AAKASH DESHMANE
ECEN 628 MIDTERM 2 TAKE HOME PART
133008022

10.8)

The resulting characteristic equation from the given controller and plant is

$$z^3 + (\alpha_1 + \beta_0 + 3)z^2 + (3\beta_0 + \alpha_0 - 2\alpha_1 + 2)z + 2(\beta_0 + \alpha_0) = 0$$

For the closed loop system to be deadbeat, all the roots must lie at the origin.

$$\therefore \beta_0 = \alpha_0$$

Apart from identifying the set of stabilising controller gains, the main objective is to identify the maximum l_2 and l_infinity norm radii of the stability ball. However, if the system with the controller is given to be deadbeat, the system while stable, will have a stability margin of zero.

Hence, the maximum radius of the stability ball for the given plant will be 0 for any norm.

In the case that the system were not to be deadbeat, the l_2 and l_infinity norm stability ball radii can be found by the following method. The code is attached at the end for your reference.

10.9) Part 1 is solved in MATLAB by using Edge Theorem and finding the root space of the given family, code attached below.

Assuming all nominal parameters to be zero and all weights to be 1, we get:
Stability margin via L2 norm:

Characteristic eq:

$$p_1 s^3 + \left(\frac{17p_1}{2} - \frac{13p_2}{2}\right)s^2 + \left(\frac{25p_2}{2} - \frac{19p_1}{2}\right)s + p2 = 0$$

Substituting s = jw and Rearranging in matrix form:

$$\begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \begin{matrix} p_1 \\ p_2 \end{matrix} = \begin{matrix} 0 \\ 0 \end{matrix}$$

Stability margin is 0 for the range of all p1 and p2 values. This is evident from the plot from the Matlab code as well as there is one root at (0,0) in the complex plane. This means that the system has MARGINAL robust stability with no margins.
If the system is marginally robustly stable, then we do not need to find the l_infinity norm of the system because it will be zero.

10.10)

$$G(s) = \frac{\alpha_2 s^2 + \alpha_1 s + \alpha_0}{s^4 + \beta_3 + 3s^3 + \beta_2 s^2 + \beta_1 s + \beta_O}$$

$$\delta(s) = D(s) + N(s)$$
$$\delta(s) = s^4 + 3s^3 + (\alpha_2 + \beta_2)s^2 + (\alpha_1 + \beta_1)s + \alpha_0 + \beta_O + \beta_3$$

Let:

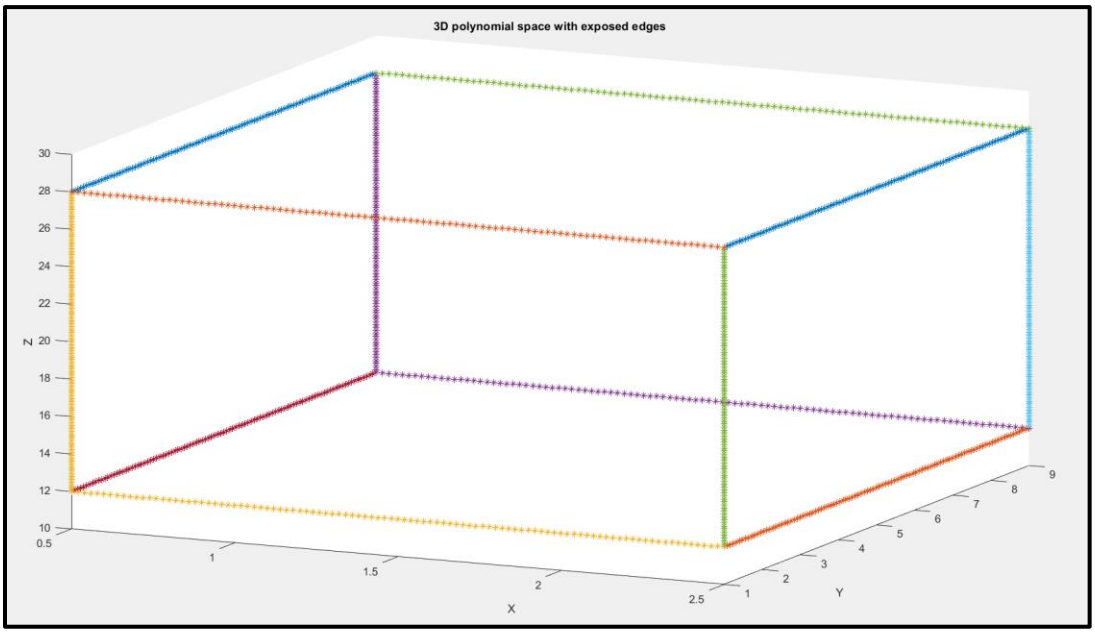
$$x = \alpha_1 + \beta_1$$
$$y = \alpha_2 + \beta_2$$
$$z = \alpha_0 + \beta_O + \beta_3$$

