Ex. No:	**Stability Analysis using Pole Zero Maps and Routh Hurwitz Criterion in Simulation Platform**
Date:	

Aim:

To perform stability analysis using Pole zero maps and Routh Hurwitz Criterion in simulation platform.

Introduction to Routh Hurwitz Criterion:

1. The Routh-Hurwitz criterion is a mathematical technique used in control theory and engineering to determine the stability of a linear time-invariant system. It is named after its developers, Edward Routh and Adolf Hurwitz. The criterion provides a systematic way to analyze the roots (or poles) of the characteristic equation of a system, and based on the locations of these roots, it can determine whether the system is stable, marginally stable, or unstable.

2. Pole Locations: The roots of the characteristic equation are known as the system's poles. The stability of the system is determined by the locations of these poles in the complex plane. Specifically, for continuous-time systems, a system is considered stable if all its poles have negative real parts (located in the left-half of the complex plane). For discrete-time systems, stability is determined by poles within the unit circle.

3. Routh-Hurwitz Criterion: The Routh-Hurwitz criterion is a systematic procedure for analyzing the stability of a system based on the coefficients of the characteristic equation. It involves constructing a special table known as the Routh array or Routh-Hurwitz table.

4. Routh-Hurwitz Table: The Routh-Hurwitz table is created using the coefficients of the characteristic equation. The table consists of rows and columns, where each entry is a coefficient value or a calculated value. The first column of the table contains the coefficients of the polynomial, and subsequent columns contain values calculated from these coefficients.

5. Stability Analysis: The Routh-Hurwitz criterion is applied to the Routh array to determine the number of sign changes in the first column. Based on these sign changes, the system's stability is determined as follows:

- If there are no sign changes in the first column, all system poles have negative real parts, and the system is stable.
- If there are sign changes in the first column but no sign changes in the subsequent columns, the system has poles with zero real parts, indicating marginally stable behavior.
- If there are sign changes in any column, excluding the case of marginally stable poles, the system is unstable.

Procedure:

Step 1. Obtain the Characteristic Equation
Step 2. Write Down the Coefficients
Step 3. Create the First Two Rows of the Routh Table
Step 4. Calculate the Remaining Rows of the Routh Table
Step 5. Check for Sign Changes in the First Column
Step 6. Determine System Stability
Step 7. Interpret the Results

Matlab code:

```
%% Routh-Hurwitz stability criterion
%
% The Routh-Hurwitz stability criterion is a necessary (and frequently
% sufficient) method to establish the stability of a single-input,
% single-output(SISO), linear time invariant (LTI) control system.
% More generally, given a polynomial, some calculations using only the
% coefficients of that polynomial can lead us to the conclusion that it
% is not stable.
% Instructions
% -----------
%
% in this program you must give your system coefficients and the
% Routh-Hurwitz table would be shown
%% Initialization
clear ; close all; clc
% Taking coefficients vector and organizing the first two rows
syms e s
syms k Integer
coeffVector = input('input vector of your system coefficients: \n i.e. [an an-1 an-2 ... a0] = ');
ceoffLength = length(coeffVector);
rhTableColumn = round(ceoffLength/2);
K=isnumeric(coeffVector);
z_c= ceoffLength-2;
% Initialize Routh-Hurwitz table with empty zero array
rhTable = zeros(ceoffLength,rhTableColumn);
rt_tab=sym(rhTable);
% Compute first row of the table
rt_tab(1,:) = coeffVector(1,1:2:ceoffLength);
% Check if length of coefficients vector is even or odd
if (rem(ceoffLength,2) ~= 0)
% if odd, second row of table will be
rt_tab(2,1:rhTableColumn - 1) = coeffVector(1,2:2:ceoffLength);
else
% if even, second row of table will be
rt_tab(2,:) = coeffVector(1,2:2:ceoffLength);
end
%% Calculate Routh-Hurwitz table's rows
% Set epss as a small value
epss = 0.01;
% Calculate other elements of the table
for i = 3:ceoffLength
% special case: row of all zeros
```

