Marwadi University Marwadi Chandarana Group NAAC A+	**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: 92400133037**

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

IDE:

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:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0+b_1z^{-1}+\cdots+b_mz^{-m}}{a_0+a_1z^{-1}+\cdots+a_nz^{-n}}$$

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

Marwadi University Marwadi Chandarana Group NAAC A+	**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: 92400133037**

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

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.
- 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)
```

Marwadi University Marwadi Chandarana Group NAAC A+	**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: 92400133037**

```
    # 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
```

Marwadi University Marwadi Chandarana Group NAAC A+	**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: 92400133037**

```
    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)
```

Marwadi University Marwadi Chandarana Group NAAC A+	**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: 92400133037**

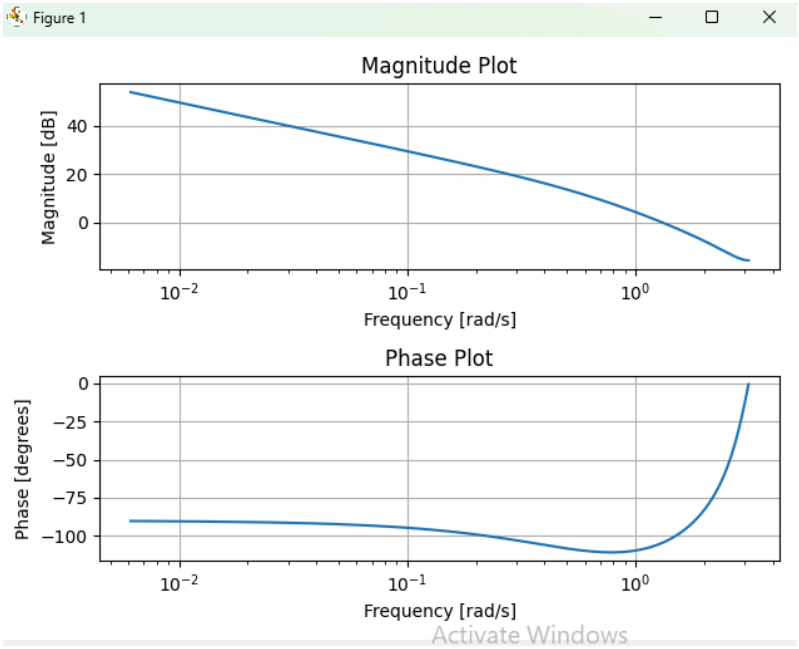
lab19 > example1.py > analyze_z_transfer_function

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

def analyze_z_transfer_function(num,den):
    system=dlti(num,den)
    zeros=system.zeros
    poles=system.poles
    print("Zeros:", zeros)
    print("Poles:", poles)

    stable=all(np.abs(p)<1 for p in poles)
    print("Stability:", "Stable" if stable else "Unstable")

    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_invariant=True
    print("Time Invariance:","Time Invariant" if time_invariant else "Time Variant")

    w, h = freqz(num,den)
    plt.figure()
    plt.subplot(2,1,1)
    plt.semilogx(w,20*np.log10(abs(h)))
    plt.title('Magnitude Plot')
    plt.xlabel('Frequency [rad/s]')
    plt.ylabel('Magnitude [dB]')
    plt.grid()

    plt.subplot(2,1,2)
    plt.semilogx(w,np.angle(h, deg=True))
    plt.title('Phase Plot')
```

lab19 > example1.py > analyze_z_transfer_function

```python
def analyze_z_transfer_function(num,den):
    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]
den = [1, -1.5, 0.5]
analyze_z_transfer_function(num, den)
```

Marwadi University Marwadi Chandarana Group NAAC A+	**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: 92400133037**



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

Marwadi University Marwadi Chandarana Group NAAC A+	**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: 92400133037**

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.
- 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.

```
lab19 > postLab19.py > ...
  1  import sympy as sp
  2
  3  def z_transform_unit_step():
  4      n,z=sp.symbols('n z')
  5      u = 1
  6
  7      U=sp.summation(u*z**(-n),(n,0,sp.oo))
  8      return sp.simplify(U)
  9
 10  U_z=z_transform_unit_step()
 11  print("Z-transform of unit step u[n]:")
 12  sp.pprint(U_z)
 13
 14  roc = "|z|>1"
 15  print("\nRegion of Convergence (ROC):", roc)
 16
 17  if roc=="|z|>1":
 18      print("Stability: Unstable (unit circle not included in ROC)")
 19  else:
 20      print("Stability: Stable")
```

```
PS G:\sem-3\python_lab> python -u "g:\sem-3\python_lab\lab19\postLab19.py"
Z-transform of unit step u[n]:
⎛    z
⎜  ─────    for z > 1 ∨ z < -1
⎜  z - 1
⎜
⎜  ∞
⎜  ___
⎜  ╲
⎜  ╲    -n
⎝  ╱
   ‾‾‾
   n = 0

Region of Convergence (ROC): |z| > 1
Stability: Unstable (unit circle not included in ROC)
```

Marwadi University Marwadi Chandarana Group NAAC A+	**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: 92400133037**

```python
import numpy as np
from scipy.signal import dlti

def stability(num,den):
    system=dlti(num,den)
    zeros=system.zeros
    poles=system.poles
    print("Zeros:", zeros)
    print("Poles:", poles)

    stable=all(np.abs(p)<1 for p in poles)
    print("Stability:", "Stable" if stable else "Unstable")

num=[0.5,-0.7,-0.9,-0.6,-0.4]
den=[1]
stability(num, den)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
TERMINAL
PS G:\sem-3\python_lab> python -u "g:\sem-3\python_lab\lab19\postLab2.py"
Zeros: [ 2.41041804+0.j         -0.75213614+0.j         -0.12914095+0.65160519j
 -0.12914095-0.65160519j]
Poles: []
Stability: Stable
```

GITHUB LINK:
https://github.com/Heer972005/Python_Lab