191IT234_Niraj_Nandish

August 25, 2020

1 Lab Week 2

1.0.1 Name: Niraj Nandish

1.0.2 Roll no.: 191IT234

1.0.3 Semester: 3

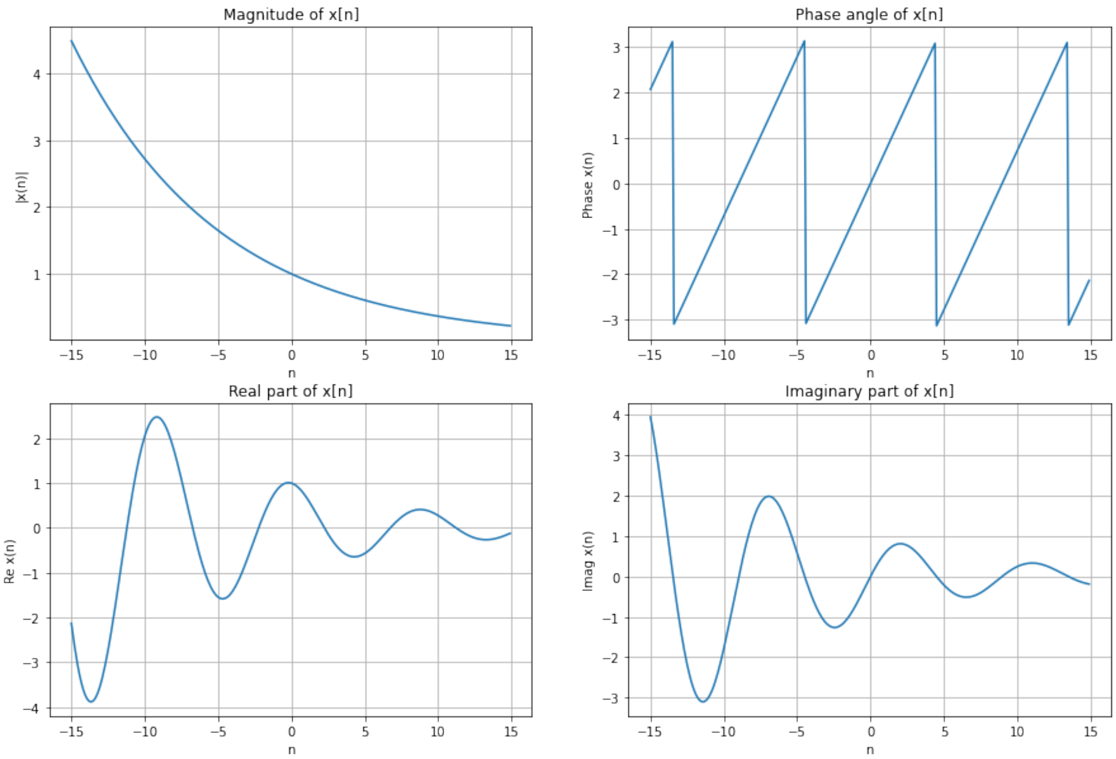
[2]:
```
# Import all required libraries
import numpy as np                                       # Contains built-in␣
 ↪functions to work on arrays
import matplotlib.pyplot as plt                          # Used to plot graphs
from scipy.integrate import quad as integrate            # Used directly to␣
 ↪integrate functions with limits
import sounddevice as sd                                 # Used to record audio␣
 ↪input from user
```

1.1 Question 1

Generate the complex-valued signal x(n)= exp(-0.1+j0.7)n , -15 n 15, and plot its magnitude, phase, the real part, and the imaginary part in four separate subplots.

[2]:
```
n = np.arange(-15,15,0.1)
x = np.exp(n*(-0.1 + 0.7j))

mag = np.absolute(x)
phase = np.angle(x)
real = np.real(x)
imag = np.imag(x)

fig, ax = plt.subplots(2,2, figsize=(15,10))

# Magnitude
ax[0,0].plot(n, mag);
ax[0,0].set(xlabel='n', ylabel='|x(n)|')
ax[0,0].set_title('Magnitude of x[n]')
ax[0,0].grid(True)

# Phase angle
```

```python
ax[0,1].plot(n, phase);
ax[0,1].set(xlabel='n', ylabel='Phase x(n)')
ax[0,1].set_title('Phase angle of x[n]')
ax[0,1].grid(True)

# Real part
ax[1,0].plot(n, real);
ax[1,0].set(xlabel='n', ylabel='Re x(n)')
ax[1,0].set_title('Real part of x[n]')
ax[1,0].grid(True)

# Imaginary part
ax[1,1].plot(n, imag);
ax[1,1].set(xlabel='n', ylabel='Imag x(n)')
ax[1,1].set_title('Imaginary part of x[n]')
ax[1,1].grid(True)

plt.show()
```

