Discrete Fourier Transform

Utkarsh Mahajan 201EC164

Arnav Raj Gupta 201EC109

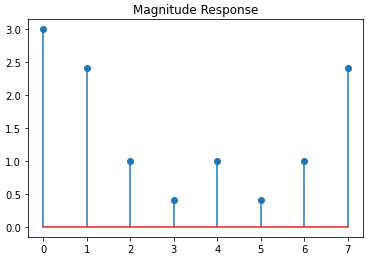
Q1. Compute the 8-point DFT of the following sequences. Plot the magnitude and phase spectrum. Observe the symmetry properties.

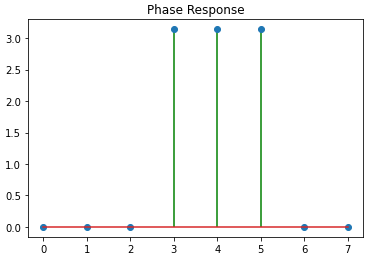
1. x[n] = {1,1,0,0,0,0,0,1}
2. x[n] = {0,1,1,0,0,0,-1,-1}
3. x[n] = {1,1,1,1,1,1,1,1}

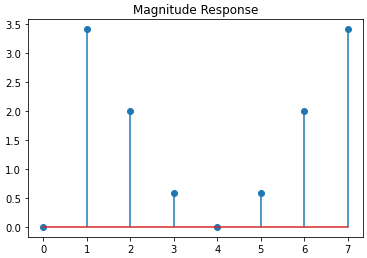
https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.fft.fft.html#numpy.fft.fft

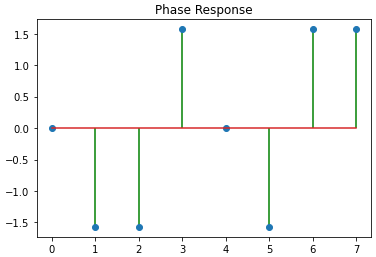
In [125…

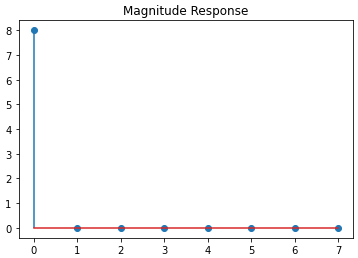
```
import numpy as np
import matplotlib.pyplot as plt

def plotdef(m, a):
    k = range(m)
    A = np.fft.fft(a,m)
    #print(A)
    plt.stem(k,abs(A))
    plt.title('Magnitude Response')
    plt.show()
    angles = np.angle(A)
    plt.stem(k, angles, linefmt='g')
    plt.title('Phase Response')
    plt.show()

plotdef(8,np.array([1,1,0,0,0,0,0,1]));
plotdef(8,np.array([0,1,1,0,0,0,-1,-1]));
plotdef(8,np.array([1,1,1,1,1,1,1,1]));
```

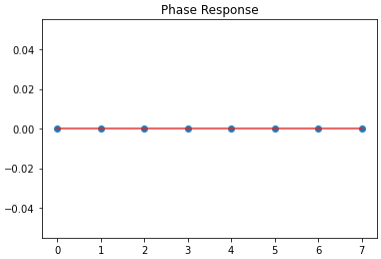












Symmetry property

As the signal is real, We can see that the DFT computed for the given sequences are all real in nature. So, the DFS is conjugate symmetric ex- X[i]=X*[-i].

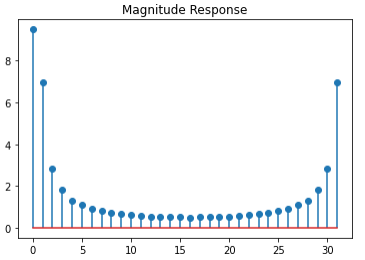
Q2. Generate the finite duration sequence $x[n] = 0.5(1-cos({\pi}n/20)), ~~0{\le}n{\lt}20$.

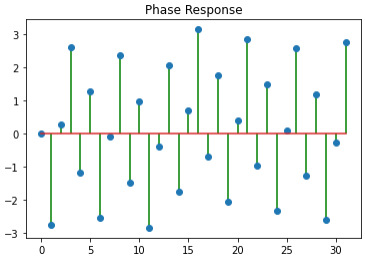
1. Append the sequence with sufficient number of zeros to compute the DFT of the sequence with length N= 32 and 256. Plot the DFT and comment on the effects of zero padding
2. Insert 16 zeros in the beginning of the sequence and repeat.