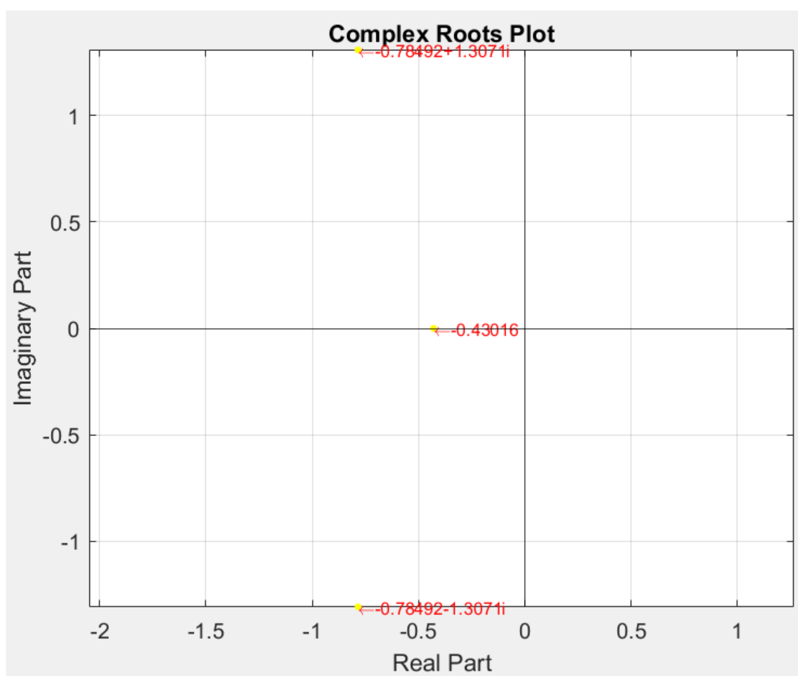
```
if rt_tab(i-1,:) == 0
order = (ceoffLength - i);
cnt1 = 0;
cnt2 = 1;
for j = 1:rhTableColumn - 1
rt_tab(i-1,j) = (order - cnt1) * rhTable(i-2,cnt2);
cnt2 = cnt2 + 1;
cnt1 = cnt1 + 2;
end
end
for j = 1:rhTableColumn - 1
% first element of upper row
firstElemUpperRow = rt_tab(i-1,1);
% compute each element of the table
rt_tab(i,j) = ((rt_tab(i-1,1) * rt_tab(i-2,j+1)) - ....
(rt_tab(i-2,1) * rt_tab(i-1,j+1))) / firstElemUpperRow;
end
% special case: zero in the first column
if rt_tab(i,1) == 0
rt_tab(i,1) = epss;
end
end
if(K==0)
assume(k>0);
x = solve(rt_tab(z_c+1,1)>0,k);
% x = real(x);
fprintf("Routh-Hurwitz Table for k = %f\n ",x(1,1));
disp(subs(rt_tab,[e k],[epss x(1,1)]));
rhTable = subs(rt_tab,[e k],[epss x(1,1)]);
routh_t = double(rhTable);
coeffVector(ceoffLength) = x(1,1);
else
routh_t = rhTable;
end
%% Compute number of right hand side poles(unstable poles)
% Initialize unstable poles with zero
unstablePoles = 0;
% Check change in signs
for i = 1:ceoffLength - 1
if sign(routh_t(i,1)) * sign(routh_t(i+1,1)) == -1
unstablePoles = unstablePoles + 1;
end
end
% Print calculated data on screen
fprintf('\n Routh-Hurwitz Table:\n')
routh_t;
% Print the stability result on screen
if unstablePoles == 0
fprintf('~~~~~> it is a stable system! <~~~~~\n')
else
fprintf('~~~~~> it is an unstable system! <~~~~~\n')
end
fprintf('\n Number of right hand side poles =%2.0f\n',unstablePoles)
```

```
reply = input('Do you want roots of system be shown? Y/N ', 's');
if reply == 'y' || reply == 'Y'
sysRoots = roots(coeffVector);
fprintf('\n Given polynomial coefficients roots :\n')
sysRoots
end
% Plot the roots on a graph
num_roots = size(sysRoots,1);
figure;
% Separate real and imaginary parts
real_parts = real(sysRoots);
imaginary_parts = imag(sysRoots);
% Create a scatter plot with different colors for each root
scatter(real_parts, imaginary_parts, 10,"yellow" , "filled");
colormap((parula(num_roots)))
% Set equal axis scaling for both real and imaginary parts
axis equal;
% Label the axes
xlabel('Real Part');
ylabel('Imaginary Part');
title('Complex Roots Plot');
% Annotate each point with its index
for i = 1:num_roots
str = strcat('\leftarrow ',string(sysRoots(i)));
text(real_parts(i), imaginary_parts(i),str,"FontSize",8,"Color",'r');
end
% Insert vertical and horizontal lines at x-axis and y-axis
xline(0, 'k-');
yline(0, 'k-');
grid on;
box on;
```

Output:

Result 1: Stable system

```
input vector of your system coefficients:
 i.e. [an an-1 an-2 ... a0] = [1 2 3 1]

 Routh-Hurwitz Table:
~~~~~> it is a stable system! <~~~~~

 Number of right hand side poles = 0
Do you want roots of system be shown? Y/N Y

 Given polynomial coefficients roots :

