Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Signals and Systems II
Course Number:	ELE632
Semester/Year (e.g.F2016)	W2023

Instructor:	Dimitri Androutsos

Assignment/Lab Number:	Lab 4
Assignment/Lab Title:	Discrete Time Fourier Transform

Submission Date:	March 26, 2023
Due Date:	March 26, 2023

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Saini	Harsanjam	501055402	10	Harsanjam

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

Introduction:

In Lab 4, we will study the discrete-time Fourier transform in this lab (DTFT). We will study the time convolution property of DTFT and learn how to compute the DTFT of a signal using the fast Fourier transform (FFT). A finite impulse response (FIR) high-pass filter will also be created, and the differences between an ideal and realisable filter will be contrasted. We anticipate learning more about DTFT and its uses in signal processing by the end of the lab.

A) Discrete Time Fourier Transform (DTFT)

Python code for Part 1, 2 and 3:

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

N0 = 128
n = np.arange(N0)
ohm = (2*np.pi/128)*np.arange(-64, 64)

x = np.concatenate(([1, 6/7, 5/7, 4/7, 3/7, 2/7, 1/7], np.zeros(121)))
X = np.fft.fft(x)

plt.plot(ohm, np.fft.fftshift(np.abs(X)))
plt.title('Magnitude of X[Ω]')
plt.xlabel('Ω')
plt.ylabel('|X[Ω]|')
plt.show()

plt.plot(ohm, np.fft.fftshift(np.angle(X)))
plt.title('Phase of X[Ω]')
plt.xlabel('Ω')
plt.ylabel('<X[Ω]')
plt.show()

X_hand = 1 + (6/7)*np.exp(-1j*ohm) \
    + (5/7)*np.exp(-1j*2*ohm) \
    + (4/7)*np.exp(-1j*3*ohm) \
    + (3/7)*np.exp(-1j*4*ohm) \
    + (2/7)*np.exp(-1j*5*ohm) \
    + (1/7)*np.exp(-1j*6*ohm)

plt.plot(ohm, np.abs(X_hand))
plt.title('Magnitude of X[Ω]')
plt.xlabel('Ω')
plt.ylabel('|X[Ω]|')
plt.show()

plt.plot(ohm, np.angle(X_hand))
plt.title('Phase of X[Ω]')
plt.xlabel('Ω')
plt.ylabel('<X[Ω]')
plt.show()

x_hand = np.fft.ifft(X_hand)