These are our perturbing polynomials.
The perturbing ranges of $\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_0, \beta_O, \beta_3$ are given to us. Plugging the minimum and maximum values of the range in the above equations, we get the ranges of x, y, and z as follows:

$$x \in [0.5,2.5]$$
$$y \in [1,9]$$
$$z \in [12,28]$$

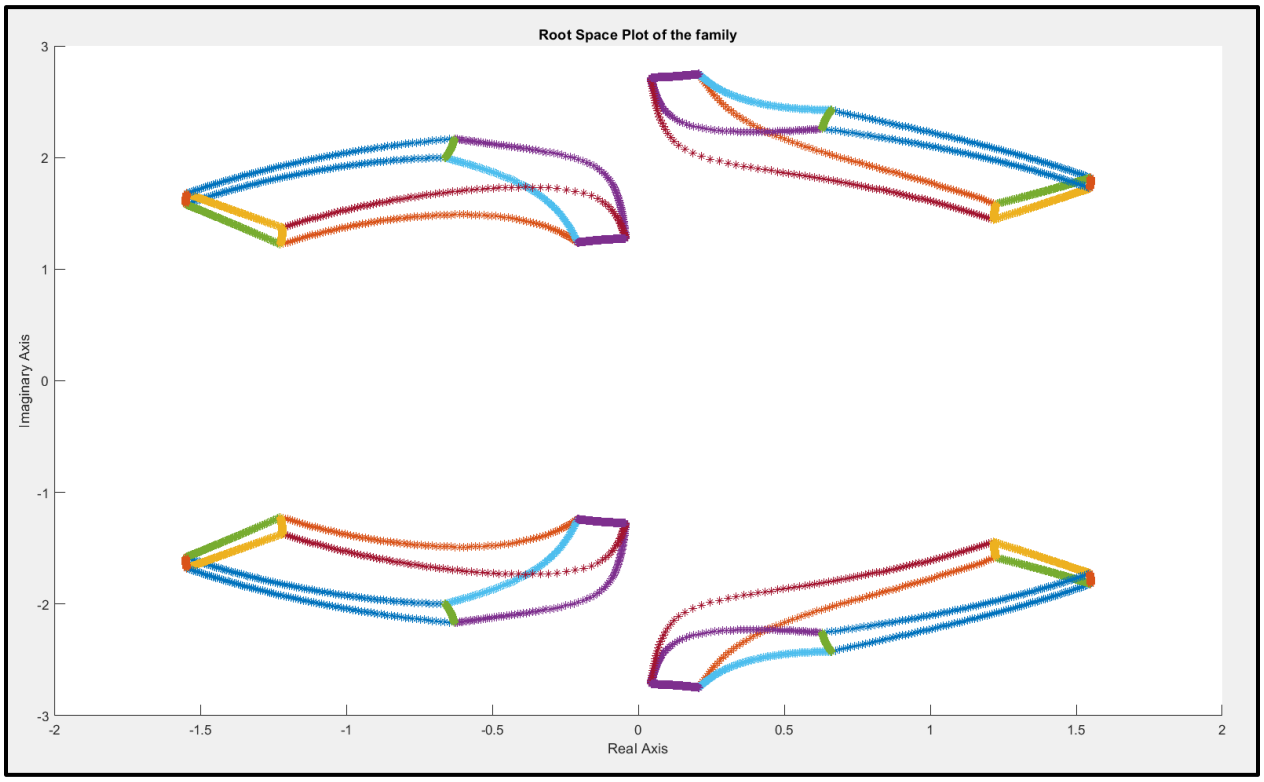
Now we follow the Edge Theorem algorithm to find the robust stability of the family of polynomials.

