EE7207 Assignment-2

Wu Tianwei

Matriculation No. G2101446F

e-mail: WU0008EI@e.ntu.edu.sg

1. Fuzzy Control

(a) According to the rules given in Table 1.2, I choose Matlab Fuzzy Logic Designer to set all the membership functions and rules and export as a $.fis$ file. Then, in the Matlab code I use $evalfis$ to get the output u in each iteration.

Matlab Code:

```
clc;
clear;

car_parking_fuzzy(0, 40, 20, 200, 1);
car_parking_fuzzy(90, -30, 10, 200, 3);
car_parking_fuzzy(220 - 360, 30, 40, 250, 5);
car_parking_fuzzy(-10, 10, 50, 160, 7);
```

```
function [] = car_parking_fuzzy(theta_i, y_i, x_i, t_end,
figure_No)
    % Read fis file
    fis = readfis('fuzzy_logic');

    % Constants
    L = 2.5;
    T = 0.1;
    v = 0.5;

    % Initial values
    t = 0;
    tsim = 1;

    theta = theta_i;
    y = y_i;
    x = x_i;
    u = deg2rad(evalfis(fis,[y, theta]));

    % Plot Data
    x_plot(tsim) = x;
    y_plot(tsim) = y;
    theta_plot(tsim) = theta;
```

```
    u_plot(tsim) = rad2deg(u);

    theta = deg2rad(theta);

    while t < t_end
        x = x + v * T * cos(theta);
        y = y + v * T * sin(theta);
        theta = theta + v * T * tan(u) / L;
        u = deg2rad(evalfis(fis,[y, rad2deg(theta)]));
        t = t + T;
        tsim = tsim + 1;

        % Update plot Data
        x_plot(tsim) = x;
        y_plot(tsim) = y;
        theta_plot(tsim) = rad2deg(theta);
        u_plot(tsim) = rad2deg(u);
    end

    figure(figure_No);
    t_plot = 0 : T : t + T;
    subplot(4, 1, 1);
    plot(t_plot, x_plot, 'b', 'LineWidth', 2);
    [main_title, ~] = title('Simulation Results', ['Initial
condition:¦È=', num2str(theta_i), ', y=', num2str(y_i), ', x=',
num2str(x_i)]);
    main_title.FontSize = 16;
    xlabel('t (second)', 'FontSize', 12, 'FontWeight', 'bold');
    ylabel('x (m)', 'FontSize', 12, 'FontWeight', 'bold');
    grid on;
    subplot(4, 1, 2);
    plot(t_plot, y_plot, 'r', 'LineWidth',2);
    xlabel('t (second)', 'FontSize', 12, 'FontWeight', 'bold');
    ylabel('y (m)', 'FontSize', 12, 'FontWeight', 'bold');
    grid on;
    subplot(4, 1, 3);
    plot(t_plot, theta_plot, 'g', 'LineWidth',2);
    xlabel('t (second)', 'FontSize', 12, 'FontWeight', 'bold');
    ylabel('¦È (degree)', 'FontSize', 12, 'FontWeight', 'bold');
    grid on;
    subplot(4, 1, 4);
    plot(t_plot, u_plot, 'c', 'LineWidth',2);
    xlabel('t (second)', 'FontSize', 12, 'FontWeight', 'bold');
    ylabel('u (degree)', 'FontSize', 12, 'FontWeight', 'bold');
```

```
    grid on;

    figure(figure_No + 1);
    plot(x_plot, y_plot, 'c', 'LineWidth', 2);
    [main_title, ~] = title('Trajactory of the Truck', ['Initial
condition:¦È=', num2str(theta_i), ', y=', num2str(y_i), ', x=',
num2str(x_i)]);
    main_title.FontSize = 16;
    xlabel('x (m)', 'FontSize', 12, 'FontWeight', 'bold');
    ylabel('y (m)', 'FontSize', 12, 'FontWeight', 'bold');
    grid on;
end
```

In this question, 4 initial conditions need to be simulated.

Case	1	2	3	4
θ	0	90	220	-10
y	40	-30	30	10
x	20	10	40	50

The simulation results are shown in the Figure 1 ~ 8.

Case 1:

