Name : Aryan Deepak Saraf
Div : D20B
Roll No. : 49

AI&DS2 Experiment 06

Aim: To implement Fuzzy Membership Functions.

Theory:

In the world of traditional logic, things are usually black or white, true or false. A statement is either completely true or completely false. However, real life is often more nuanced. Think about concepts like "tall," "hot," or "fast." When is someone exactly "tall"? It's not a precise cutoff. Fuzzy logic helps us deal with these kinds of imprecise concepts, making decisions more like how humans do.

At the heart of fuzzy logic are Fuzzy Membership Functions. These functions are like special rules that tell us how much something belongs to a certain group or category. Instead of just saying "yes" or "no" (0 or 1), they give us a degree of belonging, a value between 0 and 1. A value of 0 means it definitely doesn't belong, 1 means it definitely does, and anything in between means it partially belongs.

Imagine you're trying to decide if a person is "tall." In traditional logic, you might say anyone over 6 feet is tall. But with fuzzy logic, someone who is 5 feet 10 inches might be considered "somewhat tall" (e.g., a membership degree of 0.7), while someone who is 6 feet 2 inches is "very tall" (e.g., a membership degree of 0.95). The membership function defines this gradual transition.

There are several common types of fuzzy membership functions, each with a different shape that suits different situations:

1. Singleton Membership Function: This is the simplest type. It assigns a full membership (1) to just one specific point and zero membership to all other points. It's like saying, "Only this exact value is considered." For example, if you have a fuzzy set for "exactly 25 degrees Celsius," a singleton function would be perfect.
   Equation:
   $$\mu_A(x) = \begin{cases} 1, & x = a \\ 0, & x \neq a \end{cases}$$

2. Triangular Membership Function: As the name suggests, this function has a triangular shape. It's defined by three points: a starting point where membership is 0, a peak point where membership is 1, and an ending point where membership goes back to 0. It's very common because it's easy to understand and calculate. For instance, a "medium" temperature could be represented by a triangular function that peaks at 25°C and gradually decreases as you move away from it.

$$\mu_A(x) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a < x \le b \\ \frac{c-x}{c-b}, & b < x < c \\ 0, & x \ge c \end{cases}$$

3. Trapezoidal Membership Function: Similar to the triangular function, but it has a flat top. This means that a range of values will have full membership (1). It's defined by four points: two points where membership starts and ends at 0, and two points that define the flat top where membership is 1. This is useful when you want to define a range where something is definitely true. For example, a "comfortable" temperature might be between 20°C and 28°C, with a gradual increase and decrease outside that range.

$$\mu_A(x) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a < x \le b \\ 1, & b < x \le c \\ \frac{d-x}{d-c}, & c < x < d \\ 0, & x \ge d \end{cases}$$

4. Gaussian Membership Function: This function creates a smooth, bell-shaped curve. It's characterized by a center point and a spread (how wide the bell is). Gaussian functions are great for situations where you want very smooth transitions in membership, without sharp corners. For example, representing "optimal" performance where values close to the optimum have high membership, and membership smoothly drops off as you move further away.

$$\mu_A(x) = e^{-\frac{(x-c)^2}{2\sigma^2}}$$

These functions are crucial because they allow fuzzy systems to process vague information, make flexible decisions, and mimic human-like reasoning, which is very useful in areas like control systems, artificial intelligence, and data analysis.