Constructing the 3D polynomial space in x, y, z:



Plot 1. Exposed Edges of the polynomial space

We now walk on this exposed edge and plot its corresponding root space. This is also done in MATLAB. Code attached below.



Plot 2. Root Space plot of the entire family.

It is evident that the boundary of the root space crosses over to the right half complex place. Applying Edge Theorem, we know that all roots of the family are encompassed by the root space boundary. Hence, the system is NOT robustly stable.

AAKASH DESHMANE

133008022 ECEN 628 MIDTERM EXAM 2 TAKE HOME PART Q10.8

```
clc
clear

% Define the deadbeat controller transfer function
num_g = [1 -2];
den_g = [1 3 2];
G = tf(num_g, den_g, 1);

% Define the controller parameter values
a0 = 0;
a1 = 0;
b0 = a0;

% Set the maximum allowable perturbation size
delta = 0.01;

% Initialize the l2 and linfinity norms
eps_l2 = 0;
eps_linf = 0;

% Compute the closed-loop transfer function for the nominal controller
C = tf([a0 a1], [1 b0], 1);
T = feedback(C*G, 1);
max_eig = max(abs(eig(T)));

% Iterate until the maximum eigenvalue is greater than or equal to 1
while max_eig < 1
    % Increase the l2 and linfinity norms by the perturbation size
    eps_l2 = eps_l2 + delta;
    eps_linf = eps_linf + delta;

    % Compute the maximum eigenvalue for the current perturbation size
    max_eig = 0;
    for i = 1:length(a0)
        for j = 1:length(a1)
            for k = 1:length(b0)
                dC = tf([eps_l2*a0(i) eps_l2*a1(j)], [1 eps_linf*b0(k)], 1);
                dG = -G*dC*G / (1 + G*dC);
                max_eig = max(max_eig, max(abs(eig(dG))));
            end
        end
    end

    % Print the current l2 and linfinity norms and the maximum eigenvalue
    fprintf('eps_l2 = %f, eps_linf = %f, max_eig = %f\n', eps_l2, eps_linf, max_eig);
end

% Print the final l2 and linfinity norms
fprintf('Largest l2 stability ball: ||dC||_2 <= %f\n', eps_l2);
fprintf('Largest linfinity stability ball: ||dC||_inf <= %f\n', eps_linf);
```

```
Largest l2 stability ball: ||dC||_2 <= 0.000000
Largest linfinity stability ball: ||dC||_inf <= 0.000000
```

