

Name: M.OMAR

Class: BCE-6

Enrollment No: 01-132182-024

**Experiment No. 6**

**Pole and Zero plot in Z domain using MATLAB**

**Objectives:**

Objective of this lab is to understand following.

In this lab we learned how to find Pole and Zero plot in Z domain using poly command. We have

Observed the output. All have done using MATLAB.

## Introduction:

**NOTE: Answered here about following questions**:

* When/Why Z and inverse Z transform and what are matlab commands for its?

## This Frequent Engineering Question gives a quick overview of an important mathematical technique used in digital signal processing, calculating the z-transform.

With the z-transform, we can create transfer functions for digital filters, and we can plot poles and zeros on a complex plane for [stability analysis](https://www.allaboutcircuits.com/technical-articles/the-right-half-plane-zero-and-its-effect-on-stability/). The inverse z-transform allows us to convert a z-domain transfer function into a difference equation that can be implemented in code written for a microcontroller or digital signal processor.

### How to Calculate the z-Transform

The relationship between a discrete-time signal x[n] and its one-sided z-transform X(z) is expressed as follows:

X(z)=∞∑n=0x[n]z−nX(z)=∑n=0∞x[n]z−n

This summation begins as a sequence of individual values, and since we are summing from n = 0 to n = infinity, the sequence is of infinite length. What can we do with an infinite sequence of summed elements?

This is where convergence comes in.

#### Convergence with the z-Transform

Consider the unit step, which we define as follows:

u[n]={0n<01n≥0u[n]={0n<01n≥0

This results in the following summation:

X(z)=∞∑n=0u[n]z−n=z0+z−1+z−2+z−3+ …X(z)=∑n=0∞u[n]z−n=z0+z−1+z−2+z−3+ …

An infinite sequence of summed numbers can converge to one number. For example:

1+12+14+18+116+ … =21+12+14+18+116+ … =2

If we continue the sequence according to the same pattern and sum all the elements, as the number of elements approaches infinity, the sum approaches the number 2. With the z-transform, the elements include a variable, but convergence can still occur—the sequence converges to a variable expression instead of a number.

The sequence shown above for the unit step converges as follows:

X(z)=∞∑n=0u[n]z−n=z0+z−1+z−2+z−3+ …=zz−1X(z)=∑n=0∞u[n]z−n=z0+z−1+z−2+z−3+ …=zz−1

Not all z-transforms will converge. Here are examples of discrete-time signals that have “well-behaved” z-transforms; note that all of these x[n] functions are multiplied by the unit step, such that the z-transform operation is applied to a sequence that is zero for n < 0.

x[n]=nu[n]              X(z)=z(z−1)2x[n]=nu[n]              X(z)=z(z−1)2

x[n]=anu[n]              X(z)=zz−ax[n]=anu[n]              X(z)=zz−a

x[n]=sin(ωn)u[n]              X(z)=zsin(ω)z2−2zcos(ω)+1

* What is pole? What is Zero?

The zplane function plots poles and zeros of a linear system. For example, a simple filter with a zero at -1/2 and a complex pole pair at and is. zer = -0.5; pol = 0.9\*exp (j\*2\*pi\* [-0.3 0.3]'); To view the pole-zero plot for this filter you can use zplane. Supply column vector arguments when the system is in pole-zero form.

* Explain these two commands that used in matlab.

The residue function in the standard MATLAB language is very similar to residuez. It computes the partial fraction expansion of continuous-time systems in the Laplace domain (see reference ), rather than discrete-time systems in the z -domain as does residuez.

**[R,p,C]=residuez(d,n)**

**[n,d]= residuez[R,p,C]**

The returned column vector r contains the residues, column vector p contains the pole locations, and row vector k contains the direct terms. The number of poles is

* n = length(a)-1 = length(r) = length(p)

The direct term coefficient vector k is empty if length(b) is less than length(a); otherwise:

* length(k) = length(b) - length(a) + 1

If p(j) = ... = p(j+s-1) is a pole of multiplicity s, then the expansion includes terms of the form

* http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/signal/refr2z81.gif

[b,a] = residuez(r,p,k) with three input arguments and two output arguments, converts the partial fraction expansion back to polynomials with coefficients in row vectors b and a.

The [residue](http://www.ece.northwestern.edu/local-apps/matlabhelp/techdoc/ref/residue.html) function in the standard MATLAB language is very similar to residuez. It computes the partial fraction expansion of continuous-time systems in the Laplace domain (see reference [1]), rather than discrete-time systems in the *z*-domain as does residuez.

**Algorithm**

residuez applies standard MATLAB functions and partial fraction techniques to find r, p, and k from b and a. It finds

* The direct terms a using [deconv](http://www.ece.northwestern.edu/local-apps/matlabhelp/techdoc/ref/deconv.html) (polynomial long division) when length(b) > length(a)-1.
* The poles using p = [roots](http://www.ece.northwestern.edu/local-apps/matlabhelp/techdoc/ref/roots.html)(a).
* Any repeated poles, reordering the poles according to their multiplicities.
* The residue for each nonrepeating pole *p*i by multiplying *b*(*z*)/*a*(*z*) by 1/(1 - *p*i*z*-1) and evaluating the resulting rational function at *z* = *p*i.
* The residues for the repeated poles by solving
  + S2\*r2 = h - S1\*r1
* for r2 using \. h is the impulse response of the reduced*b*(*z*)*/a*(*z*), S1 is a matrix whose columns are impulse responses of the first-order systems made up of the nonrepeating roots, and r1 is a column containing the residues for the nonrepeating roots. Each column of matrix S2 is an impulse response. For each root *p*j of multiplicity *s*j, S2 contains *s*j columns representing the impulse responses of each of the following systems.
* http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/signal/refr2z4a.gif
* The vector h and matrices S1 and S2 have n + xtra rows, where n is the total number of roots and the internal parameter xtra, set to 1 by default, determines the degree of overdetermination of the system of equations.

## Procedure:

## Open MATLAB Software.

## We create the new project.

## Make a new script and name it on the name of your lab.

## Write the code in main file and burn the code using run command.

## In the function file write the functionality of the code and code in main file.

## The function file and the script main file must be in the same folder.

## Run the mail file and get the output of the given input and observe the result.

## 

**Task1:** To check our residue functions, let us consider the rational function:

First rearrange **X (z)** so that it is a function in ascending power of

**Now using MATLAB**

Codes and Results:

CODE

clear all

close all

clc

n=[0 1 0];

d=[3 -4 1];

[R,p,C]=residuez(n,d)

[n,d]= residuez(R,p,C)

OUTPUT:

R =

0.8040

0.3071

p =

-1.6180

0.6180

C =

0

n =

1.1111 0 0

d =

1.0000 1.0000 -1.0000

>>

**Task2 :** Compute **the inverse Z –transform of.**

**; IZI>0.9**

Codes and Results:

clear all

close all

clc

n=[1 0 0];

d=[0.9 0.9 -0.9];

[R,p,C]=residuez(n,d)

[n,d]= residuez(R,p,C)

OUTPUT:

R =

0.8040

0.3071

p =

-1.6180

0.6180

C =

0

n =

1.1111 0 0

d =

1.0000 1.0000 -1.0000

>>

Conclusion: Today wee learned about z-transform and residue function.