ANNAMALAI UNIVERSITY

WITH COURAGE AND FAITH

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.E (CSE)

VII - SEMESTER

08CP706 – SOFT COMPUTING TECHNIQUES LAB

Name: ……………………………………………………………………………………

Reg. No. :……………………………………………………………………………………

ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of work done by Mr./Ms.…...........………………………………………..Reg. No.…………………of **VII - Semester B.E(Computer Science & Engineering)** in the **08CP706 - Soft Computing Techniques Lab** during the odd semester of the academic year 2019 – 2020.

Staff In-charge **Internal Examiner**

Place: Annamalainagar **External Examiner**

Date:

INDEX

Performing Union, Intersection and Complement Operations

Ex. No: 1

Date:

Aim :

To write a Program in MATLAB to perform union, intersection and complement operations of fuzzy set.

Algorithm:

1. Read the membership values of two fuzzy sets.
2. Perform union operation by using max( ) function.
3. Perform intersection operation by using min( ) function.
4. Perform complement operation by subtracting membership value from 1
5. Display the result.

Program:

```
% Enter the membership value of fuzzy set
u = input ('Enter the membership value of First Fuzzy set');
v = input ('Enter the membership value of Second Fuzzy set');
%performs Union, Intersection and Complement operations
w=max (u, v);
p=min (u, v);
q1=1-u;
q2=1-v;
%Display Output
disp('Union of Two Fuzzy sets');
disp(w);
disp('Intersection of Two Fuzzy sets');
disp(p);
```

```
disp('Complement of First Fuzzy set');
disp(q1);
disp('Complement of Second Fuzzy set');
disp(q2);
```

Sample Input and Output:

```
Enter the membership value of First Fuzzy set [0.3 0.4]
Enter the membership value of Second Fuzzy set [0.1 0.7]
Union of Two Fuzzy sets
0.3000.   0.7000
Intersection of Two fuzzy sets
0.1000   0.4000.
Complement of First Fuzzy set
0.7000  0.6000.
Complement of Second Fuzzy set
0.9000  0.3000.
```

Result:

Thus, the MATLAB program to perform Union, Intersection and Complement operations of two Fuzzy sets has been executed successfully and the output is verified.

Implementation of De-Morgan’s Law

Ex. No: 2

Date:

Aim:

To write a Program in MATLAB to implement De-Morgan’s law.

Algorithm:

1. Read the membership values of two fuzzy set.
2. Perform Union operation by using max() function and take the complement for the fuzzy set.
3. Perform Intersection operation by using min() function and take the complement.
4. Perform Complement operation for the both fuzzy sets.
5. Perform Intersection operation and Union operation for the Complements of fuzzy set.
6. Verify the formula and display the result.

Program:

```
% Enter  the membership values of fuzzy set

u = input (‘Enter the membership values of first fuzzy set’);

v = input (‘Enter the membership values of  second fuzzy set’);

%To perform operation

w=max (u, v);

p=min (u, v);

q1=1-u;

q2=1-v;

x1=1-w;

x2=min(q1,q2);

y1=1-p;

y2=max(q1-q2);

%Display Output

disp(‘Union of two fuzzy sets  ’);
```

```
disp(w);
disp('Intersection of two fuzzy sets ');
disp(p);
disp('Complement of first fuzzy set  ');
disp(q1);
disp('Complement of second fuzzy set ');
disp(q2);
disp('De-Morgan's Law');
disp('LHS');
disp(x1);
disp('RHS');
disp(x2);
disp(LHS);
disp(y1);
disp('RHS');
disp(y2);
```

Sample Input and output:

Enter the membership values of first fuzzy set [0.3 0.4]

Enter the membership values of second fuzzy set [0.2 0.5]

Union of two fuzzy sets

0.3000.   0.5000

Intersection of two fuzzy sets

0.3000   0.4000.

Complement of first fuzzy set

0.7000 0.6000.

Complement of second fuzzy set

0.8000 0.5000.

De –Morgan’s Law

LHS

0.7000 0.5000

RHS

0.7000 0.5000

LHS

0.8000 0.6000

RHS

0.8000 0.6000

Result:

Thus, the MATLAB program for implementation of De-Morgan’s has been executed successfully and the output is verified.

Plotting Various Membership Functions

Ex. No: 3

Date:

Aim:

To write a program in MATLAB to plot triangular, trapezoidal and bell shaped membership functions.

Algorithm:

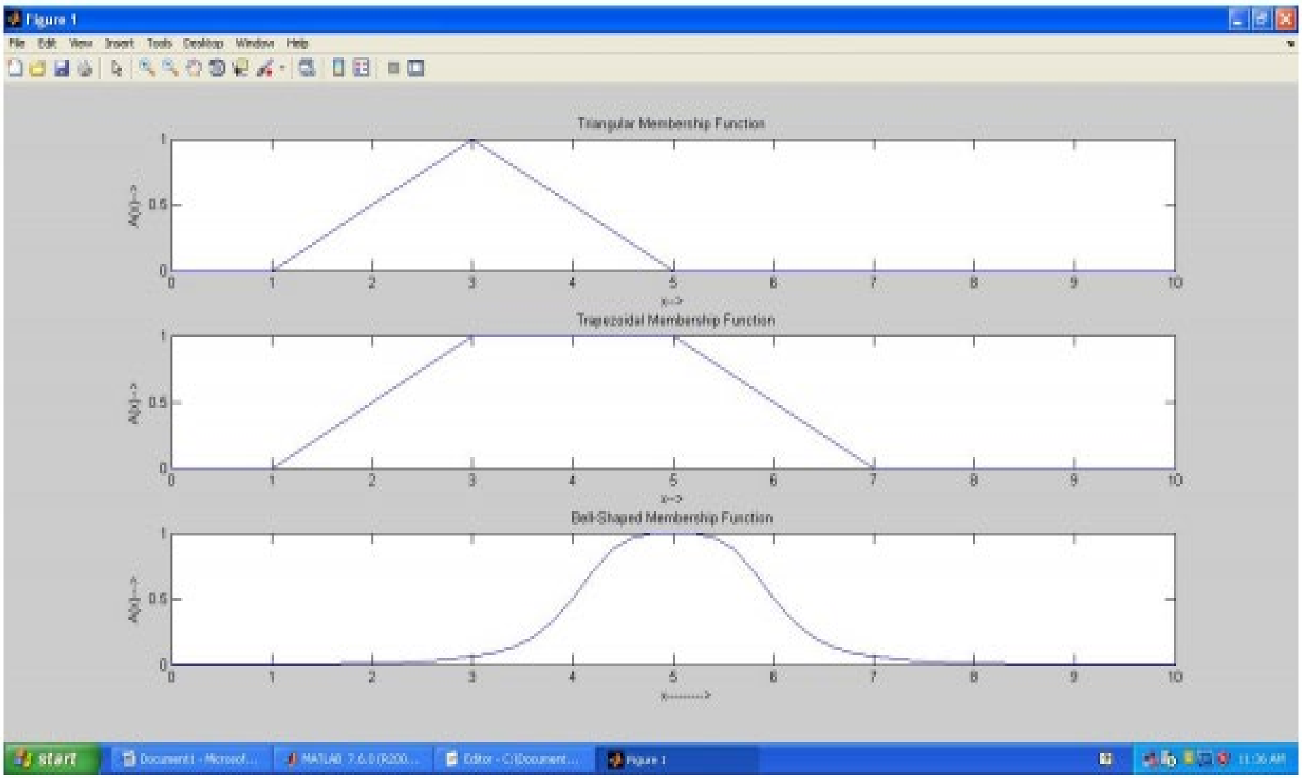
1. Set the limits of x axis.
2. Calculate y using trimf() function with three parameters for triangular membership function.
3. Calculate y using trapmf() function with four parameters for trapezoidal membership function.
4. Calculate y using gbellmf() function with three parameters for bell shaped membership function.
5. Plot x and y values.

Program:

```
%Triangular membership function

x=(0.0:1.0:10.0)’;

y1= trimf(x, [1 3 5]);

subplot(311 )

plot(x,[y1]);

%Trapezoidal membership function

x=(0.0:1.0:10.0)’;

y1= trapmf(x, [1 3 5 7]);

subplot(312)

plot(x, [y1] );

%Bell-shaped membership function

x=(0.0:0.2:10.0);
```

```
y1=gbellmf (x,[3 5 7]);
subplot(313)
plot(x, [y1]);
```

Sample Input and Output:



Result:

Thus, the MATLAB program for plotting membership functions has been executed successfully and the output is verified.

Using Fuzzy toolbox to model tips value

Ex. No: 4

Date:

Aim :

To use fuzzy toolbox to model tips value that is given after a dinner based on quality (not good, satisfying, good and delightful) and service (poor, average or good) and the tip value ranges from Rs. 10 to 100.

Procedure:

INPUTS:

Quality**:** { not good, satisfying, good, delightful }

Service **:**{ poor, average, good }

OUTPUT:

Tips**:** Tip_value ranging from Rs. 10 to 100

Use Fuzzy Inference System (FIS) Editor and perform the following

1. Go to command window in Matlab and type fuzzy.
2. New Fuzzy Logic Designer window will be opened.
3. Give Input / Output Variable .
   a. Go to Edit Window and click Add variable
   b. As per our requirements create two input variables namely quality and service
      Quality**:** { not good, satisfying, good, delightful }
      Service **:**{ poor, average, good }
   c. Similarly, one output variable as tip value ranges from 10 to 100.
4. The values for Quality and Service variables are selected for their respective ranges.
5. Quality:
   a. Double click the Quality input variable .
   b. New window will be opened and remove all the Membership Functions.
   c. Go to Edit and Click Add MFs and select the 4 Parameters for Quality table.

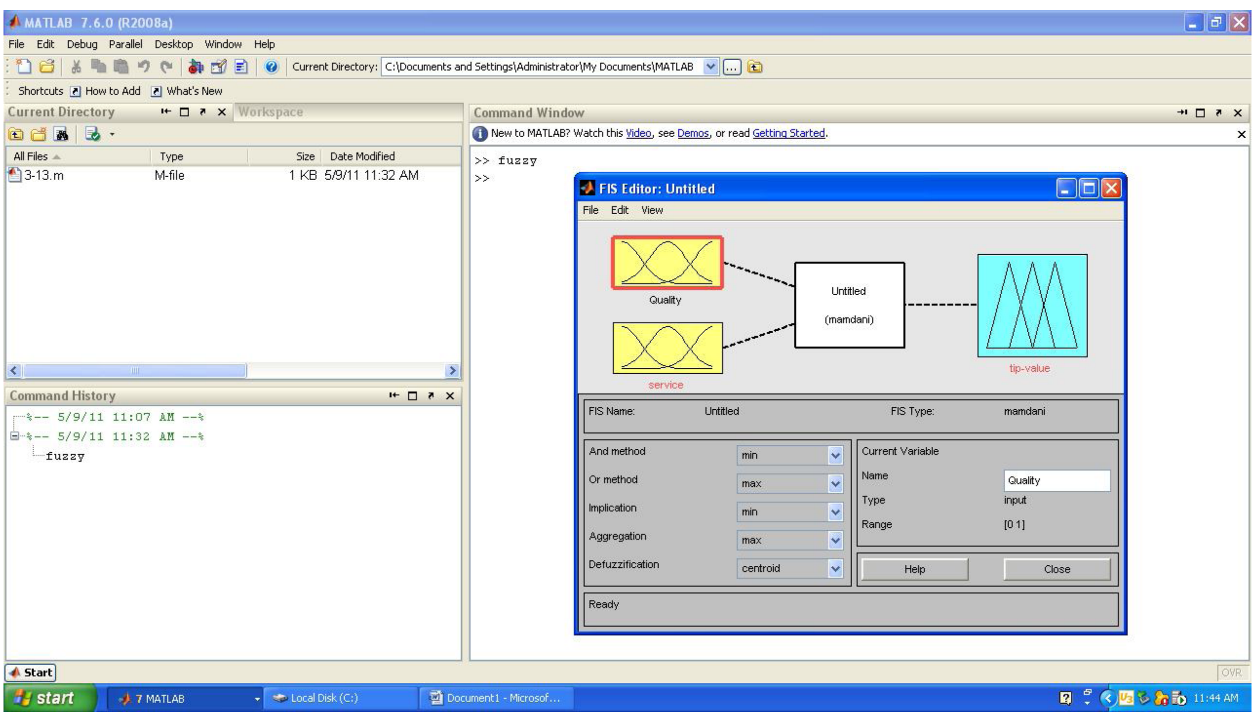
d. Change the following fields as per the table given below .

Inputs : Quality → not good, satisfying, good, delightful			
MF1: Range : [0 1 10] Name : not good Type : trapmf Parameter [0 10 30 50]	MF2: Range : [0 1 10] Name : Cool Type : trimf Parameter [30 50 70]	MF3: Range : [0 1 10] Name : Warm Type : trimf Parameter [50 70 90]	MF4: Range : [0 1 10] Name : Hot Type : trapmf Parameter [70 90 110 110]

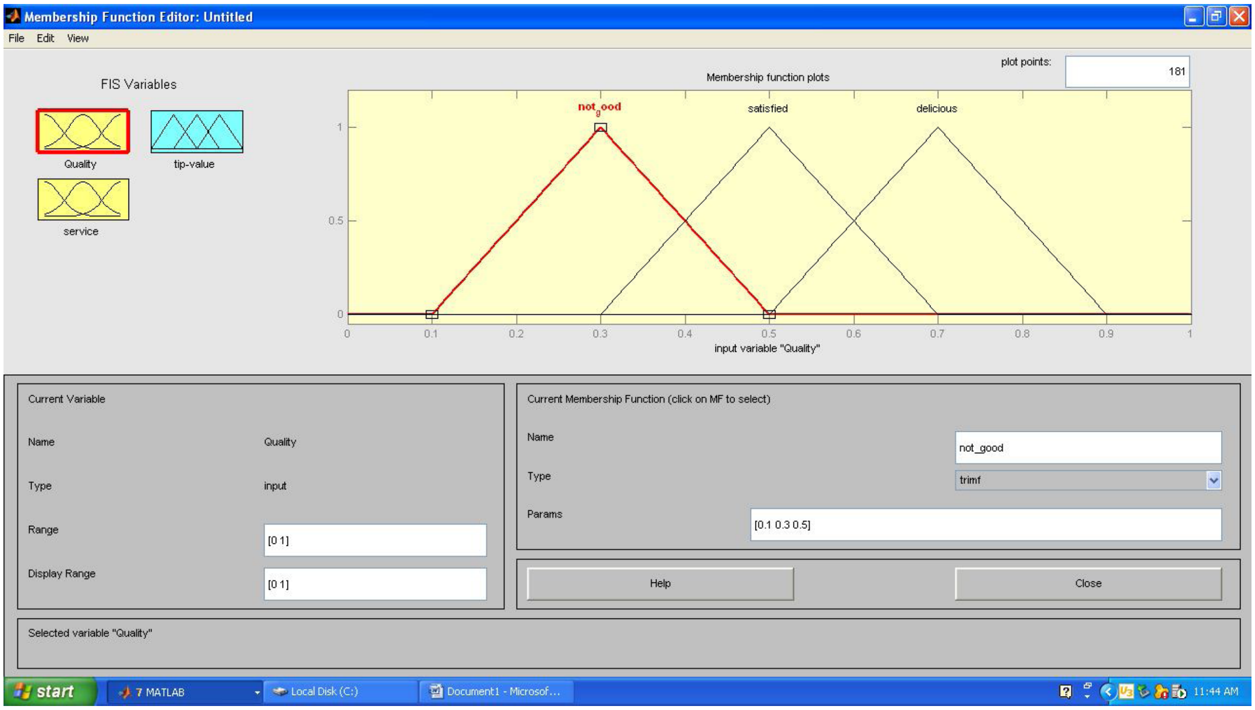
6. Similarly add the data to service and tips variables.
7. Go to Rules: Edit → Rules
8. Add the Rules
9. Go to view → Rules
10. Exit

Sample Input and Output:

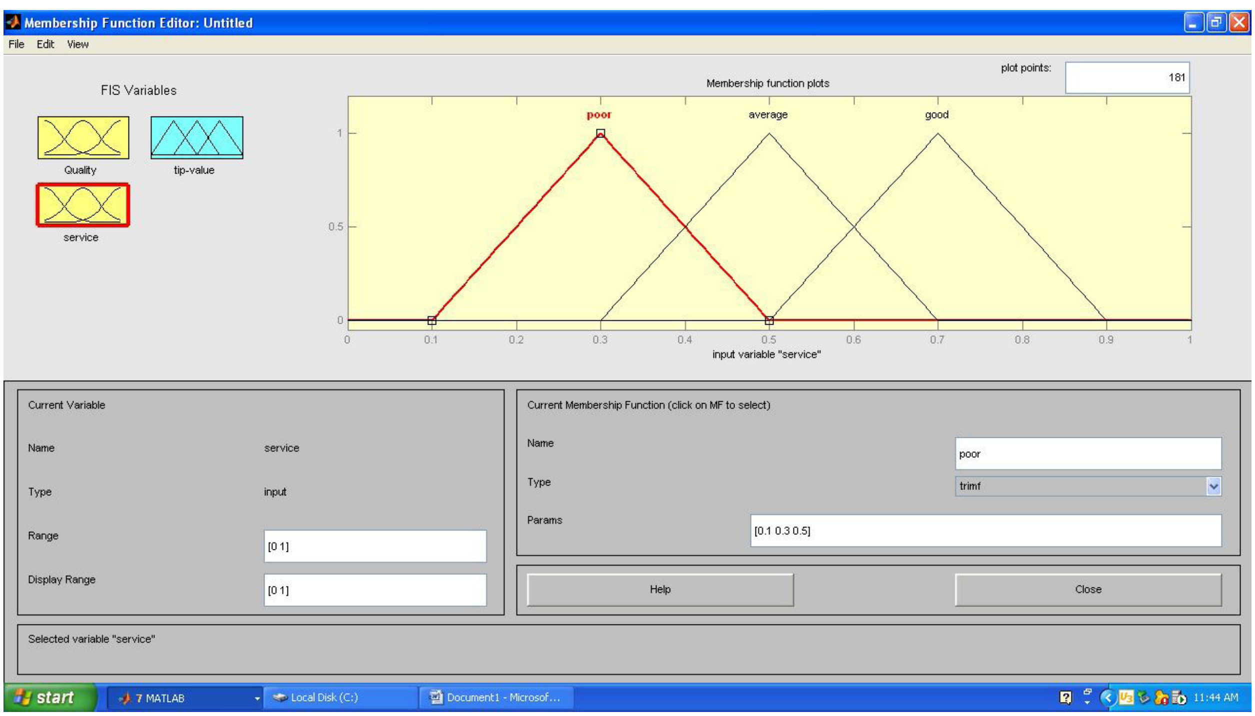
Input and Output Variable in Edit Window



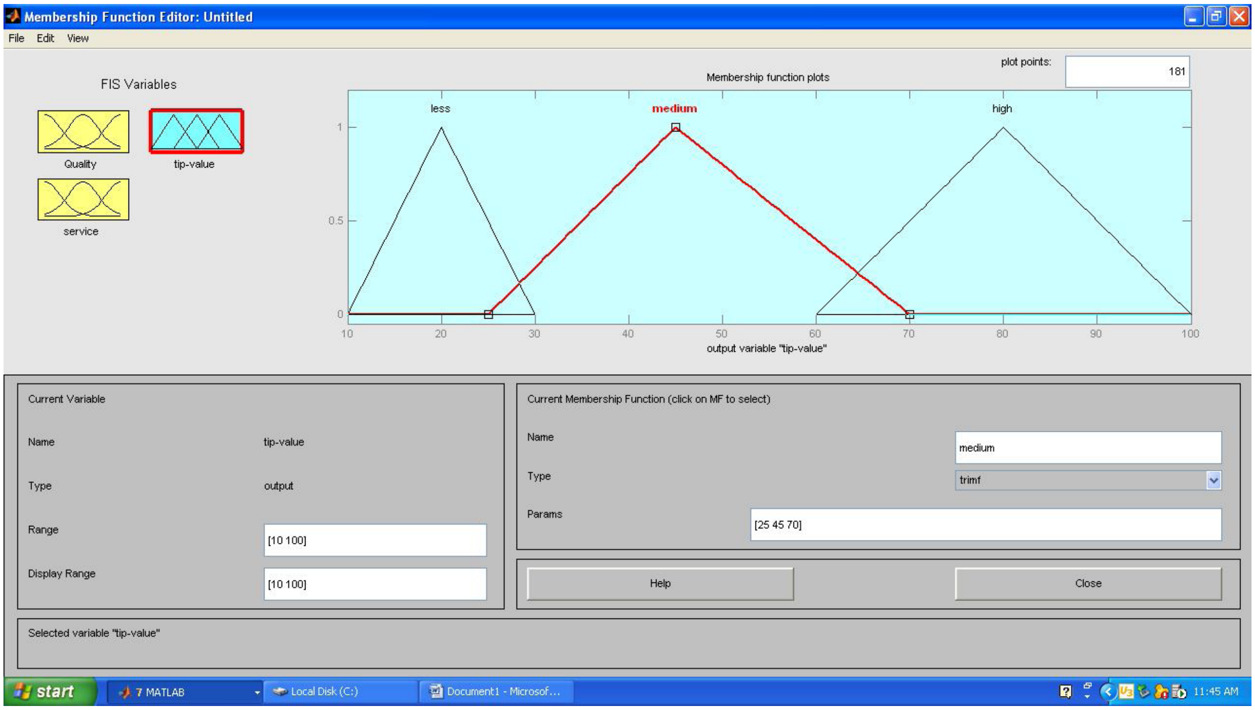
Membership functions for Quality variable



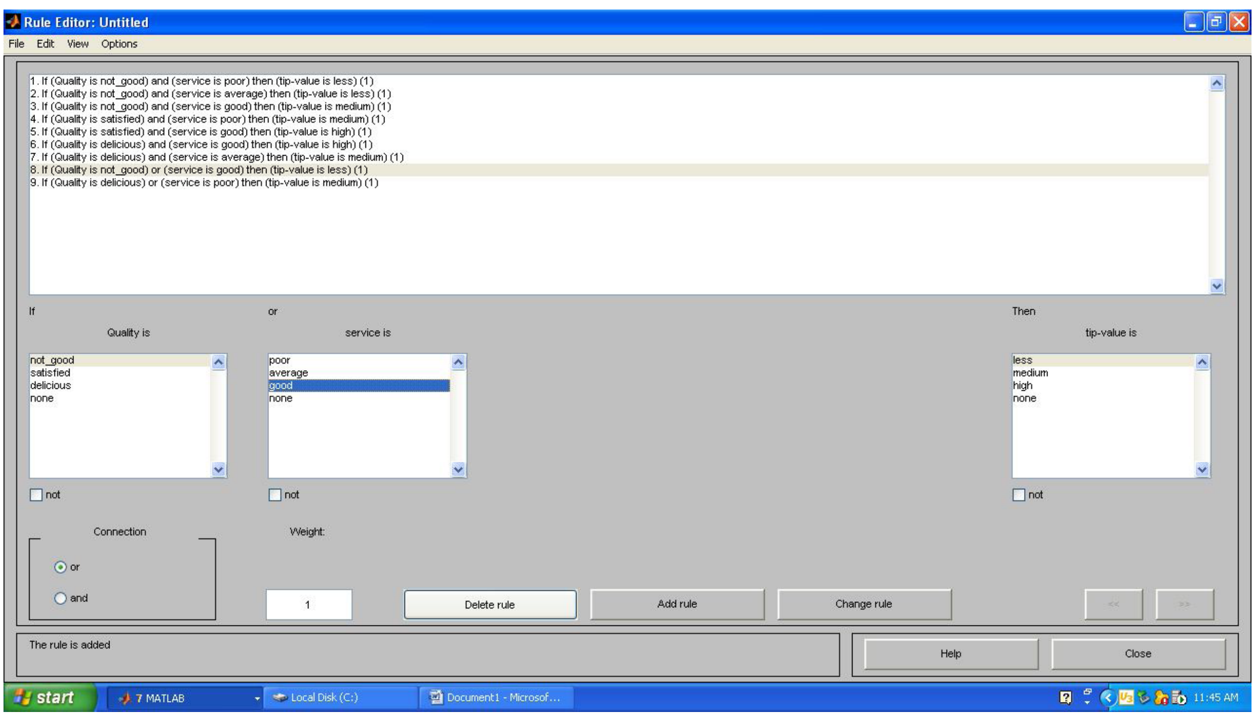
Membership functions for Service variable



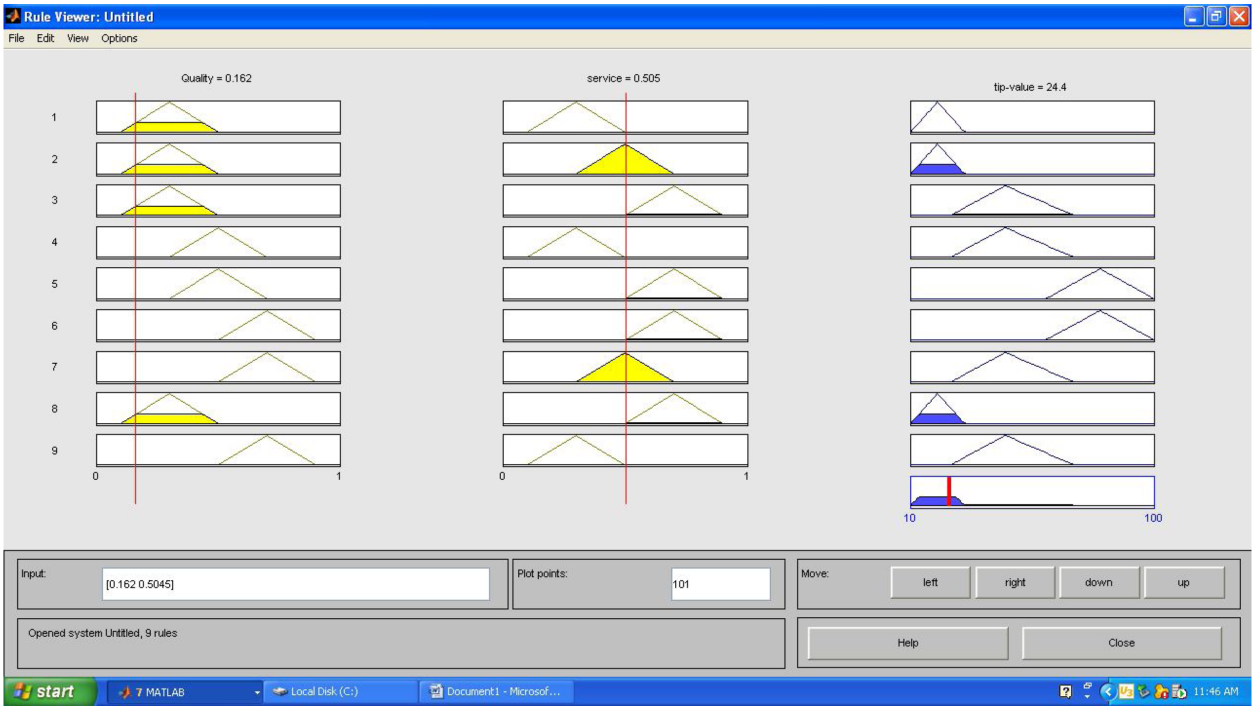
Membership functions for Tips variable



Created rules



Output



Result:

Thus, the fuzzy tool box is used to model tips value.

Implementation of Fuzzy Inference System

Ex. No: 5

Date:

Aim :

To implement a Fuzzy Inference System (FIS) for which the inputs, output and rules are given as below.

INPUTS: Temperature and Cloud Cover

Temperature: {Freeze, Cool, Warm and Hot}

Cloud Cover: {Sunny, Partly Cloud and Overcast}

OUTPUT: Speed

Speed : {Fast and Slow}

RULES:

1. If cloud cover is Sunny and temperature is warm, then drive Fast
   Sunny (Cover) and Warm (Temp) -> Fast (Speed)
2. If cloud cover is cloudy and temperature is cool, then drive Slow
   Cloudy (Cover) and Cool (Temp) -> Slow (Speed)

Procedure

1. Go to command window in Matlab and type fuzzy.
2. Now, new Fuzzy Logic Designer window will be opened.
3. Input / Output Variable
   a. Go to Edit Window and click Add variable.
   b. As per our requirements create two input variables, Temperature and Cloud Cover.
   c. Create one output variable, Speed.
4. Temperature:
   a. Double click the Temperature input variable in Fuzzy Logic Designer window.
   b. New window will be opened and remove all the Membership Functions.
   c. Now, Go to Edit and Click Add MFs and select the 4 Parameters for Temperature Class.
   d. Change the following fields as mentioned data in the given below table.

Inputs : Temperature → Freezing, Cool, Warm and Hot			
MF1: Range : [0 110] Name : Freezing Type : trapmf Parameter [0 0 30 50]	MF2: Range : [0 110] Name : Cool Type : trimf Parameter [30 50 70]	MF3: Range : [0 110] Name : Warm Type : trimf Parameter [50 70 90]	MF4: Range : [0 110] Name : Hot Type : trapmf Parameter [70 90 110 110]

5. Similarly, add the data's to the Cloud Cover variables and Speed variables.
6. Cloud Cover:

Inputs : Cloud Cover → Sunny, Partly Cloud and Overcast		
MF1: Range : [0 100] Name : Sunny Type : trapmf Parameter [0 0 20 40]	MF2: Range : [0 100] Name : Partly Cloud Type : trimf Parameter [20 50 80]	MF3: Range : [0 100] Name : Overcast Type : trapmf Parameter [60 80 100]

7. Speed:

Output : Speed → Slow and Fast	
MF1: Range : [0 100] Name : Slow Type : trapmf Parameter [0 0 25 75]	MF2: Range : [0 100] Name : Fast Type : trapmf Parameter [25 75 100 100]

8. Goto Rules: Edit → Rules
9. Add the Rules

   Rule-1 : Sunny (Cover) and Warm (Temp) -> Fast (Speed)

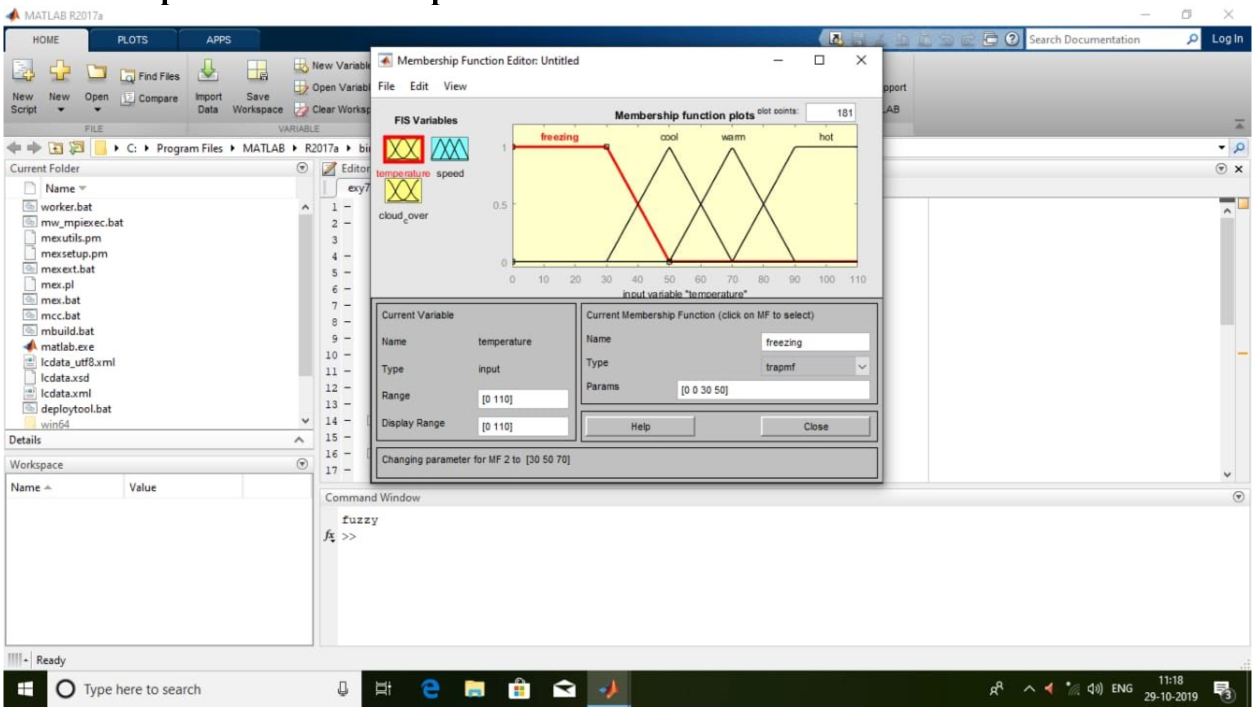
   Rule-2 : Cloudy (Cover) and Cool (Temp) -> Slow (Speed)

   10. Go to view → Rules

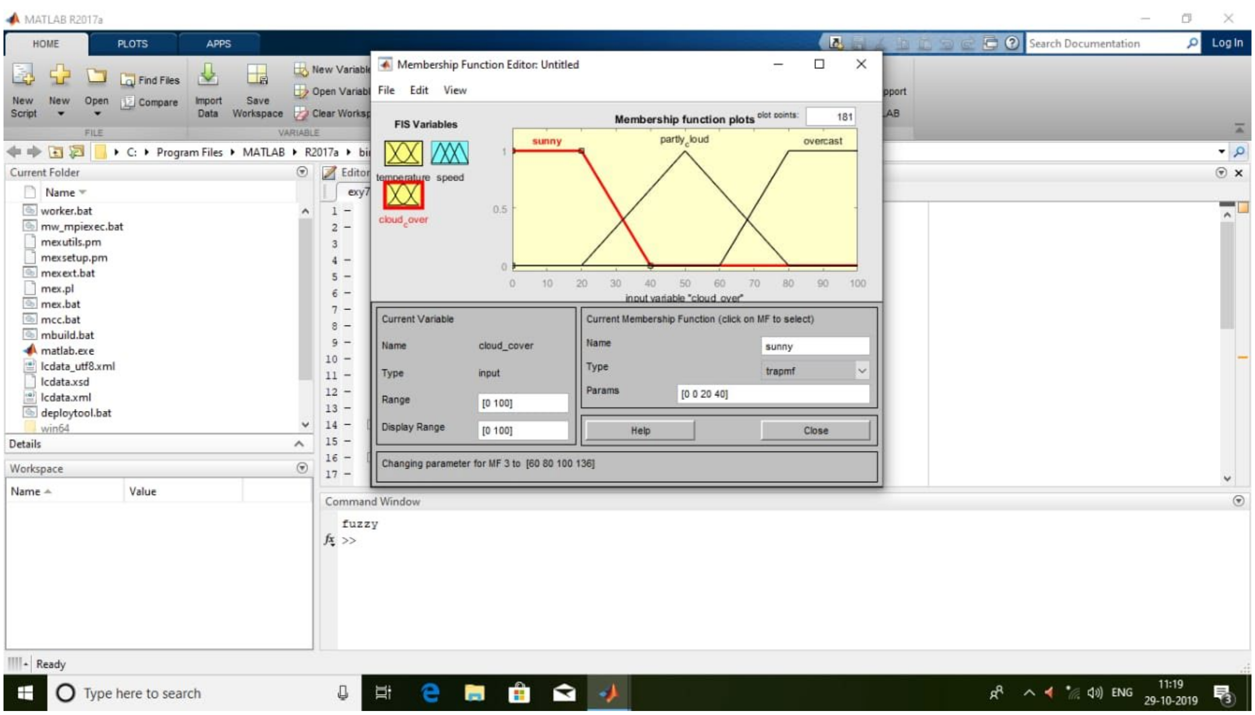
   11. Exit.

Sample Input and Output:

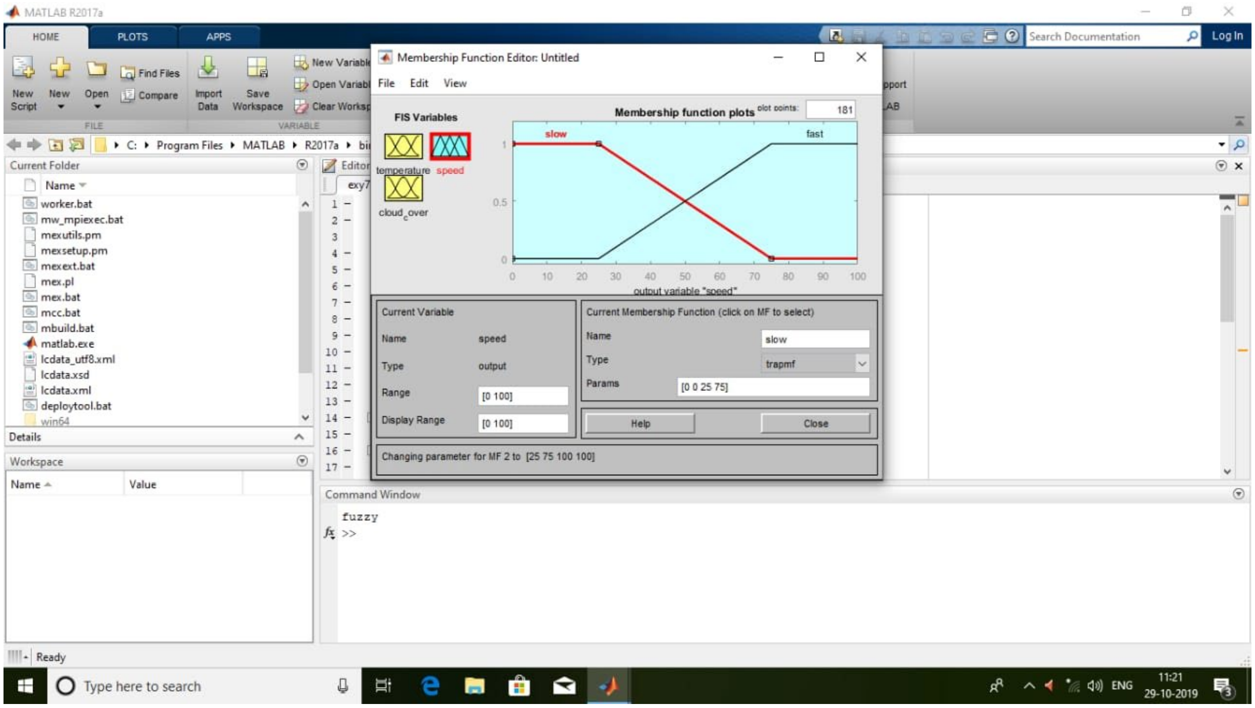
Membership functions for Temperature variable



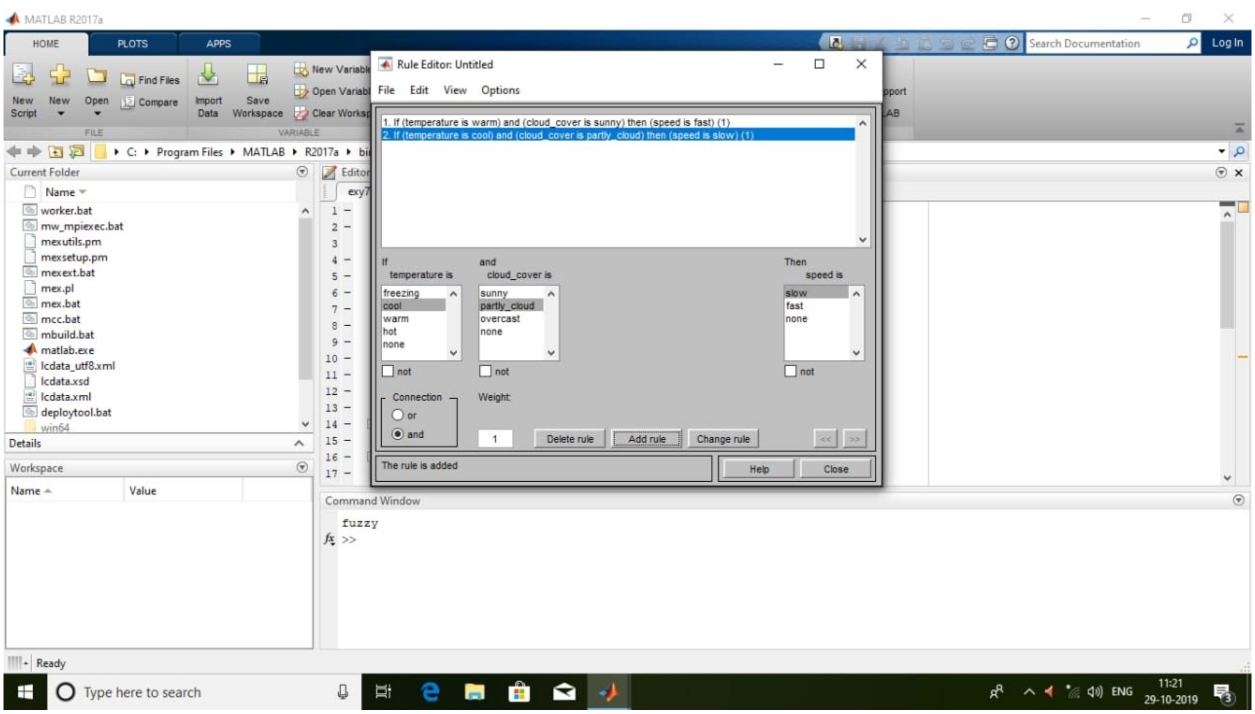
Membership functions for cloud over variable



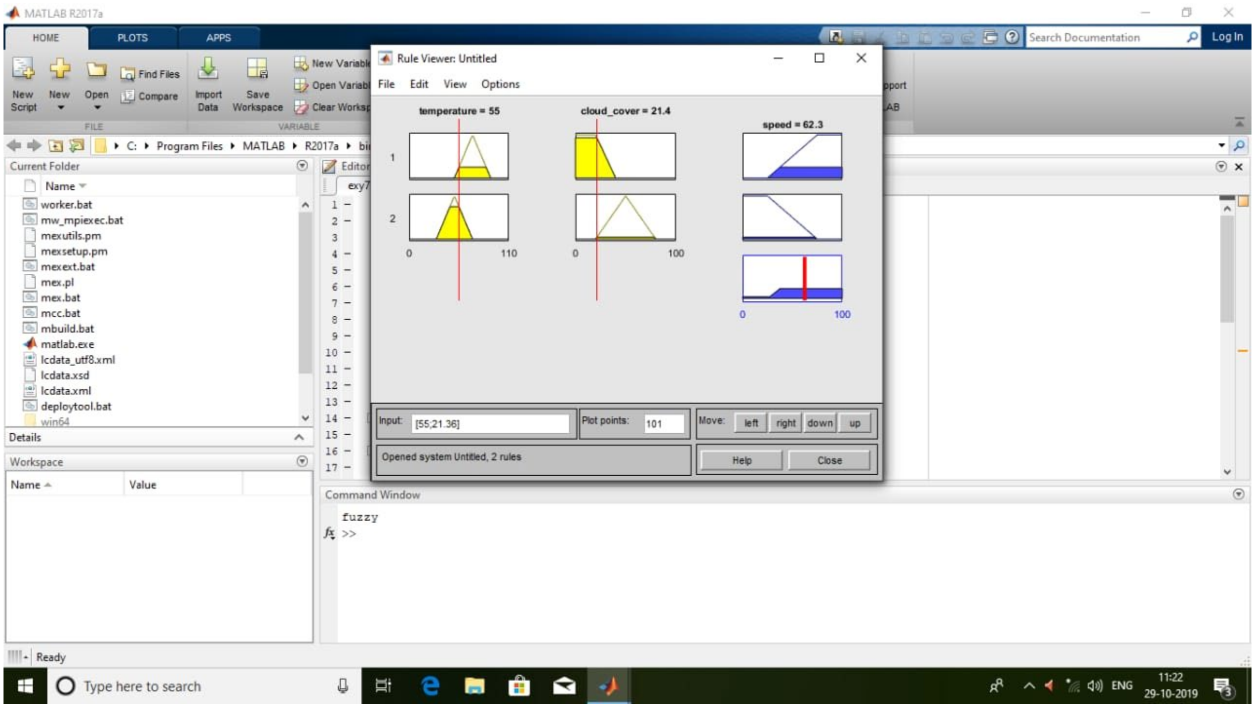
Membership functions for speed variable

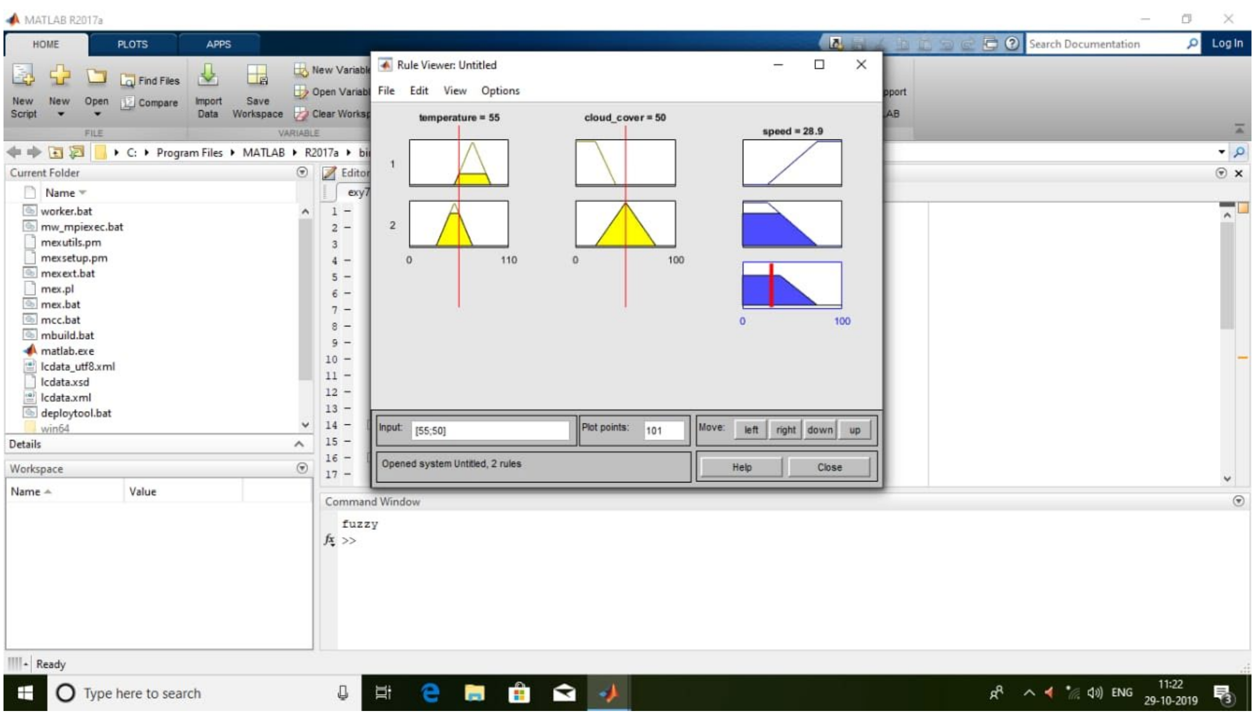


Created rules



Output





Result:

Thus a Fuzzy Inference System is implemented for temperature, cloud cover and speed using the given rules.

Simple Fuzzy Set Operations

Ex. No: 6
Date:

Aim:

To write a MATLAB program to find algebraic sum, algebraic subtraction, algebraic product, bounded sum, bounded subtraction and bounded product of two fuzzy sets.

Algorithm:

1. Read the values of the two fuzzy sets.
2. Perform the algebraic sum operation by,

   $A + B = (a + b) – (a * b)$

3. Perform the algebraic subtraction operation by,

   $A – B = (a + b`)$ where $b` = 1 - b$

4. Perform the algebraic product operation by,

   $A * B = (a * b)$

5. Perform the bounded sum operation by,

   $A \oplus B = \min [1, (a + b)]$

6. Perform bounded subtraction operation by,
   $A \ominus B = \max [0, (a - b)]$
7. Perform bounded product operation by,

   $A \odot B = \max [0, (a + b - 1)]$

8. Display the results

Program:

```
a= input('Enter the fuzzy set a' )

b= input('Enter the fuzzy set b')

c= a + b

d= a * b

as= c – d
```

```
e= 1 – b
ad= a + e
        f= a – b
bs= min (1, c)
bd= max (0, f)
        g= c – 1
bp= max (0,g)
disp('The algebraic sum')
disp(as)
disp('The algebraic difference')
disp(ad)
disp('The algebraic product')
disp(d)
disp('The bounded sum')
disp(bs)
disp('The bounded difference')
disp (bd)
disp('The bounded product')
disp(bp)
```

Output:

Enter fuzzy set a [1 0.5]

Enter fuzzy set b [0.4 0.2]

The algebraic sum

[1.0000   0.6000]

The algebraic difference

[1 0.9000]

The algebraic product

[0.4000 0.1000]

The bounded sum

[1.0000 0.7000]

The bounded difference

[0.6000 0.3000]

The bounded product

[0.4000 0]

Result:

Thus, a program to perform simple fuzzy set operations has been executed and successfully verified.

Using Hopfield network with no self connection

Ex. No: 7
Date:

Aim:

To write a MATLAB program to store the vector (1 1 1 0) and to find the weight matrix with no self connection using a discrete hopfield net with mistake in first and second component of vector that is (0 0 1 0).

Algorithm:

1. Make the initial activations of the net equal to given binary pattern
   x = (1 1 1 0).
2. Let tx = (0 0 1 0).
3. Initialize weight matrix using the formula

   w=(2*x'-1)*(2*x-1)

4. Set the diagonal values of weight matrix as 0.
5. Enter updation vector up=[4 2 1 3].
6. Perform the following for each i

   net input, Yin(up(i))=tx(up(i))+y*w(1:4,up(i))

7. Apply activation function to calculate output.
8. Test the network for convergence.

Program:

```
clc;
clear;
x=[1 1 1 0];
tx=[0 0 1 0];
w=(2*x'-1)*(2*x-1);
for i=1:4
    w(i,i)=0
end
con=1;
y=[0 0 1 0];
```

```
while con
    up=[4 2 1 3]
    for i=1:4
        yin(up(i))=tx(up(i))+y*w(1:4,up(i));
    if yin (up(i))>0
        y(up(i))=1;
    end
if y==x
    disp('Convergence has been obtained');
    disp('The converged output');
    disp(y);
    con=0;
end
end
```

Output:

up=4 2 1 3

Convergence has been obtained

The Converged Output

1 1 1 0

Result:

Thus the MATLAB program for using Hopfield network with no self connection has been successfully executed and the output is verified.

Generation of ANDNOT function using McCulloch-Pitts neural net

Ex. No: 8
Date:

Aim:

To write a MATLAB program to generate ANDNOT function using McCulloch-Pitts neural net.

Algorithm:

1. Initialize weights w1,w2 and threshold theta
2. Assign input values

   x1=[0 0 1 1]

   x2=[0 1 0 1]
3. Assign output Z =[0 0 1 0]
4. Initialize y = [0 0 0 0]
5. Repeat the following for each input
   i) Zin = x1*w1+x2*w2
   ii) If Zin > theta set y as 1 else 0
6. If y is not equal to Z update weights and repeat step 5
7. Display weights and threshold value

Program:

```
clear;

clc;

disp('Enter the weight');

w1=input('weight w1=');

w2=input('weight w2=');

disp('Enter threshold value');

theta=input('theta=');

y=[ 0 0 0 0];
```

```
x1=[0 0 1 1];
x2=[0 1 0 1];
Z  =[0 0 1 0];
Con=1;
While con
Zin=x1*w1+x2*w2;
for i=1:4
if  Zin(i)>=theta
y(i)=1;
else  y(i)=0;
end
end
disp('Output of net=');
disp(y);
if  y==z
con=0;
else
disp('Net is not learning enter another set of weights and threshold  value');
   w1=input('Weight  w1=');
   w2=input('Weight  w2=');
theta=input('theta=');
end
end
disp('McCulloch Pitts Net for ANDNOT function');
disp('Weights  of  neuron');
```

```
disp(w1);

disp(w2);

disp('Threshold value=');

disp(theta);
```

Sample Input and Output:

Enter the weights
Weight w1=1
Weight w2=1

Enter threshold value
Theta=1
Output of net= 0 1 1 1
Net is not learning

Enter another set of weights and threshold value
Weight w1=1
Weight w2= -1
Theta=1

Output of net= 0 0 1 0
McCulloch Pitts Net for ANDNOT function
Weights of neuron
1
-1
Threshold value=1

Result:

Thus, a MATLAB program to generate ANDNOT function using McCulloch-Pitts neural net has been successfully executed and the output is verified.

Finding weight matrix and bias of HebbNet to classify two dimensional input patterns

Ex. No: 9
Date:

Aim:

To write a MATLAB program to find the weight matrix and bias of Hebbnet in bipolar to classify two dimensional input patterns with their targets given below.

'*' indicates a '+' and '.' Indicates a '-'

```
*****        *****
*....        *....
*****        *****
*....        *....
*****        *....
```

Algorithm:

1. Create a single layer neural network with 25 neuron.
2. Set the initial weight and bias to zero.
3. Calculate the weights using

   $w_i(new)=w_i(old)+x_i*t$

   and bias using

   b(new)=b(old)+t(i)

4. Display the final weight matrix and bias.

Program:

```
% Hebb Net to classify 2d input patterns

 clear;
clc;

 %Input Pattern

E=[1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 1]

F=[1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 1 1 -1 -1 -1 -1 1 -1 -1 -1 -1]

x(1,1:25)=E;

 x(2,1:25)=F;

 w(1:25)=0;
```

```
t=[1 -1];
 b=0;
for i=1:2
        w=w+x(I,1:25)*t(i);
         b=b+t(i);
end
disp(‘ Weight matrix:’);
disp(w);
disp(‘Final Bias:’);
disp(b);
```

Output:

Weight matrix

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2

Final Bias

0

Result:

Thus a MATLAB program to find the weight matrix and bias to classify two dimensional input patterns in bipolar using Hebb Net have been executed and verified successfully.

Perceptron net for AND function with bipolar inputs and targets

Ex. No: 10
Date:

Aim:

To write a MATLAB program to implement AND function with bipolar input and output using Perceptron net.

Algorithm:

1. Initialize weight and bias to 0
2. Accept learning rate, alpha and threshold, theta
3. For each input calculate yin = b+x(1)*w(1)+x(2)*w(2)
4. Apply activation function
5. If calculated output ≠ target output
   i) update weight and bias
   ii) Go to step 3
6. Display final weight matrix and bias value

Program:

```
% Perceptron for AND function
clear;
clc;

x=[1  1  - 1 -1; 1 -1  1  - 1];

t=[1  -1  -1  -1];

w=[0 0];

b=0;

alpha=input('Enter Learning rate=');

theta=input('Enter Threshold Value=');

con = 1;

epoch = 0;

while con

con=0;

for i=1:4
```

```
yin=b+x(1,i)*w(1)+x(2,i)*w(2);
if yin>theta
        y=1;
end
if  yin<=theta  &  yin>= -theta
        y=0;
end
if yin < -theta
        y = -1;
end
if  y-t(i)
con=1;
for j=1:2
w(j)=w(j)+alpha*t(i)*x(j,i);
end
b=b+alpha*t(i);
end
epoch=epoch+1;
end
disp('Perceptron for AND Function');
disp('Final Weight Matrix');
disp(w);
disp('Final Bias');
disp(b);
```

Sample Input and Output:

Enter Learning rate = 1

Enter Threshold Value =0.5

Perceptron for AND Function

Final Weight Matrix

1 1

Final Bias

-1

Result:

Thus, a MATLAB program for Perception net for an AND function with bipolar inputs have been written and verified successfully.

Finding weight matrix of hetero associative neural net for mapping of vectors

Ex. No: 11
Date:

Aim:

To write a MATLAB program to calculate the weights using Hetero-associative neural net for mapping of vectors.

S1	S2	S3	S4	t1	t2
1	1	0	0	1	0
1	0	1	0	1	0
1	1	1	0	0	1
0	1	1	0	0	1

Algorithm:

1. Enter input and output vector x and t
2. Initialize weight matrix.
3. Update weight matrix by using the formula

   $w_i(new)=w_i(old)+x_i*t$

4. Display the calculated weight.

Program:

```
%Hetero-associative neural net for mapping input vectors to output vectors
clear;
clc;
x=[1 1 0 0 ; 1 0 1 0 ; 1 1 1 0 ; 0 1 1 0];
t=[1 0 ; 1 0 ; 0 1 ; 0 1];
w=zeros(4,2);
for i=1:4
w=w+x(i,1:4)'*t(i,1:2);
end
disp(' Weight matrix:');
disp(w);
```

Output:

Weight matrix

2	1
1	2
1	2
0	0

Result:

Thus a MATLAB program to calculate the weight matrix using hetero associative neural net for mapping of vectors has been executed and verified successfully.

Generation of XOR function using back propagation algorithm

Ex. No: 12
Date:

Aim:

To write a MATLAB program to train and test the back propagation neural network for the generation of XOR function.

Algorithm:

1. Enter the input vector x=[0 0 1 1 ; 0 1 0 1] and target vector y=[0 1 1 0].
2. Train the network by using the function newff().
3. Set the epoch and learning rate value.
4. Test the network by using the trained network.
5. Display the result.

Program:

Function to Training the Back Propagation neural network:

```
function[net]=trainBPN(x,y)

[n,i]=size(x);

[m,o]=size(y);

net = newff(minmax(x),[i,10,m],{'tansig','tansig','purelin'},'trainlm')

net .trainparam.epoch =50;

net.trainparam.lr=0.2;

net=train(net,x,y);

r=sim(net,x);

return;
```

Function for testing the Back Propagation neural network:

```
function[v]=testBPN(x,net)

v=sim(net,x);

return;
```

Output:

Matlab command for executing the function:

```
>>x=[0 0 1 1 ; 0 1 0 1]

>>y=[0 1 1 0]
```

Calling the function to train the network:

```
>>[net]=trainBPN(x,y)
```

Calling the function to test the network:

```
>>testBPN(x,net)

ans=

 -0.0000     1.0000     1.0000    -0.0000
```

Result:

Thus a MATLAB program to train and test the back propagation neural network for the generation of XOR function has been executed successfully and the output is verified.