Name: **Harsh Pramod Padyal**

Roll No : **40**

Div: **D20B**

Experiment 08

Aim: To design a Fuzzy control system using Fuzzy tool / library.

---

1. Introduction to Fuzzy Control Systems

Fuzzy control systems are intelligent control methodologies that use fuzzy logic to model complex, nonlinear, or uncertain systems where traditional mathematical models are difficult to define. These systems mimic human decision-making by using linguistic rules and approximate reasoning.

Key Features of Fuzzy Control Systems

- **Handles imprecise data** (no need for exact mathematical models).
- **Works with human-like reasoning** (IF-THEN rules).
- **Robust to noise and disturbances**.
- **Adaptable to different applications** (traffic control, washing machines, industrial automation).

---

2. Components of a Fuzzy Control System

A fuzzy control system consists of four main components:

(1) Fuzzification

- Converts crisp (numerical) inputs into fuzzy values using **membership functions**.
- Example:

$$\text{Input: Speed} = 45 \text{ km/h} \rightarrow \text{Fuzzy: "Medium" with } \mu = 0.7$$

(2) Fuzzy Rule Base

- Contains **IF-THEN rules** based on expert knowledge.
- Example:

$$\text{IF (Speed is High) AND (Distance is Low) THEN (Brake Hard)}$$

(3) Inference Engine

- Applies fuzzy logic operations (AND, OR, NOT) to evaluate rules.
- Common methods:
    - **Mamdani Inference** (output is a fuzzy set).

- **Sugeno Inference** (output is a mathematical function).

(4) Defuzzification

- Converts fuzzy output into a crisp (usable) value.
- Common methods:
  - **Centroid (Center of Gravity)**:
$$y^* = \frac{\int y \cdot \mu(y)\, dy}{\int \mu(y)\, dy}$$
  - **Max-Membership (Height Method)**:
$$y^* = \text{Point where } \mu(y) \text{ is maximum}$$

---

3. Steps to Design a Fuzzy Control System

To implement a fuzzy control system (as in Experiment 8), follow these steps:

Step 1: Define Input and Output Variables

- **Inputs (Antecedents)**: Variables that affect the system (e.g., temperature, speed).
- **Output (Consequent)**: The control action (e.g., fan speed, braking force).

Step 2: Define Membership Functions

- Assign linguistic terms (Low, Medium, High) to each variable.
- Example (Triangular Membership Function for Temperature):
$$\mu_{\text{Medium}}(x) = \begin{cases} 0 & \text{if } x \leq 20 \\ \frac{x-20}{10} & \text{if } 20 < x \leq 30 \\ \frac{40-x}{10} & \text{if } 30 < x \leq 40 \\ 0 & \text{if } x > 40 \end{cases}$$

Step 3: Create Fuzzy Rules

- Use IF-THEN rules based on expert knowledge.
- Example:
$$\text{IF (Temperature is High) THEN (Cooler Speed is Maximum)}$$

Step 4: Implement Inference Engine

- Combine rules using fuzzy logic operations (AND = min, OR = max).
- Example:
$$\text{Rule Output} = \min(\mu_{\text{High}}(x), \mu_{\text{Low}}(y))$$

Step 5: Perform Defuzzification

- Convert fuzzy output into a crisp value.
- Example (Centroid Method):
$$\text{Final Output} = \frac{\sum(\text{Membership} \times \text{Value})}{\sum \text{Membership}}$$

4. Applications of Fuzzy Control Systems

- **Traffic Light Control**: Adjusts signal timing based on traffic density.
- **Washing Machines**: Determines wash time based on dirt level.
- **Air Conditioners**: Adjusts cooling based on temperature and humidity.
- **Industrial Automation**: Controls robotic arms and conveyor systems.

---

5. Advantages of Fuzzy Control Systems

✓ **No need for precise mathematical models**.
✓ **Handles uncertainty and imprecision effectively**.
✓ **Easy to modify rules without changing hardware**.
✓ **Works well with human expert knowledge**.

