SC Lab/20221219

Instructions :

Assigned Task= RollNo % 11

1.Write the program and show the output online.
2.Email the program and corresponding output as a single PDF file with the filename convention (Group-RollNo-Fullname)
just after the end of the lab class at 5.10pm

0	a. Write functions to generate the following parameterized fuzzy membership functions and visualize them for different parameter values: Triangular MF, Trapezoidal MF, Gaussian MF, Generalized Bell MF, Sigmoidal MF. b. Any Neural Network Application of your own c.Any GA Application of your own
1	a. Write functions to implement following fuzzy complement operations on continuous membership functions and visualize them for different parameter values: Classical fuzzy complement, Sugeno's fuzzy complement, Yager's fuzzy complement. b. Any Neural Network Application of your own c.Any GA Application of your own
2	a. Write functions to implement following fuzzy intersection operations (T-norms) on continuous membership functions and visualize them for different parameter values: Minimum, Algebraic product, Bounded product, Drastic product. b. Any Neural Network Application of your own c.Any GA Application of your own
3	a. Write a function to compute the max-min composition of two fuzzy relations. b. Any Neural Network Application of your own

	c.Any GA Application of your own
4	a. Write a function to compute the max-product composition of two fuzzy relations. b. Any Neural Network Application of your own c.Any GA Application of your own
5	a. Demonstrate the effect of contrast intensification on a fuzzy membership function. b. Any Neural Network Application of your own c.Any GA Application of your own
6	a Write functions for implementing cylindrical extension of a 1D membership function and projection of a 2D membership function. Demonstrate the results visually. b. Any Neural Network Application of your own c.Any GA Application of your own
7	a. Write programs to solve three unconstrained function optimization problems using Genetic Algorithm. b.Any Neural Network Application of your own c.Any Fuzzy Application of your own
8	a. Write programs to solve three function optimization problems with constraint satisfaction using Genetic Algorithm. b.Any Neural Network Application of your own c.Any Fuzzy Application of your own
9	a. Plot the graphs of different activation functions. b.Any Fuzzy Application of your own c.Any GA Application of your own
10	a. Implement AND, OR, XOR Gate using Single Layer Perceptron Neural Network. b.Any Fuzzy Application of your own c.Any GA Application of your own

Name:M Gourav

Roll No:23

SIC NO:190310077

LAB TEST

Q1.

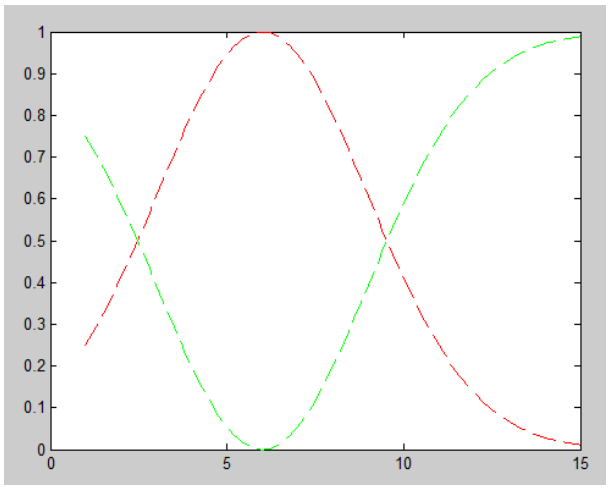
Answer:A

Complement

```
y1 = [];
y2 = [];
y3 = [];
for x=1:0.1:15
    y1(end+1) = gaussian(6,3,x);
    y2(end+1) = trapezoid(3,6,8,10,x);
    temp1 = gaussian(6,3,x);
    temp2 = trapezoid(3,6,8,10,x);
    y3(end+1) = min(temp1,temp2);
end

x = 1:0.1:15;
plot(x,y1,'--r');
hold on
plot(x,y2,'--g');
plot(x,y3,'Color',[0,0,0]);
hold off
```

Output:



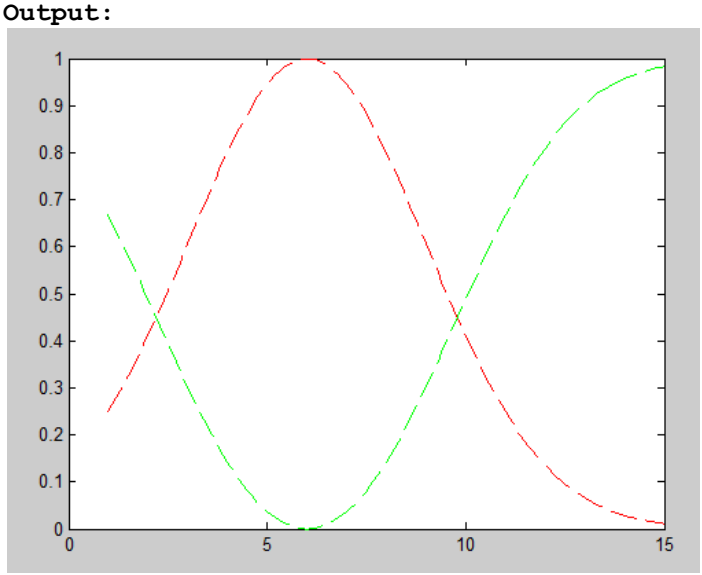
Sugeno Complement

```
y1 = [];
y2 = [];
s = 0.5;
for x=1:0.1:15
    a = gaussian(6,3,x);
    y1(end+1) = a;
    y2(end+1) = (1-a)/(1+a*s);
end

x = 1:0.1:15;
plot(x,y1,'--r');
hold on;
plot(x,y2,'--g');

hold off;
```

Output:



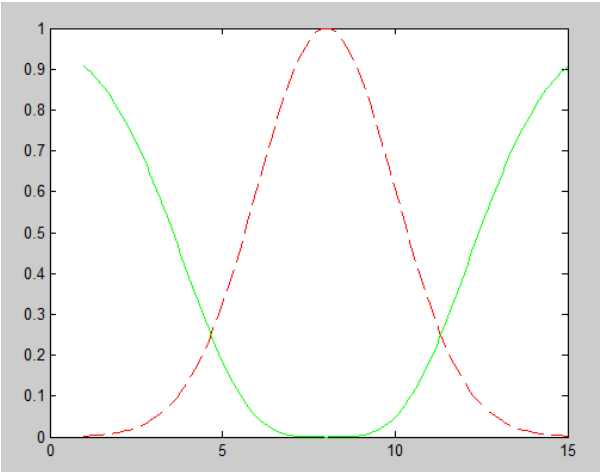
Yagers Complement

```
y1 = [];
y2 = [];
w = 0.5;
for x=1:0.1:15
    a = gaussian(8,2,x);
    y1(end+1) = a;
    y2(end+1) = power(1-power(a,w),(1/w));
end

x = 1:0.1:15;
plot(x,y1,'--r')
hold on;
plot(x,y2,'g');

hold off;
```

Output:



Gaussian

```
function mvalue = gaussian(a,sig,x)
mvalue = exp(-0.5*power((x-a)/sig,2));
end

end
```

Trapezoid

```
function membership = trapezoid(a,b,c,d,x)
if ((x<=a) || (x>=d))
    membership = 0;
elseif ((x>a) && (x<b))
    membership = (x-a)/(b-a);
elseif ((x>=b) && (x<=c))
    membership = 1;
elseif ((x>c) && (x<d))
    membership = (d-x)/(d-c);
end
```

```
end
```
----------xxxxxxxx----------xxxxxxxxxxxxx----------

Answer:B

```
disp('Enter the weights');

w1=input('Weight w1=');

w2=input('Weight w2=');

disp('Enter the threshold value');

theta=input('theta=');

y=[0 0 0 0];

x1=[1 1 0 0];

x2=[1 0 1 0];

z=[1 0 0 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('McCulloh Pitts Net for ANDNOT function');
disp('Weights of neuron');
disp(w1);
disp(w2);
disp('Threshold value=');
disp(theta);
```

Output

```
Enter the weights
Weight w1=1
Weight w2=1
Enter the threshold value
theta=2
Output of net=
     1     0     0     0

McCulloh Pitts Net for ANDNOT function
Weights of neuron
     1

     1

Threshold value=
     2
```

----------XXXXXXXXXX----------XXXXXXXXXX----------

Answer:C

Genetic Algo

```
n = 10;
xl= 0;
xu = 31;
binary_size = ceil(log2(xu-xl+1));
in_pop = round(rand(n,binary_size));
d_vals = [];
for i=1:1:n
```

```
    power = 1;
    value = 0;
    for j=binary_size:-1:1
        value = value + (in_pop(i,j)*power);
        power = power*2;
    end
    d_vals(end+1) = value;
end
prob = [];
for i=1:1:n
    prob(end+1) = fitness_values(i)/total;
end
com_prob = [];
com_prob(end+1) = prob(1);
for i=2:1:n
    com_prob(end+1) = com_prob(i-1)+prob(i);
end
sel_pop = zeros(1,n);
new_pop = zeros(n,binary_size);
for i=1:1:n
    random_value = rand;
    index = 1;
    for j=1:1:n
        if random_value <= com_prob(j)
            index = j;
            break;
        end
    end
    sel_pop(i) = d_vals(index);
    for k=1:1:binary_size
        new_pop(i,k) = in_pop(index,k);
    end
end
%Mutation
disp('Mutation')
Pm = 0.001;
new_pop = mutation(new_pop, n, binary_size, Pm);
disp(new_pop);
```

Mutation

```
function new_pop = mutation(new_pop, n, binary_size, Pm)
for i=1:n
    for j=1:binary_size
        if rand<Pm
            new_pop(i, j) = rem(new_pop(i,j)+1, 2);
        end
    end
end
end
```

Output:

```
>> genetic_algo
Mutation
     1     1     1     1     0
     1     1     1     1     0
     0     0     1     0     0
     1     1     1     1     0
     0     1     0     1     0
     0     1     0     1     1
     0     1     0     0     1
     1     1     1     1     0
     0     0     1     0     0
     1     1     1     1     0

>>
```

Soft Computing Lab Test

2 a)T Norm

```
tmin = [];

tap = [];

tbp = [];

tdp = [];

ya = [0,0.7,0.5,0.63];

xa = [4,5,6,7];

yb = [0.6, 0.4,0.2,0.33];

xb = [4,5,6,7];

for i=1:1:4

    tmin(end+1) = tminfun(ya(i),yb(i));

    tap(end+1) = tapfun(ya(i),yb(i));

    tbp(end+1) = tbpfun(ya(i),yb(i));

    tdp(end+1) = tdpfun(ya(i),yb(i));

end


plot(xa,tmin,'--b');

hold on

plot(xa, tap, 'r');

plot(xa, tbp, 'g');

plot(xa, tdp)

hold off
```

Tmin Function

```
function m = tminfun(a,b)
m = min(a,b);
end
```

Algebric Product function

```
function m = tapfun(a,b)
m = a*b;
end
```

Bounded Product Function

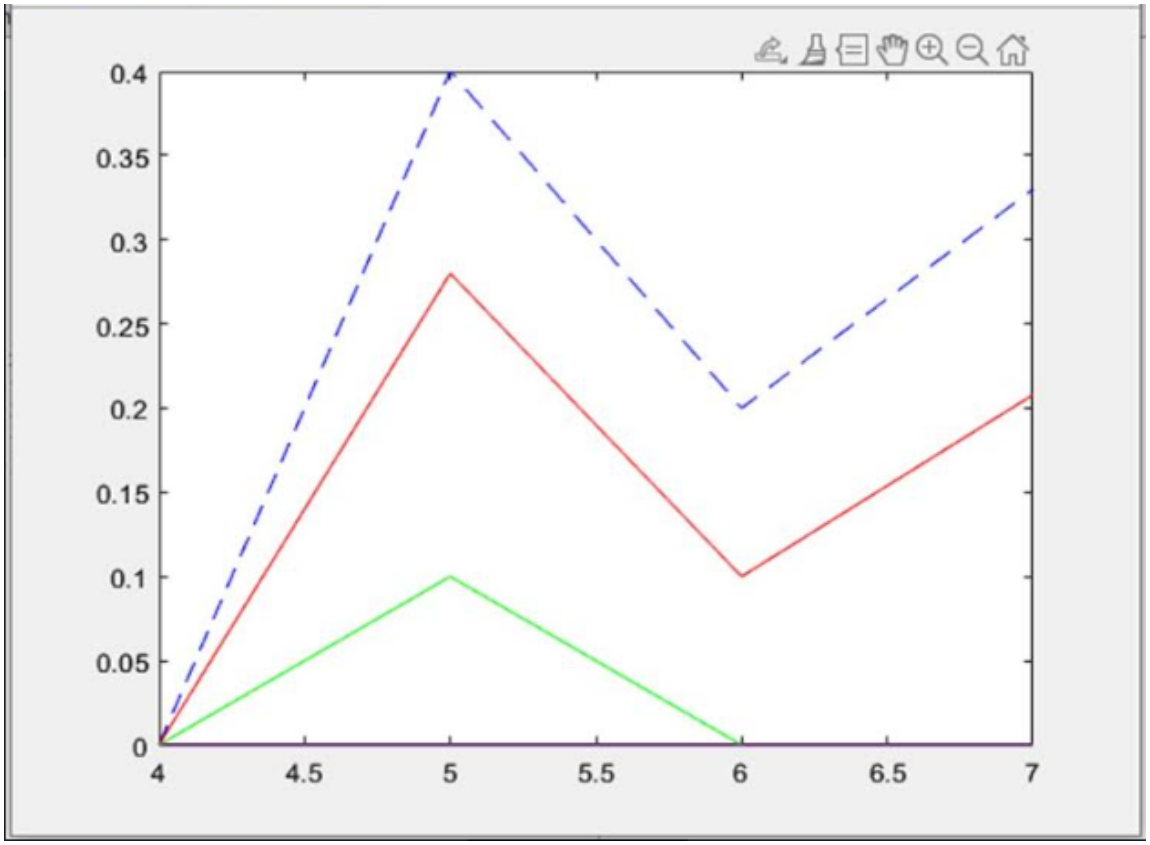
```
function m = tbpfun(a,b)
m = max(0, a+b-1);
end
```

Drastic Product Function

```
function m = tdpfun(a,b)
if (b==1)
    m = a;
elseif (a==1)
    m = b
else
    m = 0;
end
```

end

Output:-



2 b) Neural Network

```
disp('Enter the weights');
w1=input('Weight w1=');
w2=input('Weight w2=');
disp('Enter the threshold value');
theta=input('theta=');
y=[0 0 0 0];
x1=[1 1 0 0];
x2=[1 0 1 0];
z=[1 0 0 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('McCulloh Pitts Net for ANDNOT function');
disp('Weights of neuron');
disp(w1);
disp(w2);
disp('Threshold value=');
disp(theta);
```

Output:-

```
>> mccullochpits
Enter the weights
Weight w1=
1
Weight w2=
1
Enter the threshold value
theta=
1
Output of net=
     1     1     1     0

Net is not learning. Enter another set of weights and threshold value
```

```
Net is not learning. Enter another set of weights and threshold value
Weight w1=
1
Weight w2=
1
theta=
2
Output of net=
     1     0     0     0

McCulloh Pitts Net for ANDNOT function
Weights of neuron
     1
```

```
McCulloh Pitts Net for ANDNOT function
Weights of neuron
     1

     1

Threshold value=
     2
```

2 c) GA Application

```
clear;
stars = 30;
iterations = 500;
population = 100;
mutation = 0.7;
mutation_step = 0.8;
grid_size = 10;

for i = 1:stars
   for j = 1:2
      star(i,j) = rand(1)*grid_size;
   end
end
plot(star(:,1),star(:,2),'x')
hold on

for i = 1:population
   for j = 1:2
      cit(i,j)= rand(1)*grid_size;
   end
end

for j = 1:population
   for k = 1:stars
      d(k) = sqrt((star(k,1)-cit(j,1))^2 + (star(k,2)-cit(j,2))^2);
   end
   d(stars+1) = cit(j,1);
   d(stars+2) = grid_size - cit(j,1);
   d(stars+3) = cit(j,2);
   d(stars+4) = grid_size - cit(j,2);
   d = sort(d);
   cit(j,3) = d(1);
end
cit = sortrows(cit,-3);
cit(1,:)
cit(:,3) = cit(:,3)/cit(1,3);

for i = 1:iterations
   %---------- Mating Selection ----------
   cit(:,3) = cit(:,3)/cit(1,3);
   pool = [];
   for l = 1:population
      if rand(1)<cit(l,3),pool=[pool;cit(l,:)];,end
```

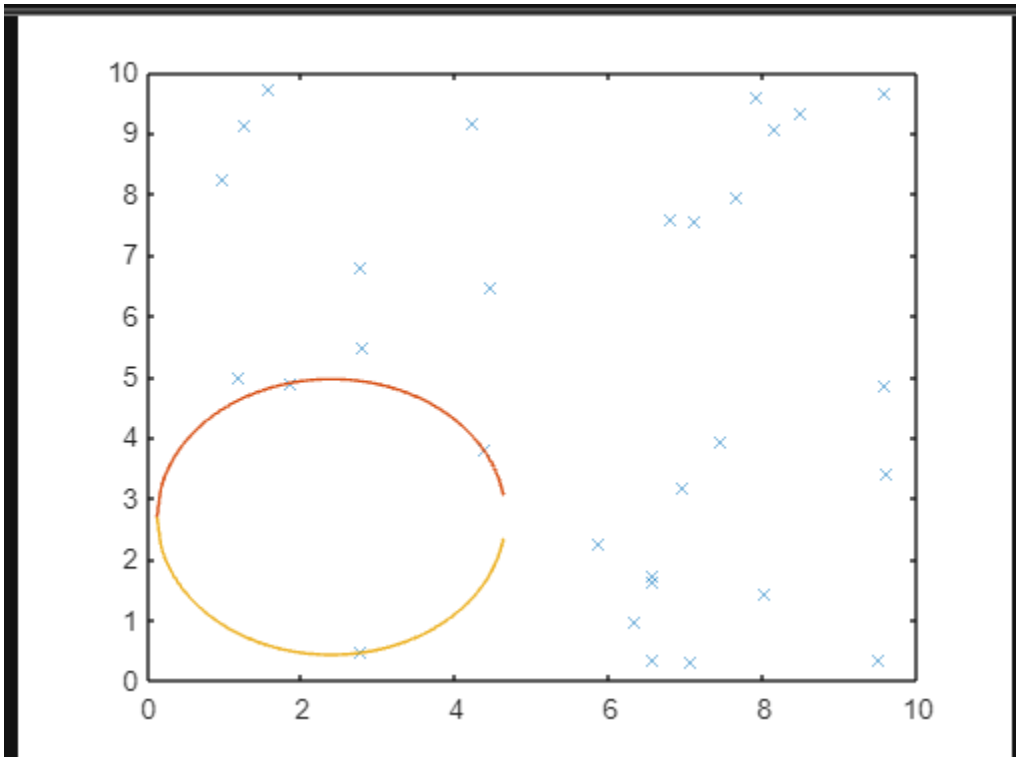
```
    end
    %--------------- Crossover ------------------
    s = size(pool,1);
    if s/2 - round(s/2) ~=0 , pool = pool(1:s-1,:); , s = size(pool,1); , end
    for m = 1:2:s
        pool = [ pool ; [pool(m,1) , pool(m+1,2) , 0]];
        pool = [ pool ; [pool(m+1,1) , pool(m,2) , 0]];
    end
    %--------------- Mutation -------------------
    s = size(pool,1);
    for m = 1:s
        if ((rand < mutation) & (pool(m,1) < grid_size - mutation_step)) ,
pool(m,1)=pool(m,1)+(2*rand - 1)*mutation_step;,end
        if ((rand < mutation) & (pool(m,2) < grid_size - mutation_step)) ,
pool(m,2)=pool(m,2)+(2*rand - 1)*mutation_step;,end
    end
    %--------------- Invironmental Selection----------
    temp = [cit ; pool];
    ts = size(temp,1);
    for j = 1:ts
        for k = 1:stars
            d(k) = sqrt((star(k,1)-temp(j,1))^2 + (star(k,2)-temp(j,2))^2);
        end
        d(stars+1) = temp(j,1);
        d(stars+2) = grid_size - temp(j,1);
        d(stars+3) = temp(j,2);
        d(stars+4) = grid_size - temp(j,2);
        d = sort(d);
        temp(j,3) = d(1);
    end
    temp = sortrows(temp,-3);
    cit = temp(1:population,:);
end
xc = cit(1,1);
yc = cit(1,2);
r  = cit(1,3);
x = (xc-r) : 0.05 : (xc+r);
for i = 1:size(x,2)
    y(i) = yc + sqrt(r^2-(x(i)-xc)^2);
end
plot (x,y)
for i = 1:size(x,2)
    y(i) = yc - sqrt(r^2-(x(i)-xc)^2);
end
```

plot (x,y)
hold off

Output:-

```
ans =

    1.9660    2.5108    1.9660
```



LAB TEST

Name: Swati Sonali

Roll: 25

SIC: 190310129

Group-B1

3) (a) Code:

```
r = [0.5 0.1; 0.2 0.9; 0.8 0.6];
s = [0.6 0.4 0.7; 0.5 0.8 0.9];
max_min = zeros(3,3);
for i=1:1:3
    for j=1:1:3
        mini = [];
        for k=1:1:2
            mini(end+1) = min(r(i, k), s(k,j));
        end
        max_min(i, j) = max(mini);
    end
end
disp(max_min);
```

Output:

```
>> composition
    0.5000    0.4000    0.5000
    0.5000    0.8000    0.9000
    0.6000    0.6000    0.7000
```

(c) Code:

```
n = 10;
xl= 0;
xu = 31;
binary_size = ceil(log2(xu-xl+1));

in_pop = round(rand(n,binary_size));
d_vals = [];
for i=1:1:n
    power = 1;
    value = 0;
    for j=binary_size:-1:1
        value = value + (in_pop(i,j)*power);
        power = power*2;
    end
    d_vals(end+1) = value;
end
```

```
legal_values = [];
for i=1:1:n
    legal_values(end+1) = legal(d_vals(i),xl,xu,binary_size);
end
fitness_values = [];
total = 0;
for i=1:1:n
    f = fitness(legal_values(i));
    fitness_values(end+1) = f;
    total = total+f;
end
prob = [];
for i=1:1:n
    prob(end+1) = fitness_values(i)/total;
end
com_prob = [];
com_prob(end+1) = prob(1);
for i=2:1:n
    com_prob(end+1) = com_prob(i-1)+prob(i);
end
sel_pop = zeros(1,n);
new_pop = zeros(n,binary_size);
for i=1:1:n
    random_value = rand;
    index = 1;
    for j=1:1:n
        if random_value <= com_prob(j)
            index = j;
            break;
        end
    end
    sel_pop(i) = d_vals(index);
    for k=1:1:binary_size
        new_pop(i,k) = in_pop(index,k);
    end
end
%disp(com_prob);
%disp(in_pop);
%disp(d_vals);
%disp(new_pop);
%disp(sel_pop);

%Crossover
cross_percent = rand*(100);
cross_freq = round((cross_percent * n)/100);
for i=1:cross_freq
    row1 = randi([1,n]);
    row2 = 0;
    while 1
        row2 = randi([1,n]);
        if row1 ~= row2
            break;
        end
    end
    [n1, n2] = crossover(new_pop(row1,:),new_pop(row2,:));
    new_pop(row1,:) = n1;
    new_pop(row2,:) = n2;
```

```
end

disp(new_pop);

%Mutation
Pm = 0.001;
new_pop = mutation(new_pop, n, binary_size, Pm);
disp(new_pop);

%%%%
fit_answer = fitness(xl);
fit = xl;
for i=1:1:n
    calculated = 0;
    mul = 1;
    for j=binary_size:-1:1
        calculated = calculated + (new_pop(i,j) * mul);
        mul = mul*2;
    end
    value = legal(calculated,xl,xu,binary_size);
    if fitness(value) > fit_answer
        fit_answer = fitness(value);
        fit = value;
    end
end
disp(fit);
```

Output:

```
>> genetic_algo
Crossover
     1     0     1     1     0
     1     1     1     1     0
     1     0     0     0     0
     1     1     1     1     0
     1     0     0     0     0
     1     1     1     0     0
     1     0     1     1     0
     1     1     1     0     0
     1     0     1     1     0
     1     0     1     1     0

Mutation
     1     0     1     1     0
     1     1     1     1     0
     1     0     0     0     0
     1     1     1     1     0
     1     0     0     0     0
     1     1     1     0     0
     1     0     1     1     0
     1     1     1     0     0
     1     0     1     1     0
     1     0     1     1     0

Fitness value:
    30
```

(b) Code:

```
disp('Enter the weights');

w1=input('Weight w1=');

w2=input('Weight w2=');

disp('Enter the threshold value');

theta=input('theta=');

y=[0 0 0 0];

x1=[1 1 0 0];

x2=[1 0 1 0];

z=[1 0 0 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('McCulloh Pitts Net for ANDNOT function');
disp('Weights of neuron');
disp(w1);
disp(w2);
disp('Threshold value=');
disp(theta);
```

Output:

```
>> NeuralNetwork
Enter the weights
Weight w1=1
Weight w2=1
Enter the threshold value
theta=2
Output of net=
     1     0     0     0

McCulloh Pitts Net for ANDNOT function
Weights of neuron
     1

     1

Threshold value=
     2
```

SC LAB TEST

Name:Sweekrit Dash

SIC:190310111

Roll :26

1) Code:

labtest1.m

```
A=[0.5 0.1;0.2 0.9;0.8 0.6]
B=[0.6 0.4 0.7;0.5 0.8 0.9]
max_prod=zeros(3,3);
for i=1:1:3
    for j=1:1:3
        prod=[];
        for k=1:1:2
            prod(end+1)=A(i,k)*B(k,j);
        end
        max_prod(i,j)=max(prod);
    end
end
disp('max product=');
disp(max_prod);
```

Output:

```
>> labtest1

A =

    0.5000    0.1000
    0.2000    0.9000
    0.8000    0.6000


B =

    0.6000    0.4000    0.7000
    0.5000    0.8000    0.9000

max product=
    0.3000    0.2000    0.3500
    0.4500    0.7200    0.8100
    0.4800    0.4800    0.5600
```

---

2) Code:

labtest1.m | LabTestGA.m | maxProduct1.m | genetic_algo.m | NN.m

```
n = 10;
xl= 0;
xu = 31;
binary_size = ceil(log2(xu-xl+1));

in_pop = round(rand(n,binary_size));
d_vals = [];
for i=1:1:n
    power = 1;
    value = 0;
    for j=binary_size:-1:1
        value = value + (in_pop(i,j)*power);
        power = power*2;
    end
    d_vals(end+1) = value;
end
legal_values = [];
for i=1:1:n
    legal_values(end+1) = legal(d_vals(i),xl,xu,binary_size);
end
fitness_values = [];
total = 0;
for i=1:1:n
    f = fitness(legal_values(i));
    fitness_values(end+1) = f;
    total = total+f;
end
prob = [];
for i=1:1:n
    prob(end+1) = fitness_values(i)/total;
end
com_prob = [];
```

labtest1.m | LabTestGA.m | maxProduct1.m | genetic_algo.m | NN.m

```
33 -    com_prob(end+1) = prob(1);
34 -    for i=2:1:n
35 -        com_prob(end+1) = com_prob(i-1)+prob(i);
36 -    end
37 -    sel_pop = zeros(1,n);
38 -    new_pop = zeros(n,binary_size);
39 -    for i=1:1:n
40 -        random_value = rand;
41 -        index = 1;
42 -        for j=1:1:n
43 -            if random_value <= com_prob(j)
44 -                index = j;
45 -                break;
46 -            end
47 -        end
48 -        sel_pop(i) = d_vals(index);
49 -        for k=1:1:binary_size
50 -            new_pop(i,k) = in_pop(index,k);
51 -        end
52 -    end
53      %disp(com_prob);
54      %disp(in_pop);
55      %disp(d_vals);
56      %disp(new_pop);
57      %disp(sel_pop);
58
59      %Crossover
60 -    disp('Crossover');
61 -    cross_percent = rand*(100);
62 -    cross_freq = round((cross_percent * n)/100);
63 -    for i=1:cross_freq
64 -        row1 = randi([1,n]);
```

labtest1.m | LabTestGA.m | maxProduct1.m | genetic_algo.m | NN.m

```
65 -        row2 = 0;
66 -        while 1
67 -            row2 = randi([1,n]);
68 -            if row1 ~= row2
69 -                break;
70 -            end
71 -        end
72 -        [n1, n2] = crossover(new_pop(row1,:),new_pop(row2,:));
73 -        new_pop(row1,:) = n1;
74 -        new_pop(row2,:) = n2;
75 -    end
76
77 -    disp(new_pop);
78
79      %Mutation
80 -    disp('Mutation')
81 -    Pm = 0.001;
82 -    new_pop = mutation(new_pop, n, binary_size, Pm);
83 -    disp(new_pop);
84
85      %%%%
86 -    fit_answer = fitness(xl);
87 -    fit = xl;
88 -    for i=1:1:n
89 -        calculated = 0;
90 -        mul = 1;
91 -        for j=binary_size:-1:1
92 -            calculated = calculated + (new_pop(i,j) * mul);
93 -            mul = mul*2;
94 -        end
95 -        value = legal(calculated,xl,xu,binary_size);
96 -        if fitness(value) > fit_answer
97 -            fit_answer = fitness(value);
98 -            fit = value;
99 -        end
100 -   end
101 -   disp('Fitness');
102 -   disp(fit);
```

Output:

```
Crossover
     1     1     1     1     1
     1     1     1     1     1
     1     1     0     1     0
     1     1     1     1     1
     1     1     0     1     0
     1     0     0     0     1
     0     1     1     1     1
     1     1     0     0     1
     1     0     0     1     0
     1     1     1     1     1

Mutation
     1     1     1     1     1
     1     1     1     1     1
     1     1     0     1     0
     1     1     1     1     1
     1     1     0     1     0
     1     0     0     0     1
     0     1     1     1     1
     1     1     0     0     1
     1     0     0     1     0
     1     1     1     1     1

Fitness
    31
```

3)code:

labtest1.m | LabTestGA.m | maxProduct1.m | genetic_algo.m | NN.m

```
disp('Enter the weights');

w1=input('Weight w1=');

w2=input('Weight w2=');

disp('Enter the threshold value');

theta=input('theta=');

y=[0 0 0 0];

x1=[1 1 0 0];

x2=[1 0 1 0];

z=[1 0 0 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('McCulloh Pitts Net for ANDNOT function');

disp('Weights of neuron');

disp(w1);
```

```

    disp(w2);

    disp('Threshold value=');

    disp(theta);
```

Output:

```
>> NN
Enter the weights
Weight w1=1
Weight w2=1
Enter the threshold value
theta=2
Output of net=
     1     0     0     0

McCulloh Pitts Net for ANDNOT function
Weights of neuron
     1

     1

Threshold value=
     2
```

Soft Computing Lab Test

Name=Sanikesh Mohanty

Sic-19030152

Roll-16

Group-B1

Answer A:

```
a=[0.4387 0.3816 0.7655 0.7952 0.1869 0.4898 0.4456 0.6463 0.7094 0.7547]
b=[0.2760 0.6797 0.6551 0.1626 0.1190 0.4984 0.9597 0.3404 0.5853 0.2238]
for i=1:10
    if a(i)<0.5
        int_a(i)=2.*(a(i).^2)
    else
        int_a(i)=1-(2.*(1-(a(i)).^2))
    end
end
for i=1:10
    if b(i)<0.
        int_b(i)=2.*(b(i).^2)
    else
        int_b(i)=1-(2.*(1-(b(i)).^2))
    end
end
int_a
int_b
```

output of A:

```
a =

  Columns 1 through 9

    0.4387    0.3816    0.7655    0.7952    0.1869    0.4898    0.4456    0.6463    0.7094

  Column 10

    0.7547


b =

  Columns 1 through 9

    0.2760    0.6797    0.6551    0.1626    0.1190    0.4984    0.9597    0.3404    0.5853

  Column 10

    0.2238
```

```
int_a =

  Columns 1 through 9

    0.3849    0.2912    0.1720    0.2647    0.0699    0.4798    0.3971   -0.1646    0.0065

  Column 10

    0.1391


int_b =

  Columns 1 through 9

   -0.8476   -0.0760   -0.1417   -0.9471   -0.9717   -0.5032    0.8420   -0.7683   -0.3148

  Column 10

   -0.8998
```

B

```
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
    end
    epoch=epoch+1;
end
disp('Perception for AND function');
disp('Final Weight Matrix');
disp(w);
disp('Final Bias');
disp(b);
```

output of B:

```
>> b_ans
Enter Learning rate=10
Enter Threshold Value=0.5
Perception for AND function
Final Weight Matrix
    10    10

Final Bias
   -10
```

Answer C:

```
a=rand(1,10);

b=rand(1,10);


union = max(a,b)

intersection=min(a,b)
```

```
complementofa = 1-a

complementofb=1-b


concentrationofa=a.^2

concentrationofb=b.^2



dilationofa=a.^0.5

dilationofb=b.^0.5


cardinalityofa=length(a)

cardinalityofb=length(b)
```

output of C:

```
union =

  Columns 1 through 9

    0.6557    0.9706    0.9572    0.9340    0.8003    0.7577    0.7431    0.9157    0.7922

  Column 10

    0.9595

intersection =

  Columns 1 through 9

    0.1576    0.0357    0.8491    0.4854    0.6787    0.1419    0.4218    0.3922    0.6555

  Column 10

    0.1712
```

```
complementofa =

  Columns 1 through 9

    0.8424    0.0294    0.0428    0.5146    0.1997    0.8581    0.5782    0.0843    0.2078

  Column 10

    0.0405


complementofb =

  Columns 1 through 9

    0.3443    0.9643    0.1509    0.0660    0.3213    0.2423    0.2569    0.6078    0.3445

  Column 10

    0.8288


concentrationofa =

  Columns 1 through 9

    0.0248    0.9421    0.9162    0.2356    0.6404    0.0201    0.1779    0.8386    0.6276


dilationofa =

  Columns 1 through 9

    0.3970    0.9852    0.9783    0.6967    0.8946    0.3767    0.6494    0.9569    0.8901

  Column 10

    0.9795


dilationofb =

  Columns 1 through 9

    0.8098    0.1890    0.9215    0.9664    0.8239    0.8705    0.8621    0.6263    0.8096

  Column 10

    0.4137
```

```
cardinalityofa =

    10


cardinalityofb =

    10

>>
```

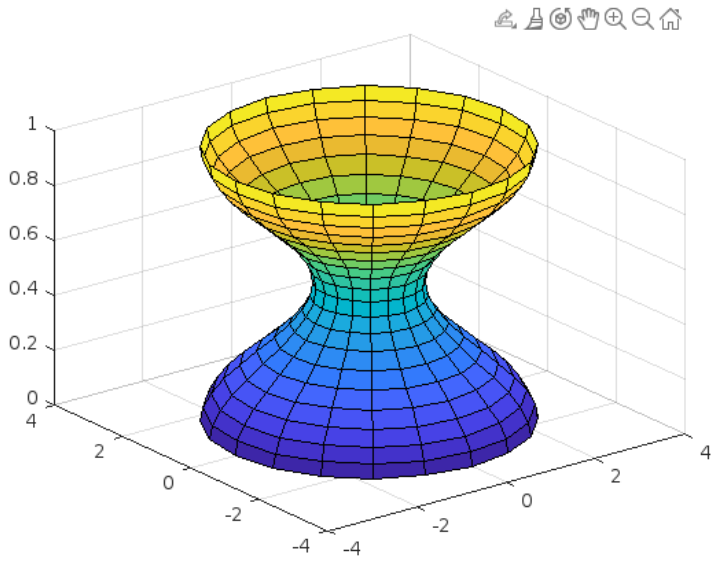
SOFT COMPUTING LAB TEST

Name: Aparna Das
Roll No: 06
SIC: 190310242
Date: 19/12/2022

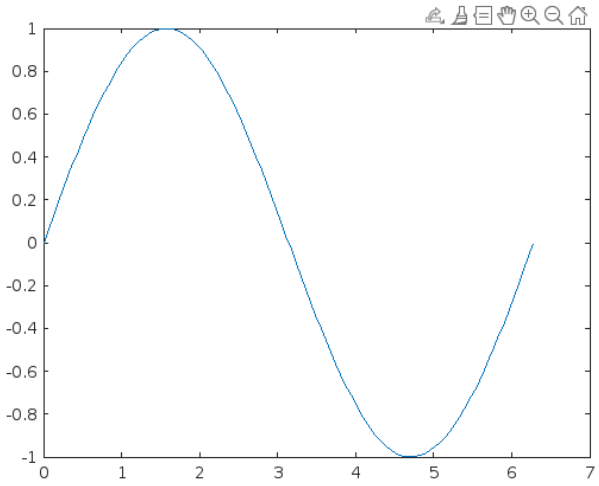
6.

a. Write functions for implementing cylindrical extension of a 1D membership function and projection of 2D membership function. Demonstrate the results visually.

```
t=0:pi/10:2*pi;
r=2+cos(t);
[X,Y,Z]=cylinder(r)
surf(X,Y,Z)
```




```
x=linspace(0,2*pi);
y=sin(x);
plot(x,y)
```



b. Any neural application of your own

```
%Neural network application
%Input 1
a=[0 1 1];
%Input 2
b=[1 0 0];
%Input 3
c=[1 0 1];
%Weights initialize
w=[1 1 1];
%input bias
bias=0.5;
y=0;
for i=1:3
   y=w(i)*a(i)+w(i)*b(i)+w(i)*c(i);
   y=y+bias;
end
ans=1/(1+exp((-1)*y));
disp('y=')
disp(y)
disp('ans=')
disp(ans)
```

```
>> labtest
y=
    2.5000

ans=
    0.9241
```

c. Fuzzy Set Operations: Union, Intersection, Complement, Concentration, Dilation, Cardinality

```
%Fuzzy operations
a=rand(1,10);
b=rand(1,10);

union = max(a,b)
intersection=min(a,b)

complementofa = 1-a
complementofb=1-b


concentrationofa=a.^2
concentrationofb=b.^2



dilationofa=a.^0.5
dilationofb=b.^0.5


cardinalityofa=length(a)
cardinalityofb=length(b)
```

```
>> labtest

union =

    0.8147    0.9706    0.9572    0.9134    0.8003    0.1419    0.4218    0.9157    0.9575    0.9649

intersection =

    0.1576    0.9058    0.1270    0.4854    0.6324    0.0975    0.2785    0.5469    0.7922    0.9595

complementofa =

    0.1853    0.0942    0.8730    0.0866    0.3676    0.9025    0.7215    0.4531    0.0425    0.0351

complementofb =

    0.8424    0.0294    0.0428    0.5146    0.1997    0.8581    0.5782    0.0843    0.2078    0.0405

concentrationofa =

    0.6638    0.8205    0.0161    0.8343    0.3999    0.0095    0.0776    0.2991    0.9168    0.9310
```

```
concentrationofb =

    0.0248    0.9421    0.9162    0.2356    0.6404    0.0201    0.1779    0.8386    0.6276    0.9206

dilationofa =

    0.9026    0.9517    0.3564    0.9557    0.7952    0.3123    0.5277    0.7395    0.9785    0.9823

dilationofb =

    0.3970    0.9852    0.9783    0.6967    0.8946    0.3767    0.6494    0.9569    0.8901    0.9795

cardinalityofa =

    10

cardinalityofb =

    10
```

LAB TEST

Name – Md Adnan Zakki

Roll No – 18

Group – B1

SIC - 190310250

7.(a) Genetic Algorithm –

```
pop_size = 5;
init_pop = zeros(1, pop_size);
lower_lim = 20;
upper_lim = 51;
binary_size = ceil(log2(upper_lim - lower_lim));
binary_range = pow2(binary_size);
init_pop_bin = round(rand(pop_size, binary_size));
dec_pop = zeros(1, pop_size);
disp(init_pop_bin);
for i = 1:pop_size
   ind = 0;
   val = 0;
   for j = binary_size:-1:1
      if(init_pop_bin(i, j) == 1)
         val = val + 2^ind;
      end
      ind = ind + 1;
   end
   disp(val);
   dec_pop(i) = decode(val, lower_lim, upper_lim, binary_range);
end
disp(dec_pop);
sel_arr = Selection(dec_pop, pop_size);
disp(sel_arr)
```

Objective function –

```
function [res] = ObjectiveFunc(x)
res = x.^2;
end
```

Fitness function –

```
function [ fitness ] =  FittnessFunc(x)
fitness = ObjectiveFunc(x);
end
```

```
GA
   1     1     0     0     0
   0     1     0     0     1
   0     0     1     1     1
   0     1     1     1     0
   0     0     0     1     0

  24

   9

   7

  14

   2

  44    29    27    34    22

  29    22    44    27    44
```

7.(c) Fuzzy application –

```
u = input('Enter the membership value of first fuzzy set');
v = input('Enter the mebership value of seconf fuxxy set');

w = max(u,v);
p = min(u,v);

q1 = 1-u;
q2 = 1-v;

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);
```

```
Enter the membership value of first fuzzy set[0 1]
Enter the mebership value of seconf fuxxy set[1 1.2]
Union of two fuzzy sets
    1.0000    1.2000

Intersection of two fuzzy sets
     0     1

Complement of first fuzzy set
     1     0

Complement of second fuzzy set
         0   -0.2000
```

Code of application-

```
function m = test2(a,b,c,x)
   if(x < a || x > c)
      m=0;
   elseif(x >= a && x <=b)
      m = (x-a)/(b-a);
   else
      m = (c-x)/(c-b);
   end
end
```

```
>> test2(4,8,12,2)

ans =

     0
```

7.(b) <u>Neural Network</u>

```
disp('Enter the weights');

w1=input('Weight w1=');

w2=input('Weight w2=');

disp('Enter the threshold value');

theta=input('theta=');

y=[0 0 0 0];

x1=[1 1 0 0];

x2=[1 0 1 0];

z=[1 0 0 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('McCulloh Pitts Net for ANDNOT function');
disp('Weights of neuron');
disp(w1);
disp(w2);
disp('Threshold value=');
disp(theta);
```

Output –

```
Enter the weights
Weight w1=1
Weight w2=1
Enter the threshold value
theta=2
Output of net=
     1     0     0     0

McCulloh Pitts Net for ANDNOT function
Weights of neuron
     1

     1

Threshold value=
     2
```

SOFT COMPUTING LAB TEST

Q9. (a) Plot the graphs of different activation functions.

Plot Bell

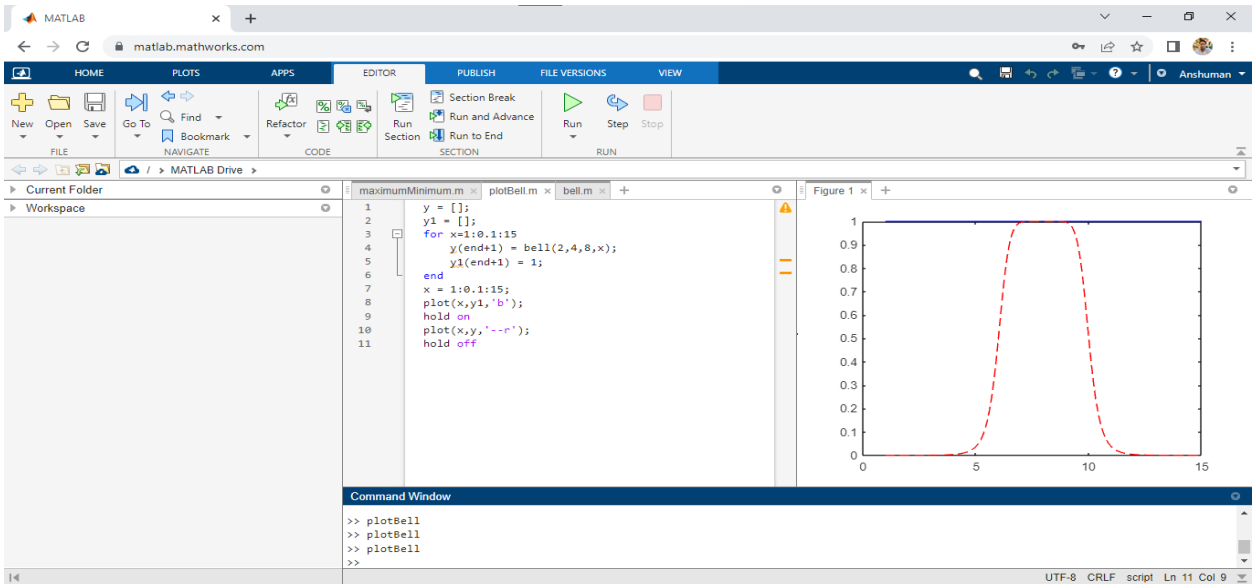
plotBell.m

```
y = [];
y1 = [];
for x=1:0.1:15
    y(end+1) = bell(2,4,8,x);
    y1(end+1) = 1;
end
x = 1:0.1:15;
plot(x,y1,'b');
hold on
plot(x,y,'--r');
hold off
```

bell.m

```
function mvalue = bell(a,b,c,x)
mvalue = 1/(1+power(abs((x-c)/a),(2*b)));
end
```

Output:



Plot triangle

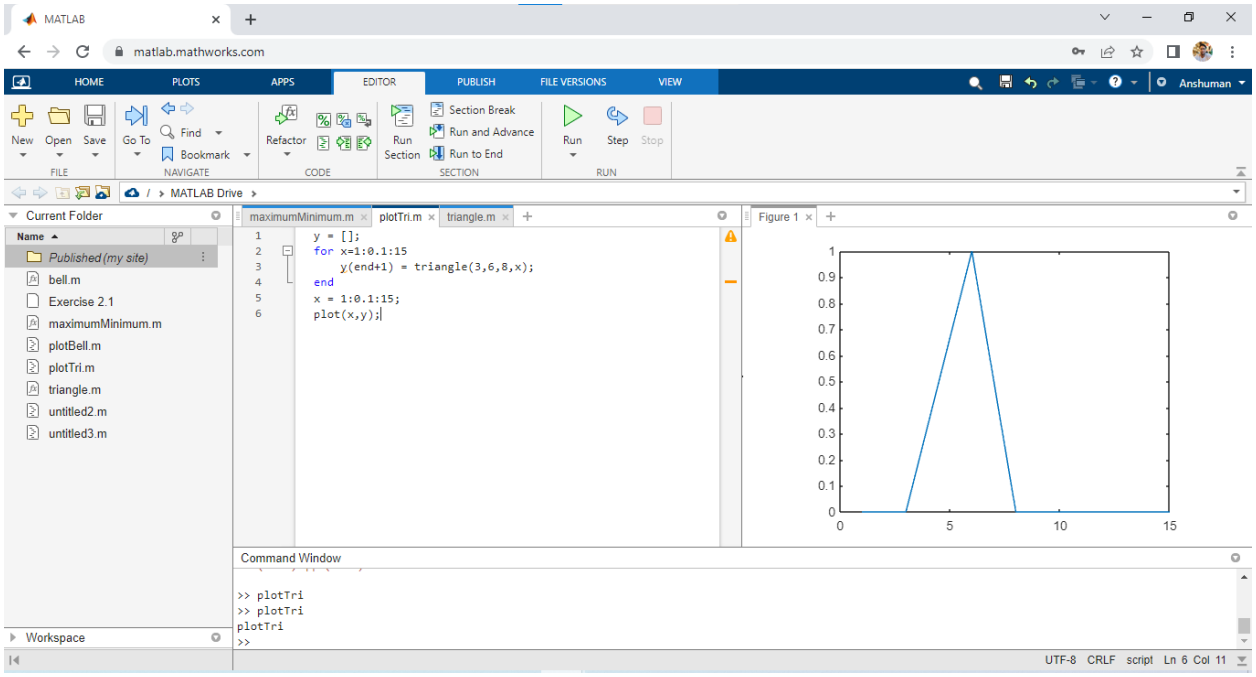
plotTri.m

```
y = [];
for x=1:0.1:15
    y(end+1) = triangle(3,6,8,x);
end
x = 1:0.1:15;
plot(x,y);
```

triangle.m

```
function mux = triangle(a,b,c,x)
if (x<=a) || (x>=c)
    mux = 0;
elseif ((x>a) && (x<=b))
    mux = (x-a)/(b-a);
elseif ((x>b) && (x<c))
    mux = (c-x)/(c-b);
end
end
```

Output:



Plot Sigmoid

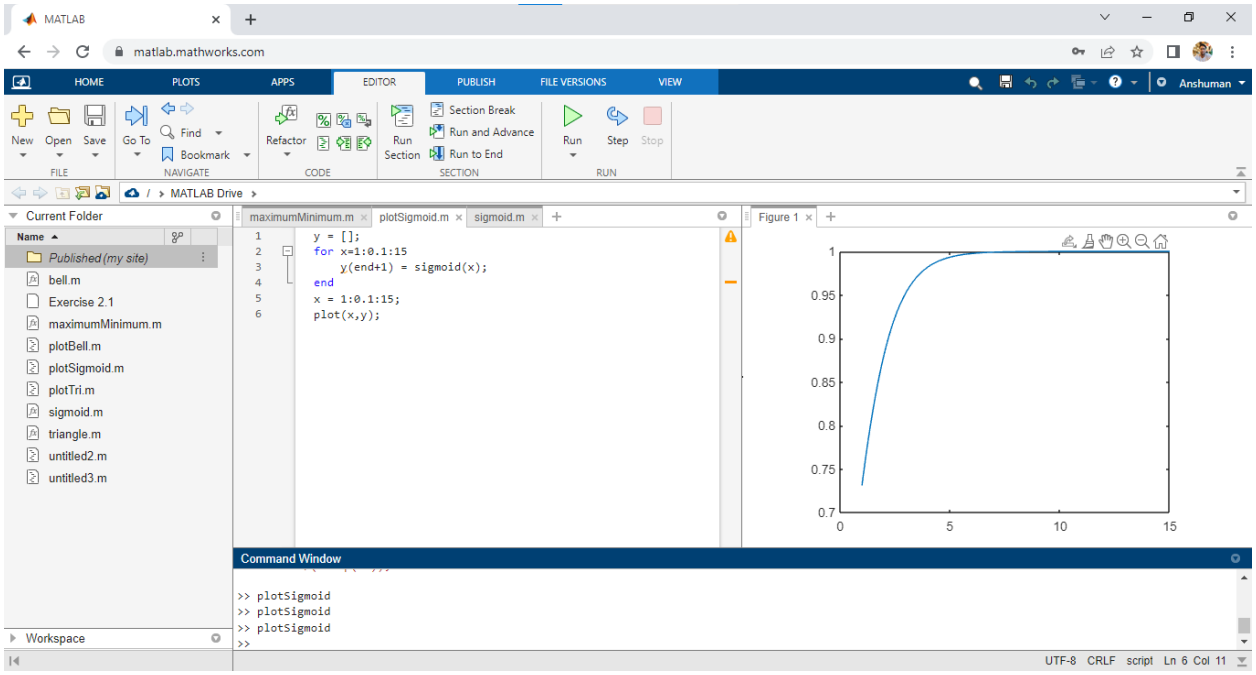
plotSigmoid.m

```
y = [];
for x=1:0.1:15
    y(end+1) = sigmoid(x);
end
x = 1:0.1:15;
plot(x,y);
```

sigmoid.m

```
function mvalue = sigmoid(x)
mvalue = 1/(1+exp(-x));
end
```

Output:



Plot Gaussian

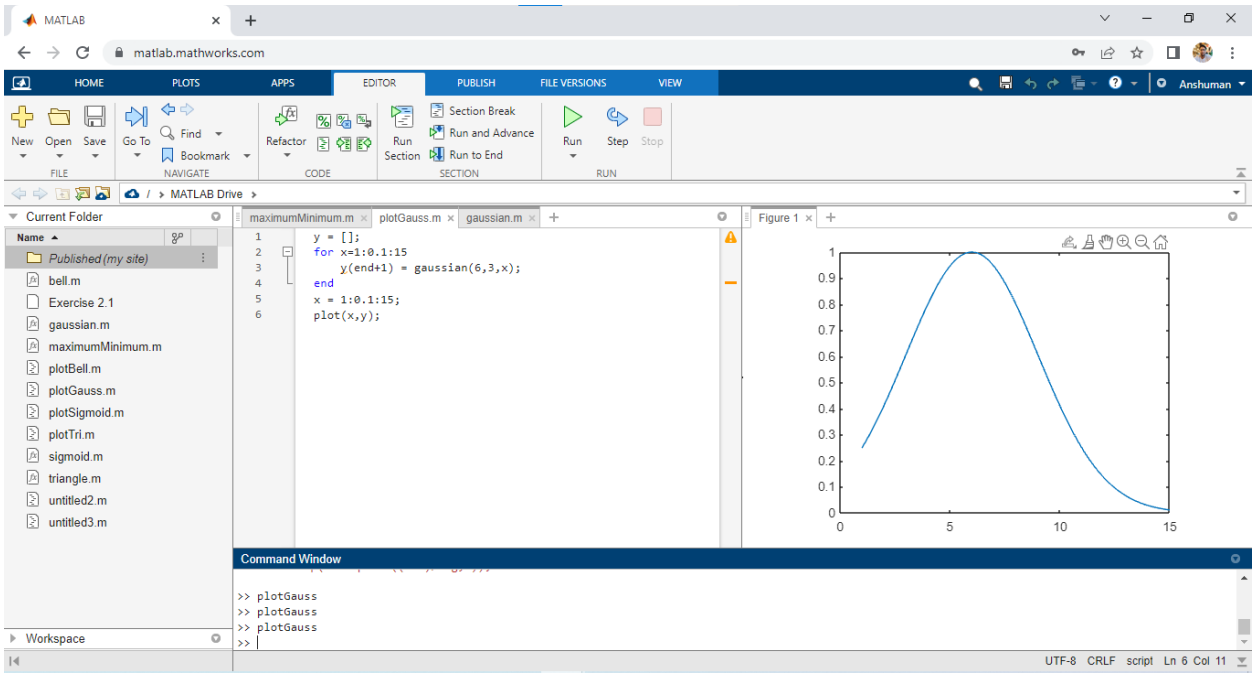
plotGauss.m

```
y = [];
for x=1:0.1:15
    y(end+1) = gaussian(6,3,x);
end
x = 1:0.1:15;
plot(x,y);
```

Gaussian.m

```
function mvalue = gaussian(a,sig,x)
mvalue = exp(-0.5*power((x-a)/sig,2));
end
```

Output:

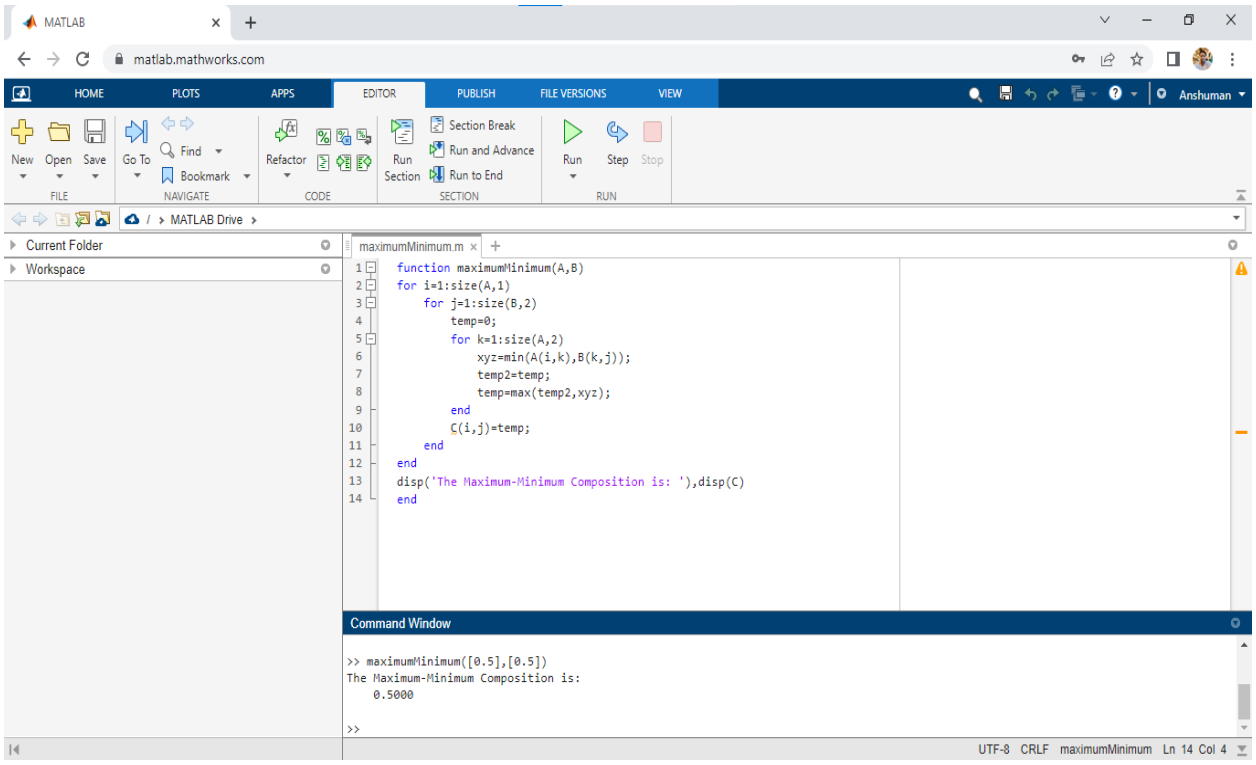


Q9. (b) Any Fuzzy application of your own.

Maximum-Minimum:

```
function maximumMinimum(A,B)
for i=1:size(A,1)
    for j=1:size(B,2)
        temp=0;
        for k=1:size(A,2)
            xyz=min(A(i,k),B(k,j));
            temp2=temp;
            temp=max(temp2,xyz);
        end
        C(i,j)=temp;
    end
end
disp('The Maximum-Minimum Composition is: '),disp(C)
end
```

Output:



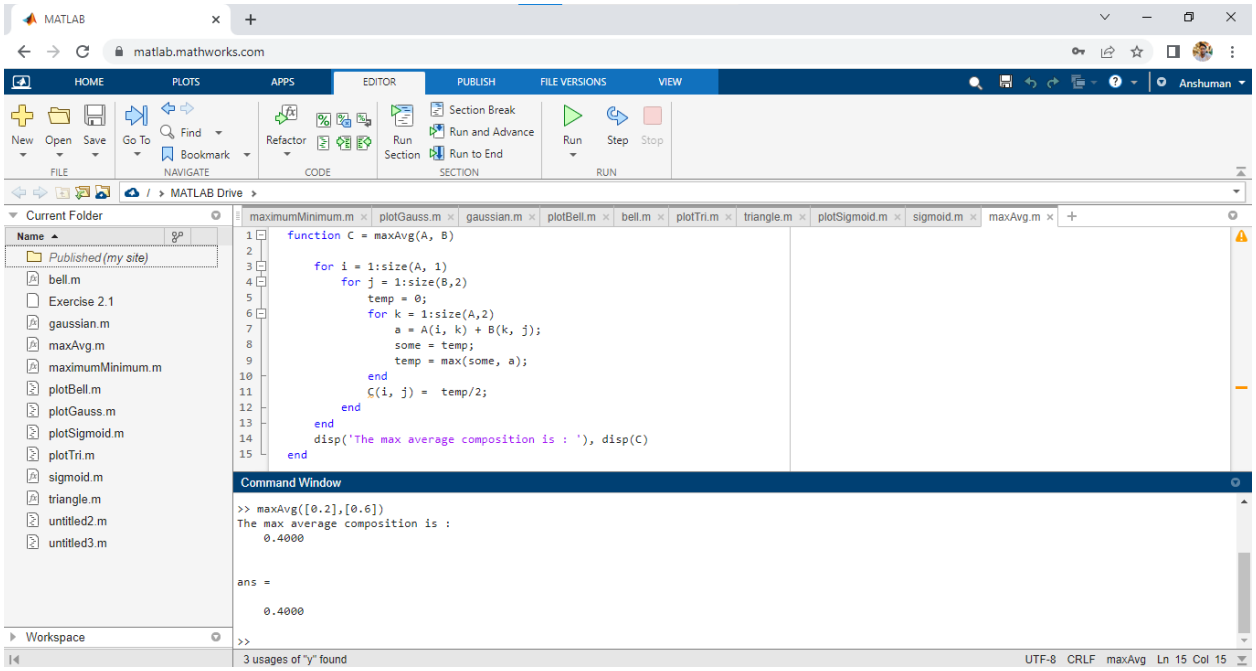
Q9. (c)

Maximum Average:

```
function C = maxAvg(A, B)

    for i = 1:size(A, 1)
        for j = 1:size(B,2)
            temp = 0;
            for k = 1:size(A,2)
                a = A(i, k) + B(k, j);
                some = temp;
                temp = max(some, a);
            end
            C(i, j) =  temp/2;
        end
    end
    disp('The max average composition is : '), disp(C)
end
```

Output:



Name – Anshuman Panda

Roll No.- 09 Section – B1 Semester – 7th SIC - 190310031

SOFT COMPUTING LAB TEST

Q1- a) Using Perceptron

AND Function

```
%? Qa %?
%? And, OR, XOR gate using perceptron %?
%? AND Gate %?
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
   end
   epoch = epoch + 1;
end
disp('Perceptron for AND Function');
disp('Final Weight Matrix : ');
disp(w);
disp('Final Bias ');
disp(b);
```

Output:

```
Enter Learning Rate :
  0.5
Enter Threshold Value :
  2
Perceptron for AND Function
Final Weight Matrix :
   2.5000    2.5000

Final Bias
  -2.5000
```

OR Function

```
%? Qa %?
%? And, OR, XOR gate using perceptron %?
%? OR Gate %?
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
   end
   epoch = epoch + 1;
```

```
end
disp('Perceptron for OR Function');
disp('Final Weight Matrix : ');
disp(w);
disp('Final Bias ');
disp(b);
```

Output:

```
Enter Learning Rate :
  0.5
Enter Threshold Value :
  2
Perceptron for OR Function
Final Weight Matrix :
   2.5000    2.5000

Final Bias
   2.5000
```

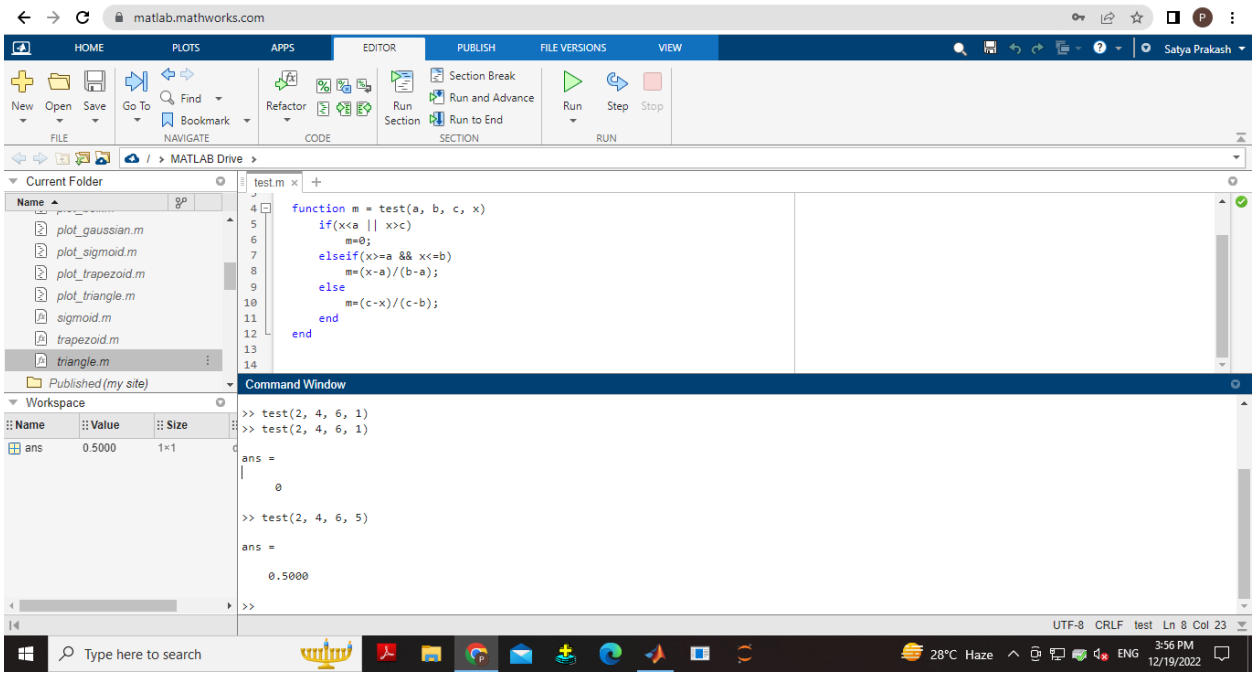
XOR Function

```
%? Qa %?
%? And, OR, XOR gate using perceptron %?
%? XOR Gate %?
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
    end
    epoch = epoch + 1;
end
disp('Perceptron for XOR Function');
disp('Final Weight Matrix : ');
disp(w);
disp('Final Bias ');
disp(b);
```

Q1- b) Any fuzzy application
Triangle Function



Q1- c)
Fuzzy Function
Trapezoid Function

```
%? Qc ?%
%? Any Fuzzy Application ?%
%? Trapezoid Function ?%
function m = test_c(a,b,c,d,x)
    if ((x<=a) || (x>=d))
        m = 0;
```

```
    elseif ((x>a) && (x<b))
        m = (x-a)/(b-a);
    elseif ((x>=b) && (x<=c))
        m = 1;
    elseif ((x>c) && (x<d))
        m = (d-x)/(d-c);
    end
end
```

Output:

```
  test_c(2, 4, 8, 10, 2)

ans =

    0

  test_c(2, 4, 8, 10, 5)

ans =

    1

  test_c(2, 4, 8, 10, 9)

ans =

   0.5000
```

Name - Satya Prakash Sahoo
Lab Roll - 10
Sic: 190310153
Group: CSE B1
Semester: 7th