plt.stem(n, x_hand.real)
plt.title('Plot of x[n]')
plt.xlabel('n')
plt.ylabel('x[n]')
plt.show()
```

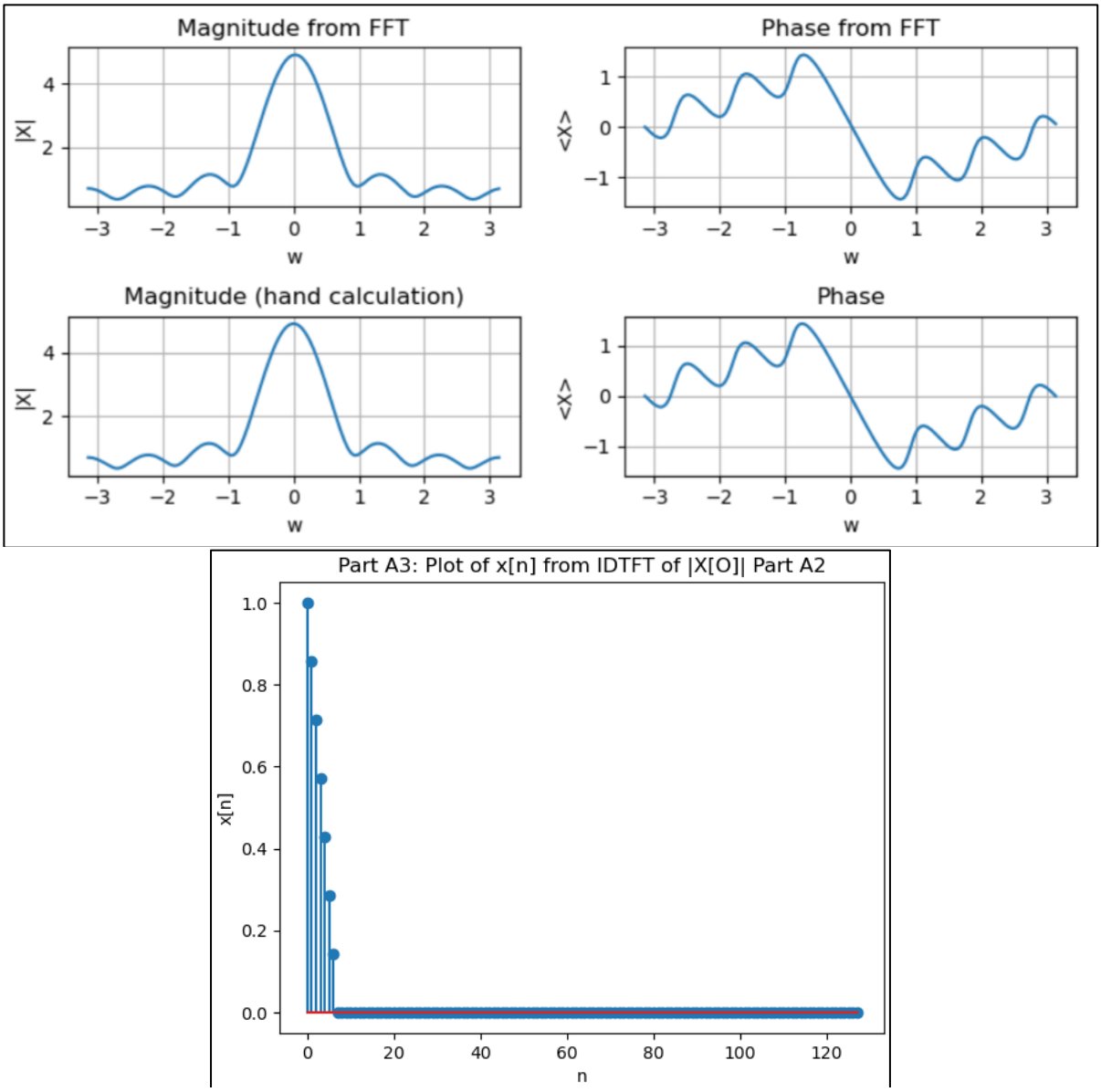


Figure 1: Plot for $x[n]$, the magnitude and phase of X (Ω) including hand calculation.

The Python generated and the hand calculated DTFT are both the same with the slight differences due to approximations in calculation.

DTFT by hand:

DTFT by hand

$$X(\Omega) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\Omega n}$$

$$x(n) = \left(-\frac{1}{7}n + 1\right)\left(u(n) - u(n-6)\right)$$

$$X(\Omega) = \sum_{n=-\infty}^{\infty} \left(-\frac{1}{7}n + 1\right)\left(u(n) - u(n-6)\right) e^{-j\Omega n}$$

$$= \sum_{n=0}^{6} \left(-\frac{1}{7}n + 1\right) e^{-j\Omega(n+6)}$$

$$X(\Omega) = 1 + \frac{6}{7} e^{-j\Omega} + \frac{5}{7} e^{-2j\Omega} + \frac{4}{7} e^{-3j\Omega} + \frac{3}{7} e^{-4j\Omega}$$
$$+ \frac{2}{7} e^{-5j\Omega} + \frac{1}{7} e^{-6j\Omega}$$

B) Time Convolution

1) Python code for Part 1: $x[n] = \sin\left(\frac{2\pi n}{10}\right)\ (u[n] - u[n-10])$

S=4 from my student number. Therefore, to better visualize the differences between the two parts, I chose **S=10**, like the MATLAB report.

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

# Part 1 - DTFT plot of x[n]
n = np.arange(16)
u_c = lambda t: 1.0 * (t >= 0)
u = lambda n: u_c(n) * (np.mod(n, 1) == 0)
x = np.sin(2 * np.pi * n / 10) * (u(n) - u(n - 10))

omega = np.linspace(-np.pi, np.pi, 1001)
W_omega = np.exp(-1j) ** (np.arange(len(x)).reshape(-1, 1) * omega)
X = np.dot(x, W_omega)

plt.stem(n, x)
plt.title('Plot of x[n]')
plt.xlabel('n')
plt.ylabel('x[n]')
plt.show()

plt.plot(omega, abs(X))
plt.title('Magnitude of X[Ω]')
plt.xlabel('Ω')
plt.ylabel('|X[Ω]|')
plt.show()

plt.plot(omega, np.angle(X))
plt.title('Phase of X[Ω]')
plt.xlabel('Ω')
plt.ylabel('<X[Ω]')
plt.show()
```