In [123…

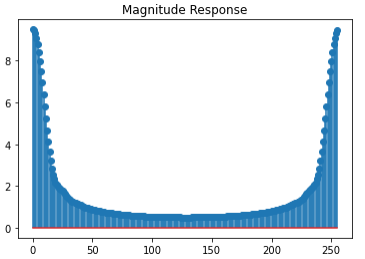
```
x = [0.5*(1-np.cos((n*np.pi)/20)) for n in range(0,20)]
#1
print('1)N=32')
plotdef(32,np.array([*x,*([0]*(32-20))]));
print('1)N=256')
plotdef(256,np.array([*x,*([0]*(256-20))]));
#2
print('2)N=36')
plotdef(36,np.array([*([0]*(16)),*x]));
print('2)N=256')
plotdef(256,np.array([*([0]*(16)),*x,*([0]*(256-20-16))]));
```

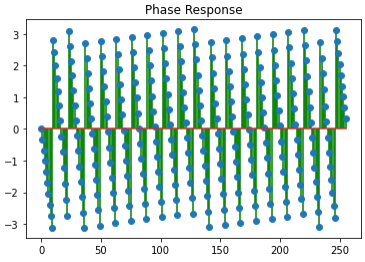
1)N=32



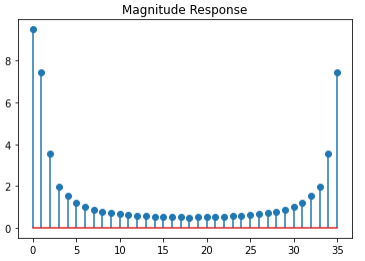


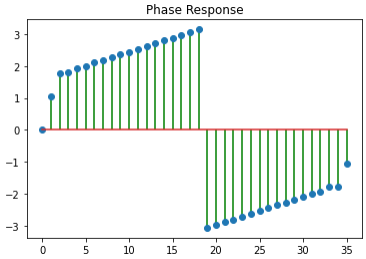
1)N=256



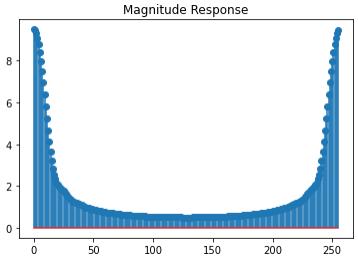


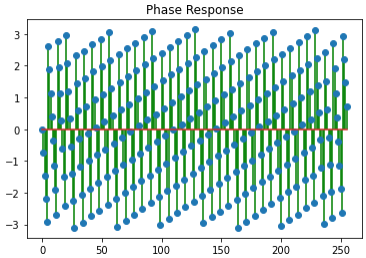
2)N=36





2)N=256





Effect of zero padding on DFT:

Zero padding refers to adding zero at the end of a discrete time-domain signal to increase its length. It results in more accurate amplitude estimates of resolvable signal components. While, zero padding does not improve the spectral (frequency) resolution of the DFT. The resolution is determined by the no of samples and the sample rate.

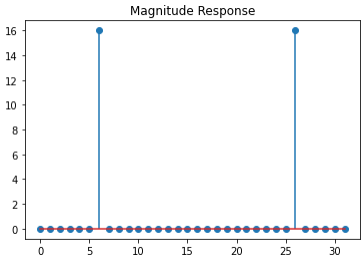
Q3. Plot the 32 point DFT of the following signals and comment on the spectral estimate.

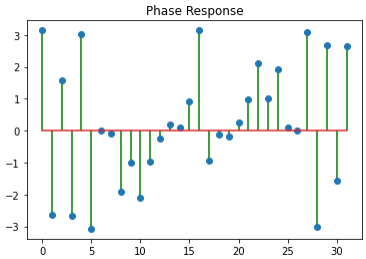
1. $x[n] = cos(3{\pi}n/8)$
2. $x[n] = cos(3{\pi}n/16)$
3. $x[n] = cos(3{\pi}n/17)$

In [116…

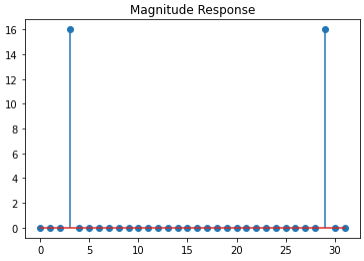
```
print('3.1')
plotdef(32,np.array([np.cos(3*n*np.pi/8) for n in range(0,32)]));
print('3.2')
plotdef(32,np.array([np.cos(3*n*np.pi/16) for n in range(0,32)]));
print('3.3')
plotdef(32,np.array([np.cos(3*n*np.pi/17) for n in range(0,32)]));
```

