Generating Fuzzy Rules By Learning from Examples

Li-Xin Wang and Jerry M. Mende

CONTENT

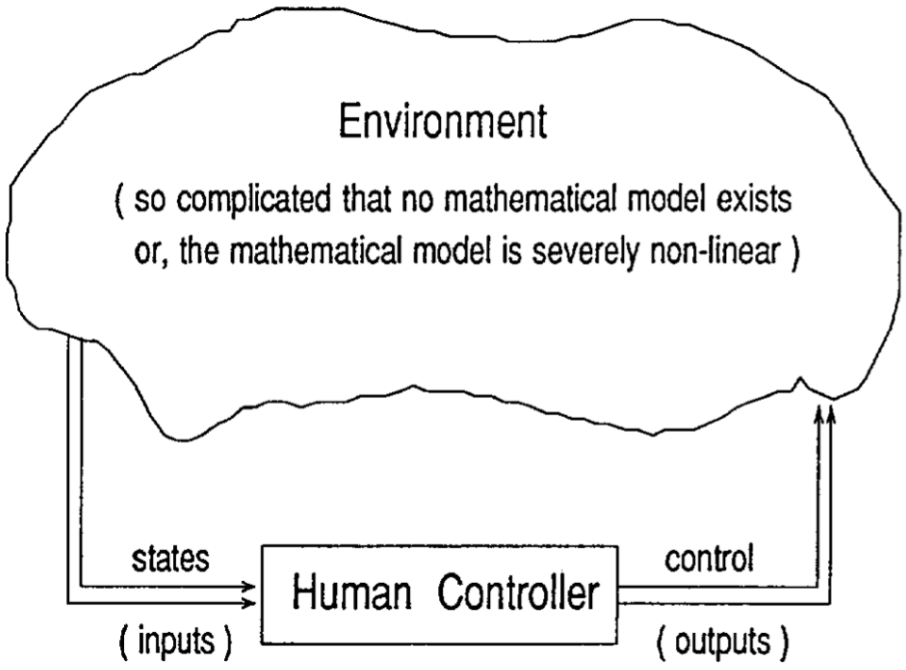
1. INTRODUCTION

This paper provide the general method to generate fuzzy rule from numerical data. This method combine both numerical and linguistic information into a common framework, which gives a fuzzy rule base.

We have four sections in this paper. In Section 2, we propose a five step procedure for generating fuzzy rules from numerical data pairs. In Section 3, we prove that the resulting mapping is capable of approximating any nonlinear continuous function on a compact set to arbitrary accuracy. In Section IV, we are going to deal with the application of our new method.

Section -1)

This section deals with the advantage of combining both numerical and linguistic information into a common framework to generate fuzzy rule base. This fuzzy rule base can be applied to replace human controller as shown in below fig.



Section -2)

In this section we are going to generate fuzzy rule from numerical data by following below steps.

- Division of input and output spaces of given numerical data into fuzzy.
- Generation of fuzzy rules from given data.
- Assignment of degree to each of the generated rule

- Create a combined fuzzy rule base based on both the generated rules and linguistic rules of human expert.
- Determination of mapping from input space to output space based on the combined fuzzy rule base using a defuzzifying procedure.

2. GENERATING FUZZY RULES FROM NUMERICAL DATA.

Below input-output data pairs are provided to us.

($x1$ (1) , $x2$ (1) , y (1)), ($x1$ (2) , $x2$ (2) , y (2)), … … ….

The task is to generate a set of fuzzy rules from the desired input-output pairs and use these fuzzy rules to determine the mapping f : ($x1$, $x2$) → y

Our approach consists of following five steps for generating fuzzy rule

a. Divide input and output space into fuzzy regions Suppose that domain interval of $x1$, $x2$ and y are

   $x1$ → [$x1$ −, $x1$ +] , $x2$ → [$x2$ −, $x2$ +] and y → [y−, y+]

   Where the domain interval of variable means that most probably variable will lie in this interval. Divide each domain interval into 2N+1 region and assign each region a fuzzy membership function. Then assign the symbolic representation for each membership function.

b. Generating fuzzy rule from given data pairs

   First, we have to determine the degree of given inputs $x1$ (i) , $x2$ (i) and output y(i) in different regions. Second, assign a given inputs or outputs to the region with maximum degree.
   In system identification and modeling problems there are lots of data pairs, and each data pair generates one rule, it is highly probable that there will be some conflicting rules, i.e., rules that have the same IF part but a different THEN part. One way to resolve this conflict is to assign a degree to each rule generated from data pairs, and accept only the rule from a conflict group that has maximum degree. In this way not only is the conflict problem resolved, but also the number of rules is alleviate. In this paper, the degree of rule is defined as product of degree of data pairs that generate the rule.

   In practical situation, we often have some a priori information about the data pairs. For example, if we let an expert check given data pairs, the expert may suggest that some are very useful and crucial, but others are very unlikely and may be caused just by measurement errors.

c. Assign a Degree to Each Rule

We can therefore assign a degree to each data pair that represents our belief of its usefulness. In this sense, the data pairs constitute a fuzzy set, i.e., the fuzzy set is defined as the useful measurements; a data pair belongs to this set to a degree assigned by a human expert. Hence based on the expert opinion for good data we assign higher degrees, and for bad data we assign lower degrees. In this way, human experience about the data is used in a common base as other information.
For example-

Rule 1 has degree, D(Rule1) = mB1 (x1)mS1 (x2)mCE (y)= 0.8 * 0.7 * 0.9 = 0.504

Rule 2 has degree D(Rule2) = mB1 (x1)mCE (x2)mB1 (y)= 0.6 * 1* 0.7 = 0.42

d. Combination of fuzzy rule base
The form of combined fuzzy rule base is shown in below figure.

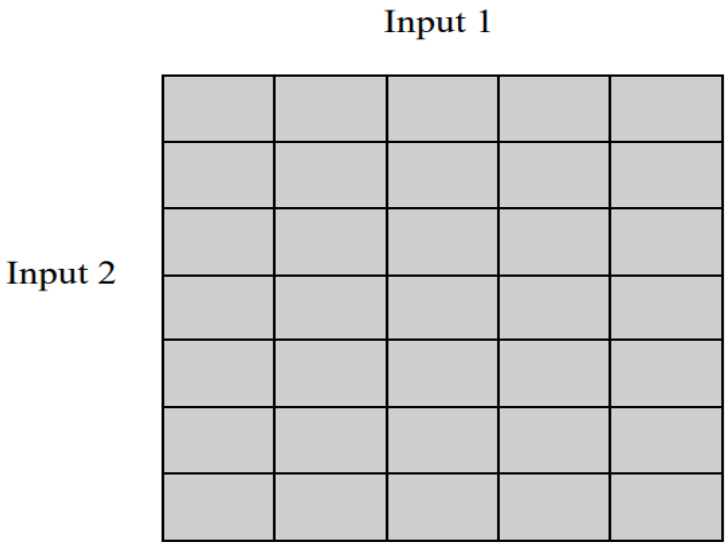


Fig. 2: Combined fuzzy rule base

A combined fuzzy rule base is assigned rules from generated numerical data or linguistic rules. If there is more than one rule in one box of the fuzzy rule base, then use the rule that has maximum degree. In this way, both numerical and linguistic information are codified into a common framework-the combined fuzzy rule base.

e. Generation of mapping from inputs to output

To generate a mapping from inputs to output, defuzzification and aggregation are used. Defuzzification is used to convert fuzzy output into crisp output for each rule. The crisp output of each rule is then aggregated into a final crisp output with help of output degrees for each rule.

