Batch: SC_1 **Roll No.: 1911027**

Experiment No. 08

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Defuzzification methods.

Aim : To Write a program to implement defuzzification methods.

Expected Outcome of Experiment:

CO4 : Apply basics of Fuzzy logic and neural networks

Books/ Journals/ Websites referred:

- J.S.R.Jang, C.T.Sun and E.Mizutani, "Neuro-Fuzzy and Soft Computing", PHI, 2004, Pearson Education 2004.
- Davis E.Goldberg, "Genetic Algorithms: Search, Optimization and Machine Learning", Addison Wesley, N.Y., 1989.
- S. Rajasekaran and G.A.V.Pai, "Neural Networks, Fuzzy Logic and Genetic Algorithms", PHI, 2003.
- http://library.thinkquest.org/C007395/tqweb/history.html

Pre Lab/ Prior Concepts:

A fuzzy set is a pair (U, m) where U is a set and $m: U \to [0, 1]$.

For each $x \in U$, the value $m(x)$ is called the **grade** of membership of x in (U, m). For a finite set $U = \{x_1, \ldots, x_n\}$, the fuzzy set (U, m) is often denoted by $\{m(x_1)/x_1, \ldots, m(x_n)/x_n\}$.

Let $x \in U$. Then x is called **not included** in the fuzzy set (U, m) if $m(x) = 0$, x is called **fully included** if $m(x) = 1$, and x is called a **fuzzy member** if $0 < m(x) < 1$. The set

$\{x \in U \mid m(x) > 0\}$ is called the **support** of (U, m) and the set $\{x \in U \mid m(x) = 1\}$ is called its **kernel**. The function m is called the **membership function** of the fuzzy set (U, m).

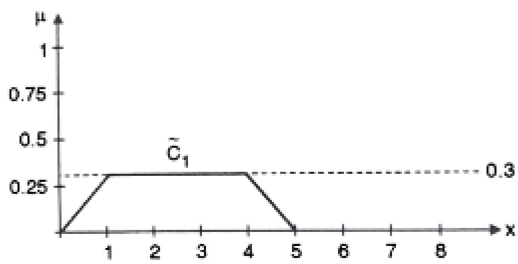
Implementation Details:

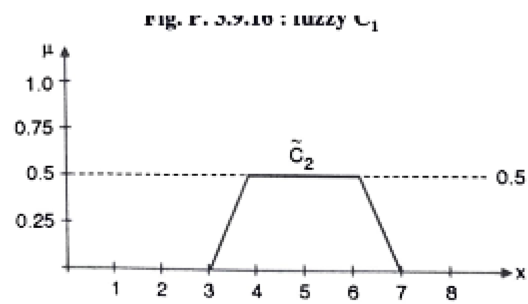
Defuzzification :

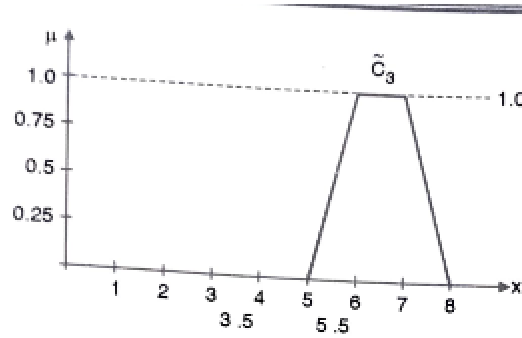
Defuzzification is the process of converting fuzzy values into crisp output.
There are five methods for defuzzification:

1. centroid of area
2. bisector of area
3. middle of maximum
4. smallest of maximum
5. largest of maximum

Suppose you have the following regions to be defuzzified.







