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

AI&DS2 Experiment 08

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

Theory:

Introduction

Fuzzy Logic Control (FLC) is a control methodology that handles uncertainty, imprecision, and non-linearity in systems. Unlike traditional control methods that rely on exact mathematical models, FLC mimics human reasoning by processing linguistic rules and handling degrees of truth.

In classical Boolean logic, a statement is either True (1) or False (0), but in fuzzy logic, the truth value can take any value in the interval:

$$\mu_A(x) \in [0,1]$$

where:

- $\mu_A(x)$ = degree of membership of element x in fuzzy set A
- $\mu_A(x) = 0 \rightarrow$ completely not in the set
- $\mu_A(x) = 1 \rightarrow$ fully in the set
- $0 < \mu_A(x) < 1 \rightarrow$ partial membership

Architecture of a Fuzzy Logic Control System

A typical FLC consists of the following stages:

1. Fuzzifier
    - Converts crisp inputs x_{crisp} into fuzzy values using membership functions $\mu_A(x)$.
    - Example: Temperature = 25◦C may be:
      $\mu_{Warm}(x)=0.8$, $\mu_{Hot}(x)=0.2$

2. Fuzzy Knowledge Base
    - Contains:
        - Membership functions (defines fuzzy sets)
        - Fuzzy rule base (IF–THEN rules)

3. Fuzzy Rule Base
    - A collection of rules of the form:
      IF X is A AND Y is B THEN Z is C

      Example:
      IF temperature is Hot AND pressure is High THEN fan speed is Fast

4. Inference Engine
    - Determines the degree to which each rule is satisfied and combines the results.
    - Common types:
        - Mamdani Method (outputs fuzzy sets)

- Sugeno Method (outputs crisp values)

5. Defuzzifier
   - Converts the aggregated fuzzy output into a crisp value:
     - Centroid Method:
       $$z^* = \frac{\int z \cdot \mu(z)dz}{\int \mu(z)dz}$$
     - Other methods: Mean of Maxima (MOM), Center of Sums (COS)

Steps in Designing a Fuzzy Logic Control System

1. Identification of Variables
   - Inputs: $X1, X2, \ldots$
   - Outputs: $Y1, Y2, \ldots$
   - Define the universe of discourse for each variable.

2. Fuzzy Subset Configuration
   - Define linguistic terms for each variable.
     Example for temperature:
     {Cold, Warm, Hot}

3. Membership Functions
   - Choose shapes:
     - Triangular:
       $$\mu_A(x) = \max\left(0, \min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right)\right)$$
     - Trapezoidal, Gaussian, etc.

4. Fuzzy Rule Base Formation
   - Formulate IF–THEN rules from expert knowledge.

5. Fuzzification
   - Convert crisp inputs into fuzzy values using $\mu_A(x)$.

6. Inference
   - Apply fuzzy operators (AND → min, OR → max) to determine rule firing strength.

7. Defuzzification
   - Convert fuzzy output to crisp value ycrispy_{crisp}ycrisp.

Advantages of FLC

- Robustness against uncertainty
- No need for precise mathematical model
- Human-like reasoning
- Adaptability to changing conditions
- Tolerance to noisy or imprecise data
- Reduced development time

Applications of FLC

- Consumer Electronics: Washing machines, air conditioners, cameras
- Automotive: ABS, cruise control, engine management
- Industrial: Robotics, chemical process control
- Medical: Diagnosis, drug delivery systems
- Finance: Risk analysis, portfolio optimization
- Environmental: Pollution control, energy management

Code :

```
!pip install scikit-fuzzy numpy scipy networkx
```

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
```

```
# New Antecedent/Consequent objects hold universe variables and membership functions
# Health of the enemy (0-100)
health = ctrl.Antecedent(np.arange(0, 101, 1), 'health')
# Distance to player (0-100 units)
distance = ctrl.Antecedent(np.arange(0, 101, 1), 'distance')
# Aggressiveness of the enemy (0-100)
aggressiveness = ctrl.Consequent(np.arange(0, 101, 1), 'aggressiveness')
```