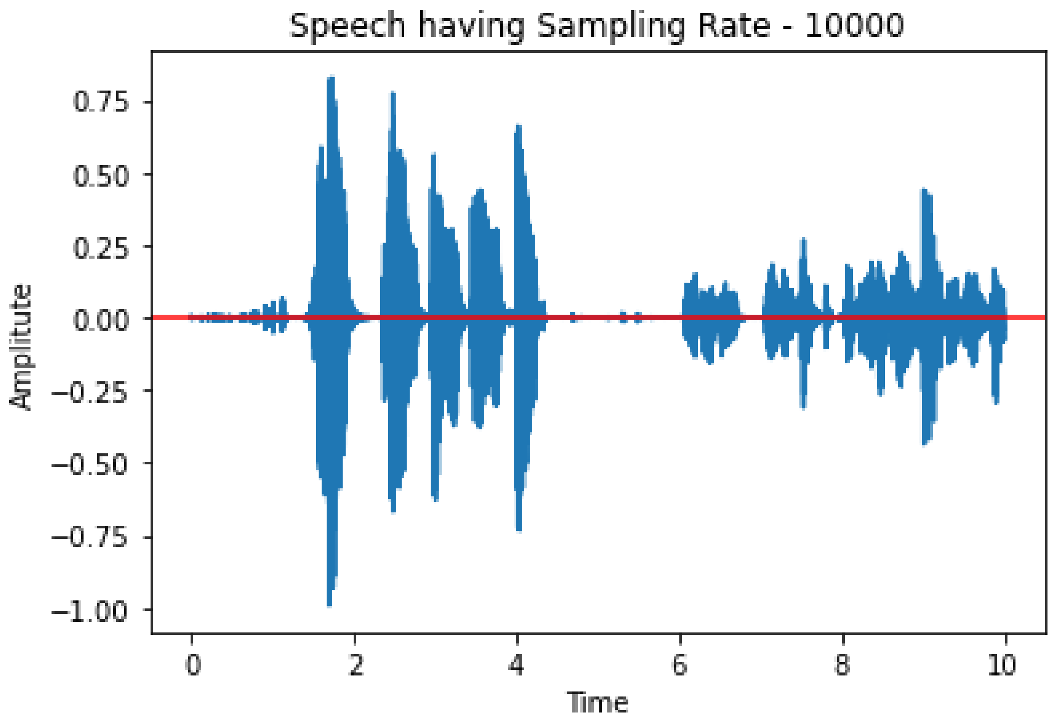


1.2 Question 2

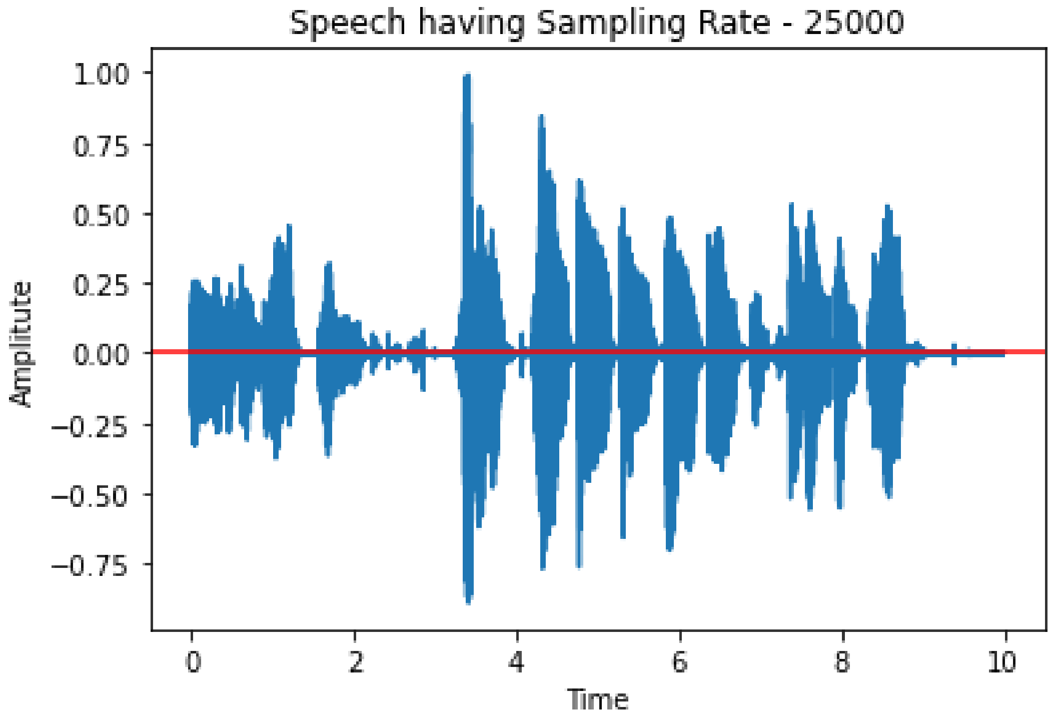
Record 10 seconds of your speech with a microphone with different sampling rate (atleast 3) and plot the signal as a function of time.

```
[3]: sec = 10;

     for sample_rate in [10000, 25000, 50000]:
         print("Recording....")
         res = sd.rec(int(sample_rate*sec),sample_rate,1)
         sd.wait()
         print("Recording has stopped! Your signal:")
         time = np.arange(0,10,1/sample_rate)

         plt.plot(time,res)
         plt.axhline(y=0, color='r')
         plt.xlabel('Time')
         plt.ylabel('Amplitute')
         plt.title('Speech having Sampling Rate - {}'.format(sample_rate))
         plt.show()
```

