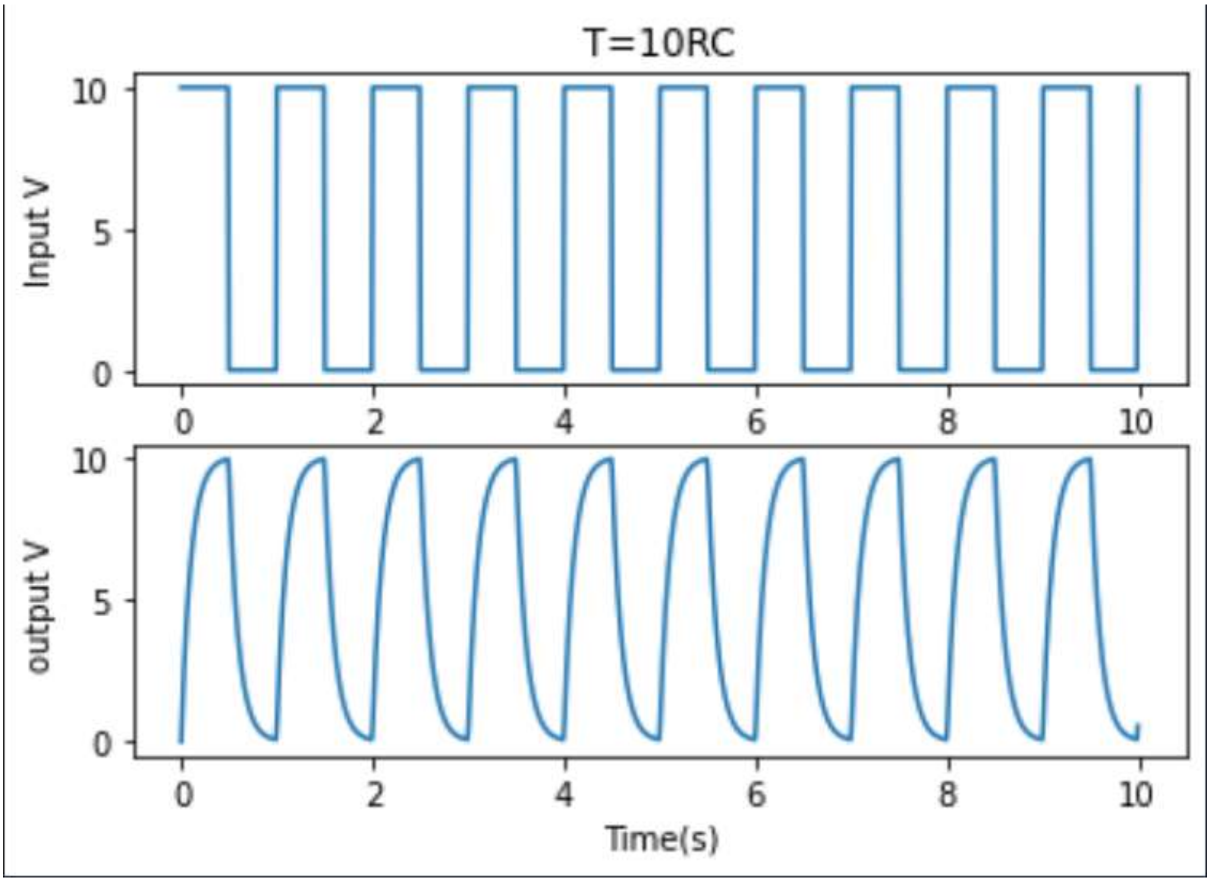
```
import numpy as np

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



T=0.1RC
Input V
10
5
0
0.00
0.02
0.04
0.06
0.08
0.10
output V
0.4
0.2
0.0
0.00
0.02
0.04
0.06
0.08
0.10
Time(s)


T=RC
Input V
output V
Time(s)


T=10RC
Input V
output V
Time(s)

Q 2

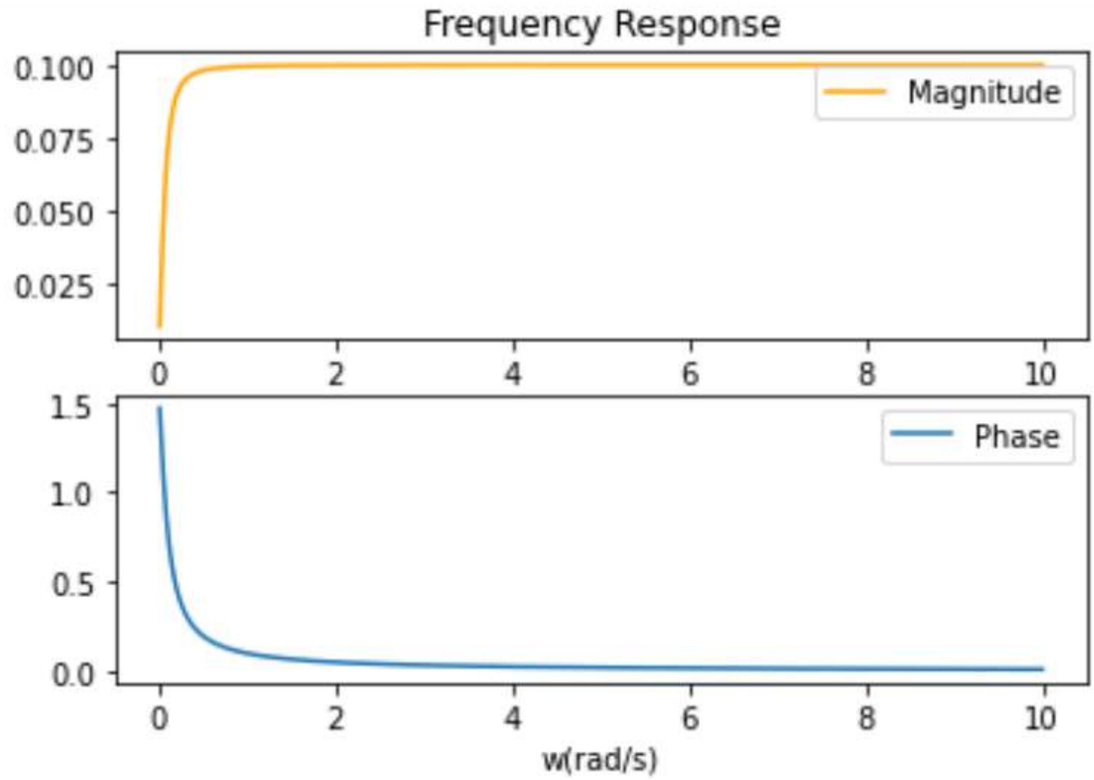
