**Practical No: - 1**

**Aim: Write a program to implement the single layer perceptron.**

**Code:**

#include<stdio.h>

#include<conio.h>

void main()

{

int In,b[20],wt[20],er,i,n,m;

clrscr();

printf("\n enter size");

scanf("\n %d",&m);

printf("\n enter input values ");

for(i=0;i<m;i++)

{

scanf("\n %d",&b[i]);

}

printf("enter value of weight");

for(i=0;i<m;i++)

{

scanf("\n %d",&wt[i]);

}

printf("enter desired weight");

scanf("\n %d",&er);

do

{

I=0;

for(i=0;i<m;i++)

{

I=b[i]\*wt[i]+In;

}

printf("\n input is %d",In);

if(In>er)

{

for(i=0;i<m;i++)

{

wt[i]=wt[i]-1;

}

}

else

{

for(i=0;i<m;i++)

{

wt[i]=wt[i]+1;

}

}

printf("\nto continue: yes=1\n no=0\n");

scanf("\n %d",&n);

}

while(n==1);

printf("\n final weights are");

for(i=0;i<m;i++)

{

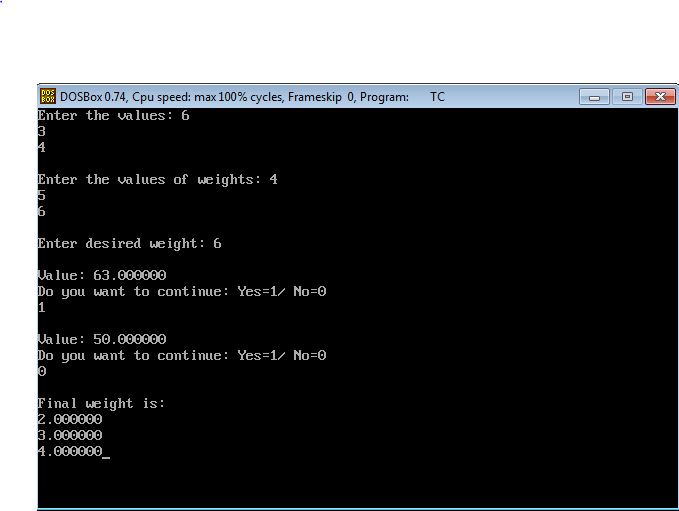
printf("\n %d",wt[i]);

}

getch();

}

**Output:**



**Practical No: - 2**

**Aim: Write a program to implement backpropagation.**

#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

float d,dw2,err,i,j,D,v0[2],w1[4],w2[2],v2[2],o[2],n,m,fout,finput;

int x;

clrscr();

printf("enter value\n");

for(i=0;i<2;i++)

{

scanf("\n%f",&v0[i]);

}

printf("\nenter weights");

for(i=0;i<=3;i++)

{

scanf("\n%f",&w1[i]);

}

printf("\nenter weight for 2nd layer");

for(i=0;i<2;i++)

{

scanf("\n%f",&w2[i]);

}

printf("\ndesired weight");

scanf("\n%f",&D);

do

{

v2[1]=v0[1]\*w1[1]+v0[2]\*w1[2];

v2[2]=v0[1]\*w1[3]+v0[2]\*w1[4];

o[1]=1/(1+exp(-v2[1]));

o[2]=1/(1+exp(-v2[2]));

finput=o[1]\*w2[1]+o[2]\*w2[2];

fout=1/(1+exp(-finput));

d=D-fout;

err=fout\*(1-fout)\*d;

dw2=err\*fout;

for(i=0;i<2;i++)

{

w2[i]=w2[i]+dw2;

}

printf("\ndiff is %f",d);

printf("\nweight of final layer is");

for(i=0;i<2;i++)

{

printf("\x%f",w2[i]);

}

printf("\ndo u want to continue:\tyes:1 \tno:0");

scanf("\n\x\x\x%d",&x);

}

while(x==1);

for(i=0;i<2;i++)

{

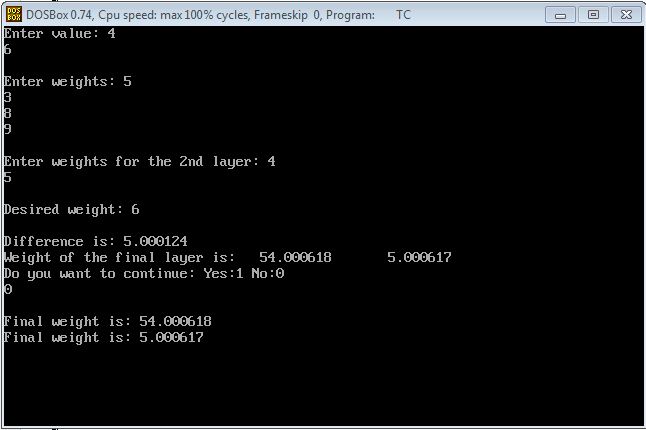
printf("\nfinal weight is %f",w2[i]);

}

getch();

}

**OUTPUT:**

****

**Practical No : - 3**

**Aim : Write a program to implement basic fuzzy set Operation using fuzzy header file.**

**Code :**

#include<stdio.h>

#include<conio.h>

#include"aakash.h"

#include<stdlib.h>

void main()

{

int i,j,k,ch;

float a[5]={1.0,0.9,0.5,0.8,1.0},b[5]={0.2,0.4,0.9,0.6,0.1},p,r,ans,temp;

clrscr();

for(i=0;i<=4;i++)

{

printf("%f",a[i]);

printf("\t%f",b[i]);

printf("\n");

}

printf("1.Union 2.Intersection 3.Complement ");

printf("Enter the value:");

scanf("%d",&ch);

printf("\n\n");

switch(ch)

{

case 1:

{

for(i=0;i<=4;i++)

{

p=a[i];

r=b[i];

ans=uni(p,r);

printf("\n %f",ans);

printf("\t");

}

break;

}

case 2:

{

for(i=0;i<=4;i++)

{

p=a[i];

r=b[i];

ans=inter(p,r);

printf("\n %f",ans);

printf("\t");

}

break;

}

case 3:

{

for(i=0;i<=4;i++)

{

p=a[i];

r=b[i];

ans=comple(p);

temp=comple(r);

printf("\n %f",ans);

printf("\n %f",temp);

printf("\t ");

}

break;

}

}

getch();

}

**Header File : ”aakash.h”**

float uni(float p,float q)

{

if(p>=q)

{

printf("\nunion of p & q:",&p);

return p;

}

else

{

printf("\n union of p & q :",&q);

return q;

}

}

float inter(float w,float x)

{

if(w<=x)

{

printf("\nintersection of w & x :",&w);

return w;

}

else

{

printf("\nintersection of w & x",&x);

return x;

}

}

float comple(float e)

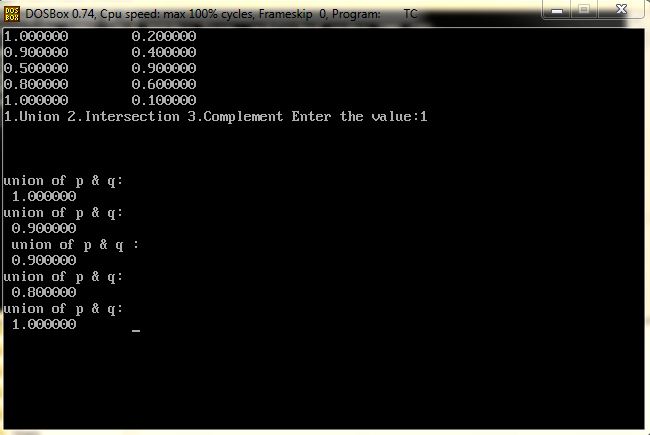
{

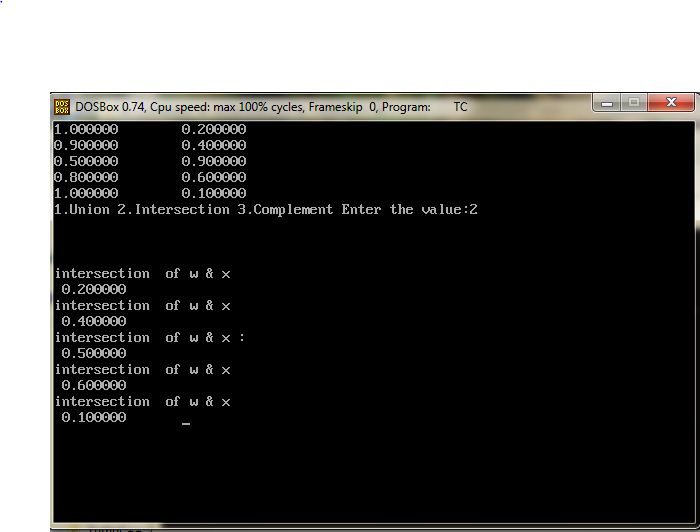
float ans;

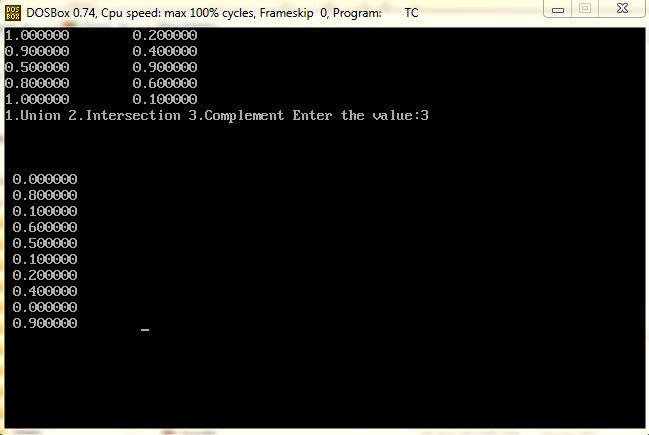
ans=(1.0-e);

return ans;

}

**Output: **

****

****

**Practical No : - 4**

**Aim : Write a program to implement fuzzy operation on relation set.**

**Code :**

#include<stdio.h>

#include<conio.h>

#include "Header.H"

void main()

{

int i,j,k,ch;

float a[3][3]={0.1,0.2,0.7,0.4,0.5,0.73,0.7,0.11,0.9},b[3][3]={0.7,0.12,0.3,0.41,0.5,0.3,0.70,0.8,0.9},max,min,ans[3][3],temp,bi1;

clrscr();

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

printf("\t%.2f",a[i][j]);

}

printf("\n");

}

printf("\n");

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

printf("\t%.2f",b[i][j]);

}

printf("\n");

}

printf("\n 1.Union 2.Intersection 3.Composition max-min 4.Composition max-star");

printf("\n Enter the Value:");

scanf("%d",&ch);

printf("\n\n");

switch(ch)

{

case 1: {

printf("\n\tUnion of Matrix :\n");

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

ans[i][j]=uni(a[i][j],b[i][j]);

printf("\t%.2f",ans[i][j]);

}

printf("\n");

}

break;

}

case 2:{

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

ans[i][j]=inter(a[i][j],b[i][j]);

printf("\t%.2f",ans[i][j]);

}

printf("\n");

}

break;

}

case 3:{

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

max=-1;

for(k=0;k<3;k++)

{

max=uni(max,inter(a[i][k],b[k][j]));

}

ans[i][j]=max; }

}

}

printf("\nMAX MIN of a & b is");

printf("\n");

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

printf("\t%.2f",ans[i][j]);

}

printf("\n");

}

break;

case 4: {

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

max=-1;

for(k=0;k<3;k++)

{

max=uni(max,(a[i][k]\*b[k][j]));

}

ans[i][j]=max; }

}

}

printf("\nMAX MIN of a & b is");

printf("\n");

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

printf("\t%.2f",ans[i][j]);

}

printf("\n");

}

break;

}

getch();

}

**Header File :**

float uni(float p,float q)

{

if(p>=q)

{

return p;

}

else

{

return q;

}

}

float inter(float w,float x)

{

if(w>=x)

{

return x;

