**Batch: A2 Roll No.: 16010121045**

**Experiment No. 08**

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

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title:** Defuzzification methods. |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Aim :** To understand the concept of Defuzzification. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO4 :** Apply basics of Fuzzy logic and neural networks **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

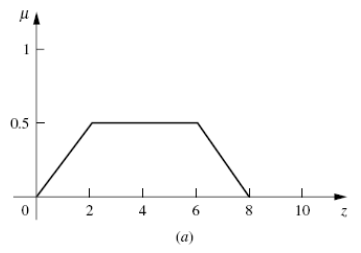
**Books/ Journals/ Websites referred:**

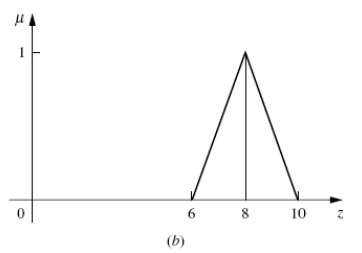
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Pre Lab/ Prior Concepts:**

Defuzzification :

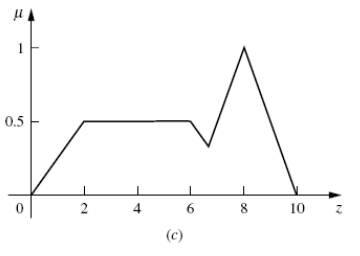
Defuzzification is the process of producing a quantifiable result in Crisp logic, given fuzzy sets and corresponding membership degrees. It is the process that maps a fuzzy set to a crisp set. It is typically needed in fuzzy control systems. These will have a number of rules that transform a number of variables into a fuzzy result, that is, the result is described in terms of membership in fuzzy sets. Defuzzification is the conversion of a fuzzy quantity to a precise quantity, just as fuzzification is the conversion of a precise quantity to a fuzzy quantity. µ

For example, **Fig (a)** shows the first part of the Fuzzy output and **Fig (b)** shows the second part of the Fuzzy output.





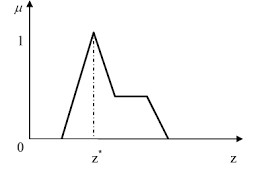
Then **Fig (c)** shows the union of the two parts (a) and (b).



Different Defuzzification methods

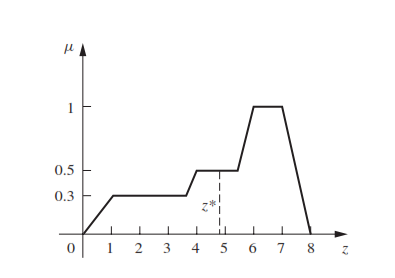
1. **Max membership method**

This method is also known as height method and is limited to peak output functions. This method is given by the algebraic expression:  
**µ**(z\*) >= **µ**(z) for all z ∊ Z.



1. **Center of gravity or centroid**

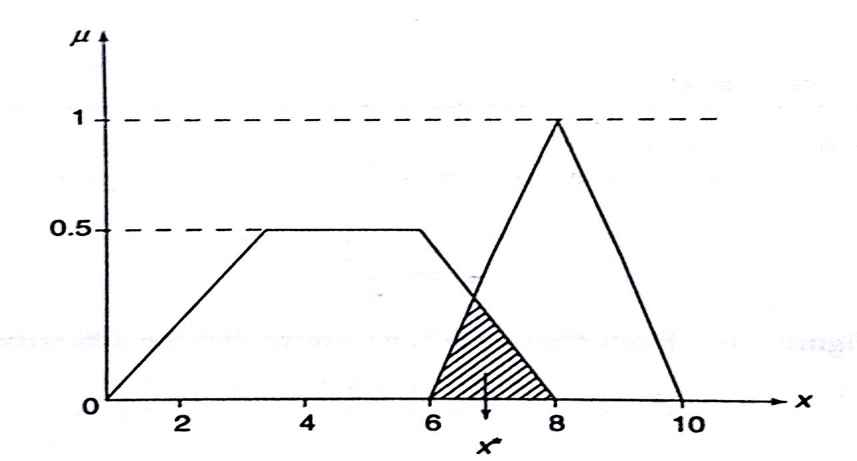
This method is also known as the centre of mass, centre of area or centre of gravity. It is the most commonly used defuzzification method. The defuzzified output z\* is given by:  
**z\* = ∫µ(z).zdz / ∫µ(z)dz**