$$\theta = 0, y = 40, x = 20$$

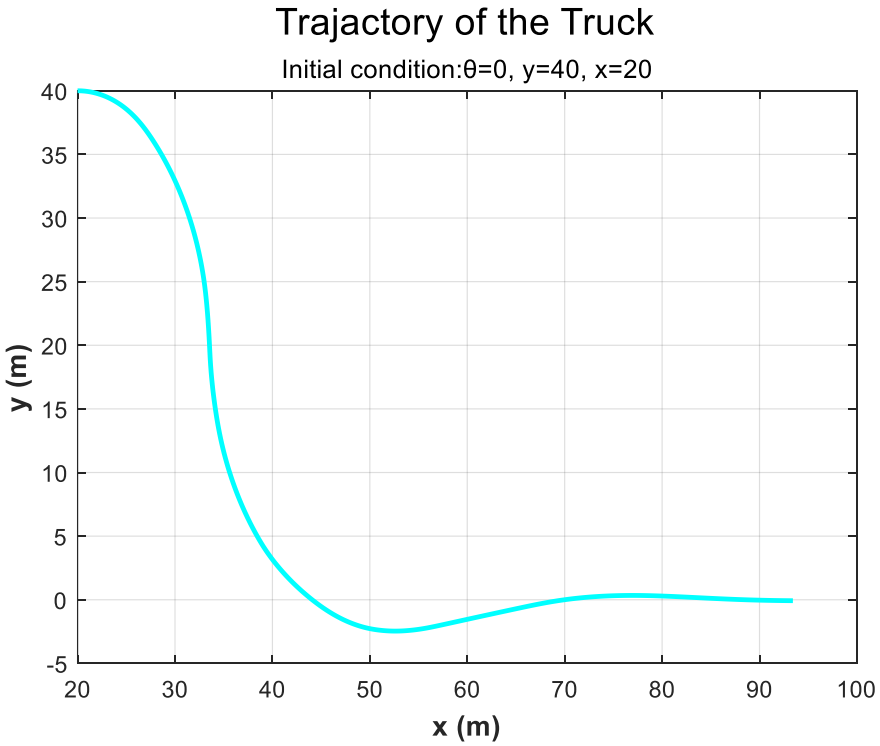


Figure 1

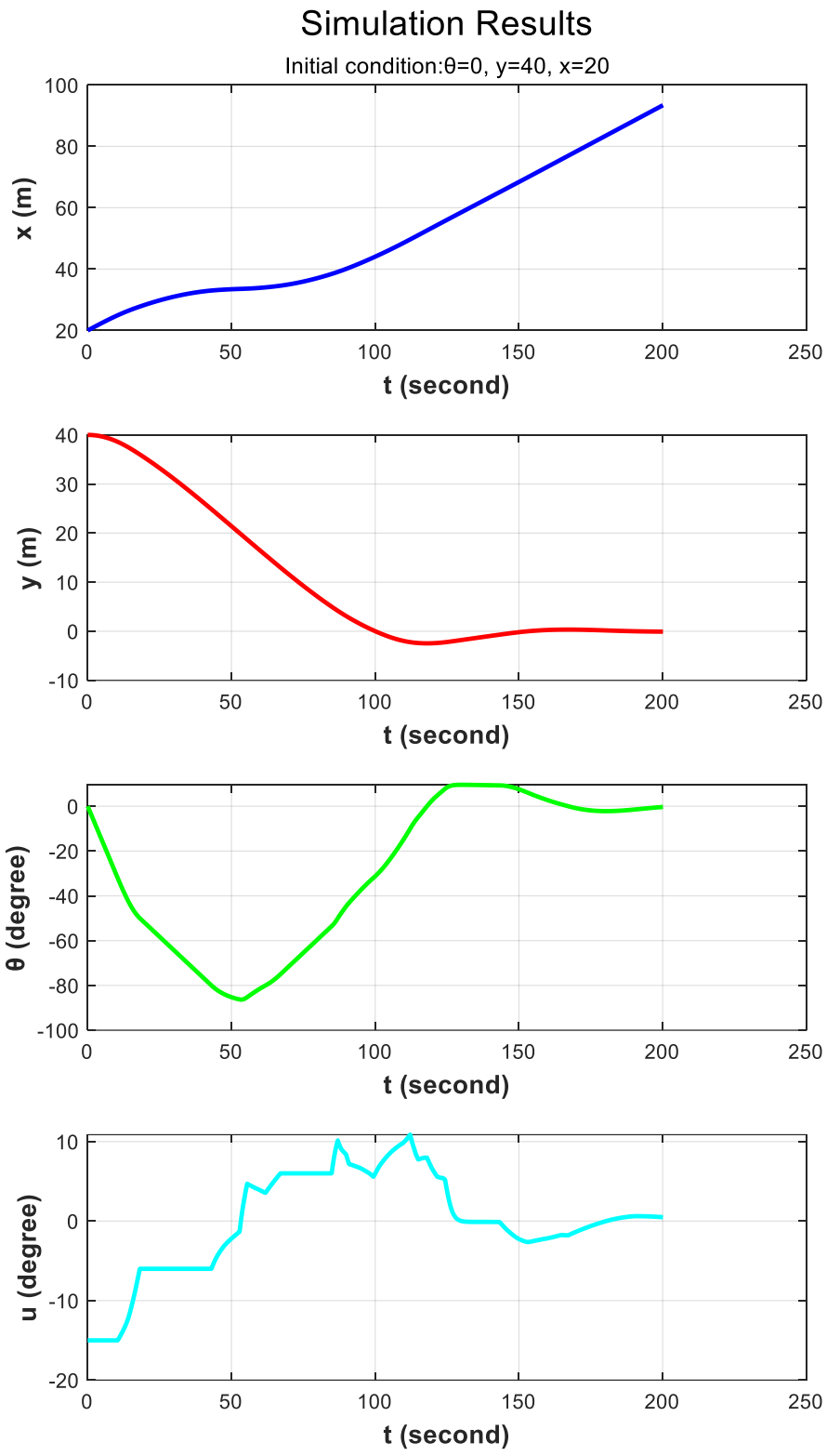

Simulation Results
Initial condition:θ=0, y=40, x=20
x (m)
t (second)
y (m)
t (second)
θ (degree)
t (second)
u (degree)
t (second)

Figure 2

Case 2:

$$\theta = 90, y = -30, x = 10$$

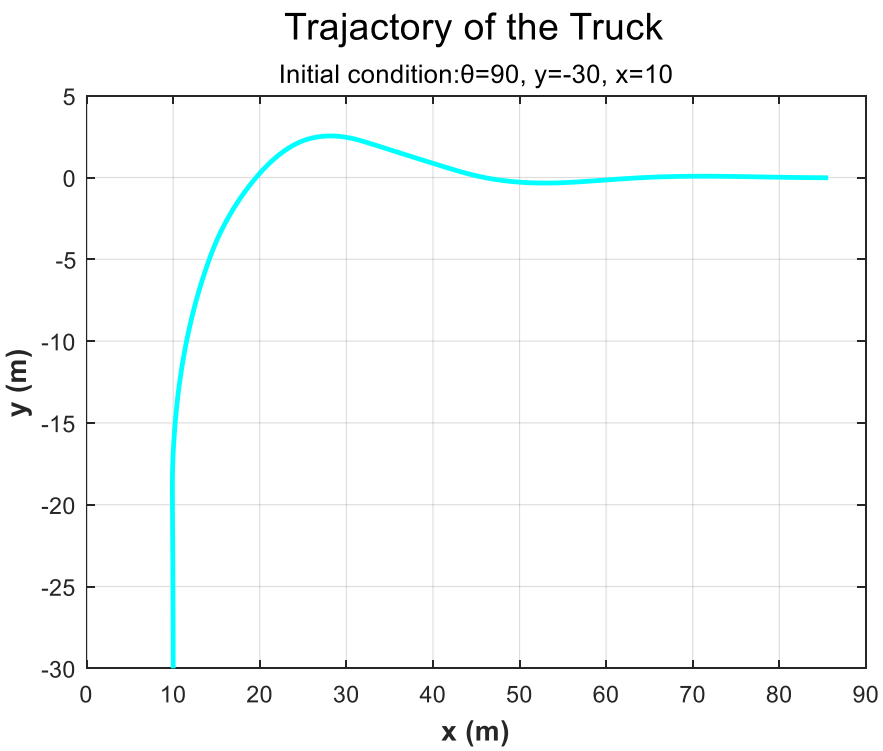


Figure 3

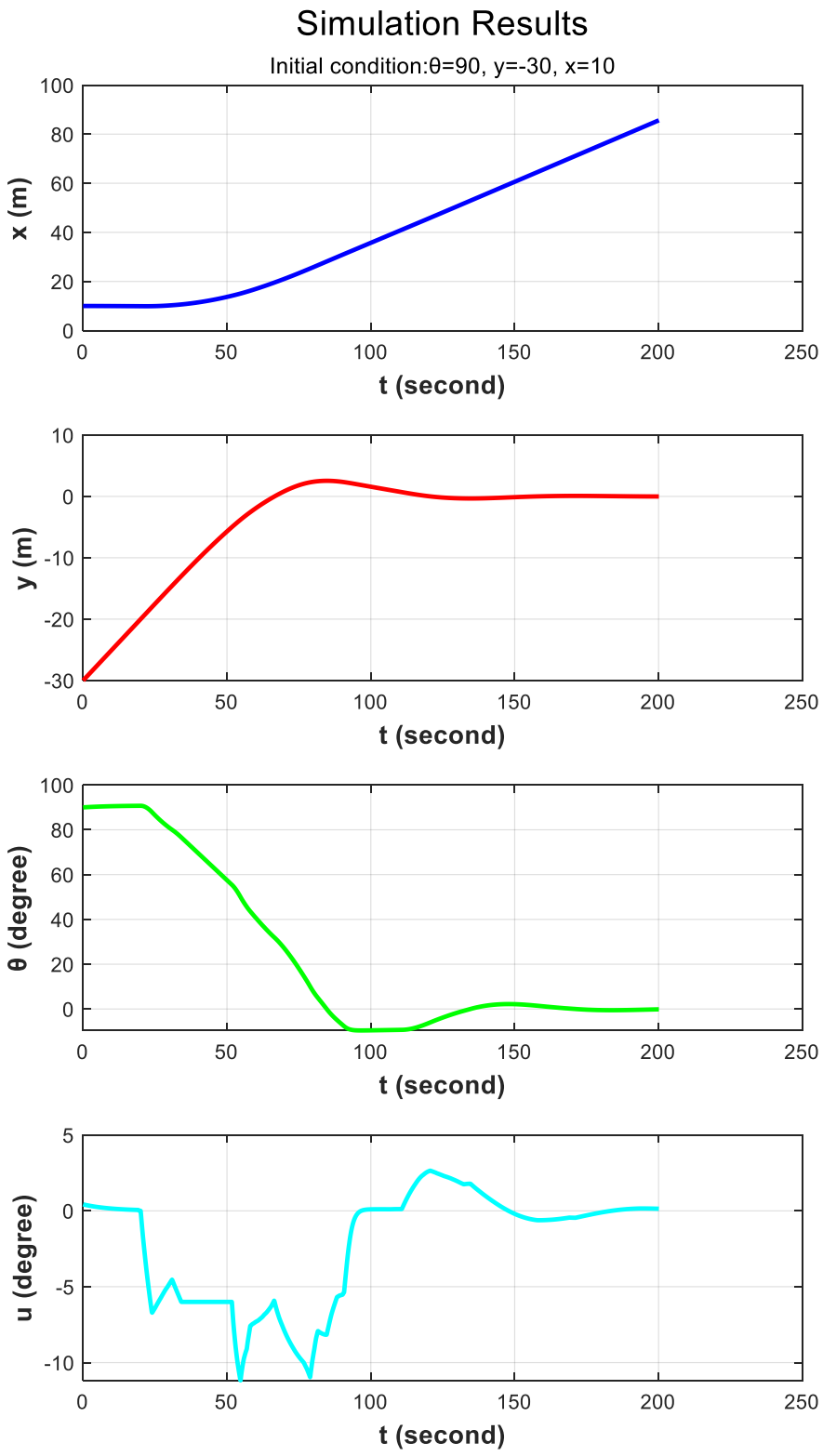

Simulation Results
Initial condition:θ=90, y=-30, x=10
x (m)
t (second)
y (m)
t (second)
θ (degree)
t (second)
u (degree)
t (second)