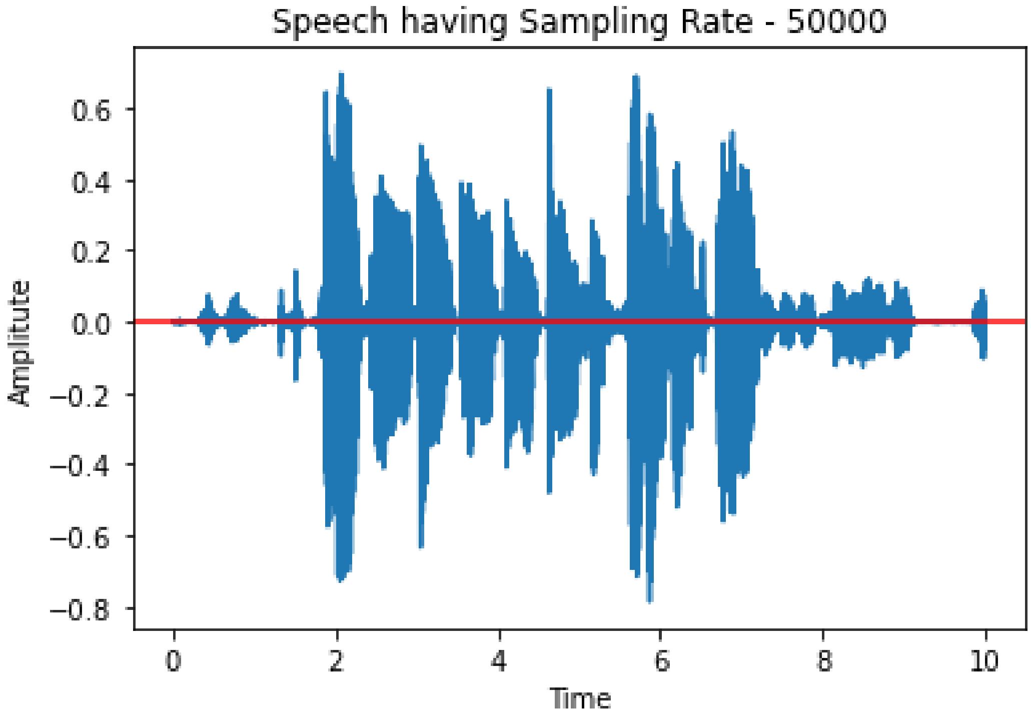
```
Recording…
Recording has stopped! Your signal:
```



```
Recording…
Recording has stopped! Your signal:
```



```
Recording…
Recording has stopped! Your signal:
```

