```
% Initialize cell array to store data for each step
step_array_of_array = cell(3, 1);



A_prime = []; % veriable use to stor the totale gain of the plant from physical testing
start_step_value = []; % used to recored starting value form sep test
ending_step_value = [];% used to recored the ending value form sep test
constat_V_to_m = []; %That he used to convert from volts to metres

for step_number = 1:3
    % Read data from the CSV file
    step_array_of_array = readmatrix([ 'step ', num2str(step_number), '.csv']);

    % Calculate delta t and delta y
    delta_time = diff(step_array_of_array(:, 1));
    delta_position = diff(step_array_of_array(:, 4));

    velocities = [0; delta_position ./ delta_time];

    % Add velocities to the main array
    step_array_of_array(:, 5) = velocities;

    % Initialize variables for data extraction
    step_correct_start = [];
    time_start = 0;
    start_higt = 0;

    for i = 1:length(velocities)
        if (velocities(i) > 0.5) && (time_start == 0)
            time_start = step_array_of_array(i, 1);
            start_higt = step_array_of_array(i, 4);
        end
        % anilisine and storing each steps perameters
        if velocities(i) > 0.5
            step_correct_start = [step_correct_start; step_array_of_array(i, 1) - ↙
time_start, velocities(i), step_array_of_array(i, 4) - start_higt];
            if step_array_of_array(i, 3) < 10
                constat_V_to_m = [constat_V_to_m; step_number, step_array_of_array(i, ↙
3) / step_array_of_array(i, 4)];
            end
        end
    end

    % Store corrected data in the cell array
    steps{step_number} = step_correct_start;

    % Calculate A_prime
    A_prime = [A_prime; max(steps{step_number}(:, 2))];
    disp("A_prime for " + step_number + ' is ' + max(steps{step_number}(:, 2)));
```

```
    % Store start and end values
    start_step_value = [start_step_value; step_array_of_array(5, 2)];
    ending_step_value = [ending_step_value; step_array_of_array(500, 2)];
end

% Calculate damping coefficient
index = [];
tau = [];
b_The_damping_coefficient = [];

for step_number = 1:3
    targetValue = A_prime(step_number) * 0.63;
    [~, index] = min(abs(steps{step_number}(:, 2) - targetValue));
    tau = [tau; steps{step_number}(index, 1)];
end

b_The_damping_coefficient = 1 ./ tau;
b_The_damping_coefficient_ave = mean(b_The_damping_coefficient);

% Plot step response form data gathered
figure;
for step_number = 1:3
    subplot(3, 1, step_number);
    plot(steps{step_number}(:, 1), steps{step_number}(:, 2));
    title(['Step ', num2str(step_number)]);
    xlabel("time in s");
    ylabel("velocity in m/s");
    line(xlim, [A_prime A_prime], 'Color', 'r'); % Adding the maxe value line
    line([tau(step_number) tau(step_number)], ylim, 'Color', 'g'); % Adding the tau
value line
end




% Calculate average values
A_prime_ave = mean(A_prime);
ending_step_value_ave = mean(ending_step_value);
start_step_value_ave = mean(start_step_value);
b = ending_step_value_ave - start_step_value_ave;
constat_V_to_m_ave = mean(constat_V_to_m(:, 2));

% Calculate tau
tau_ave = mean(tau);

A = A_prime_ave / (b * constat_V_to_m_ave);
disp("A prime ave = " + A_prime_ave + newline + "the value of b = " + b + newline + ...
    + newline + "k average = " + constat_V_to_m_ave ...
    + newline + "A ave = " + A ...