Figure 4

Case 3:

$$\theta = 220, y = 30, x = 40$$

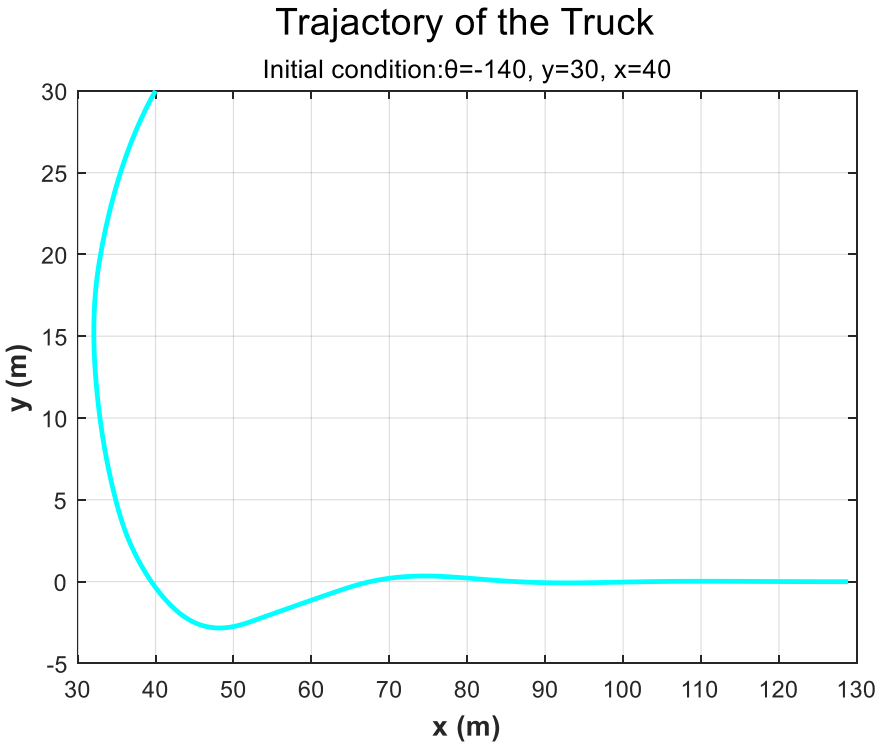


Figure 5

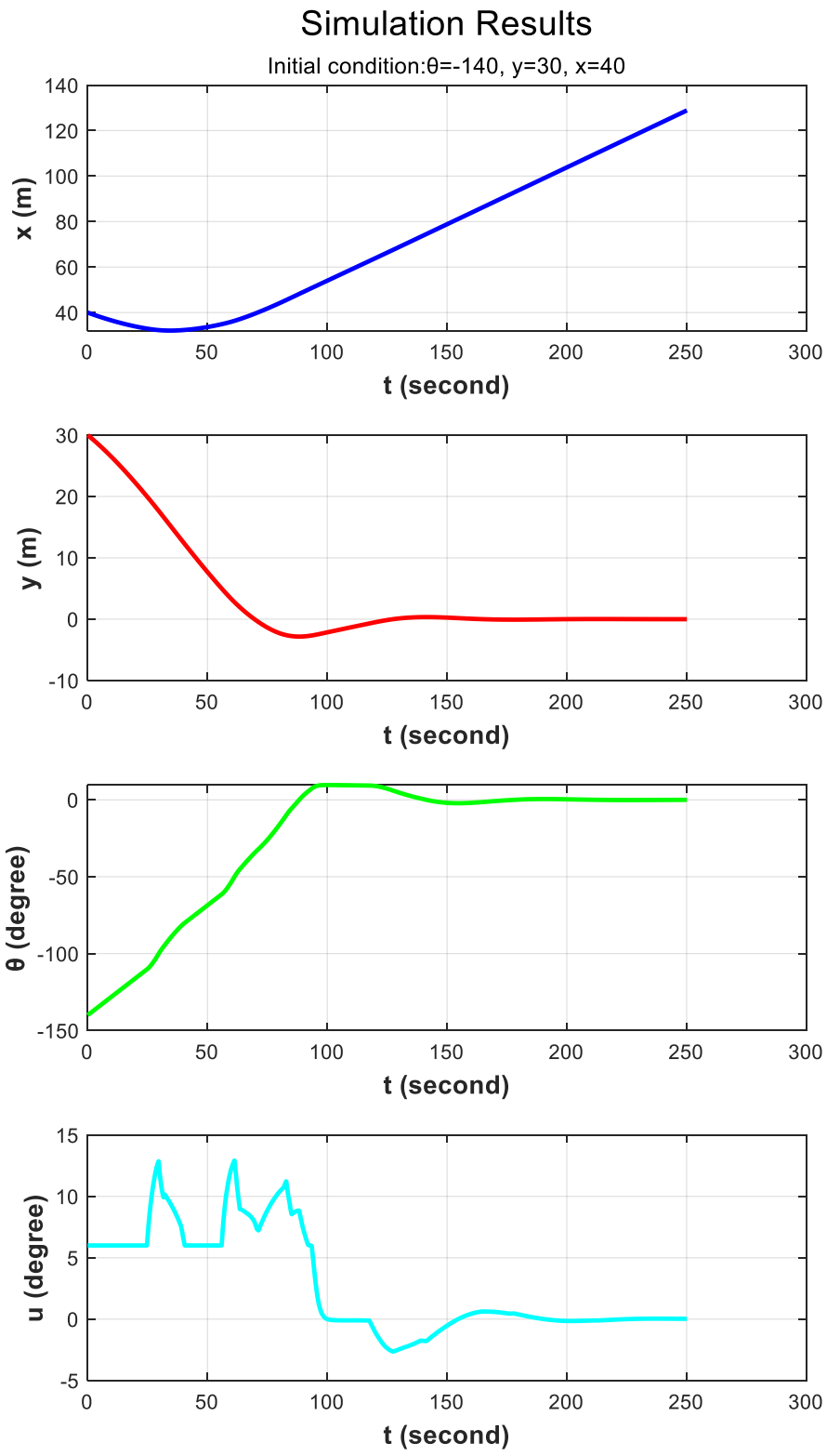

Simulation Results
Initial condition:θ=-140, y=30, x=40
x (m)
t (second)
y (m)
t (second)
θ (degree)
t (second)
u (degree)
t (second)

Figure 6

Case 4:

$$\theta = -10, y = 10, x = 50$$

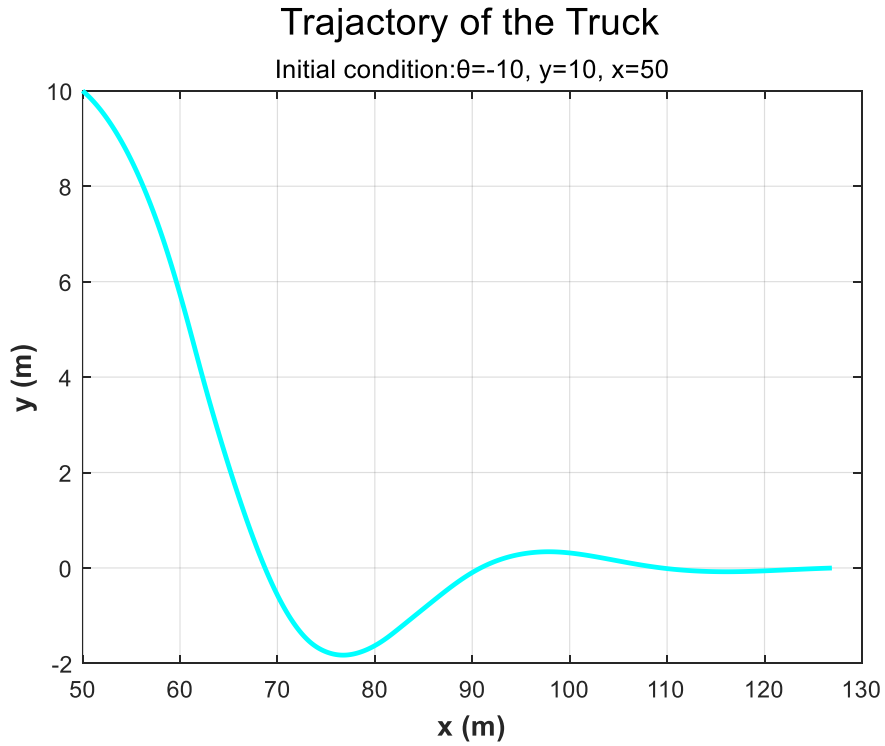


Figure 7

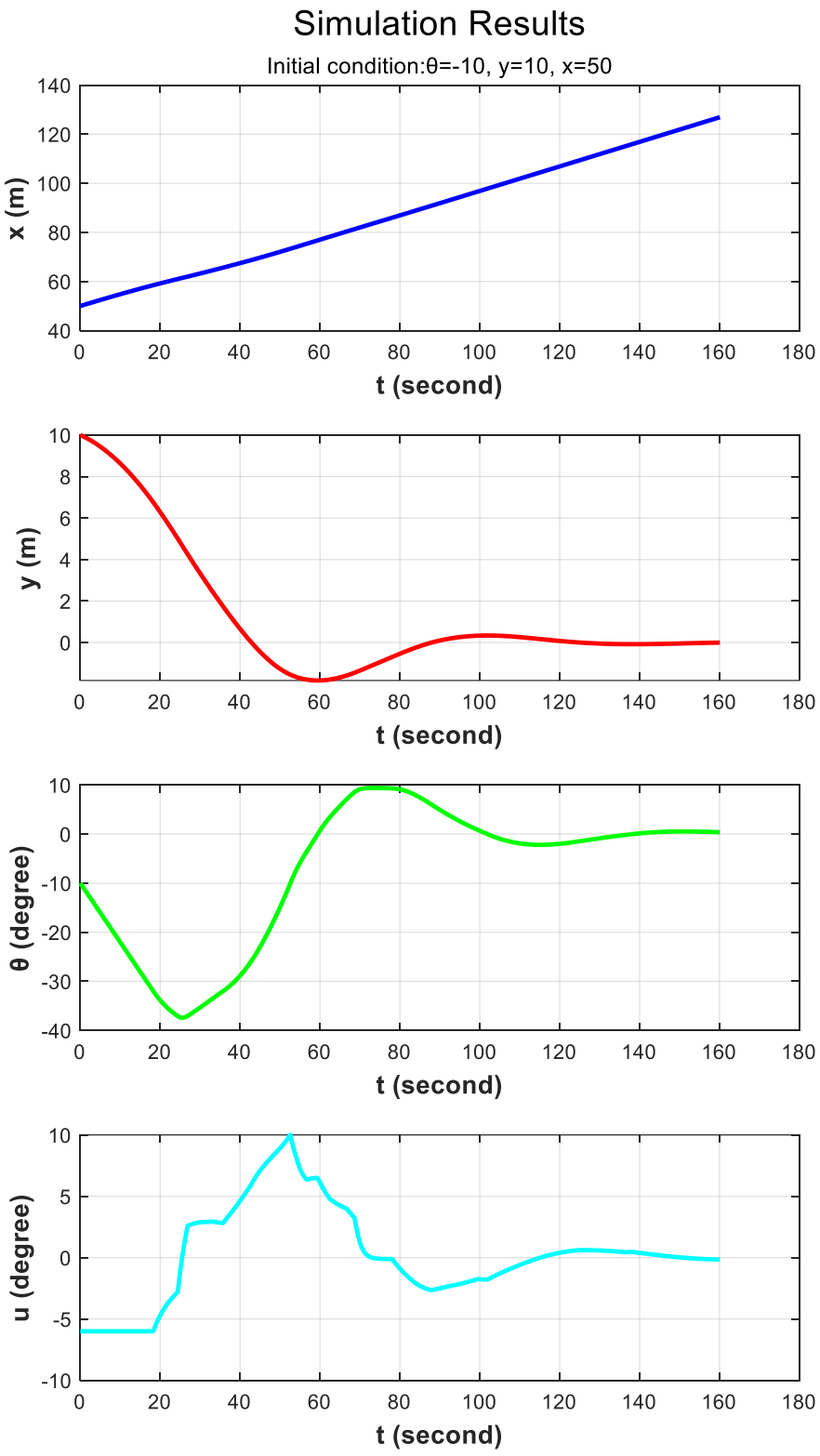
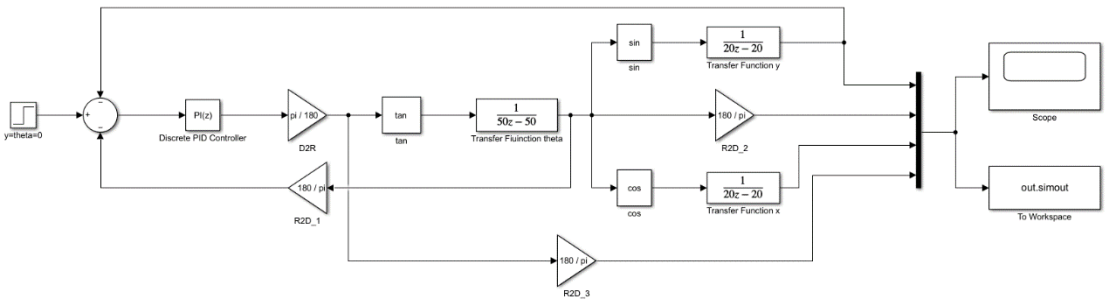

Simulation Results
Initial condition:θ=-10, y=10, x=50
x (m)
t (second)
y (m)
t (second)
θ (degree)
t (second)
u (degree)
t (second)

Figure 8

(b) Non-fuzzy logic controller

Here, I choose PID controller to reach $y = \theta = 0$.



Case 1:

$$\theta = 0, y = 40, x = 20$$

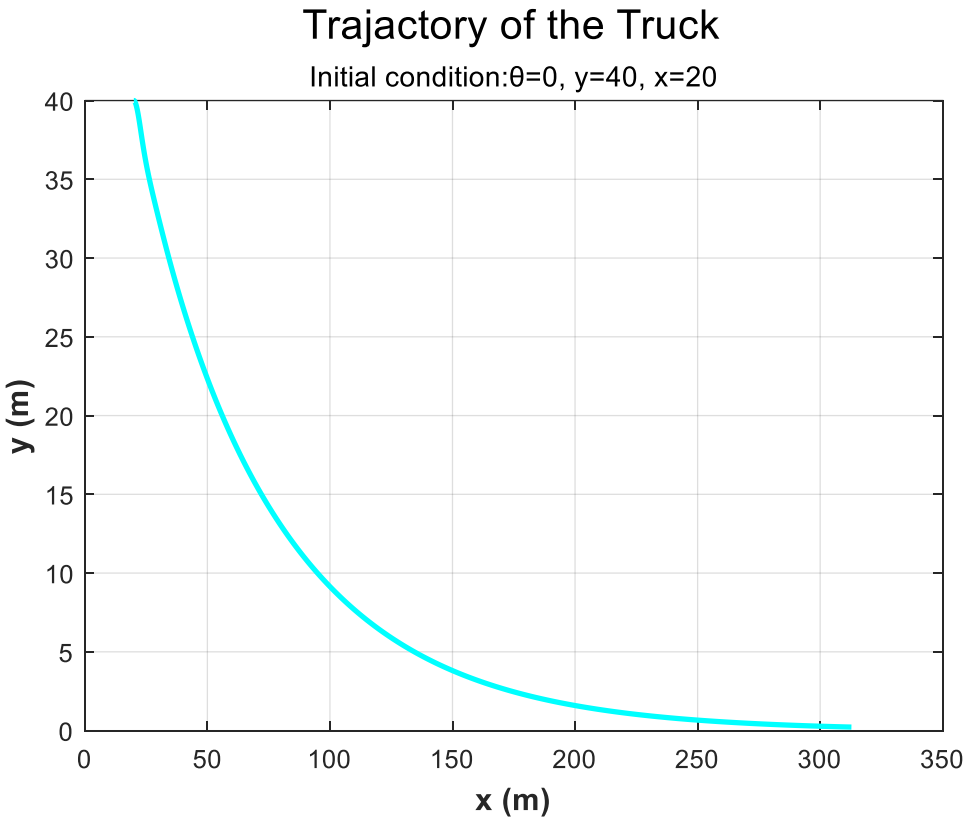


Figure 9

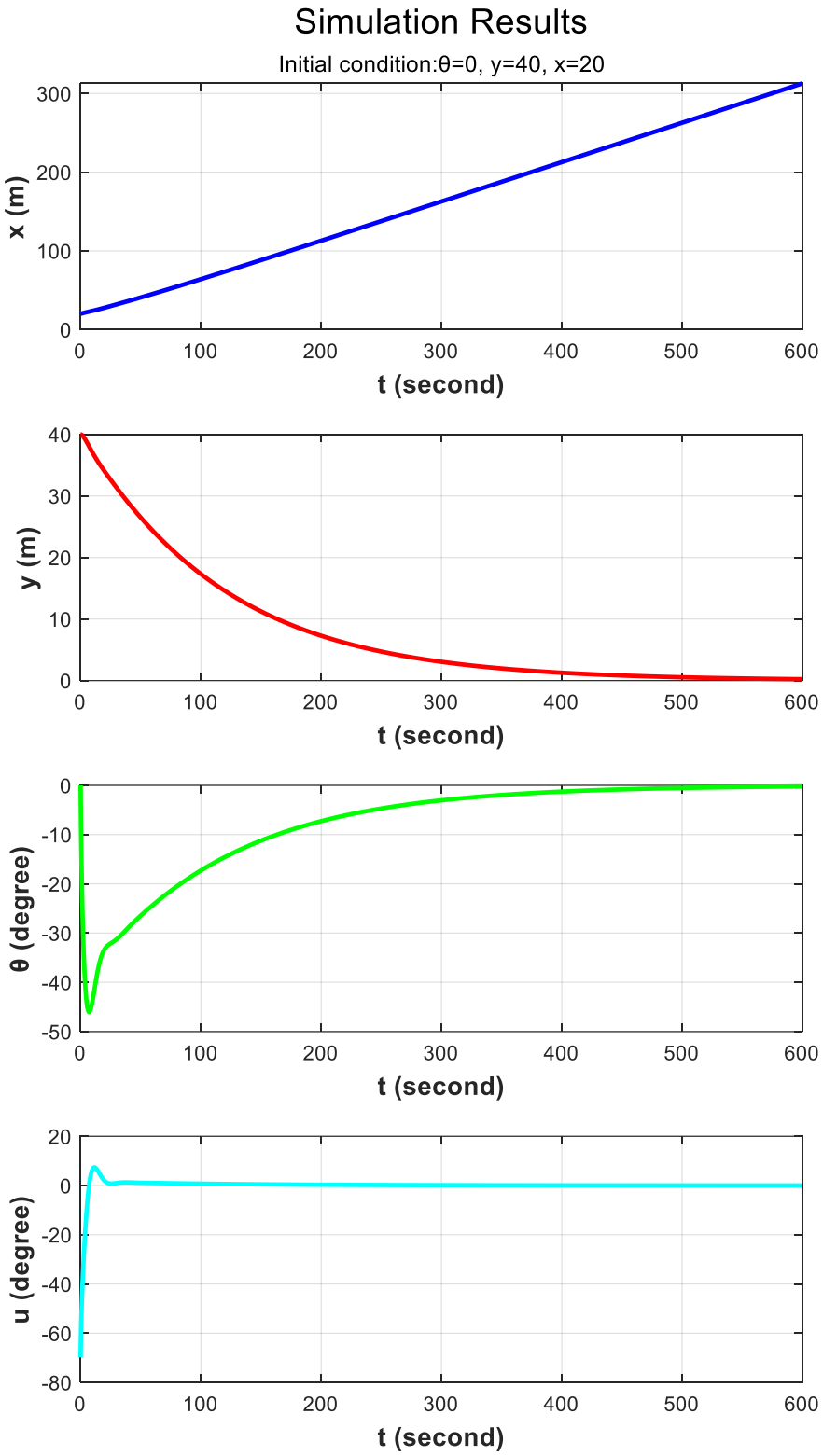

Simulation Results
Initial condition:θ=0, y=40, x=20
x (m)
t (second)
y (m)
t (second)
θ (degree)
t (second)
u (degree)
t (second)

Figure 10

Case 2:

$$\theta = 90, y = -30, x = 10$$

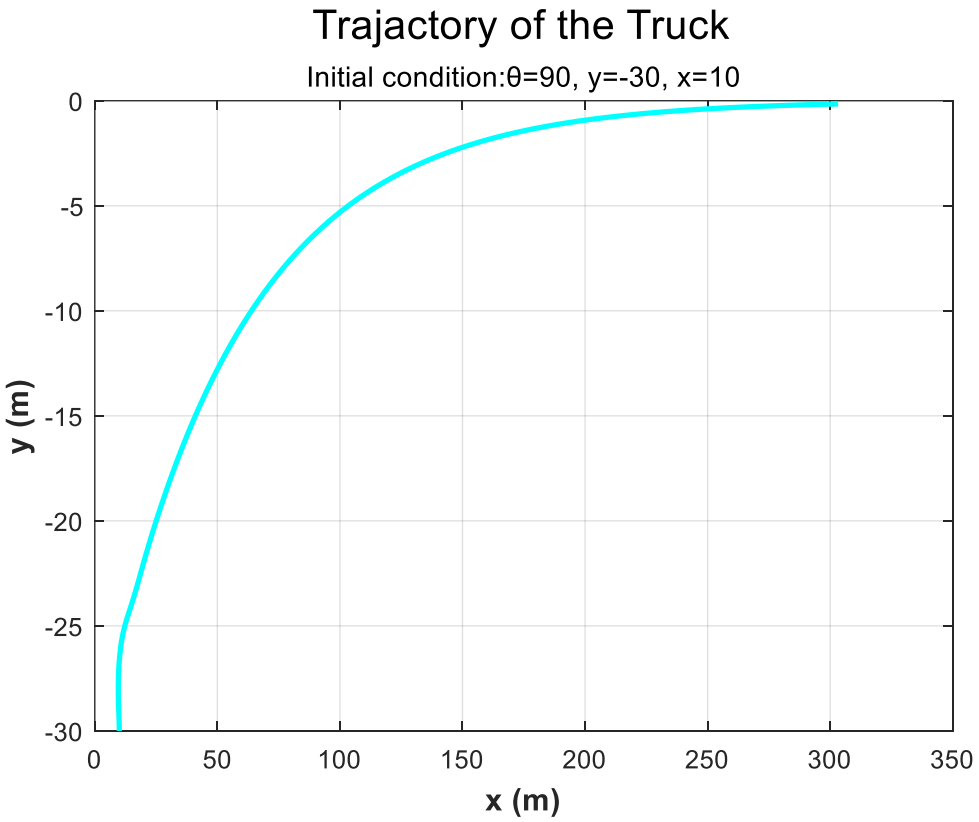


Figure 11

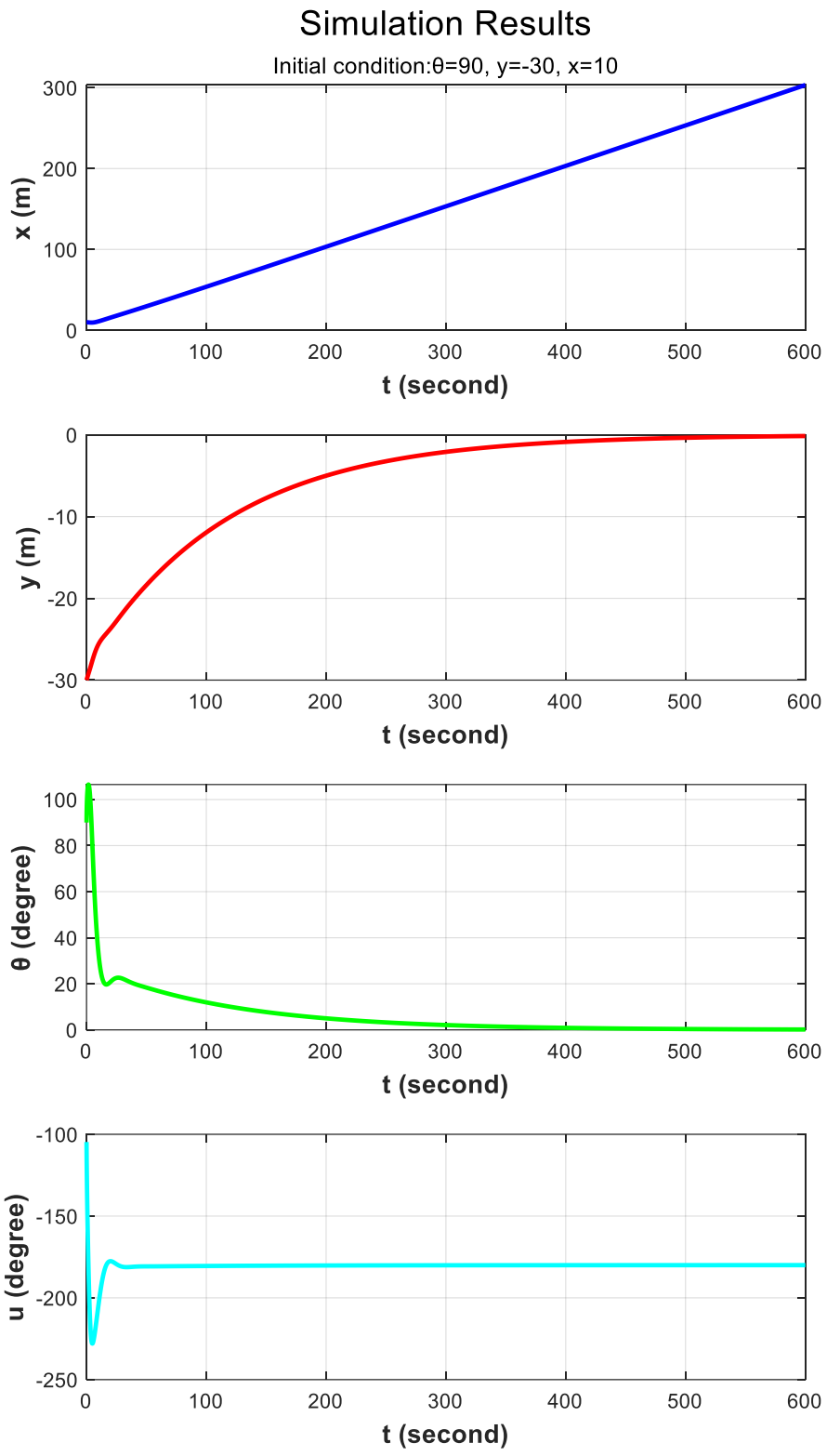

Simulation Results
Initial condition:θ=90, y=-30, x=10
x (m)
y (m)
θ (degree)
u (degree)
t (second)