```
# Auto-membership function population is possible with .automf(3, 5, or 7)
# This will create 'poor', 'average', 'good' membership functions
health.automf(3)
distance.automf(3)
```

```
# Custom membership functions for aggressiveness
aggressiveness['low'] = fuzz.trimf(aggressiveness.universe, [0, 0, 50])
aggressiveness['moderate'] = fuzz.trimf(aggressiveness.universe, [25, 50, 75])
aggressiveness['high'] = fuzz.trimf(aggressiveness.universe, [50, 100, 100])
```

```
# Define fuzzy rules
# Using 'poor', 'average', 'good' for health and distance as per automf(3)
rule1 = ctrl.Rule(health['poor'] & distance['poor'], aggressiveness['high'])
rule2 = ctrl.Rule(health['poor'] & distance['average'], aggressiveness['high'])
rule3 = ctrl.Rule(health['poor'] & distance['good'], aggressiveness['moderate'])
```

```
rule4 = ctrl.Rule(health['average'] & distance['poor'], aggressiveness['high'])
```

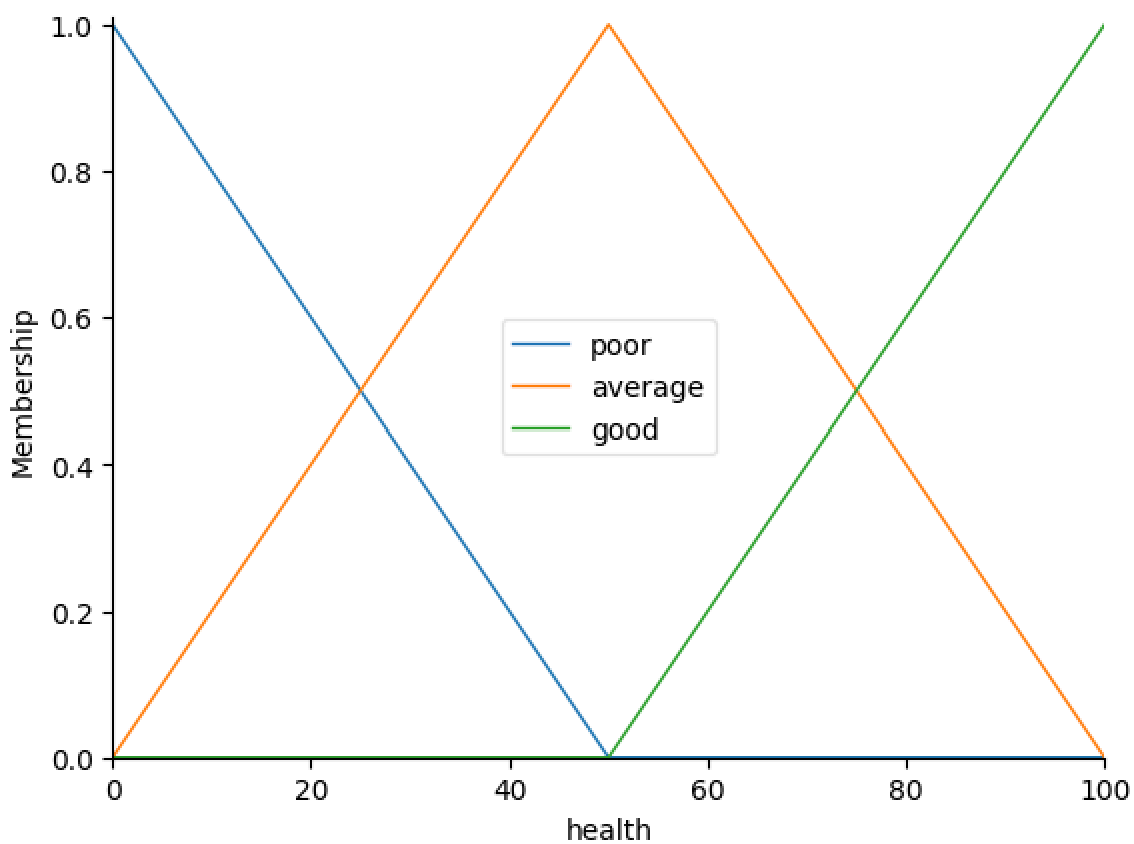
```
rule5 = ctrl.Rule(health['average'] & distance['average'], aggressiveness['moderate'])
rule6 = ctrl.Rule(health['average'] & distance['good'], aggressiveness['low'])
```