2) $V_{out} = V_{in} \times \frac{X_R}{X_C + X_R}$

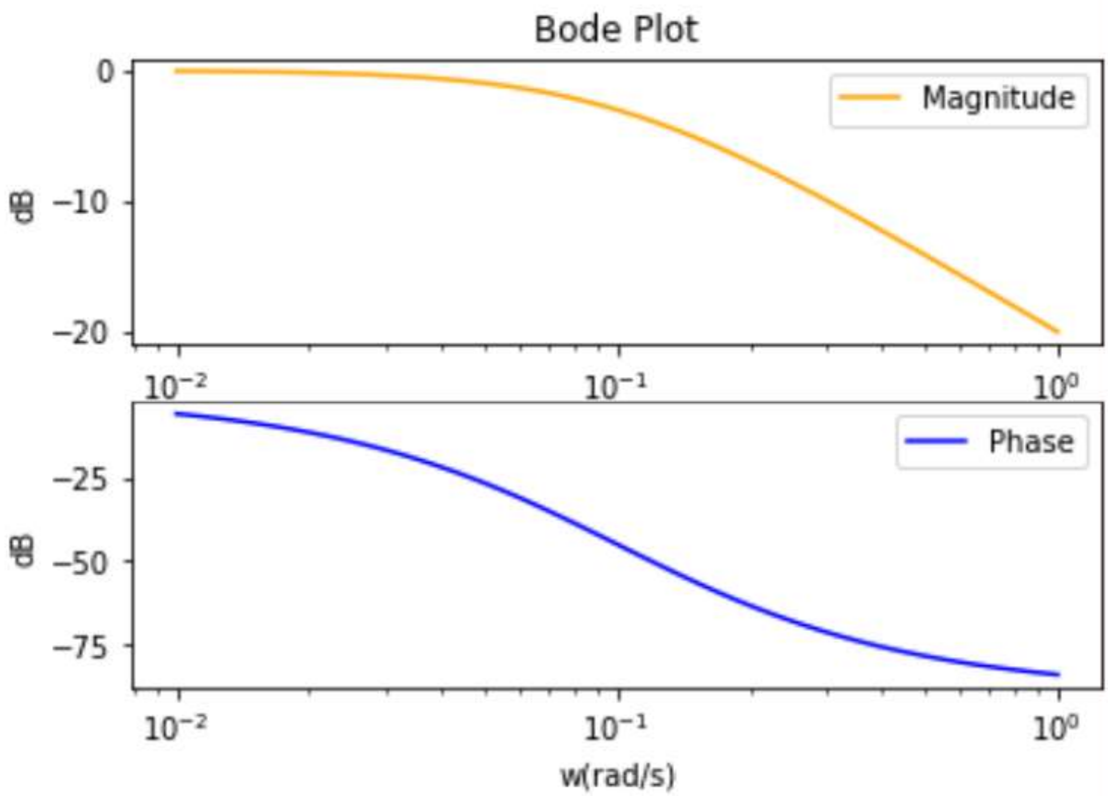
$= V_{in} \times \frac{R}{\frac{1}{j\omega C} + R}$

$H(s) = \frac{sRC}{1 + sRC}$

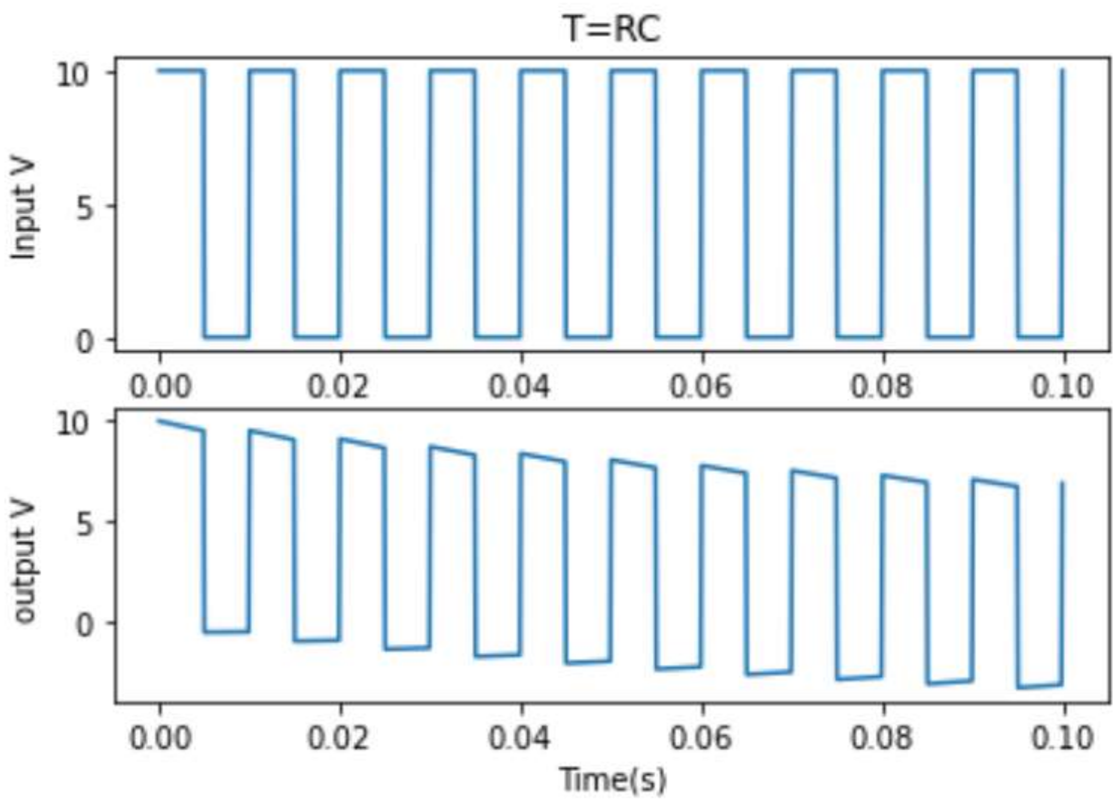
$= \frac{0.1s}{1 + 0.1s}$ $(RC = 0.1)$

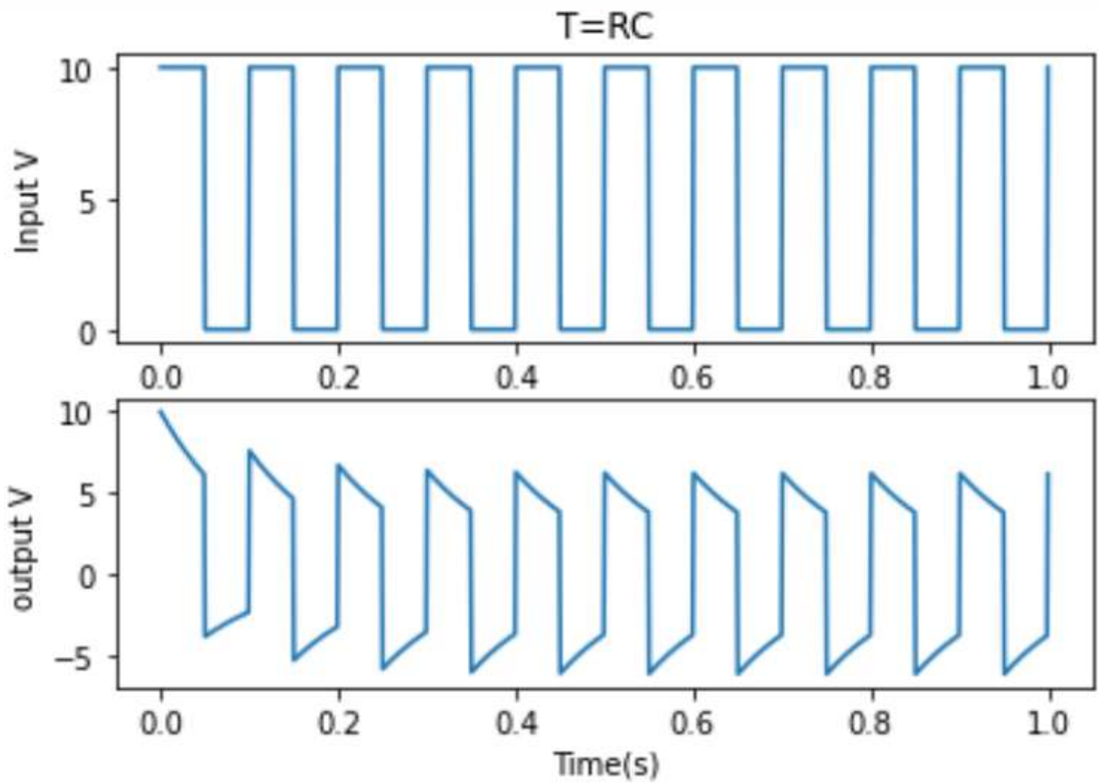
