Discrete Time Fourier Transform (DTFT)

Submitted by:

Utkarsh Mahajan 201EC164

Arnav Raj 201EC109

Q1. Compute the DTFT (magnitude and phase) of the following (use *scipy.signal.freqz*). Plot from $\omega = -2\pi$ to 2π.

https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.signal.freqz.html

1. $r[n] = u[n] - u[n-5]$
2. $r[n-7]$
3. $r[n+4]$
4. $r[-n]$
5. $(-1)^n r[n]$

What is the period of the DTFT ?

2π, since we can see that the frequency response and the phase response values periodically repeat with a period of 2π.

There are two different discontinuities in the phase spectrum. Identify and explain why it is happeneing?

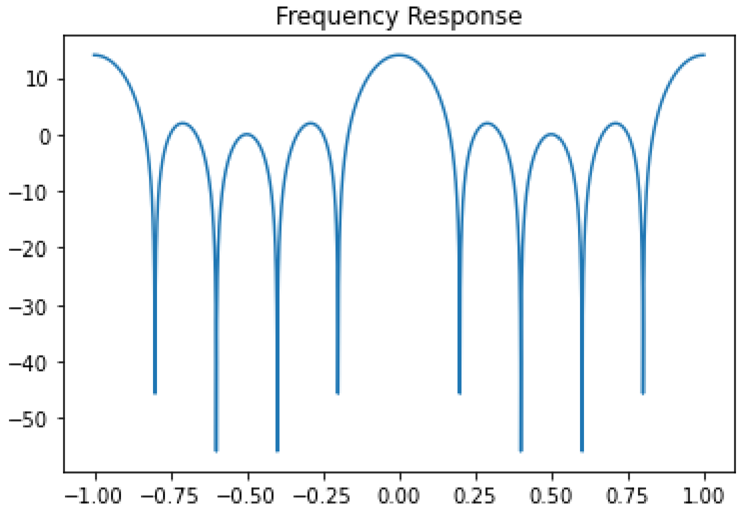
Ans)From the phase frequency plot below, we can see the 2 distinct discontinuities, These are caused by Gibbs Phenomenon.

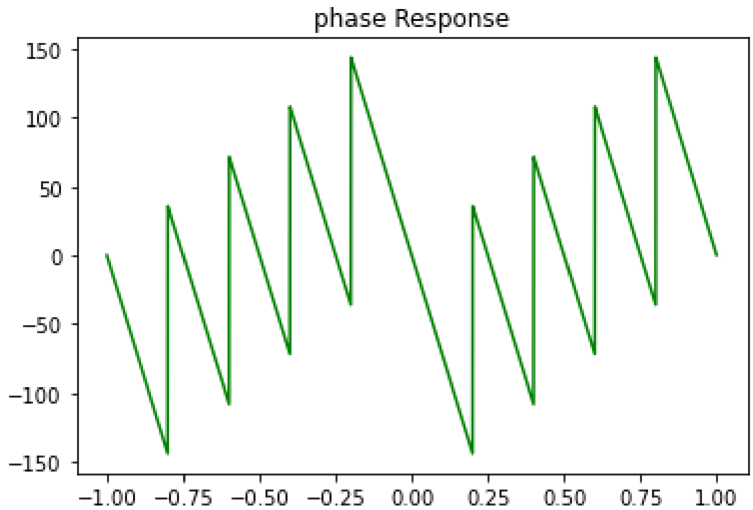
Observe the symmetries and relations between the spectra.

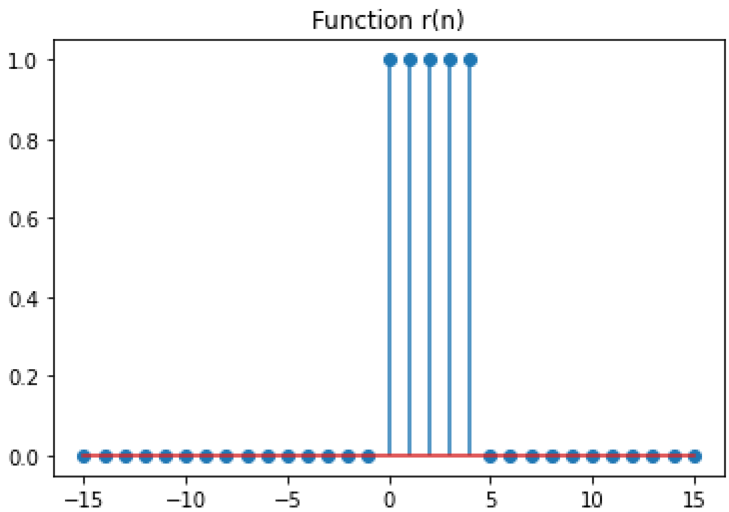
Ans) We can see that the fourier transfer is even which is because the function is real valued.

In [27]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig

#1
b=np.ones(5)
a=[1]
c=np.arange(-2*np.pi, 2*np.pi, 4*np.pi/4096)
w1,h1=sig.freqz(b,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('phase Response')
```

```
plt.plot(w1/(2*np.pi), angles, 'g')
plt.show()

n = np.arange(-15,16)
y = [np.heaviside(n,1)-np.heaviside(n-5,1)]
plt.title('Function r(n)')
plt.stem(n,y[0])
plt.show()

i=1;
while (h1db[0]!=h1db[i]):
    i+=1;
print('Periodicity of dtft is '+ str(i))
```

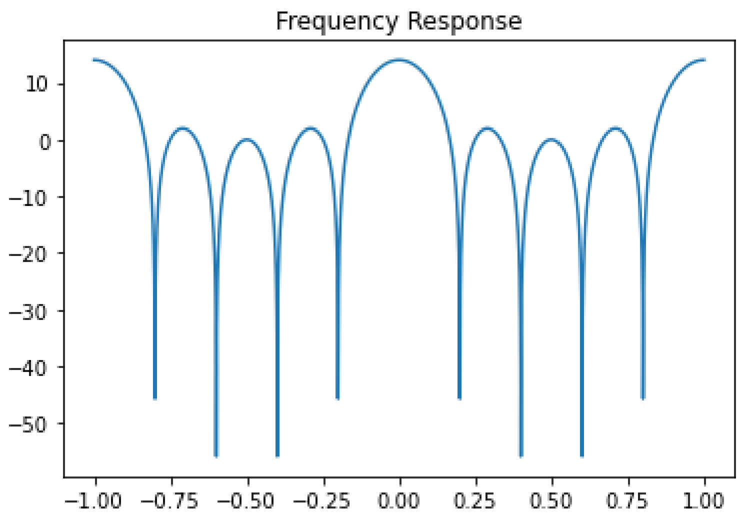


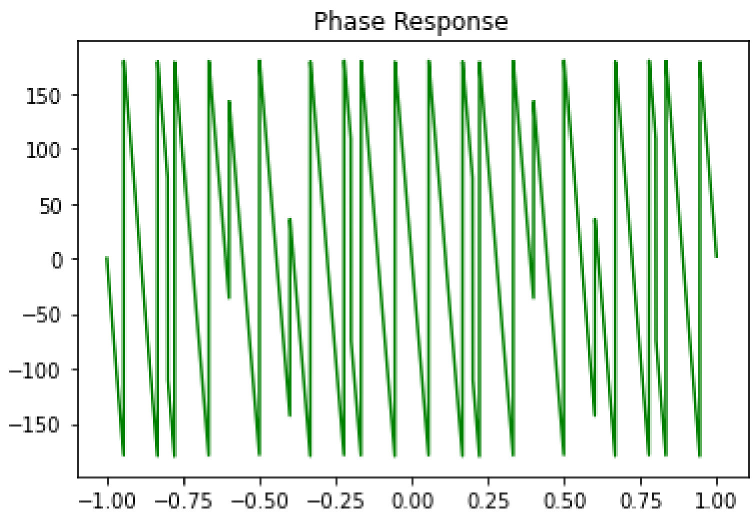


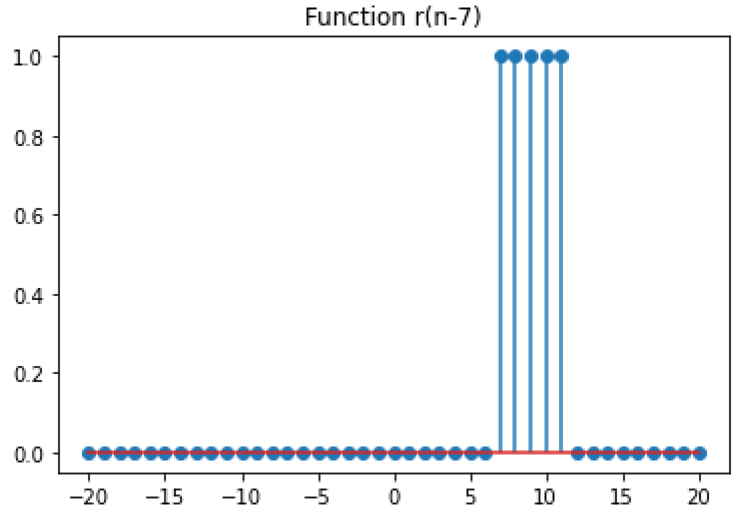