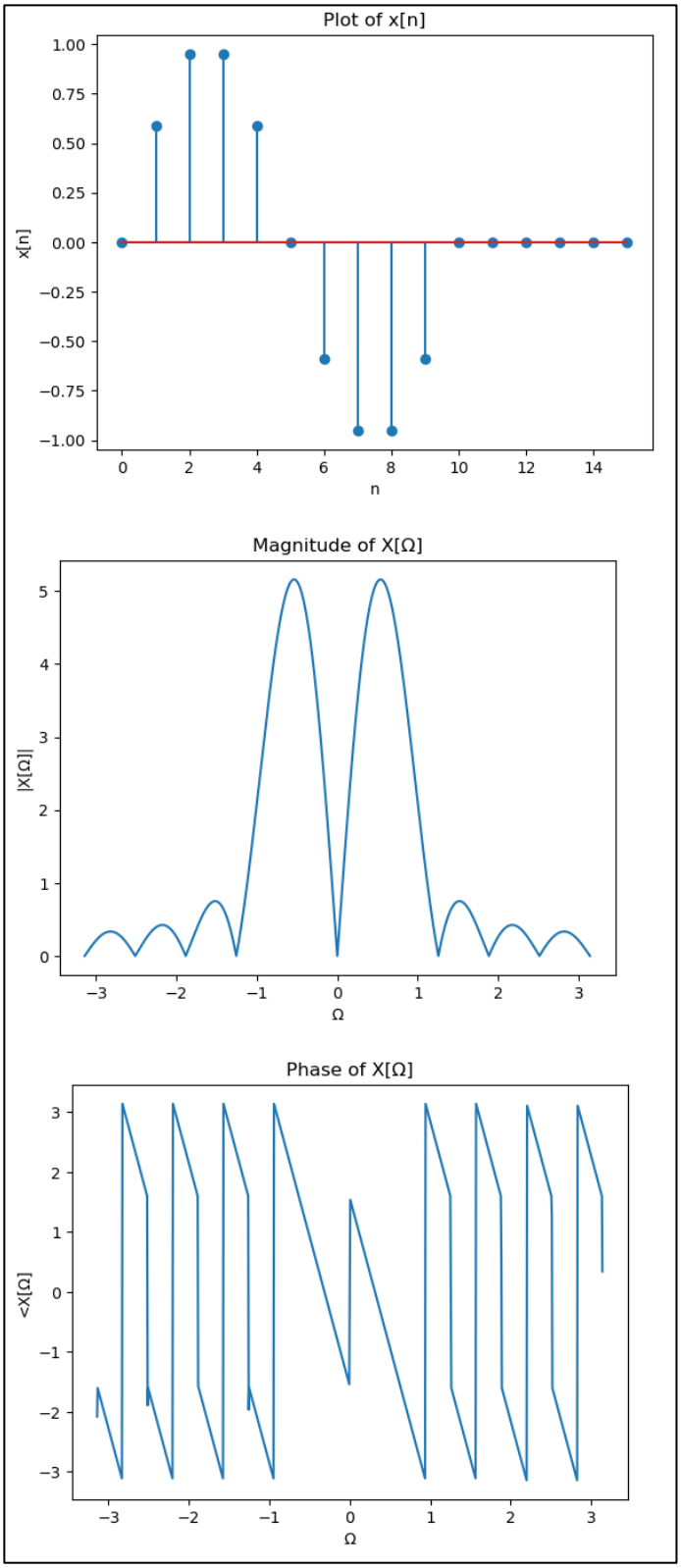


Figure 2: Plot for DTFT of signal $x[n] = \sin\left(\frac{2\pi n}{10}\right)\ (u[n] - u[n-10])$

2) Python code for Part 2:

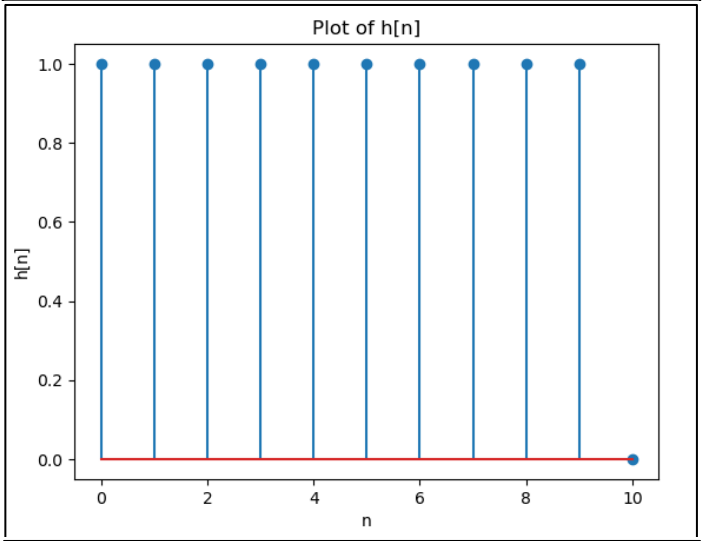
```
# Part 2 - DTFT plot of h[n]
n = np.arange(10)
x = u(n) - u(n - 9)

W_omega = np.exp(-1j) ** (np.arange(len(x)).reshape(-1, 1) * omega)
H = np.dot(x, W_omega)

plt.stem(n, x)
plt.title('Plot of h[n]')
plt.xlabel('n')
plt.ylabel('h[n]')
plt.show()

plt.plot(omega, abs(H))
plt.title('Magnitude of H[Ω]')
plt.xlabel('Ω')
plt.ylabel('|H[Ω]|')
plt.show()

plt.plot(omega, np.angle(H))
plt.title('Phase of H[Ω]')
plt.xlabel('Ω')
plt.ylabel('<H[Ω]')
plt.show()
```



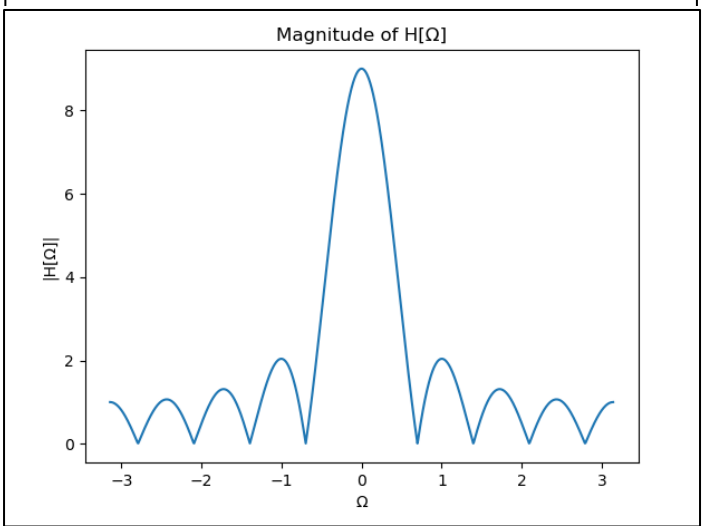


Figure 3: Plot for DTFT of signal h[n]

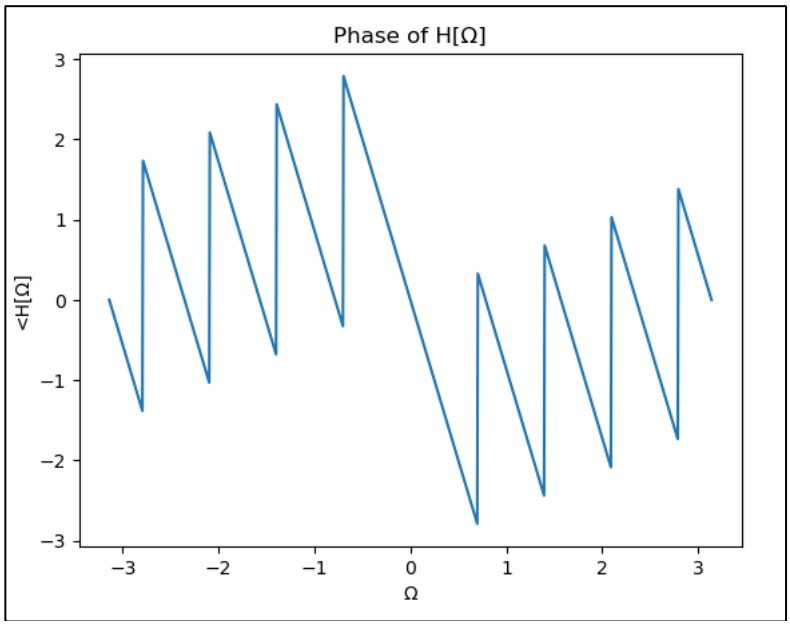


Figure 4: Plot for impulse response h[n] and magnitude H(Ω) of filter with length 71.

