Harshit Sharma D20b/51

Harshit Sharma D20b/51

Experiment No. 04 Aim: To implement different types of Fuzzy Membership Functions, including:

Singleton

Triangular

Trapezoidal

Gaussian

Theory: What is a Fuzzy Membership Function? In Fuzzy Logic, a membership function (MF) defines how much a value belongs to a fuzzy set. Instead of just being in or out like in classical sets (0 or 1), fuzzy sets allow values to partially belong — anywhere between 0 and 1.

For example:

In a fuzzy set called "Warm Temperature", 25°C might have a membership of 0.8 (fairly warm), while 15°C might have only 0.2.

Types of Fuzzy Membership Functions:

1. Singleton Membership Function It assigns full membership (1) to only one specific value and 0 to all others.

Useful when a single value is exactly and completely representative of a fuzzy set.

Formula:

μ (x) = { 1 if x = x 0 0 if x ≠ x 0 μ(x)={ 1 0

if x=x 0

if x □ =x 0

Example: If 20°C is considered a perfect "ideal temperature", we can define a singleton at 20.

2. Triangular Membership Function Defined by three points: a (start), b (peak), and c (end).

Creates a triangle shape.

It’s simple, efficient, and commonly used.

Formula:

μ (x) = { 0 $x \le a$ or $x \ge c$ $x - a$ $b - a$ $a \le x \le b$ $c - x$ $c - b$ $b \le x \le c$ μ(x)= ⎩ ⎨ ⎧

0 b−a x−a

c−b c−x

x≤a or x≥c a≤x≤b b≤x≤cExample: "Warm" temperature from 20°C to 40°C, with 30°C as the warmest.

3. Trapezoidal Membership Function Similar to triangular but has a flat top, meaning multiple values can have full membership (μ = 1).

Defined by four points: a, b, c, d.

Formula:

μ (x) = { 0 x ≤ a or x ≥ d x − a b − a a ≤ x ≤ b 1 b ≤ x ≤ c d − x d − c c ≤ x ≤ d μ(x)= ⎩ ⎨ ⎧

0 b−a x−a

1 d−c d−x

x≤a or x≥d a≤x≤b b≤x≤c c≤x≤d

Example: "Comfortable temperature" between 25°C and 30°C, gradually rising from 15°C and falling after 35°C.

4. Gaussian Membership Function Has a smooth bell-shaped curve, defined by a center (c) and standard deviation (σ).

It gives smooth transitions, making it very useful in real-world fuzzy systems.

Formula:

μ (x) = e − (x − c) 2 2 σ 2 μ(x)=e − 2σ 2

(x−c) 2

Example: "Hot" temperature centered at 38°C, with gradual rise/fall depending on σ. Why Use Fuzzy Membership Functions? Classical logic only supports 0 or 1 (True or False).

But in real life, many situations are not black and white.

Fuzzy logic uses membership functions to allow partial truth.

These functions help machines think more like humans, especially in:

Control systems (like air conditioners)

Decision-making (medical, financial, etc.)

AI and machine learning

Summary of Key Parameters: Function Type Parameters Shape Singleton x_0 Point Triangular a, b, c Triangle Trapezoidal a, b, c, d Trapezoid Gaussian c (center), σ (std) Bell Curve

Conclusion: In this experiment, we successfully implemented different types of fuzzy membership functions using Python. These functions help represent uncertain or imprecise information, which is a common requirement in fuzzy logic systems. Understanding and using these functions is essential for building intelligent and flexible systems in fields like artificial intelligence, control systems, and decision-making tools.

```
!pip install matplotlib numpy
```