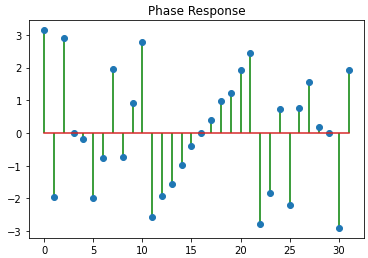
3.1



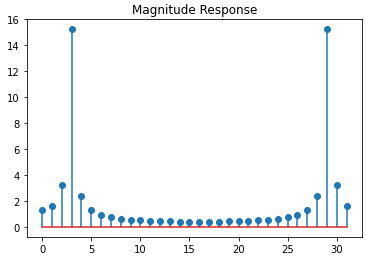


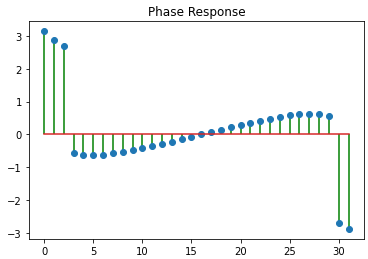
3.2





3.3





Spectral estimate:

The period of the first signal is 16 ,so when we plot a 32 point DFT for this, two copies of the signal is taken. However , since 32 is a multiple of 16 , we get an accurate spectral estimate. Similarly, for the second signal, the period is 32, so in this case also , the estimate would be accurate. However , in the third case , the period of the signal is 34 and we are considering only a 32 point DFT, this results in spectral leakage and the spectral estimate would not be accurate.

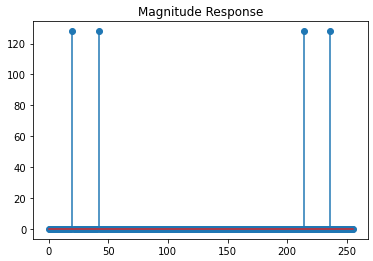
Q4. Estimation of signal corrupted by noise: Generate the signal $s[n] = cos({\pi}5n/32) +cos({\pi}21n/64) $ for 256 samples. Generate a noisy version of the signal x[n] by adding white gaussian noise e[n] with variance 0.3 to the original signal s[n]. Compute the energy in the error signal $\sum_{n}^{}(x[n] - s[n])^2$

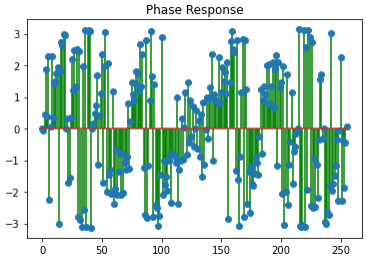
- Plot the 256 point DFTs $S[k]$ and $X[k]$. Find the set S_{nz} of the values of k for which $S[k]$ is non-zero.
- Find $\hat{X}[k] = X[k]$ if $k~ \in \mathcal ~{S_{nz}}$ and zero otherwise. Find the IDFT $\hat{x}[n]$ from $\hat{X}[k]$ to recover the original signal. Plot $\hat{x}[n]$. Find the error signal energy after recovery $\sum_{n}^{}(\hat{x}[n] - s[n])^2$

In [122…

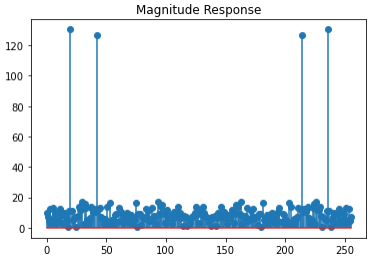
```
def plotidft(m, a):
    k = range(m)
    A = np.fft.ifft(a,m)
    plt.stem(k,abs(A))
    plt.show()


s = [ np.cos((5*n*np.pi)/32)+np.cos((21*n*np.pi)/64) for n in range(0,256)]
e = np.random.normal(0, np.sqrt(0.3), 256)
x = [ s[i]+e[i] for i in range(0,256)]
print('for s[k]')
plotdef(256,np.array(s))
print('for x[k]')
plotdef(256,np.array(x))
energy=0
for i in range (0,256):
    energy += (x[i]-s[i])*(x[i]-s[i])
print('energy in error signal:' + str(energy) + ' units')
print("s[k] is non zero for k:")
snz = []
sd = np.fft.fft(s,256) #s[k]
for i in range (0,256):
    if(sd[i]!=0): snz.append(i);
print(snz)

#xt[k]=x[k] if k is in snz or zero
xt = np.zeros(256)
for i in range(0,256):
    if(abs(sd[i])>=1): xt[i]=x[i];
print("Inverse DFT of x'[k]:")
plotidft(256,np.array(xt))


energy=np.sum(np.power(xt-s,2))
print('energy in error signal:' + str(energy) + ' units')
```

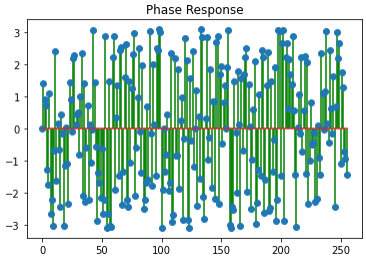
for s[k]



