TRANSFORM ANALYSIS

Utkarsh Mahajan 201EC164

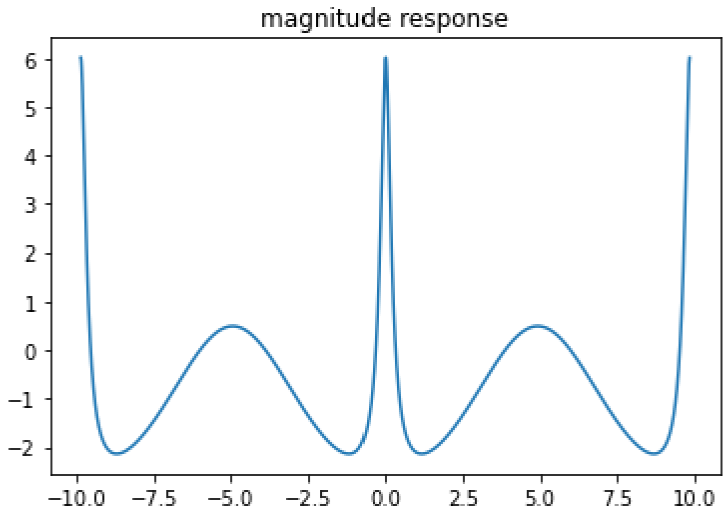
Arnav Raj 201EC109

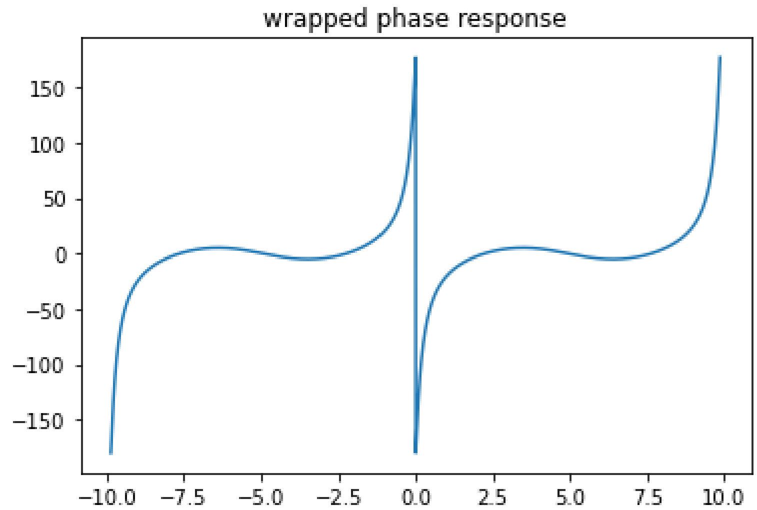
Q1. Plot the magnitude and phase response of the system transfer function $H(z) = \frac{1-0.8z^{-1}}{1-0.9z^{-1}-0.2z^{-2}}$. Plot both the wrapped and unwrapped phase responses.

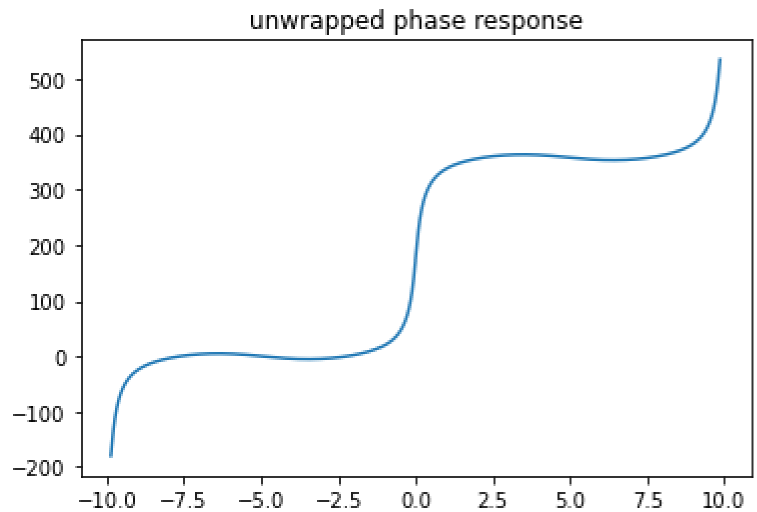
(You can use the built in function numpy.unwrap)

In [33]:

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import scipy.signal as sig

def pltwt(a, b, title):
    plt.title(title)
    plt.plot(a,b)
    plt.show()