Published with MATLAB® R2023a

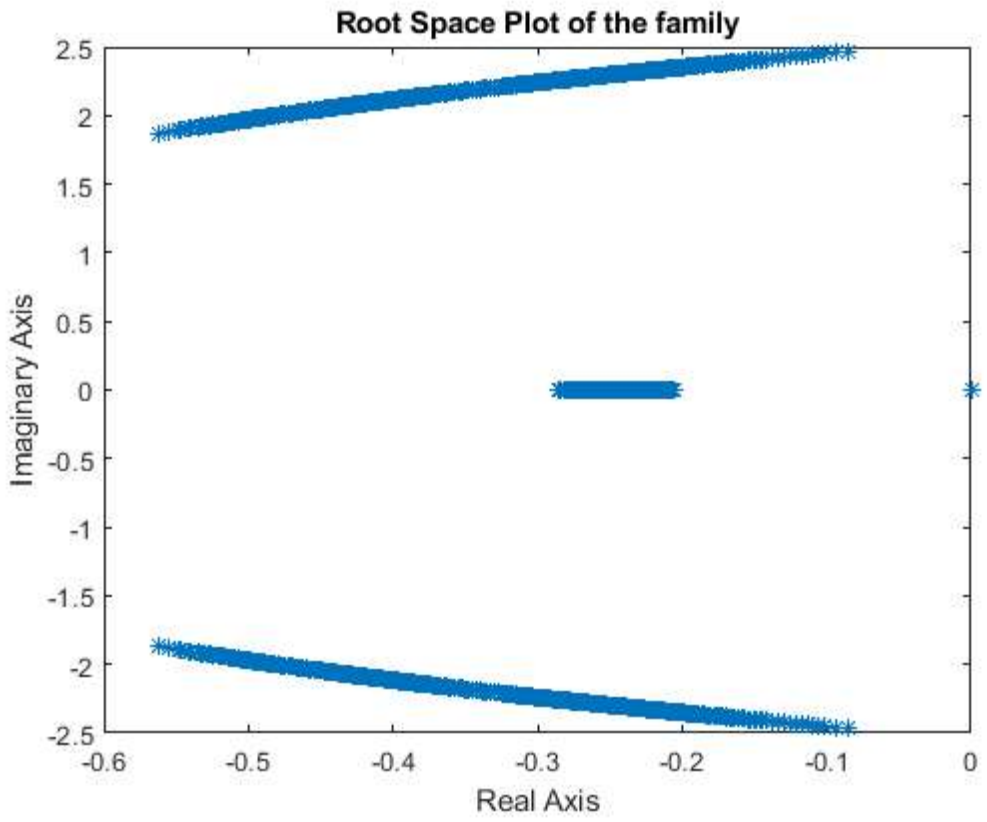
ECEN 628 ASSIGNMENT 3

AAKASH DESHMANE 133008022 Q.10.9

```
%_____________________________________________________________________

clear
clc
syms s p1 p2
format long
tstart = cputime;
% Closed loop characteristic polynomial

eqn = p1*s*(s + 9.5)*(s - 1) - p2*(6.5*s + 0.5)*(s - 2);
sol = solve(eqn,s,'MaxDegree',3);

n = 20;
p = zeros(3*n^4,2);
count = 1;

% Iterating through all family of polynomials

for p_1 = linspace(1,1.1,n)
    for p_2 = linspace(1.2,1.25,n)

        % Solving for the particular polynomial
        s1 = double(subs(sol,{p1,p2},{p_1,p_2}));

        % Calculating 1st root
        x = s1(1);
        r = real(x);
        i = imag(x);
        p(count,:) = [r i];
        count = count + 1;

        % Calculating 2nd root
        x = s1(2);
        r = real(x);
        i = imag(x);
        p(count,:) = [r i];
        count = count + 1;

        % Calculating 3rd root
        x = s1(3);
        r = real(x);
        i = imag(x);
        p(count,:) = [r i];
        count = count + 1;

    end
end

% Stability Condition
if max(p)>0
```

```
    disp('Roots are present in right half of complex plane. Hence, System is NOT ROBUSTLY STABLE!')
else
    disp('System is R0BUSTLY STABLE!')
end

% Plotting
plot(p(:,1),p(:,2),'*')
xlabel('Real Axis')
ylabel('Imaginary Axis')
title('Root Space Plot of the family')
tend = cputime - tstart
```

```
System is R0BUSTLY STABLE!

tend =

   1.937500000000000
```