Code :

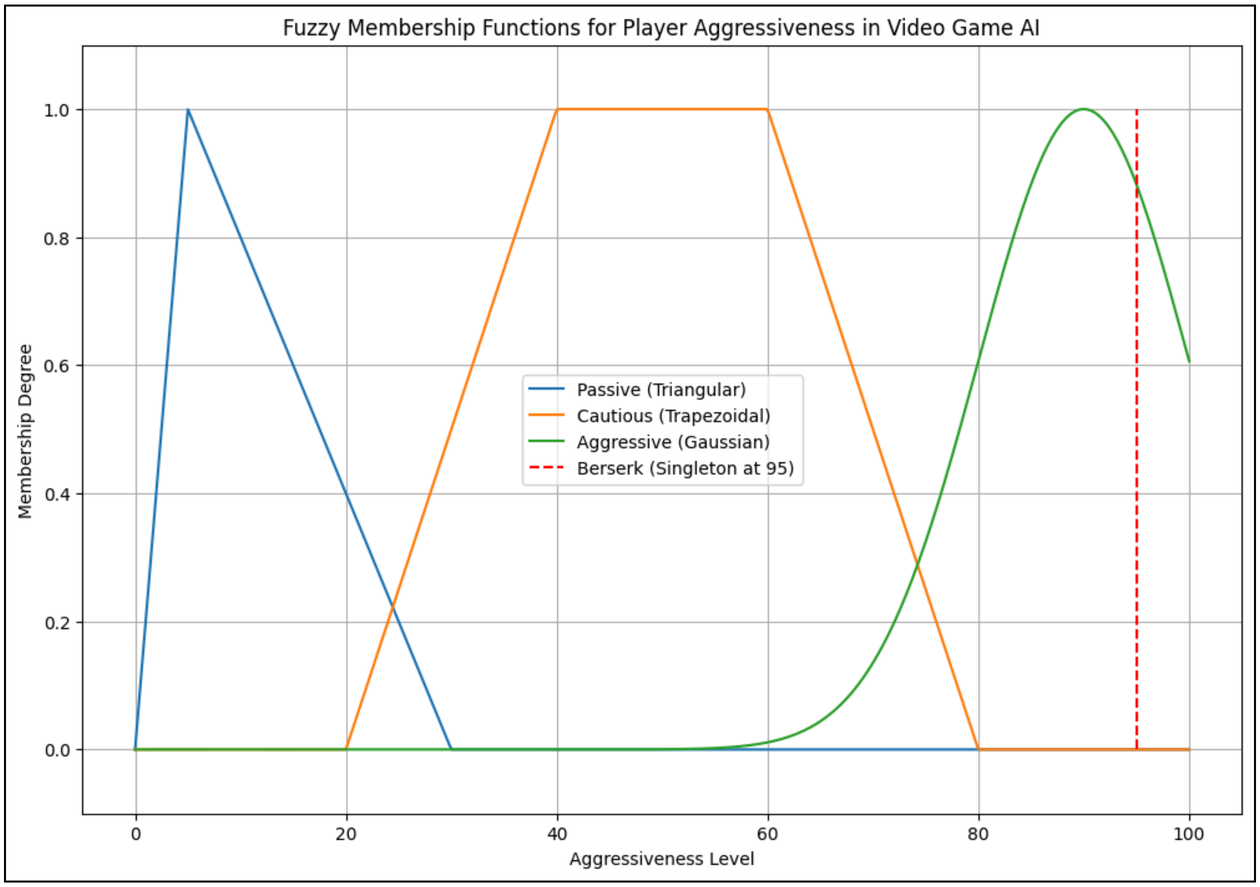
Step 1: Setup and Membership Function Definitions

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

# 1. Singleton Membership Function
def singleton_mf(x, x0):
  return np.where(x == x0, 1.0, 0.0)
```

```python
# 2. Triangular Membership Function
def triangular_mf(x, a, b, c):
    return np.maximum(np.minimum((x - a) / (b - a), (c - x) / (c - b)), 0.0)

# 3. Trapezoidal Membership Function
def trapezoidal_mf(x, a, b, c, d):
    return np.maximum(np.minimum(np.minimum((x - a) / (b - a), 1.0), (d - x) / (d - c)), 0.0)

