Table of Contents


```
close all; clear; clc;
```

Problem 2

```
t = 1;
q = -0.05 + 0.02*t;

t = 2;
vbe = [20 + cos(4*t+pi/6); 0.3*sin(4*t); 2*sin(4*t)];
wb = [0.06; -0.05 + 0.02*t; 0.03 - 0.02*t + 0.005*t^2];
wbe = [sin(t/100); 2*cos(t/100); -0.4];

V = vbe - wbe;

alpha = atan(V(3)/V(1)) * 180/pi;


% Part 3
t = 3;
vbe_dot = [-4*sin(4*t + pi/6); 1.2*cos(4*t); 8*cos(4*t)];
wb = [0.06; -0.05 + 0.02*t; 0.03 - 0.02*t + 0.005*t^2];
wtilde = [0, -wb(3), wb(2); wb(3), 0, -wb(1); -wb(2), wb(1), 0];
vbe = [20 + cos(4*t+pi/6); 0.3*sin(4*t); 2*sin(4*t)];

abe = vbe_dot + wtilde*vbe;
```

Problem 3

```
a1 = 18.74;
a2 = -6.41;
kp = 0.89;
kd = -0.8;
ki = 0.01;

% Values of feed forward to loop through
kff_vals = linspace(-100, 0, 100);

% Vectors to store the poles of the closed loop tf
poles = zeros(length(kff_vals), 3);

for i = 1:length(kff_vals)
    kff = kff_vals(i);
```

```
    % Define numerator and denom of the tf
    numerator = [kp*kd + kp*kff, ki*kd + ki*kff];
    denominator = [1/a2, a1/a2 + kd, kp*kd + kp*kff, ki*kd + ki*kff];

    % Create the tf
    sys = tf(numerator, denominator);

    % Calculate the poles of the denominator
    poles(i,:) = pole(sys);

    if(sum(imag(poles(i,:)) ~= 0) > 0)
        kff;
        x = imag(poles(i,:));
    else
        kff;
        x = real(poles(i,:));
    end

end

% Create color gradients
red_colors = [linspace(0.2,1,100)', zeros(100,1), zeros(100,1)];
blue_colors = [zeros(100,1), zeros(100,1), linspace(0.2,1,100)'];

% Apply the 'autumn' colormap
colors = cool(length(kff_vals));


figure();
for i = 1:length(poles(:,1))
    pole1 = plot(real(poles(i,1)), imag(poles(i,1)), '-x', 'MarkerSize', 4,
'Color', colors(i,:));
    hold on;
    pole2 = plot(real(poles(i,2)), imag(poles(i,2)), '-^', 'MarkerSize', 4,
'Color', colors(i,:));
    pole3 = plot(real(poles(i,3)), imag(poles(i,3)), '.', 'MarkerSize', 20,
'Color', 'g');
end


xlabel('Real Axis')
ylabel('Imaginary Axis')
title('Root Locus')
legend([pole1, pole2, pole3], 'Pole 1', 'Pole 2', 'Pole 3');


% Part 3
% Define numerator and denom of the tf
kff = 0;
numerator = [kp*kd + kp*kff, ki*kd + ki*kff];
denominator = [1/a2, a1/a2 + kd, kp*kd + kp*kff, ki*kd + ki*kff];

% Create the tf
```