}

else

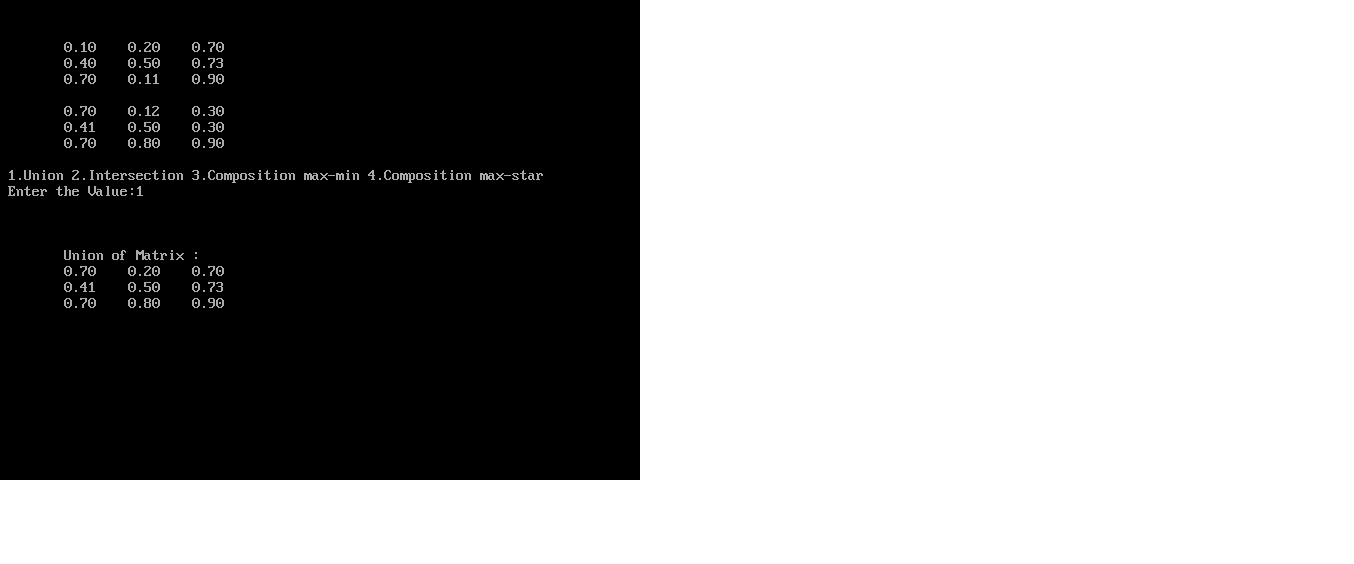
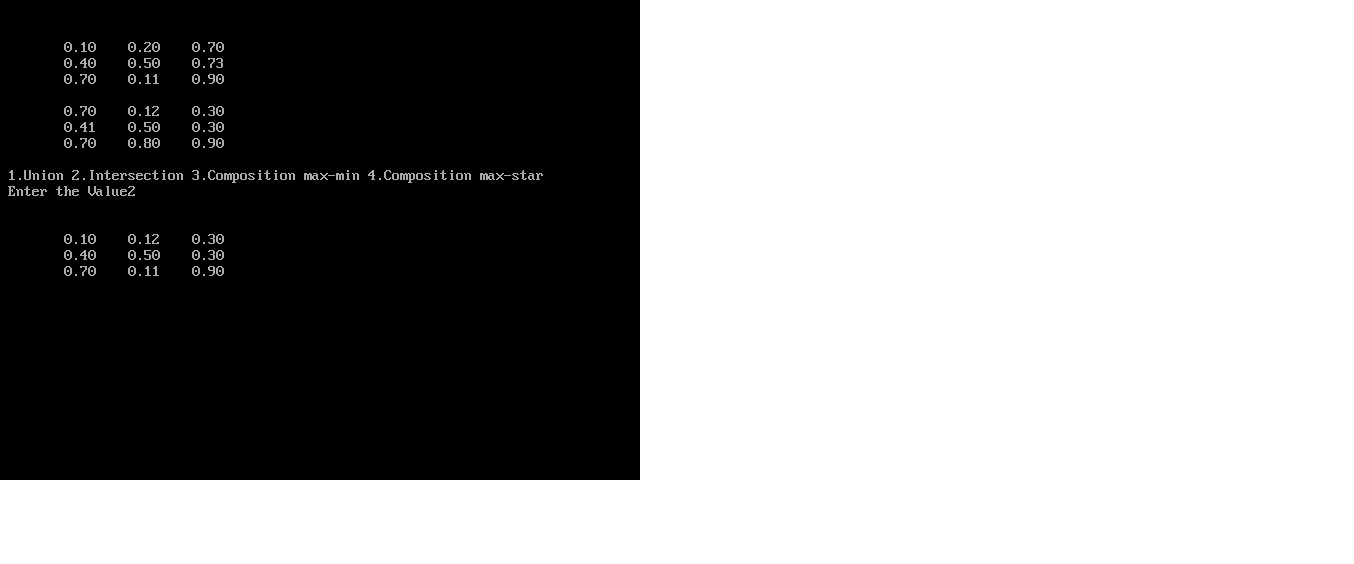
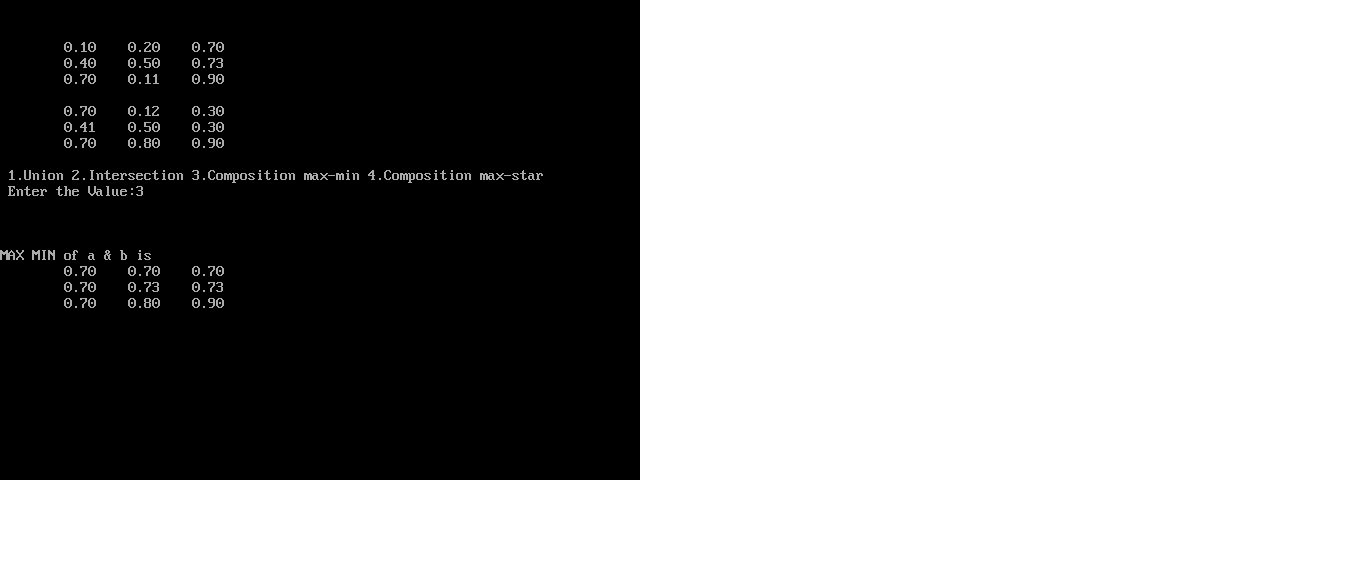
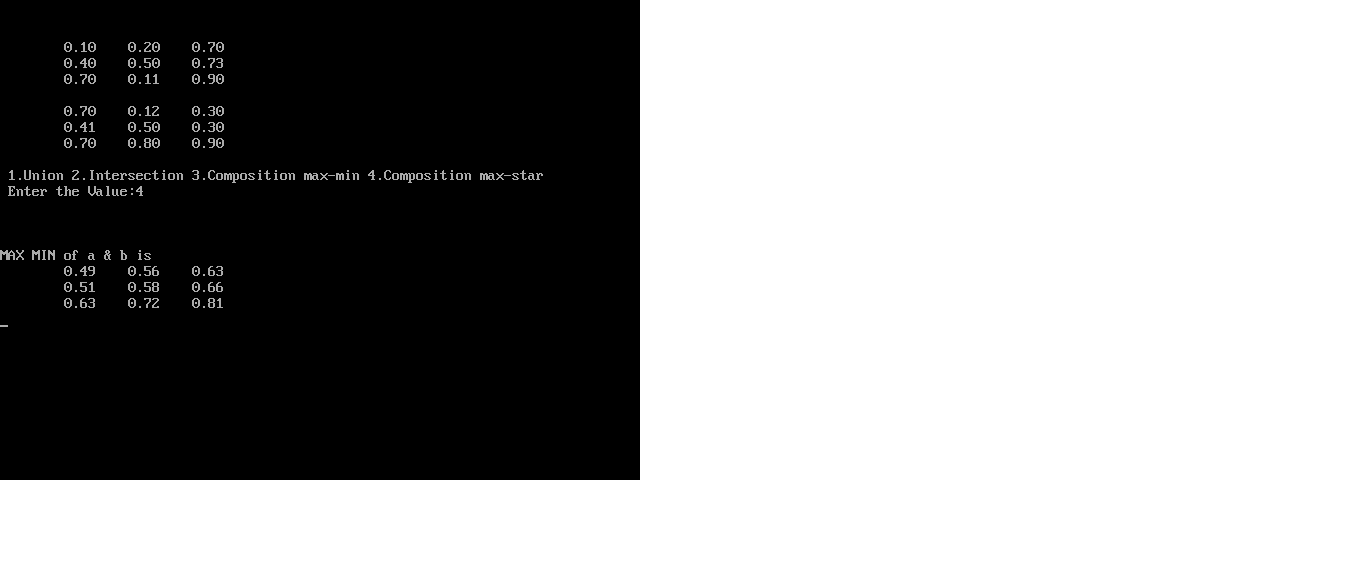
{

return w;

}

}

**Output:**

****

**Practical No: - 5**

**Aim: Create a fuzzy logic system for Washing Machine using MATLAB.**

1. **INTRODUCTION TO WASHING MACHIN’S WORKING :**

A washing machine basically works in three cycles. These are washing, rinsing and spinning. Different mechanical parts work together to carry out each cycle. In the operation water enters the washing machine through little pipes. The motor starts to turn the washing machine's inner tub. The water drains through these holes in the inner tub. The water gets there due to the spinning of the tub and centrifugal force. These components have to be present in any type of washing machine: inner wash tub (involves the adding of water and detergent mixture to the clothes), agitator (which enhances the action of detergents enzymes on clothes), motor (which accelerates the process of washing), outer wash tub (where discharge the detergent particles trapped in the washed fabric), drain tube (which is called the spin cycle, is to remove out as much water as possible from the wet clothes) and power parts which modulates the speed to smoothen the start-up effect.

Before starting to use the washing machine, one needs to consider a few elements such as the load of the clothes, temperature of the water, rinse cycles and their durations. The machine fills the tub with water after the clothes are filled in the tub. The machine stirs the clothes around the agitator and after some time, the washer drains the water and spins the clothes to remove most of the water. It then refills and agitates the clothes some more so as to rinse out the soap and then drains and spins again. In each of the four corners of the machine, there is a mechanism that works a like a disc brake. The part attached to the washer frame is a spring which squeezes two pads against the metal plate that is attached to the black frame. One can see where the pads have polished the plate from the movement during vibration.

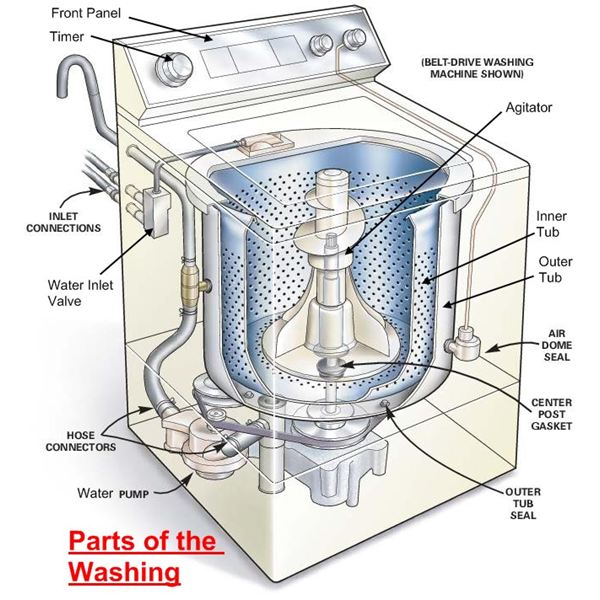
**The Parts of an Air Conditioner**

**1) Water inlet control valve**: Near the water inlet point of the washing there is water inlet control valve. When you load the clothes in washing machine, this valve gets opened automatically and it closes automatically depending on the total quantity of the water required. The water control valve is actually the solenoid valve.

**2) Water pump**: The water pump circulates water through the washing machine. It works in two directions, re-circulating the water during wash cycle and draining the water during the spin cycle.

**3) Tub**: There are two types of tubs in the washing washing machine: inner and outer. The clothes are loaded in the inner tub, where the clothes are washed, rinsed and dried. The inner tub has small holes for draining the water. The external tub covers theinner tub and supports it during various cycles of clothes washing.

**4) Agitator or rotating disc**: The agitator is located inside the tub of the washing machine. It is the important part of the washing machine that actually performs the cleaning operation of the clothes. During the wash cycle the agitator rotates continuously and produces strong rotating currents within the water due to which the clothes also rotate inside the tub.



In some washing machines, instead of the long agitator, there is a disc that contains blades on its upper side. The rotation of the disc and the blades produce strong currents within the water and the rubbing of clothes that helps in removing the dirt from clothes.

**5) Motor of the washing machine**: The motor is coupled to the agitator or the disc and produces it rotator motion. These are multispeed motors, whose speed can be changed as per the requirement. In the fully automatic washing machine the speed of the motor i.e. the agitator changes automatically as per the load on the washing machine.

**6) Timer**: The timer helps setting the wash time for the clothes manually. In the automatic mode the time is set automatically depending upon the number of clothes inside the washing machine.

**7) Printed circuit board (PCB)**: The PCB comprises of the various electronic components and circuits, which are programmed to perform in unique ways depending on the load conditions (the condition and the amount of clothes loaded in the washing machine). They are sort of artificial intelligence devices that sense the various external conditions and take the decisions accordingly. These are also called as fuzzy logic systems. Thus the PCB will calculate the total weight of the clothes, and find out the quantity of water and detergent required, and the total time required for washing the clothes. Then they will decide the time required for washing and rinsing.

**8) Drain pipe**: The drain pipe enables removing the dirty water from the washing that has been used for the washing purpose.

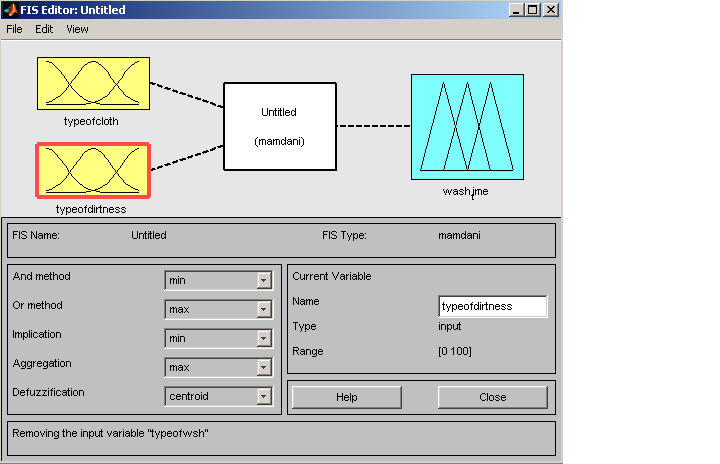
**STEPS TO IMPLEMENT FIS FOR WASHING MACHINE IN MATLAB**

1. **Select the no of inputs:** in washing machine system we have taken three inputs. The three inputs are as following :

Type of cloth

Type of dirtness

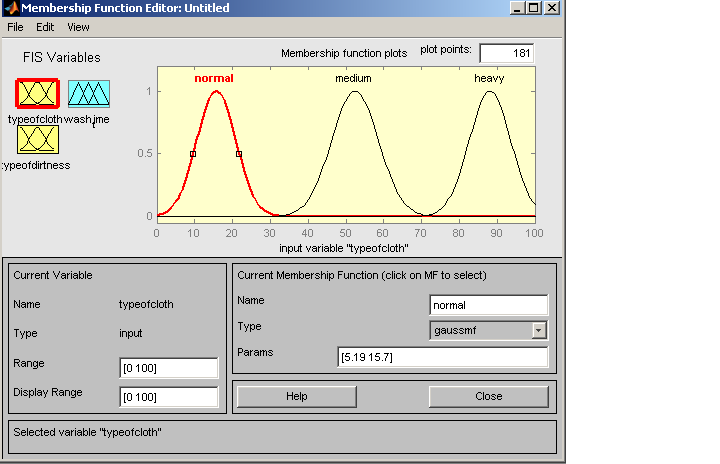
Water-level [17 60 litre]”

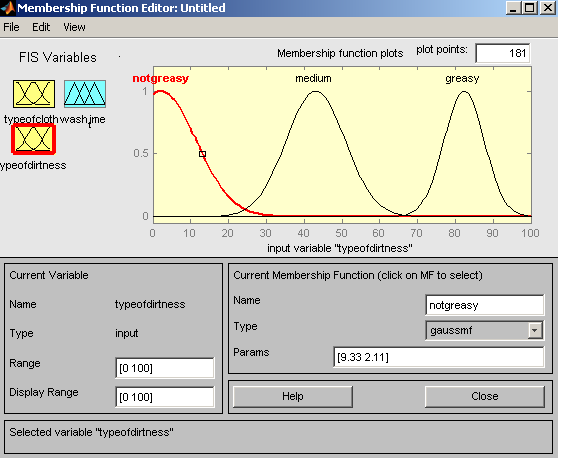


1. **Membership functions for all inputs and its linguistic variables:**

PROGRAM [10 47]:

|  |  |
| --- | --- |
| *linguistic variable* | *membership function* |
| Normal | Guassmf |
| Medium | Guassmf |
| Heavy | Guassmf |





|  |  |
| --- | --- |
| *linguistic variable* | *membership function* |
| Not greasy | Gauss2mf |
| Medium | Gauss2mf |
| Greasy | Gauss2mf |

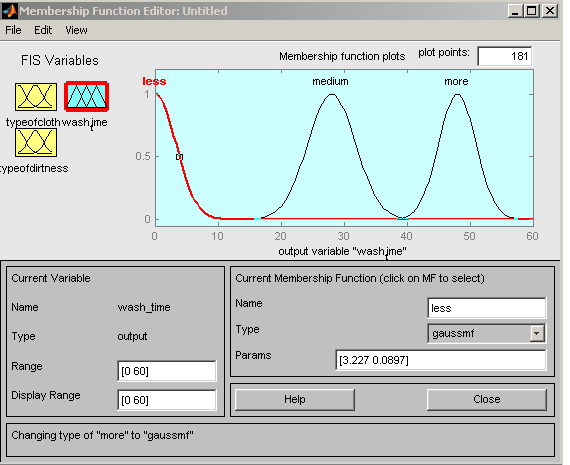
1. **Select the no of output :** in this system the no of outputs are two as following :

Washtime

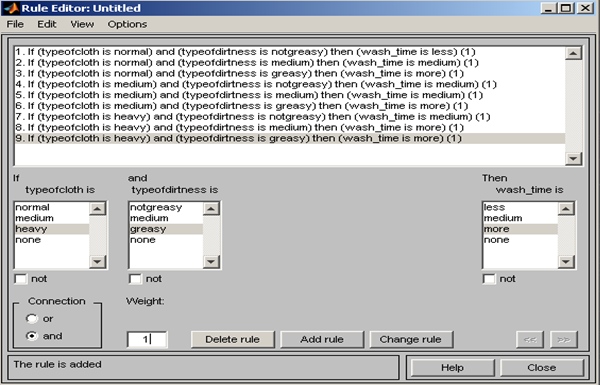
1. **Membership functions for all outputs and its linguistic variables :**

TIME[8-60 sec]:

|  |  |
| --- | --- |
| *linguistic variable* | *membership function* |
| Less | Guassmf |
| Medium | Guassmf |
| More | Guassmf |

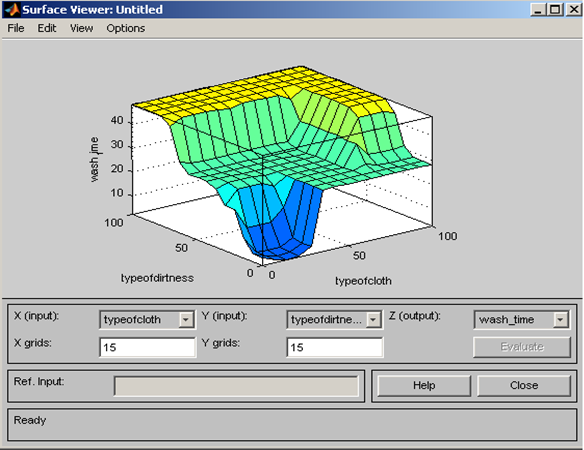


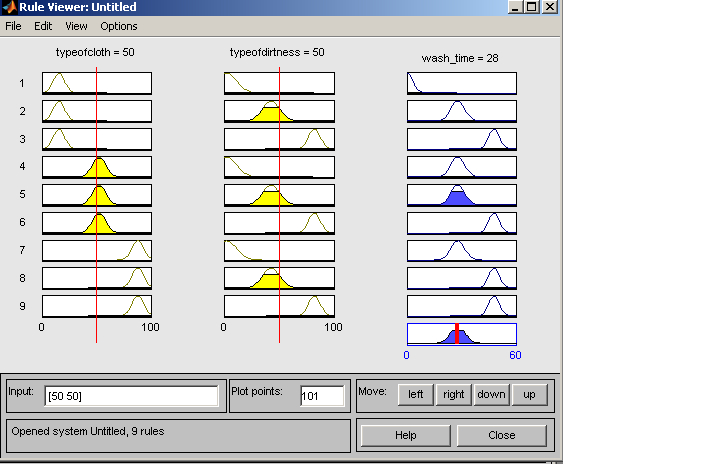
1. **Rules for Washing Machine Fuzzy inference system**



1. **Output of rules in rule viewer/ surface viewer :**

In this screen shot the inputs are water-level and Program and output shown is the time.





In above screen shot the inputs are type of cloth is 50, type of dirtness is 50 and wash then the output of wash time is 28. This output is based on seventh rule defined in the FIS.

**In this FIS system :**

**Implication method is : Min**

**Aggregation method is : Max**

**Defuzzification method is : Centroid**

Thus the whole FIS is design in the matlab.

**PROGRAM:**

function varargout = gui1(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @gui1\_OpeningFcn, ...

'gui\_OutputFcn', @gui1\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

function gui1\_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);

function varargout = gui1\_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function slider1\_Callback(hObject, eventdata, handles)

x = get(handles.slider1,'Value')

set(handles.text1,'String',num2str(x))

function slider1\_CreateFcn(hObject, eventdata, handles)

usewhitebg = 1;

if usewhitebg

set(hObject,'BackgroundColor',[.9 .9 .9]);

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

function slider2\_Callback(hObject, eventdata, handles)

y = get(handles.slider2,'Value')

set(handles.text2,'String',num2str(y))

function slider2\_CreateFcn(hObject, eventdata, handles)

usewhitebg = 1;

if usewhitebg

set(hObject,'BackgroundColor',[.9 .9 .9]);

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

function pushbutton1\_Callback(hObject, eventdata, handles)

p = get(handles.slider1,'Value');

q = get(handles.slider2,'Value');

a=newfis('wash\_machine1');

a = addvar(a,'input','x',[0 100]);

a = addmf(a,'input',1,'A1','trimf',[0 0 50]);

a = addmf(a,'input',1,'A2','trimf',[0 50 100]);

a = addmf(a,'input',1,'A3','trimf',[50 100 100]);

a = addvar(a,'input','y',[0 100]);

a = addmf(a,'input',2,'B1','trimf',[0 0 50]);

a = addmf(a,'input',2,'B2','trimf',[0 50 100]);

a = addmf(a,'input',2,'B3','trimf',[50 100 100]);

a = addvar(a,'output','z',[0 60]);

a = addmf(a,'output',1,'C1','trimf',[0 8 12]);

a = addmf(a,'output',1,'C2','trimf',[8 12 20]);

a = addmf(a,'output',1,'C3','trimf',[12 20 30]);

a = addmf(a,'output',1,'C4','trimf',[30 40 50]);

a = addmf(a,'output',1,'C5','trimf',[50 60 70]);

rulelist=[ ...

1 1 1 1 1

2 2 3 1 1

3 3 5 1 1

1 2 3 1 1

1 3 3 1 1

2 1 2 1 1

2 3 4 1 1

3 1 2 1 1

3 2 4 1 1 ];

a=addrule(a,rulelist);

z = evalfis([p q], a)

set(handles.text3,'String',num2str(z))

set(handles.slider4,'Value',z)

function slider3\_Callback(hObject, eventdata, handles)

function slider3\_CreateFcn(hObject, eventdata, handles)

usewhitebg = 1;

if usewhitebg

set(hObject,'BackgroundColor',[.9 .9 .9]);

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

function slider4\_Callback(hObject, eventdata, handles)

function slider4\_CreateFcn(hObject, eventdata, handles)

usewhitebg = 1;

if usewhitebg

set(hObject,'BackgroundColor',[.9 .9 .9]);

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

**Practical No: - 6**

**Aim: Create a fuzzy logic system for Air conditioner using MATLAB.**

**PROBLEM SUMMARY:**

The task of dehumidification and temperature decrease goes hand in hand in case of conventional AC. Once target temperature is reached AC seizes to function like a dehumidifier. Also complex interactions between user preferences, actual room temperature and humidity level are very difficult to model mathematically. But in this work this limitation has been taken into cogitation and overcome to a great extent using fuzzy logic to represent the intricate influences of all these parameters. The optimal limits of comfort zone, typically marked at a temperature of 25°C and dew point 11°C, are used as the targets. Conventional AC system controls humidity in its own way without giving the users any scope for changing the set point for the targeted humidity unlike the scope it offers to change the set point for the targeted temperature through a thermostat. This causes a significant level of flexibility as well as efficiency loss especially in hot and humid countries like India. For instance at higher humidity level (say at dew point 18°C) an occupant may perceive same comfort level at 22ºC as he would perceive at 26ºC at dew point 15°C. This translates to huge energy and monitory saving in terms of reduced compressor/fan duty cycle. In the developed scheme, the sensor captured temperature, user temperature preference and humidity readings are fuzzified. These are used to decide the fuzzy qualifier, which is decoded into a crisp value that in turn controls different aspects of the AC. In the problem dew point (Td) temperature is used to measure humidity instead of relative humidity (RH), this is because RH is a function of both temperature and moisture content while Td is a function of moisture content only. Hence it becomes very easy to model comfort level on the basis of Td. Human reaction to different levels of dew point



**INTRODUCTION TO AIR CONDITIONER’S WORKING :**

Air conditioners use refrigeration to chill indoor air, taking advantage of a remarkable physical law: When a [liquid](http://science.howstuffworks.com/liquid-info.htm) converts to a [gas](http://science.howstuffworks.com/gas-substance-info.htm) (in a process called **phase conversion**), it absorbs heat. Air conditioners exploit this feature of phase conversion by forcing special chemical compounds to evaporate and condense over and over again in a closed system of coils.

The compounds involved are **refrigerants** that have properties enabling them to change at relatively low temperatures. Air conditioners also contain fans that move warm interior air over these cold, refrigerant-filled coils. In fact, central air conditioners have a whole system of ducts designed to funnel air to and from these serpentine, air-chilling coils.

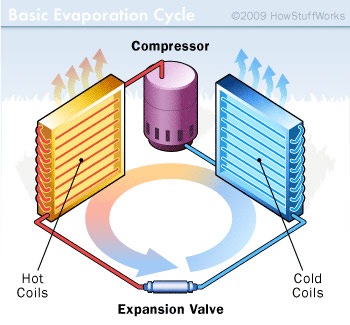
When hot air flows over the cold, low-pressure **evaporator coils**, the refrigerant inside absorbs heat as it changes from a liquid to a gaseous state. To keep cooling efficiently, the air conditioner has to convert the refrigerant gas back to a liquid again. To do that, a compressor puts the gas under high pressure, a process that creates unwanted heat. All the extra heat created by compressing the gas is then evacuated to the outdoors with the help of a second set of coils called **condenser coils**, and a second fan. As the gas cools, it changes back to a liquid, and the process starts all over again. Think of it as an endless, elegant cycle: liquid refrigerant, phase conversion to a gas/ heat absorption, compression and phase transition back to a liquid again.

It's easy to see that there are two distinct things going on in an air conditioner. Refrigerant is chilling the indoor air, and the resulting gas is being continually compressed and cooled for conversion back to a liquid again. On the next page, we'll look at how the different parts of an air conditioner work to make all that possible.

**The Parts of an Air Conditioner**

Let's get some housekeeping topics out of the way before we tackle the unique components that make up a standard air conditioner. The biggest job an air conditioner has to do is to cool the indoor air. That's not all it does, though. Air conditioners monitor and regulate the air temperature via a thermostat. They also have an onboard filter that removes airborne particulates from the circulating air. Air conditioners function as dehumidifiers. Because temperature is a key component of relative humidity, reducing the temperature of a volume of humid air causes it to release a portion of its moisture. That's why there are drains and moisture-collecting pans near or attached to air conditioners, and why air conditioners discharge water when they operate on humid days. Still, the major parts of an air conditioner manage refrigerant and move air in two directions: indoors and outside:

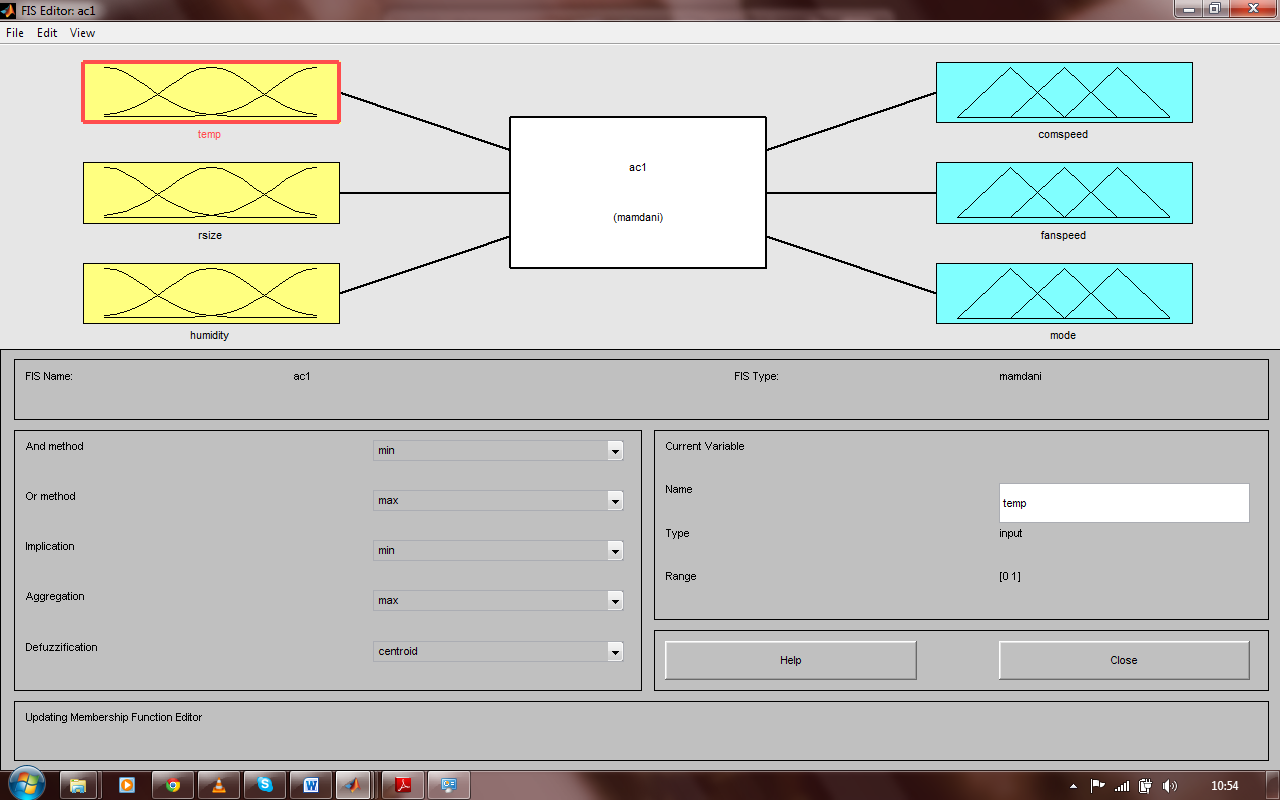
* 1. **Evaporator -** Receives the liquid refrigerant
  2. **Condenser -** Facilitates heat transfer
  3. **Expansion valve -** regulates refrigerant flow into the evaporator
  4. **Compressor -** A pump that pressurizes refrigerant



The cold side of an air conditioner contains the evaporator and a fan that blows air over the chilled coils and into the room. The hot side contains the compressor, condenser and another fan to vent hot air coming off the compressed refrigerant to the outdoors. In between the two sets of coils, there's an **expansion valve**. It regulates the amount of compressed liquid refrigerant moving into the evaporator. Once in the evaporator, the refrigerant experiences a pressure drop, expands and changes back into a gas. The **compressor** is actually a large electric pump that pressurizes the refrigerant gas as part of the process of turning it back into a liquid. There are some additional sensors, timers and valves, but the evaporator, compressor, condenser and expansion valve are the main components of an air conditioner.

Although this is a conventional setup for an air conditioner, there are a couple of variations you should know about. Window air conditioners have all these components mounted into a relatively small metal box that installs into a window opening. The hot air vents from the back of the unit, while the condenser coils and a fan cool and re-circulate indoor air. Bigger air conditioners work a little differently: Central air conditioners share a control thermostat with a home's heating system, and the compressor and condenser, the hot side of the unit, isn't even in the house. It's in a separate all-weather housing outdoors. In very large buildings, like hotels and hospitals, the exterior condensing unit is often mounted somewhere on the roof.

**STEPS TO IMPLEMENT FIS FOR AIR CONDITIONER IN MATLAB :**



1. **Select the no of inputs :** in air conditioner system we have taken three inputs. The three inputs are as following :

Temperature

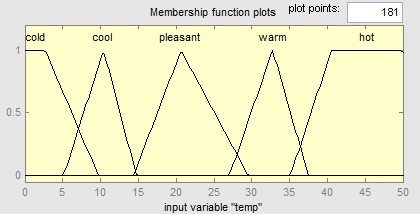
Room size

Humidity

1. **Membership functions for all inputs and its linguistic variables :**

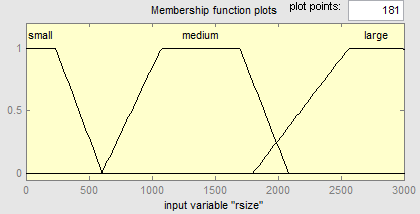
Temperature (0 to 50 degree celcius) :

|  |  |
| --- | --- |
| **Linguistic variables** | **Membership function** |
| Cold | Trapmf |
| Cool | Trimmf |
| Pleasant | Trimmf |
| Warm | Trimmf |
| Hot | Trapmf |



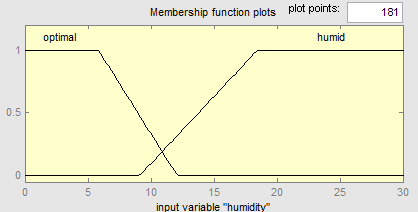
Room size (0 to 3000 square feet) :

|  |  |
| --- | --- |
| Linguistic variable | Membership function |
| Small | Trapmf |
| Medium | Trapmf |
| Large | Trapmf |



Humidity (0 to 30 grams per cubic meter) :

|  |  |
| --- | --- |
| **Linguistic variable** | **Membership function** |
| Optimal | Trapmf |
| Humid | Trapmf |



1. **Select the no of output :** in this system the no of outputs are three as following :

Compressor speed

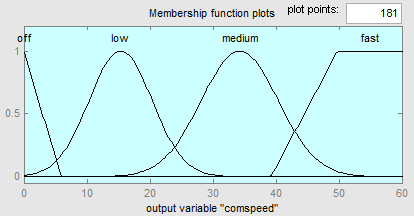
Fan speed

Mode

1. **Membership functions for all outputs and its linguistic variables :**

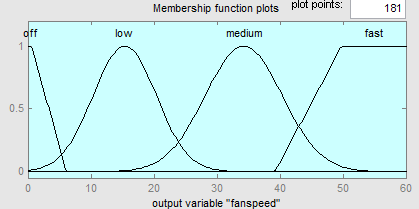
Compressor speed (0 to 60 rotation par minute) :

|  |  |
| --- | --- |
| **Linguistic variables** | **Membership function** |
| Off | Trimmf |
| Low | Gaussmf |
| Medium | Gaussmf |
| Fast | Trapmf |



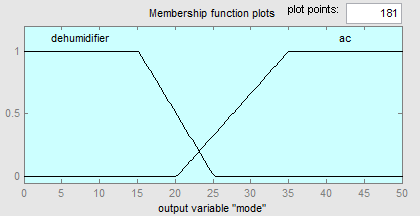
fan speed :

|  |  |
| --- | --- |
| Linguistic variables | Membership function |
| Off | Trimmf |
| Low | Gaussmf |
| Medium | Gaussmf |
| Fast | Trapmf |

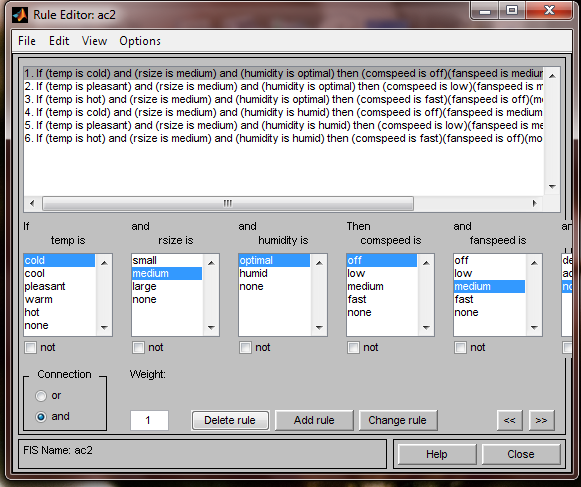


Mode :

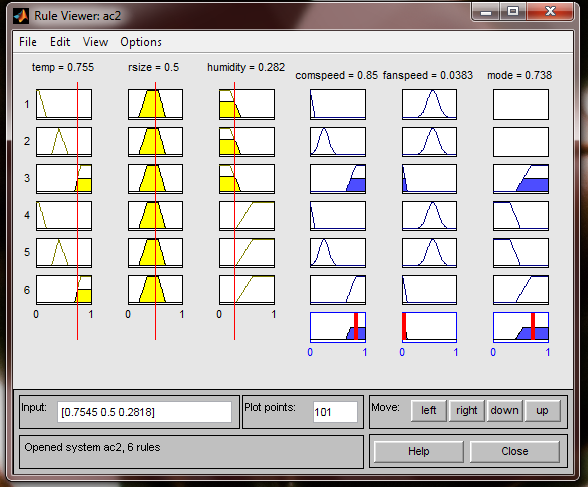
|  |  |
| --- | --- |
| **Linguistic variables** | **Membership function** |
| Dehumidifier | Trapmf |
| Ac | Trapmf |



1. **Rules for air conditioner FIS system :**



1. **Output of rules in rule viewer :**In this screen shot the inputs are temperature is hot room size is medium and humidity is optimal the n the output of compressor speed is fast fan speed is off and mode is AC. this output is based on six rules defined in the FIS.



**In this FIS system:**

**Implication method is: Min**

**Aggregation method is: Max**

**Defuzzification method is: Centroid**

**Air Conditioner that uses Fuzzy inference system:**

an industrial air conditioner designed by Mitsubishi uses 25 heating rules and 25 cooling rules. A temperature sensor provides input, with control outputs fed to an inverter, a compressor valve, and a fan motor. Compared to the previous design, the fuzzy controller heats and cools five times faster, reduces power consumption by 24%, increases temperature stability by a factor of two, and uses fewer sensors.