3) Python code for Part 3:

```
# Part 3 - Convolution Plot of X[Ω] and H[Ω]
Y = X * H

plt.plot(omega, abs(Y))
plt.title('Magnitude of Y[Ω]')
plt.xlabel('Ω')
plt.ylabel('|Y[Ω]|')
plt.show()

plt.plot(omega, np.angle(Y))
plt.title('Phase of Y[Ω]')
plt.xlabel('Ω')
plt.ylabel('<Y[Ω]')
plt.show()
```

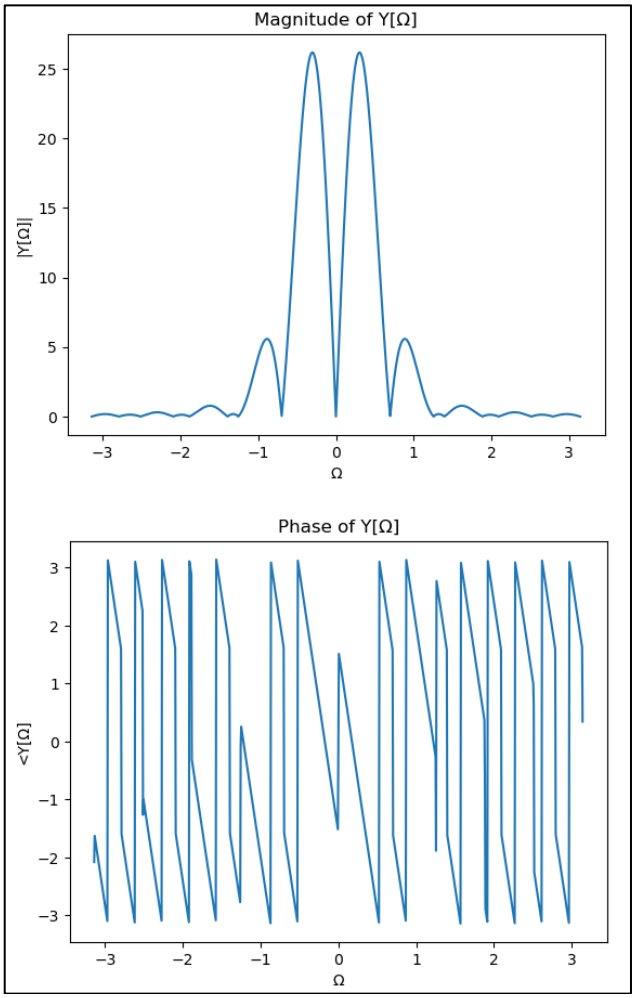


Figure 5: Plot for convolution of X (Ω) and H(Ω)

4) Python code for Part 4:

```
# Part 4 - Convolution Plot of x[n] and h[n] by conv command
import numpy as np
import matplotlib.pyplot as plt

n = np.arange(16)
u_c = lambda t: 1.0*(t>=0)
u = lambda n: u_c(n)*(np.mod(n,1)==0)

h = u(np.arange(10))
x = np.sin(2*np.pi*n/10)*(u(n) - u(n-10))
n = np.arange(25)

y = np.convolve(x, h)

plt.stem(n, y)
plt.title('plot of y[n] using conv command')
plt.xlabel('n')
plt.ylabel('y[n]')
plt.show()
```

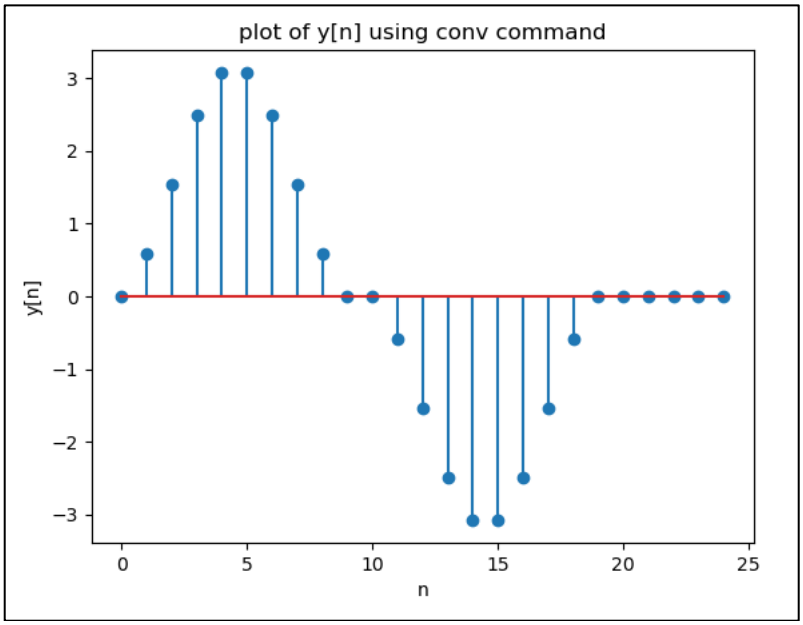


Figure 6: Plot for convolution of y[n] using the ***convolve*** command.