Install and import required libraries

```
!pip install scikit-fuzzy
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
```

```
Collecting scikit-fuzzy
  Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl.metadata (2.6 kB)
Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl (920 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 920.8/920.8 kB 11.4 MB/s eta 0:00:00
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.5.0
```

Define antecedents (inputs) and consequent (output)

```
# Input 1: Traffic Density (0-100%)
traffic_density = ctrl.Antecedent(np.arange(0, 101, 1), 'traffic_density')
# Input 2: Emergency Vehicle Presence (0-100% likelihood)
emergency = ctrl.Antecedent(np.arange(0, 101, 1), 'emergency')
# Output: Green Light Duration (0-60 seconds)
green_duration = ctrl.Consequent(np.arange(0, 61, 1), 'green_duration')
```

Define membership functions for inputs

```
# Traffic density categories
traffic_density['low'] = fuzz.trimf(traffic_density.universe, [0, 0, 30])
traffic_density['medium'] = fuzz.trimf(traffic_density.universe, [20, 50, 80])
traffic_density['high'] = fuzz.trimf(traffic_density.universe, [70, 100, 100])
```

```
# Emergency vehicle presence categories
emergency['none'] = fuzz.trimf(emergency.universe, [0, 0, 30])
```

```
emergency['possible'] = fuzz.trimf(emergency.universe, [20, 50, 80])
emergency['confirmed'] = fuzz.trimf(emergency.universe, [70, 100, 100])
```

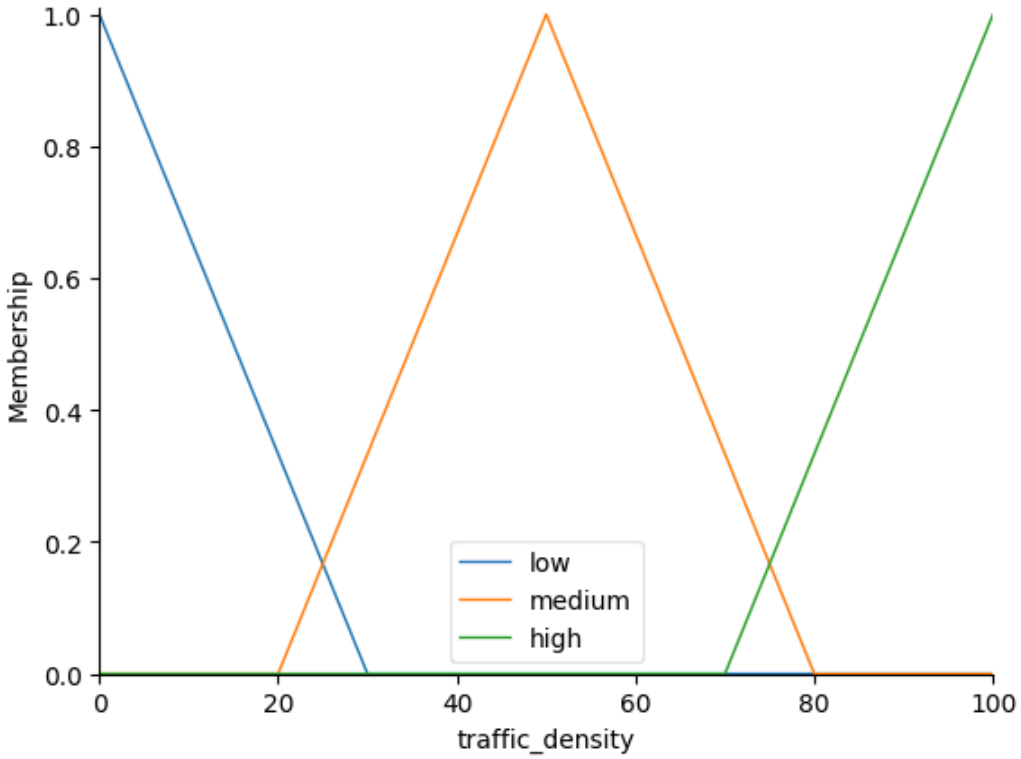
Define membership functions for output

```
green_duration['very_short'] = fuzz.trimf(green_duration.universe, [0, 0, 15])
green_duration['short'] = fuzz.trimf(green_duration.universe, [10, 20, 30])
green_duration['medium'] = fuzz.trimf(green_duration.universe, [25, 35, 45])
green_duration['long'] = fuzz.trimf(green_duration.universe, [40, 50, 60])
green_duration['very_long'] = fuzz.trimf(green_duration.universe, [55, 60, 60])
```

Visualize membership functions

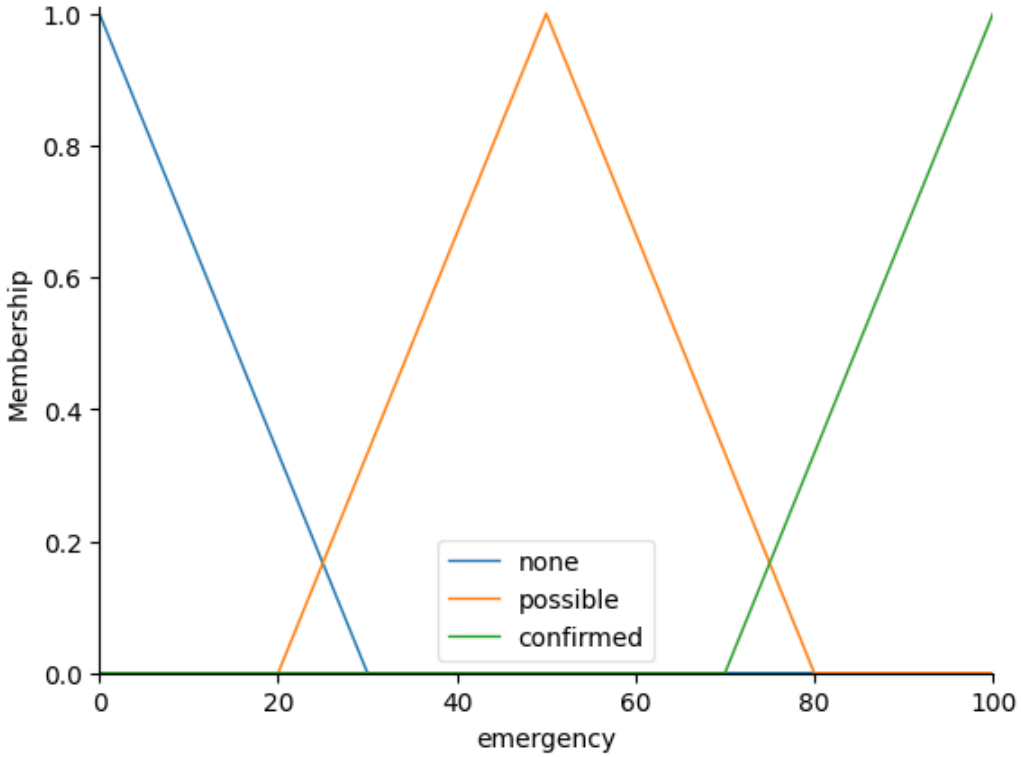
```
print("Traffic Density Membership Functions")
traffic_density.view()
plt.show()
```

```
Traffic Density Membership Functions
```