```
rule7 = ctrl.Rule(health['good'] & distance['poor'], aggressiveness['moderate'])
rule8 = ctrl.Rule(health['good'] & distance['average'], aggressiveness['low'])
rule9 = ctrl.Rule(health['good'] & distance['good'], aggressiveness['low'])
```

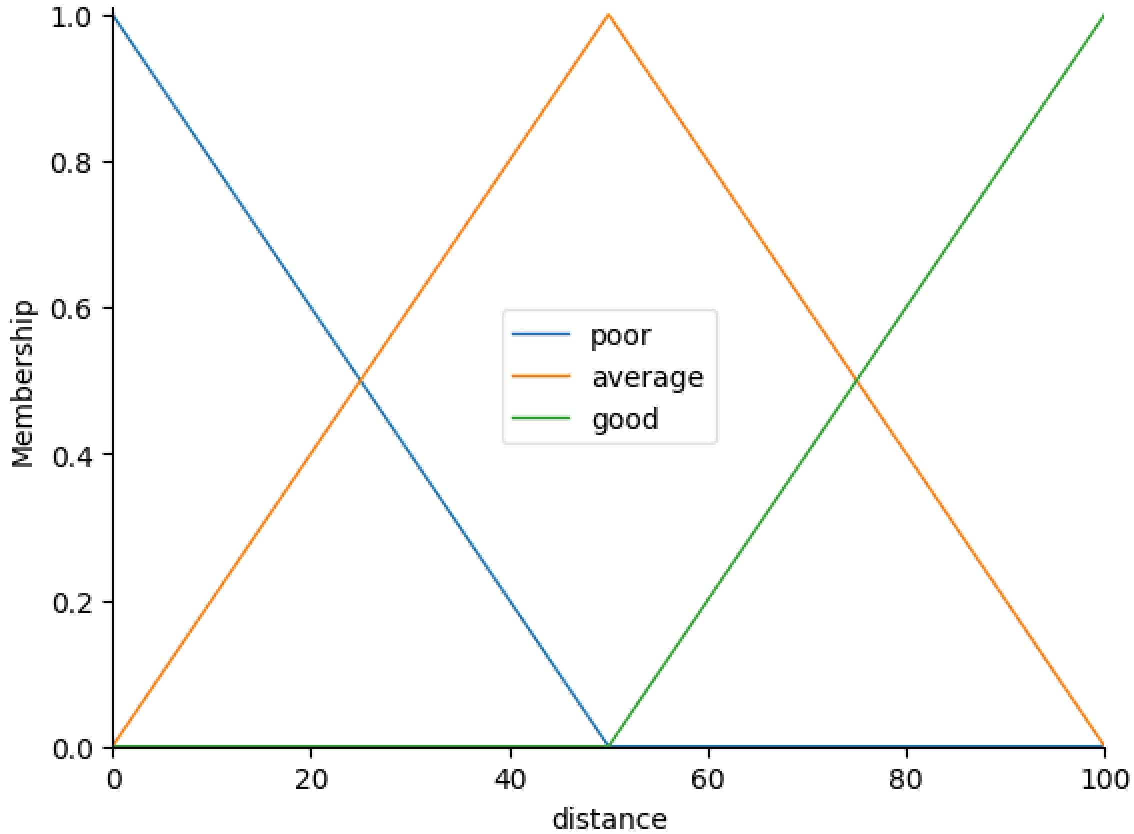
```
# Control System Creation and Simulation
aggressiveness_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9])
aggressiveness_sim = ctrl.ControlSystemSimulation(aggressiveness_ctrl)
```

```
# Function to get aggressiveness
def get_aggressiveness(current_health, current_distance):
  aggressiveness_sim.input['health'] = current_health
  aggressiveness_sim.input['distance'] = current_distance
  aggressiveness_sim.compute()
  return aggressiveness_sim.output['aggressiveness']
```

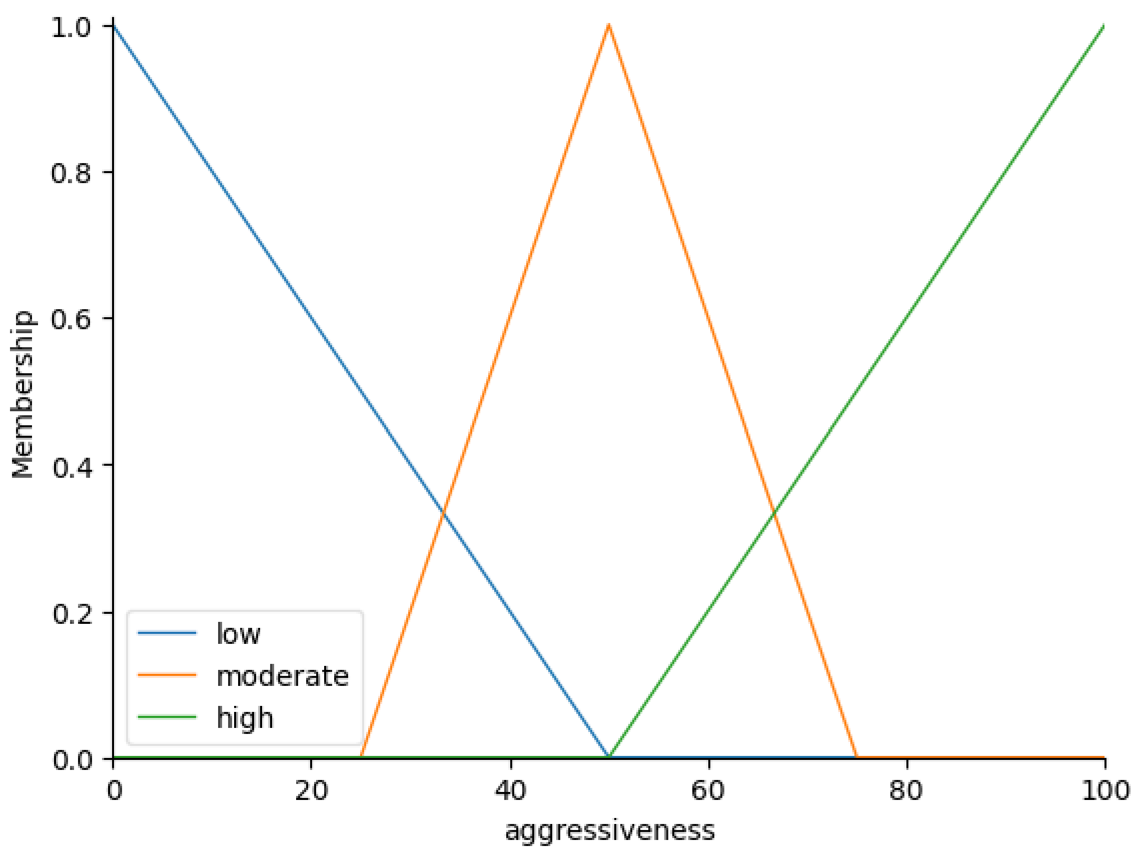
```
health.view()
```




```
distance.view()
```




```
aggressiveness.view()
```



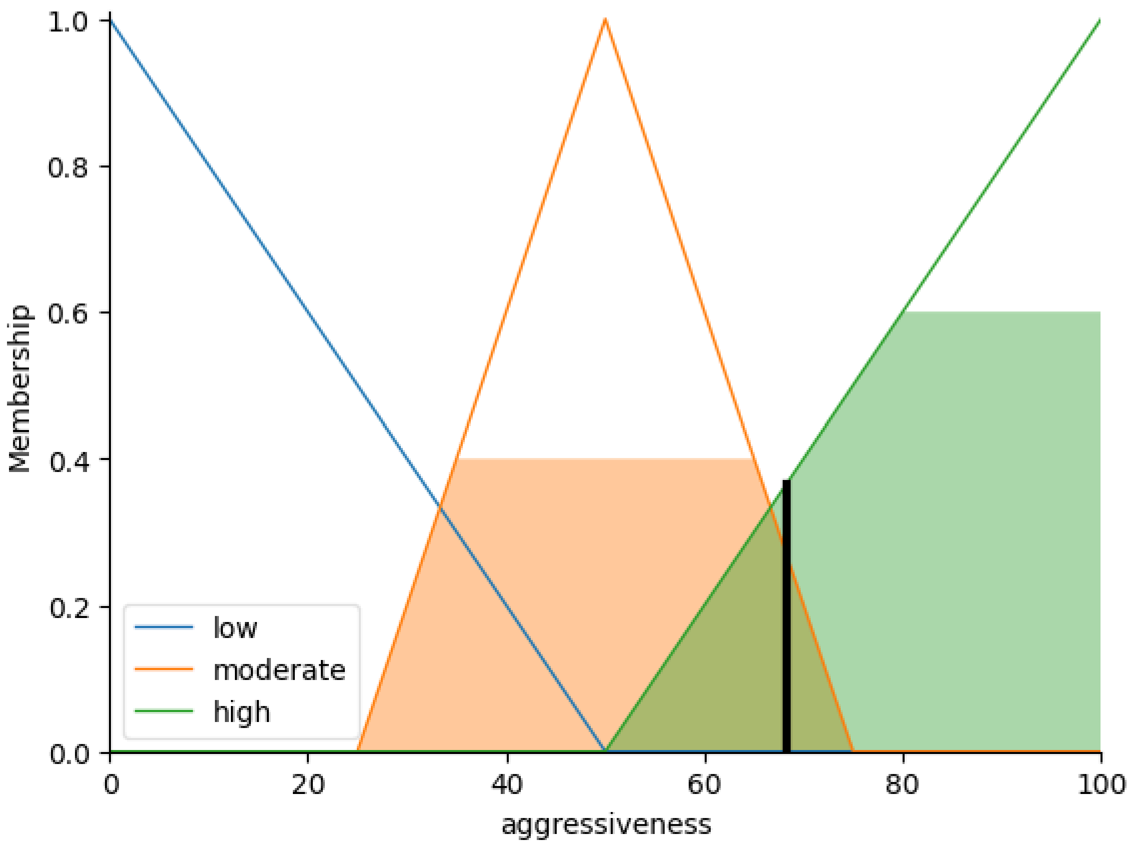

```
# Scenario 1: Low health, close distance (should be highly aggressive)
health_val1 = 30
```

```
distance_val1 = 20
aggro1 = get_aggressiveness(health_val1, distance_val1)
print(f"Aggressiveness for Health={health_val1}, Distance={distance_val1}: {aggro1:.2f}")