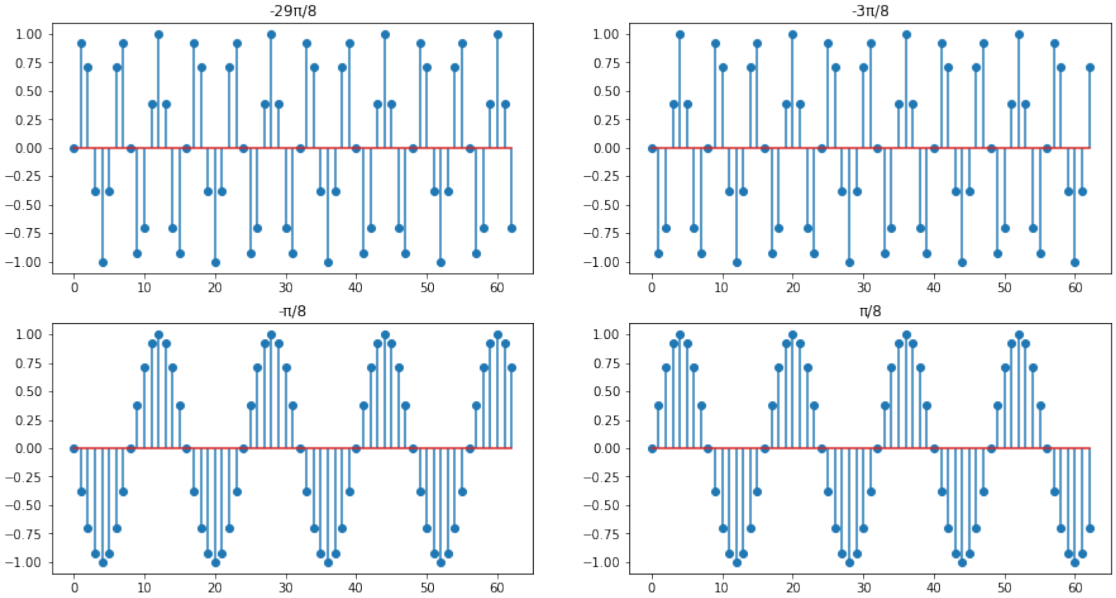


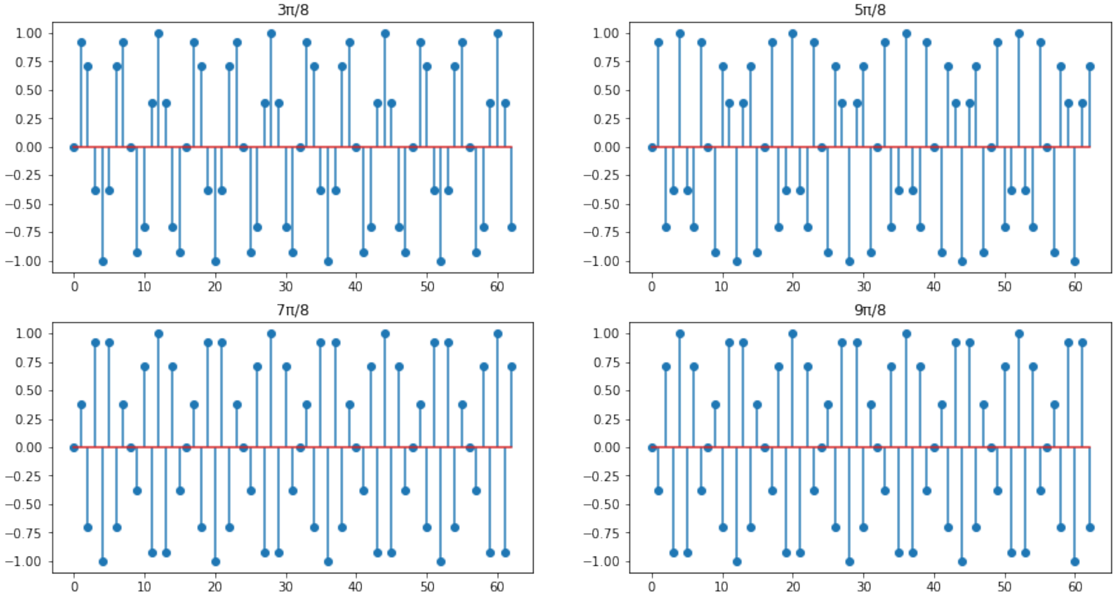
1.3 Question 3

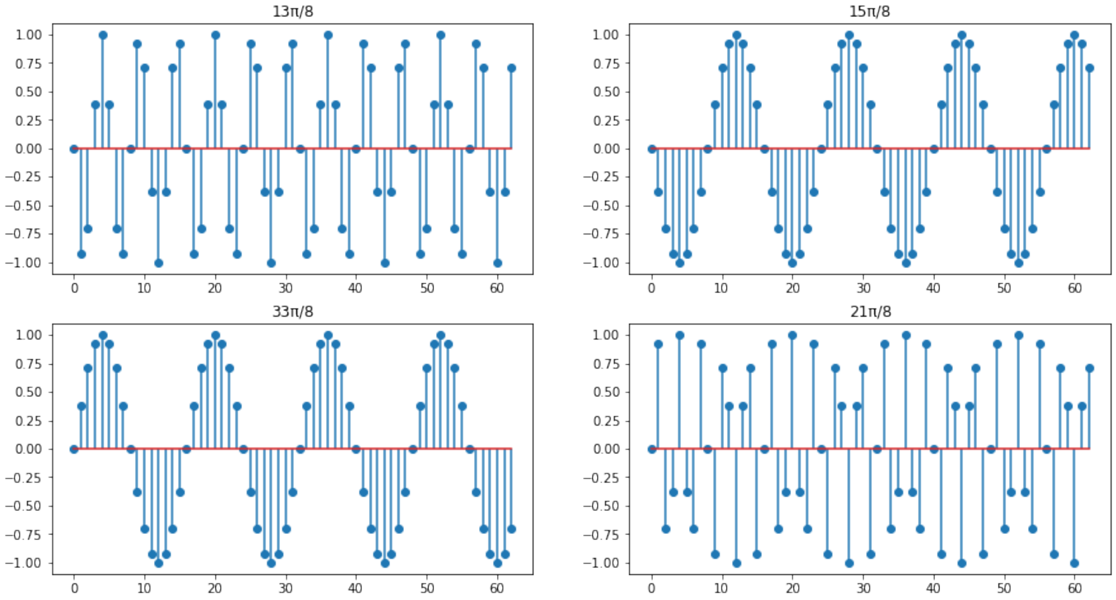
[5]:

```
# 1. Plot the discrete-time signal x[n] = sin ( 0n) for the following values of␣
 ↪ 0: -29 /8 , -3 /8 , - /8 ,  /8 , 3 /8 , 5 /8 , 7 /8 , 9 /8 , 13 /8 , 15 /8 , 33 /
 ↪8 , and 21 /8 .
## a. Plot each signal for 0   n   63.
## b. Label each graph with the frequency.
## c. Use the subplot function to plot four graphs per figure.

time = np.arange(0,63,1)
n = time*np.pi/8
x1 = np.sin((-29)*n)
x2 = np.sin((-3)*n)
x3 = np.sin((-1)*n)
x4 = np.sin(n)
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,figsize=(15,8))
ax1.stem(time,x1,use_line_collection=True)
ax1.set_title('-29 /8')
ax2.stem(time,x2,use_line_collection=True)
ax2.set_title('-3 /8')
ax3.stem(time,x3,use_line_collection=True)
ax3.set_title('- /8')
ax4.stem(time,x4,use_line_collection=True)
ax4.set_title(' /8')

x1 = np.sin(3*n)
x2 = np.sin(5*n)
x3 = np.sin(7*n)
x4 = np.sin(9*n)
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,figsize=(15,8))
ax1.stem(time,x1,use_line_collection=True)
ax1.set_title('3 /8')
ax2.stem(time,x2,use_line_collection=True)
ax2.set_title('5 /8')
ax3.stem(time,x3,use_line_collection=True)
ax3.set_title('7 /8')
ax4.stem(time,x4,use_line_collection=True)
ax4.set_title('9 /8')

x1 = np.sin(13*n)
x2 = np.sin(15*n)
x3 = np.sin(33*n)
x4 = np.sin(21*n)
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,figsize=(15,8))
```

```
ax1.stem(time,x1,use_line_collection=True)
ax1.set_title('13 /8')
ax2.stem(time,x2,use_line_collection=True)
ax2.set_title('15 /8')
ax3.stem(time,x3,use_line_collection=True)
ax3.set_title('33 /8')
ax4.stem(time,x4,use_line_collection=True)
ax4.set_title('21 /8')
plt.show()
```