5) Python code for Part 5:

```
# Part 5 - DTFT Plot of y[n] from Part 4

omega = np.linspace(-np.pi,np.pi,1001)
W_omega = np.exp(-1j*np.arange(len(y))[:,None]*omega)
Y = np.dot(y, W_omega)

plt.plot(omega, np.abs(Y))
plt.title('Magnitude of Y[Ω] from part 4')
plt.xlabel('Ω')
plt.ylabel('|Y[Ω]|')
plt.show()

plt.plot(omega, np.angle(Y))
plt.title('Phase of Y[Ω]')
plt.xlabel('Ω')
plt.ylabel('<Y[Ω]>')
plt.show()
```

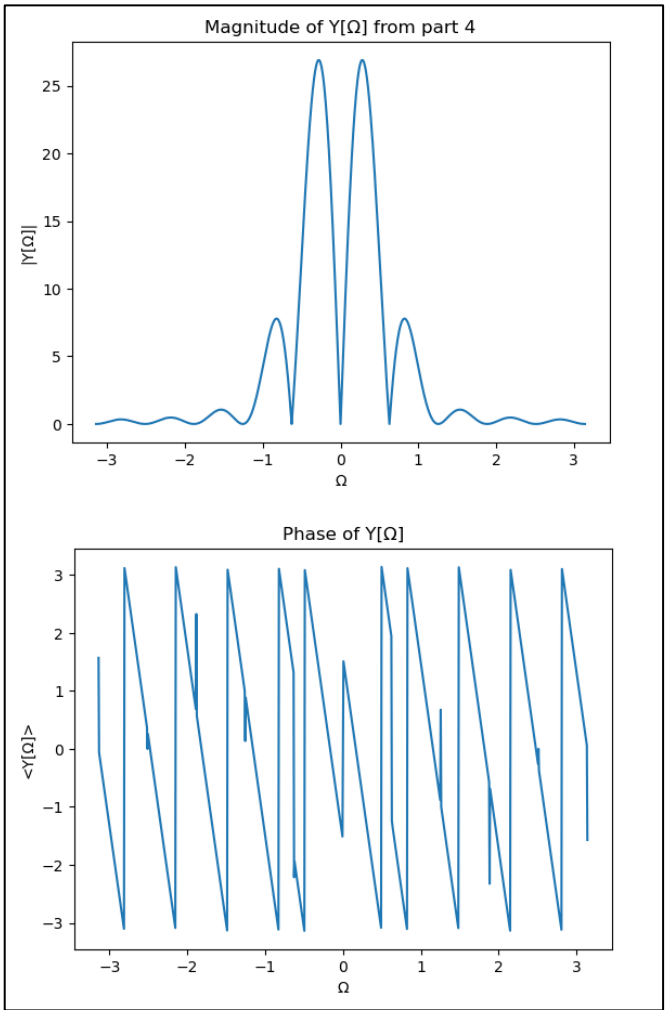


Figure 7: Plot for DTFT of signal y[n]

6) The Fourier transform's feature that the convolution of two functions in the time domain is identical to the product in the frequency domain explains why the results in parts 3 and 5 were the same.

C) FIR Filter Design by Frequency Sampling

1 & 2) Python code for Part 1 & 2: Filter length is 35 points.

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

ohm0 = 2*np.pi/3
N = 35
n = np.arange(N)
Omega = np.linspace(0, 2*np.pi*(1-1/N), N)
H_d = lambda Omega: (np.mod(Omega, 2*np.pi) > ohm0) * (np.mod(Omega, 2*np.pi) < 2*np.pi - ohm0)

H = H_d(Omega) * np.exp(-1j * Omega * ((N-1)/2))
h = np.fft.ifft(H)

plt.plot(Omega, abs(H))
plt.title('Original Highpass Filter Magnitude |Y[Ω]|')
plt.xlabel('Ω')
plt.ylabel('|H[Ω]|')
plt.show()

plt.plot(Omega, np.angle(H))
plt.title('Original Highpass Filter Phase <H[Ω]')
plt.xlabel('Ω')
plt.ylabel('<H[Ω]')
plt.show()

plt.stem(n, h)
plt.title('Impulse Response (h[n]) of Filter H[Ω]')
plt.xlabel('n')
plt.ylabel('h[n]')
plt.show()


