<table>
<tr><td></td><td colspan="2">Marwadi University
Faculty of Engineering & Technology
Department of Information and Communication Technology</td></tr>
<tr><td>Subject: Programming With Python (01CT1309)</td><td colspan="2">Aim: Analysis of LTI System Responses to Standard Inputs Using Python</td></tr>
<tr><td>Experiment No: 19</td><td>Date:</td><td>Enrollment No: 92400133055</td></tr>
</table>

GITHUB

Aim: Analysis of LTI System Responses to Standard Inputs Using Python

IDE: Visual Studio Code

Analyzing Discrete-Time Systems Using Z-Transform

The Z-transform is used for analyzing discrete-time signals and systems. The Z-transform of a discrete-time signal $x[n]$ is given by:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n}$$

where z is a complex variable, $X(z)$ represents the Z-transform of the signal.

Z-Transform Function

For an LTI system, the Z-transform function $H(z)$ is defined as:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}}$$

where $B(z)$ is the numerator polynomial, $A(z)$ is the denominator polynomial.

Stability

A discrete-time system is stable if all poles of its Z-transfer function lie inside the unit circle in the Z-plane. To check stability:

Calculate the poles of $H(z)$

Check if the magnitude of each pole is less than 1.

Causality

A system is causal if its impulse response $h[n]$ is zero $n < 0$. This generally means that the numerator polynomial should not have terms that depend on future values.

Time Invariance

A system is time-invariant if a time shift in the input results in an equivalent time shift in the output. For LTI systems, if the system is defined properly, it is generally assumed to be time-invariant.

Example

$$H(z) = \frac{(z^2 + 0.5)}{(z^2 - 1.5z + 0.5)}$$

Bode Plot Analysis Stability:

Check the gain and phase margins.

Ensure that both margins are positive for stability.

Causality:

Examine the magnitude and phase at low frequencies.

Confirm that the system behaves as a causal system (magnitude starts lower, phase starts near 0 and decreases).

Time Invariance:

If the system is LTI, it is inherently time-invariant.

Marwadi University	Marwadi University Faculty of Engineering & Technology Department of Information and Communication Technology	
Subject: Programming With Python (01CT1309)	**Aim:** Analysis of LTI System Responses to Standard Inputs Using Python	
Experiment No: 19	**Date:**	**Enrollment No: 92400133055**

Analyse the impulse response (if available) to verify consistent responses to delayed inputs.

Python Implementation import
numpy as np import
matplotlib.pyplot as plt
from scipy.signal import TransferFunction, lti