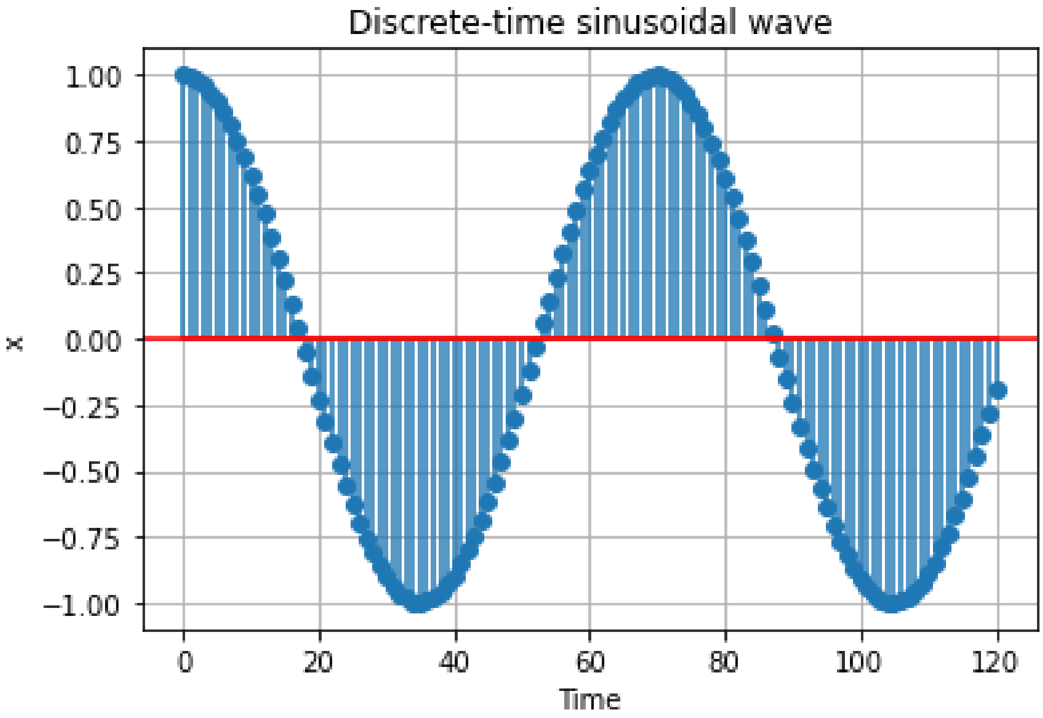






[3]:

```
# 2. Plot Discrete-time signal x[n] = cos(0.09n) for 0  n  120.
# For your plot, turn the grid on and scale the axes using the python␣
 ↪statements grid

time = np.arange(0,121)
amp = np.cos(0.09*time)

plt.stem(time, amp, use_line_collection=True)
plt.grid(True)
plt.axhline(y=0, color='r')
plt.title('Discrete-time sinusoidal wave')
plt.xlabel('Time')
plt.ylabel('x')
plt.show()
```



1.3.1 3. Is this signal periodic? Explain.

Answer:

x[n] = cos(0.09n) is not periodic, as we need x[n + N] = x[n]   n   Z.
This follows that x[n + N] = cos(0.09n + 0.09N) => N =   = m * 2 /0.09 where m   Z.
Therefore N is not an integer and hence the signal is not periodic.

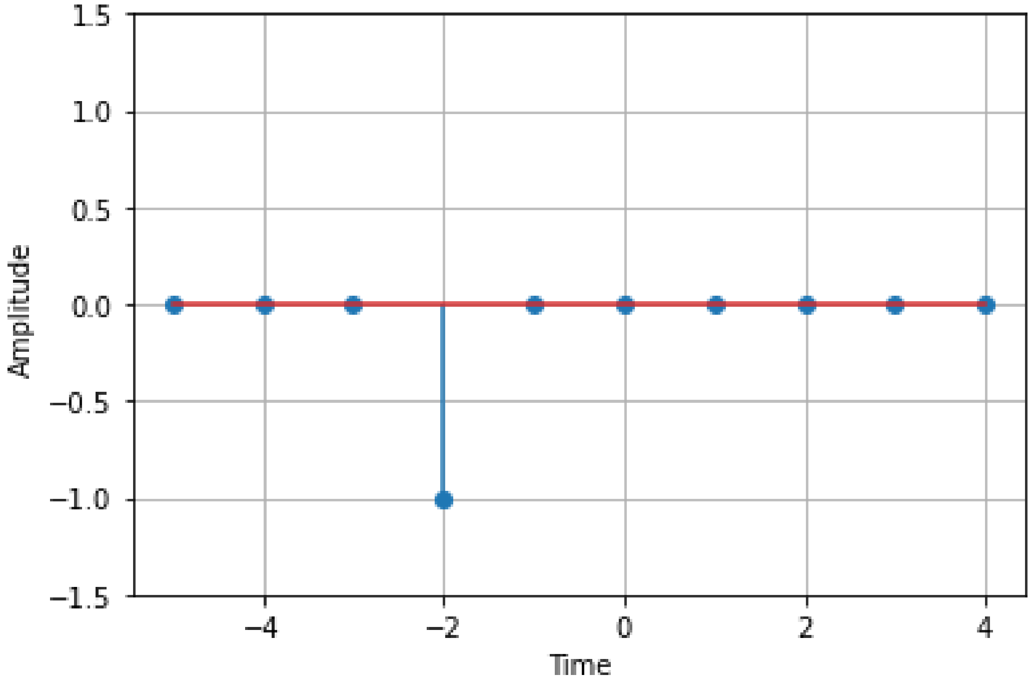
1.4 Question 4

Consider an input x[n] and a unit impulse response h[n] given by x[n] = (1/2)n-2*u[n-2]; h[n] = u[n+2].