# 4. Gaussian Membership Function
def gaussian_mf(x, c, sigma):
    return np.exp(-((x - c)**2) / (2 * sigma**2))
```

Step 2: Video Game AI Example - Player Aggressiveness

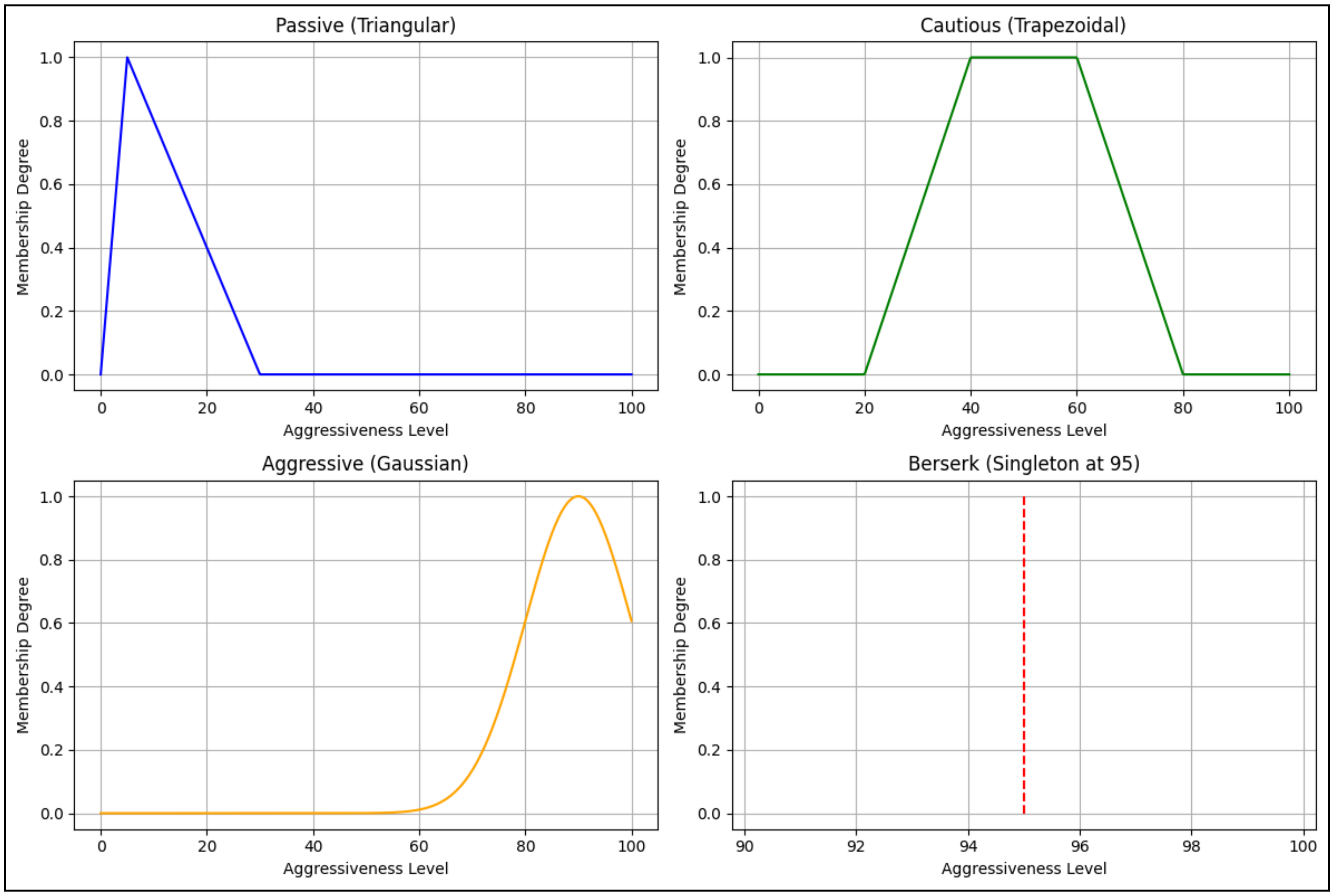
```python
# Define the universe of discourse for player aggressiveness (e.g., 0 to 100)
x_aggressiveness = np.linspace(0, 100, 500)