```
Periodicity of dtft is 2048
```

In [28]:
```
#2
b=[*np.zeros(7), *np.ones(5)]
a=[1]
w1,h1=sig.freqz(b,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('Phase Response')
plt.plot(w1/(2*np.pi), angles, 'g')
plt.show()

n = np.arange(-20,21)
y = [np.heaviside(n-7,1)-np.heaviside(n-5-7,1)]
plt.title('Function r(n-7)')
plt.stem(n,y[0])
plt.show()
```

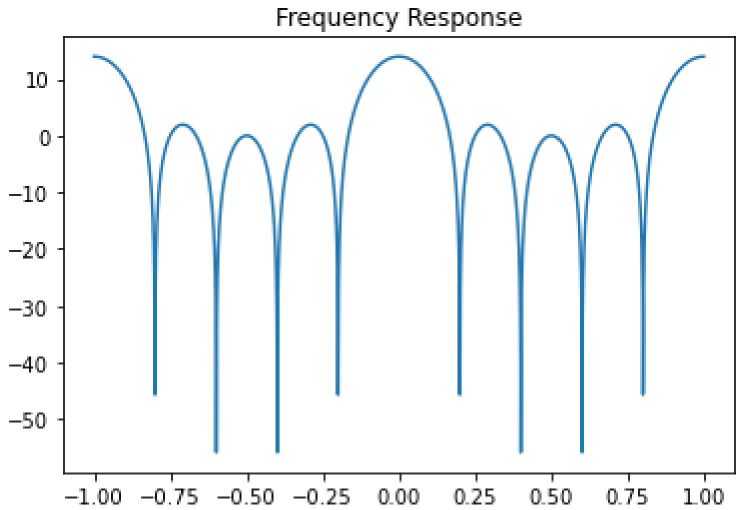


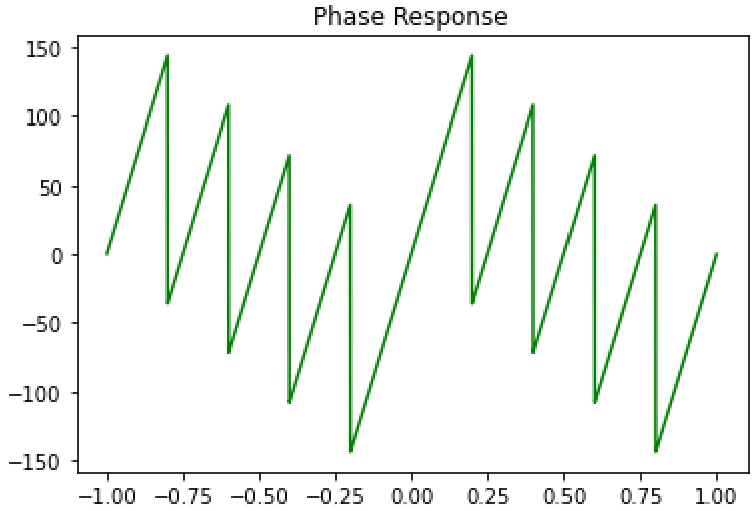


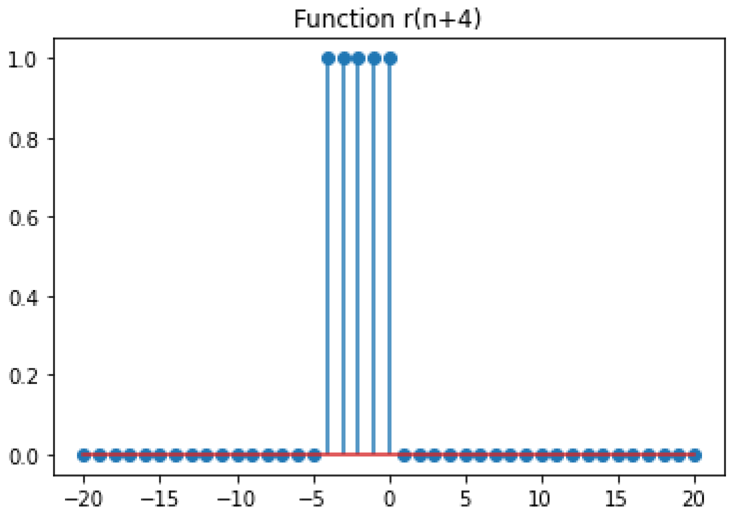


In [29]:

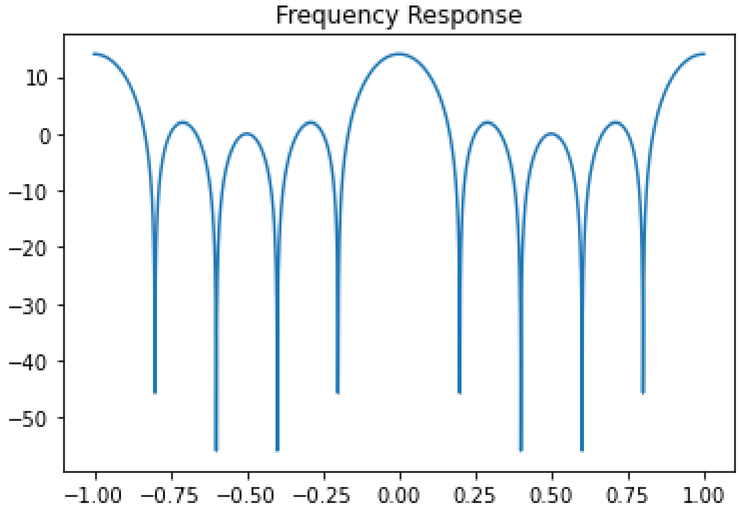
```
#3
#since r(n+4) is equivalent to r(-n). we will just use the time reversal pr
b=[*np.ones(5)]
a=[1]
w1,h1=sig.freqz(b,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('Phase Response')
plt.plot(w1/(2*np.pi), np.flipud(angles), 'g')
plt.show()