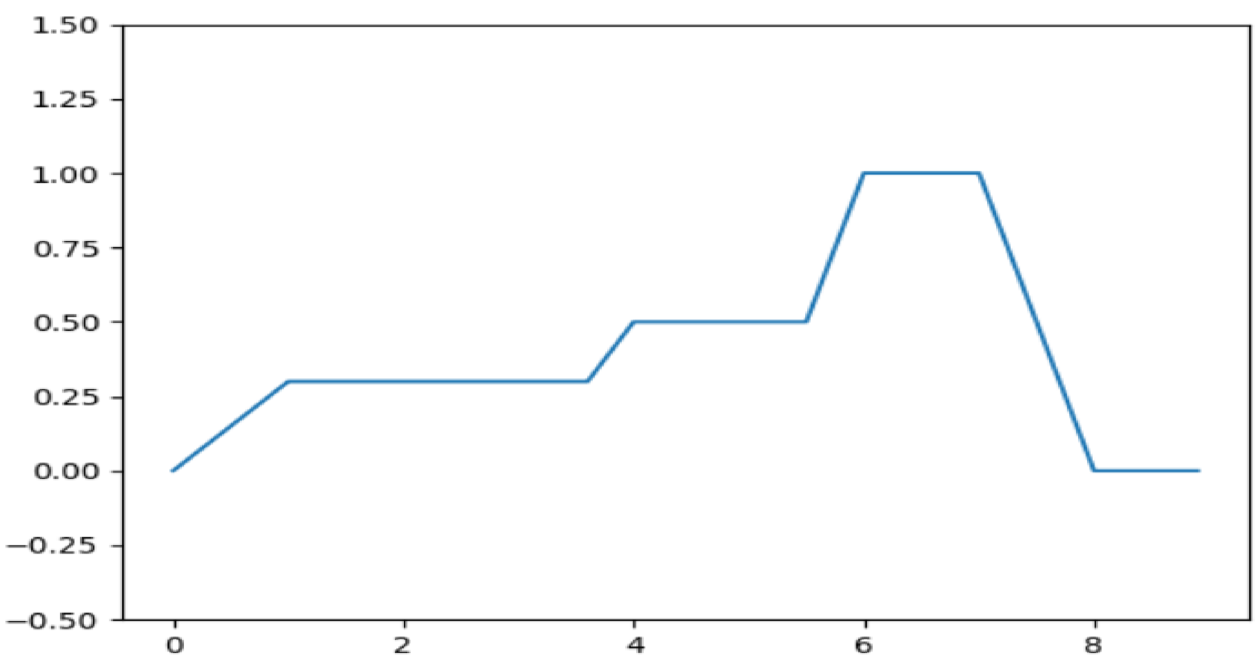
Union of area:

Implementation details:

Code:

```
from matplotlib import pyplot as plt
import numpy as np
import skfuzzy as fuzz_defuzz
inputs_x=np.arange(0,9,0.1)
coord_trap_1=[0,1,4,5]
coord_trap_2=[3,4,6,7]
coord_trap_3=[5,6,7,8]
trap_mem_1=fuzz_defuzz.trapmf(inputs_x,coord_trap_1)
trap_mem_2=fuzz_defuzz.trapmf(inputs_x,coord_trap_2)
trap_mem_3=fuzz_defuzz.trapmf(inputs_x,coord_trap_3)
final_mf=np.maximum(0.5*trap_mem_2,np.maximum(0.3*trap_mem_1,1*trap_mem_3))
plt.yticks=[0,0.5,1]
plt.ylim([-0.5,1.5])
plt.plot(inputs_x,final_mf)
plt.show()
```

Output:



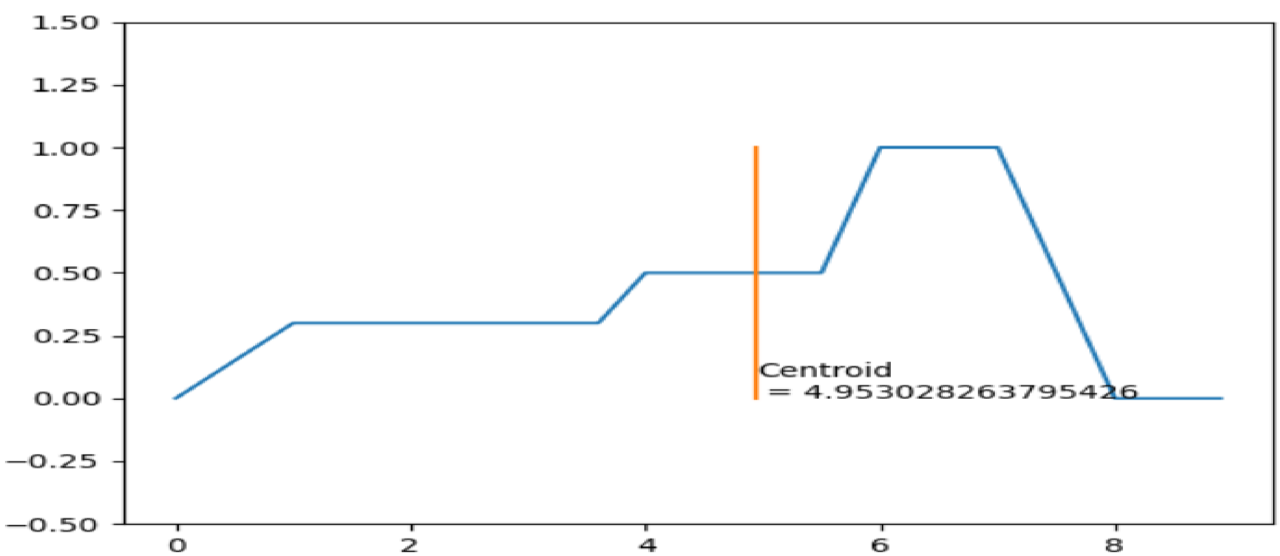
Centroid of Area:

Implementation details:

Code:

```
from matplotlib import pyplot as plt
import numpy as np
import skfuzzy as fuzz_defuzz
inputs_x=np.arange(0,9,0.1)
coord_trap_1=[0,1,4,5]
coord_trap_2=[3,4,6,7]
coord_trap_3=[5,6,7,8]
trap_mem_1=fuzz_defuzz.trapmf(inputs_x,coord_trap_1)
trap_mem_2=fuzz_defuzz.trapmf(inputs_x,coord_trap_2)
trap_mem_3=fuzz_defuzz.trapmf(inputs_x,coord_trap_3)
final_mf=np.maximum(0.5*trap_mem_2,np.maximum(0.3*trap_mem_1,1*trap_mem_3))
plt.yticks=[0,0.5,1]
plt.ylim([-0.5,1.5])
plt.plot(inputs_x,final_mf)
centroid=fuzz_defuzz.defuzz(inputs_x,final_mf,'centroid')
line_centroid=plt.plot([centroid,centroid],[0,1])
text_centroid=plt.text(centroid,0,f'Centroid\n = {centroid}')
plt.show()
```

Output:



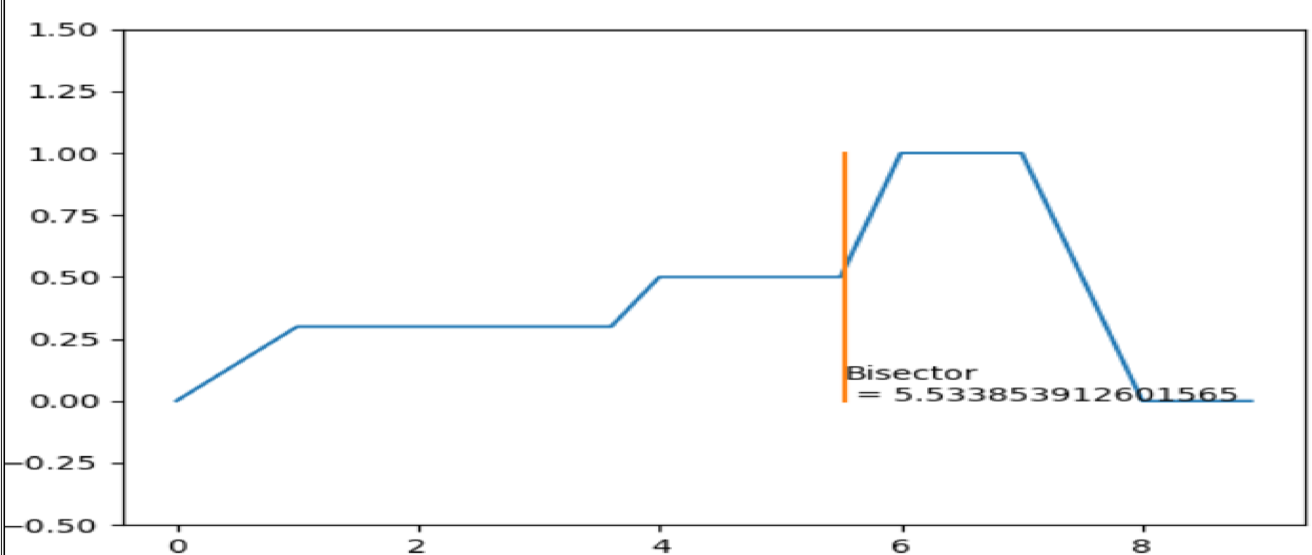
Bisector of Area :

Implementation details:

Code:

```
from matplotlib import pyplot as plt
import numpy as np
import skfuzzy as fuzz_defuzz
from skfuzzy.defuzzify.defuzz import bisector
inputs_x=np.arange(0,9,0.1)
coord_trap_1=[0,1,4,5]
coord_trap_2=[3,4,6,7]
coord_trap_3=[5,6,7,8]
trap_mem_1=fuzz_defuzz.trapmf(inputs_x,coord_trap_1)
trap_mem_2=fuzz_defuzz.trapmf(inputs_x,coord_trap_2)
trap_mem_3=fuzz_defuzz.trapmf(inputs_x,coord_trap_3)
final_mf=np.maximum(0.5*trap_mem_2,np.maximum(0.3*trap_mem_1,1*trap_mem_3))
plt.yticks=[0,0.5,1]
plt.ylim([-0.5,1.5])
plt.plot(inputs_x,final_mf)
bisector=fuzz_defuzz.defuzz(inputs_x,final_mf,'bisector')
line_bisector=plt.plot([bisector,bisector],[0,1])
text_bisector=plt.text(bisector,0,f'Bisector\n = {bisector}')
plt.show()
```