# Fuzzy sets for player aggressiveness
# 'Passive' - Low aggressiveness, represented by a triangular function
passive = triangular_mf(x_aggressiveness, 0, 5, 30)

# 'Cautious' - Moderate aggressiveness, represented by a trapezoidal function
cautious = trapezoidal_mf(x_aggressiveness, 20, 40, 60, 80)

# 'Aggressive' - High aggressiveness, represented by a Gaussian function
aggressive = gaussian_mf(x_aggressiveness, 90, 10)

# 'Berserk' - Very high aggressiveness, represented by a singleton function at a specific point
berserk_point = 95
berserk = singleton_mf(x_aggressiveness, berserk_point)

plt.figure(figsize=(12, 8))
plt.plot(x_aggressiveness, passive, label='Passive (Triangular)')
plt.plot(x_aggressiveness, cautious, label='Cautious (Trapezoidal)')
plt.plot(x_aggressiveness, aggressive, label='Aggressive (Gaussian)')
plt.vlines(berserk_point, 0, 1, colors='red', linestyles='dashed', label='Berserk (Singleton at 95)')

plt.title('Fuzzy Membership Functions for Player Aggressiveness in Video Game AI')
plt.xlabel('Aggressiveness Level')
plt.ylabel('Membership Degree')
plt.ylim(-0.1, 1.1)
plt.legend()
plt.grid(True)
plt.show()
```



Step 3: Individual Plots of All Four Functions

```
# Universe of discourse for aggressiveness
x_aggressiveness = np.linspace(0, 100, 500)

# Fuzzy sets
passive = triangular_mf(x_aggressiveness, 0, 5, 30)
cautious = trapezoidal_mf(x_aggressiveness, 20, 40, 60, 80)
aggressive = gaussian_mf(x_aggressiveness, 90, 10)
berserk_point = 95
berserk = singleton_mf(x_aggressiveness, berserk_point)

# Plotting in 2×2 layout
fig, axs = plt.subplots(2, 2, figsize=(12, 8))

# Passive (Triangular)
axs[0, 0].plot(x_aggressiveness, passive, color='blue')
axs[0, 0].set_title("Passive (Triangular)")
axs[0, 0].set_xlabel("Aggressiveness Level")
axs[0, 0].set_ylabel("Membership Degree")
axs[0, 0].grid(True)
```

```python
# Cautious (Trapezoidal)
axs[0, 1].plot(x_aggressiveness, cautious, color='green')
axs[0, 1].set_title("Cautious (Trapezoidal)")
axs[0, 1].set_xlabel("Aggressiveness Level")
axs[0, 1].set_ylabel("Membership Degree")
axs[0, 1].grid(True)

# Aggressive (Gaussian)
axs[1, 0].plot(x_aggressiveness, aggressive, color='orange')
axs[1, 0].set_title("Aggressive (Gaussian)")
axs[1, 0].set_xlabel("Aggressiveness Level")
axs[1, 0].set_ylabel("Membership Degree")
axs[1, 0].grid(True)

# Berserk (Singleton)
axs[1, 1].vlines(berserk_point, 0, 1, colors='red', linestyles='dashed')
axs[1, 1].set_title("Berserk (Singleton at 95)")
axs[1, 1].set_xlabel("Aggressiveness Level")
axs[1, 1].set_ylabel("Membership Degree")
axs[1, 1].grid(True)

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



<u>Conclusion:</u>

In conclusion, fuzzy membership functions are essential tools in fuzzy logic, allowing us to represent and work with imprecise information in a way that mirrors human reasoning. By understanding and implementing different types like Singleton, Triangular, Trapezoidal, and Gaussian functions, we can build more flexible and intelligent systems capable of handling real-world uncertainties.

49_Prac_07