n = np.arange(-20,21)
y = [np.heaviside(n+4,1)-np.heaviside(n-5+4,1)]
plt.title('Function r(n+4)')
plt.stem(n,y[0])
plt.show()
```

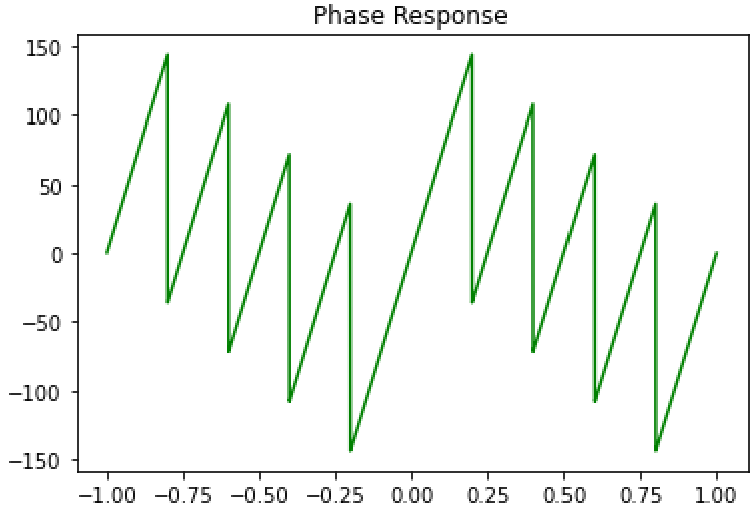


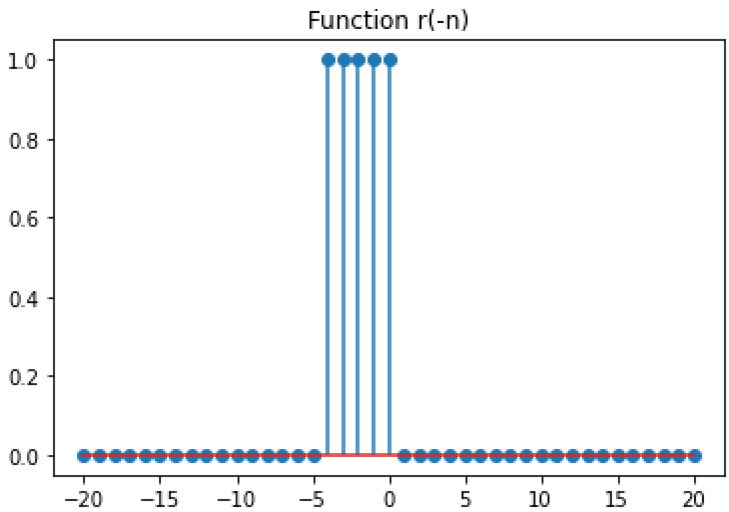




In [30]:
```
#4
b=[*np.ones(5)]
a=[1]
w1,h1=sig.freqz(b,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('Phase Response')
plt.plot(w1/(2*np.pi), np.flipud(angles), 'g')
plt.show()

n = np.arange(-20,21)
y = [np.heaviside(-n,1)-np.heaviside(-n-5,1)]
plt.title('Function r(-n)')
plt.stem(n,y[0])
plt.show()
```



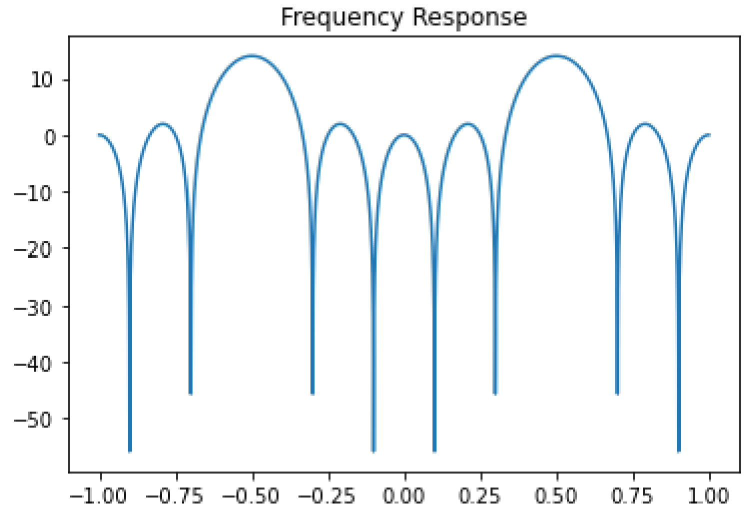


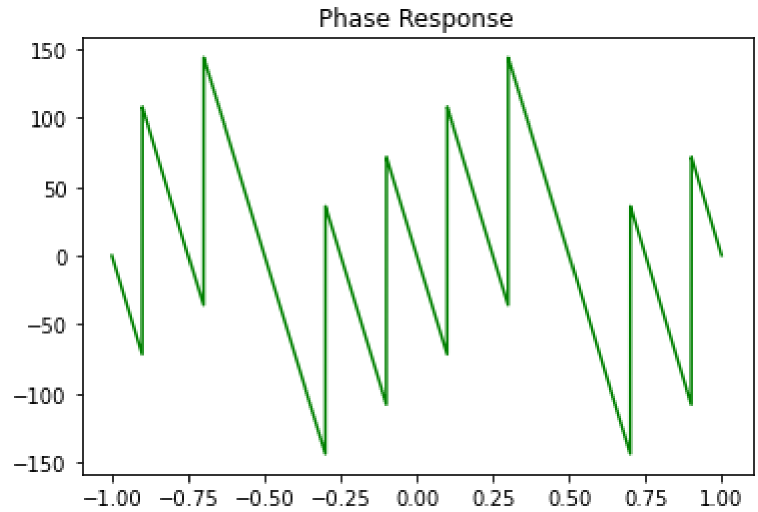


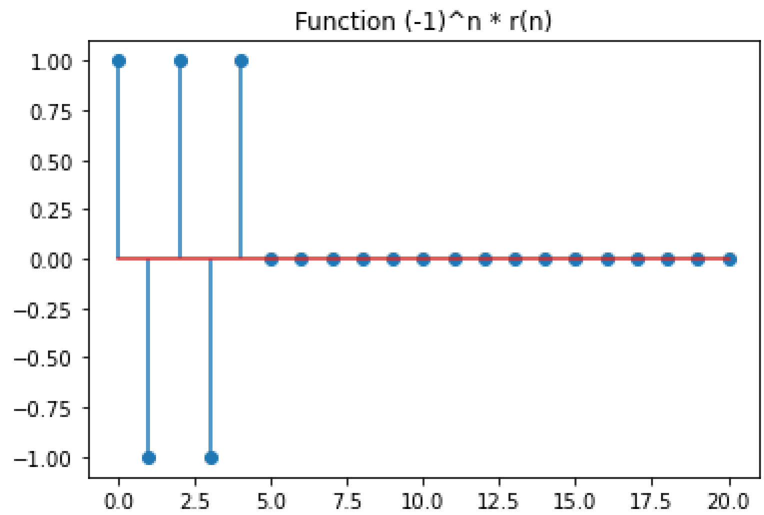
In [31]:

```
#4
b=[1,-1,1,-1,1]
a=[1]
w1,h1=sig.freqz(b,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('Phase Response')
plt.plot(w1/(2*np.pi), angles, 'g')
```