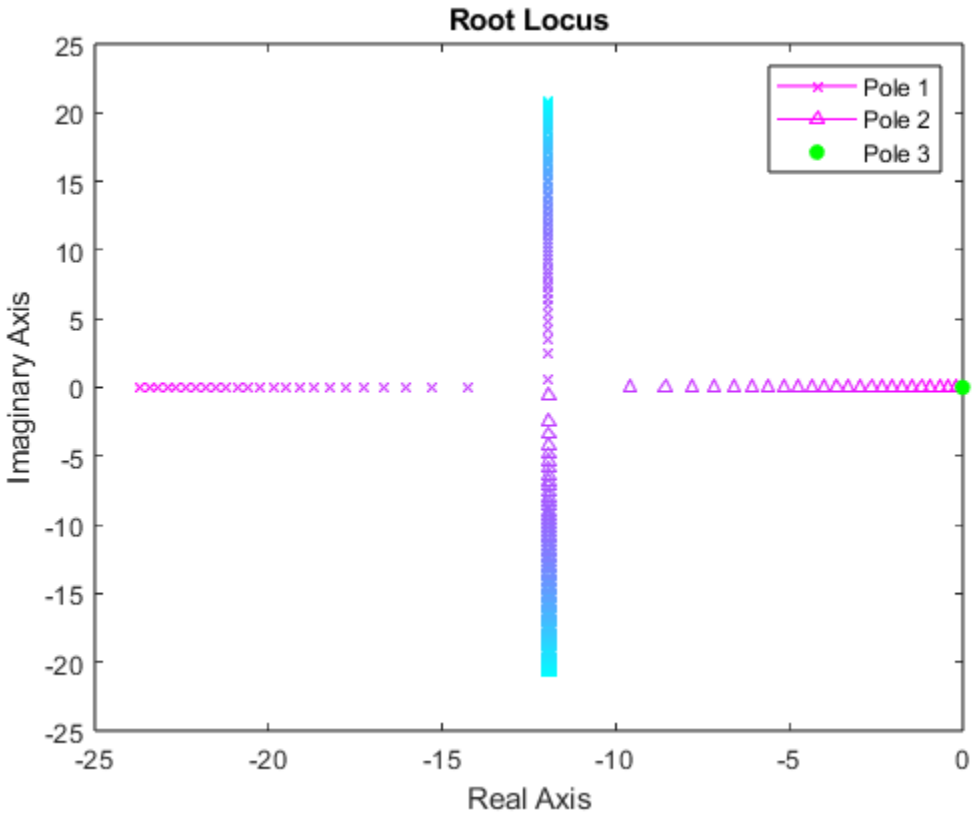
```
sys = tf(numerator, denominator);

% Calculate the poles of the denominator
pole_part3 = pole(sys);
zero_part3 = zero(sys);


% Define numerator and denom of the tf
kff = a1/a2;
numerator = [kp*kd + kp*kff, ki*kd + ki*kff];
denominator = [1/a2, a1/a2 + kd, kp*kd + kp*kff, ki*kd + ki*kff];

% Create the tf
sys = tf(numerator, denominator);

% Calculate the poles of the denominator
pole_part3_2 = pole(sys);
zero_part3_2 = zero(sys);
```



Problem 4

```
load('RAAVENMatrices.mat');

[vec_lat, vals_lat] = eig(Alat);
[vec_lon, vals_lon] = eig(Alon);
```

```
dutch_roll = vals_lat(4,4);
wn_dutch = sqrt(real(dutch_roll)^2 + imag(dutch_roll)^2);
damp_dutch = -real(dutch_roll) / wn_dutch;

short = vals_lon(2,2);
wn_short = sqrt(real(short)^2 + imag(short)^2);
damp_short = -real(short) / wn_short;

phugoid = vals_lon(4,4);
wn_phugoid = sqrt(real(phugoid)^2 + imag(phugoid)^2);
damp_phugoid = -real(phugoid) / wn_phugoid;

damp(Alon)

roll = vals_lat(2,2);
spiral = vals_lat(3,3);
t_roll = -1 / real(roll);
t_spiral = -1 / real(spiral);


% Part 4
roll_eigenval = real(vals_lat(2,2));
roll_eigenvec = real(vec_lat(:,2));

roll_init = roll_eigenvec .* (0.035/roll_eigenvec(4));


% Part 5
kff_roll = -1*Alat(2,2) / Blat(2,1);
```

Pole	Damping	Frequency (rad/TimeUnit)	Time Constant (TimeUnit)
0.00e+00	-1.00e+00	0.00e+00	Inf
-3.67e+00 + 1.10e+01i	3.15e-01	1.16e+01	2.72e-01
-3.67e+00 - 1.10e+01i	3.15e-01	1.16e+01	2.72e-01
-1.02e-02 + 7.59e-01i	1.35e-02	7.59e-01	9.77e+01
-1.02e-02 - 7.59e-01i	1.35e-02	7.59e-01	9.77e+01

Published with MATLAB® R2023b