1. **Centre of sums**

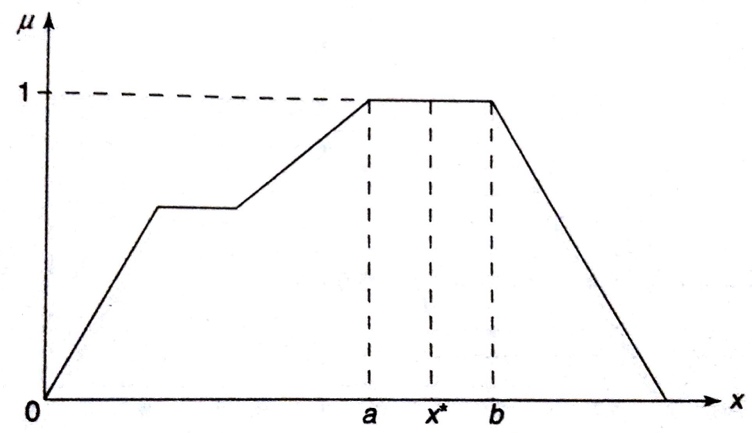
This method employs the algebraic sum of the individual fuzzy subsets instead of their union. The calculations here are very fast, but the main drawback is that the intersecting areas are added twice. The defuzzified value z\* is given by

**z\* = ∫ z\*∑µ(z).zdz / ∫ ∑µ(z)dz**



1. **Mean of maximum method**

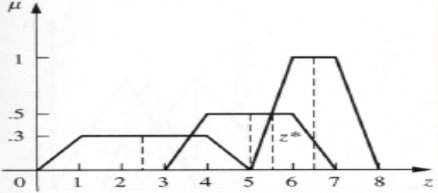
This method is also known as the middle of the maxima. This is closely related to the max-membership method, except that the locations of the maximum membership can be nonunique. The output here is given by:  
**z\* = ∑z' / n**; where z' is the maximum value of the membership function.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Weighted average method**

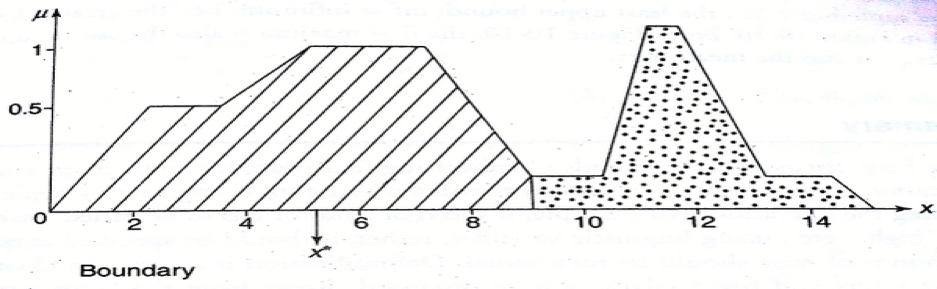
This method is valid for symmetrical output membership functions only. Each membership function is weighted by its maximum membership value. The output in the case is given by  
z\* = ∑µ(z').z' / ∑µ(z') ; where z' is the maximum value of the membership function.