Figure 12

Case 3:

$$\theta = 220, y = 30, x = 40$$

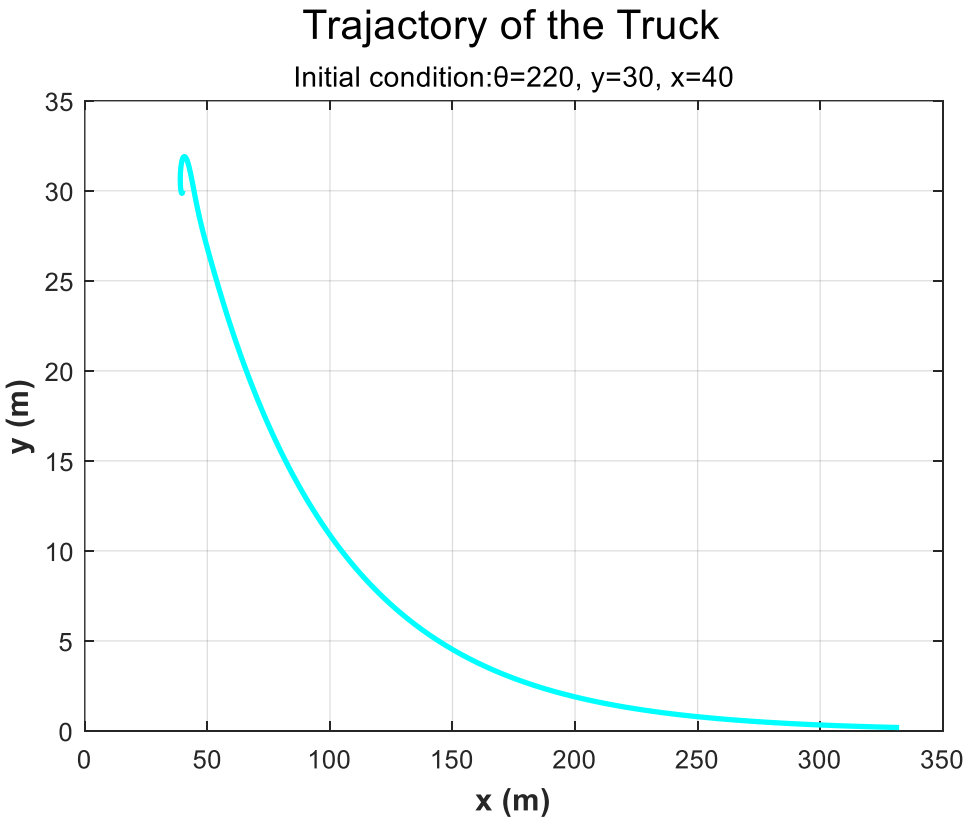


Figure 13

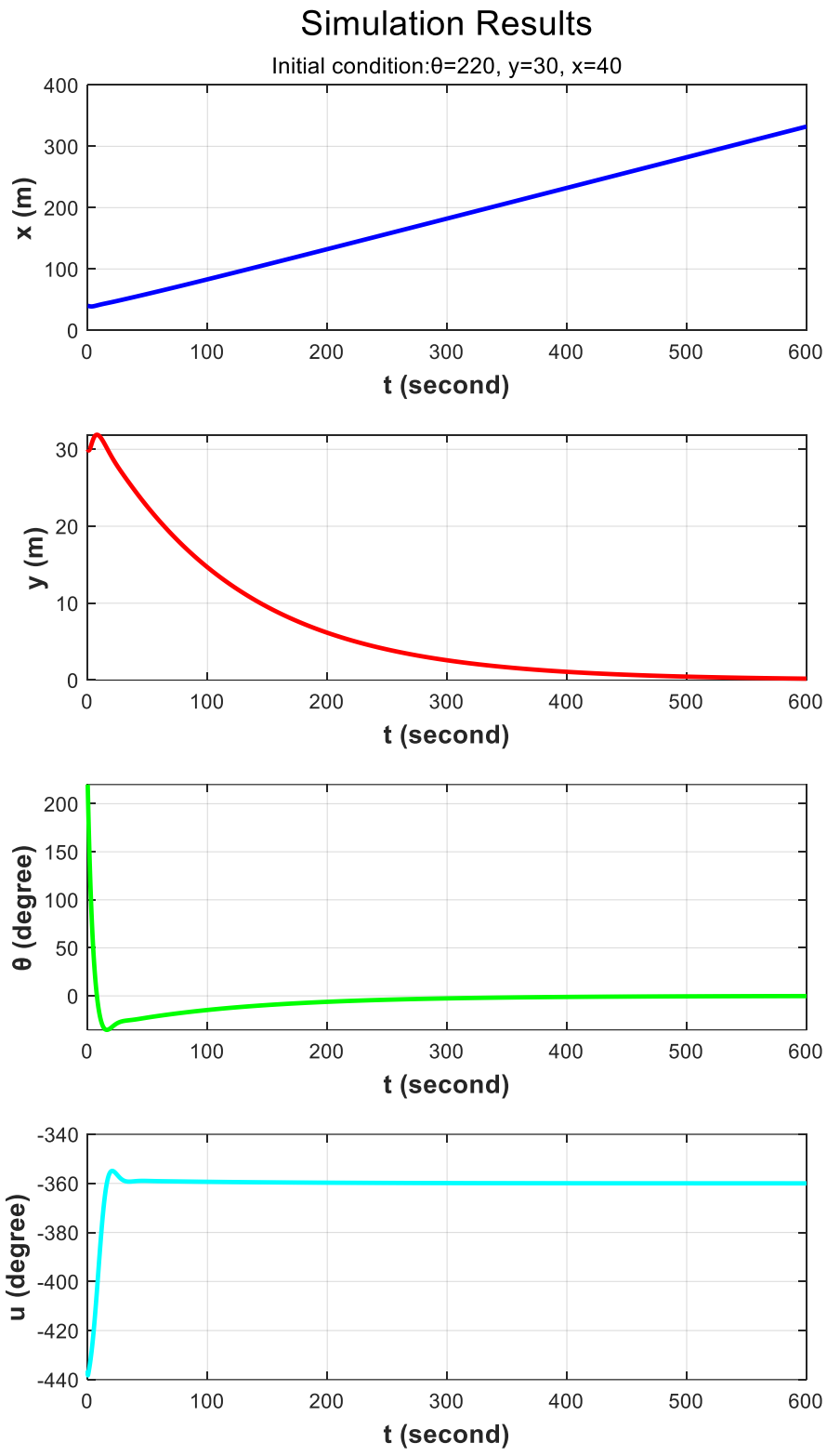

Simulation Results
Initial condition:θ=220, y=30, x=40
x (m)
t (second)
y (m)
t (second)
θ (degree)
t (second)
u (degree)
t (second)