Defuzzification:

a. Centroid defuzzification method is used. The centroid of the output fuzzy subspace of a rule is defined as the minimum of the core of that fuzzy subspace. Therefore, the centroid is the minimum value of the output values in the output fuzzy subspace of the rule at which the membership value is 1.

b. Other methods of defuzzification may also be used according to the nature of problem that is being dealt with. The centroid can be taken as the minimum value corresponding to the maximum membership degree. For limited range practical data, this proves computationally efficient & more effective method.

Aggregation:

Defuzzification gives crisp output for each rule. To get overall crisp output, these defuzzified outputs are aggregated. For aggregation, output degree for each rule is calculated to represent the extent to which the rule contributes towards the final output.

3. Application to Truck backer-Upper Control

The control for Backing a truck to a loading dock is a nonlinear control problem for which no traditional control system design methods exist. As mentioned in the paper Nguyen and Widrow develop a neural network controller for the truck backer-upper problem and Kong and Kosko propose a fuzzy control strategy for the same problem.

The numerical/neural network controller only uses numerical data, and cannot utilize linguistic rules determined from expert drivers; on the other hand, the fuzzy controller only uses linguistic rules, and cannot utilize sampled data.

In the given example, the truck position is exactly determined by the three state variables, two input variable and one output variable
Whose inputs are ϕ E [-90°,2700] and x E [0,20], and whose output is θ E [-4O°,4O°], such that the final states will be (Xf,ϕf) = (10,9O°).

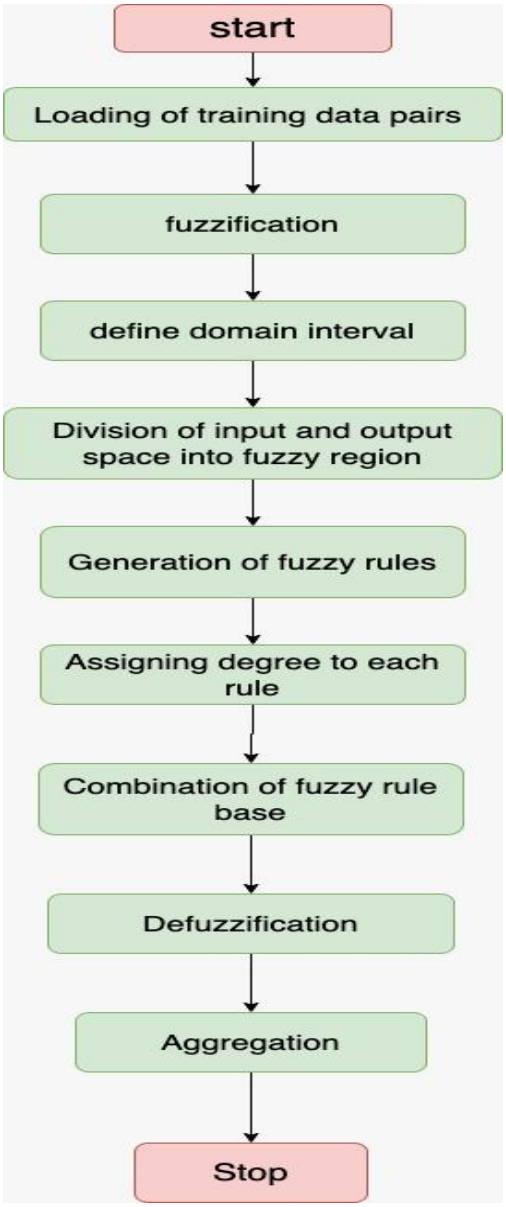
For finding out the next position of truck, following equations of dynamic state representation of truck are used and this is approximated kinematics:

$$x(t+1) = x(t) + \cos[\emptyset(t) + \theta(t)] + sin[\,\theta(t)]sin[\emptyset(t)]$$

$$y(t+1) = y(t) + \sin[\emptyset(t) + \theta(t)] - sin[\,\theta(t)]cos[\emptyset(t)]$$

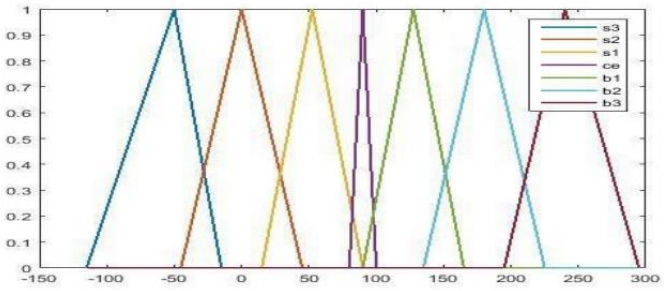
$$\emptyset(t+1) = \emptyset(t) - \sin{-1}\ [\ 2\sin\{\theta(t)\}/b]$$

The flowchart for Truck backer-Upper Control is shown below

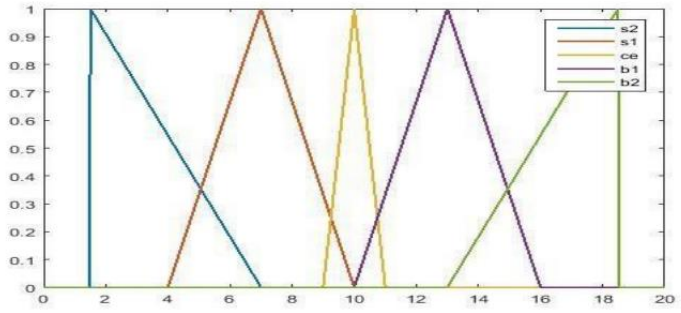


4. RESULTS OF TRUCK BACKER-UPPER PROBLEM

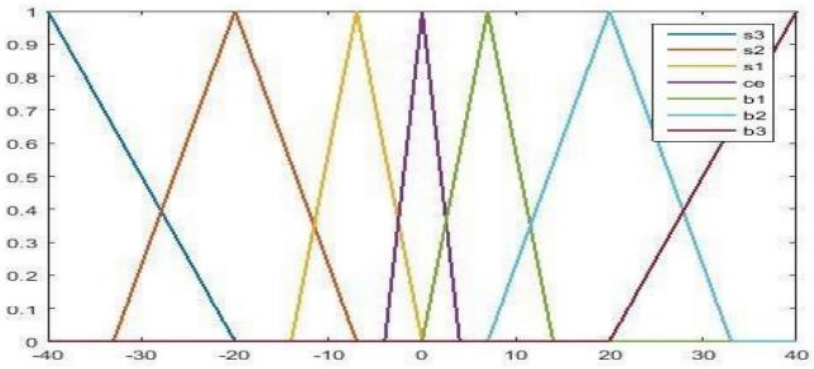
We are going to divide our inputs into fuzzy regions. The regions used are shown in below figure.