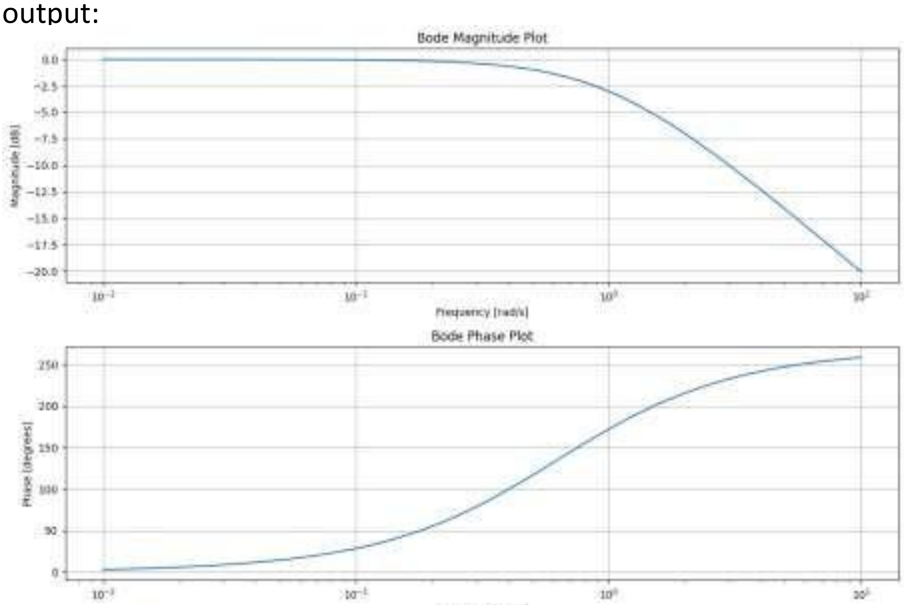
```
def analyze_z_transfer_function(num, den):
# Create a Transfer Function object
    system = TransferFunction(num, den)

    # Get the poles and zeros     zeros = system.zeros
poles = system.poles     print("Zeros:", zeros)
print("Poles:", poles)     # Stability Analysis     stable =
all(np.abs(pole) < 1 for pole in poles)
print("Stability:", "Stable" if stable else "Unstable")

    # Causality Analysis
    causal = all(num[i] == 0 for i in range(len(num) - 1) if num[i + 1] == 0)
print("Causality:", "Causal" if causal else "Non-Causal")
    # Time Invariance Analysis
    time_invariant = True  # For Z-transforms, generally time-invariant if system defined properly
print("Time Invariance:", "Time Invariant" if time_invariant else "Time Variant")
    # Bode plot (magnitude and phase)
w, mag, phase = bode(system)
    # Plot Bode plot
plt.figure(figsize=(12, 8))
plt.subplot(2, 1, 1)
    plt.semilogx(w, mag)    # Bode magnitude plot
    plt.title('Bode Magnitude Plot')
plt.xlabel('Frequency [rad/s]')
    plt.ylabel('Magnitude [dB]')
    plt.grid()
plt.subplot(2, 1, 2)
    plt.semilogx(w, phase)   # Bode phase plot
```

```
    plt.title('Bode Phase Plot')
plt.xlabel('Frequency [rad/s]')    plt.ylabel('Phase
[degrees]')
    plt.grid()
plt.tight_layout()
plt.show()
# Example: Analyzing a specific system H(z) = (z^2 + 0.5)/(z^2 - 1.5z + 0.5)
num = [1, 0.5]  # Numerator coefficients den = [1, -1.5, 0.5]  #
Denominator coefficients
analyze_z_transfer_function(num, den)
```

output:




```
Zeros: [-0.5]
Poles: [1.  0.5]
Stability: Unstable
Causality: Causal
Time Invariance: Time Invariant
```

Transfer Function:

$$H(z) = \frac{0.5}{1 - 0.8z^{-1}}$$

Causality: This system is causal because the denominator has a non-negative exponent (i.e., all powers of z^{-1} are non-negative).

Stability: The system is stable if the poles (the roots of the denominator) lie inside the unit circle. Here, the pole is $z = 0.8$, which is inside the unit circle, so the system is stable.
Time Invariance: The system is time-invariant because the coefficients do not depend on time.
Transfer function:

$$H(z) = \frac{1 - z^{-1}}{1 - 0.5z^{-1}}$$

Causality: This system is causal.

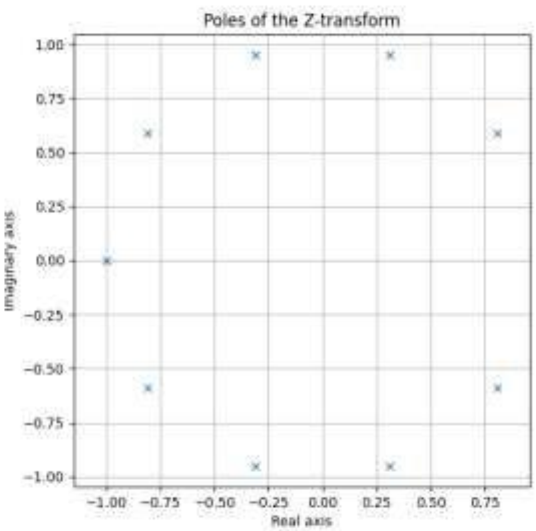
Stability: The pole at $z = 0.5$ is inside the unit circle, making the system stable. Time Invariance: The system is time-invariant

