Practical 7

Name: Sharvari Kishor More
Class: D20B
Roll No.: 35

Aim: To implement fuzzy set Properties

Theory:

1. Fuzzy Set Theory

Fuzzy set theory extends classical set theory to handle **degrees of membership**.

- In **classical sets**, an element either belongs to a set or not (membership is 0 or 1).
- In **fuzzy sets**, membership can take any value in **[0,1]**, representing the degree to which an element belongs.

2. Properties of Fuzzy Sets

In **fuzzy set theory**, properties are mathematical operations that define how two fuzzy sets interact.

If $\mu_A(x)$ and $\mu_B(x)$are membership values of an element xxx in fuzzy sets AAA and BBB

1. **Union (A ∪ B):**

$$\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

2. **Intersection (A ∩ B):**

$$\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

3. **Complement (A′):**

$$\mu_{A'}(x) = 1 - \mu_A(x)$$

4. **Scalar Multiplication:**

$$\mu_{\alpha A}(x) = \alpha \cdot \mu_A(x), \quad 0 \leq \alpha \leq 1$$

5. **Sum of Fuzzy Sets:**

$$\mu_{A+B}(x) = \min(1, \mu_A(x) + \mu_B(x))$$

3. Fuzzification

Fuzzification is the process of converting a **crisp input** (exact value) into **fuzzy values** by mapping it to membership functions.

Example (Temperature = 25°C):

- Cold → 0.0
- Warm → 0.5
- Hot → 0.25

4. Rule Base

Fuzzy rules are **IF–THEN statements**. Example for Temperature Control System:

- IF Temperature is Cold THEN Fan Speed is Low
- IF Temperature is Warm THEN Fan Speed is Medium
- IF Temperature is Hot THEN Fan Speed is High

5. Defuzzification

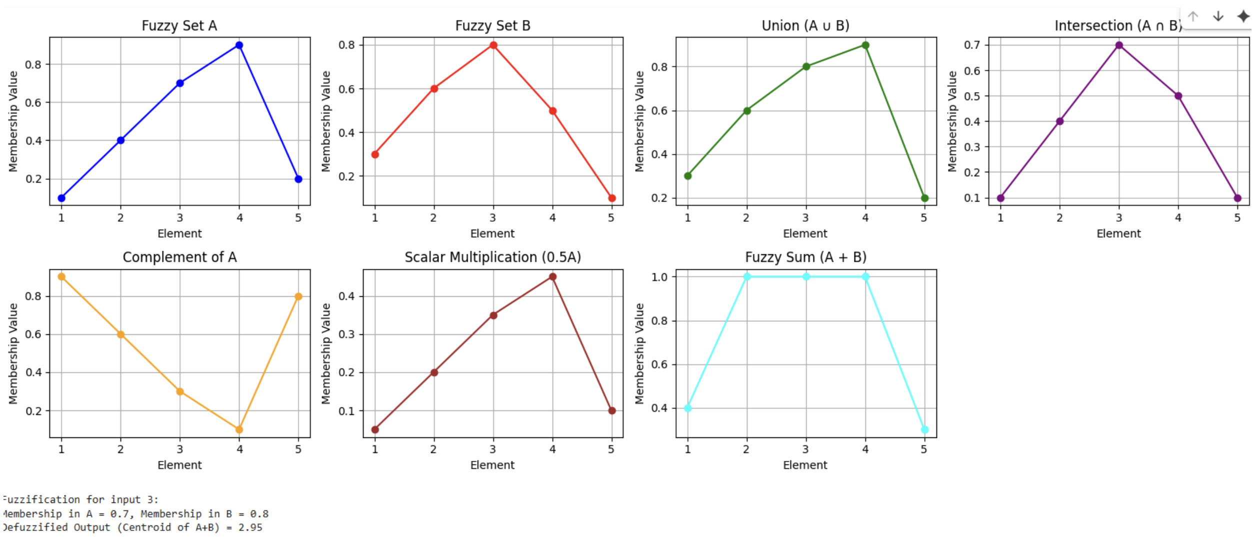
Defuzzification is the reverse process of fuzzification, where fuzzy values are converted back into a **single crisp output**.
Most common method: **Centroid Method**

Code:

General Fuzzy properties:

```
import matplotlib.pyplot as plt
import numpy as np

# Universe of discourse
x = [1, 2, 3, 4, 5]

# Two fuzzy sets A and B
A = [0.1, 0.4, 0.7, 0.9, 0.2]
B = [0.3, 0.6, 0.8, 0.5, 0.1]

# ---- Fuzzy Properties ----
A_union_B = [max(a, b) for a, b in zip(A, B)]              # Union
A_intersection_B = [min(a, b) for a, b in zip(A, B)]          # Intersection
A_complement = [1 - a for a in A]                           # Complement
A_scalar = [0.5 * a for a in A]                           # Scalar Multiplication
A_sum_B = [min(1, a + b) for a, b in zip(A, B)]             # Fuzzy Sum

# ---- Arrange plots in 2x4 grid ----
fig, axes = plt.subplots(2, 4, figsize=(16, 6))

# List of plots
plots = [
    ("Fuzzy Set A", A, 'blue'),
    ("Fuzzy Set B", B, 'red'),
    ("Union (A ∪ B)", A_union_B, 'green'),
    ("Intersection (A ∩ B)", A_intersection_B, 'purple'),
    ("Complement of A", A_complement, 'orange'),
    ("Scalar Multiplication (0.5A)", A_scalar, 'brown'),
    ("Fuzzy Sum (A + B)", A_sum_B, 'cyan')
]

# Draw each plot
for ax, (title, y_values, color) in zip(axes.flat, plots):
    ax.plot(x, y_values, 'o-', color=color)
    ax.set_title(title)
    ax.set_xlabel('Element')
    ax.set_ylabel('Membership Value')
    ax.grid(True)

# Hide the last empty subplot (since we have 7 plots, grid = 8 slots)
axes.flat[-1].axis('off')

plt.tight_layout()
plt.show()

# ---- Fuzzification (Crisp Input = 3) ----
crisp_input = 3
mu_A = A[crisp_input-1]
mu_B = B[crisp_input-1]
print(f"\nFuzzification for input {crisp_input}:")
print(f"Membership in A = {mu_A}, Membership in B = {mu_B}")

# ---- Defuzzification (Centroid Method) ----
def defuzz(x, mfx):
    numerator = sum([xi * mi for xi, mi in zip(x, mfx)])
    denominator = sum(mfx)
    return numerator/denominator if denominator != 0 else 0

# Defuzzify fuzzy sum (A+B)
crisp_output = defuzz(x, A_sum_B)
print(f"Defuzzified Output (Centroid of A+B) = {crisp_output:.2f}")
```

Output:



```
Fuzzification for input 3:
Membership in A = 0.7, Membership in B = 0.8
Defuzzified Output (Centroid of A+B) = 2.95
```

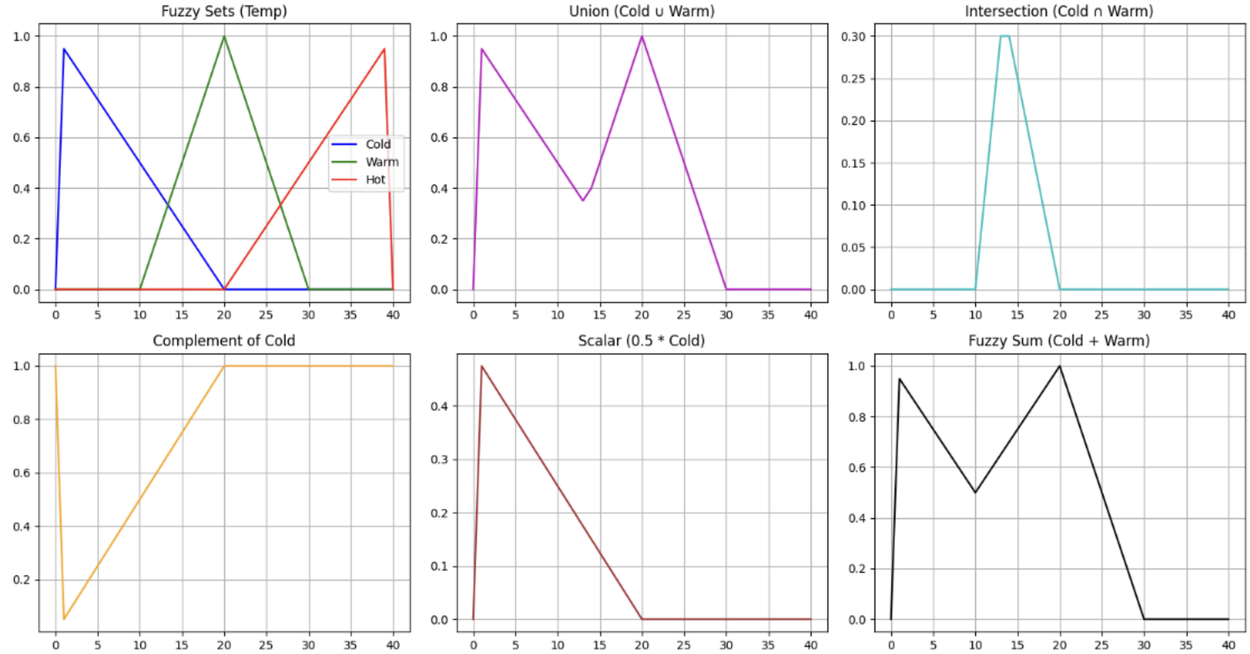
Code 2:

Example on Fuzzy properties

```
import numpy as np
import matplotlib.pyplot as plt

# Universe of discourse
x_temp = np.arange(0, 41, 1)

# ---- Define Membership Function (manual trimf) ----
def trimf(x, params):
    a, b, c = params
    y = np.zeros_like(x, dtype=float)
    for i, xi in enumerate(x):
        if xi <= a or xi >= c:
            y[i] = 0
        elif a < xi < b:
            y[i] = (xi - a) / (b - a)
        elif b <= xi < c:
            y[i] = (c - xi) / (c - b)
        elif xi == b:
            y[i] = 1
    return y

# Define fuzzy sets
cold = trimf(x_temp, [0, 0, 20])
warm = trimf(x_temp, [10, 20, 30])
hot  = trimf(x_temp, [20, 40, 40])

# ---- Fuzzy Properties ----
```

```
union_cold_warm = np.maximum(cold, warm)
inter_cold_warm = np.minimum(cold, warm)
comp_cold = 1 - cold
scalar_cold = 0.5 * cold
sum_cold_warm = np.minimum(1, cold + warm)

# ---- Plot side by side ----
fig, axes = plt.subplots(2, 3, figsize=(15, 8))

# Original sets
axes[0,0].plot(x_temp, cold, 'b', label="Cold")
axes[0,0].plot(x_temp, warm, 'g', label="Warm")
axes[0,0].plot(x_temp, hot, 'r', label="Hot")
axes[0,0].set_title("Fuzzy Sets (Temp)")
axes[0,0].legend(); axes[0,0].grid(True)

# Union
axes[0,1].plot(x_temp, union_cold_warm, 'm')
axes[0,1].set_title("Union (Cold ∪ Warm)")
axes[0,1].grid(True)

# Intersection
axes[0,2].plot(x_temp, inter_cold_warm, 'c')
axes[0,2].set_title("Intersection (Cold ∩ Warm)")
axes[0,2].grid(True)

# Complement
axes[1,0].plot(x_temp, comp_cold, 'orange')
axes[1,0].set_title("Complement of Cold")
axes[1,0].grid(True)

# Scalar Multiplication
axes[1,1].plot(x_temp, scalar_cold, 'brown')
axes[1,1].set_title("Scalar (0.5 * Cold)")
axes[1,1].grid(True)

# Fuzzy Sum
axes[1,2].plot(x_temp, sum_cold_warm, 'k')
axes[1,2].set_title("Fuzzy Sum (Cold + Warm)")
axes[1,2].grid(True)

plt.tight_layout()
plt.show()
```

Output:



Conclusion

- We successfully implemented the **properties of fuzzy sets,** including union, intersection, complement, scalar multiplication, and fuzzy sum.
- We applied **fuzzification** to convert crisp temperature input into fuzzy values and used a **rule base** to infer fan speed.
- Finally, using **defuzzification (centroid method)**, we obtained a crisp fan speed output from fuzzy rules.
- Thus, fuzzy set operations and fuzzy inference (fuzzification + defuzzification) were implemented and demonstrated.

This is the Colab file.