Name : Naikwadi Yash Shivdas

AI&DS II Experiment 08

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

THEORY:

Introduction to Fuzzy Control Systems

A fuzzy control system uses fuzzy logic instead of classical (crisp) logic to make decisions and control various processes. Fuzzy logic, introduced by Lotfi Zadeh in 1965, enables systems to handle uncertain, imprecise, or vague information, mimicking the way humans make decisions.

Unlike traditional binary logic (where variables are true or false, 0 or 1), fuzzy logic variables can take values between 0 and 1. This property allows the fuzzy control system to interpret "real-world" information more naturally and build intelligent controllers for complex or poorly defined systems.

Components of a Fuzzy Control System

A typical fuzzy control system consists of:

- Fuzzifier: Converts crisp (precise) input values into fuzzy values using membership functions.
- Fuzzy Rule Base: Contains IF-THEN rules based on expert knowledge or observed behavior.
- Inference Engine: Processes and evaluates the rules, combining them to make decisions.
- Defuzzifier: Converts fuzzy output values back to crisp values for the real-world system.

Steps to Design a Fuzzy Control System

1. Identify Inputs and Outputs:
   Decide which variables will be the inputs (e.g., load size, dirtiness) and what the output should be (e.g., washing time).
2. Fuzzification:
   - Define fuzzy sets for each input and output variable.
   - Use membership functions, which can be triangular, trapezoidal, or bell-shaped, to represent these sets.

   Mathematical Example (Triangular Membership Function):

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

   Where:
   - a,b,c are the three corners of the triangle.
   - $\mu_A(x)$: Membership degree of element x.
3. Build Rule Base:
   - Construct fuzzy IF-THEN rules that link inputs to outputs using linguistic variables (e.g., "IF load size is large AND dirtiness is high THEN wash time is long").
   - Each rule represents an expert's understanding or control policy.
4. Inference Mechanism:

  - Apply the rules to the input values by evaluating the degree to which each rule applies.
  - Aggregation combines the results of all rules.
  - The most common method is the Mamdani inference, using minimum and maximum operators.
5. Defuzzification:
  - Convert the aggregated fuzzy output back to a crisp (numerical) value.
  - Common methods: Centroid (center of gravity), Maximum membership.

Equation (Centroid Defuzzification):

$$y^* = \frac{\int y \cdot \mu_B(y)\, dy}{\int \mu_B(y)\, dy}$$

Where:
  - $y*$ is the defuzzified output.
  - $\mu_B(y)$: Membership degree of the output.

Advantages of Fuzzy Control Systems
- Can handle vague or incomplete data.
- No need for an exact mathematical model of the system.
- Mimics human thinking for flexible and smooth decision-making.
- Used in many real-world applications like washing machines, air conditioning, automotive systems, and more.

Example in Real Life: Washing Machine

A fuzzy controller in a washing machine can take input parameters such as the load size and dirtiness level. These inputs are processed using fuzzy logic to determine an appropriate wash time:
- If the load size is large and dirtiness is high, then the wash time is set to long.
- If the load size is small and dirtiness is low, then the wash time is set to short.

This ensures efficient washing cycles tailored to the laundry needs, improving wash quality and conserving resources.

CODE:

Install and Import Required Libraries

```
# Install scikit-fuzzy (only necessary in Colab or a fresh environment)
!pip install -q scikit-fuzzy

import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
```