Figure 14

Case 4:

$$\theta = -10, y = 10, x = 50$$

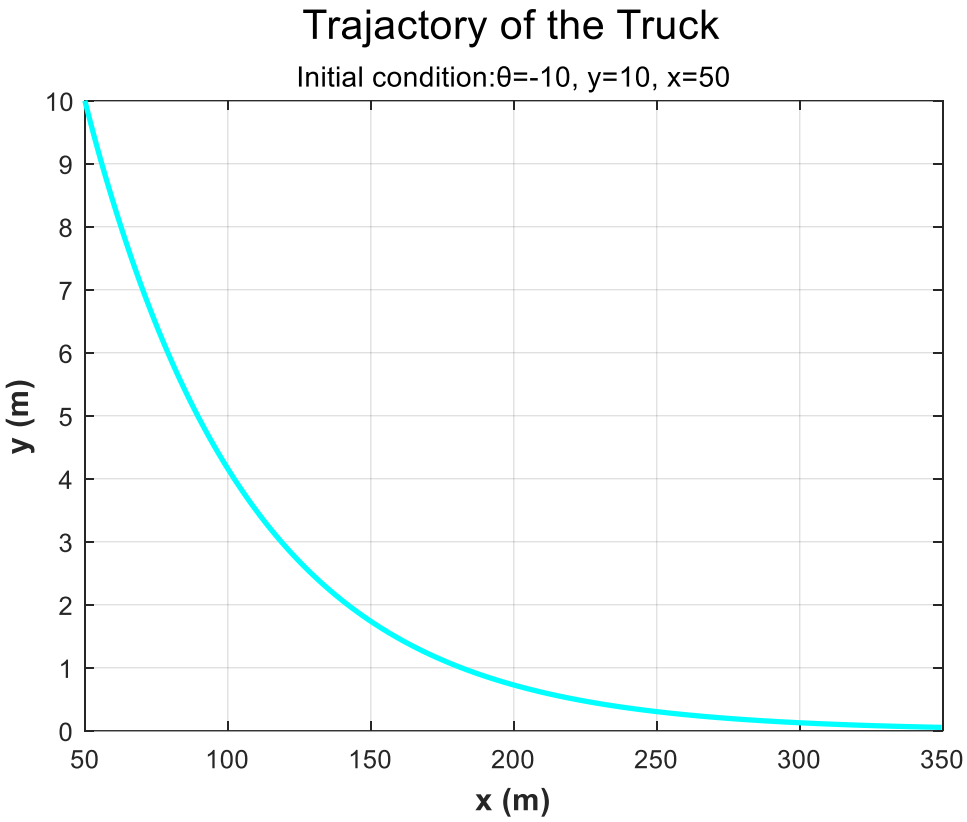


Figure 15

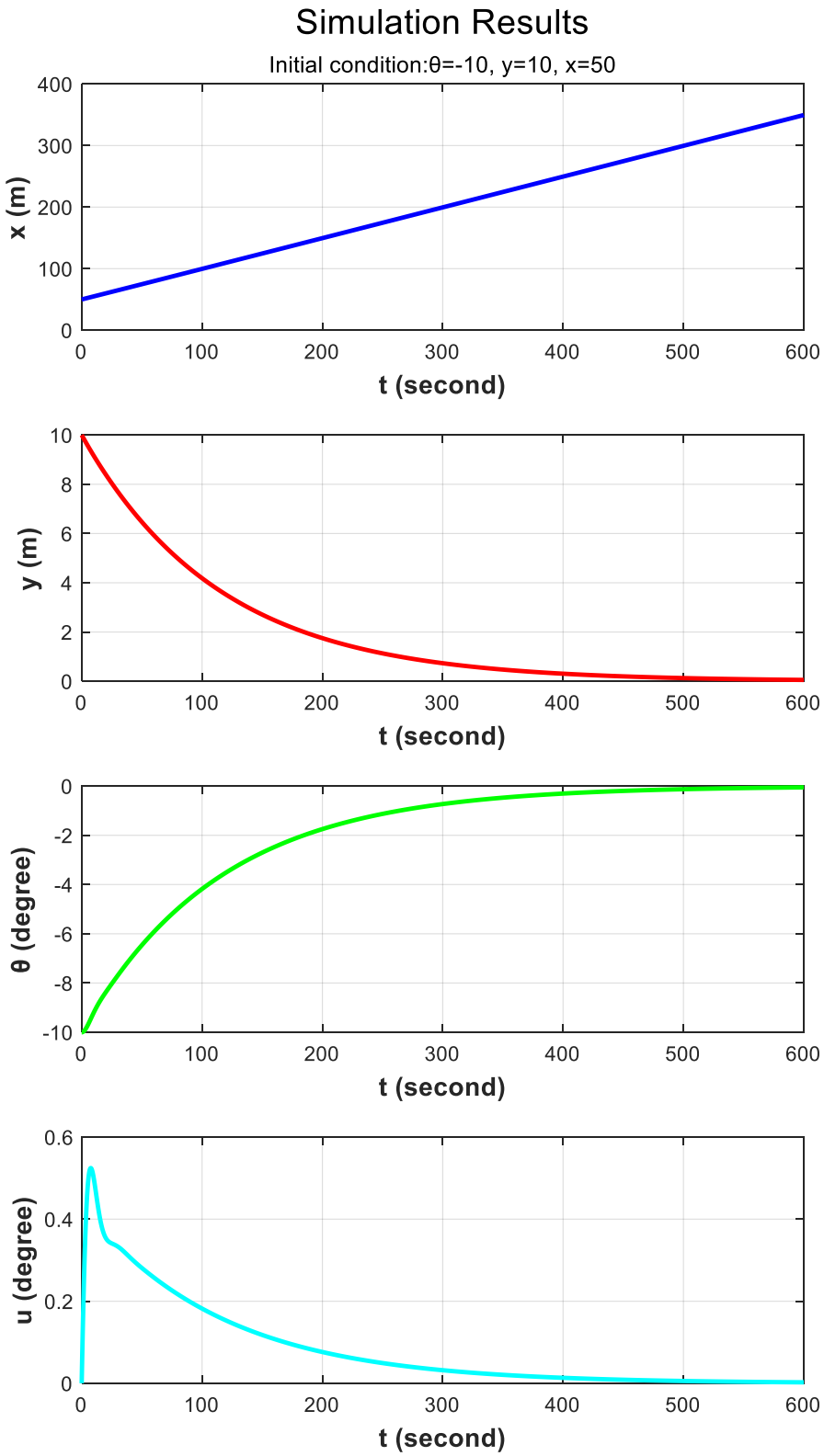

Simulation Results
Initial condition:θ=-10, y=10, x=50
x (m)
t (second)
y (m)
t (second)
θ (degree)
t (second)
u (degree)
t (second)

Figure 16

2. Clustering

(a) Use the cosine method in Equation (2.1) of Section 2.1 to generate the fuzzy tolerance relation R1.

Solution:

Matlab Code:

```
% Original matrix
x = [0.1, 0.0, 0.2, 0.8, 0.3, 0.0, 0.5, 0.6, 0.0, 0.1, 0.3, 0.1,
0.2, 0.2, 0.1, 0.2;
    0.7, 0.5, 0.2, 0.1, 0.0, 0.4, 0.0, 0.3, 0.5, 0.6, 0.2, 0.5,
0.0, 0.6, 0.7, 0.4;
    0.2, 0.5, 0.2, 0.0, 0.4, 0.0, 0.4, 0.0, 0.1, 0.0, 0.1, 0.4,
0.2, 0.1, 0.1, 0.2;
    0.0, 0.0, 0.4, 0.1, 0.3, 0.6, 0.1, 0.1, 0.4, 0.3, 0.4, 0.0,
0.6, 0.1, 0.1, 0.2];

x = x';

n = size(x, 1);
m = size(x, 2);

R1 = ones(n, n);

% Question(a): Compute R1
for i = 1 : n
    for j = 1 : n
        % Calculate Numerator and Denominator
        Numerator = 0;
        Den1 = 0;
        Den2 = 0;
        for k = 1 : m
            Numerator = Numerator + x(i, k) * x(j, k);
            Den1 = Den1 + x(i, k) * x(i, k);
            Den2 = Den2 + x(j, k) * x(j, k);
        end
        Numerator = abs(Numerator);
        Denominator = sqrt(Den1 * Den2);
        R1(i, j) = Numerator / Denominator;
    end
end
```

The fuzzy tolerance relation R_1 is shown in the Figure 17.