# You can also visualize the defuzzification process for a specific input
aggressiveness_sim.input['health'] = health_val1
aggressiveness_sim.input['distance'] = distance_val1
aggressiveness_sim.compute()
print(f"\nVisualizing for Health={health_val1}, Distance={distance_val1}:")
aggressiveness.view(sim=aggressiveness_sim)
plt.show()
```

Aggressiveness for Health=30, Distance=20: 68.34

Visualizing for Health=30, Distance=20:



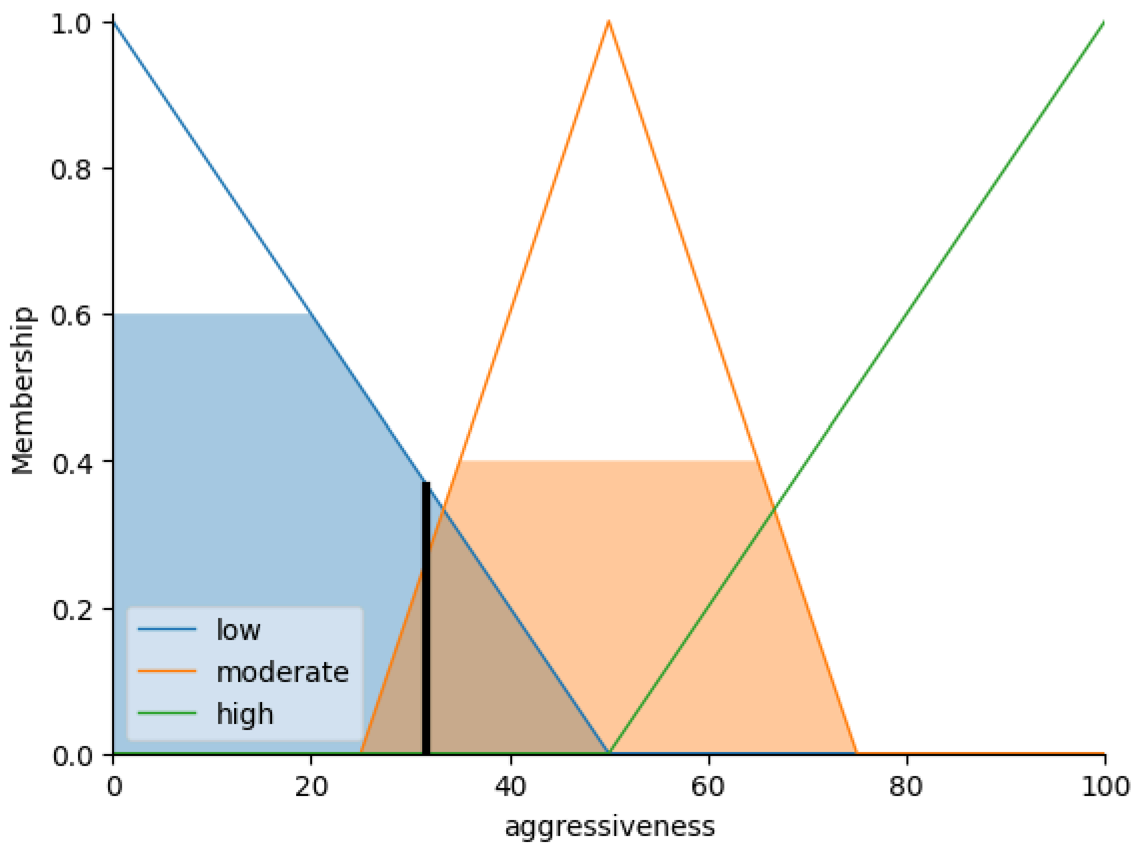

```
# Scenario 2: High health, far distance (should be low aggressiveness)
health_val2 = 80
distance_val2 = 70
aggro2 = get_aggressiveness(health_val2, distance_val2)
print(f"Aggressiveness for Health={health_val2}, Distance={distance_val2}: {aggro2:.2f}")

# You can also visualize the defuzzification process for a specific input
aggressiveness_sim.input['health'] = health_val2
```