sysRoots =

  -0.7849 + 1.3071i
  -0.7849 - 1.3071i
  -0.4302 + 0.0000i
```



Result 2: Zero in first column

input vector of your system coefficients:
 i.e. [an an-1 an-2 ... a0] = [1 1 2 2 3 5]

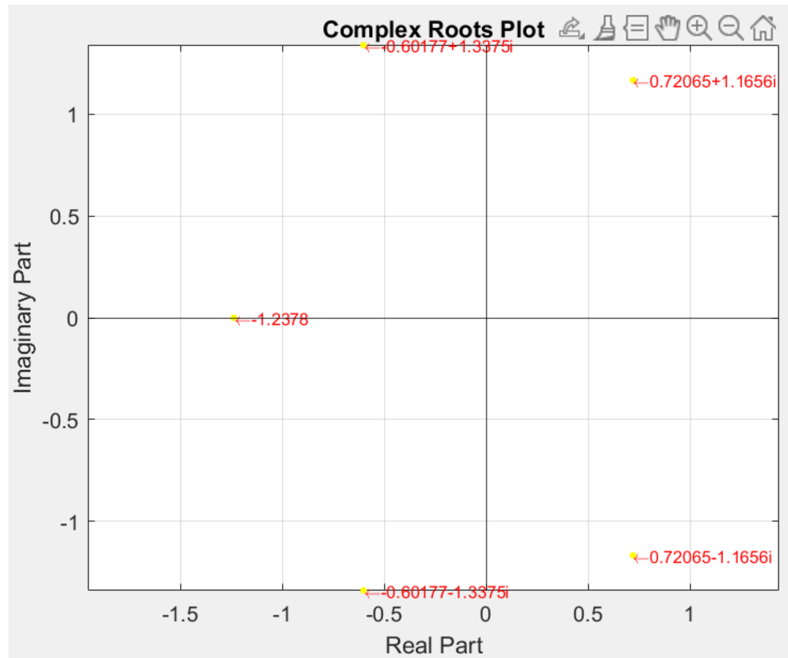
 Routh-Hurwitz Table:
~~~~~> it is a stable system! <~~~~~

 Number of right hand side poles = 0
Do you want roots of system be shown? Y/N Y

 Given polynomial coefficients roots :

sysRoots =

  0.7207 + 1.1656i
  0.7207 - 1.1656i
 -0.6018 + 1.3375i
 -0.6018 - 1.3375i
 -1.2378 + 0.0000i



Result 3: Row with all zeros

Input vector of your system coefficients:
 i.e. [an an-1 an-2 ... a0] = [1 9 24 24 24 24 23 15]

 Routh-Hurwitz Table:

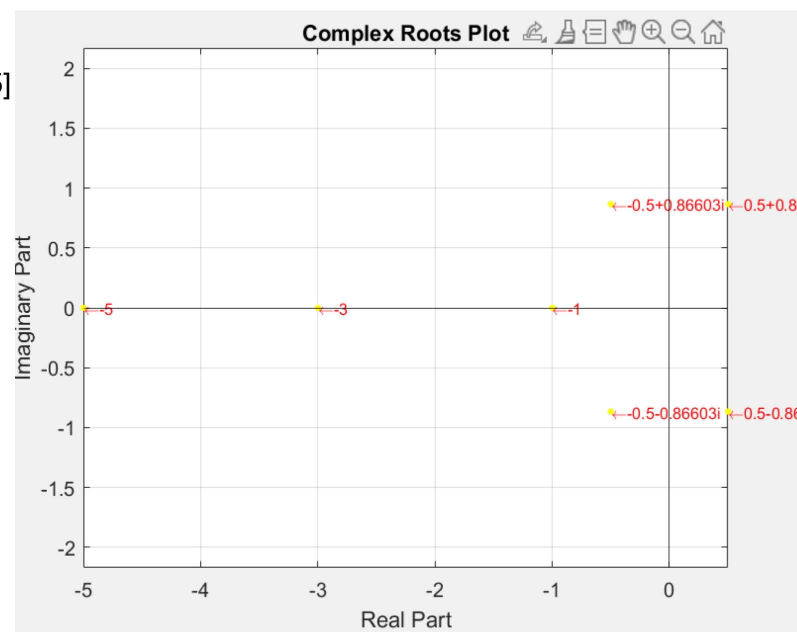
rhTable =

1.0000	24.0000	24.0000	23.0000
9.0000	24.0000	24.0000	15.0000
21.3333	21.3333	21.3333	0
15.0000	15.0000	15.0000	0
0.0100	0	0	0
15.0000	15.0000	0	0
-0.0100	0	0	0
15.0000	0	0	0

~~~~~> it is an unstable system! <~~~~~

 Number of right hand side poles = 2
Do you want roots of system be shown? Y/N Y

 Given polynomial coefficients roots :



```
sysRoots =

  -3.0000 + 0.0000i
  -5.0000 + 0.0000i
   0.5000 + 0.8660i
   0.5000 - 0.8660i
  -0.5000 + 0.8660i
  -0.5000 - 0.8660i
  -1.0000 + 0.0000i
```

Result 4: Finding k value

```
input vector of your system coefficients:
 i.e. [an an-1 an-2 ... a0] = [1 k 2 1]
Routh-Hurwitz Table for k = 1.500000
 [  1, 2]
[3/2, 1]
[4/3, 0]
[  1, 0]


 Routh-Hurwitz Table:
~~~~~> it is a stable system! <~~~~~

 Number of right hand side poles = 0
```

RESULT:

Thus stability analysis using Pole zero maps and Routh Hurwitz Criterion in the simulation platform is performed.