R1 =

1.0000	0.8660	0.5143	0.2513	0.2567	0.5284	0.2730	0.5417	0.7769	0.8628	0.4721	0.9239	0.1231	0.9659	0.9813	0.8744
0.8660	1.0000	0.5345	0.0870	0.4851	0.3922	0.4364	0.3128	0.6547	0.6255	0.3873	0.9820	0.2132	0.7638	0.7845	0.8018
0.5143	0.5345	1.0000	0.5118	0.8427	0.8386	0.6415	0.6130	0.8165	0.7245	0.9661	0.5832	0.9117	0.6415	0.5766	0.8571
0.2513	0.0870	0.5118	1.0000	0.5700	0.1707	0.7787	0.9437	0.1709	0.3085	0.6742	0.2469	0.4082	0.4368	0.2731	0.5118
0.2567	0.4851	0.8427	0.5700	1.0000	0.4281	0.8997	0.5310	0.4234	0.3034	0.7828	0.5028	0.8273	0.3440	0.2378	0.6482
0.5284	0.3922	0.8386	0.1707	0.4281	1.0000	0.1284	0.3680	0.9415	0.8588	0.8102	0.4280	0.7526	0.6419	0.6538	0.7338
0.2730	0.4364	0.6415	0.7787	0.8997	0.1284	1.0000	0.7053	0.1905	0.1820	0.6480	0.5000	0.5583	0.3571	0.2140	0.5832
0.5417	0.3128	0.6130	0.9437	0.5310	0.3680	0.7053	1.0000	0.4323	0.5870	0.7537	0.4778	0.4001	0.7053	0.5725	0.7245
0.7769	0.6547	0.8165	0.1709	0.4234	0.9415	0.1905	0.4323	1.0000	0.9555	0.7606	0.6905	0.6048	0.8333	0.8559	0.8748
0.8628	0.6255	0.7245	0.3085	0.3034	0.8588	0.1820	0.5870	0.9555	1.0000	0.7268	0.7053	0.4446	0.9328	0.9405	0.8916
0.4721	0.3873	0.9661	0.6742	0.7828	0.8102	0.6480	0.7537	0.7606	0.7268	1.0000	0.4789	0.8808	0.6480	0.5570	0.8281
0.9239	0.9820	0.5832	0.2469	0.5028	0.4280	0.5000	0.4778	0.6905	0.7053	0.4789	1.0000	0.2326	0.8571	0.8559	0.8748
0.1231	0.2132	0.9117	0.4082	0.8273	0.7526	0.5583	0.4001	0.6048	0.4446	0.8808	0.2326	1.0000	0.2791	0.2091	0.5698
0.9659	0.7638	0.6415	0.4368	0.3440	0.6419	0.3571	0.7053	0.8333	0.9328	0.6480	0.8571	0.2791	1.0000	0.9843	0.9331
0.9813	0.7845	0.5766	0.2731	0.2378	0.6538	0.2140	0.5725	0.8559	0.9405	0.5570	0.8559	0.2091	0.9843	1.0000	0.8910
0.8744	0.8018	0.8571	0.5118	0.6482	0.7338	0.5832	0.7245	0.8748	0.8916	0.8281	0.8748	0.5698	0.9331	0.8910	1.0000

Figure 17

(b) Use Algorithm 2.1 in Section 2.2 to transform R1 into a fuzzy equivalence relation R.

Solution:

Matlab Code:

```
% Question(b): Transform R1 into a fuzzy equivalence relation R
R = composition(R1);
while ~isequal(R, R1)
    R1 = R;
    R = composition(R1);
end
```

```
function [R] = composition(R1)
    R1_len = size(R1, 1);
    R = ones(R1_len, R1_len);
    for i = 1 : R1_len
        row_vec = R1(i, :);
        for j = 1 : R1_len
            col_vec = R1(j, :)';
            new_vec = zeros(R1_len, 1);
            for k = 1 : R1_len
                new_vec(k) = min(row_vec(k), col_vec(k));
            end
            R(i, j) = max(new_vec);
        end
    end
end
```

The fuzzy equivalence relation R is shown in Figure 18.

R =

1.0000	0.9239	0.8571	0.7787	0.8427	0.9405	0.8427	0.7787	0.9405	0.9405	0.8571	0.9239	0.8571	0.9813	0.9813	0.9331
0.9239	1.0000	0.8571	0.7787	0.8427	0.9239	0.8427	0.7787	0.9239	0.9239	0.8571	0.9820	0.8571	0.9239	0.9239	0.9239
0.8571	0.8571	1.0000	0.7787	0.8427	0.8571	0.8427	0.7787	0.8571	0.8571	0.9661	0.8571	0.9117	0.8571	0.8571	0.8571
0.7787	0.7787	0.7787	1.0000	0.7787	0.7787	0.7787	0.9437	0.7787	0.7787	0.7787	0.7787	0.7787	0.7787	0.7787	0.7787
0.8427	0.8427	0.8427	0.7787	1.0000	0.8427	0.8997	0.7787	0.8427	0.8427	0.8427	0.8427	0.8427	0.8427	0.8427	0.8427
0.9405	0.9239	0.8571	0.7787	0.8427	1.0000	0.8427	0.7787	0.9415	0.9415	0.8571	0.9239	0.8571	0.9405	0.9405	0.9331
0.8427	0.8427	0.8427	0.7787	0.8997	0.8427	1.0000	0.7787	0.8427	0.8427	0.8427	0.8427	0.8427	0.8427	0.8427	0.8427
0.7787	0.7787	0.7787	0.9437	0.7787	0.7787	0.7787	1.0000	0.7787	0.7787	0.7787	0.7787	0.7787	0.7787	0.7787	0.7787
0.9405	0.9239	0.8571	0.7787	0.8427	0.9415	0.8427	0.7787	1.0000	0.9555	0.8571	0.9239	0.8571	0.9405	0.9405	0.9331
0.9405	0.9239	0.8571	0.7787	0.8427	0.9415	0.8427	0.7787	0.9555	1.0000	0.8571	0.9239	0.8571	0.9405	0.9405	0.9331
0.8571	0.8571	0.9661	0.7787	0.8427	0.8571	0.8427	0.7787	0.8571	0.8571	1.0000	0.8571	0.9117	0.8571	0.8571	0.8571
0.9239	0.9820	0.8571	0.7787	0.8427	0.9239	0.8427	0.7787	0.9239	0.9239	0.8571	1.0000	0.8571	0.9239	0.9239	0.9239
0.8571	0.8571	0.9117	0.7787	0.8427	0.8571	0.8427	0.7787	0.8571	0.8571	0.9117	0.8571	1.0000	0.8571	0.8571	0.8571
0.9813	0.9239	0.8571	0.7787	0.8427	0.9405	0.8427	0.7787	0.9405	0.9405	0.8571	0.9239	0.8571	1.0000	0.9843	0.9331
0.9813	0.9239	0.8571	0.7787	0.8427	0.9405	0.8427	0.7787	0.9405	0.9405	0.8571	0.9239	0.8571	0.9843	1.0000	0.9331
0.9331	0.9239	0.8571	0.7787	0.8427	0.9331	0.8427	0.7787	0.9331	0.9331	0.8571	0.9239	0.8571	0.9331	0.9331	1.0000

Figure 18

(c) Generate the α-cut R_α in Section 2.3 to provide classification classes for α-cut level = 0.4 and α-cut level = 0.8?

Solution:

Matlab Code:

```
%% Question(c): alpha = 0.4, 0.8
% rules: >=0.4 =1, <0.4 =0;>=0.8 =1, <0.8 =0
R_4 = zeros(n, n);
R_8 = zeros(n, n);
for i = 1 : n
    for j = 1 : n
        if (R(i, j) >= 0.4)
            R_4(i, j) = 1;
        end
        if (R(i, j) >= 0.8)
            R_8(i, j) = 1;
        end
    end
end
```

The 4-cut R_4 and 8-cut R_8 are shown in Figure 19 and Figure 20.

R_4 =

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 19

R_8 =

1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1

Figure 20

The classification is shown in Table 1.

α-cut level	Classification Results
4	$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}\}$
8	$\{x_1, x_2, x_3, x_5, x_6, x_7, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}\}$,$\{x_4, x_8\}$

Table 1

(d) If we want to classify the flooding for the whole county into 3 classes - Red, Yellow and Green. What should be the appropriate the α-cut value?

Solution:

Matlab Code:

```
%% Question(d): 3 classes
target_alpha = [];
for alpha = 0.8 : 0.0001 : 1.0
    R_test = R;
    for i = 1 : n
        for j = 1 : n
            if R(i, j) >= alpha
                R_test(i, j) = 1;
            end
            if R(i, j) < alpha
                R_test(i, j) = 0;
            end
        end
    end
    % Evaluation
    R_test =  unique(R_test, 'rows');
    class_num = size(R_test, 1);
    if class_num == 3
       fprintf("Proper alpha %f\n", alpha);
       target_alpha = [target_alpha alpha];
    end
end
if size(target_alpha, 2) > 1
    min_alpha = min(target_alpha);
    max_alpha = max(target_alpha);
end
```

When there are 3 classes, the α-cut value is from $0.8427 \sim 0.8571$.