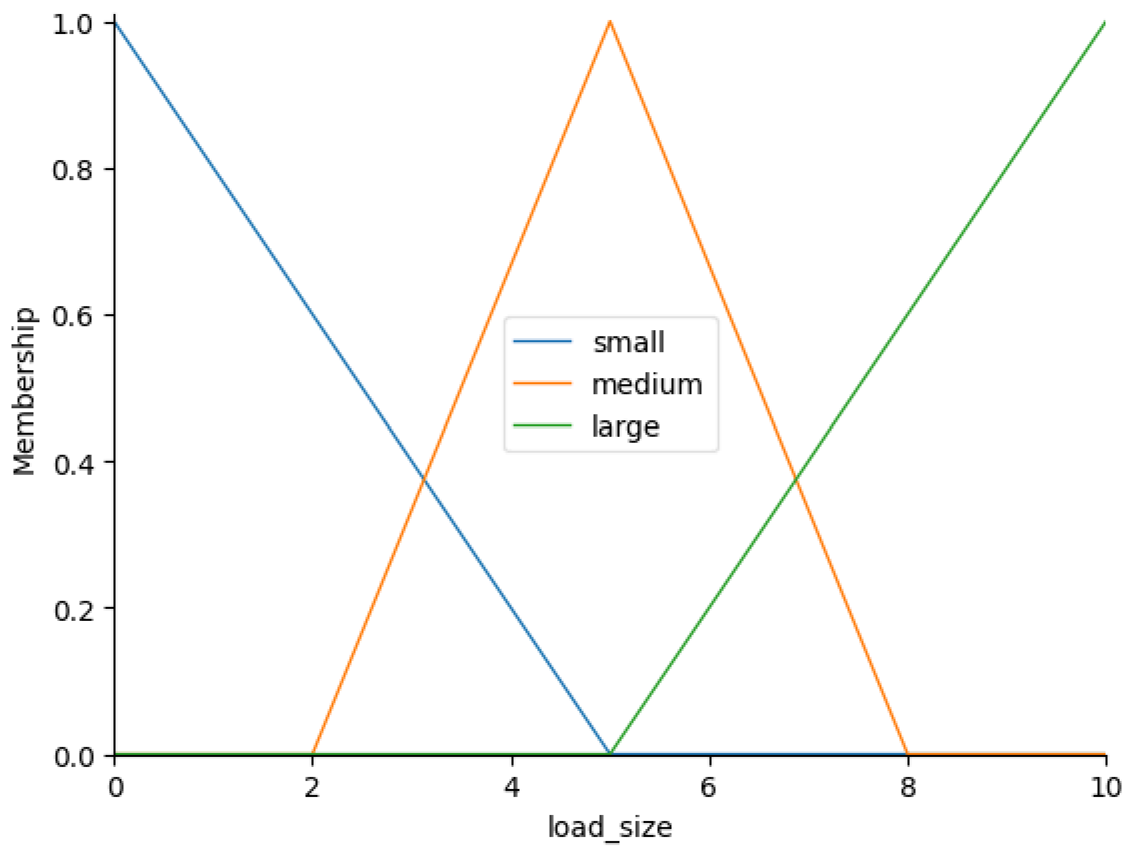
Define Fuzzy Variables for a New Example

```
# Define fuzzy variables
load_size = ctrl.Antecedent(np.arange(0, 11, 1), 'load_size')     # kg, 0-10
dirtiness = ctrl.Antecedent(np.arange(0, 101, 1), 'dirtiness')     # %, 0-100
rinse_time = ctrl.Consequent(np.arange(5, 61, 1), 'rinse_time')     # minutes,
5-60
```