w, H = freqz(h, 1, np.linspace(0, 2*np.pi, 1002))
plt.plot(w, abs(H))
plt.title('Highpass Filter Magnitude by freqz command')
plt.xlabel('Ω')
plt.ylabel('|Y[Ω]|')
plt.show()
```

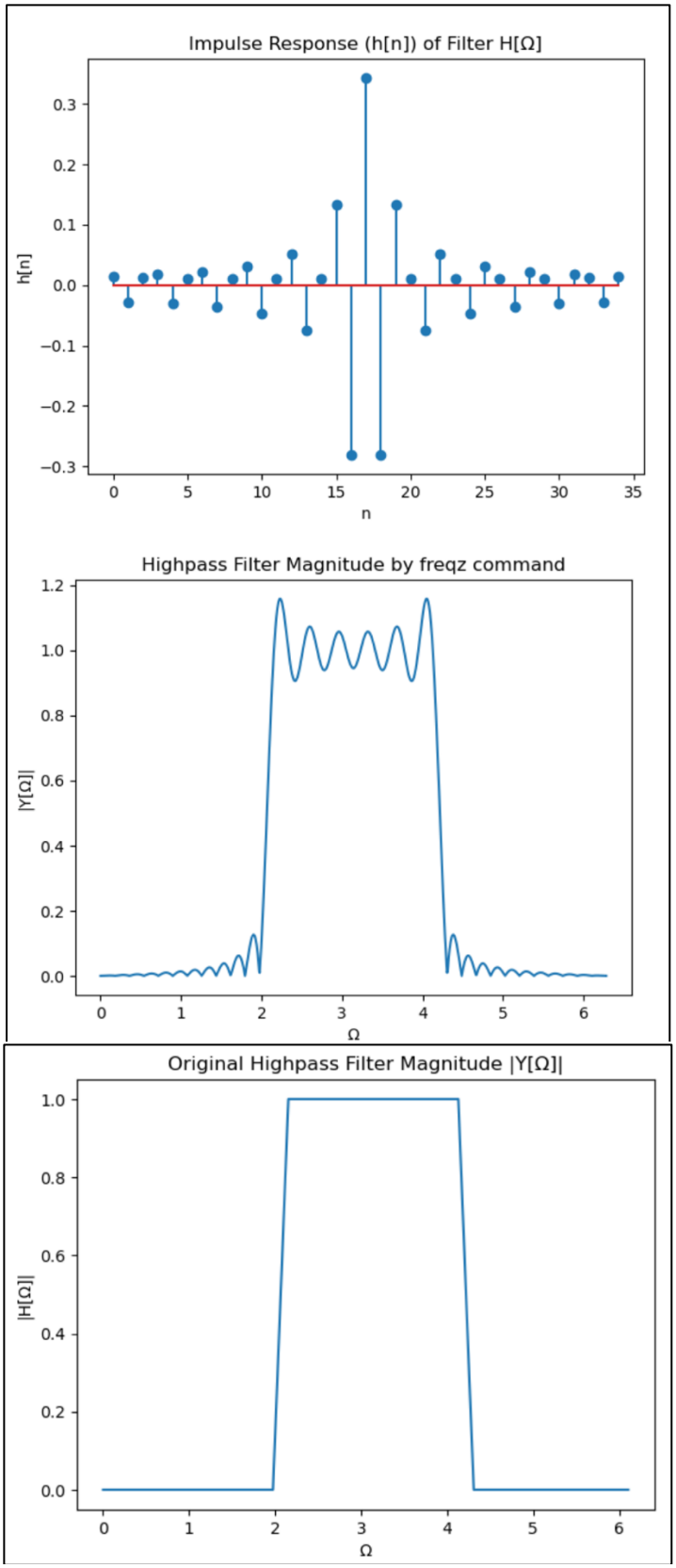


Figure 8: Plot for impulse response h[n] and magnitude H (Ω) of filter with length 35.

3) The optimal filter is almost flawless. This is due to the absence of edge noise (i.e., the bandwidth length). In the meantime, the input signal would be impacted by the conventional filter's edge noise. The first signal is regarded as an "ideal filter" and has extremely strict bounds at precisely 2 pi/3 and 4 pi/3, which is the primary distinction between the two signals. The second signal, however, is a more accurate representation of a filter's signal, with "ripples" continuing past the cut-off points. The second signal simulates a real-world scenario in which we would require "infinite" points to obtain the first signal.

4) Python code for Part 4: Filter Length is 71 points

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

ohm0 = 2*np.pi/3
N = 71
n = np.arange(N)
Omega = np.linspace(0, 2*np.pi*(1-1/N), N)
H_d = lambda Omega: (np.mod(Omega, 2*np.pi) > ohm0) * (np.mod(Omega, 2*np.pi) < 2*np.pi - ohm0)

H = H_d(Omega) * np.exp(-1j * Omega * ((N-1)/2))
h = np.fft.ifft(H)

plt.plot(Omega, abs(H))
plt.title('Original Highpass Filter Magnitude |Y[Ω]|')
plt.xlabel('Ω')
plt.ylabel('|H[Ω]|')
plt.show()

plt.plot(Omega, np.angle(H))
plt.title('Original Highpass Filter Phase <H[Ω]')
plt.xlabel('Ω')
plt.ylabel('<H[Ω]')
plt.show()

plt.stem(n, h)
plt.title('Impulse Response (h[n]) of Filter H[Ω]')
plt.xlabel('n')
plt.ylabel('h[n]')
plt.show()


w, H = freqz(h, 1, np.linspace(0, 2*np.pi, 1002))
plt.plot(w, abs(H))
plt.title('Highpass Filter Magnitude by freqz command')
plt.xlabel('Ω')
plt.ylabel('|Y[Ω]|')
plt.show()
```