Published with MATLAB® R2023a

AAKASH DESHMANE

133008022 ECEN 628 MIDTERM EXAM 2 TAKE HOME PART Q10.10

Contents

INITIALIZATION

```
clear
close all
clc
syms s a b c
total_roots = []
```

```
total_roots =

     []
```

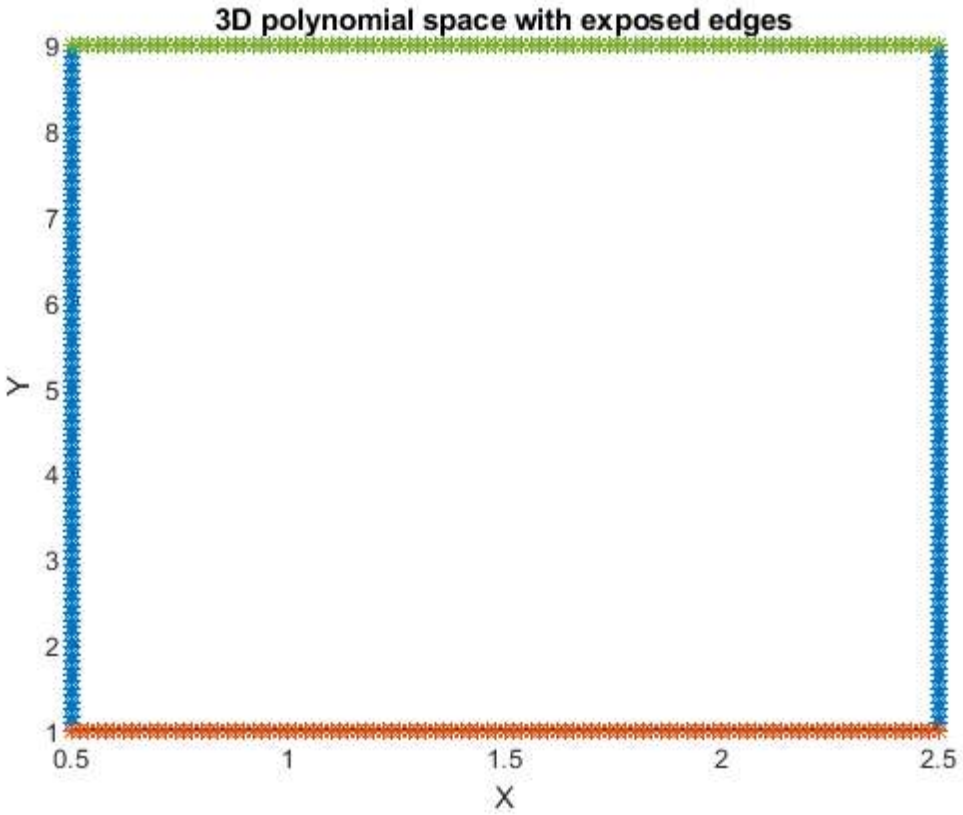
EXPOSED EDGE PLOTTING

```
hold on
figure (1)
edge_1 = walk_on_edge([2.5,1,28],[2.5,9,28]);
line = walk_on_edge([2.5,1,28],[2.5,9,28]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_2 = walk_on_edge([2.5,1,12],[2.5,9,12]);
line = walk_on_edge([2.5,1,12],[2.5,9,12]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_3 = walk_on_edge([0.5,1,12],[0.5,1,28]);
line = walk_on_edge([0.5,1,12],[0.5,1,28]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_4 = walk_on_edge([0.5,9,12],[0.5,9,28]);
line = walk_on_edge([0.5,9,12],[0.5,9,28]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_5 = walk_on_edge([2.5,1,12],[2.5,1,28]);
line = walk_on_edge([2.5,1,12],[2.5,1,28]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_6 = walk_on_edge([2.5,9,12],[2.5,9,28]);
line = walk_on_edge([2.5,9,12],[2.5,9,28]);
```

```
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_7 = walk_on_edge([0.5,1,12],[0.5,9,12]);
line = walk_on_edge([0.5,1,12],[0.5,9,12]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_8 = walk_on_edge([0.5,1,28],[0.5,9,28]);
line = walk_on_edge([0.5,1,28],[0.5,9,28]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_9 = walk_on_edge([0.5,1,28],[2.5,1,28]);
line = walk_on_edge([0.5,1,28],[2.5,1,28]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_10 = walk_on_edge([0.5,1,12],[2.5,1,12]);
line = walk_on_edge([0.5,1,12],[2.5,1,12]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_11 = walk_on_edge([0.5,9,12],[2.5,9,12]);
line = walk_on_edge([0.5,9,12],[2.5,9,12]);
plot3(line(:,1),line(:,2),line(:,3),'*')

edge_12 = walk_on_edge([0.5,9,28],[2.5,9,28]);
line = walk_on_edge([0.5,9,28],[2.5,9,28]);
plot3(line(:,1),line(:,2),line(:,3),'*')

xlabel('X')
ylabel('Y')
zlabel('Z')
title('3D polynomial space with exposed edges')
hold off
```

```
     8

     8

     8

     8

    16

    16

    16

    16

    16

    16

    16

    16

    16

     8
```

```
8
8
8
2
2
2
2
2
2
2
2
```