(a)



(b)



(c)

Fuzzy region (a) Input x (b) Input phi (c) Output theta

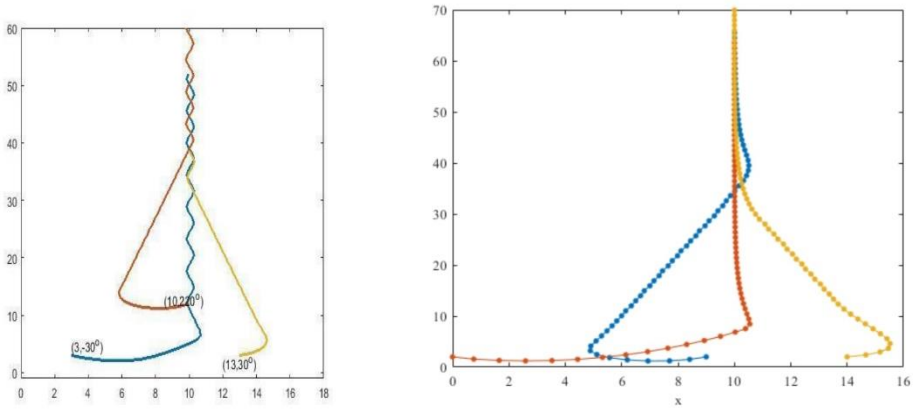
Numerical Control and simulation result:

For this simulation purpose, we used the 24 sets of the fuzzy rules as generated from the data after removing conflicting rule which are mentioned in the below Table

Input: x

Input: Phi	***S2***	***S1***	***CE***	***B1***	***B2***
S3	S2	S3			
S2	S2	S3	S3	S3	
S1	B1	S1		S3	S2
CE	B2	B2		S2	S2
B1	B2	B3		B1	S1
B2		B3	B3	B3	B2
B3				B3	B2

Truck trajectory using numerical data only is shown below




```
%% Step 1: Divide the input and output spaces into fuzzy regions.
% Divide each domain interval into 2N+1 regions
% m_x, m_phi and m_theta

x = 0:0.1:20; % Universe of discourse of input x.
M_x(:,1) = trapmf(x,[0 0 1.5 7]);
M_x(:,2) = trimf(x,[4 7 10]);
M_x(:,3) = trimf(x,[9 10 11]);
M_x(:,4) = trimf(x,[10 13 16]);
M_x(:,5) = trapmf(x,[13 18.5 20 20]);

theta = -40:0.4:40; % Universe of discourse of output theta.
M_theta(:,1) = trimf(theta,[-40 -40 -20]);
M_theta(:,2) = trimf(theta,[-33 -20 -7]);
M_theta(:,3) = trimf(theta,[-14 -7 0]);
M_theta(:,4) = trimf(theta,[-4 0 4]);
```

```
M_theta(:,5) = trimf(theta,[0 7 14]);
M_theta(:,6) = trimf(theta,[7 20 33]);
M_theta(:,7) = trimf(theta,[20 40 40]);

phi = -115:0.1:295; % Universe of discourse of input phi.
M_phi(:,1) = trimf(phi,[-115 -65 -15]);
M_phi(:,2) = trimf(phi,[-45 0 45]);
M_phi(:,3) = trimf(phi,[15 52.5 90]);
M_phi(:,4) = trimf(phi,[80 90 100]);
M_phi(:,5) = trimf(phi,[90 127.5 165]);
M_phi(:,6) = trimf(phi,[135 180 225]);
M_phi(:,7) = trimf(phi,[195 245 295]);



%% Step 2: Generate Fuzzy Rules from Given Data Pairs.
% Calculate the Degree for given datasets

% Input x
MU_x_1 = trapmf(tb(:,1),[0 0 1.5 7]);
MU_x_2 = trimf(tb(:,1),[4 7 10]);
MU_x_3 = trimf(tb(:,1),[9 10 11]);
MU_x_4 = trimf(tb(:,1),[10 13 16]);
MU_x_5 = trapmf(tb(:,1),[13 18.5 20 20]);
MU_x = [MU_x_1 MU_x_2 MU_x_3 MU_x_4 MU_x_5];

%Output theta
MU_theta_1 = trimf(tb(:,3),[-40 -40 -20]);
MU_theta_2 = trimf(tb(:,3),[-33 -20 -7]);
MU_theta_3 = trimf(tb(:,3),[-14 -7 0]);
MU_theta_4 = trimf(tb(:,3),[-4 0 4]);
MU_theta_5 = trimf(tb(:,3),[0 7 14]);
MU_theta_6 = trimf(tb(:,3),[7 20 33]);
MU_theta_7 = trimf(tb(:,3),[20 40 40]);
MU_theta = [MU_theta_1 MU_theta_2 MU_theta_3 MU_theta_4 MU_theta_5 MU_theta_6
MU_theta_7];

% Input phi
MU_phi_1 = trimf(tb(:,2),[-115 -65 -15]);
MU_phi_2 = trimf(tb(:,2),[-45 0 45]);
MU_phi_3 = trimf(tb(:,2),[15 52.5 90]);
MU_phi_4 = trimf(tb(:,2),[80 90 100]);
MU_phi_5 = trimf(tb(:,2),[90 127.5 165]);
MU_phi_6 = trimf(tb(:,2),[135 180 225]);
MU_phi_7 = trimf(tb(:,2),[195 245 295]);
MU_phi = [MU_phi_1 MU_phi_2 MU_phi_3 MU_phi_4 MU_phi_5 MU_phi_6 MU_phi_7];

[deg_x,ind_x] = max(MU_x,[],2);
[deg_phi,ind_phi] = max(MU_phi,[],2);
[deg_theta,ind_theta] = max(MU_theta,[],2);

rule = [ind_x ind_phi ind_theta];
rule_deg = deg_x.*deg_phi.*deg_theta;

%% Step 3: Assign a Degree to Each Rule.
% For conflicting rule in rule base, assign the maximum degree for the
% conflict group.
temp = sortrows([rule rule_deg]);
[p q]=size(temp);
```

```
value=temp(1,4);
j=1;
m=zeros(p,4);
c=0;
for i=1:p-1
    if(isequal([temp(i,1) temp(i,2)],[temp(i+1,1) temp(i+1,2)]))
        if(value<temp(i+1,4))
            value=temp(i+1,4);
            j=i+1;
        end
    else
      value=temp(j+1,4);
      m(c+1,:)=temp(j,:);
      c=c+1;
    end
end
M=m(1:c,:);
M_final=[M;  5 7 7 0.3625];
rules = sortrows(M_final);

%% Step 4: Create a Combined Fuzzy Rule Base.
% Not required in this case. It requires only in the case where or
% connectors are used.

%% Step 5: Determine a Mapping Based on the Combined Fuzzy Rule Base.
% Defuzzification also present in this step. The defuzzification strategy
% is Centroid of Area (COA).
figure;
axis([0 20 -10 100]); % x axis and y-axis limit.
iterat = 70;
y_t = [-40 -20 -7 0 7 20 40]; % the points where membership value is 1.

%% Input 1
in_x = 9; % Input x (by user)
in_phi = 220; % Input phi (by user)
% theta_output = 0; % Output theta
sam_y = 2; % initial y position

% plot(x_input,y_sample,'r*');
% title(sprintf('Case 1: x = %s and phi = %s - Truck Trajectory',
num2str(x_input), num2str(in_phi)));
% hold on;

fin_y = zeros(1,1);
fin_x = zeros(1,1);
fin_x(1,1) = in_x;
fin_y(1,1) = sam_y;
for i = 1:iterat % no. of iterations for trajectory tracking.

    % value of x.
    mu_x1 = trapmf(in_x,[0 0 1.5 7]);
    mu_x2 = trimf(in_x,[4 7 10]);
    mu_x3 = trimf(in_x,[9 10 11]);
    mu_x4 = trimf(in_x,[10 13 16]);
    mu_x5 = trapmf(in_x,[13 18.5 20 20]);

    % value of phi.
    mu_phi1 = trimf(in_phi,[-115 -65 -15]);
    mu_phi2 = trimf(in_phi,[-45 0 45]);
```

```
    mu_phi3 = trimf(in_phi,[15 52.5 90]);
    mu_phi4 = trimf(in_phi,[80 90 100]);
    mu_phi5 = trimf(in_phi,[90 127.5 165]);
    mu_phi6 = trimf(in_phi,[135 180 225]);
    mu_phi7 = trimf(in_phi,[195 245 295]);

    mu_x = [mu_x1 mu_x2 mu_x3 mu_x4 mu_x5];
    mu_p = [mu_phi1 mu_phi2 mu_phi3 mu_phi4 mu_phi5 mu_phi6 mu_phi7];
    mo = mu_x(rules(:,1)).*mu_p(rules(:,2)); % product operation to determine the
degree of output control.
    y_bar = y_t(rules(:,3));
    out_theta = sum(mo.*y_bar)/sum(mo); % Value of theta from the rule base.

    % Approximate kinematics of truck backer upper control for calculating the
next states.
    in_x = in_x + cosd(in_phi + out_theta) + sind(out_theta)*sind(in_phi);
    fin_x(i+1,1) = in_x;
    sam_y = sam_y + sind(in_phi + out_theta) - sind(out_theta)*cosd(in_phi); %
displacement on y axis.
    fin_y(i+1,1) = sam_y;
    in_phi = in_phi - asind(2*sind(out_theta)/q);

end

% Plot the trajectory.
plot(fin_x,fin_y,'.-','MarkerSize',12);
set(gca,'Fontsize',10,'FontName','Times New Roman');
hold on;
xlabel('x','FontSize',10,'FontName','Times New Roman');

%% Input 2
in_x = 0; % Input x (by user)
in_phi = -30; % Input phi (by user)
% theta_output = 0; % Output theta
sam_y = 2; % initial y position

% plot(x_input,y_sample,'r*');
% title(sprintf('Case 1: x = %s and phi = %s - Truck Trajectory',
num2str(x_input), num2str(in_phi)));
% hold on;

fin_y = zeros(1,1);
fin_x = zeros(1,1);
fin_x(1,1) = in_x;
fin_y(1,1) = sam_y;
for i = 1:iterat % no. of iterations for trajectory tracking.

    % value of x.
    mu_x1 = trapmf(in_x,[0 0 1.5 7]);
    mu_x2 = trimf(in_x,[4 7 10]);
    mu_x3 = trimf(in_x,[9 10 11]);
    mu_x4 = trimf(in_x,[10 13 16]);
    mu_x5 = trapmf(in_x,[13 18.5 20 20]);

    % value of phi.
    mu_phi1 = trimf(in_phi,[-115 -65 -15]);
    mu_phi2 = trimf(in_phi,[-45 0 45]);
    mu_phi3 = trimf(in_phi,[15 52.5 90]);
    mu_phi4 = trimf(in_phi,[80 90 100]);
```

```
    mu_phi5 = trimf(in_phi,[90 127.5 165]);
    mu_phi6 = trimf(in_phi,[135 180 225]);
    mu_phi7 = trimf(in_phi,[195 245 295]);

    mu_x = [mu_x1 mu_x2 mu_x3 mu_x4 mu_x5];
    mu_p = [mu_phi1 mu_phi2 mu_phi3 mu_phi4 mu_phi5 mu_phi6 mu_phi7];
    mo = mu_x(rules(:,1)).*mu_p(rules(:,2)); % product operation to determine the
degree of output control.
    y_bar = y_t(rules(:,3));
    out_theta = sum(mo.*y_bar)/sum(mo); % Value of theta from the rule base.

    % Approximate kinematics of truck backer upper control for calculating the
next states.
    in_x = in_x + cosd(in_phi + out_theta) + sind(out_theta)*sind(in_phi);
    fin_x(i+1,1) = in_x;
    sam_y = sam_y + sind(in_phi + out_theta) - sind(out_theta)*cosd(in_phi); %
displacement on y axis.
    fin_y(i+1,1) = sam_y;
    in_phi = in_phi - asind(2*sind(out_theta)/q);

end

% Plot the trajectory.
plot(fin_x,fin_y,'-','MarkerSize',12);
set(gca,'Fontsize',10,'FontName','Times New Roman');
hold on;
xlabel('x','FontSize',10,'FontName','Times New Roman');

%% Input 3
in_x = 14; % Input x (by user)
in_phi = 30; % Input phi (by user)
% theta_output = 0; % Output theta
sam_y = 2; % initial y position

% plot(x_input,y_sample,'r*');
% title(sprintf('Case 1: x = %s and phi = %s - Truck Trajectory',
num2str(x_input), num2str(in_phi)));
% hold on;

fin_y = zeros(1,1);
fin_x = zeros(1,1);
fin_x(1,1) = in_x;
fin_y(1,1) = sam_y;
for i = 1:iterat % no. of iterations for trajectory tracking.

    % value of x.
    mu_x1 = trapmf(in_x,[0 0 1.5 7]);
    mu_x2 = trimf(in_x,[4 7 10]);
    mu_x3 = trimf(in_x,[9 10 11]);
    mu_x4 = trimf(in_x,[10 13 16]);
    mu_x5 = trapmf(in_x,[13 18.5 20 20]);

    % value of phi.
    mu_phi1 = trimf(in_phi,[-115 -65 -15]);
    mu_phi2 = trimf(in_phi,[-45 0 45]);
    mu_phi3 = trimf(in_phi,[15 52.5 90]);
    mu_phi4 = trimf(in_phi,[80 90 100]);
    mu_phi5 = trimf(in_phi,[90 127.5 165]);
    mu_phi6 = trimf(in_phi,[135 180 225]);
```

```
    mu_phi7 = trimf(in_phi,[195 245 295]);

    mu_x = [mu_x1 mu_x2 mu_x3 mu_x4 mu_x5];
    mu_p = [mu_phi1 mu_phi2 mu_phi3 mu_phi4 mu_phi5 mu_phi6 mu_phi7];
    mo = mu_x(rules(:,1)).*mu_p(rules(:,2)); % product operation to determine the
degree of output control.
    y_bar = y_t(rules(:,3));
    out_theta = sum(mo.*y_bar)/sum(mo); % Value of theta from the rule base.

    % Approximate kinematics of truck backer upper control for calculating the
next states.
    in_x = in_x + cosd(in_phi + out_theta) + sind(out_theta)*sind(in_phi);
    fin_x(i+1,1) = in_x;
    sam_y = sam_y + sind(in_phi + out_theta) - sind(out_theta)*cosd(in_phi); %
displacement on y axis.
    fin_y(i+1,1) = sam_y;
    in_phi = in_phi - asind(2*sind(out_theta)/q);

end

% Plot the trajectory.
plot(fin_x,fin_y,'*','MarkerSize',6);
set(gca,'Fontsize',10,'FontName','Times New Roman');
hold on;
xlabel('x','FontSize',10,'FontName','Times New Roman');
```