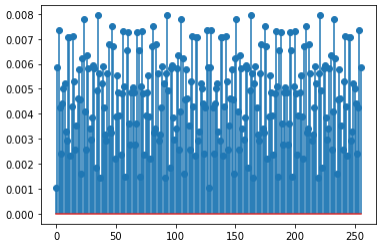



```
for x[k]
```






```
energy in error signal:78.06777243230599 units
s[k] is non zero for k:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255]
Inverse DFT of x'[k]:
```



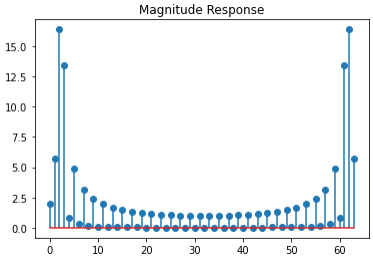
```
energy in error signal:255.562560563063 units
```

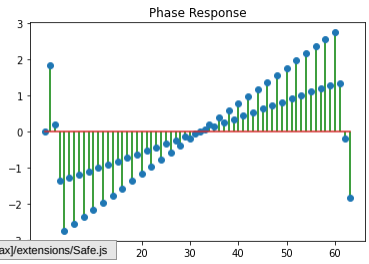
Q5. Generate the following signals $x_1[n] = cos({\pi}n/15),~~x_2[n] = cos(3{\pi}n/15)$ for n = 0 to 31. Plot the magnitude of 64 point DFT of the two signals $X_1[k]$ and $X_2[k]$ (with zero padding). Now concatenate the two signals to form a 64 length sequence $x[n]$. Plot the 64 point DFT magnitude $X[k]$ of $x[n]$. Also find the sum of two signals and plot the 64 point DFT of the result. (Based on the results, comment on the limitation of DFT.)

In [113…

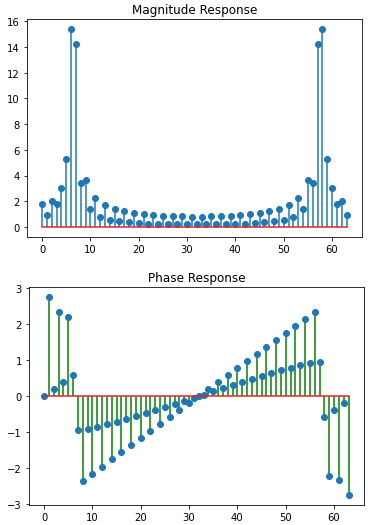
```
x1 = [ np.cos(n*np.pi/15) for n in range(0,32) ]
x2 = [ np.cos(3*n*np.pi/15) for n in range(0,32) ]
print('x1[k]')
plotdef(64,np.array([*x1,*[0]*32]))
print('x2[k]')
plotdef(64,np.array([*x2,*[0]*32]))
print('x12[k]')
plotdef(64,np.array([*x1,*x2]))
xs = np.zeros(64)
for i in range(0,32):
    xs[i]= x1[i]+x2[i]
print('X[k]')
plotdef(64,np.array([*xs]))
```

```
x1[k]
```

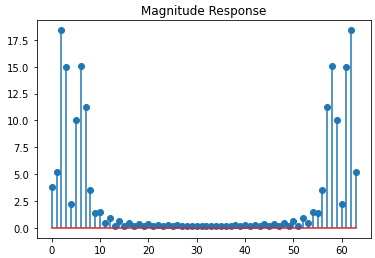


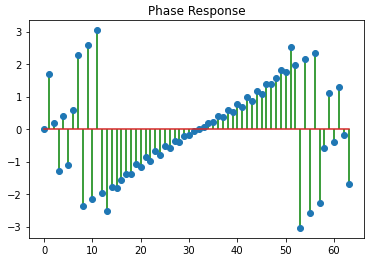


x2[k]

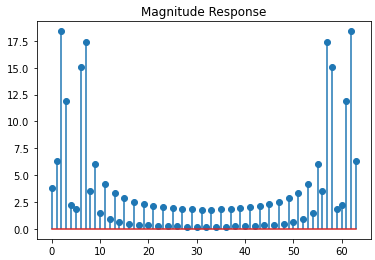


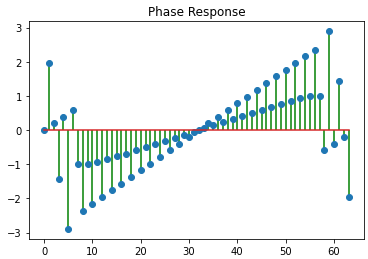
x12[k]





X[k]





From above, We can conclude the following limitations of DFT: picket-fence, leakage, aliasing effects and amplitude modulation spectrum.

In this question, we see that period of the two signals is not an integer multiple of 32 or 64. This results in spectral leakage , a limitation of DFT

In [ ]: