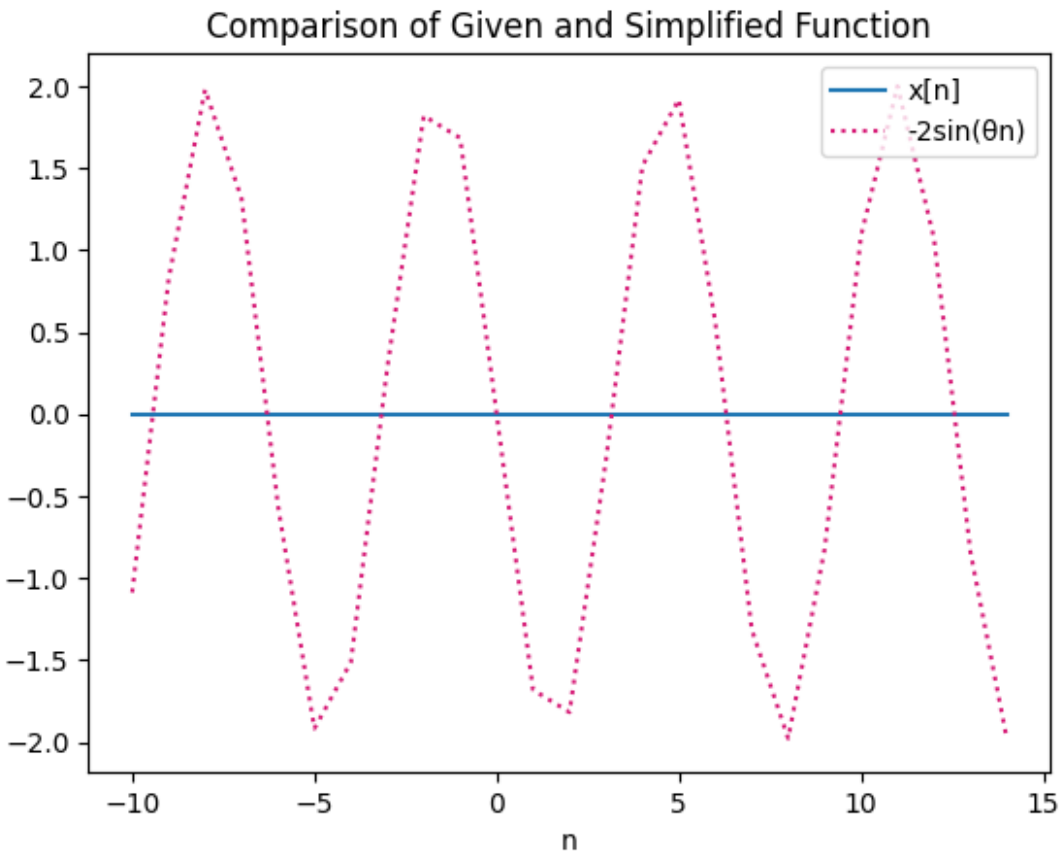
##2.1 COMPLEX ALGEBRA 1

```
import numpy as np

n = 3 #sample value only

x1 = 0
x2 = n*1j #n can be any real number

y1 = -1
y2 = 0j

z1 = 1
z2 = 0j

#equations
a = -(1/n)*abs(x1+x2)
b = -abs(y1+y2)
c = z2-z1

print("a: ", a.real)
print("b: ", b.real)
print("c: ", c.real)
```

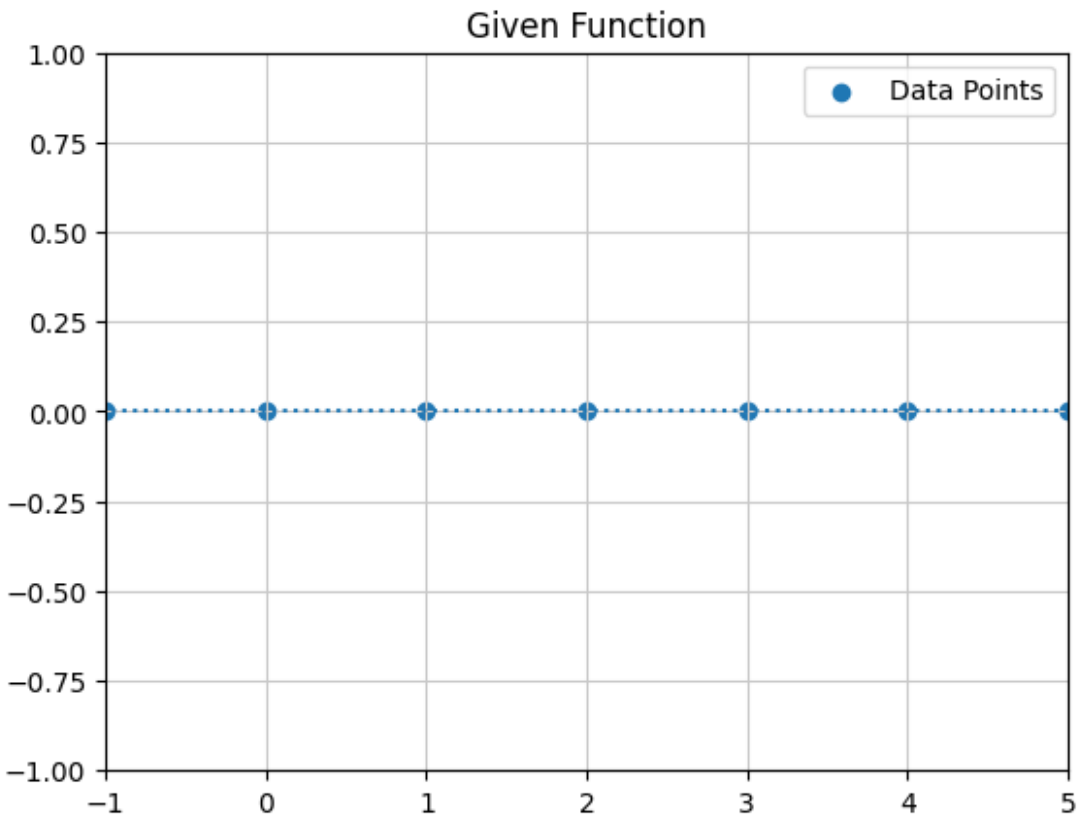
```
a:  -1.0
b:  -1.0
c:  -1.0
```

##2.2 COMPLEX ALGEBRA 2

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

n_val = np.arange(-10,15)
theta = 1

#equations
xn = (np.exp(1j * np.pi/2) + np.exp(-1j * np.pi/2)) / 2 - (np.exp(1j *
theta * n_val) + np.exp(-1j * theta * n_val)) / 1j
xn1 = -2*np.sin(theta*n_val)

plt.plot(n_val, xn.real, label='x[n]' )
plt.plot(n_val, xn1, linestyle=':', label='-2sin(θn)',
color='#d61573')
plt.title('Comparison of Given and Simplified Function')
plt.xlabel('n')
plt.legend(loc='upper right')
plt.show()
```




```
import matplotlib.pyplot as plt

plt.xlim(-1, 5)
plt.ylim(-1, 1)
plt.scatter(n_val, xn.real, label='Data Points')
plt.plot(n_val, xn.real, linestyle=':')

plt.title('Given Function')
plt.grid(True, color='#c9c9c9')
plt.legend()
plt.show()
```



##2.3 PERIODIC SIGNALS

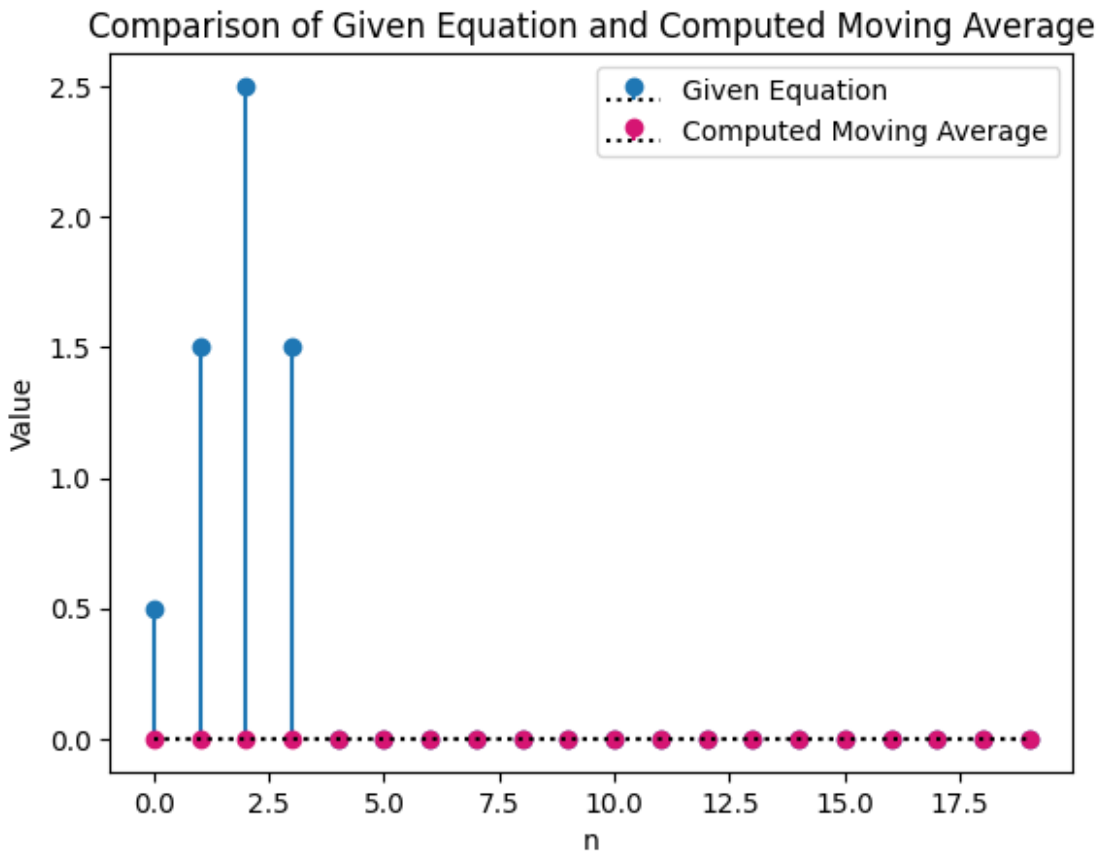
```
import numpy as np
import math

def calculate_period(M,N):
  if M % N == 0:
    return N
  else:
    return N//math.gcd(M,N)

vals = [(1,1),(1,3),(56,9)] # M,N

for M,N in vals:
  min_period = calculate_period(M,N)
  print("The minimum period for e^[j(",M,"/",N,")2πn] is",min_period,
"sample/s")
```

```
The minimum period for e^[j( 1 / 1 )2πn] is 1 sample/s
The minimum period for e^[j( 1 / 3 )2πn] is 3 sample/s
The minimum period for e^[j( 56 / 9 )2πn] is 9 sample/s
```

2.4 SIGNAL OPERATIONS 1

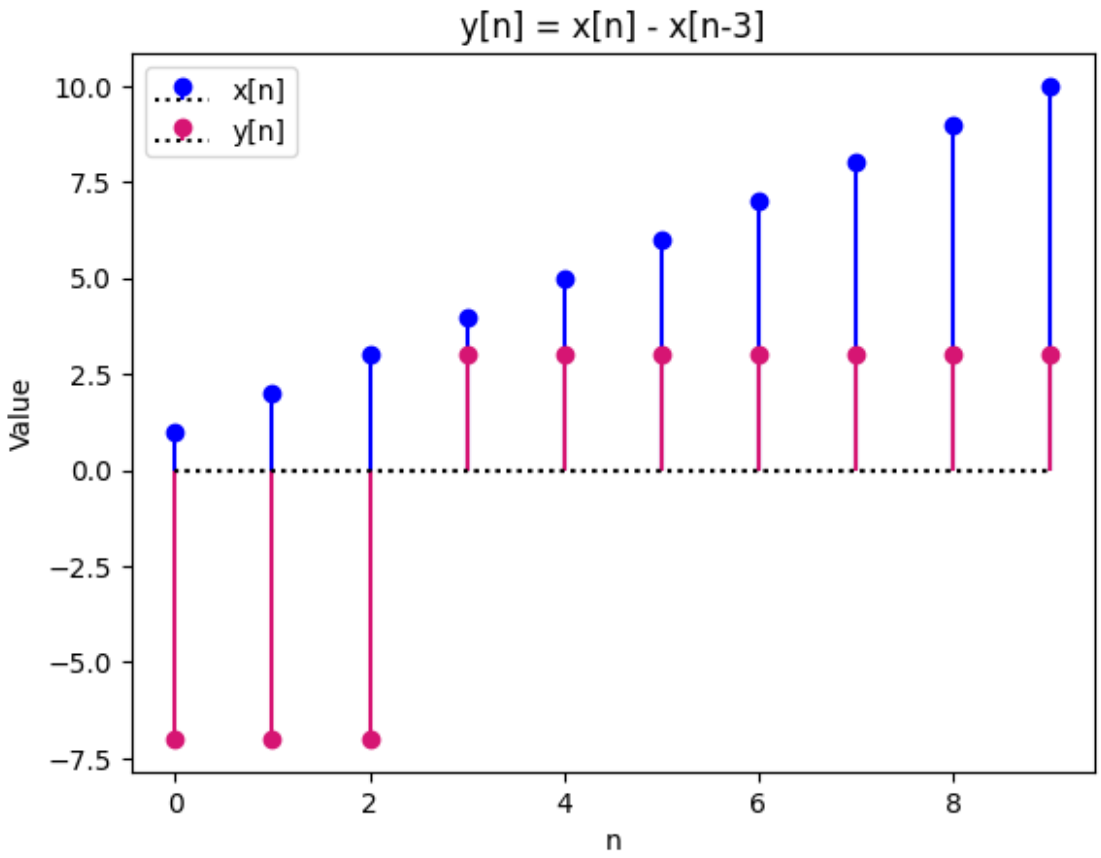
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import unit_impulse

def delta(n, shift=0):
    return unit_impulse(len(n), idx=np.where(n == shift)[0])

# Given function
n_val = np.arange(0, 20)
equation= (delta(n_val) + 2 * delta(n_val - 1) + 3 * delta(n_val - 2)
+
                   delta(n_val - 1) + 2 * delta(n_val - 2) + 3 *
delta(n_val - 3)) / 2

# Computed moving average δ(6n-11)
moving_ave = delta(6*n_val, shift=11)

plt.stem(n_val, equation, label='Given Equation', basefmt='k:')
plt.stem(n_val, moving_ave, markerfmt='#d61573',
linefmt='#d61573',label='Computed Moving Average', basefmt='k:')
plt.xlabel('n')
plt.ylabel('Value')
plt.title('Comparison of Given Equation and Computed Moving Average')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/scipy/signal/
_waveforms.py:666: FutureWarning: elementwise comparison failed;
returning scalar instead, but in the future will perform elementwise
comparison
  elif idx == 'mid':
```



2.5 SIGNAL OPERATIONS 2

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

n_val = np.arange(0, 10)
xn = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

yn = xn - np.roll(xn, shift=3)

plt.stem(n_val, xn, label='x[n]', linefmt='b', markerfmt='bo',
basefmt='k:')
plt.stem(n_val, yn, label='y[n]', linefmt='#d61573',
markerfmt='#d61573', basefmt='k:')
plt.xlabel('n')
plt.ylabel('Value')
plt.title('y[n] = x[n] - x[n-3]')
plt.legend()
plt.show()
```



2.6 SIGNAL OPERATIONS 3

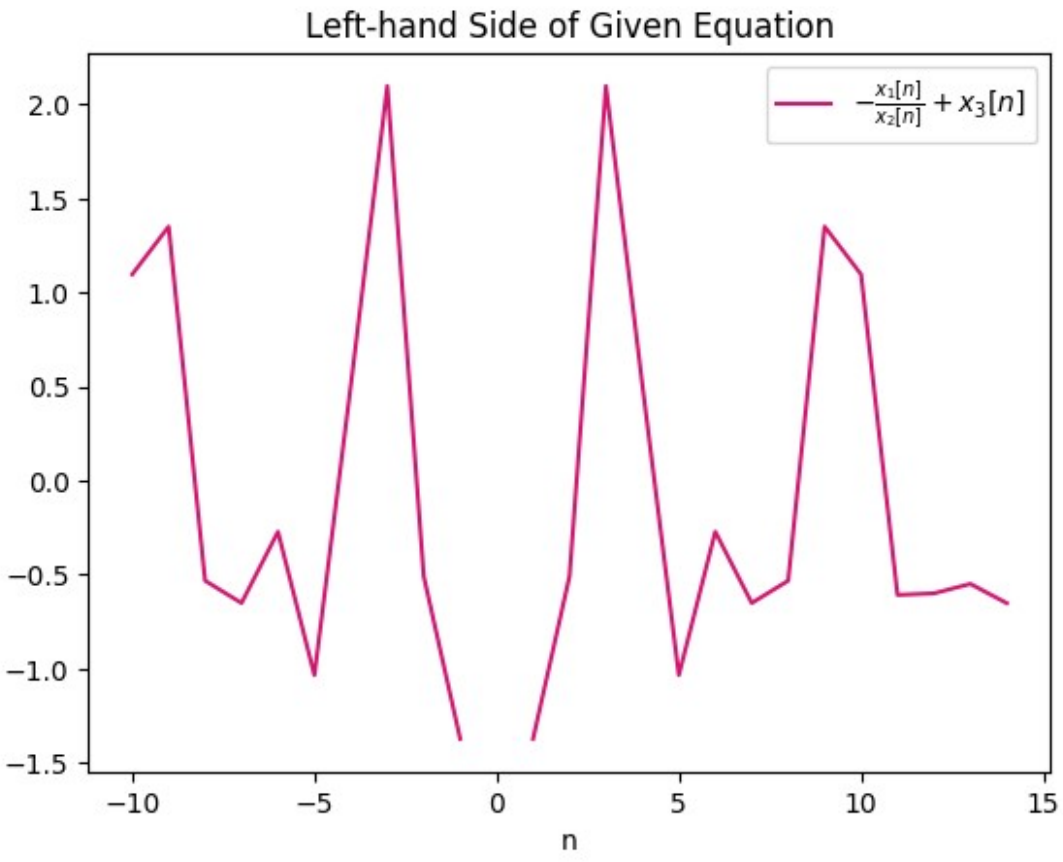
```
import numpy as np

def one_step_matrix():
    one_step_operator = np.array([[1, 0.5, -1],
                                  [-2, 1, 1],
                                  [-1, -1, 0]])
    F = (1 / np.sqrt(2)) * one_step_operator

    return F

#1/sqrt(2) * matrix
F = np.round(one_step_matrix(),5)
print("Matrix F: ", "\n",F)
```