Numerical and linguistic information control and simulation result:

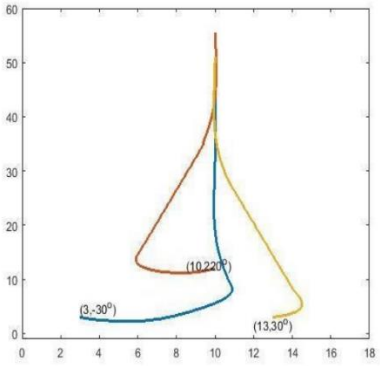
For this simulation purpose, we used 27 sets of the fuzzy rules as generated from the data after removing conflicting rule, which are mentioned in the below table

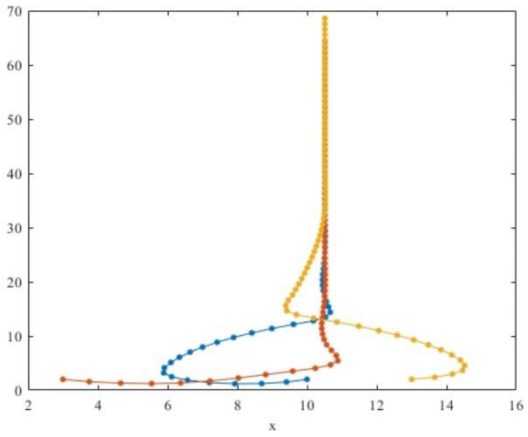
Rule base for Numerical and linguistic information control

Input: x

Input: Phi	***S2***	***S1***	***CE***	***B1***	***B2***
S3	S2	S3			
S2	S2	S3	S3	S3	
S1	B1	S1	S2	S3	S2
CE	B2	B2	CE	S2	S2
B1	B2	B3	B2	B1	S1
B2		B3	B3	B3	B2
B3				B3	B2

Truck trajectory using both numerical data and linguistic information are shown in below Fig.






```
%% Step 1: Divide the input and output spaces into fuzzy regions.
% Divide each domain interval into 2N+1 regions
% m_x, m_phi and m_theta

x = 0:0.1:20; % Universe of discourse of input x.
m_x(:,1) = trapmf(x,[0 0 1.5 7]);
m_x(:,2) = trimf(x,[4 7 10]);
m_x(:,3) = trimf(x,[9 10 11]);
m_x(:,4) = trimf(x,[10 13 16]);
m_x(:,5) = trapmf(x,[13 18.5 20 20]);
```

```
phi = -115:0.1:295; % Universe of discourse of input phi.
m_phi(:,1) = trimf(phi,[-115 -65 -15]);
m_phi(:,2) = trimf(phi,[-45 0 45]);
m_phi(:,3) = trimf(phi,[15 52.5 90]);
m_phi(:,4) = trimf(phi,[80 90 100]);
m_phi(:,5) = trimf(phi,[90 127.5 165]);
m_phi(:,6) = trimf(phi,[135 180 225]);
m_phi(:,7) = trimf(phi,[195 245 295]);

theta = -40:0.4:40; % Universe of discourse of output theta.
m_theta(:,1) = trimf(theta,[-40 -40 -20]);
m_theta(:,2) = trimf(theta,[-33 -20 -7]);
m_theta(:,3) = trimf(theta,[-14 -7 0]);
m_theta(:,4) = trimf(theta,[-4 0 4]);
m_theta(:,5) = trimf(theta,[0 7 14]);
m_theta(:,6) = trimf(theta,[7 20 33]);
m_theta(:,7) = trimf(theta,[20 40 40]);

%% Step 2: Generate Fuzzy Rules from Given Data Pairs.
% Calculate the Degree for given datasets

% Input x
mu_x1 = trapmf(tb(:,1),[0 0 1.5 7]);
mu_x2 = trimf(tb(:,1),[4 7 10]);
mu_x3 = trimf(tb(:,1),[9 10 11]);
mu_x4 = trimf(tb(:,1),[10 13 16]);
mu_x5 = trapmf(tb(:,1),[13 18.5 20 20]);
mu_x_test = [mu_x1 mu_x2 mu_x3 mu_x4 mu_x5];

% Input phi
mu_phi1 = trimf(tb(:,2),[-115 -65 -15]);
mu_phi2 = trimf(tb(:,2),[-45 0 45]);
mu_phi3 = trimf(tb(:,2),[15 52.5 90]);
mu_phi4 = trimf(tb(:,2),[80 90 100]);
mu_phi5 = trimf(tb(:,2),[90 127.5 165]);
mu_phi6 = trimf(tb(:,2),[135 180 225]);
mu_phi7 = trimf(tb(:,2),[195 245 295]);
mu_phi = [mu_phi1 mu_phi2 mu_phi3 mu_phi4 mu_phi5 mu_phi6 mu_phi7];

%Output theta
mu_theta1 = trimf(tb(:,3),[-40 -40 -20]);
mu_theta2 = trimf(tb(:,3),[-33 -20 -7]);
mu_theta3 = trimf(tb(:,3),[-14 -7 0]);
mu_theta4 = trimf(tb(:,3),[-4 0 4]);
mu_theta5 = trimf(tb(:,3),[0 7 14]);
mu_theta6 = trimf(tb(:,3),[7 20 33]);
mu_theta7 = trimf(tb(:,3),[20 40 40]);
mu_theta = [mu_theta1 mu_theta2 mu_theta3 mu_theta4 mu_theta5 mu_theta6
mu_theta7];

[avg_degree_x,avg_index_x] = max(mu_x_test,[],2);
[avg_degree_phi,avg_index_phi] = max(mu_phi,[],2);
[avg_degree_theta,avg_index_theta] = max(mu_theta,[],2);

rules = [avg_index_x avg_index_phi avg_index_theta];
rules_degree = avg_degree_x.*avg_degree_phi.*avg_degree_theta;

%% Step 3: Assign a Degree to Each Rule.
```

```
% For conflicting rule in rule base, assign the maximum degree for the
% conflict group.
r_temp = sortrows([rules rules_degree]);
% r_temp = r_temp(end:-1:1,:);
% rules_degree = rules_degree(end:-1:1,:);
% rules = rules(end:-1:1,:);
temp1(:,1:2) = r_temp(:,1:2);
% temp1(:,3) = r_temp(:,4);
[new_rule, rule_index] = unique(temp1,'rows','sorted');
Rule_temp = zeros(size(new_rule,1),4);
for i = 1:size(new_rule,1)
    if i==size(new_rule,1)
        [temp3, temp4] = max( r_temp(rule_index(i):size(r_temp,1),4) );
        Rule_temp(i,:) = r_temp(temp4+rule_index(i)-1,:);
%     elseif i==1
%         [temp3, temp4] = max( r_temp(rule_index(i):rule_index(i+1)-1,4) );
%         Rule_temp(i,:) = r_temp(temp4,:);
    else
        [temp3, temp4] = max( r_temp(rule_index(i):rule_index(i+1)-1,4) );
        Rule_temp(i,:) = r_temp(temp4+rule_index(i)-1,:);
    end
end
% new_rule = [new_rule r_temp(rule_index,3:4)];
% final_rules = new_rule;
final_rules = sortrows(Rule_temp);

%% Step 4: Create a Combined Fuzzy Rule Base.
% Not required in this case. It requires only in the case where or
% connectors are used.

%% Step 5: Determine a Mapping Based on the Combined Fuzzy Rule Base.
% Defuzzification also present in this step. The defuzzification strategy
% is Centroid of Area (COA).
figure;
axis([0 20 -10 100]); % x axis and y-axis limit.
iter = 70;
y_t = [-40 -20 -7 0 7 20 40]; % the points where membership value is 1.

%% Input 1
x_input = 10; % Input x (by user)
phi_input = 220; % Input phi (by user)
% theta_output = 0; % Output theta
y_sample = 2; % initial y position

% plot(x_input,y_sample,'r*');
% title(sprintf('Case 1: x = %s and phi = %s - Truck Trajectory',
num2str(x_input), num2str(phi_input)));
% hold on;

y_final = zeros(1,1);
x_final = zeros(1,1);
x_final(1,1) = x_input;
y_final(1,1) = y_sample;
for i = 1:iter % no. of iterations for trajectory tracking.

    % value of x.
    mu_x1_test = trapmf(x_input,[0 0 1.5 7]);
    mu_x2_test = trimf(x_input,[4 7 10]);
    mu_x3_test = trimf(x_input,[9 10 11]);
```

```
    mu_x4_test = trimf(x_input,[10 13 16]);
    mu_x5_test = trapmf(x_input,[13 18.5 20 20]);

    % value of phi.
    mu_phi1_test = trimf(phi_input,[-115 -65 -15]);
    mu_phi2_test = trimf(phi_input,[-45 0 45]);
    mu_phi3_test = trimf(phi_input,[15 52.5 90]);
    mu_phi4_test = trimf(phi_input,[80 90 100]);
    mu_phi5_test = trimf(phi_input,[90 127.5 165]);
    mu_phi6_test = trimf(phi_input,[135 180 225]);
    mu_phi7_test = trimf(phi_input,[195 245 295]);

    mu_x_test = [mu_x1_test mu_x2_test mu_x3_test mu_x4_test mu_x5_test];
    mu_p_test = [mu_phi1_test mu_phi2_test mu_phi3_test mu_phi4_test mu_phi5_test
mu_phi6_test mu_phi7_test];
    mo = mu_x_test(final_rules(:,1)).*mu_p_test(final_rules(:,2)); % product
operation to determine the degree of output control.
    y_bar = y_t(final_rules(:,3));
    theta_output = sum(mo.*y_bar)/sum(mo); % Value of theta from the rule base.

    % Approximate kinematics of truck backer upper control for calculating the
next states.
    x_input = x_input + cosd(phi_input + theta_output) +
sind(theta_output)*sind(phi_input);
    x_final(i+1,1) = x_input;
    y_sample = y_sample + sind(phi_input + theta_output) -
sind(theta_output)*cosd(phi_input); % displacement on y axis.
    y_final(i+1,1) = y_sample;
    phi_input = phi_input - asind(2*sind(theta_output)/b);

end

% Plot the trajectory.
plot(x_final,y_final,'.-','MarkerSize',12);
set(gca,'Fontsize',10,'FontName','Times New Roman');
hold on;
xlabel('x','FontSize',10,'FontName','Times New Roman');

%% Input 2
x_input = 3; % Input x (by user)
phi_input = -30; % Input phi (by user)
% theta_output = 0; % Output theta
y_sample = 2; % initial y position

% plot(x_input,y_sample,'r*');
% title(sprintf('Case 1: x = %s and phi = %s - Truck Trajectory',
num2str(x_input), num2str(phi_input)));
% hold on;

y_final = zeros(1,1);
x_final = zeros(1,1);
x_final(1,1) = x_input;
y_final(1,1) = y_sample;
for i = 1:iter % no. of iterations for trajectory tracking.

    % value of x.
    mu_x1_test = trapmf(x_input,[0 0 1.5 7]);
    mu_x2_test = trimf(x_input,[4 7 10]);
    mu_x3_test = trimf(x_input,[9 10 11]);
```

```
    mu_x4_test = trimf(x_input,[10 13 16]);
    mu_x5_test = trapmf(x_input,[13 18.5 20 20]);

    % value of phi.
    mu_phi1_test = trimf(phi_input,[-115 -65 -15]);
    mu_phi2_test = trimf(phi_input,[-45 0 45]);
    mu_phi3_test = trimf(phi_input,[15 52.5 90]);
    mu_phi4_test = trimf(phi_input,[80 90 100]);
    mu_phi5_test = trimf(phi_input,[90 127.5 165]);
    mu_phi6_test = trimf(phi_input,[135 180 225]);
    mu_phi7_test = trimf(phi_input,[195 245 295]);

    mu_x_test = [mu_x1_test mu_x2_test mu_x3_test mu_x4_test mu_x5_test];
    mu_p_test = [mu_phi1_test mu_phi2_test mu_phi3_test mu_phi4_test mu_phi5_test
mu_phi6_test mu_phi7_test];
    mo = mu_x_test(final_rules(:,1)).*mu_p_test(final_rules(:,2)); % product
operation to determine the degree of output control.
    y_bar = y_t(final_rules(:,3));
    theta_output = sum(mo.*y_bar)/sum(mo); % Value of theta from the rule base.

    % Approximate kinematics of truck backer upper control for calculating the
next states.
    x_input = x_input + cosd(phi_input + theta_output) +
sind(theta_output)*sind(phi_input);
    x_final(i+1,1) = x_input;
    y_sample = y_sample + sind(phi_input + theta_output) -
sind(theta_output)*cosd(phi_input); % displacement on y axis.
    y_final(i+1,1) = y_sample;
    phi_input = phi_input - asind(2*sind(theta_output)/b);

end

% Plot the trajectory.
plot(x_final,y_final,'.-','MarkerSize',12);
set(gca,'Fontsize',10,'FontName','Times New Roman');
hold on;
xlabel('x','FontSize',10,'FontName','Times New Roman');

%% Input 3
x_input = 13; % Input x (by user)
phi_input = 30; % Input phi (by user)
% theta_output = 0; % Output theta
y_sample = 2; % initial y position

% plot(x_input,y_sample,'r*');
% title(sprintf('Case 1: x = %s and phi = %s - Truck Trajectory',
num2str(x_input), num2str(phi_input)));
% hold on;

y_final = zeros(1,1);
x_final = zeros(1,1);
x_final(1,1) = x_input;
y_final(1,1) = y_sample;
for i = 1:iter % no. of iterations for trajectory tracking.

    % value of x.
    mu_x1_test = trapmf(x_input,[0 0 1.5 7]);
    mu_x2_test = trimf(x_input,[4 7 10]);
    mu_x3_test = trimf(x_input,[9 10 11]);
```

```
    mu_x4_test = trimf(x_input,[10 13 16]);
    mu_x5_test = trapmf(x_input,[13 18.5 20 20]);

    % value of phi.
    mu_phi1_test = trimf(phi_input,[-115 -65 -15]);
    mu_phi2_test = trimf(phi_input,[-45 0 45]);
    mu_phi3_test = trimf(phi_input,[15 52.5 90]);
    mu_phi4_test = trimf(phi_input,[80 90 100]);
    mu_phi5_test = trimf(phi_input,[90 127.5 165]);
    mu_phi6_test = trimf(phi_input,[135 180 225]);
    mu_phi7_test = trimf(phi_input,[195 245 295]);

    mu_x_test = [mu_x1_test mu_x2_test mu_x3_test mu_x4_test mu_x5_test];
    mu_p_test = [mu_phi1_test mu_phi2_test mu_phi3_test mu_phi4_test mu_phi5_test
mu_phi6_test mu_phi7_test];
    mo = mu_x_test(final_rules(:,1)).*mu_p_test(final_rules(:,2)); % product
operation to determine the degree of output control.
    y_bar = y_t(final_rules(:,3));
    theta_output = sum(mo.*y_bar)/sum(mo); % Value of theta from the rule base.

    % Approximate kinematics of truck backer upper control for calculating the
next states.
    x_input = x_input + cosd(phi_input + theta_output) +
sind(theta_output)*sind(phi_input);
    x_final(i+1,1) = x_input;
    y_sample = y_sample + sind(phi_input + theta_output) -
sind(theta_output)*cosd(phi_input); % displacement on y axis.
    y_final(i+1,1) = y_sample;
    phi_input = phi_input - asind(2*sind(theta_output)/b);

end

% Plot the trajectory.
plot(x_final,y_final,'.-','MarkerSize',12);
set(gca,'Fontsize',10,'FontName','Times New Roman');
hold on;
xlabel('x','FontSize',10,'FontName','Times New Roman');
```