**Real life example Mitsubishi air conditioner:**Example: Air-conditioning System by Mitsubishi

**Problem description:** Industrial air-conditioning system that shall be able to react flexibly to changing ambient conditions

**Realization:**

* 50 rules
* 6 linguistic variables
* Resolution: 8 bit
* Input variables: room temperature, wall temperature and temporal evaluation of this signals

**Development:**

* 4 days to create the prototype
* 20 days for testing and integration
* 80 days for optimization with real test objects
* Implementation as pure software solution on standard microcontroller

**Results:**

* Reduction of starting processes down to 40 percent of the standard solution
* Sustaining of the temperature even with interference factors (like open window, etc.) substantially improved
* Fewer sensors required
* Established energy saving by testing: 24 percent



**Future directions :**

The paper simplified the problem by not allowing AC to reverse operation and act like a heat pump and humidifier. By eliminating these restrictions we can go for an all weather AC that would work in almost any part of the world. Also by adding infra red sensors to detect presence of occupants we can go one step ahead in user satisfaction. These sensors can aggregate data such as occupant location and body temperature. These data can further help control temperature, humidity and fin direction automatically for maximum comfort while reducing energy consumption. Application of neural networks and genetic algorithm will allow the controller to adapt to individual user, room environment and weather. An AC that will be “intelligent” in true sense!

**Practical No: - 7**

**AIM:** **Write a program to implement the genetic algorithm.**

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

int a[10][10],visited[10],n,cost=0;

void get()

{

int i,j;

printf("Enter No. of Cities: ");

scanf("%d",&n);

printf("\nEnter Cost Matrix: \n");

for( i=0;i<n;i++)

{

printf("\n Enter Elements of Row # : %d\n",i+1);

for( j=0;j<n;j++)

scanf("%d",&a[i][j]);

visited[i]=0;

}

printf("\n\nThe cost list is:\n\n");

for( i=0;i<n;i++)

{

printf("\n\n");

for( j=0;j<n;j++)

printf("\t%d",a[i][j]);

}

}

void mincost(int city)

{

int i,ncity;

visited[city]=1;

printf("%d –>",city+1);

ncity=least(city);

if(ncity==999)

{

ncity=0;

printf("%d",ncity+1);

cost+=a[city][ncity];

return;

}

mincost(ncity);

}

int least(int c)

{

int i,nc=999;

int min=999,kmin;

for(i=0;i<n;i++)

{

if((a[c][i]!=0)&&(visited[i]==0))

if(a[c][i]<min)

{

min=a[i][0]+a[c][i];

kmin=a[c][i];

nc=i;

}

}

if(min!=999)

cost+=kmin;

return nc;

}

void put()

{

printf("\n\nMinimum cost:");

printf("%d",cost);

}

void main()

{

clrscr();

get();

printf("\n\nThe Path is:\n\n");

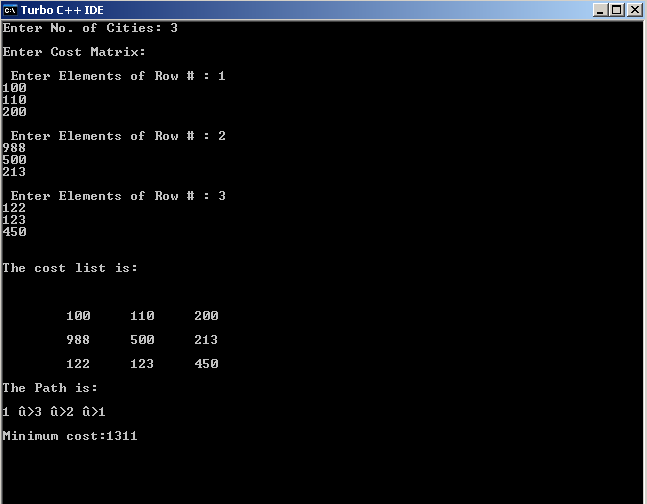
mincost(0);

put();

getch();

}

**Output:**



**Practical No: - 8**

**AIM: -** **Case study- Optical Character Reorganisation is using MATLAB.**

**What is OCR?****

Next to keypunching, Optical Character Recognition is the oldest data entry technique in existence. Long before the first key-to-disk system of CRT was used; Optical Character Readers were entering data in commercial and government EDP installations.

The popularity of OCR has been increasing each year with the advent of fast microprocessors providing the vehicle for vastly improved recognition techniques. This can be shown in OCR wands now reading print that, over 10 years ago, large batch readers would have rejected. There have also been tremendous improvements in increasing both effective read rates and accuracy. Data Entry through OCR is faster, more accurate, and generally more efficient than keystroke data entry. Desktop OCR scanners can read typewritten data into a computer at rates up to 2400 words per minute!

**How Does OCR Work?**

There are two basic methods used for OCR: Matrix matching and feature extraction. Of the two ways to recognize characters, matrix matching is the simpler and more common.

Matrix Matching compares what the OCR scanner sees as a character with a library of character matrices or templates. When an image matches one of these prescribed matrices of dots within a given level of similarity, the computer labels that image as the corresponding ASCII character.

Feature Extraction is OCR without strict matching to prescribed templates. Also known as Intelligent Character Recognition (ICR), or Topological Feature Analysis, this method varies by how much "computer intelligence" is applied by the manufacturer. The computer looks for general features such as open areas, closed shapes, diagonal lines, line intersections, etc. This method is much more versatile than matrix matching. Matrix matching works best when the OCR encounters a limited repertoire of type styles, with little or no variation within each style. Where the characters are less predictable, feature, or topographical analysis is superior.

**OCR Fonts**

What is a font? A font is the term given to a set of characters, usually 0 - 9, A through Z, and a few special characters. Each character within a font will have a defined reproducible size and shape. For OCR, these are defined by ANSI, the American National Standards Institute.

OCR fonts, or characters, that can be read by the lower speed, lower cost systems we are discussing here require well defined character shapes that are very reproducible and designed to be both machine and human readable. These unique and well defined character sets allow for greater accuracy.

[**OCR Scanners**](http://www.dataid.com/productsocr.htm)

OCR reading devices are fundamentally classified with two categories, Text Input and Data Capture.

Text input devices are page readers or document scanners that scan entire documents or large portions of documents. The source data is entered with the intention of someone editing it during or after it is scanned. Text input devices have varying degrees of automation from hand fed to having automatic feeding, reading, sorting, and stacking capabilities.

Data Capture devices are designed to capture repetitive data and to perform formatting functions on the data as it is being entered. The data delivered from the scanner to the computer must be very accurate because it is entered without the intention of being edited later, so accuracy must be higher than text input.

**Program:**

warning off

clc, close all, clear all

imagen=imread('TEST\_1.jpg');

imshow(imagen);

title('INPUT IMAGE WITH NOISE')

if size(imagen,3)==3

imagen=rgb2gray(imagen);

end

threshold = graythresh(imagen);

imagen =~im2bw(imagen,threshold);

imagen = bwareaopen(imagen,30);

word=[ ];

re=imagen;

fid = fopen('text.txt', 'wt');

load templates

global templates

num\_letras=size(templates,2);

while 1

[fl re]=lines(re);

imgn=fl;

[L Ne] = bwlabel(imgn);

for n=1:Ne

[r,c] = find(L==n);

n1=imgn(min(r):max(r),min(c):max(c));

img\_r=imresize(n1,[42 24]);

letter=read\_letter(img\_r,num\_letras);

word=[word letter];

end

fprintf(fid,'%s\n',word);

word=[ ];

if isempty(re)

break

end

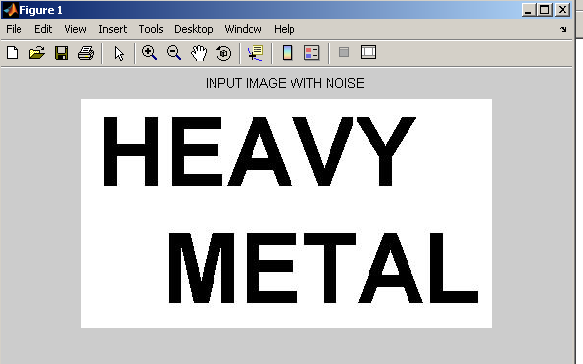
end

fclose(fid);

winopen('text.txt')

clear all

**Input File:**



**Output: -**