```python
plt.show()

n = np.arange(0,21)
y = [(np.heaviside(n,1)-np.heaviside(n-5,1))*np.power(-1,n)]
plt.title('Function (-1)^n * r(n)')
plt.stem(n,y[0])
plt.show()
```







Q2. Consider the sinusoid $s[n] = (Acos(\omega_0 n + \phi))$ where A = 2; $\omega_0 = \pi/4$; $\phi = \pi/6$. Compute and plot the DTFT from $[-\pi \quad +\pi]$. Use samples from a finite time window

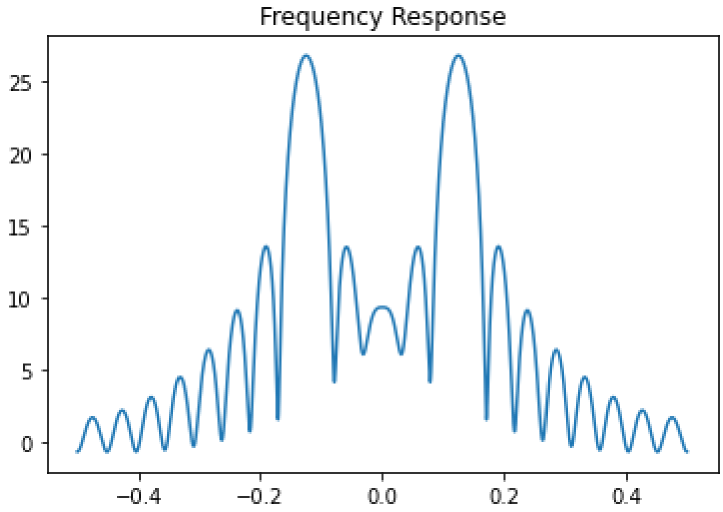
1. $n = [0, 21]$
2. $n = [0, 201]$

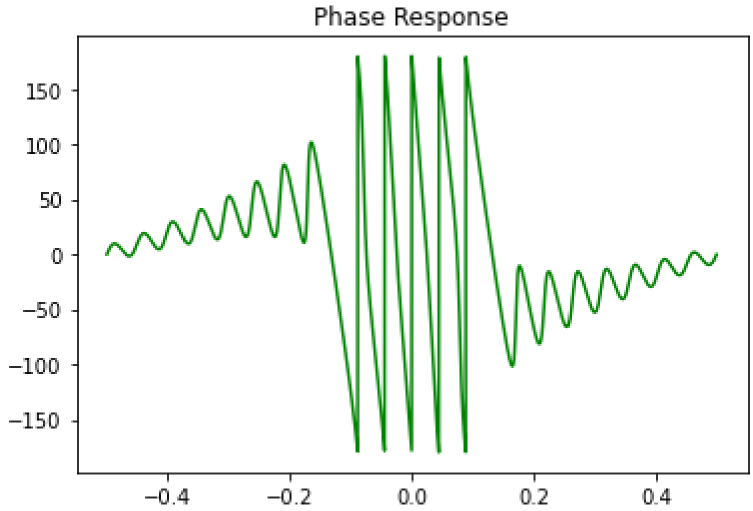
Observe and compare the spectrum in both cases.

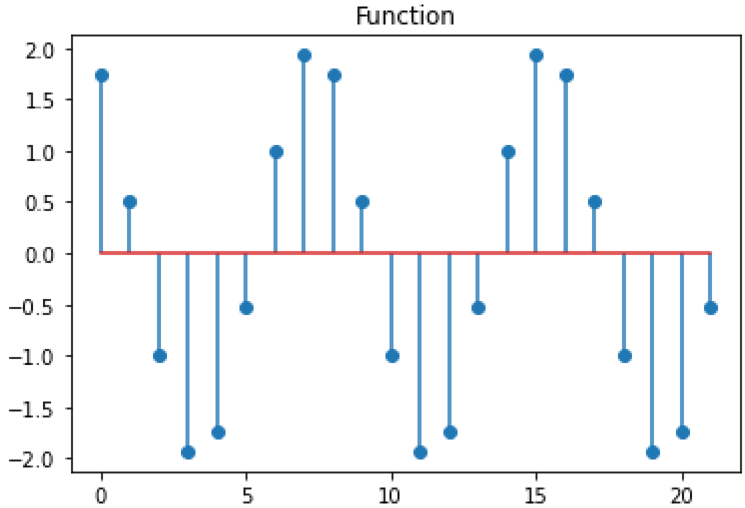
Frequency response with higher range is tending to be more sharper at peaks.

Note: While plotting the spectrum please obtain lot of points (eg. 4096) so that the details are not lost.

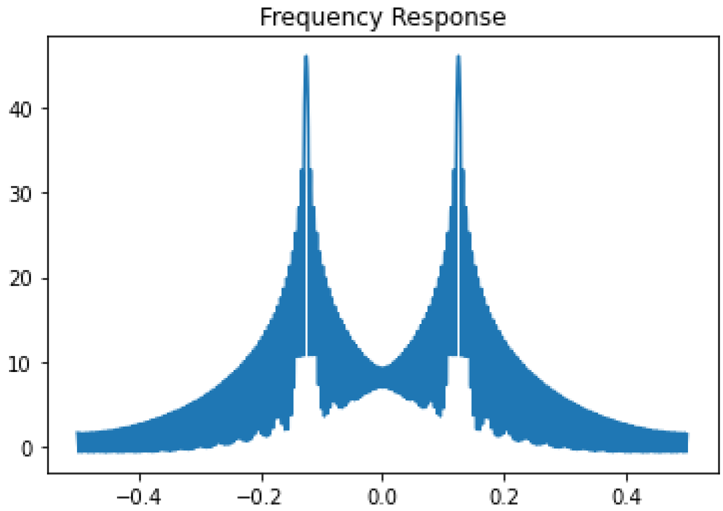
In [32]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig


#for range [0,21]
A=2
w= np.pi/4
th = np.pi/6
c=np.arange(-1*np.pi, np.pi, 2*np.pi/4096)
n = np.arange(0,22);
s = A * np.cos(w*n + th)


a=[1]
w1,h1=sig.freqz(s,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('Phase Response')
plt.plot(w1/(2*np.pi), angles, 'g')
plt.show()


plt.title('Function')
plt.stem(n,s)
plt.show()
```

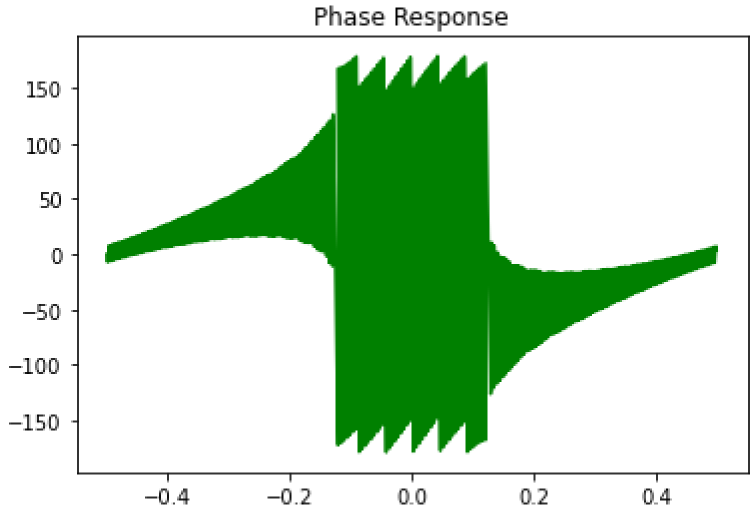


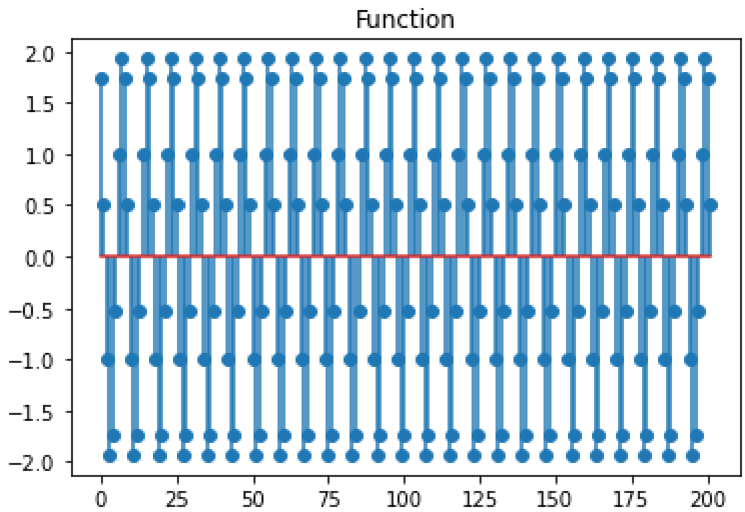




In [33]:
```
#for range [0,201]
A=2
w= np.pi/4
th = np.pi/6
c=np.arange(-1*np.pi, np.pi, 2*np.pi/2048)
n = np.arange(0,202);
s = A * np.cos(w*n + th)


a=[1]
w1,h1=sig.freqz(s,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('Phase Response')
plt.plot(w1/(2*np.pi), angles, 'g')
plt.show()


plt.title('Function')
plt.stem(n,s)
plt.show()
```







