Implementation of Z-Transform and Inverse Z-Transform

D

Objectives

1. To understand the Z-Transform and its application in analyzing discrete-time signals and systems.
2. To explore the region of convergence (ROC) and its role in system stability.
3. To apply and interpret the inverse Z-Transform to recover time-domain signals from their frequency-domain representations.

Apparatus

Software: MATLAB R2022b

Hardware: Personal Computer

Theory

In digital signal processing (DSP), the Z-Transform is a powerful mathematical tool that transforms discrete time-domain signals into complex frequency-domain representations, similar to the Laplace Transform used for continuous-time systems. While the Laplace Transform typically applies to differential equations in continuous systems, the Z-Transform is used for difference equations that describe discrete-time systems. The general form of a linear constant-coefficient difference equation for such systems is represented by,

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k)$$

where $y(n)$ denotes the output signal at time n, and $x(n)$ represents the input signal. Here, a_k and b_k are constant coefficients that define the system's characteristics, while N and M denote the maximum delays in output and input terms, respectively.

In DSP Z-Transform can be expressed as either a bilateral (two-sided) or a unilateral (one-sided) form, each suited for different applications.

- Bilateral Z-Transform

Typically, bilateral Z-Transform is used in analyzing signals that extend over all discrete time (from $-\infty$ to $+\infty$) and is defined for a discrete-time signal $x[n]$ as,

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

Here, z is a complex variable that can be expressed as $z = re^{j\omega}$, where r is the magnitude and ω the phase. This form is particularly useful for systems where signals exist over the entire discrete time range.

- Unilateral Z-Transform

In cases where $x[n]$ is defined only for $n \geq 0$, the unilateral Z-Transform is used. Generally, it is defined as,

$$X(z) = \sum_{n=0}^{+\infty} x[n]z^{-n}$$

The unilateral form is especially helpful in solving initial-value problems for discrete systems, as it excludes terms before $n = 0$.

In MATLAB, the `syms` function is used to define symbolic variables, allowing for algebraic manipulation of mathematical expressions. The `ztrans` function computes the Z-Transform of a discrete-time signal, either in its bilateral or unilateral form, depending on the signal's definition.

- Region of Convergence (ROC)

The ROC of a Z-Transform is the set of values in the z-plane for which the sum defining $X(z)$ converges. The ROC is essential as it determines the existence and stability of the transform. In the context of discrete-time Fourier analysis, the Z-Transform of a sequence $x[n]$ is essentially a discrete-time Fourier Transform (DTFT) weighted by an exponential factor. The ROC, influenced by this weighting, is defined by the range of values for r for which the transform converges.

For Linear Time-Invariant (LTI) systems, the Z-Transform is represented as a ratio of two polynomials in z^{-1}, commonly written as,

$$x(z) = \frac{P(z)}{D(z)}$$

where $P(z)$ and $D(z)$ are the polynomials of z^{-1}. In this representation, the roots of the numerator $P(z)$ are called zeros, and the roots of the denominator $D(z)$ are known as poles. The placement of poles and zeros in the z-plane is critical in determining the system's response characteristics. The determination of system stability involves analyzing the locations of these poles and zeros relative to the unit circle in the z-plane and confirming that the ROC includes the unit circle. Table 1 presents commonly used Z-transform pairs and their Regions of Convergence.

- Pole-Zero plot and Stability Analysis

A pole-zero plot provides a graphical representation of the poles and zeros of $x(z)$ in the z-plane. In discrete-time systems, stability depends on the positions of the poles relative to the unit circle (i.e., |z|=1) in the z-plane. For a system to be stable, all poles must be located strictly inside the unit circle. If the poles lie exactly on the unit circle without exceeding it, the system is considered marginally stable. However, if any pole falls outside the unit circle, the system is classified as unstable.

In MATLAB, the function `zplane` is commonly used to generate these plots. Poles are marked with '×' and zeros with 'o'.

Table 1. Common z-transform pairs.