b = [1, -0.8]
a = [1, -0.9, -0.2]
c= np.arange(-2*np.pi,2*np.pi,4*np.pi/4096)
w, h = sig.freqz(b, a, c)
hdb = 20*np.log10(abs(h))
angle = np.angle(h,deg=True)
pltwt(w/2*np.pi,hdb, 'magnitude response')
pltwt(w/2*np.pi,angle, 'wrapped phase response')
pltwt(w/2*np.pi,np.unwrap(angle), 'unwrapped phase response')
```







Q2. Plot the response of the system defined by the transfer function

$$H(z) = \frac{\left(1 - 0.98e^{j0.8\pi}z^{-1}\right)\left(1 - 0.98e^{-j0.8\pi}z^{-1}\right)}{\left(1 - 0.8e^{j0.4\pi}z^{-1}\right)\left(1 - 0.8e^{-j0.4\pi}z^{-1}\right)} \prod_{k=1}^{4} \left(\frac{\left(c_k^* - z^{-1}\right)\left(c_k - z^{-1}\right)}{\left(1 - c_k z^{-1}\right)\left(1 - c_k^* z^{-1}\right)} \right)^2,$$

where $c_k = 0.95e^{j(0.15\pi + 0.02\pi k)}$ for $k = 1, 2, 3, 4$.

Input to the system is $x[n] = x_3[n] + x_1[n-61] + x_2[n-122]$, where
$x_1[n] = w[n]\cos(0.2\pi n)$,

$x_2[n] = w[n]\cos(0.4\pi n - \frac{\pi}{2})$,

$x_3[n] = w[n]\cos(0.8\pi n + \frac{\pi}{5})$, and

$$w[n] = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi n}{60}), & 0 \le n \le 60 \\ 0, & otherwise \end{cases}$$

Plot the following:

1. Pole Zero locations of the system
2. Magnitude and unwrapped Phase response of the system

3. Group delay of the system
4. Time domain plot of the input signal
5. Magnitude of DTFT of the input signal
6. Response of the system to the input $x[n]$

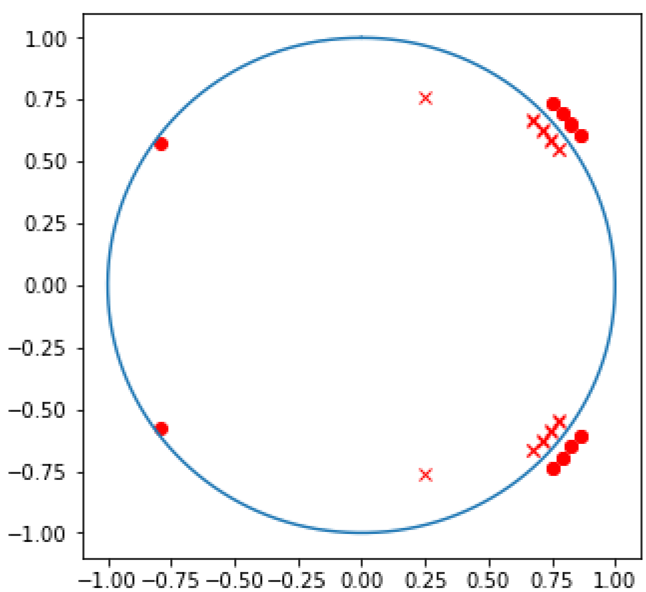
(For finding the product of the polynomials, use the function numpy.polynomial.polynomial.polymul)

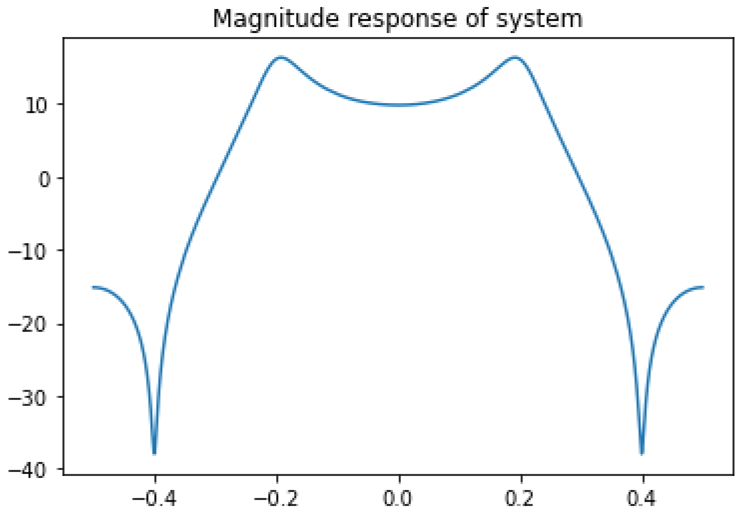
In [32]:
```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import numpy.polynomial.polynomial as pm
import scipy.signal as sig
import itertools
import collections
from sympy import *

def pltwt(a, b, title):
    plt.title(title)
    plt.plot(a,b)
    plt.show()

def plwt(a, b, c, title):
    plt.title(title)
    plt.plot(a,b)
    plt.show()
pi = np.pi
#phase
phase = [0.15*pi + 0.02*pi*k for k in range(1,5)]
ck = [0.95*(complex(np.cos(k),np.sin(k))) for k in phase]
ck5=0.98*complex((np.cos(0.8*np.pi)),(np.sin(0.8*np.pi)))
ck5_num = [1,-ck5]
ck6=0.8*complex((np.cos(0.4*np.pi)),(np.sin(0.4*np.pi)))
ck6_den = [1,-ck6]
c_conj = [np.conj(k) for k in ck ]
#simplifying numerator and denominator
cn1 = [0.95**2,-0.95*(ck[0]+c_conj[0]),1]
cn1 = pm.polymul(cn1,cn1)
cd1 = [1,-0.95*(ck[0]+c_conj[0]),0.95**2]
cd1 = pm.polymul(cd1,cd1)
cn2 = [0.95**2,-0.95*(ck[1]+c_conj[1]),1]
cn2 = pm.polymul(cn2,cn2)
cd2 = [1,-0.95*(ck[1]+c_conj[1]),0.95**2]
cd2 = pm.polymul(cd2,cd2)
cn3 = [0.95**2,-0.95*(ck[2]+c_conj[2]),1]
cn3 = pm.polymul(cn3,cn3)
cd3 = [1,-0.95*(ck[2]+c_conj[2]),0.95**2]
cd3 = pm.polymul(cd3,cd3)
c_num4 = [0.95**2,-0.95*(ck[3]+c_conj[3]),1]
c_num4 = pm.polymul(c_num4,c_num4)
c_den4 = [1,-0.95*(ck[3]+c_conj[3]),0.95**2]
c_den4 = pm.polymul(c_den4,c_den4)
c_simp = pm.polymul(cn1,cn2)
c_simp1 = pm.polymul(c_simp,cn3)
c_simp2 = pm.polymul(c_simp1,c_num4)
c_simp_d = pm.polymul(cd1,cd2)
c_simp1_d = pm.polymul(c_simp_d,cd3)
```

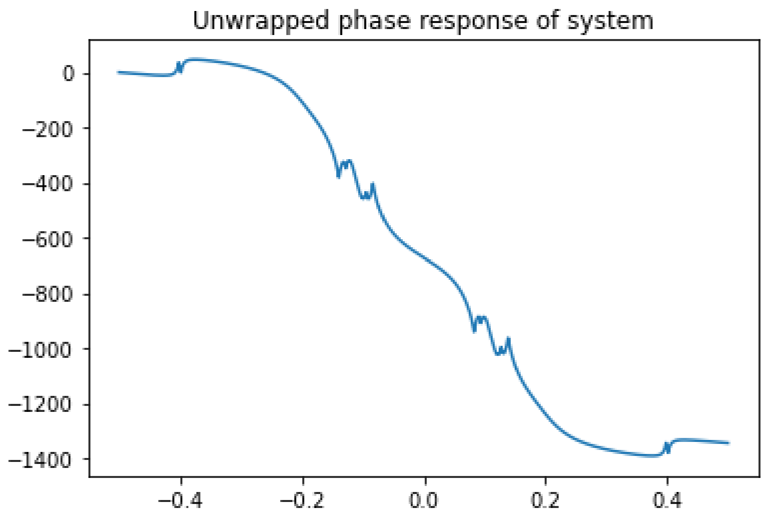
```
c_simp2_d = pm.polymul(c_simp1_d,c_den4)
cktotal = pm.polymul(ck5_num,np.conj(ck5_num))
num_final = pm.polymul(cktotal,c_simp2)
cktotald = pm.polymul(ck6_den,np.conj(ck6_den))
den_final = pm.polymul(cktotald,c_simp2_d)
#final matrices
num_final=list(num_final)
den_final=list(den_final)
#final coefficients(numerator and denominator)
b_matrix_for_response=num_final
a_matrix_for_response=den_final
#pole zero location
def polezero(b,a):