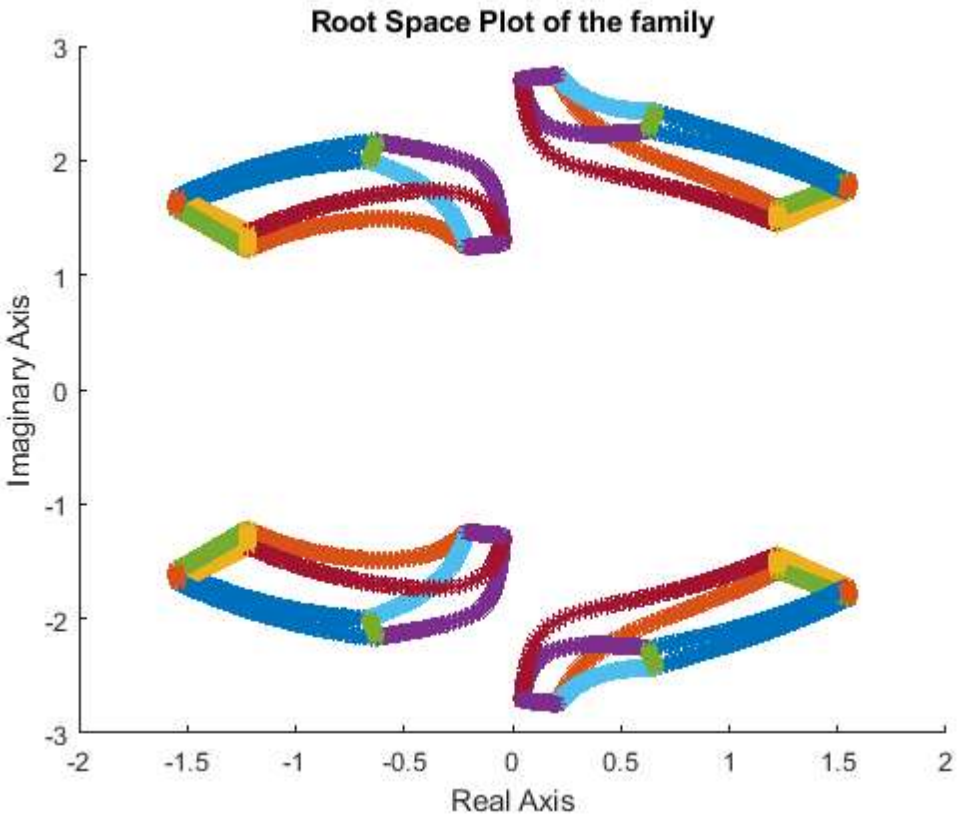


ROOT SPACE PLOTTING

```
figure(2)
hold on
% Plot edge 1
plot_roots = root_space(edge_1);
total_roots = [total_roots ; plot_roots];
```

```
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 2
plot_roots = root_space(edge_2);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 3
plot_roots = root_space(edge_3);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 4
plot_roots = root_space(edge_4);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 5
plot_roots = root_space(edge_5);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 6
plot_roots = root_space(edge_6);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 7
plot_roots = root_space(edge_7);
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 8
plot_roots = root_space(edge_8);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 9
plot_roots = root_space(edge_9);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 10
plot_roots = root_space(edge_10);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 11
plot_roots = root_space(edge_11);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

% Plot edge 12
plot_roots = root_space(edge_12);
total_roots = [total_roots ; plot_roots];
plot(plot_roots(:,1),plot_roots(:,2),'*')

xlabel('Real Axis')
ylabel('Imaginary Axis')
```

```
title('Root Space Plot of the family')

hold off
```



STABILITY CRITERIA CHECK ACCORDING TO EDGE THEOREM

```
clc

if max(total_roots(:,1))>0

    disp('Roots are present in right half of complex plane. Hence, System is NOT ROBUSTLY STABLE!')
else
    disp('System is R0BUSTLY STABLE!')
end
```

```
Roots are present in right half of complex plane. Hence, System is NOT ROBUSTLY STABLE!
```

FUNCTIONS

```
% 1) ROOT SPACE CURVE GENERATION
function roots = root_space(polynomials)
    syms s a b c
    eqn = s^4 + b*s^2 + a*s + c;
    sol = solve(eqn,s,'MaxDegree',4);
    count = 1;
    len = length(polynomials(:,1));
    roots = zeros(4*len,2);
```

```