Signal	Transform	ROC
$\delta[n]$	1	$\lvert z\rvert > 0$
au[n]	$\frac{az}{z-1}$	$\lvert z\rvert > 1$
u[-n-1]	$\frac{z}{z-1}$	$\lvert z\rvert < 1$
$a^n u[n]$	$\frac{z}{z-a}$	$\lvert z\rvert >$ a
$-a^n u[-n-1]$	$\frac{z}{z-a}$	$\lvert z\rvert <$ a
$na^n u[n]$	$\frac{az}{(z-a)^2}$	$\lvert z\rvert <$ a
$\cos(an)u[n]$	$\frac{z[z-[\cos(a)]}{z^2-2z[cosa]+1}$	$\lvert z\rvert > 1$
$\sin(an)u[n]$	$\frac{z\sin(a)}{z^2-2z[\cos(a)]+1}$	$\lvert z\rvert > 1$

- Inverse Z-Transform

The inverse Z-Transform recovers the original time-domain signal $x[n]$ from its Z-Transform $x(z)$. Mathematically, it is defined by the contour integral,

$$x[n] = \frac{1}{2\pi j}\oint_C X(z)z^{n-1}dz$$

where C is a closed contour encircling the ROC in a counterclockwise direction. This approach allows us to translate back from the frequency domain to the discrete time-domain representation, enabling the analysis of the time-domain signal's characteristics directly from its Z-Transform.

In MATLAB, the inverse Z-Transform can be computed using the `inztrans` function, which transforms a given Z-domain expression back into its corresponding discrete-time signal in the time domain by first defining the signal symbolically with `syms` function

MATLAB code Overview

1. Define the System Coefficients

   Discrete-time systems can be represented in the general form of a linear constant-coefficient difference equation. Consider a digital system with a numerator coefficient numerator coefficient b = [0.2] and denominator coefficients a = [1, -0.52, 0.68].

   MATLAB code for defining the System Coefficient:

```
% Define system coefficients
b = [0.2];            % Numerator coefficients
a = [1, -0.52, 0.68]; % Denominator coefficients
```

2. Transfer Function H(z)

   The transfer function H(z) describes the input-output relationship of the system. In MATLAB tf function can be used to define H(z) for a discrete-time system.

   MATLAB code for defining and displaying the Transfer Function is,

```
H = tf(b, a, -1); % -1 specifies a discrete-time
transfer function disp('Transfer Function H(z):');
H
```

3. Pole-Zero Plot of H(z)

   The Poles and zeros are fundamental characteristics that provide insights into the stability and frequency response of H(z). A pole-zero plot helps visualize these aspects on the Z-plane.

   MATLAB code for Pole-Zero Plot is,

```
figure;
zplane(b, a);
title('Pole-Zero Plot of H(z)');
grid on;
```

4. Impulse and Step Responses

   The impulse and step responses are essential in understanding how a system responds to basic input signals.

MATLAB code for analyzing Impulse and Step Responses is,

```
% Impulse response
impz(b, a);

% Step response
stepz(b, a);
```

5. Frequency Response Analysis

   The frequency response describes how H(z) behaves across different frequencies. Magnitude response shows the system's gain at each frequency. Meanwhile, phase response reveals the phase shift introduced at each frequency.

   MATLAB code for analyzing Frequency Response is,

```
% Frequency Response with Z-domain Analysis
w = linspace(0, pi, 500);
[h, w] = freqz(b, a, w);
magH = abs(h);                   % Magnitude response
phaH = angle(h) * 180/pi;        % Phase response in degrees
```

6. Stability Analysis

   Stability is determined by the location of poles. If all poles lie inside the unit circle, the system is stable. This check is crucial for practical applications of the filter.

   MATLAB code for determining and display poles is,

```
% Determine and display poles
poles = roots(a);
disp('Poles of H(z):');
disp(poles);
```

Procedure

1. Open MATLAB and create a new script by navigating to **Home → New → Script**.

2. Implement the MATLAB code in the script to create a system with numerator coefficient b = [0.2] and denominator coefficients a = [1, -0.52, 0.68].

3. Use MATLAB commands to create and display the transfer function in the command window, representing the system in the Z-domain.

4. Visualize the poles and zeros of the transfer function on the z-plane, as shown in Figure 1.

5. Determine the system's response to a step input and an impulse input. Plot these responses with appropriately labeled axes, as shown in Figure 2 and Figure 3, respectively.

6. Define a frequency range from 0 to π radians to examine the system's response. Use MATLAB to calculate and plot the magnitude and phase responses over this range, as shown in Figure 4.

7. Analyze and display the system's stability in the command window by analyzing the locations of its poles relative to the unit circle.