```
aggressiveness_sim.input['distance'] = distance_val2
aggressiveness_sim.compute()
print(f"\nVisualizing for Health={health_val2}, Distance={distance_val2}:")
aggressiveness.view(sim=aggressiveness_sim)
plt.show()
```

Aggressiveness for Health=80, Distance=70: 31.66

Visualizing for Health=80, Distance=70:

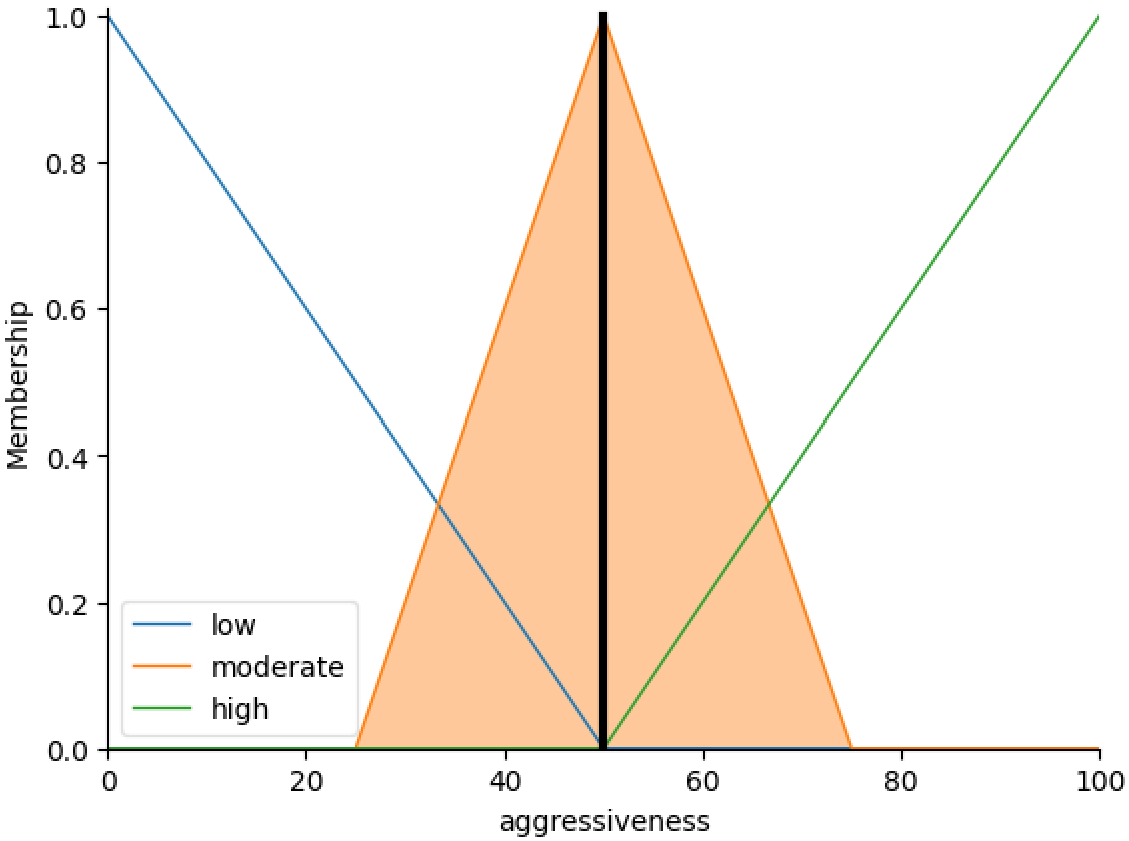



```
# Scenario 3: Medium health, medium distance (should be moderate aggressiveness)
health_val3 = 50
distance_val3 = 50
aggro3 = get_aggressiveness(health_val3, distance_val3)
print(f"Aggressiveness for Health={health_val3}, Distance={distance_val3}: {aggro3:.2f}")

# You can also visualize the defuzzification process for a specific input
aggressiveness_sim.input['health'] = health_val3
aggressiveness_sim.input['distance'] = distance_val3
aggressiveness_sim.compute()
print(f"\nVisualizing for Health={health_val3}, Distance={distance_val3}:")
aggressiveness.view(sim=aggressiveness_sim)
plt.show()
```

Aggressiveness for Health=50, Distance=50: 50.00

Visualizing for Health=50, Distance=50:



Conclusion:

In conclusion, Fuzzy Logic Control provides a powerful and flexible framework for designing intelligent control systems that can effectively manage complex and uncertain environments. Its ability to incorporate human knowledge and reason with imprecise information makes it a valuable tool in modern engineering and artificial intelligence applications.

49_Prac_08