  z,p,k =sig.tf2zpk(b,a)
  print(z,p)
  plt.figure(figsize=(5, 5))
  plt.plot(z.real, z.imag, 'ro', p.real, p.imag, 'rx')
  n1 = np.arange(0,np.pi*2,1/100)
  plt.plot(np.sin(n1),np.cos(n1))
  plt.show()
polezero(b_matrix_for_response,a_matrix_for_response)
#magnitude and unwrapped phase response
c=np.arange(-1*np.pi, 1*np.pi, 2*np.pi/4096)
w2,h2=sig.freqz(b_matrix_for_response,a_matrix_for_response,c)
h2db=20*np.log10(abs(h2))
pltwt(w2/(2*np.pi),h2db, "Magnitude response of system")
angles2 = np.angle(h2,deg=True)
k=np.unwrap(angles2)

plwt(w2/(2*np.pi), k, 'g', "Unwrapped phase response of system")
#grp delay
w_grp_delay1,gd1=sig.group_delay((b_matrix_for_response,a_matrix_for_response)

pltwt(w_grp_delay1, gd1, "Group delay of system")
#time domain plot of input
n = np.arange(0,61)  # Range of n
w_n=0.54-0.46*(np.cos((2*(np.pi)*n)/60))
x1_n=w_n*np.cos(0.2*(np.pi)*n)
x2_n=w_n*np.cos((0.4*(np.pi)*n)-((np.pi)/2))
x3_n=w_n*np.cos((0.8*(np.pi)*n)+((np.pi)/5))
n=list(n)
n1=[x+61 for x in n]
n2=[x+122 for x in n]
dict_1={}
dict_2={}
dict_3={}
for i in range(len(n1)):
  dict_1[n1[i]]=x1_n[i]
for i in range(len(n2)):
  dict_2[n2[i]]=x2_n[i]
for i in range(len(n)):
  dict_3[n[i]]=x3_n[i]
Cdict = collections.defaultdict(int)
for key, val in itertools.chain(dict_1.items(), dict_2.items(),dict_3.items())
    Cdict[key] += val
n4=Cdict.keys()
x_n=Cdict.values()
x_n=list(x_n)
plt.title("Time domain of input")
```

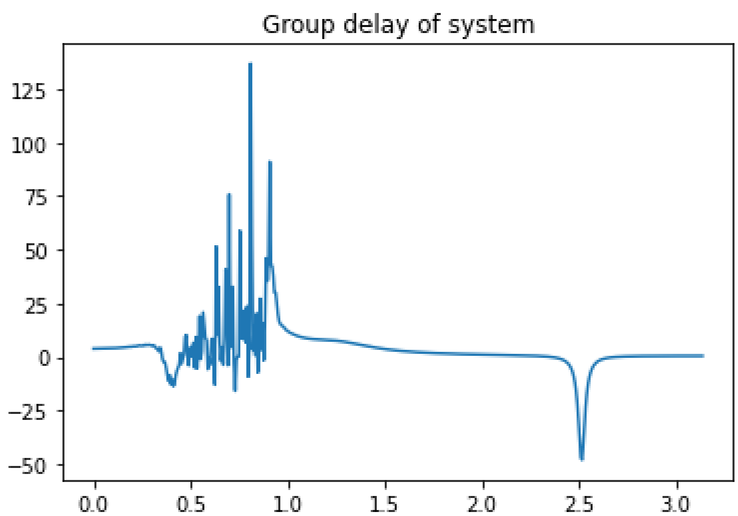
```
plt.stem(n4,x_n,use_line_collection=True)
plt.figure(figsize = (20,6))
#magnitude of DTFT of input signal
b_dtft=[]
for i in range(len(x_n)):
  b_dtft.append(x_n[i])
a_dtft=[1]
c=np.arange(-1*np.pi, 1*np.pi, 2*np.pi/4096)
w3,h3=sig.freqz(b_dtft,a_dtft,c)
h3db=20*np.log10(abs(h3))
pltwt(w3/(2*np.pi),h3db, "Magnitude of dtft of input signal")