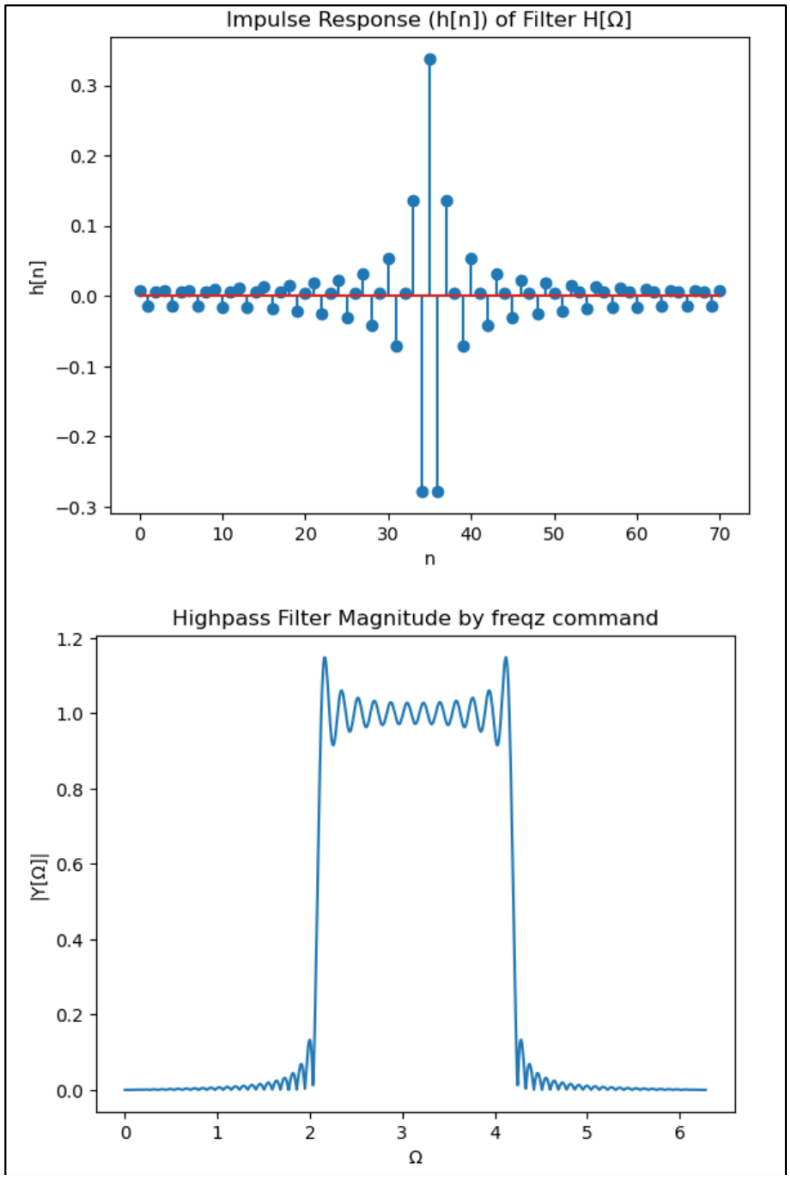


Figure 9: Plot for impulse response h[n] and magnitude H (Ω) of filter with length 71.

5) When the filter length N is increased from 35 to 71, the "resolution" likewise increases because the impulse response is more accurately represented because there are more points and a wider variety of signal representations.