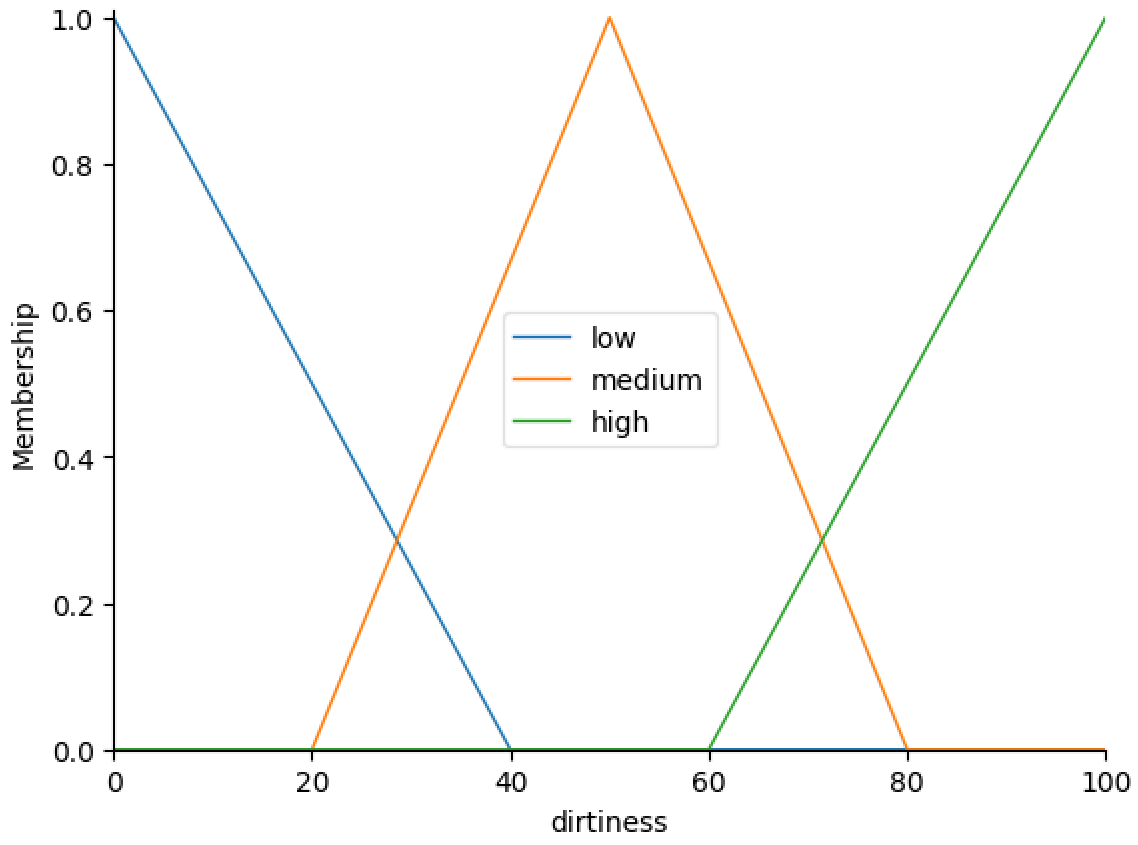
```
# Membership functions for load_size
load_size['small'] = fuzz.trimf(load_size.universe, [0, 0, 5])
load_size['medium'] = fuzz.trimf(load_size.universe, [2, 5, 8])
load_size['large'] = fuzz.trimf(load_size.universe, [5, 10, 10])

# Membership functions for dirtiness
dirtiness['low'] = fuzz.trimf(dirtiness.universe, [0, 0, 40])
dirtiness['medium'] = fuzz.trimf(dirtiness.universe, [20, 50, 80])
dirtiness['high'] = fuzz.trimf(dirtiness.universe, [60, 100, 100])

# Membership functions for rinse_time
rinse_time['short'] = fuzz.trimf(rinse_time.universe, [5, 10, 20])
rinse_time['normal'] = fuzz.trimf(rinse_time.universe, [15, 25, 40])
rinse_time['long'] = fuzz.trimf(rinse_time.universe, [30, 45, 60])
```

Visualize Membership Functions (Optional but Recommended)

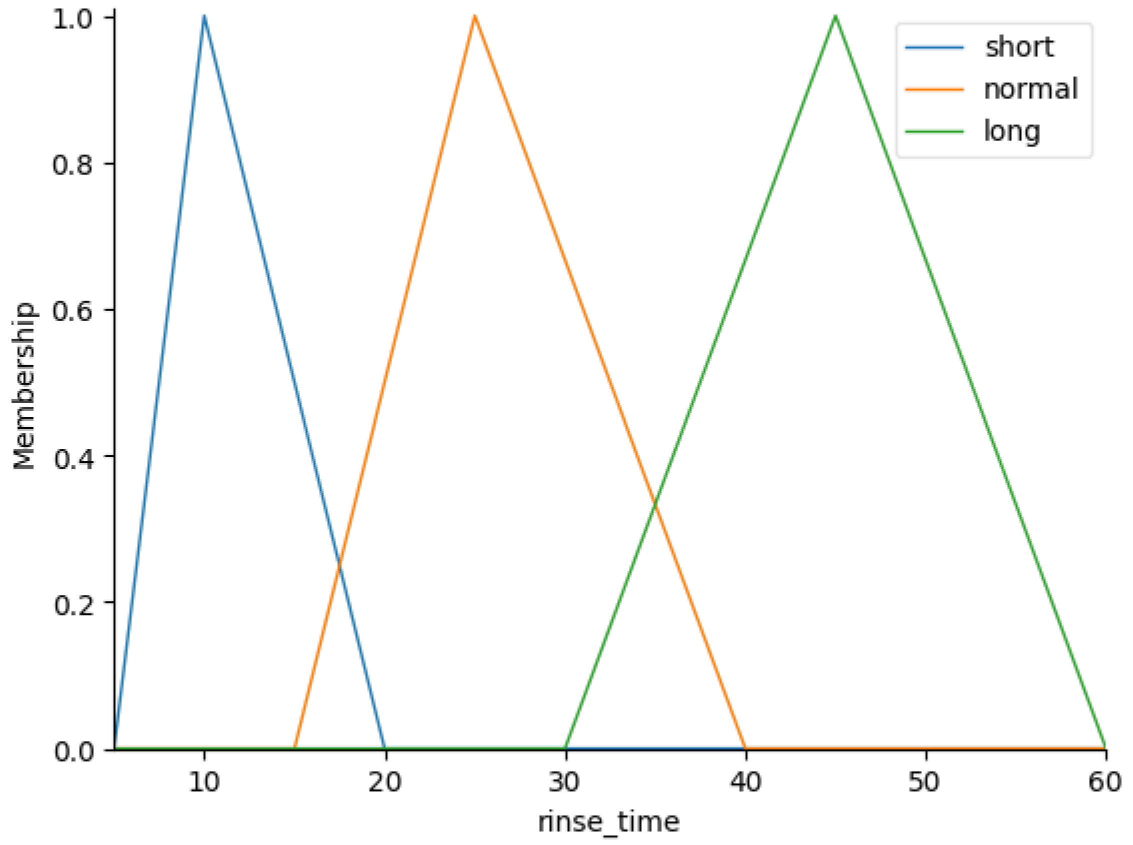
```
load_size.view()
```



```
dirtiness.view()
```



```
rinse_time.view()
```