#phase of the dtft of the input signal
angles3 = np.angle(h3,deg=True)
plwt(w3/(2*np.pi), angles3, 'g', "Phase of dtft of input signal")
#response of the system to the input x_n
b_matrix_for_lti=num_final
a_matrix_for_lti=den_final
n_for_stem=np.arange(0,180)
response=sig.lfilter(b_matrix_for_lti,a_matrix_for_lti,x_n)
plt.stem(n_for_stem,response[:180],use_line_collection=True)
plt.title("Response of the system to the input")
plt.show()
```

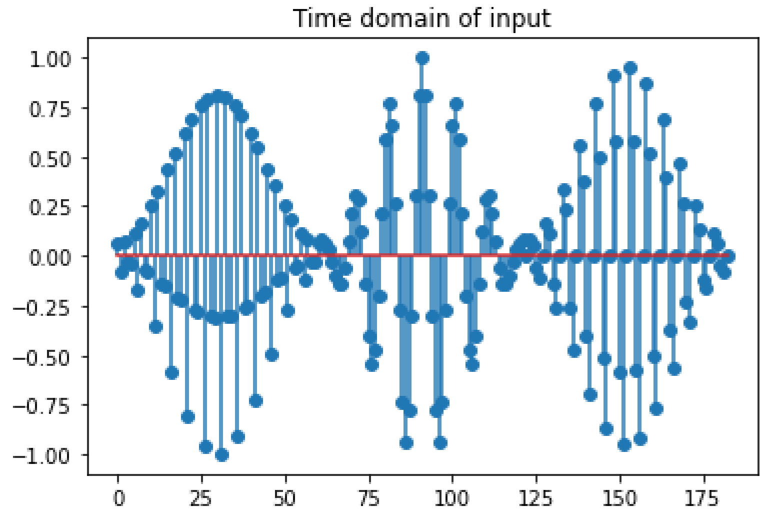
```
[-0.79283665-0.57602955j -0.79283665+0.57602955j  0.75011337+0.73881267j
  0.75010368+0.73816142j  0.79021795+0.69655213j  0.79007267+0.69438636j
  0.82722925+0.65233598j  0.82694647+0.64993803j  0.86082078+0.60637988j
  0.86067319+0.60549613j  0.86067249-0.6054978j   0.86082145-0.60637811j
  0.75011441-0.73881144j  0.75010265-0.73816271j  0.82723175-0.65233121j
  0.82694394-0.64994262j  0.79022084-0.696548j    0.79006985-0.6943907j ]
[0.2472136 +0.76084521j 0.67750736+0.66700111j 0.67641804+0.66600325j
 0.71111069+0.62618901j 0.71504098+0.6292781j  0.74407683+0.58598596j
 0.74886896+0.58921849j 0.77782047+0.54738906j 0.77585673+0.54624652j
 0.2472136 -0.76084521j 0.67749787-0.66700354j 0.67642723-0.66599981j
 0.71114232-0.6261835j  0.71500923-0.62927985j 0.74883421-0.58921764j
 0.7441122 -0.58598959j 0.77780764-0.54738715j 0.77586937-0.5462504j ]
```

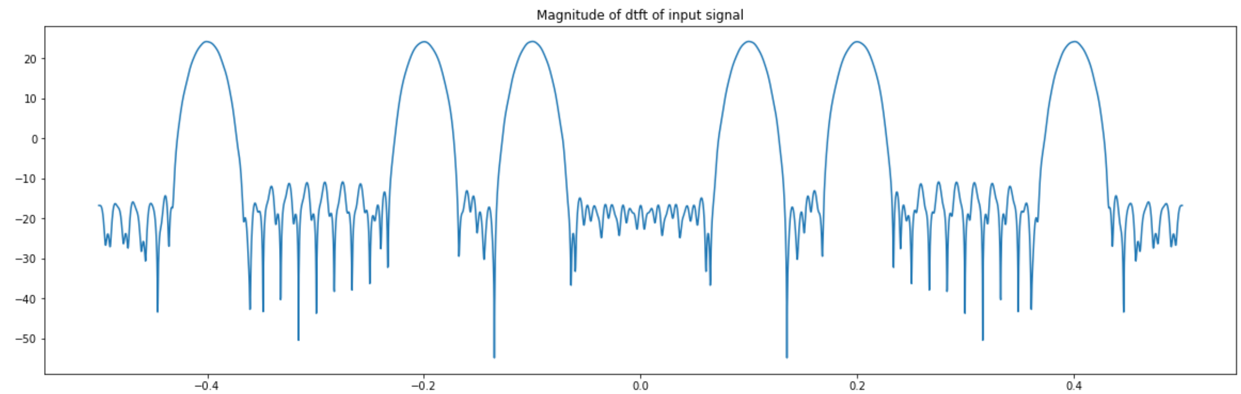


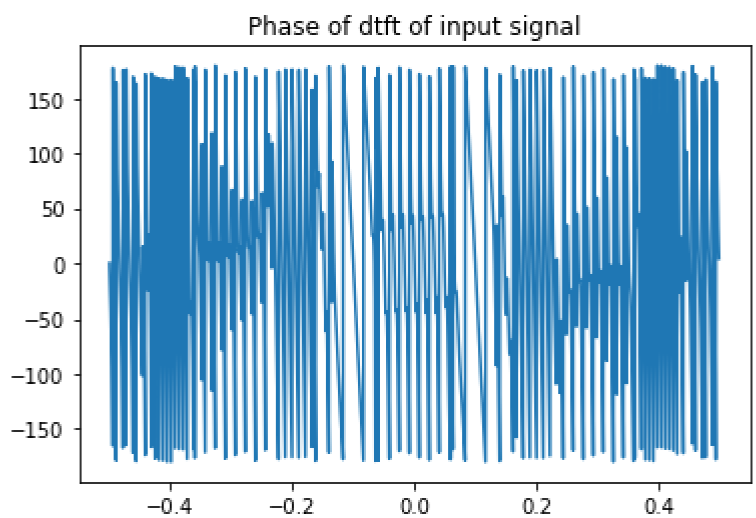

Magnitude response of system
10
0
−10
−20
−30
−40
−0.4
−0.2
0.0
0.2
0.4


Unwrapped phase response of system
0
−200
−400
−600
−800
−1000
−1200
−1400
−0.4
−0.2
0.0
0.2
0.4

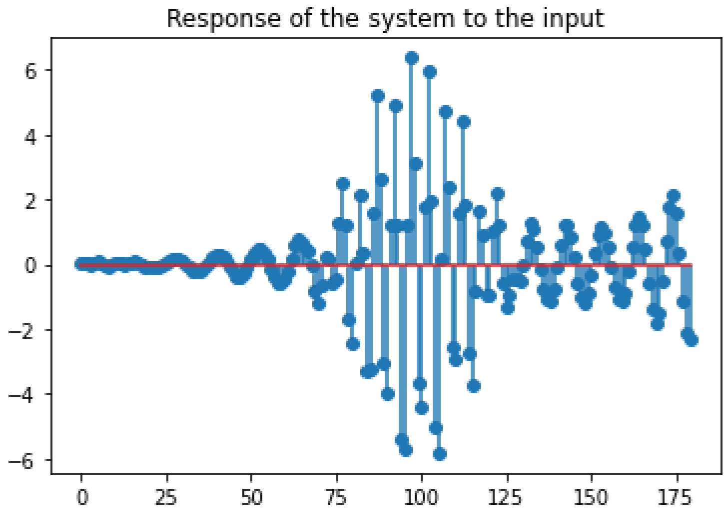

Group delay of system
125
100
75
50
25
0
−25
−50
0.0
0.5
1.0
1.5
2.0
2.5
3.0





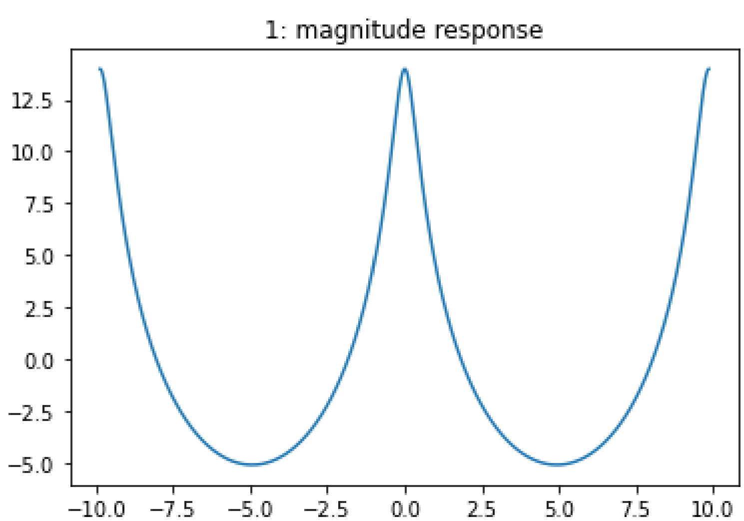


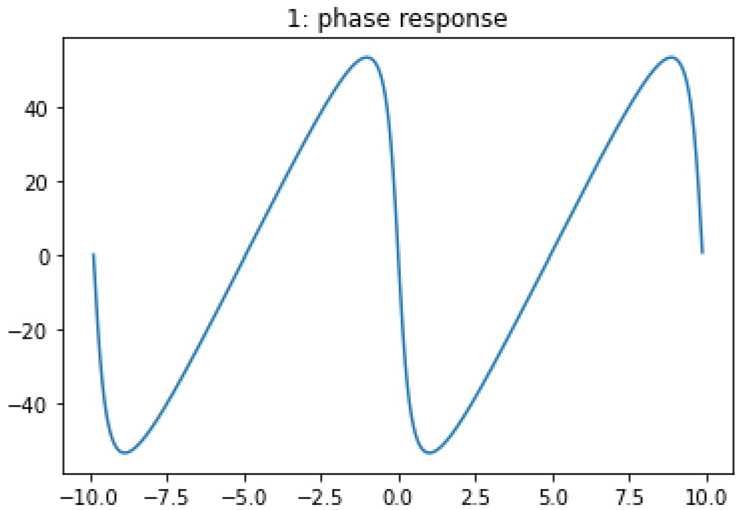