Determine and plot the output y[n] = x[n] * h[n]

[7]:

```
n = np.arange(-5,5,1)
u = np.where(n-2==0,1,0)
x = np.array(0.5*n - 2*u)
h = np.where(n+2==0,1,0)
y = x*h
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.stem(n,y,use_line_collection=True)
plt.grid(True)
plt.ylim([-1.5,1.5])
plt.show()
```



1.5 Question 5

Consider the following discrete-time signals with a fundamental period of 6:

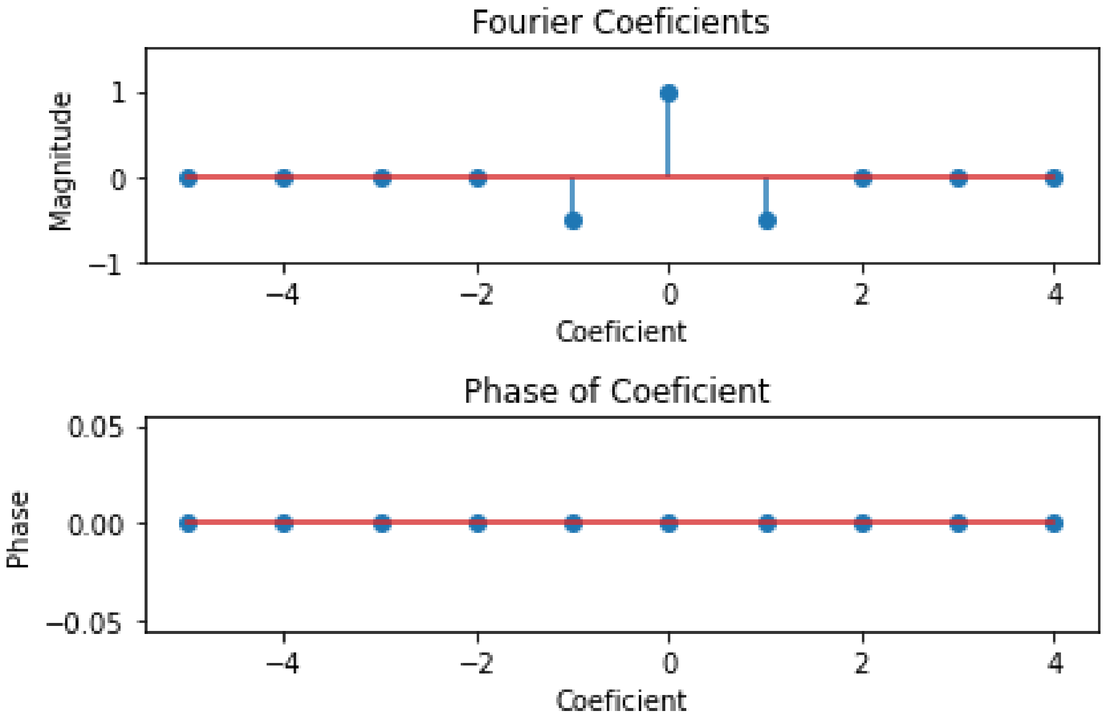
x[n] = 1 – cos((2 /6)*n)

Determine the Fourier series coefficients. Plot the magnitude and phase of each coefficients.

[6]:

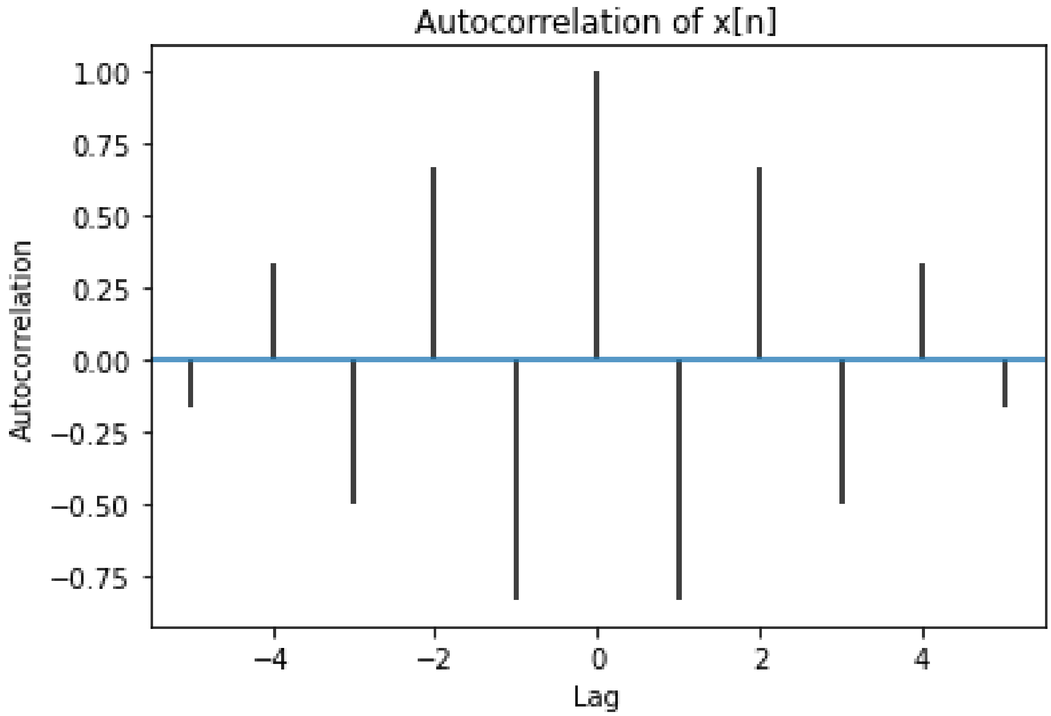
```
n = np.arange(-5,5)
y = np.zeros(n.size)
y[4] = -0.5
y[5] = 1
y[6] = -0.5

fig, ax = plt.subplots(2)

ax[0].stem(n, y, use_line_collection=True)
ax[0].set_xlabel("Coeficient")
ax[0].set_ylabel("Magnitude")
ax[0].set_title("Fourier Coeficients")
ax[0].set_ylim(-1,1.5)

y = np.zeros(n.size)
```

```
ax[1].stem(n, y, use_line_collection=True)
ax[1].set_xlabel("Coeficient")
ax[1].set_ylabel("Phase")
ax[1].set_title("Phase of Coeficient ")
plt.tight_layout()
plt.show()
```