Q3. Consider a signal $x[n] \; = \; u[n] - u[n-6]$. Plot the DTFT of the signal $X(e^{j\omega})$ in $[-\pi \quad +\pi]$. Consider an expanded version of the signal

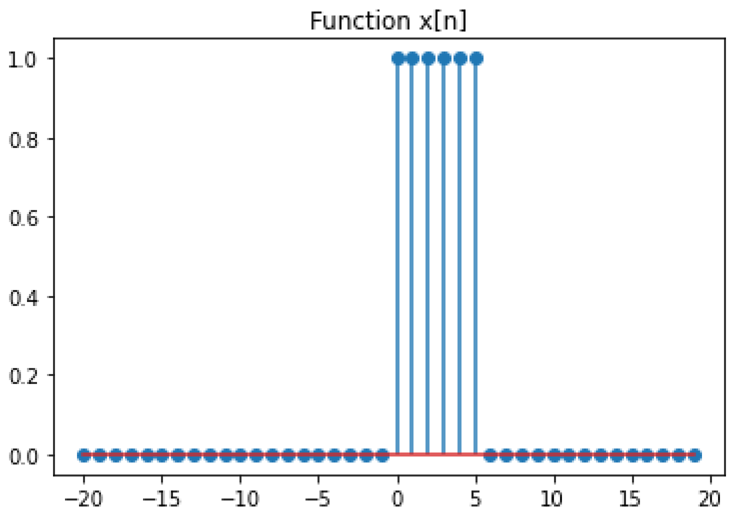
$$z[n] = \begin{cases} x[\frac{n}{2}] \;\;, & n \;\; even \\ 0 \;\;, & n \;\;\; odd \end{cases}$$

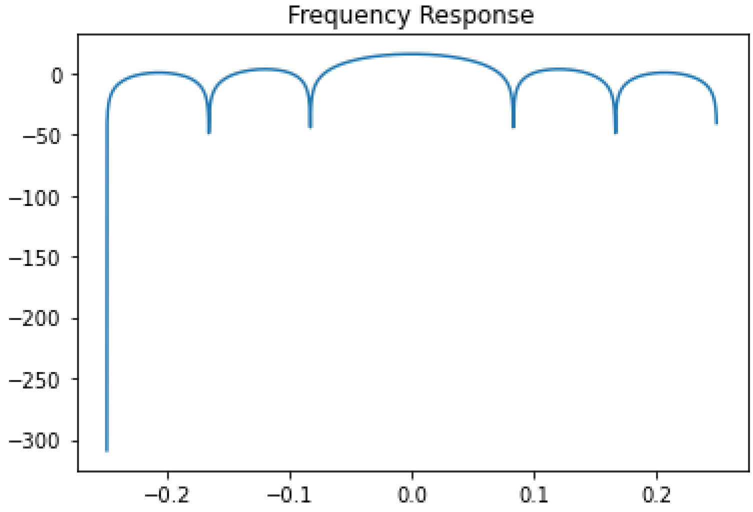
Plot $Z(e^{j\omega})$(magnitude and phase separately). What is the periodicity of the DTFT? Observe the effect of expanding time axis in the frequency domain.

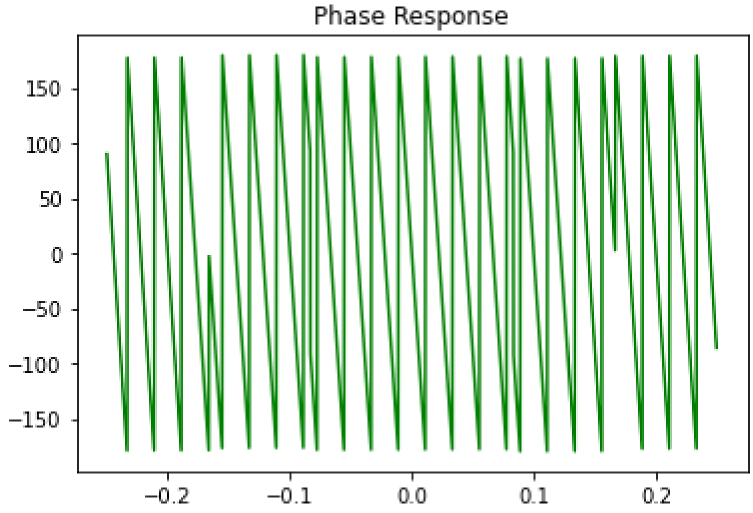
Ans) Periodicity of the dtft is π as obtained below from the frequency and phase response plot. We can see that the values repeat after 2048 units while the total plot

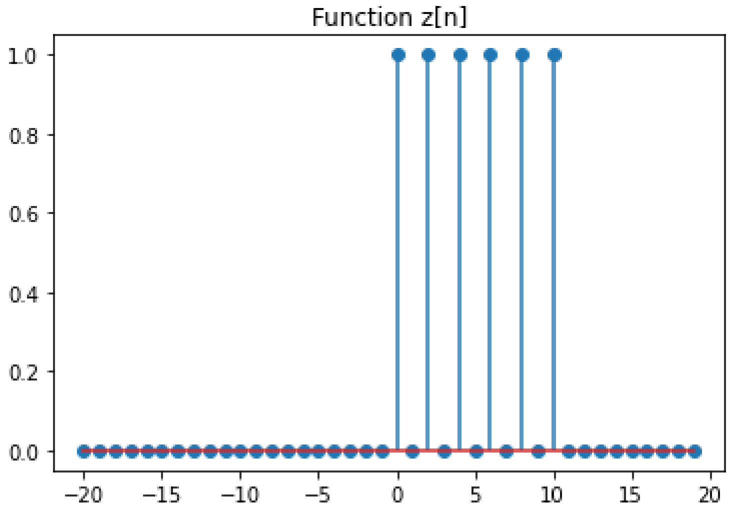
is for 4096 units. as it comprises of 2pi range. we can calculate by 2048/4096 * 2pi .after which we get Pi as the period.