```
/usr/lib/python3.10/site-packages/numpy/ma/core.py:3377: ComplexWarning: Cas
ting complex values to real discards the imaginary part
  _data[indx] = dval
/usr/lib/python3.10/site-packages/matplotlib/cbook/__init__.py:1298: Complex
Warning: Casting complex values to real discards the imaginary part
  return np.asarray(x, float)
/usr/lib/python3.10/site-packages/matplotlib/cbook/__init__.py:1298: Complex
Warning: Casting complex values to real discards the imaginary part
  return np.asarray(x, float)
```

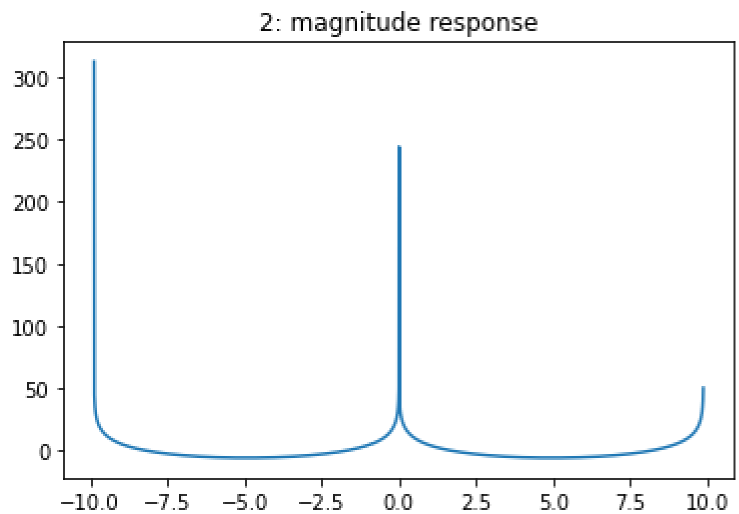


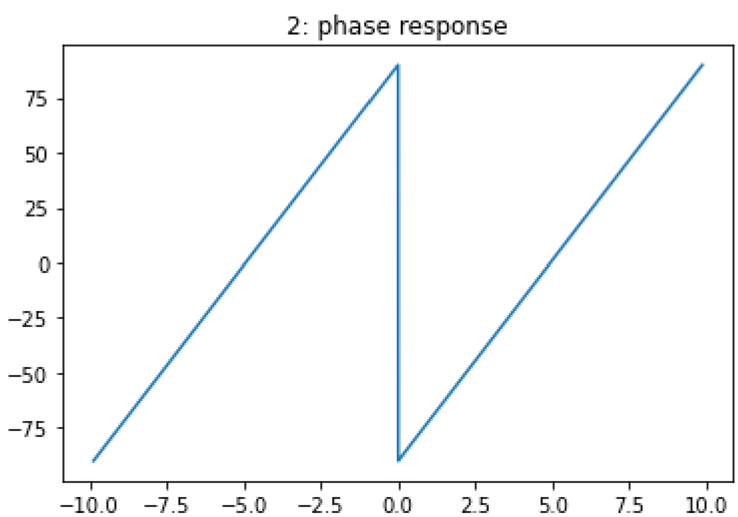
Q4. Plot the Magnitude and phase responses for the system $H(z) = \frac{1}{1-cz^{-1}}$ for $c = 0.8, 1, 2, 0.8e^{j\frac{\pi}{3}}, e^{j\frac{\pi}{3}}, 2e^{j\frac{\pi}{3}}$

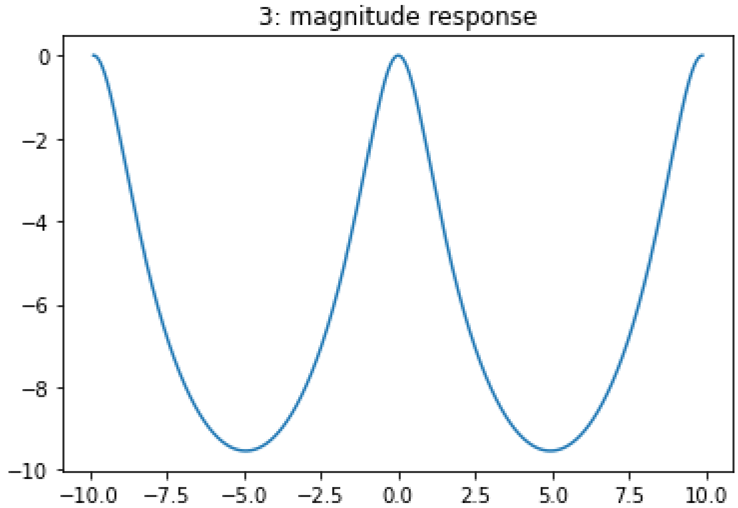
In [20]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
def pltwt(a, b, title):
    plt.title(title)
    plt.plot(a,b)
    plt.show()

exp = complex(np.cos(np.pi/3), np.sin(np.pi/3))
array = [0.8, 1, 2, 0.8*exp, exp, 2*exp]
b = [1,0]
c = np.arange(-2* np.pi, 2*np.pi, 4*np.pi/4096)
for i, n in enumerate(array):
    a=[1,-n]
    w,h=sig.freqz(b,a,c)
    hdb=20*np.log10(abs(h))
    angle=np.angle(h,deg=True)
    pltwt(w/2*np.pi,hdb,str(i+1) +': magnitude response')
    pltwt(w/2*np.pi,angle,str(i+1) +': phase response')
```



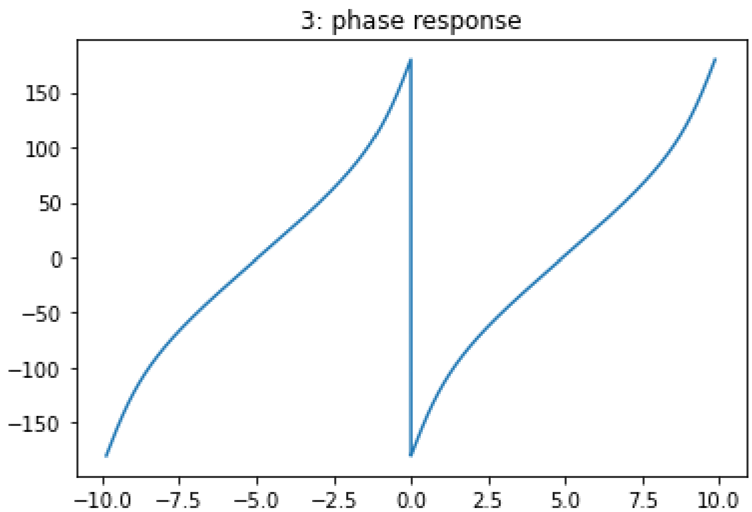

1: phase response
40
20
0
−20
−40
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


2: magnitude response
300
250
200
150
100
50
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

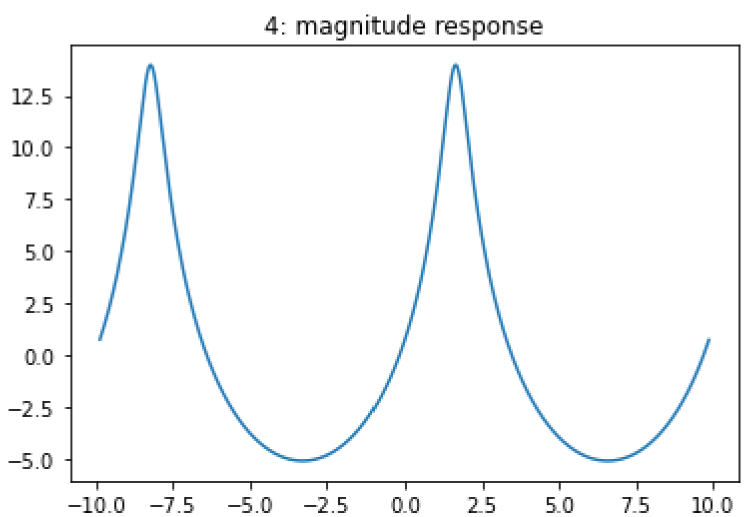

2: phase response
75
50
25
0
−25
−50
−75
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0



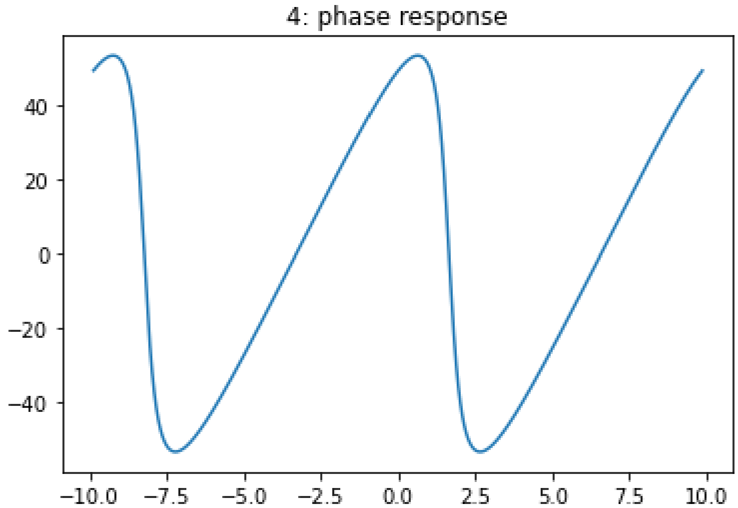
3: magnitude response
0
−2
−4
−6
−8
−10
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

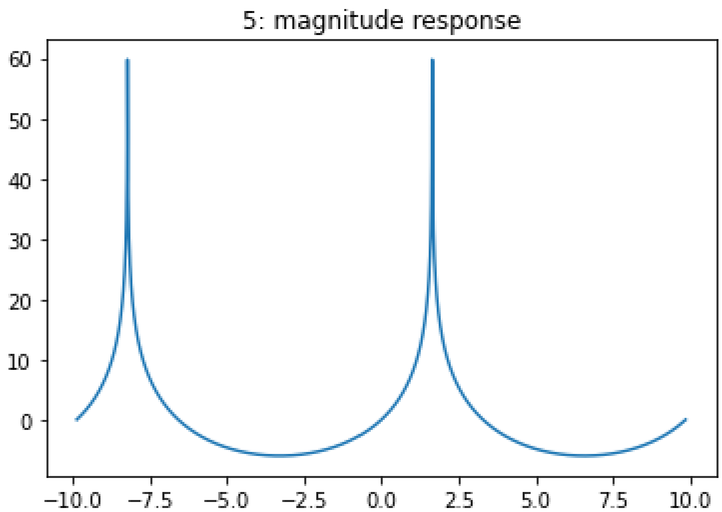


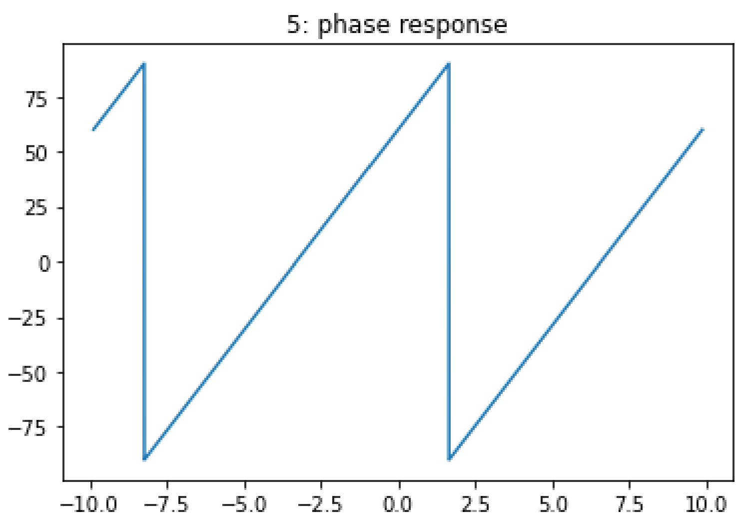
3: phase response
150
100
50
0
−50
−100
−150
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

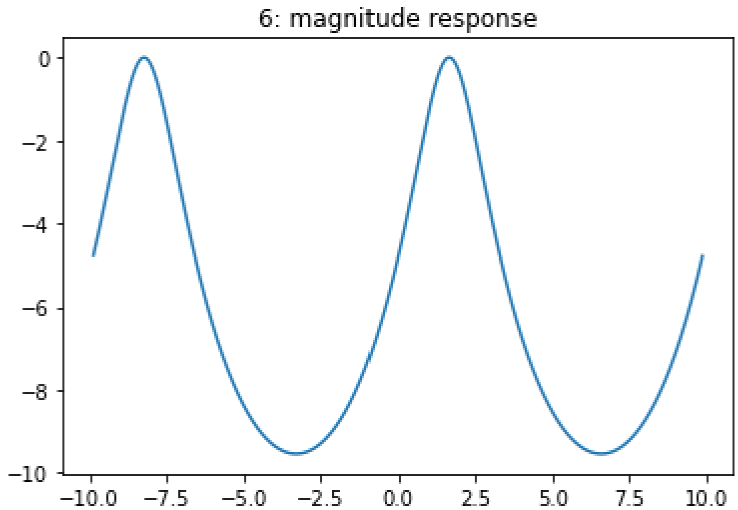


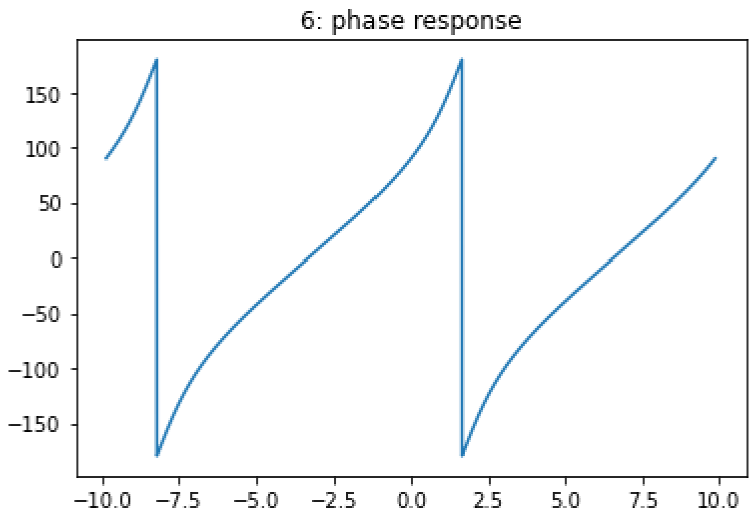
4: magnitude response
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


4: phase response
40
20
0
−20
−40
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0


5: magnitude response
60
50
40
30
20
10
0
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0