8. Save the script with an appropriate name (e.g., system_response_analysis.m).

9. Save the figures in JPEG format by selecting **File → Save As** in the Figure window.

Results

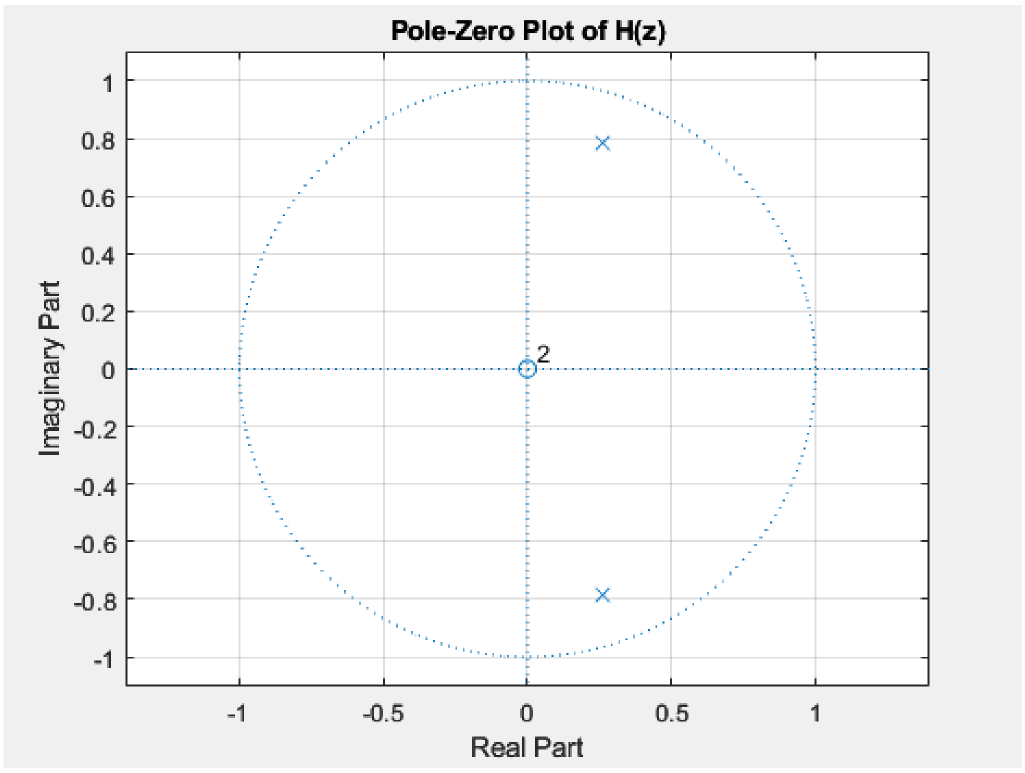

Figure 1. Poles and zeros of the system.

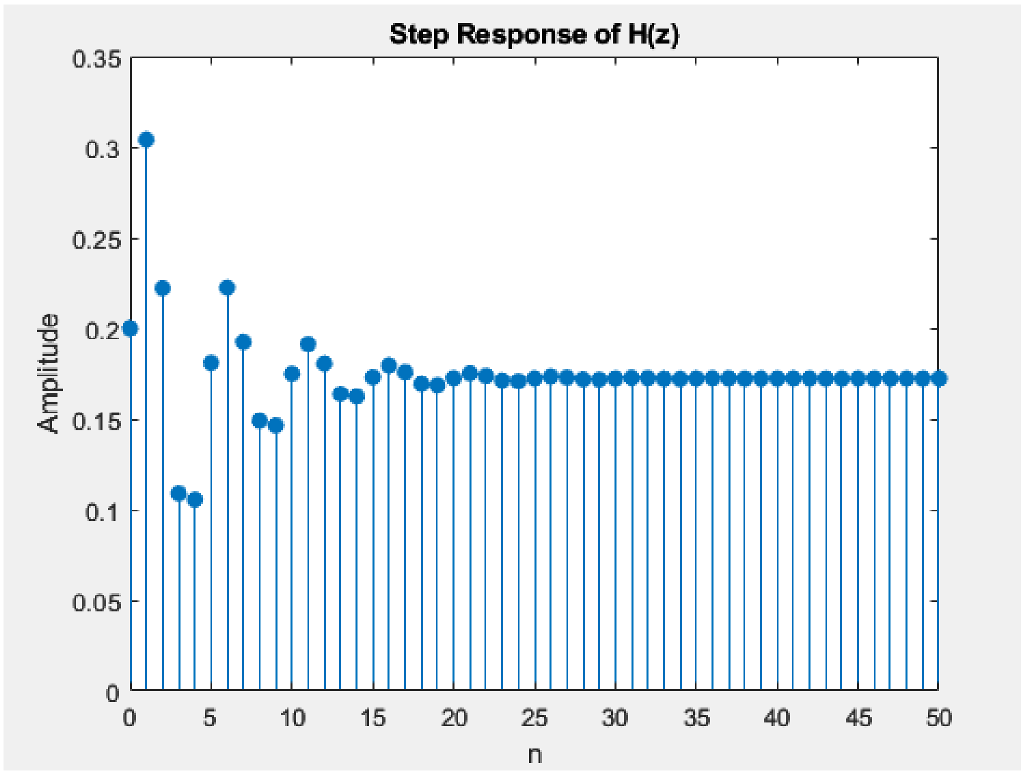

Figure 2. System response to a step input.