Post Lab Exercise:

 Write a Python function to compute the Z-transform of an unit step function. verify whether the system is stable or unstable. Code:

```
import sympy as sp import numpy as np import
matplotlib.pyplot as plt n = 10 z_transform =
[1] * n poles = np.roots(z_transform)
is_stable = True for pole in poles:     if
abs(pole) >= 1:          is_stable = False
break print("Is the system stable?",
is_stable) plt.figure(figsize=(6, 6))
plt.plot(np.real(poles), np.imag(poles), 'x')
plt.xlabel('Real axis') plt.ylabel('Imaginary
axis') plt.title('Poles of the Z-transform')
plt.grid(True) plt.axis('equal')
plt.show()
```

Output:



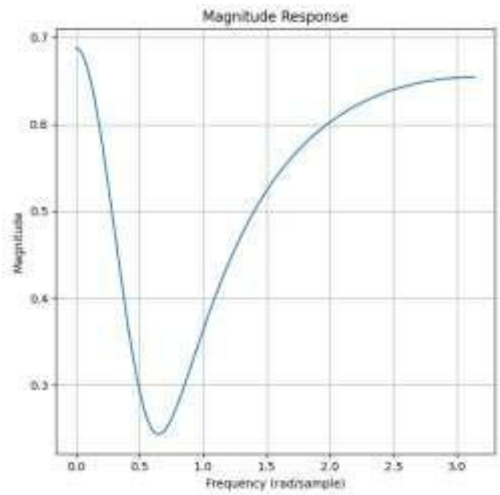
- Implement this for the system $H(z) = \frac{0.5(z-0.7)(z-0.9)}{(z-0.6)(z-0.4)}$ and verify whether the system is stable or unstable.

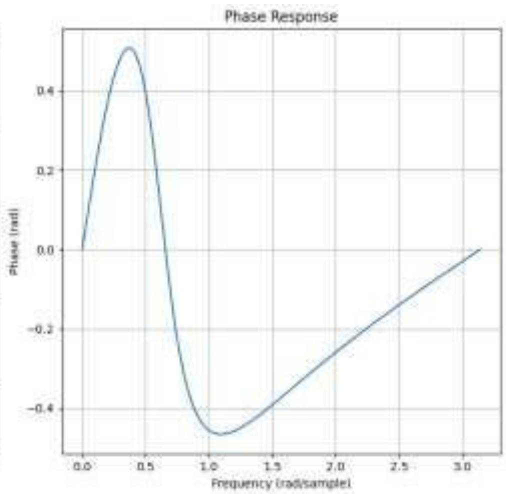
Code:

```
import numpy as np import
matplotlib.pyplot as plt
 num_coeffs = [0.5, -0.65,
0.315] den_coeffs = [1, -1,
0.24] def H_z(z):
    num = np.polyval(num_coeffs[::-1],
z)     den = np.polyval(den_coeffs[::-
1], z)     return num / den def
find_roots(coeffs):
    return np.roots(coeffs) poles =
find_roots(den_coeffs) print("Poles:", poles) stable
= all(np.abs(pole) < 1 for pole in poles)
print("System is", "stable" if stable else
"unstable") w = np.linspace(0, np.pi, 1000) z =
np.exp(1j * w) H = H_z(z) plt.figure(figsize=(12, 6))
plt.subplot(121) plt.plot(w, np.abs(H))
plt.xlabel('Frequency (rad/sample)')
```

```
plt.ylabel('Magnitude')
plt.title('Magnitude Response')
plt.grid() plt.subplot(122)
plt.plot(w, np.angle(H))
plt.xlabel('Frequency
(rad/sample)') plt.ylabel('Phase
(rad)') plt.title('Phase Response')
plt.grid() plt.tight_layout()
plt.show()
```

Output:






```
Poles: [0.6 0.4]
System is stable
```