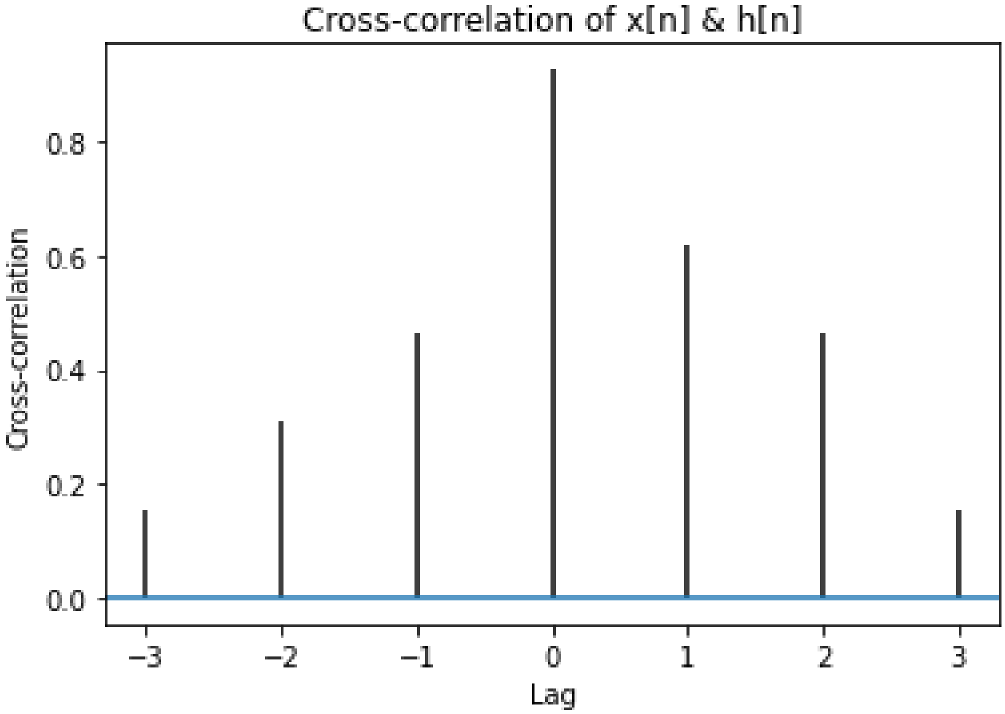


1.6 Question 6

[9]:
```
# 1. Find the autocorrelation of x[n] = [1, -1, 1, -1, 1, -1]. Plot the output.

x = np.array([1.0, -1.0, 1.0, -1.0, 1.0, -1.0])
plt.acorr(x, maxlags=5)
plt.title('Autocorrelation of x[n]')
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
plt.show()
```



[10]:
```
# 2. Find the cross correlation between two sequences x[n] and h[n]: x[n] = [1,␣
↪0, 2, 1]; h[n] = [1, 1, 2, 1]

x = np.array([1.0, 0, 2.0, 1.0])
h = np.array([1.0, 1.0, 2.0, 1.0])
plt.xcorr(x, h, maxlags=3)
plt.title('Cross-correlation of x[n] & h[n]')
plt.xlabel('Lag')
plt.ylabel('Cross-correlation')
plt.show()
```


Cross-correlation of x[n] & h[n]
Cross-correlation
0.0
0.2
0.4
0.6
0.8
−3
−2
−1
0
1
2
3
Lag