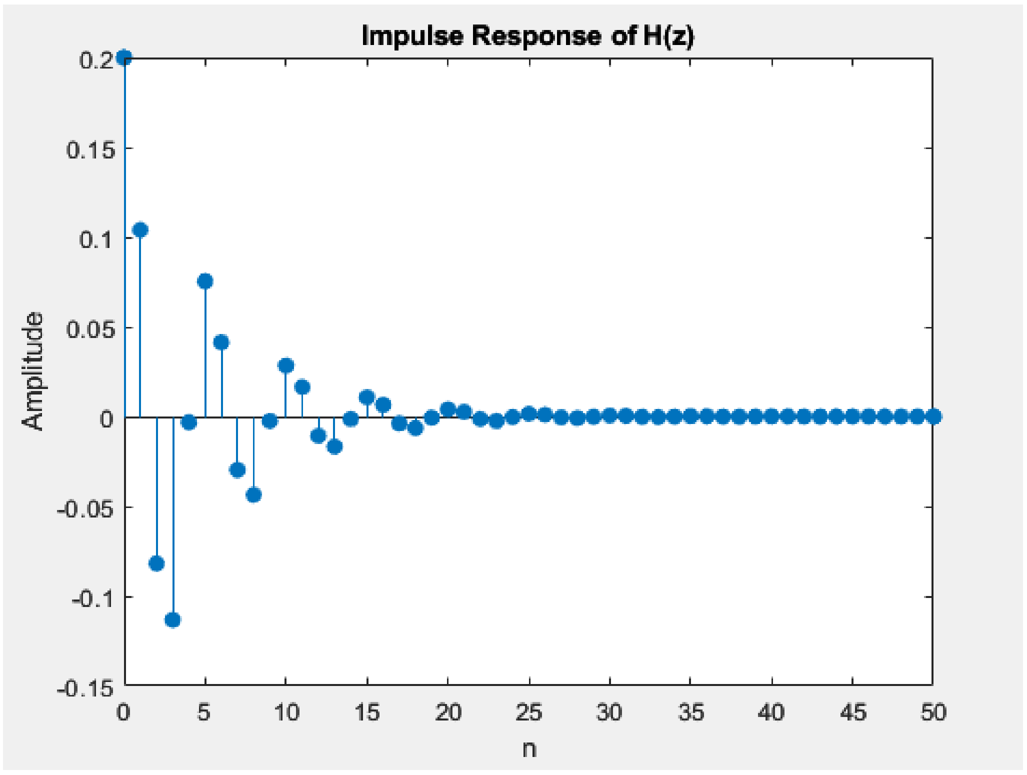

Figure 3. System response to an impulse input.