    for i = 1: len

        a_sub = polynomials(i,1);
        b_sub = polynomials(i,2);
        c_sub = polynomials(i,3);


        s1 = double(subs(sol,{a,b,c},{a_sub,b_sub,c_sub}));

        % Calculating 1st root
        x = s1(1);
        r = real(x);
        i = imag(x);
        roots(count,:) = [r i];
        count = count + 1;

        % Calculating 2nd root
        x = s1(2);
        r = real(x);
        i = imag(x);
        roots(count,:) = [r i];
        count = count + 1;

        % Calculating 3rd root
        x = s1(3);
        r = real(x);
        i = imag(x);
        roots(count,:) = [r i];
        count = count + 1;

        % Calculating 4th root
        x = s1(4);
        r = real(x);
        i = imag(x);
        roots(count,:) = [r i];
        count = count + 1;
    end

end

% 2) EDGE GENERATION
function line = walk_on_edge(start_point, end_point)
    n = 100;
    l = 1/n;
    %lambda = 0;
    d1 = 0;
    for i = 1:3
        d = (end_point(i) - start_point(i))^2;
        d1 = d1 + d;
    end
    distance = sqrt(d1);
    disp(distance)

    if start_point(1) ~= end_point(1)
        line = zeros(length(linspace(0,1,n)),3);
        i = 1;
        for lambda = 0:l:1
            line(i,:) = [start_point(1)+lambda*distance start_point(2) start_point(3)];
```

```
            i = i + 1;
        end
    end

    if start_point(2) ~= end_point(2)
        line = zeros(length(linspace(0,1,n)),3);
        i = 1;
        for lambda = 0:1:1
            line(i,:) = [start_point(1) start_point(2)+lambda*distance start_point(3)];
            i = i + 1;
        end
    end

    if start_point(3) ~= end_point(3)
        line = zeros(length(linspace(0,1,n)),3);
        i = 1;
        for lambda = 0:1:1
            line(i,:) = [start_point(1) start_point(2) start_point(3)+lambda*distance ];
            i = i + 1;
        end
    end


end

%_____________________________________________________________________
```

Published with MATLAB® R2023a