```

```
    + newline + "tau and b =  " + tau_ave);



% Define the transfer function G
s = tf('s');
G = tf(A_prime_ave, [tau_ave, 1]);

% Plot step response for G for velocity
figure;
hold on;
step(G);


% Plot step responses for each step
plot(steps{1}(:, 1), steps{1}(:, 2), 'r');
plot(steps{2}(:, 1), steps{2}(:, 2), 'g');
plot(steps{3}(:, 1), steps{3}(:, 2), 'b');

legend('Transfer Function', 'Step 1', 'Step 2', 'Step 3');
xlabel('Time in seconds');
ylabel('Amplitude of velocity');
title('Step Response for velosity');
grid on;
hold off

% Define controller and plant
k = 0.0033;
gain = tf(k);
Constat_V_to_m_ave = tf(constat_V_to_m_ave);
plant = tf(A * constat_V_to_m_ave / (s + ((s^2) * tau_ave)));

% ploting the  step responses for each step potition
figure;
hold on;
step(plant)

plot(steps{1}(:, 1), steps{1}(:, 2), 'r');
plot(steps{2}(:, 1), steps{2}(:, 2), 'g');
plot(steps{3}(:, 1), steps{3}(:, 2), 'b');
tow_ave
legend('Transfer Function', 'Step 1', 'Step 2', 'Step 3');
xlabel('Time in seconds');
ylabel('Amplitude of velocity');
title('Step Response Comparison');
grid on;

hold off;
```

```
gain

% Calculate Kv
Kv = dcgain(s * G);

% Calculate steady-state error for ramp input
ess = 1 / Kv;
disp("Steady-state error: " + ess);

%controlSystemDesigner(plant);

% Closed-loop system
G_closed_loop = feedback(plant * gain, 1);

% ploting the closed loop respons
s = tf('s');
figure;
hold on;
step(G_closed_loop);
step(tf(1), "r");
title("step respon at 3.3 gain");
hold off;

% Plot ramp response
figure;
hold on;
step(G_closed_loop/s);
step(1/s, "r");
xlim([0, 200]);
title('Ramp Response');
hold off;

% Interpolate values for error calculation
t = 0:0.01:200;
y1 = step(G_closed_loop/s, t);
y2 = step(1/s, t, "r");
t_desired = 120;
y1_desired = interp1(t, y1, t_desired);
y2_desired = interp1(t, y2, t_desired);
difference = abs(y1_desired - y2_desired);
calculating20error = t_desired - t_desired * (1 - 0.2);

disp("Difference between the two values = " + difference + " 20% error = " +↙
calculating20error);

% Anilisiny the controles data with gain at 3.3

test_time = [];

for i = 1:2
```

```
    % Read data from the CSV file
    step_array_of_array = readmatrix(['test 3.3 gain ', num2str(i), '.csv']);
    figure
    hold on
    plot(step_array_of_array(:,1),step_array_of_array(:,4), "B");
    xline([0 ,120]);
    yline(7,"r","step test");
    title("test " +i+" where gain is set to 3.3");
    xlabel("time in stcondes");
    ylabel("altetued in m");
end

% Anilisiny the controles data with gain at 33

step_array_of_array = readmatrix(['gain at 33.csv']);
    figure
    hold on;
    plot(step_array_of_array(:,1),step_array_of_array(:,4), "B");
    xline([0 ,160]);
    yline(7,"r","step test");
    title("test " +i+" where gain is set to 33");
    xlabel("time in stcondes");
    ylabel("altetued in m");
hold off;

% cheching areodnamic tolerensis +10%
tua_toleran_up = tau_ave*(1.1);
A_tolerans_up = A_prime_ave*(1.1);

G_tolarece_up = tf(A_tolerans_up*constat_V_to_m_ave, [tua_toleran_up, 1]);
figure;
hold on;
step(G_tolarece_up);


% Plot step responses for each step
plot(steps{1}(:, 1), steps{1}(:, 2), 'r');
plot(steps{2}(:, 1), steps{2}(:, 2), 'g');
plot(steps{3}(:, 1), steps{3}(:, 2), 'b');

legend('Transfer Function', 'Step 1', 'Step 2', 'Step 3');
xlabel('Time in seconds');
ylabel('Amplitude of velocity');
title('Step Response for velosity with + 10% arodynamic tolaranc' );
grid on;
hold off;

% effect on the control system
plant_10_plus = tf(A_tolerans_up * constat_V_to_m_ave / (s + ((s^2) * ↙
tua_toleran_up)));
```

```
% Closed-loop system +10%
G_closed_loop_10_plus = feedback(plant_10_plus * gain, 1);

% ploting the closed loop respons

figure;
hold on;
step(G_closed_loop_10_plus);
step(tf(1), "r");
title("step respon at 3.3 gain with + 10% arodynamic tolaranc " );
hold off;

% cheching areodnamic tolerensis +10%
tua_toleran_down = tau_ave*(0.9);
A_tolerans_down = A_prime_ave*(0.9);

G_tolarece_down = tf(A_tolerans_down*constat_V_to_m_ave, [tua_toleran_down, 1]);
figure;
hold on;
step(G_tolarece_down);


% Plot step responses for each step
plot(steps{1}(:, 1), steps{1}(:, 2), 'r');
plot(steps{2}(:, 1), steps{2}(:, 2), 'g');
plot(steps{3}(:, 1), steps{3}(:, 2), 'b');

legend('Transfer Function', 'Step 1', 'Step 2', 'Step 3');
xlabel('Time in seconds');
ylabel('Amplitude of velocity');
title('Step Response for velosity with - 10% arodynamic tolaranc' );
grid on;
hold off;

% effect on the control system
plant_10_min = tf(A_tolerans_down * constat_V_to_m_ave / (s + ((s^2) * ↙
tua_toleran_down)));

% Closed-loop system +10%
G_closed_loop_10_min = feedback(plant_10_min * gain, 1);

% ploting the closed loop respons

figure;
hold on;
step(G_closed_loop_10_min);
step(tf(1), "r");
title("step respon at 3.3 gain with - 10% arodynamic tolaranc " );
hold off;
```