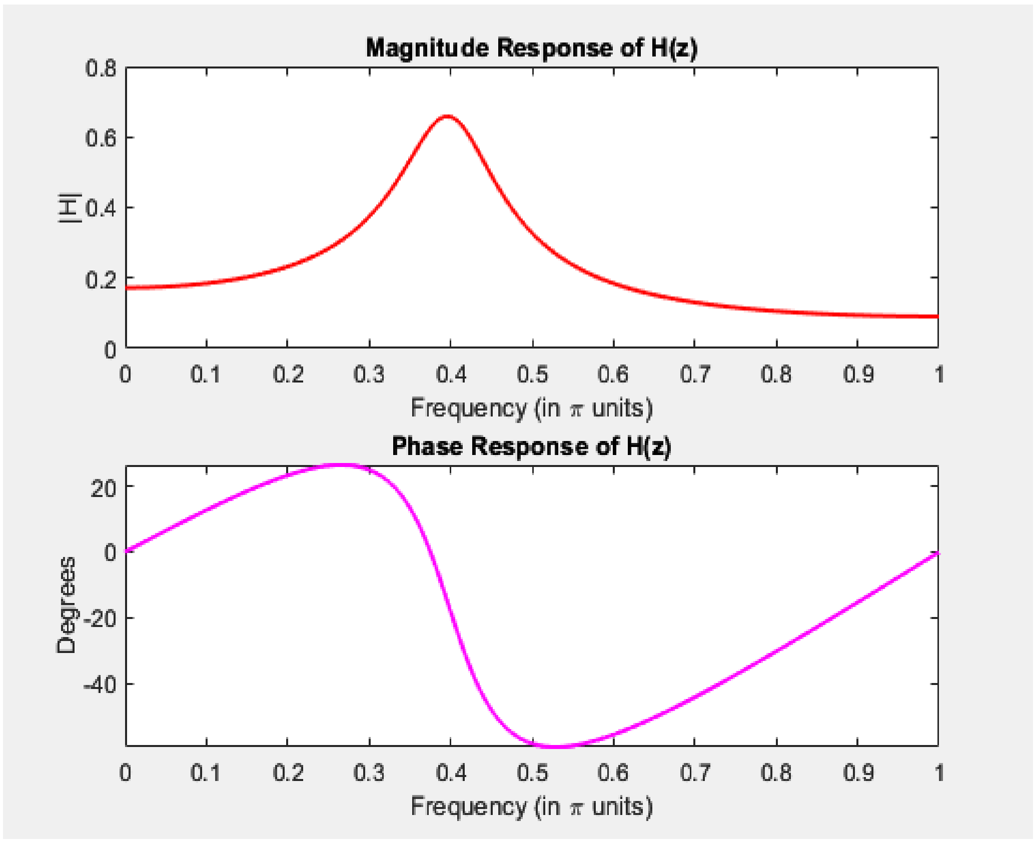

Figure 4. Magnitude and phase responses of the system.

Exercise

1. Consider a discrete-time system represented by the following difference equation

$$y(n) = 0.5\, y(n-1) - 0.2\, y(n-2) + x(n) - 0.6\, x(n-1)$$

Write a MATLAB program to perform the following tasks.

(a) Plot the poles and zeros of the transfer function H(z) in the complex plane using the the `zplane` function.

(b) Plot the magnitude and phase response of the system over a range $0 \le \omega \le \pi$ of frequencies using `freqz` function. Depict the magnitude response in a dash magenta color line while phase response in a solid blue line.

(c) Generate a discrete-time sinusoidal input signal $x(n) = \sin(0.1\pi n)$ for $n = 0,1,\dots,49$. Use the `filter` function to compute the output $y(n)$ of the system in response to this input signal. Plot both the input signal $x(n)$ and the output signal $y(n)$ on the same graph to observe the system's behavior. Use a solid green line to represent output signal and a dash blue line to represent input signal.

Note - Label all graphs appropriately, and add grid lines to all subplots for better visualization. Include legends in relevant plots for clarity.

2. Consider the following rational z-transform G(z) defined by,

$$G(z) = \frac{4 + 3z^{-1} + 2z^{-2} + 0.5\,z^{-3} + 1.5\,z^{-4}}{1 + 6z^{-1} + 8z^{-2} + 2z^{-3} + 0.6z^{-4}}$$

Write a MATLAB program to perform the following tasks.

(a) Compute the partial-fraction expansion of $G(z)$. Use MATLAB's `residue` function to determine the residues, poles, and any direct terms of the transfer function, and display these values in the command window.

(b) Plot the pole-zero plot of $G(z)$. Factorize the transfer function and use MATLAB's `zplane` function to plot the poles and zeros in the complex plane.

(c) Based on the pole-zero plot, determine and display in the command window whether the system is stable.

(d) Plot the impulse response of $G(z)$ using MATLAB's `impz` function for a time range from $n = 0$ to $n = 20$.

3. Consider the following rational z-transform H(z) defined by,

$$H(z) = \frac{z^2 + 2\,z + 3}{z^2 - 1.2\,z + 0.8}$$

Write a MATLAB program to perform the following tasks.

(a) Compute the inverse z-transform of H(z) using the `iztrans` function and display the result in the command window.

(b) Factorize H(z) and use MATLAB's `zplane` function to plot the poles and zeros in the complex plane. Add relevant titles and labels to the plot.

(c) Use the `residuez` function to determine the residues, poles, and any direct terms for H(z). Display these values in the command window.

(d) Based on the pole-zero plot, comment on the stability of the system.

4. (a) Write a MATLAB program to compute the Z-Transform of the discrete-time signal $x[n] = -(0.3)^n$ for $0 \geq n \geq 20$. Display the result of the Z-Transform in the command window. In the complex plane, plot the unit circle with a magenta dash line and a line thickness of 1.5, for the full angular range from 0 to 2π.

(b) Mark the Region of Convergence (see Table 1 for reference) on the same plot for the discrete-time signal and fill the region in yellow. Add legends to the graph, label the axes, and include grid lines for clarity.

(c) Provide an explanation of the ROC and discuss its importance in assessing the stability of the signal.