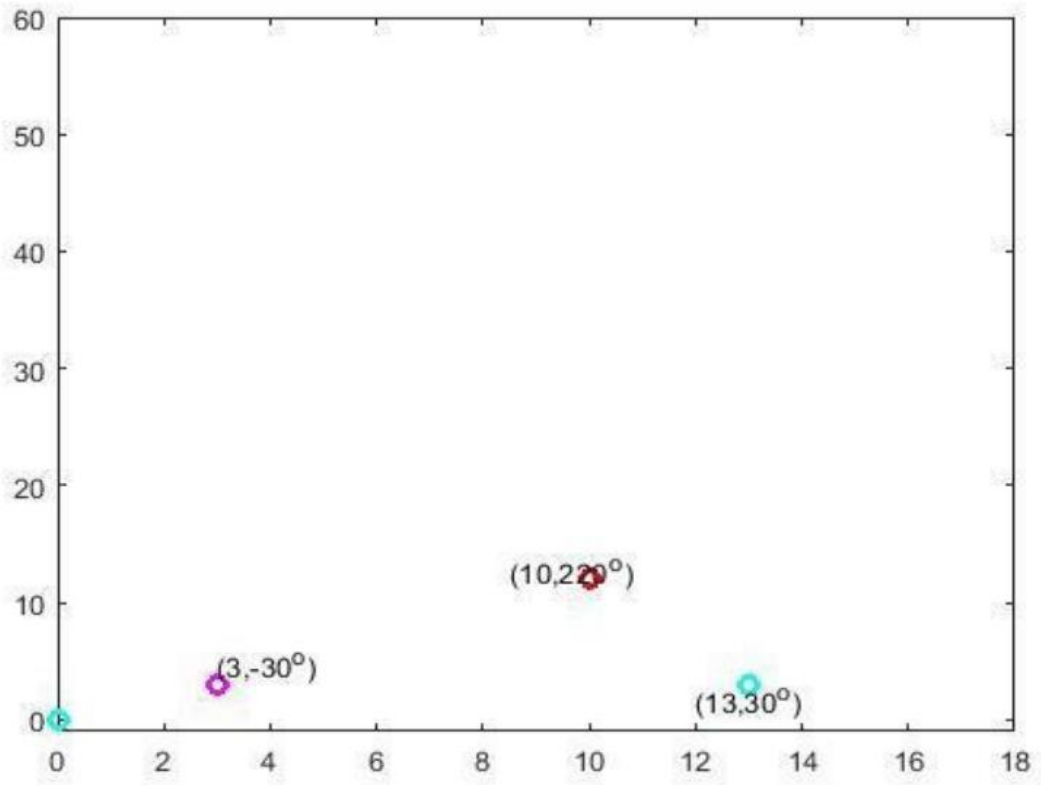
Selected linguistic rule and simulation result:

For this simulation purpose, we used the 3 sets of selected fuzzy rules as generated from the data which are shown in below table

Input: x

Input: Phi	S2	S1	CE	B1	B2
S3					
S2					
S1			S2		
CE			CE		
B1			B2		
B2					
B3					

Truck trajectory using selected linguistic information is shown in below Fig.



5. Application to Time Series Prediction

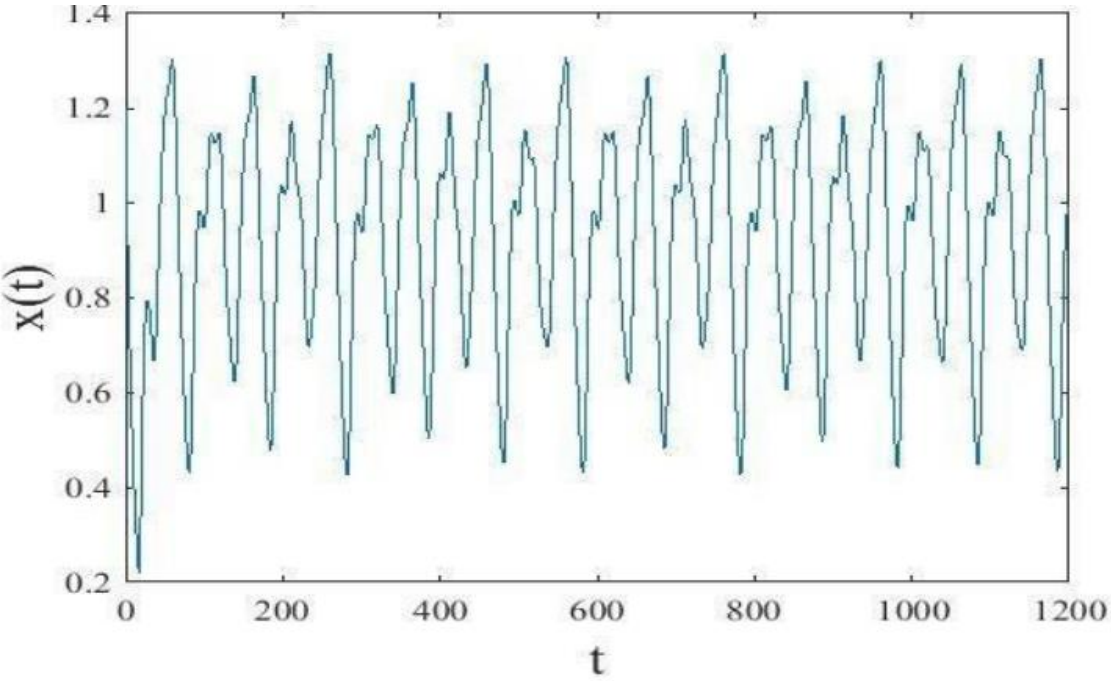
Let z(k) where k = 1, 2,……n be time series. The problem of time series prediction can be formulated as: z(k− m+1), z(k− m+ 2).....z(k) , and determine z(k+1) where, m = number of inputs to determine output.

Mackey-Glass Chaotic series:

Chaotic time are generated from deterministic nonlinear system and are sufficiently that they appear to be random time series, however there are underlying deterministic maps that generate the series, chaotic time series are not random time series. The Mackey-Glass chaotic time series is generated from the following delay differential:

$$\frac{d(x)}{dt} = \frac{0.2x(t-\tau)}{1+x\,\hat{}\,10(t-\tau)} - 0.1x(t) \quad \text{where } \tau > 17$$

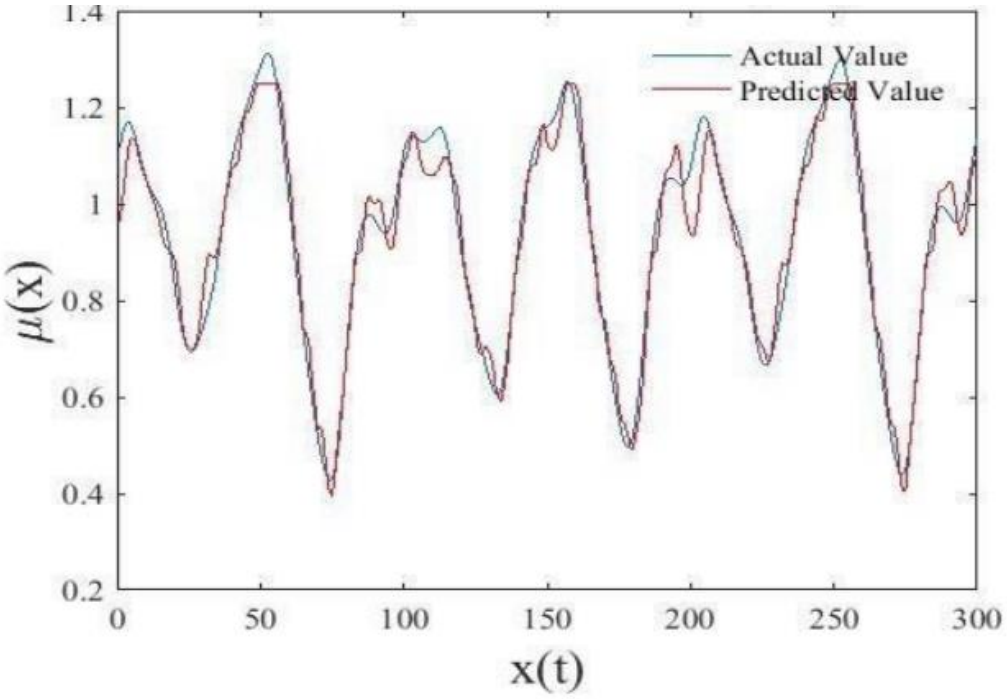
Below Fig. shows the chaotic time series when tau = 17.