1. **Centre of Largest Area**

This method can be adopted when the output of at least two convex fuzzy subsets which are not overlapping. The output, in this case, is biased towards a side of one membership function. When output fuzzy st has at least two convex regions, then the centre of gravity of the convex fuzzy subregion having the largest are is used to obtain the defuzzified value z\*. The value is given by

z\* = ∫ µc(z).zdz / ∫ ∑µc(z)dz



**Implementation Details:**

*import* matplotlib.pyplot *as* plt

def plot\_membership\_function(data):

plt.plot(data, label='Membership Function')

plt.xlabel('Index')

plt.ylabel('Membership Value')

plt.legend()

plt.show()

def lambda\_cut\_method(data):

lambda\_value = float(input("Enter lambda value: "))

cut\_data = [x *for* x *in* data *if* x >= lambda\_value]

print("Lambda Cut Method Result:", cut\_data)

plot\_membership\_function(data)

def weighted\_average(data):

weights = [float(w) *for* w *in* input("Enter weights corresponding to each defuzzy values: ").split()]

*if* len(weights) != len(data):

print("Error: Number of weights should match the number of data points.")

*return*

weighted\_avg = sum([data[i] \* weights[i] *for* i *in* range(len(data))]) / sum(weights)

print("Weighted Average:", weighted\_avg)

*# Plot weighted average*

plt.axhline(y=weighted\_avg, color='r', linestyle='--', label='Weighted Average')

plot\_membership\_function(data)

def height\_of\_maxima(data):

max\_value = max(data)

print("Height of Maxima:", max\_value)

*# Plot maxima*

max\_indexes = [i *for* i, value *in* enumerate(data) *if* value == max\_value]

plt.scatter(max\_indexes, [max\_value] \* len(max\_indexes), color='r', label='Maxima')

plot\_membership\_function(data)

def first\_of\_maxima(data):

max\_value = max(data)

first\_index = data.index(max\_value)

print("First of Maxima:", data[first\_index])

*# Plot first maxima*

plt.scatter(first\_index, max\_value, color='r', label='First of Maxima')

plot\_membership\_function(data)

def last\_of\_maxima(data):

max\_value = max(data)

max\_indexes = [i *for* i, value *in* enumerate(data) *if* value == max\_value]

last\_index = max\_indexes[-1]

print("Last of Maxima:", data[last\_index])

*# Plot last maxima*

plt.scatter(last\_index, max\_value, color='r', label='Last of Maxima')

plot\_membership\_function(data)

def mean\_of\_maxima(data):

max\_value = max(data)

max\_indexes = [i *for* i, value *in* enumerate(data) *if* value == max\_value]

mean\_maxima = sum([data[i] *for* i *in* max\_indexes]) / len(max\_indexes)

print("Mean of Maxima:", mean\_maxima)

*# Plot mean maxima*

plt.axhline(y=mean\_maxima, color='r', linestyle='--', label='Mean of Maxima')

plot\_membership\_function(data)

def centre\_of\_centroid(data):

centroid = sum(data) / len(data)

print("Centre of Centroid:", centroid)

*# Plot centroid*

plt.axhline(y=centroid, color='r', linestyle='--', label='Centre of Centroid')

plot\_membership\_function(data)

def centre\_of\_sum(data):

total\_sum = sum(data)

sum\_data = [data[i] / total\_sum *for* i *in* range(len(data))]

weighted\_sum = sum([(i + 1) \* sum\_data[i] *for* i *in* range(len(data))])

print("Centre of Sum:", weighted\_sum)

*# Plot centre of sum*

plt.axvline(x=weighted\_sum, color='r', linestyle='--', label='Centre of Sum')

plot\_membership\_function(data)

*# Main program*

data = [float(x) *for* x *in* input("Enter defuzzy values: ").split()]

*while* True:

print("\nMenu:")

print("1. Lambda Cut Method")

print("2. Weighted Average")

print("3. Height of Maxima")

print("4. First of Maxima")

print("5. Last of Maxima")

print("6. Mean of Maxima")

print("7. Centre of Centroid")

print("8. Centre of Sum")

print("9. Exit")

choice = input("Enter your choice: ")

*if* choice == '1':

lambda\_cut\_method(data)

*elif* choice == '2':

weighted\_average(data)

*elif* choice == '3':

height\_of\_maxima(data)

*elif* choice == '4':

first\_of\_maxima(data)

*elif* choice == '5':

last\_of\_maxima(data)

*elif* choice == '6':

mean\_of\_maxima(data)

*elif* choice == '7':

centre\_of\_centroid(data)

*elif* choice == '8':

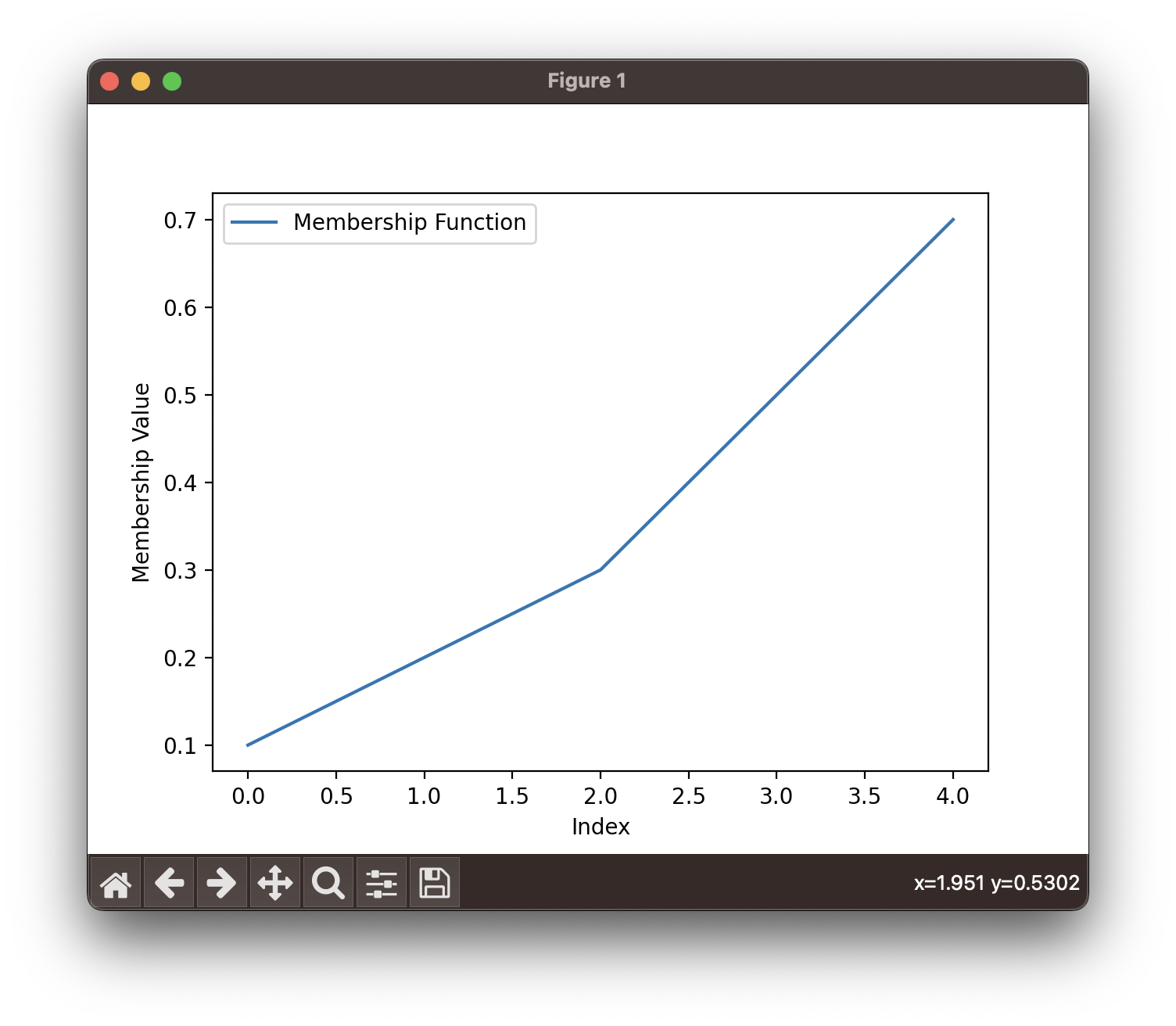
centre\_of\_sum(data)

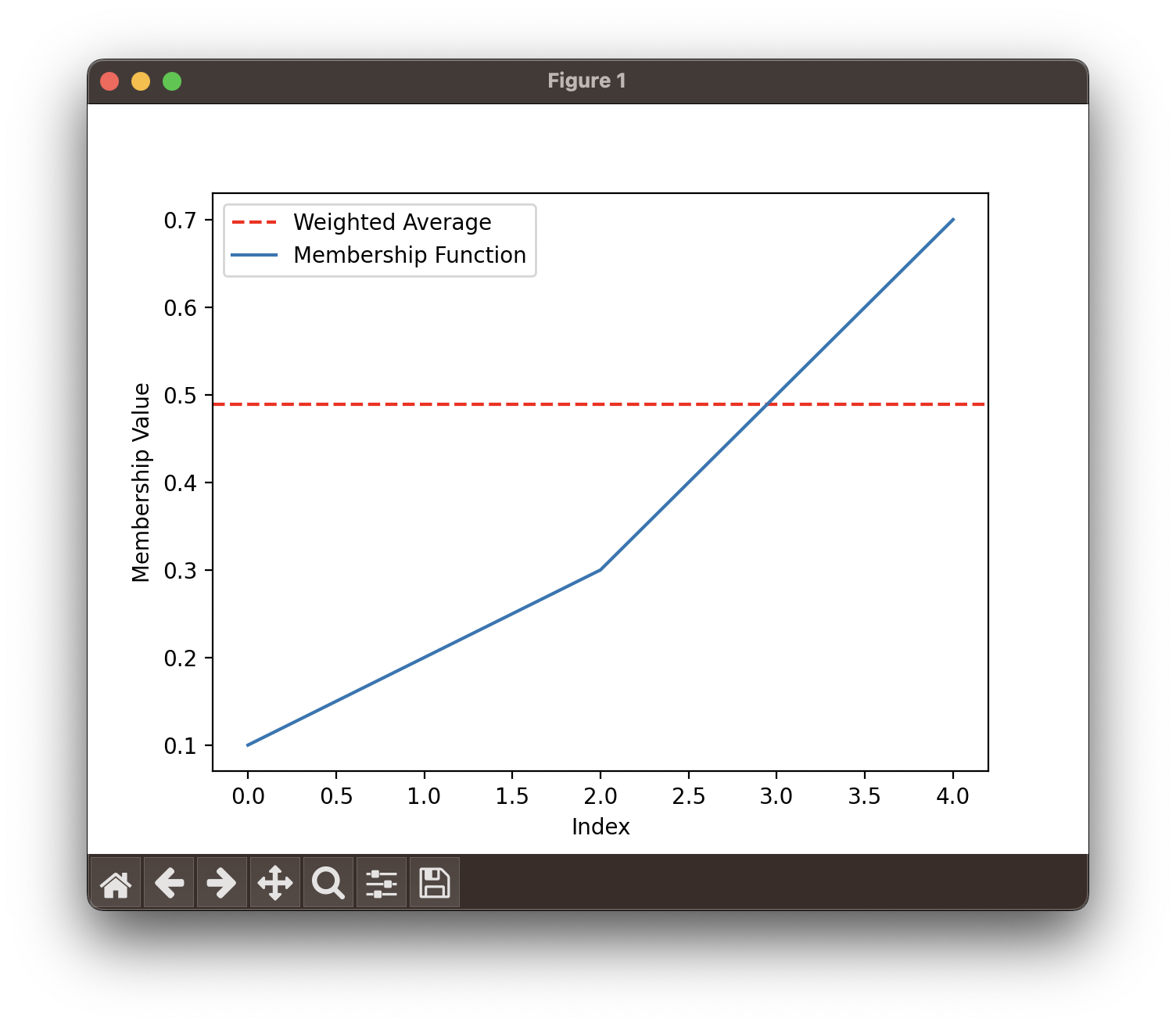