```
Matrix F:
 [[ 0.70711  0.35355 -0.70711]
 [-1.41421  0.70711  0.70711]
 [-0.70711 -0.70711  0.     ]]
```

2.7 FREQUENCY ANALYSIS

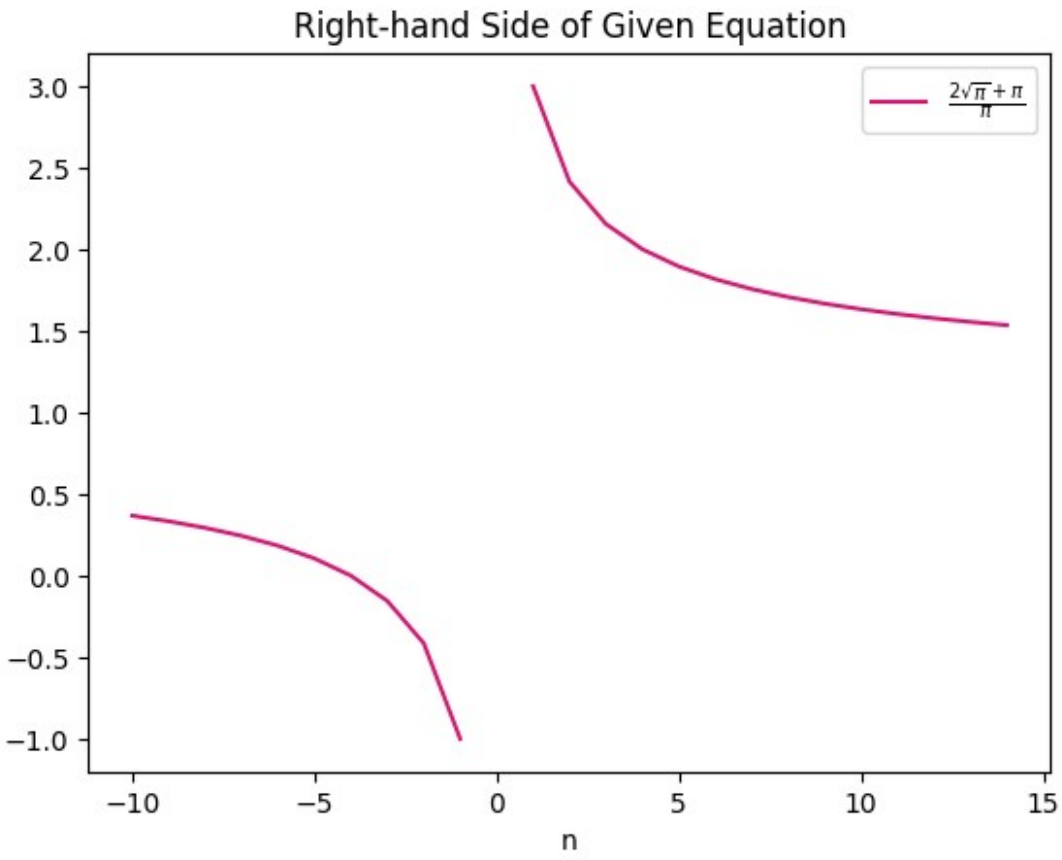
```
import numpy as np
import matplotlib.pyplot as plt
import math

n = 1 #np.arange(-10,15)
pi = np.arange(-10,15)

x1 = np.exp(1j * pi/2 * n)
x2 = np.exp(-1j * pi/2 * n)
x3 = ((2/np.sqrt(abs(pi)))*np.exp(1j * 2*pi * n))

#equations
xn = -(x1/x2) + x3

plt.plot(pi, xn, linestyle='-', label=r'$-\frac{x_1[n]}{x_2[n]} +
x_3[n]$', color='#d61573')
plt.title('Left-hand Side of Given Equation')
plt.xlabel('n')
plt.legend(loc='upper right')
plt.show()
```

```
<ipython-input-158-e9c95f40f6ca>:10: RuntimeWarning: divide by zero
encountered in divide
  x3 = ((2/np.sqrt(abs(pi)))*np.exp(1j * 2*pi * n))
<ipython-input-158-e9c95f40f6ca>:10: RuntimeWarning: invalid value
encountered in multiply
  x3 = ((2/np.sqrt(abs(pi)))*np.exp(1j * 2*pi * n))
```




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

pi = np.arange(-10,15)

ge = (2 * np.sqrt(np.abs(pi)) + pi) / pi

plt.plot(pi, ge.real, linestyle='-', label=r'$\frac{2\sqrt{\pi} + \pi}
{\pi}$', color='#d61573')
plt.title('Right-hand Side of Given Equation')
plt.xlabel('n')
plt.legend(loc='upper right')
plt.show()
```

```
<ipython-input-155-174a546b2eaa>:6: RuntimeWarning: invalid value
encountered in divide
  ge = (2 * np.sqrt(np.abs(pi)) + pi) / pi
```


Right-hand Side of Given Equation
$\frac{2\sqrt{\pi}+\pi}{\pi}$
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−10
−5
0
5
10
15
n