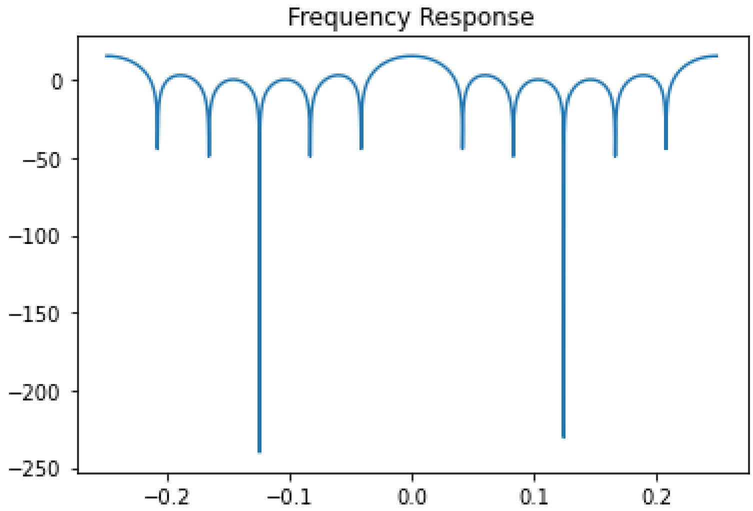
In [34]:
```
n = np.arange(-20,20)
x = [np.heaviside(n, 1) - np.heaviside(n-6, 1)]
plt.title('Function x[n]')
plt.stem(n,x[0])
plt.show()

a=[1]
c=np.arange(-1*np.pi, np.pi, 4*np.pi/4096)
w1,h1=sig.freqz(x[0],a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(4*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('Phase Response')
plt.plot(w1/(4*np.pi), angles, 'g')
plt.show()

x = [(np.heaviside((n/2), 1) - np.heaviside((n/2)-6, 1)) if  (n%2==0) else (
plt.title('Function z[n]')
plt.stem(n,x)
plt.show()

a=[1]
c=np.arange(-1*np.pi, 1*np.pi, 2*np.pi/4096)
w1,h1=sig.freqz(x,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(4*np.pi),h1db)
plt.show()
i=0
while (i!=4080):
    if (h1db[i]<-200): print(str(i))
    i+=1
print('length' +str(len(h1db)))
angles = np.angle(h1,deg=True)
plt.title('Phase Response')
plt.plot(w1/(4*np.pi), angles, 'g')
plt.show()
```



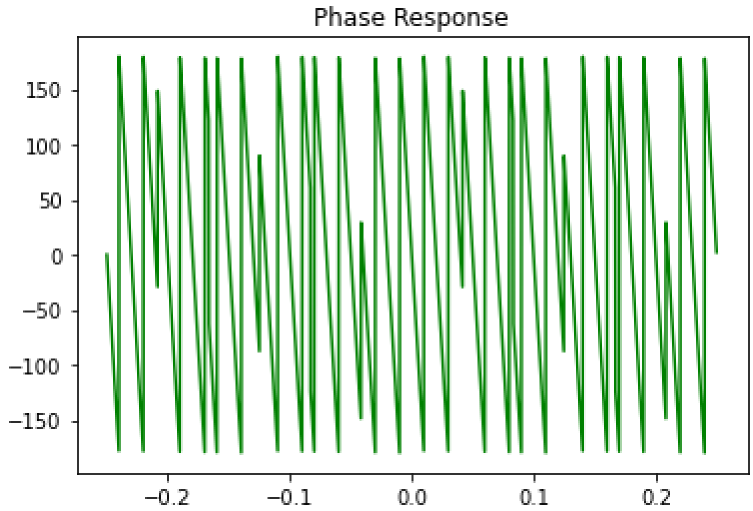

Frequency Response
0
−50
−100
−150
−200
−250
−300
−0.2
−0.1
0.0
0.1
0.2


Phase Response
150
100
50
0
−50
−100
−150
−0.2
−0.1
0.0
0.1
0.2


Function z[n]
1.0
0.8
0.6
0.4
0.2
0.0
−20
−15
−10
−5
0
5
10
15
20



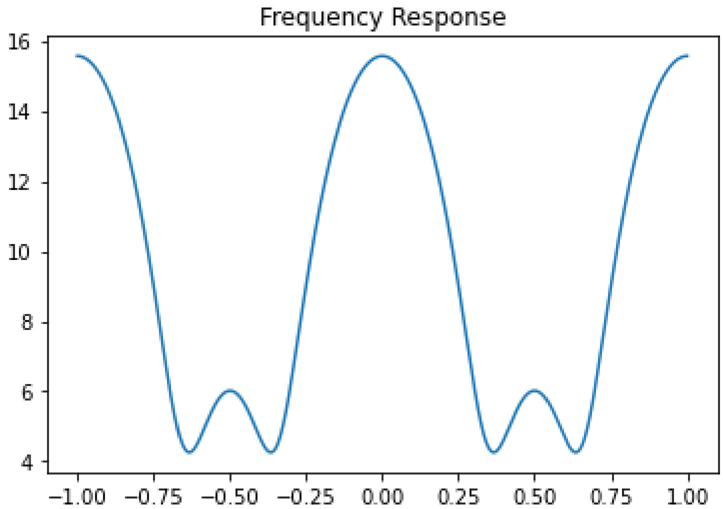

```
1024
3072
length4096
```

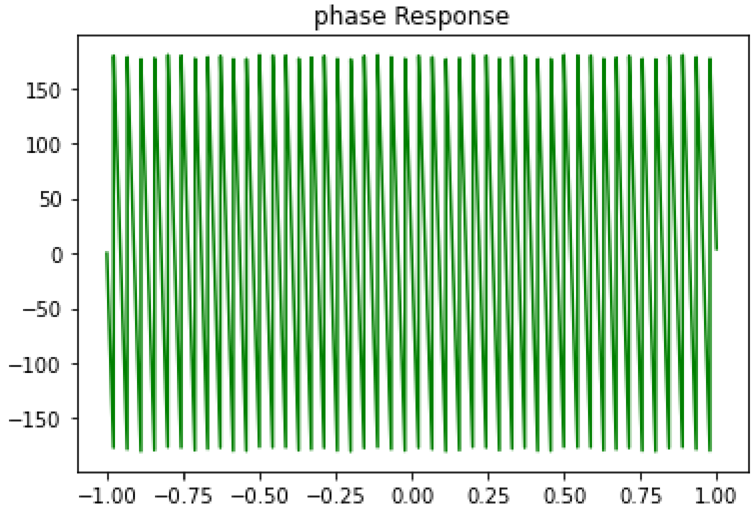


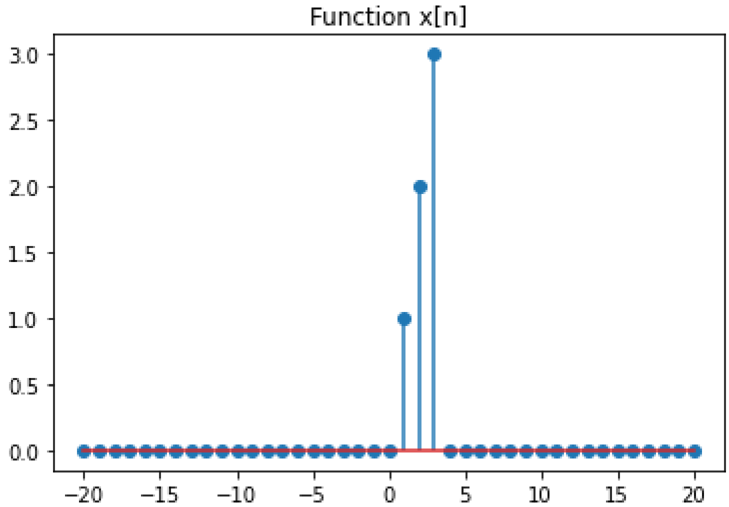
Q4. Consider the signals $x[n] = n(u[n] - u[n-4])$ and $y[n] = 0.9^n(u[n] - u[n-10])$. Find the convolution $z[n] = x[n] * y[n]$ of the signals. Plot $X(e^{j\omega})$, $Y(e^{j\omega})$ and $Z(e^{j\omega})$ (magnitude and phase separately).

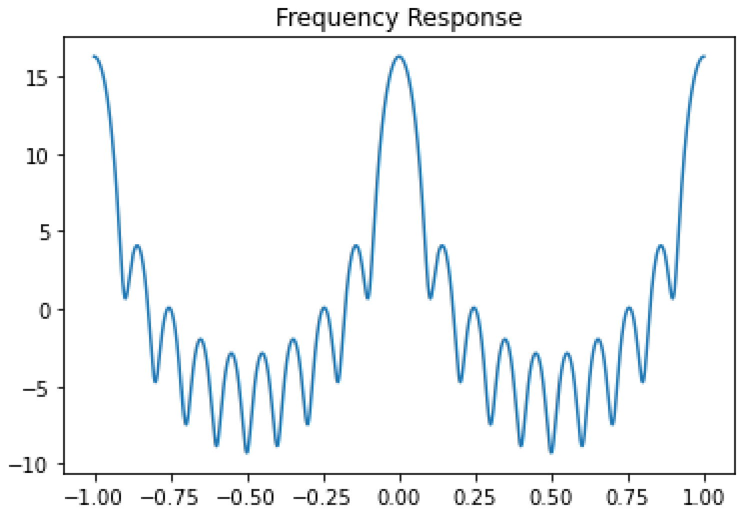
In [35]:
```
##code
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig

n= np.arange(-20, 21)
x = n * (np.heaviside(n,1)-np.heaviside(n-4,1))
y = np.power(0.9, n) * (np.heaviside(n,1)-np.heaviside(n-10,1))
z = np.convolve(x, y)

#x
a=[1]
c=np.arange(-2*np.pi, 2*np.pi, 4*np.pi/4096)
w1,h1=sig.freqz(x,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('phase Response')
```

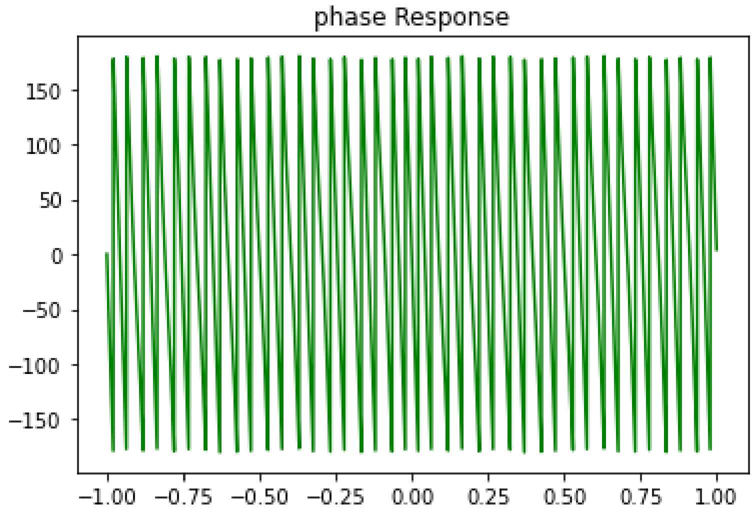
```
plt.plot(w1/(2*np.pi), angles, 'g')
plt.show()

plt.title('Function x[n]')
plt.stem(n,x)
plt.show()

#y
a=[1]
c=np.arange(-2*np.pi, 2*np.pi, 4*np.pi/4096)
w1,h1=sig.freqz(y,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('phase Response')
plt.plot(w1/(2*np.pi), angles, 'g')
plt.show()

plt.title('Function y[n]')
plt.stem(n,y)
plt.show()

#z
a=[1]
c=np.arange(-2*np.pi, 2*np.pi, 4*np.pi/4096)
w1,h1=sig.freqz(z,a,c)

h1db=20*np.log10(abs(h1))
plt.title('Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('phase Response')
plt.plot(w1/(2*np.pi), angles, 'g')
plt.show()

plt.title('Convoluted Function z[n]')
plt.stem(np.arange(-40,41),z)
plt.show()
```

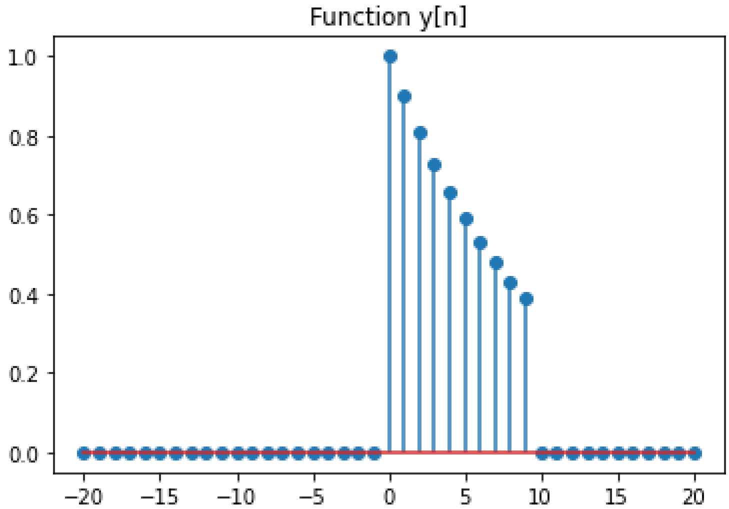


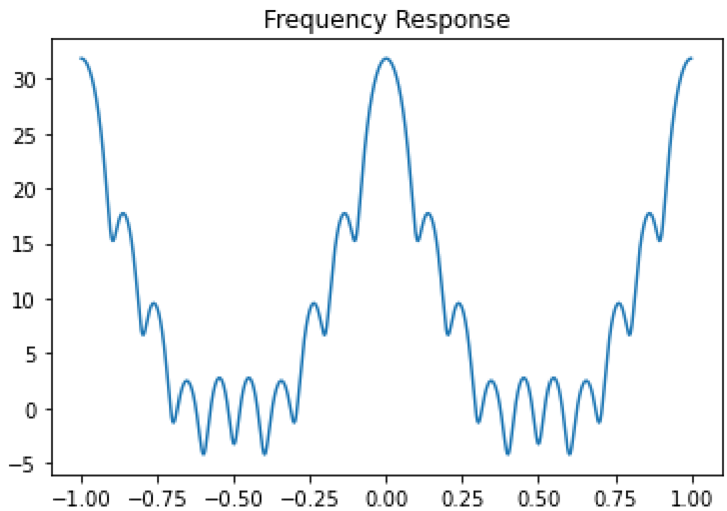

phase Response

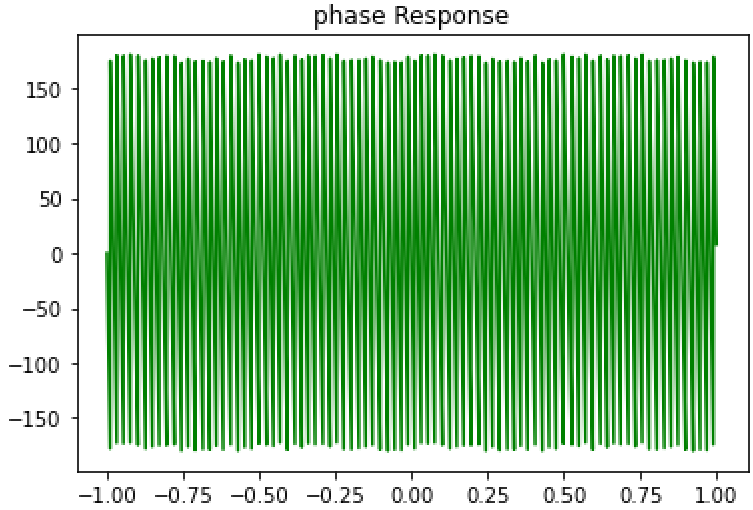

Function x[n]