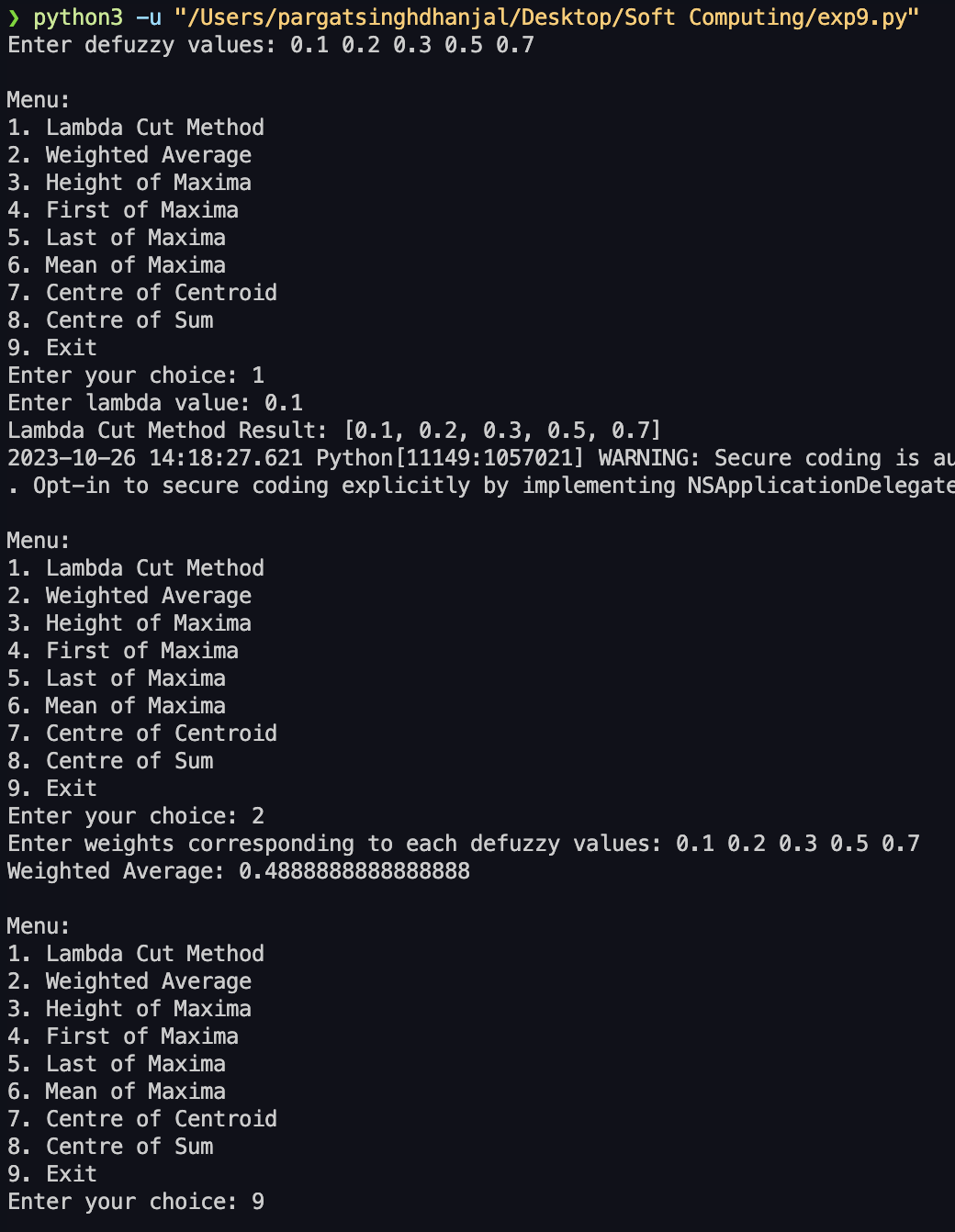
*elif* choice == '9':