As all codes are same for q1 and q2 , the code is not given . the only difference is denominator is [0.1,0] instead of [1]

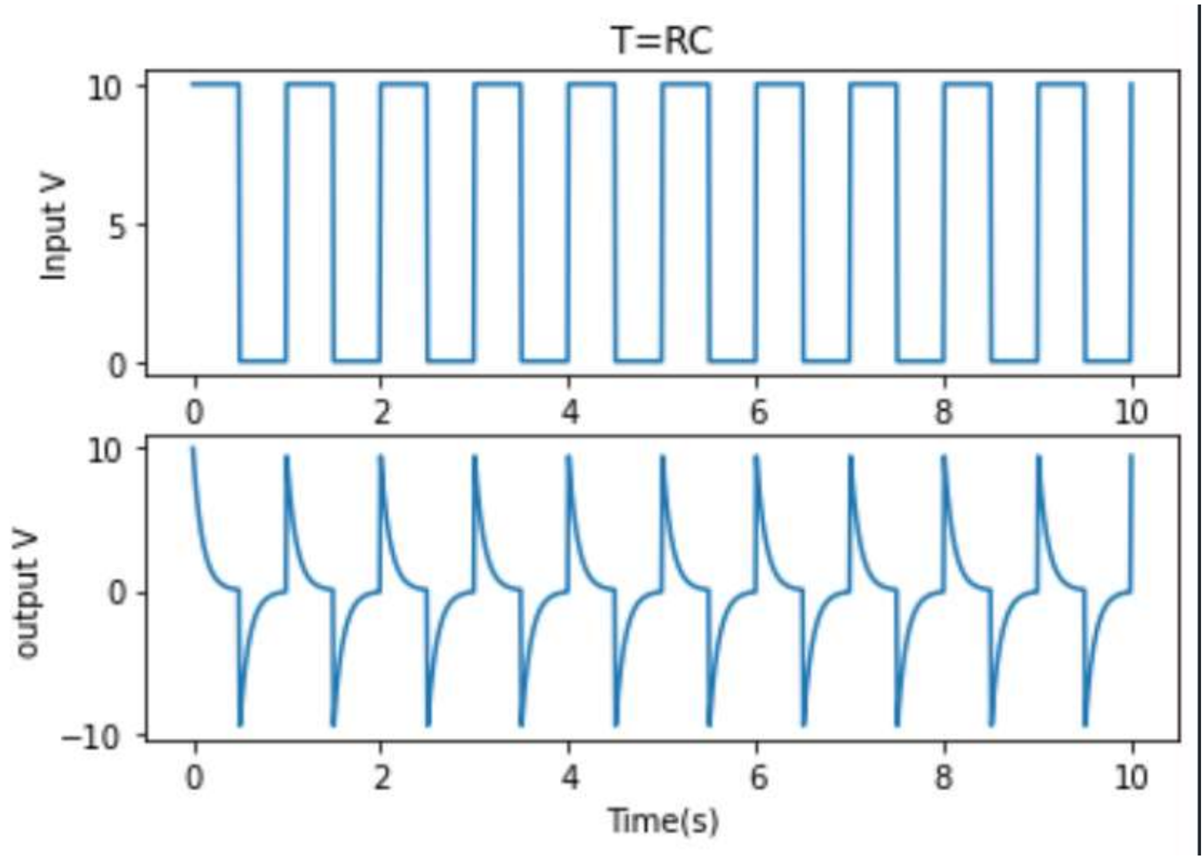




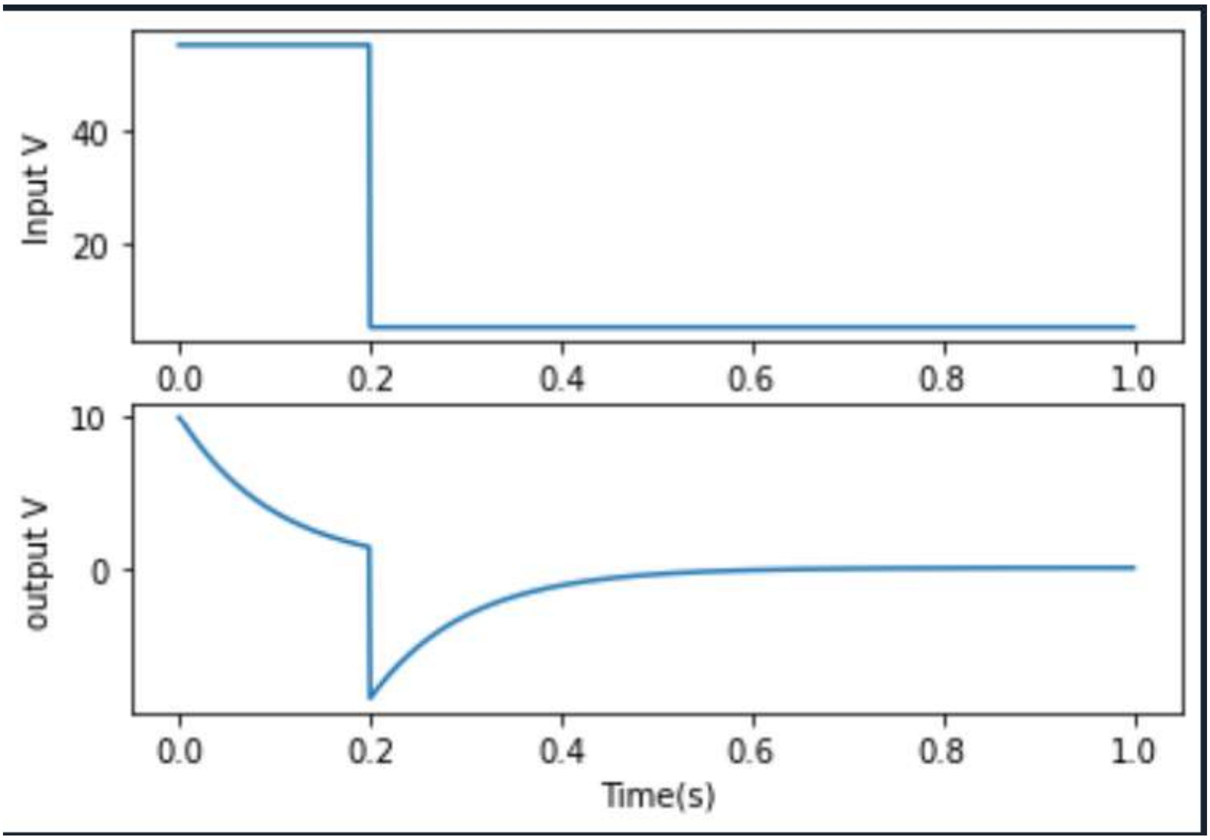
Square wave







Input Pulse



Q3

3) $L = \frac{RQ}{\omega_0}$

and $R = 100\,k\Omega$ & $C = 10^{-6} F$

thus $\frac{V_c}{V_s} = \frac{1}{-10^{-9}\omega^2 + 10^{-4.5}Q^{-1}j\omega + 1}$

$= \frac{1}{10^{-9}s^2 + 10^{-4.5}Q^{-1}s + 1}$

Frequency response and bode plot combined

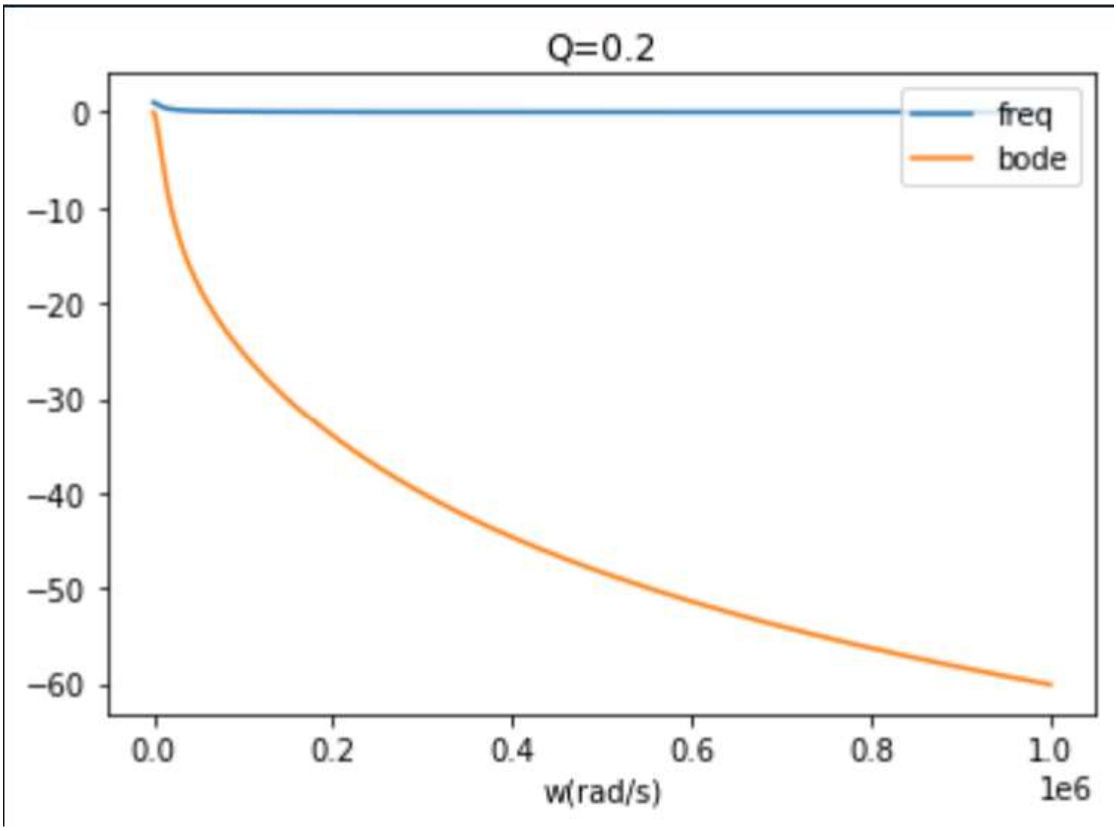
Code:

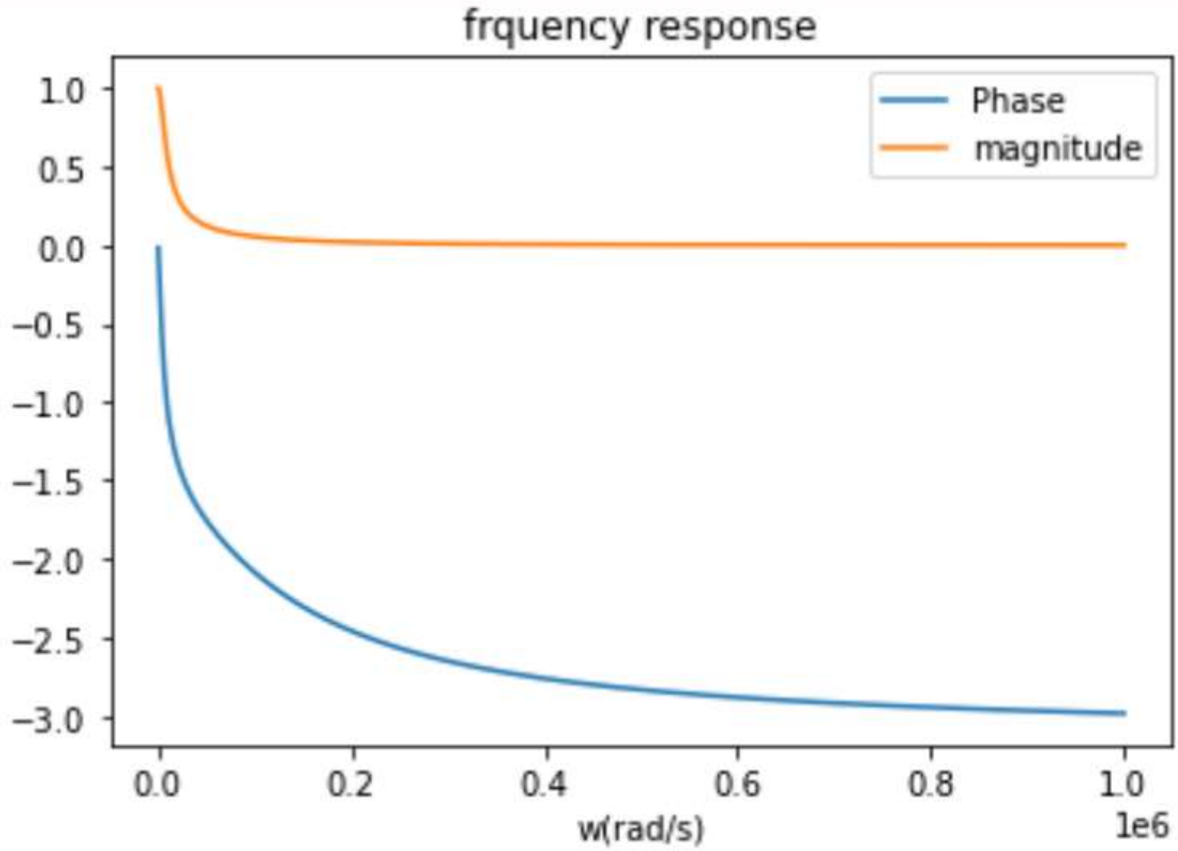
```
import numpy as np

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")
```

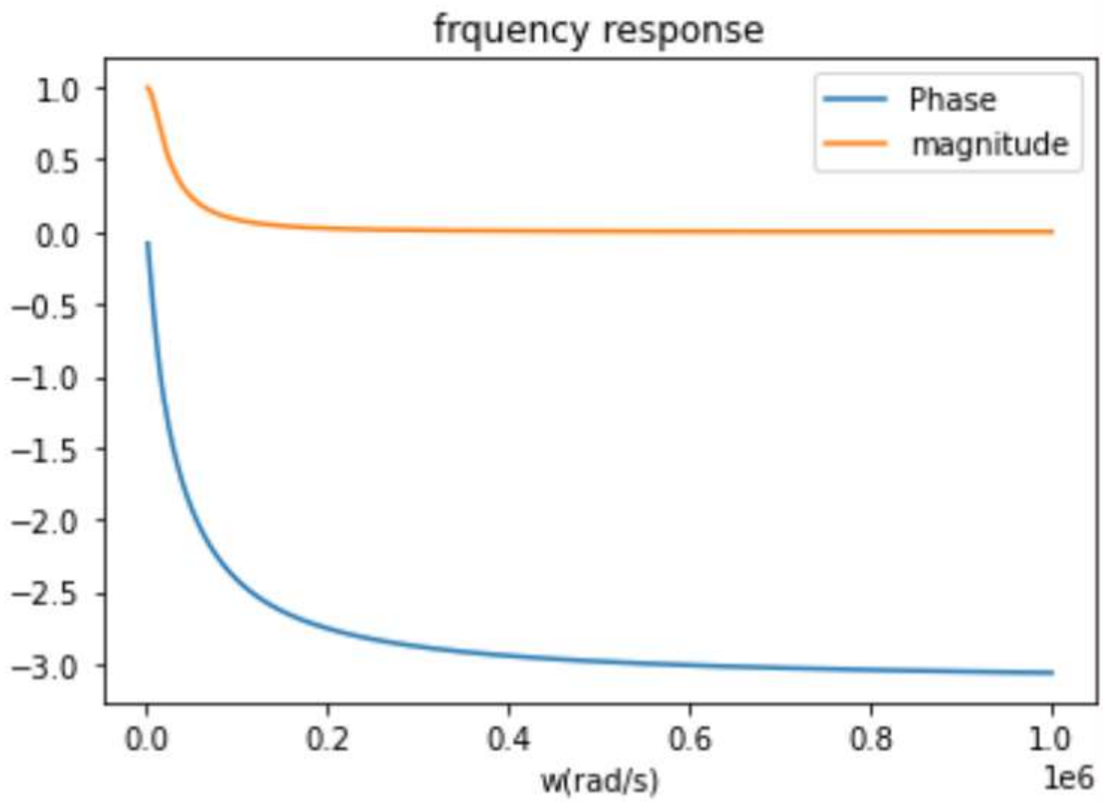
The codes for all other values of q are same but only diff is denominator is changed

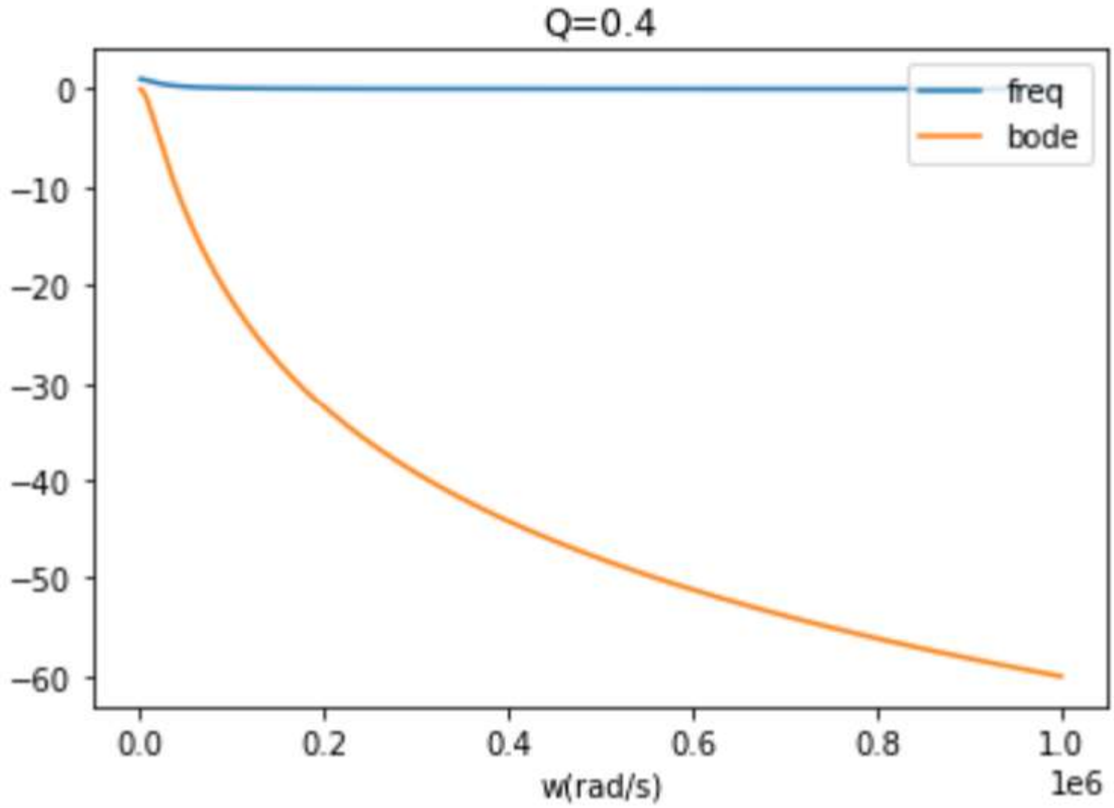
Overdamped



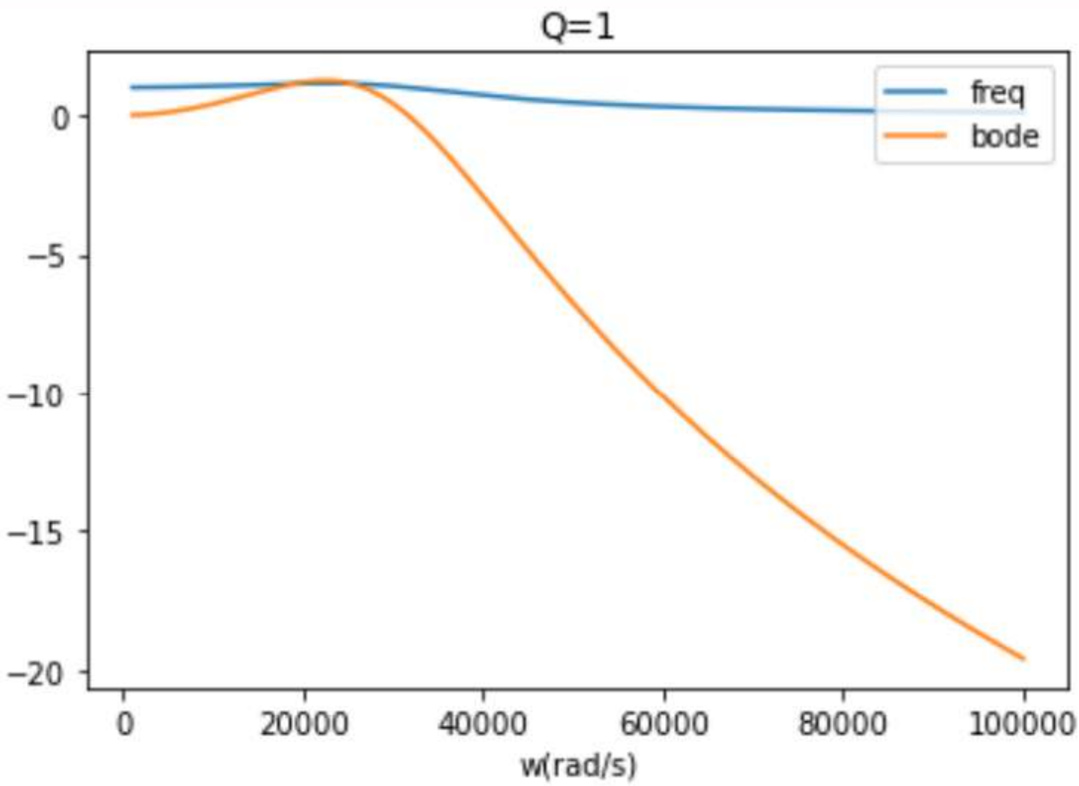

frquency response
Phase
magnitude
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
w(rad/s)

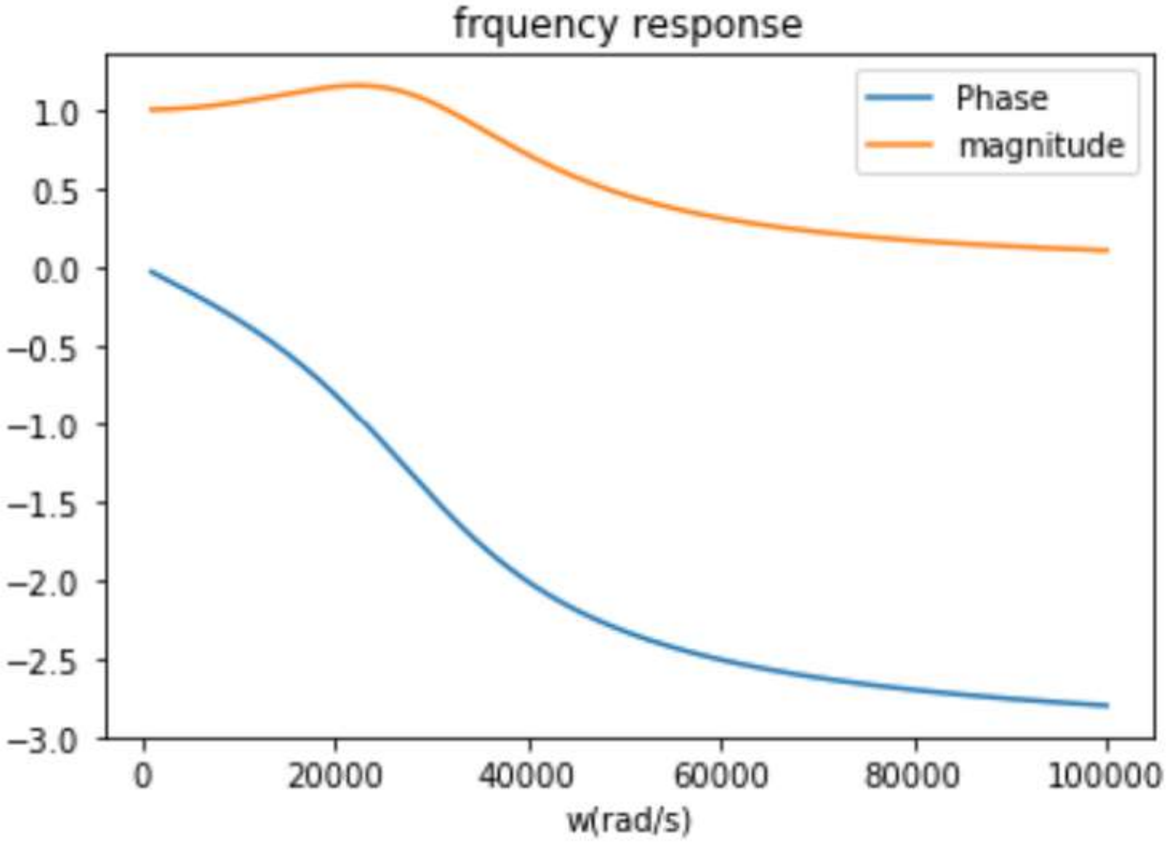
Q=0.4


frquency response
Phase
magnitude
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
w(rad/s)

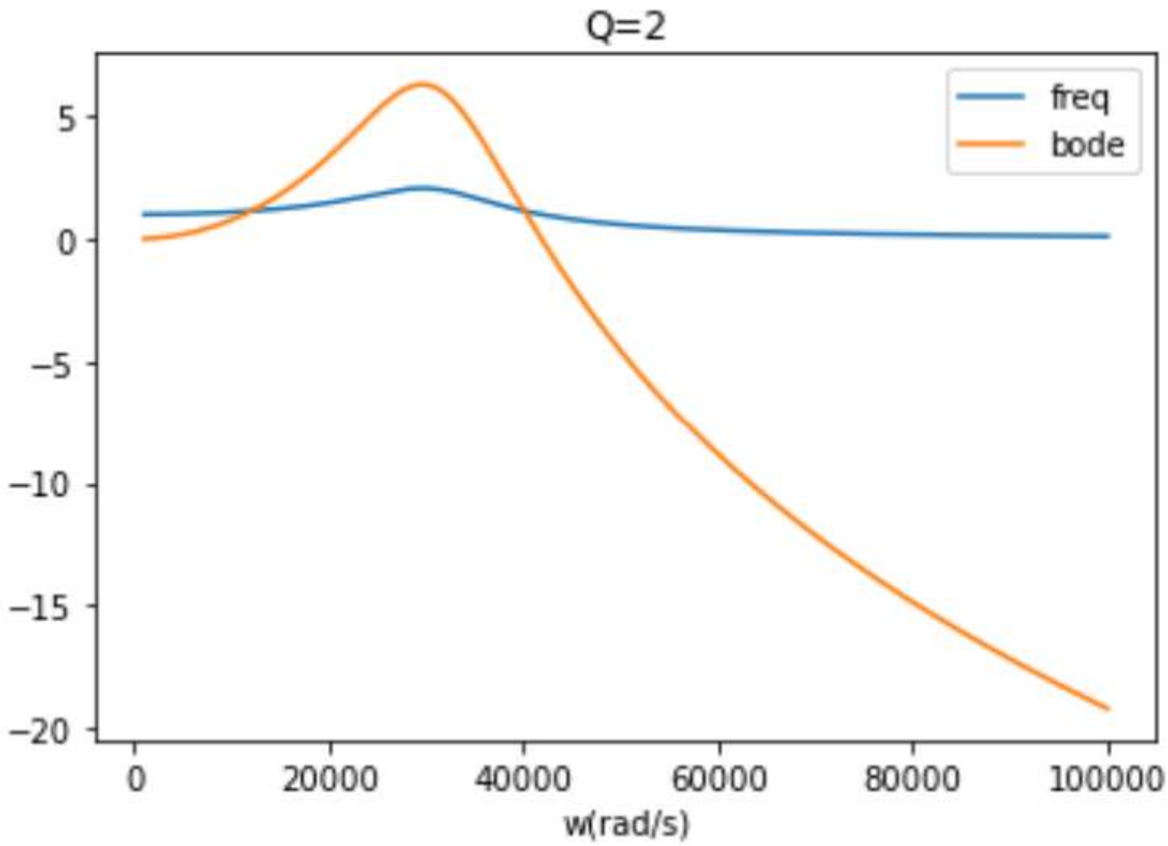


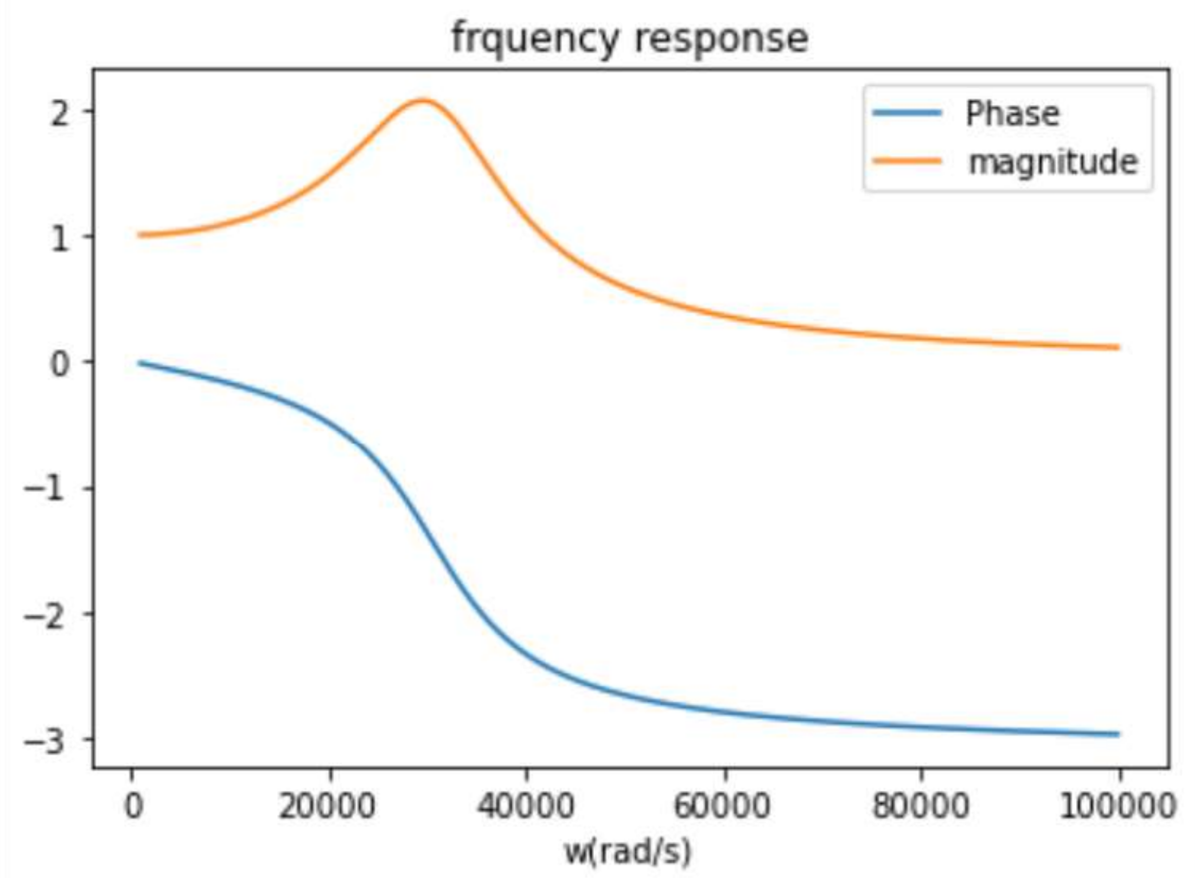
Underdamped Q=1



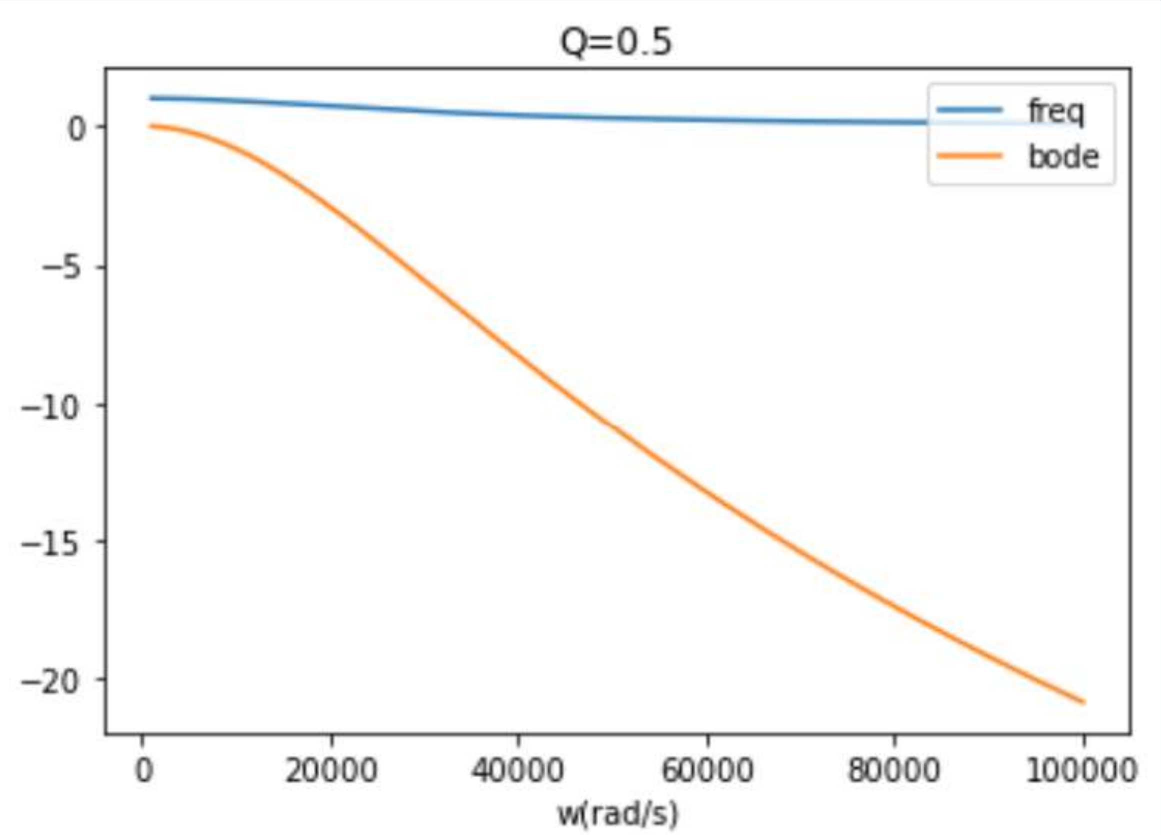

frquency response
Phase
magnitude
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0
20000
40000
60000
80000
100000
w(rad/s)

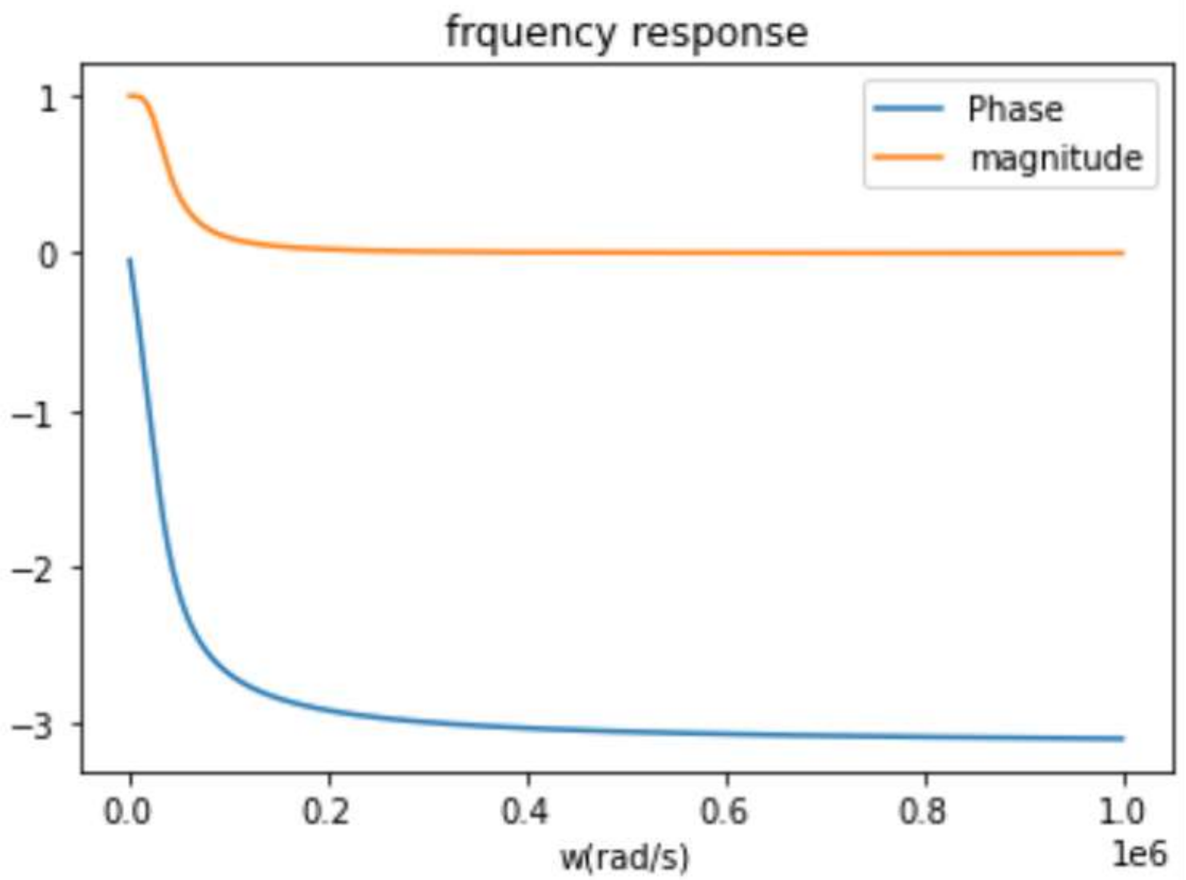
Q = 2


Q=2
freq
bode
5
0
−5
−10
−15
−20
0
20000
40000
60000
80000
100000
w(rad/s)



Critically Damped





2.

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))

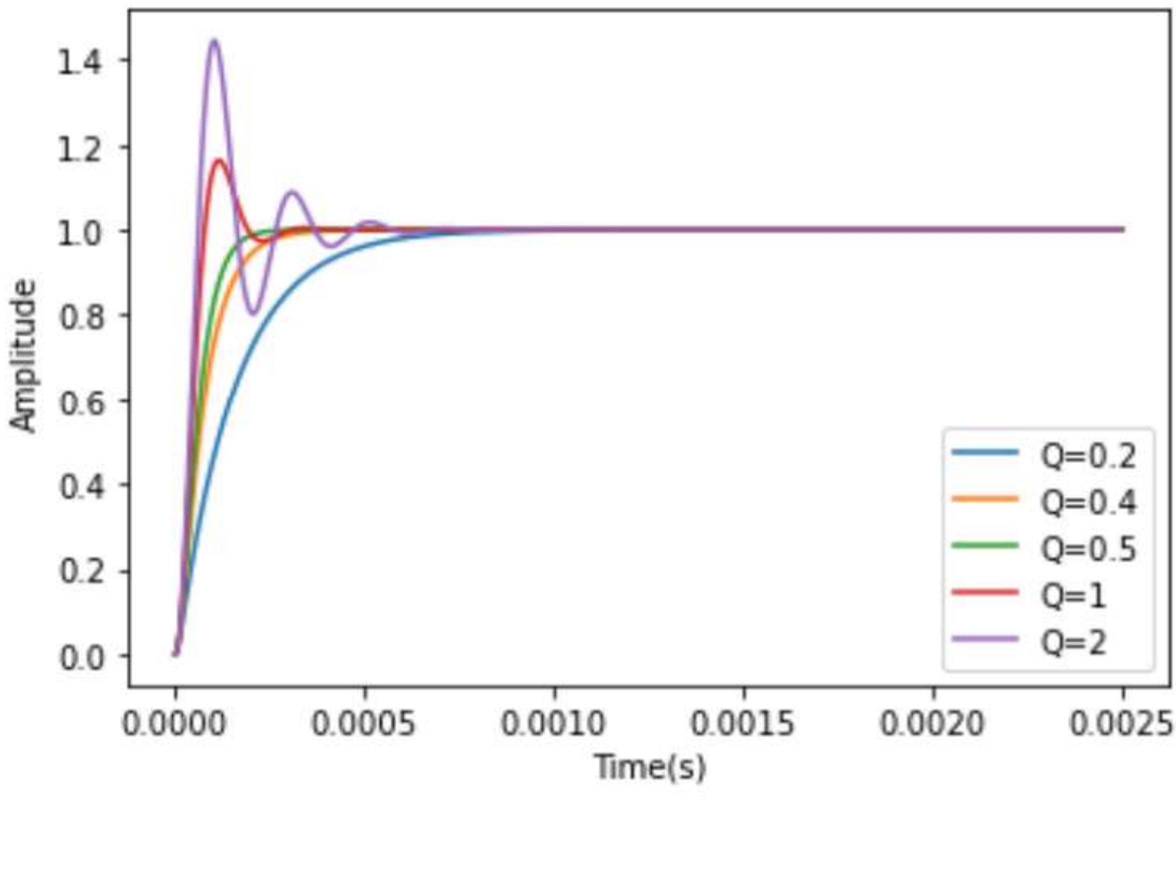
t,s6 = signal.step((numerator,denominator_6),T=np.linspace(0,0.0025,1000))