Output:



Middle, Smallest, and Largest of Maximum:

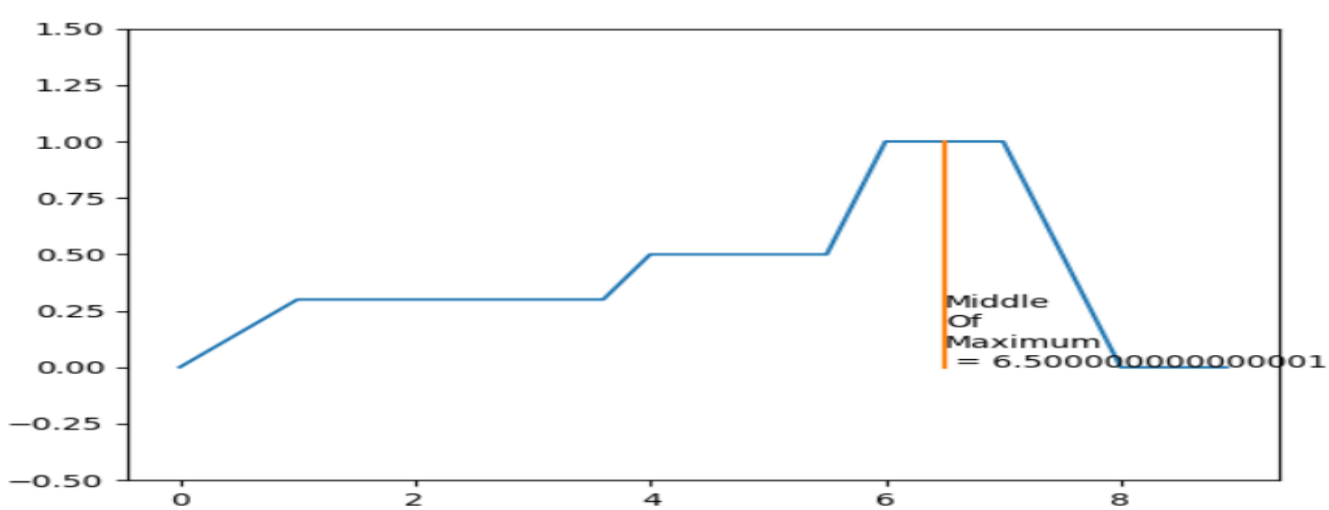
Implementation details:

Middle:

Code:

```
from matplotlib import pyplot as plt
import numpy as np
import skfuzzy as fuzz_defuzz
from skfuzzy.defuzzify.defuzz import bisector
inputs_x=np.arange(0,9,0.1)
coord_trap_1=[0,1,4,5]
coord_trap_2=[3,4,6,7]
coord_trap_3=[5,6,7,8]
trap_mem_1=fuzz_defuzz.trapmf(inputs_x,coord_trap_1)
trap_mem_2=fuzz_defuzz.trapmf(inputs_x,coord_trap_2)
trap_mem_3=fuzz_defuzz.trapmf(inputs_x,coord_trap_3)
final_mf=np.maximum(0.5*trap_mem_2,np.maximum(0.3*trap_mem_1,1*trap_mem_3))
plt.yticks=[0,0.5,1]
plt.ylim([-0.5,1.5])
plt.plot(inputs_x,final_mf)
middle_of_maximum=fuzz_defuzz.defuzz(inputs_x,final_mf,'mom')
line_middle_of_maximum=plt.plot([middle_of_maximum,middle_of_maximum],[0,1])
text_middle_of_maximum=plt.text(middle_of_maximum,0,f'Middle\nOf\nMaximum\n =
{middle_of_maximum}')
plt.show()
```

Output:

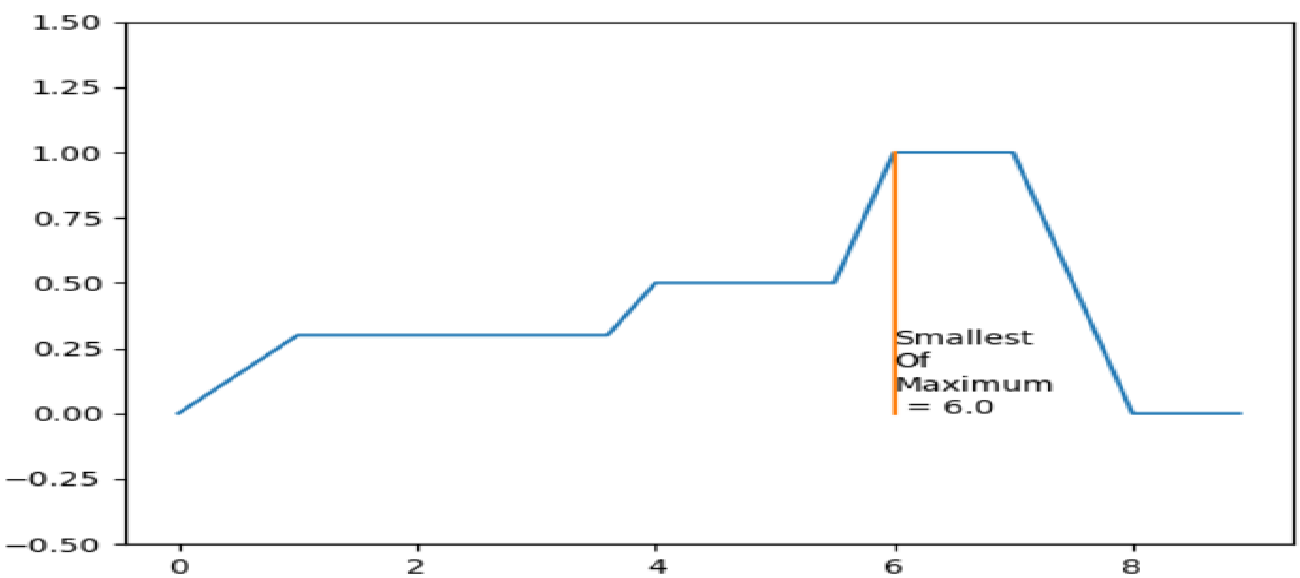


Smallest:

Code:

```
from matplotlib import pyplot as plt
import numpy as np
import skfuzzy as fuzz_defuzz
from skfuzzy.defuzzify.defuzz import bisector
inputs_x=np.arange(0,9,0.1)
coord_trap_1=[0,1,4,5]
coord_trap_2=[3,4,6,7]
coord_trap_3=[5,6,7,8]
trap_mem_1=fuzz_defuzz.trapmf(inputs_x,coord_trap_1)
trap_mem_2=fuzz_defuzz.trapmf(inputs_x,coord_trap_2)
trap_mem_3=fuzz_defuzz.trapmf(inputs_x,coord_trap_3)
final_mf=np.maximum(0.5*trap_mem_2,np.maximum(0.3*trap_mem_1,1*trap_mem_3))
plt.yticks=[0,0.5,1]
plt.ylim([-0.5,1.5])
plt.plot(inputs_x,final_mf)
smallest_of_maximum=fuzz_defuzz.defuzz(inputs_x,final_mf,'som')
line_smallest_of_maximum=plt.plot([smallest_of_maximum,smallest_of_maximum],[0,1])
text_smallest_of_maximum=plt.text(smallest_of_maximum,0,f'Smallest\nOf\nMaximum\n = 
{smallest_of_maximum}')
plt.show()
```

Output:

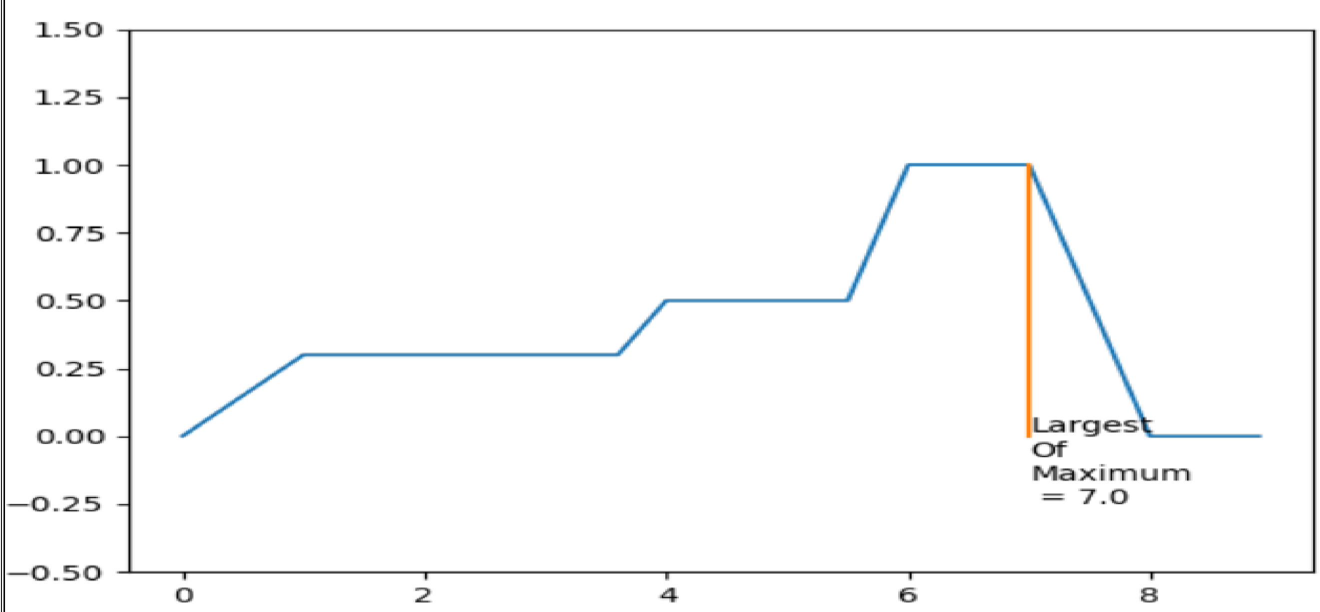


Largest:

Code:

```
from matplotlib import pyplot as plt
import numpy as np
import skfuzzy as fuzz_defuzz
from skfuzzy.defuzzify.defuzz import bisector
inputs_x=np.arange(0,9,0.1)
coord_trap_1=[0,1,4,5]
coord_trap_2=[3,4,6,7]
coord_trap_3=[5,6,7,8]
trap_mem_1=fuzz_defuzz.trapmf(inputs_x,coord_trap_1)
trap_mem_2=fuzz_defuzz.trapmf(inputs_x,coord_trap_2)
trap_mem_3=fuzz_defuzz.trapmf(inputs_x,coord_trap_3)
final_mf=np.maximum(0.5*trap_mem_2,np.maximum(0.3*trap_mem_1,1*trap_mem_3))
plt.yticks=[0,0.5,1]
plt.ylim([-0.5,1.5])
plt.plot(inputs_x,final_mf)
largest_of_maximum=fuzz_defuzz.defuzz(inputs_x,final_mf,'lom')
line_largest_of_maximum=plt.plot([largest_of_maximum,largest_of_maximum],[0,1])
text_largest_of_maximum=plt.text(largest_of_maximum,-
0.25,f'\n\n\nLargest\nOf\nMaximum\n = {largest_of_maximum}')
plt.show()
```

Output:



Conclusion: Thus, we have successfully implemented defuzzification methods. Also implemented centroid of area, bisector of area, smallest, largest and middle of maximum defuzzification methods using python programming language and also plotted each of the mentioned in graph.

Post Lab Descriptive Questions:

1. Explain a fuzzy system for a fully automatic washing machine.

ANS) Design a controller to determine the wash time of a domestic washing machine. Assume that input is dirt and grease on cloths. Use three descriptors for input variables and five descriptors for output variables. Derive set of rules for controller action and defuzzification. The design should be supported by figures wherever possible. Show that if the cloths are soiled to a larger degree the wash time will be more and vice-versa.

Step 1: Identify input and output variables and decide descriptors for the same.

- Here inputs are 'dirt' and 'grease'. Assume that they are measured in percentage (%). That is amount of dirt and grease is measured in percentage.
- Output is 'wash time' measured in minutes.
- We use three descriptors for each of the input variables.

Descriptors for dirt are as follows: SD = Small Dirt, MD = Medium Dirt, LD = Large Dirt

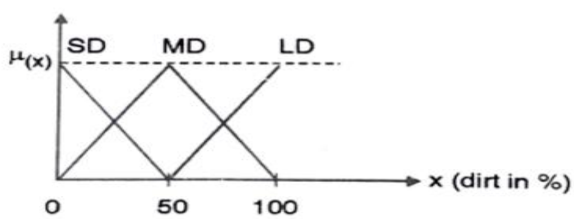
Descriptors for grease are: NG = No grease, MG = Medium grease, LG = Large grease

We use five descriptors for output variable. So, descriptors for wash time are

VS = Very Short, S = Short, M = Medium, L = Large, VL = Very Large

Step 2: Define membership functions for each of the input and output variables. We use triangular MFs because of their simplicity.

(1) Membership functions for dirt:

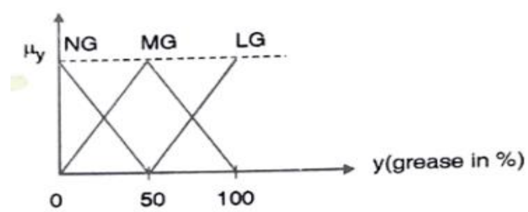


$$\mu_{SD}(x) = \frac{50-x}{50}, \quad 0 \le x \le 50$$

$$\mu_{MD}(x) = \begin{cases} \frac{x}{50}, & 0 \le x \le 50 \\ \frac{100-x}{50}, & 50 < x \le 100 \end{cases}$$

$$\mu_{LD}(x) = \frac{x-50}{50}, \quad 50 \le x \le 100$$

(2) Membership functions for grease:

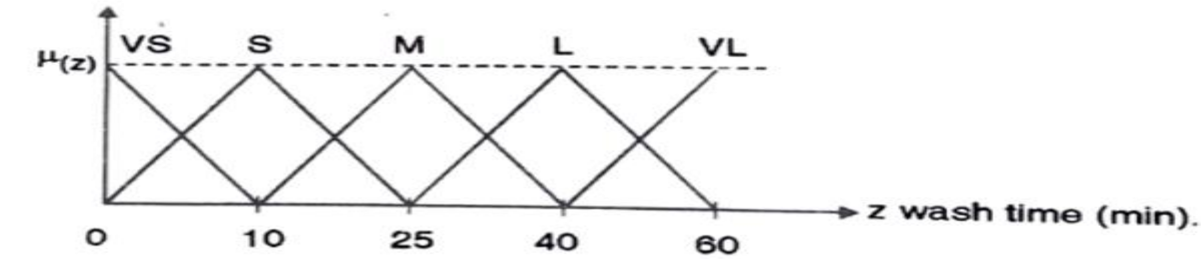


$$\mu_{NG}(y) = \frac{50-y}{50}, \quad 0 \le y \le 50$$

$$\mu_{MG}(y) = \begin{cases} \frac{y}{50}, & 0 \le y \le 50 \\ \frac{100-y}{50}, & 50 < y \le 100 \end{cases}$$

$$\mu_{LG}(y) = \frac{y-50}{50}, \quad 50 \le y \le 100$$

(3) Membership functions for Wash time:



$$\mu_{vs}(z) = \frac{10-z}{10}, 0 \leq z \leq 10$$

$$\mu_{s}(z) = \begin{cases} \frac{z}{10}, & 0 \leq z \leq 10 \\ \frac{25-z}{15}, & 10 \leq z \leq 25 \end{cases}$$

$$\mu_{M}(z) = \begin{cases} \frac{z-10}{15}, & 10 \leq z \leq 25 \\ \frac{40-z}{15}, & 25 < z \leq 40 \end{cases}$$

$$\mu_{L}(z) = \begin{cases} \frac{z-25}{15}, & 25 \leq z \leq 40 \\ \frac{60-z}{20}, & 40 < z \leq 60 \end{cases}$$

$$\mu_{VL}(z) = \frac{z-40}{20}, \quad 40 \leq z \leq 60$$

Step 3: Form a Rule base:

x \ y	NG	MG	LG
SD	VS	M	L
MD	S	M	L
LD	M	L	VL

The above matrix represents all nine rules. For example, first rule can be "If dirt is small and no grease then wash time is very short" similarly all nine rules can be defined using if --- then. E.g. "If dirt is large and grease is also large than wash time will be very large".

Step 4: Rule Evaluation:

Assume that dirt = 60% and grease = 70%. Dirt = 60 % maps to the following two MFs of "dirt" variable.

$\mu_{MD}(x) = \dfrac{100 - x}{50}$ and $\mu_{LD}(x) = \dfrac{x - 50}{50}$

Similarly grease = 70 % maps to the following two MFs of "grease" variable.

$\mu_{MG}(y) = \dfrac{100 - y}{50}$ and $\mu_{LG}(y) = \dfrac{y - 50}{50}$

Evaluate: µMD (x) and µLD(x) for x = 60, we will get,

$$\mu_{MD}(60) = \frac{100 - 60}{50} = \frac{4}{5}$$

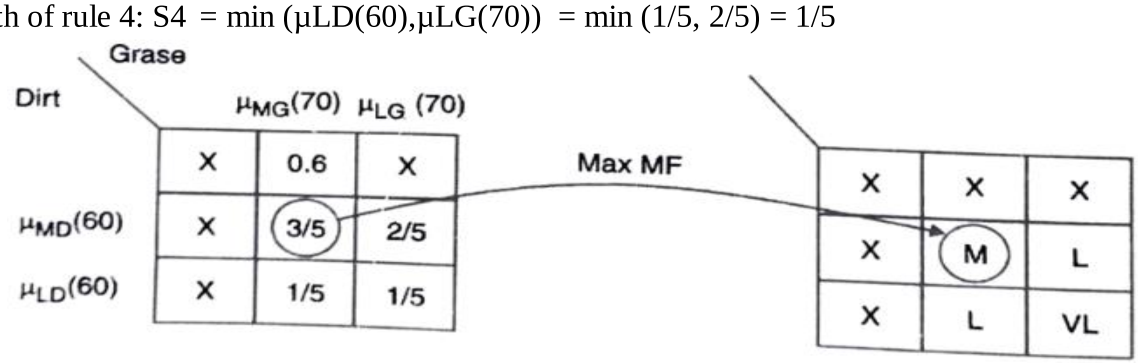
$$\mu_{LD}(60) = \frac{60 - 50}{50} = \frac{1}{5}$$