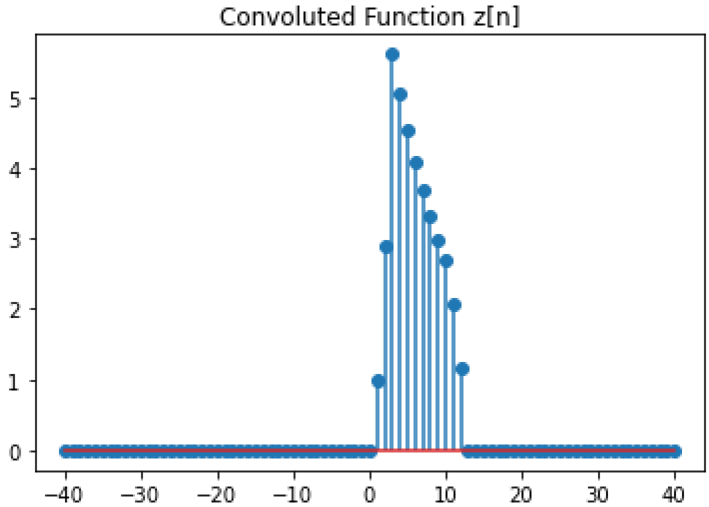

Frequency Response


phase Response
150
100
50
0
−50
−100
−150
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


Function y[n]
1.0
0.8
0.6
0.4
0.2
0.0
−20 −15 −10 −5 0 5 10 15 20


Frequency Response
30
25
20
15
10
5
0
−5
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

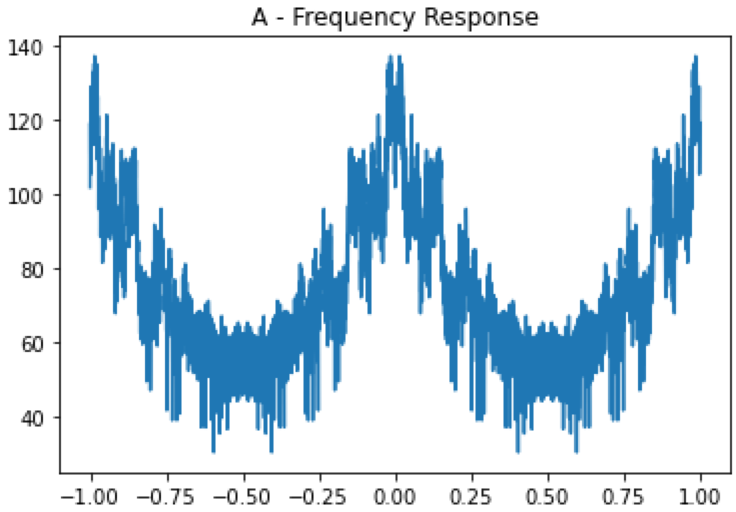


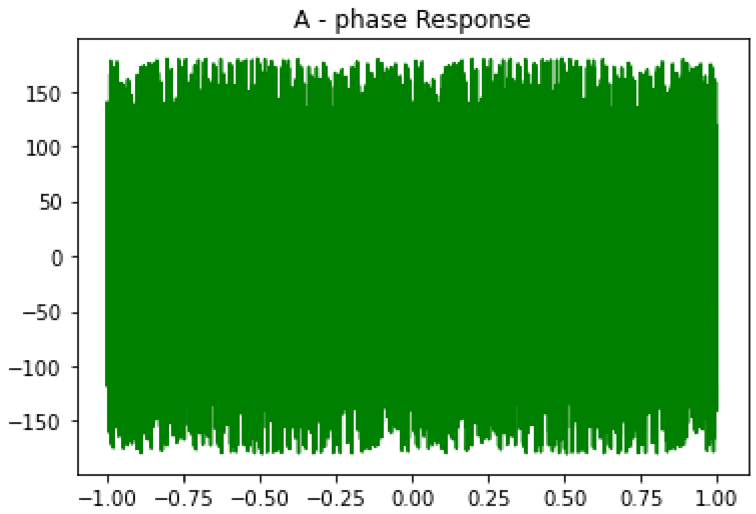


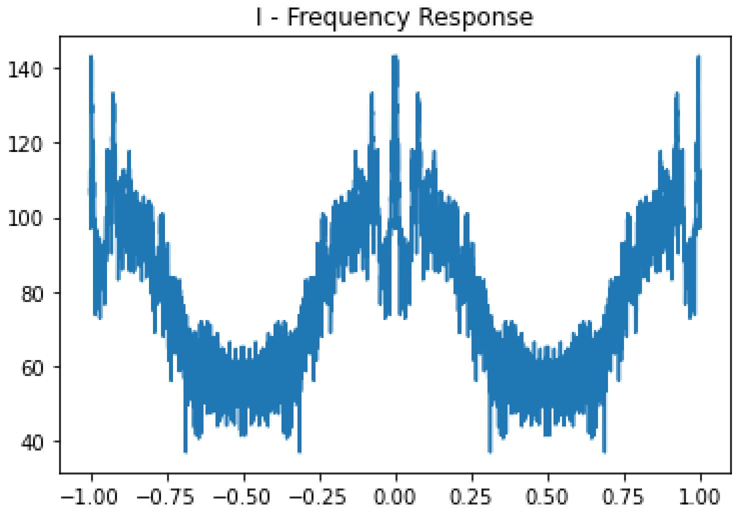
Q5. Find the DTFT of the given speech signals that corresponds to two vowels (/a/ and /i/). Which vowel has more high frequency content?

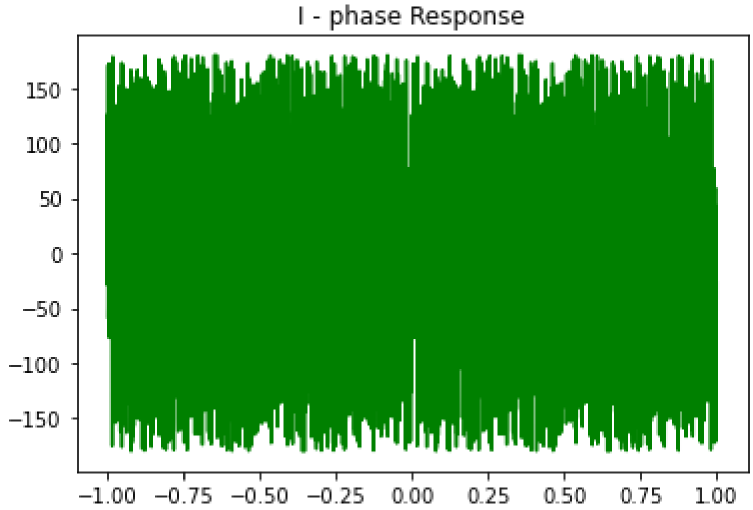
In [36]:
```
##code

import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
import scipy.io.wavfile as wav

c=np.arange(-2*np.pi, 2*np.pi, 4*np.pi/4096)

rate_a, data_a = wav.read('a.wav')

rate_i, data_i = wav.read('i.wav')

a=[1]
w1,h1=sig.freqz(data_a,a,c)

h1db=20*np.log10(abs(h1))
plt.title('A - Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('A - phase Response')
plt.plot(w1/(2*np.pi), angles, 'g')
plt.show()
```

```
afreq = sum(abs(h1))
w1,h1=sig.freqz(data_i,a,c)

h1db=20*np.log10(abs(h1))
plt.title('I - Frequency Response')
plt.plot(w1/(2*np.pi),h1db)
plt.show()

angles = np.angle(h1,deg=True)
plt.title('I - phase Response')
plt.plot(w1/(2*np.pi), angles, 'g')
plt.show()


ifreq = sum(abs(h1))
if(afreq>ifreq):
    print('A has more frequency content and hence higher frequency content'
elif(ifreq>afreq):
    print('I has more frequency content and hence higher frequency content'
else :
    print('Both have equal frequency content')
```









I has more frequency content and hence higher frequency content