Define Fuzzy Rules

```
rule1 = ctrl.Rule(load_size['small'] & dirtiness['low'], rinse_time['short'])
```

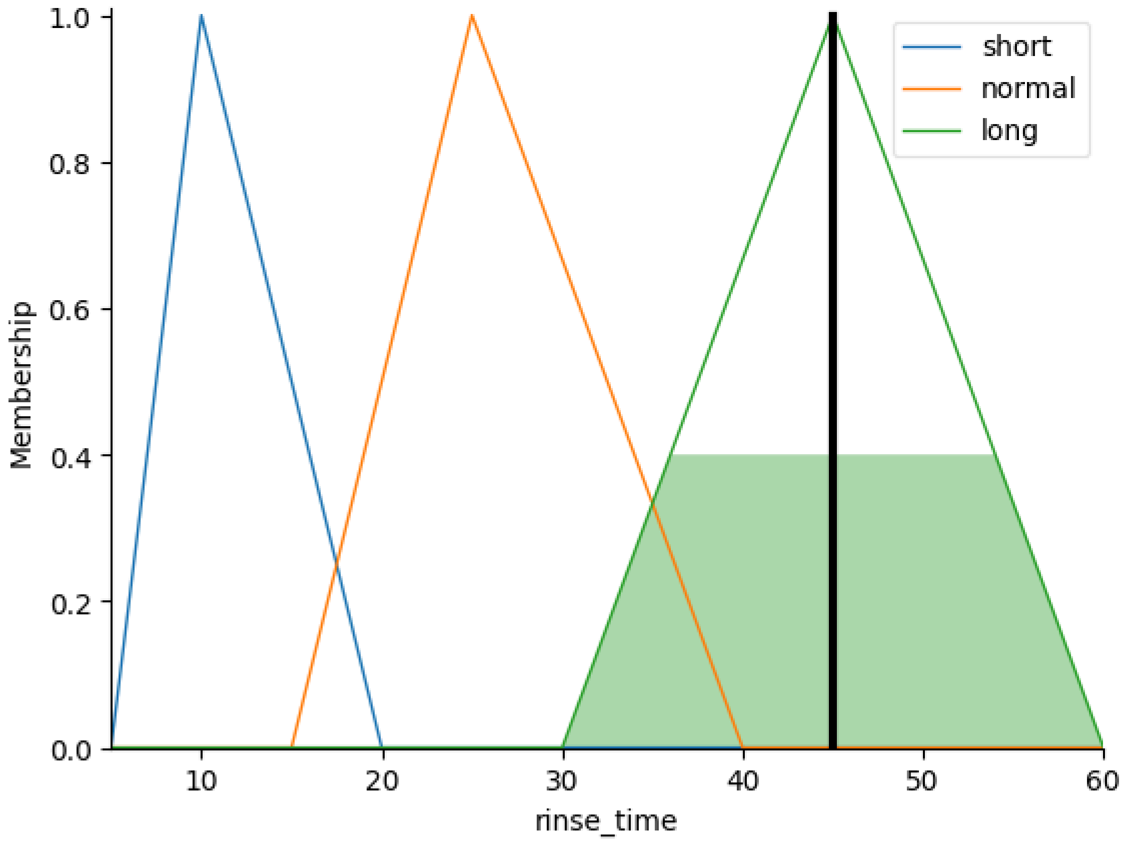
```
rule2 = ctrl.Rule(load_size['small'] & dirtiness['medium'], rinse_time['normal'])
rule3 = ctrl.Rule(load_size['small'] & dirtiness['high'], rinse_time['normal'])
rule4 = ctrl.Rule(load_size['medium'] & dirtiness['low'], rinse_time['short'])
rule5 = ctrl.Rule(load_size['medium'] & dirtiness['medium'], rinse_time['normal'])
rule6 = ctrl.Rule(load_size['medium'] & dirtiness['high'], rinse_time['long'])
rule7 = ctrl.Rule(load_size['large'] & dirtiness['low'], rinse_time['normal'])
rule8 = ctrl.Rule(load_size['large'] & dirtiness['medium'], rinse_time['long'])
rule9 = ctrl.Rule(load_size['large'] & dirtiness['high'], rinse_time['long'])
```

Create and Simulate the Fuzzy Control System

```
# Create the control system
rinse_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9])
wash_sim = ctrl.ControlSystemSimulation(rinse_ctrl)
```

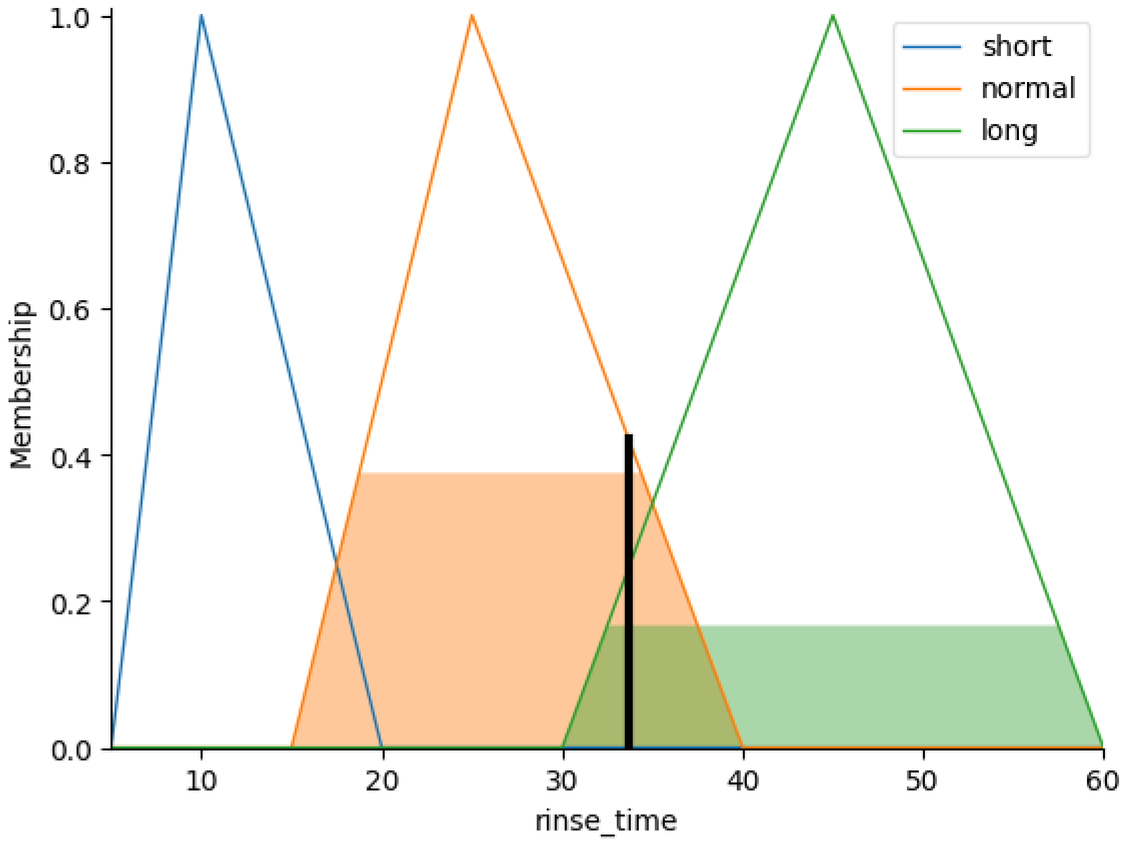
```
# Example 1: Use load size of 7kg and dirtiness of 80%
wash_sim.input['load_size'] = 7
wash_sim.input['dirtiness'] = 80

# Compute the result
wash_sim.compute()

print(f"Recommended rinse time: {wash_sim.output['rinse_time']:.2f} minutes")
rinse_time.view(sim=wash_sim)
```

```
Recommended rinse time: 45.00 minutes
```



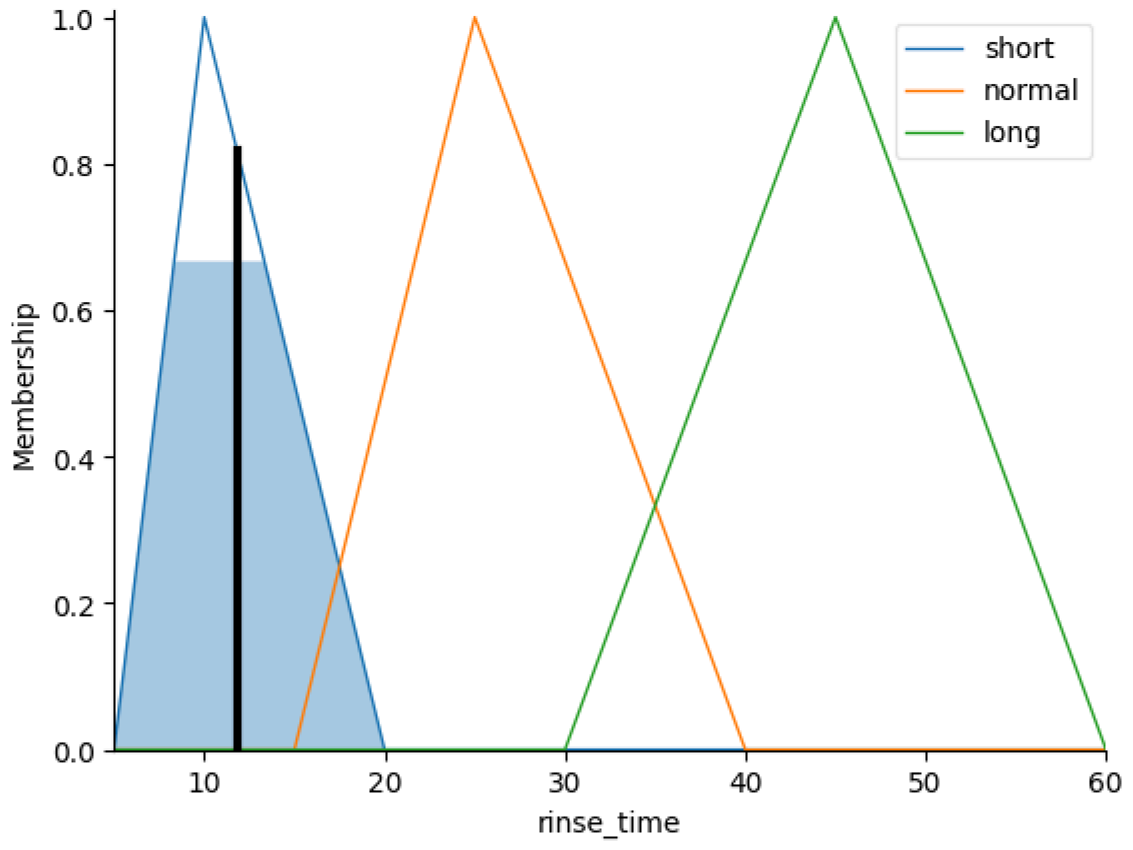
```python
# Example 2: Use load size of 10kg and dirtiness of 25%
wash_sim.input['load_size'] = 10
wash_sim.input['dirtiness'] = 25

# Compute the result
wash_sim.compute()

print(f"Recommended rinse time: {wash_sim.output['rinse_time']:.2f} minutes")
rinse_time.view(sim=wash_sim)
```

```
Recommended rinse time: 33.67 minutes
```



```python
# Example 3: Use load size of 4kg and dirtiness of 5%
wash_sim.input['load_size'] = 4
wash_sim.input['dirtiness'] = 5

# Compute the result
wash_sim.compute()

print(f"Recommended rinse time: {wash_sim.output['rinse_time']:.2f} minutes")
rinse_time.view(sim=wash_sim)
```

```
Recommended rinse time: 11.81 minutes
```



CONCLUSION:

In this experiment, we successfully designed a fuzzy control system for a washing machine using a fuzzy logic tool. The system was able to decide the appropriate washing time based on the load size and dirtiness, showing how fuzzy logic can make smart decisions even with uncertain or vague information. This shows the usefulness of fuzzy logic for solving real-life problems where exact rules are hard to define.

39_colab_08