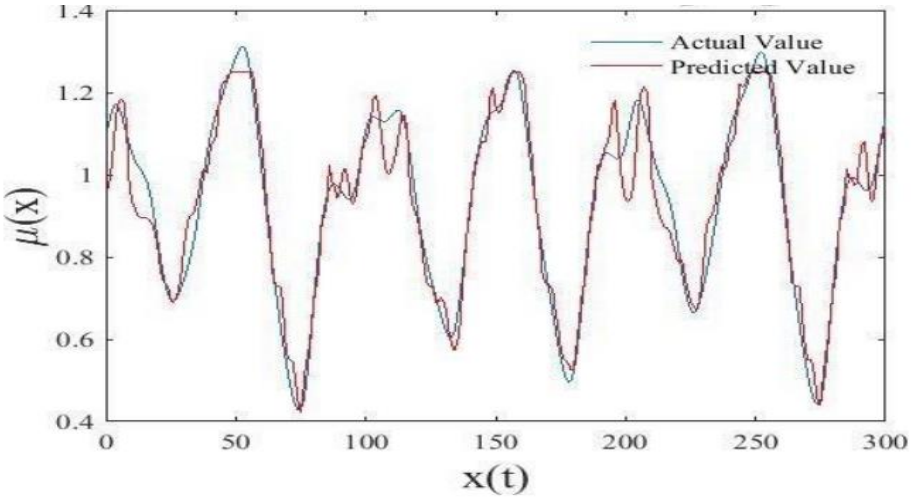
6. RESULTS OF TIME SERIES PREDICTION

For simulation of this paper, the value of m = 9 and l =1 have been chosen. m is the number inputs used to predict the value of next time instance. Above fig. shows 1200 points of

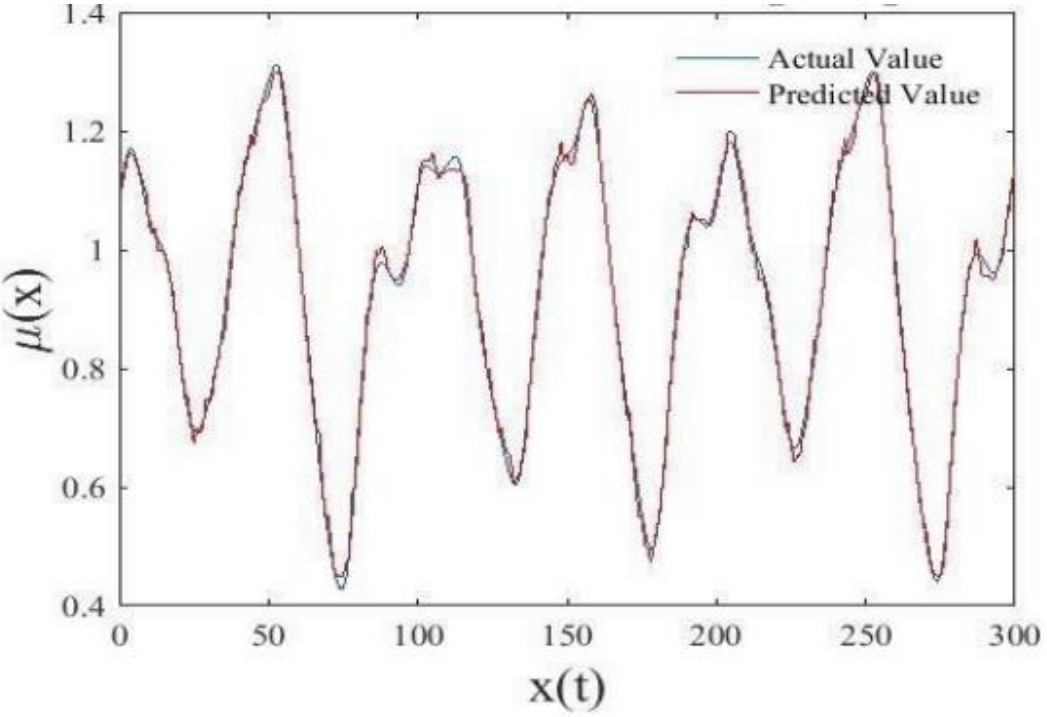
chaotic time-series. The test has been performed for various combinations of training data and test data.



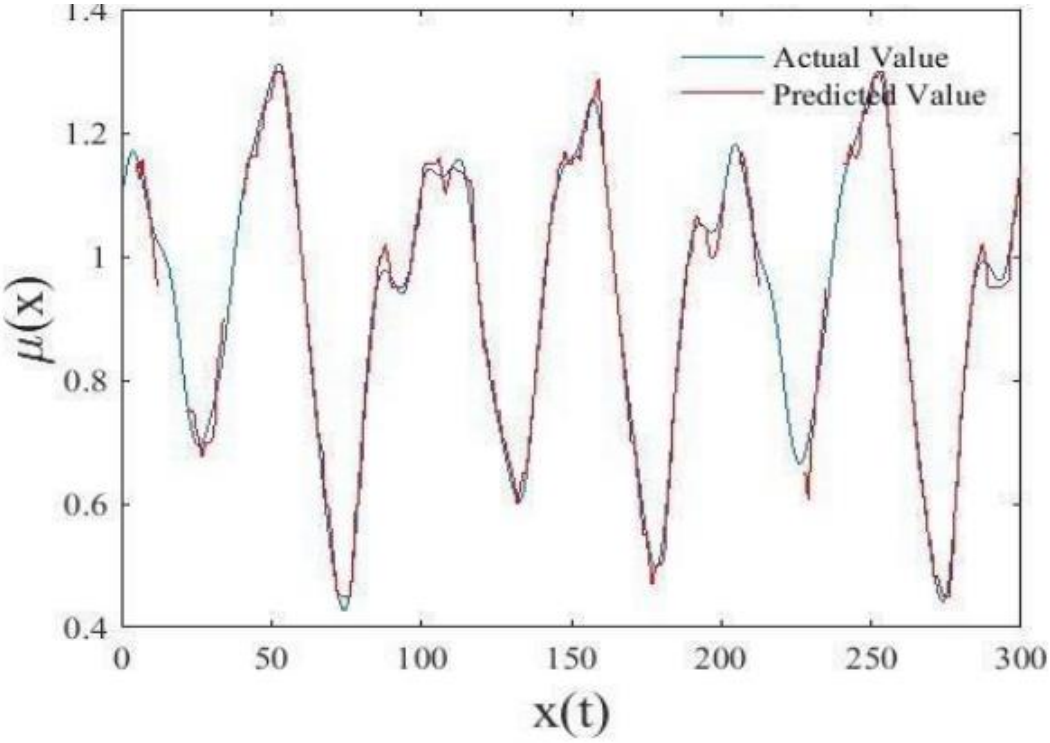
In above fig.: Time-Series Prediction from x(701) to x(1000) using numerical fuzzy predictor when 700 training data (from x(1) to x(700) are used) and number of fuzzy region is 9



In above Fig. : Time-Series Prediction from x(701) to x(1000) using numerical fuzzy predictor when 200 training data (from x(501) to x(700) are used) and number of fuzzy region is 9



In above fig. 10: Time-Series Prediction from x(701) to x(1000) using numerical fuzzy predictor when 700 training data (from x(1) to x(700) are used) and number of fuzzy region is 29



In above fig. 11: Time-Series Prediction from x(701) to x(1000) using numerical fuzzy predictor when 200 training data (from x(501) to x(700) are used) and number of fuzzy region is 29