Similarly evaluate µMG(y) and µLG(y) for y = 70, we will get,

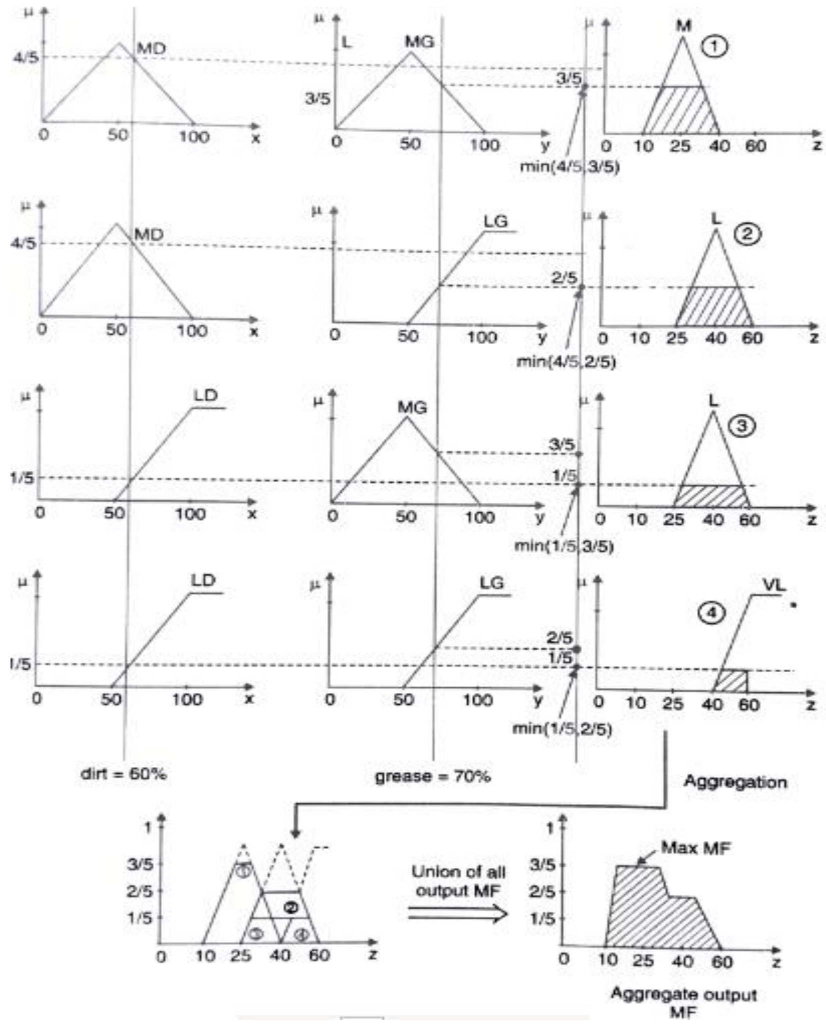
$$\mu_{MG}(70) = \frac{100 - 70}{50} = \frac{3}{5}$$

$$\mu_{LG}(70) = \frac{70 - 50}{50} = \frac{2}{5}$$

The above four equation leads to the following four rules that we are suppose to evaluate.
(1) dirt is medium and grease is medium.
(2) dirt is medium and grease is large.
(3) dirt is large and grease is medium.
(4) dirt is large and grease is large.
Since the antecedent part of each of the above rule is connected by and operator we use min operator to evaluate strength of each rule.
Strength of rule 1: S1 = min (µMD(60),µMG(70)) = min (4/5, 3/5) = 3/5
Strength of rule 2: S2 = min (µMD(60),µLG(70)) = min (4/5, 2/5) = 2/5
Strength of rule 3: S3 = min (µLD(60),µMG(70)) = min (1/5, 3/5) = 1/5
Strength of rule 4: S4 = min (µLD(60),µLG(70)) = min (1/5, 2/5) = 1/5



Step 5: Defuzzification:



Since, we use "Mean of max" defuzzification technique, we first find the rule with the maximum strength.

= Max (S1, S2, S3, S4) = Max (3/5, 2/5, 1/5, 1/5) = 3/5

This corresponds to rule 1. The rule 1: "dirt is medium and grease is medium" has maximum strength 3/5. The above rule corresponds to the output MF µM (z). To find out the final defuzzified value, we now take average (mean) of µM (z).

$$\mu_M(z) = \frac{z-10}{15} \quad \text{and} \qquad \mu_M(z) = \frac{40-2}{15}$$

$$\therefore \ 3/5 = \frac{z-10}{15} \qquad 3/5 = \frac{40-2}{15}$$

$$\therefore \ Z = 19 \qquad z = 31$$

$$\therefore z^* = \frac{19+31}{2} = \mathbf{25\ min}$$

Date: 24 / 11 / 2021 **Signature of faculty in-charge**