5: phase response
75
50
25
0
−25
−50
−75
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0





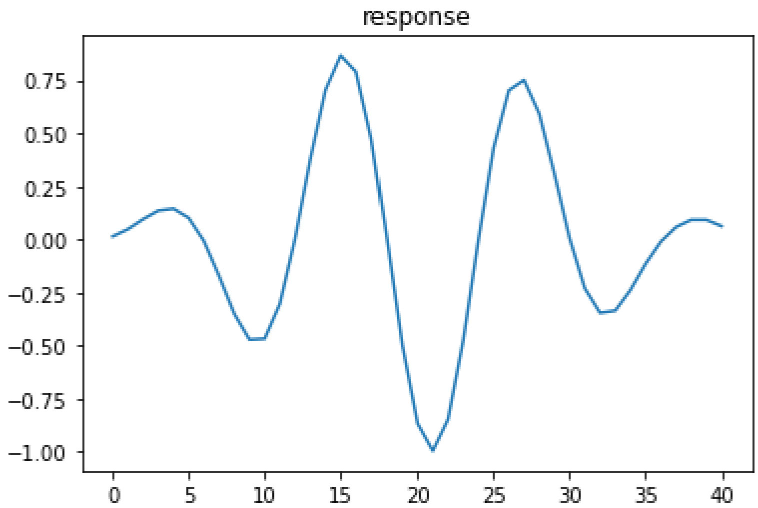
Q5. Impulse response of a system is given by $h[n] = \frac{sin\left(\frac{\pi}{2}(n-20)\right)}{\pi(n-20)}$. Plot the response of this system to the input $x[n] = w[n]\sin(\frac{\pi}{6}n)$, where

$$w[n] = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi n}{40}), & 0 \leq n \leq 40 \\ 0, & otherwise \end{cases}.$$

Compare the theoretical group delay and the group delay observed from the plot.

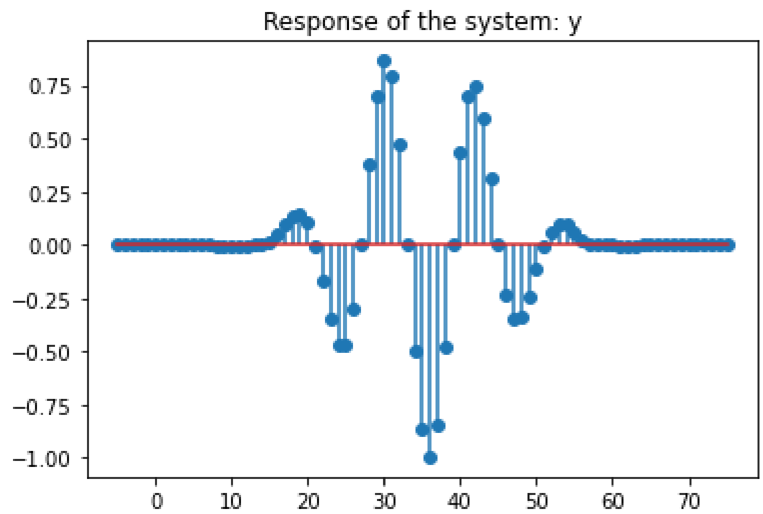
In [23]:

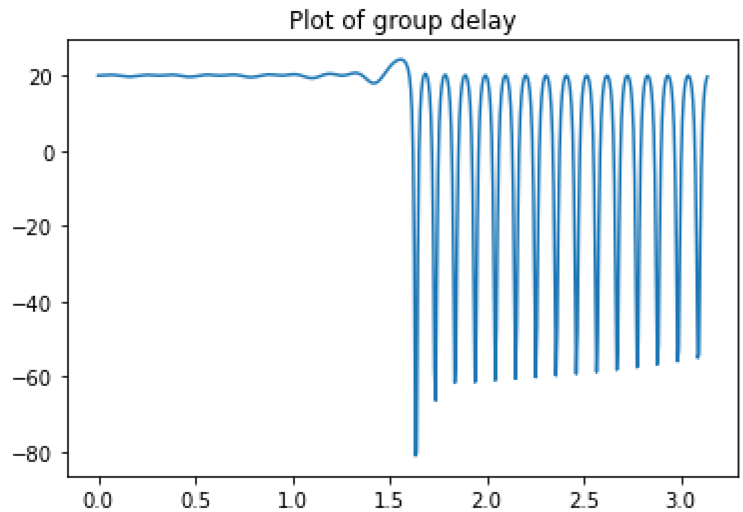
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
def pltwt(a, b, title):
    plt.title(title)
    plt.plot(a,b)
    plt.show()

n = np.arange(0,41)
w = 0.54-0.46*np.cos(2*n*np.pi/40)
x = w*np.sin(n*np.pi/6)
h = 0.5*np.sinc((n-20)/2)
y = sig.convolve(x,h,mode='same')
pltwt(n, y,'response')
```



In [30]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
from scipy import signal
from numpy import pi, diff, unwrap, angle

def pltwt(a, b, title):
    plt.title(title)
    plt.plot(a,b)
    plt.show()

n_q5=np.arange(0,41)
n_q5a=np.arange(-5,100)
h_n_q5=[]
w_n_q5=0.54-0.46*(np.cos((2*(np.pi)*n_q5)/40))
x_n_q5=(w_n_q5)*(np.sin((np.pi)*n_q5/6))
for i in range(len(n_q5)):
  h_n_q5.append((np.sinc((i-20)/2))/2)
y_n_q5=np.convolve(x_n_q5,h_n_q5)
plt.stem(n_q5a[:81],y_n_q5[:81],use_line_collection=True)
plt.title("Response of the system: y")
plt.show()
def h(n):
    return (np.sinc((n-20)/2.0)/2.0)
h = np.array([h(i) for i in n])

b_matrix_for_z=h
a_matrix_for_z=[1]
freq , groupdelay = sig.group_delay((b_matrix_for_z,a_matrix_for_z))
pltwt(freq,groupdelay, "Plot of group delay")
```





Since the slope of unwrapped phase response is constant for specific intervals , we see group delay being constant in the beginning.