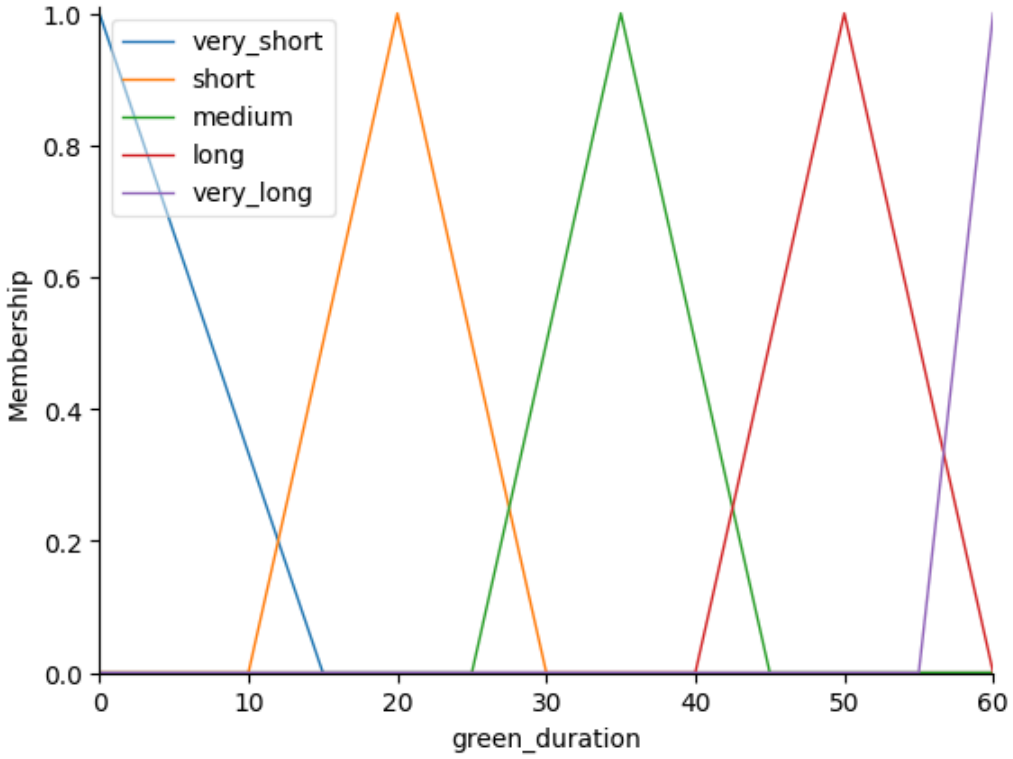

```
print("\nEmergency Vehicle Presence Membership Functions")
emergency.view()
plt.show()
```

```
Emergency Vehicle Presence Membership Functions
```




```
print("\nGreen Light Duration Membership Functions")
green_duration.view()
plt.show()
```

```
Green Light Duration Membership Functions
```



Define fuzzy rules

```
rule1 = ctrl.Rule(emergency['confirmed'], green_duration['very_long'])
rule2 = ctrl.Rule(traffic_density['high'] & emergency['none'], green_duration['long'])
rule3 = ctrl.Rule(traffic_density['high'] & emergency['possible'], green_duration['medium'])
rule4 = ctrl.Rule(traffic_density['medium'] & emergency['none'], green_duration['medium'])
rule5 = ctrl.Rule(traffic_density['medium'] & emergency['possible'], green_duration['short'])
rule6 = ctrl.Rule(traffic_density['low'] & emergency['none'], green_duration['short'])
rule7 = ctrl.Rule(traffic_density['low'] & emergency['possible'], green_duration['very_short'])
rule8 = ctrl.Rule(emergency['possible'] & traffic_density['high'], green_duration['medium'])
```

Create control system

```
traffic_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8])
traffic_sim = ctrl.ControlSystemSimulation(traffic_ctrl)
```