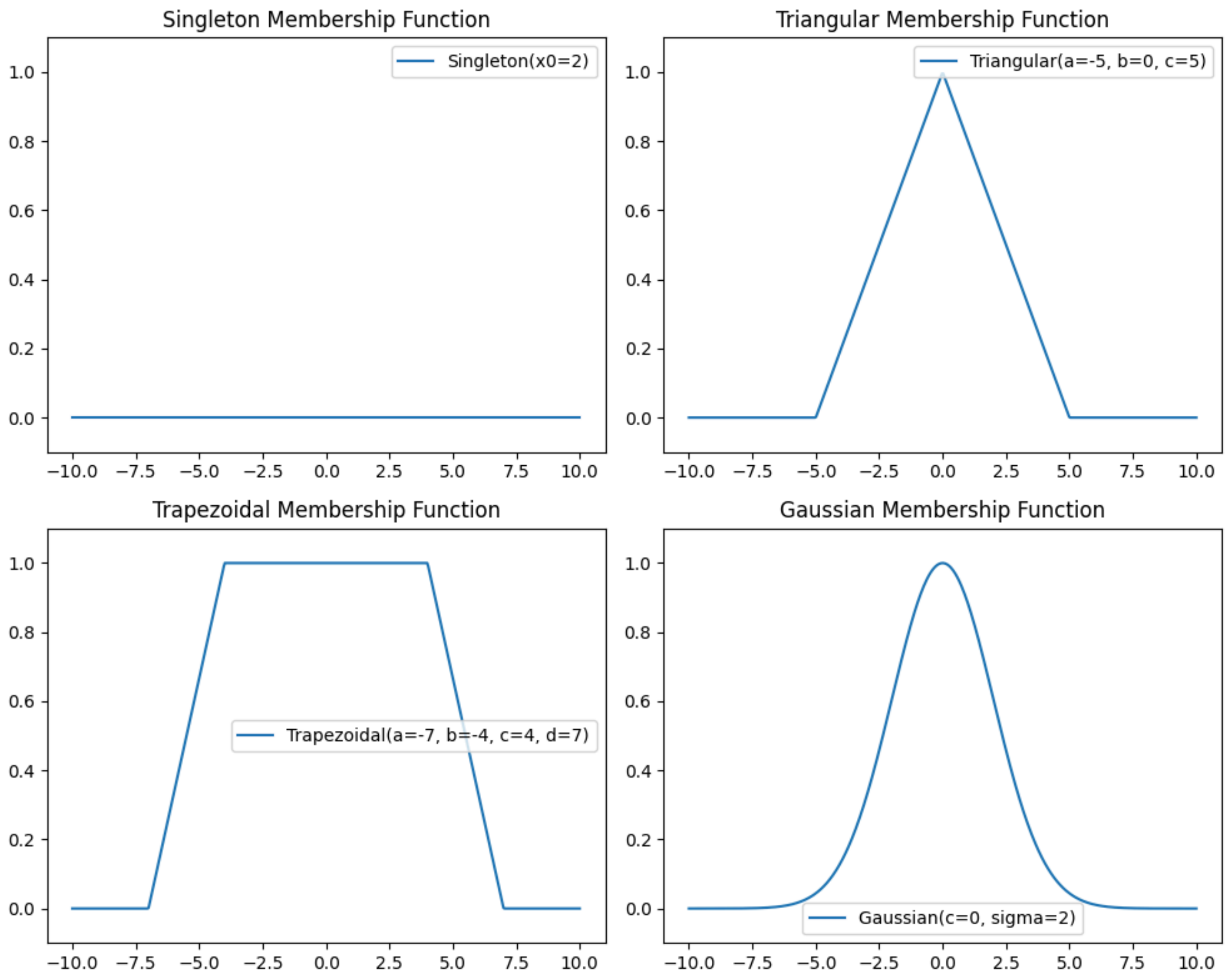
```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages 
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f
```

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

```
# Singleton Membership Function
def singleton_mf(x, x0):
    return np.where(x == x0, 1, 0)

# Triangular Membership Function
def triangular_mf(x, a, b, c):
    return np.maximum(np.minimum((x-a)/(b-a), (c-x)/(c-b)), 0)

# Trapezoidal Membership Function
def trapezoidal_mf(x, a, b, c, d):
    return np.maximum(np.minimum(np.minimum((x-a)/(b-a), 1), (d-x)/(d-c)), 0)

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

```
x = np.linspace(-10, 10, 500)  # Input range

plt.figure(figsize=(10, 8))

# Singleton
plt.subplot(2, 2, 1)
plt.plot(x, singleton_mf(x, 2), label="Singleton(x0=2)")
plt.title('Singleton Membership Function')
plt.ylim(-0.1, 1.1)
plt.legend()

# Triangular
```

```
    plt.subplot(2, 2, 2)
    plt.plot(x, triangular_mf(x, -5, 0, 5), label="Triangular(a=-5, b=0, c=5)")
    plt.title('Triangular Membership Function')
    plt.ylim(-0.1, 1.1)
    plt.legend()

    # Trapezoidal
    plt.subplot(2, 2, 3)
    plt.plot(x, trapezoidal_mf(x, -7, -4, 4, 7), label="Trapezoidal(a=-7, b=-4, c=4, d=7)")
    plt.title('Trapezoidal Membership Function')
    plt.ylim(-0.1, 1.1)
    plt.legend()

    # Gaussian
    plt.subplot(2, 2, 4)
    plt.plot(x, gaussian_mf(x, 0, 2), label="Gaussian(c=0, sigma=2)")
    plt.title('Gaussian Membership Function')
    plt.ylim(-0.1, 1.1)
    plt.legend()