*break*

*else*:

print("Invalid choice. Please enter a valid option.")



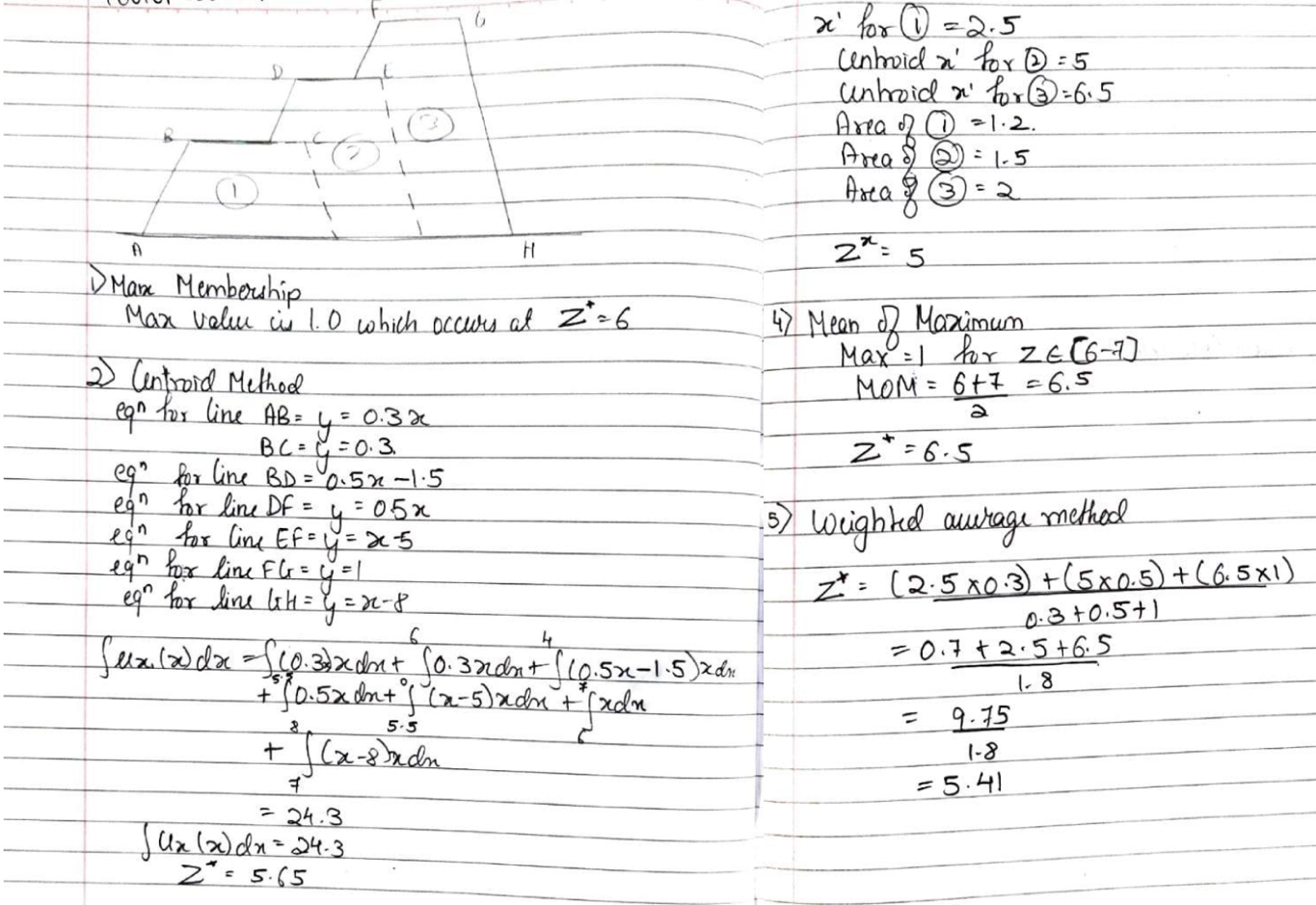


**Conclusion:** Implementation of defuzzification methods was done successfully.

**Post Lab Descriptive Questions :**

1. **Let there be 3 different fuzzy sets as shown in the figures below:-**

****

Now we shall calculate (manually) the defuzzified value using all the above methods one by one. 

**Date: \_\_\_\_\_\_\_\_ Signature of faculty in-charge**