#ploting the transient response

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

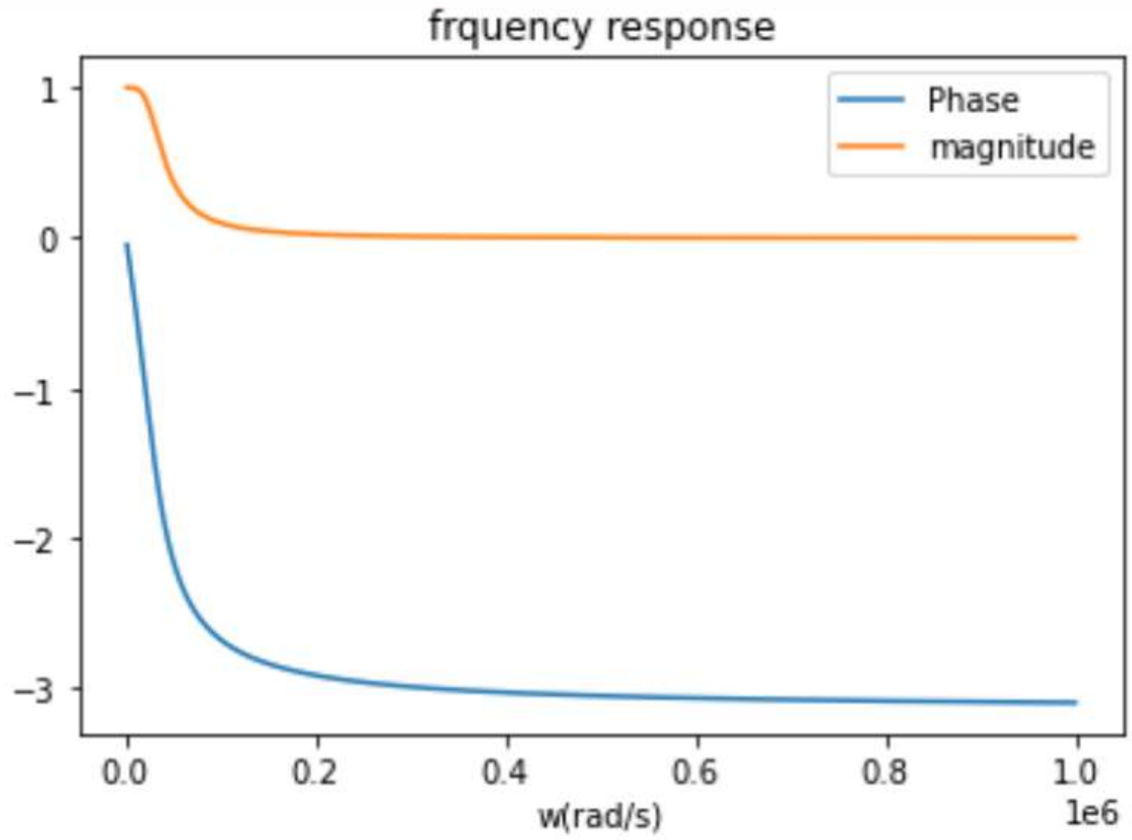
plt.xlabel("Time(s)")

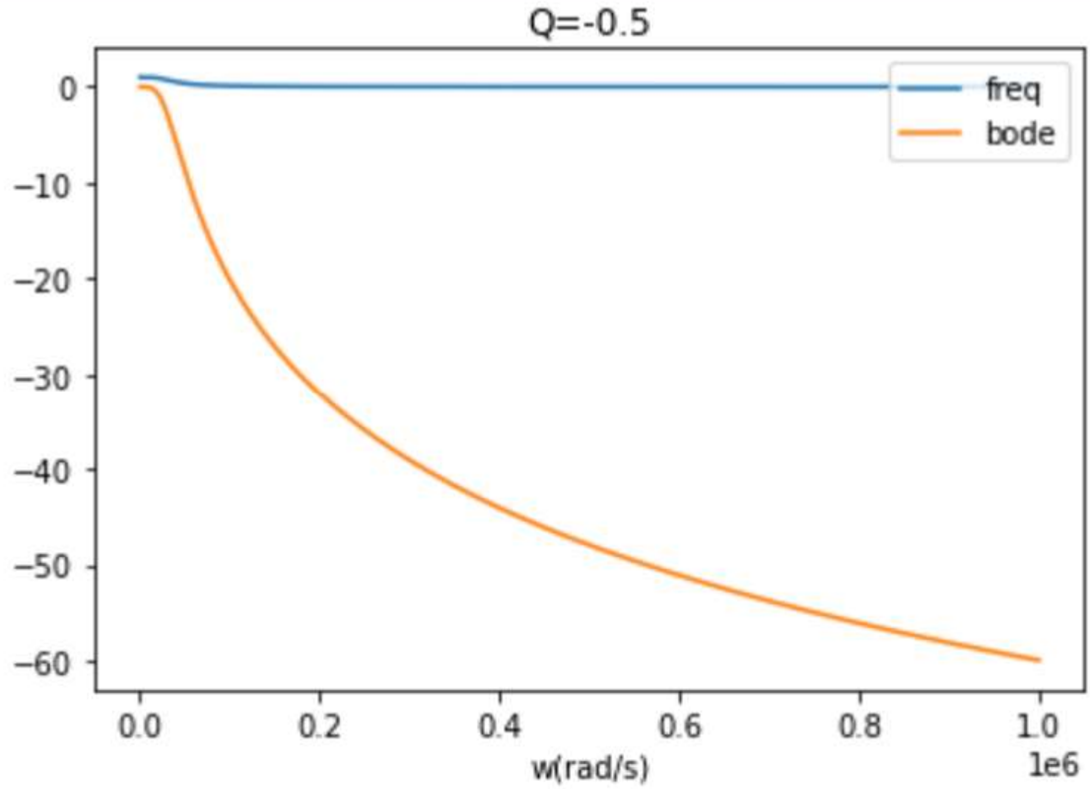
plt.ylabel("Amplitude")

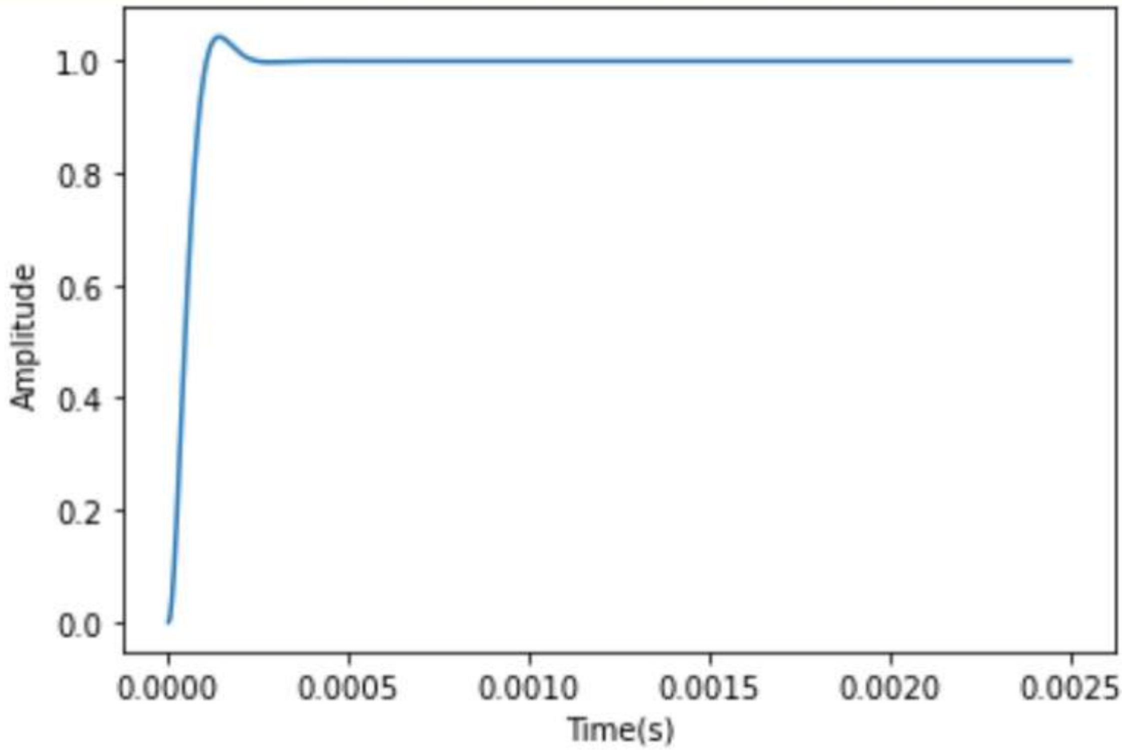


3.

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.