Create function to compute green light duration

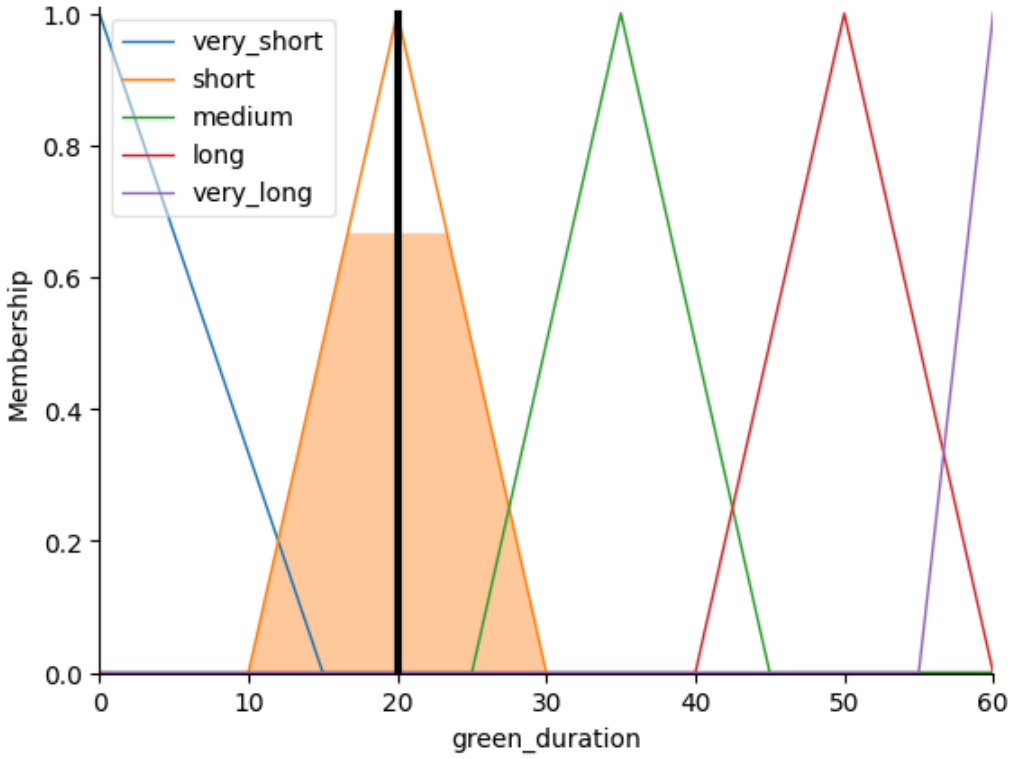
```
def compute_green_duration(traffic, emergency_level):
    # Input validation
    if traffic < 0 or traffic > 100:
        raise ValueError("Traffic density must be between 0-100")
    if emergency_level < 0 or emergency_level > 100:
        raise ValueError("Emergency level must be between 0-100")

    # Pass inputs to the control system
    traffic_sim.input['traffic_density'] = traffic
    traffic_sim.input['emergency'] = emergency_level

    # Compute the result
    traffic_sim.compute()

    # Visualize the result
    green_duration.view(sim=traffic_sim)
    plt.show()

    return traffic_sim.output['green_duration']
```

Main execution (user input)

```
if __name__ == "__main__":
    print("Traffic Light Duration Calculator using Fuzzy Logic")
    print("-----------------------------------------------------")

    while True:
        try:
            traffic = float(input("Enter traffic density (0-100%): "))
            emergency_level = float(input("Enter emergency vehicle likelihood (0-100%): "))

            duration = compute_green_duration(traffic, emergency_level)
            print(f"\nRecommended green light duration: {duration:.2f} seconds")
```

```
        another = input("\nCalculate another? (y/n): ")
        if another.lower() != 'y':
            break

    except ValueError as e:
        print(f"Error: {e}. Please enter valid numbers.")
```

```
Traffic Light Duration Calculator using Fuzzy Logic
--------------------------------------------------
Enter traffic density (0-100%): 40
Enter emergency vehicle likelihood (0-100%): 43
```




```
Recommended green light duration: 20.00 seconds

Calculate another? (y/n): n
```

Conclusion

Fuzzy control systems provide an efficient way to automate decision-making in complex environments. By using membership functions, rule-based reasoning, and defuzzification, they bridge the gap between human intuition and machine precision. This theory aligns with Experiment 8, where you design a fuzzy control system using Python libraries like `scikit-fuzzy`.