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



Singleton Membership Function

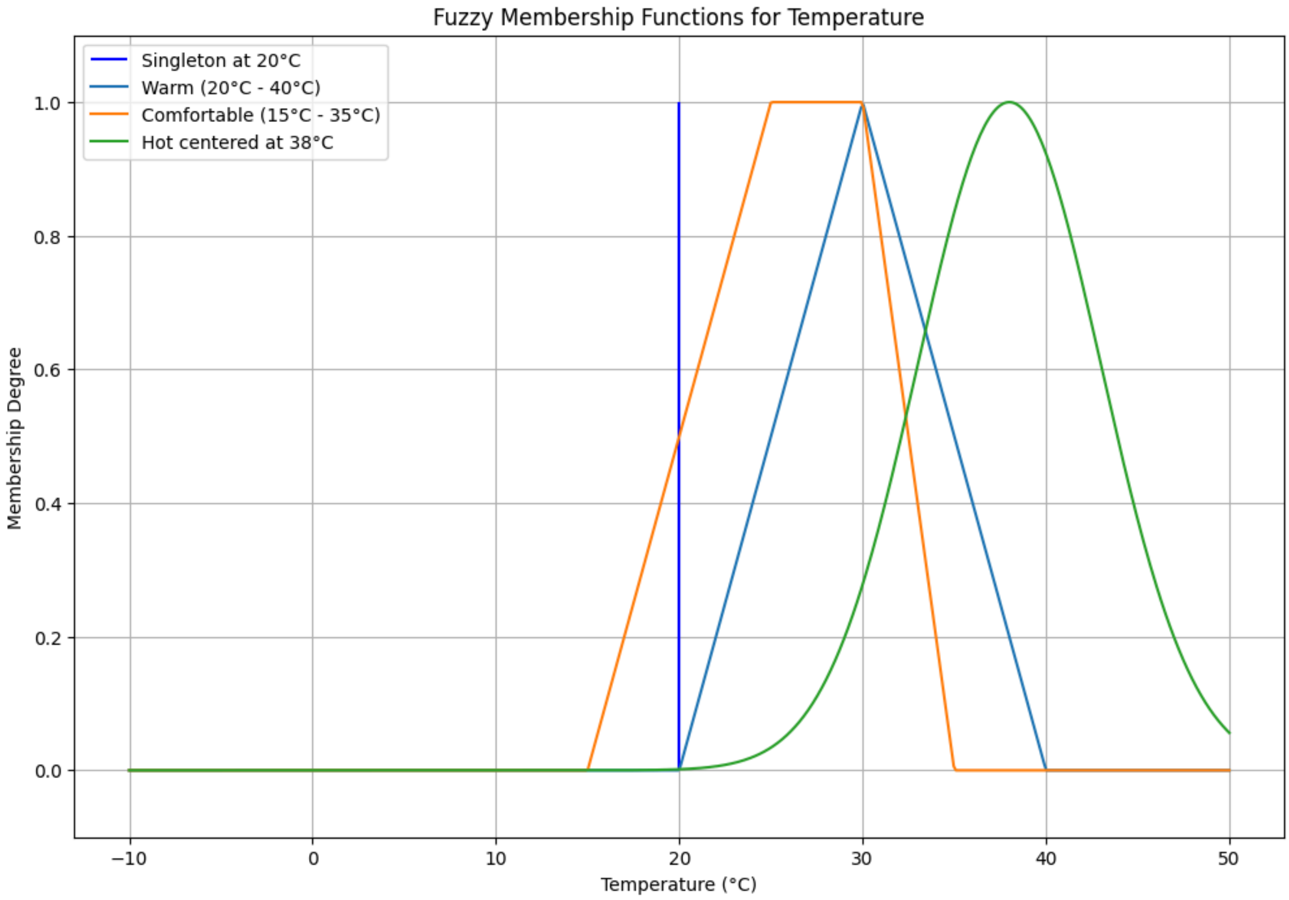
Triangular Membership Function

Trapezoidal Membership Function

Gaussian Membership Function

```
x = np.linspace(-10, 50, 500)

singleton_temp = singleton_mf(x, 20)
warm_temp = triangular_mf(x, 20, 30, 40)
comfortable_temp = trapezoidal_mf(x, 15, 25, 30, 35)
hot_temp = gaussian_mf(x, 38, 5)

plt.figure(figsize=(12, 8))
plt.vlines(20, 0, 1, colors='blue', label="Singleton at 20°C", linestyles='solid')
plt.plot(x, warm_temp, label="Warm (20°C - 40°C)")
plt.plot(x, comfortable_temp, label="Comfortable (15°C - 35°C)")
```

```
    plt.plot(x, hot_temp, label="Hot centered at 38°C")
    plt.title('Fuzzy Membership Functions for Temperature')
    plt.xlabel('Temperature (°C)')
    plt.ylabel('Membership Degree')
    plt.ylim(-0.1, 1.1)
    plt.grid(True)
    plt.legend()
    plt.show()
```


Fuzzy Membership Functions for Temperature
Singleton at 20°C
Warm (20°C - 40°C)
Comfortable (15°C - 35°C)
Hot centered at 38°C
Membership Degree
Temperature (°C)