Name: Atif Ansari **Roll no:** 04 **Class:** D20B

AIDS 2 EXP 8

Aim:

To design a Fuzzy Control System Using Fuzzy Tool

Theory:

Fuzzy logic control systems handle imprecise or uncertain information using linguistic variables rather than precise numerical values. The key components include:

- **Fuzzification:** Converting crisp inputs to fuzzy sets using membership functions.
- **Fuzzy Inference:** Applying if-then rules to determine system behavior.
- **Defuzzification:** Converting fuzzy outputs back to crisp values.

Applications include:

- Temperature control systems
- Automotive systems (ABS, transmission control)
- Consumer electronics (washing machines, air conditioners)
- Industrial automation

Theoretical Foundations:

- **Fuzzy Set Theory:** Extends classical set theory to handle partial membership.
- **Linguistic Variables:** Terms like "hot," "cold" instead of precise numbers.
- **Membership Functions:** Define how each point is mapped to a membership value (0-1).
- **Mamdani Inference:** A common fuzzy inference method using min-max operations.

Objective:

The primary objective is to design and implement a fuzzy logic control system for temperature regulation, demonstrating how fuzzy logic can handle imprecise inputs, model human-like reasoning, and provide smooth control outputs.

Libraries Used:

- **NumPy:** For numerical operations and defining input/output ranges.
- **skfuzzy:** A Python library for fuzzy logic operations, including defining fuzzy variables, membership functions, and control systems.
- **Matplotlib:** For visualizing membership functions and system response.

Steps:

1. **Define Fuzzy Variables:** Create antecedent (temperature) and consequent (cooling power) variables with appropriate ranges.
2. **Define Membership Functions:** Assign triangular membership functions to represent linguistic terms (e.g., cold, cool, moderate, warm, hot for temperature; low, medium, high for cooling power).
3. **Visualize Membership Functions:** Plot membership functions to verify their shapes and overlaps.
4. **Define Fuzzy Rules:** Create if-then rules to map input conditions to output actions.
5. **Create Control System:** Build and simulate the fuzzy control system using the defined rules.
6. **Test the System:** Input sample temperatures and compute corresponding cooling power outputs.
7. **Visualize Results:** Plot the system response to show how cooling power varies with temperature.

Code:

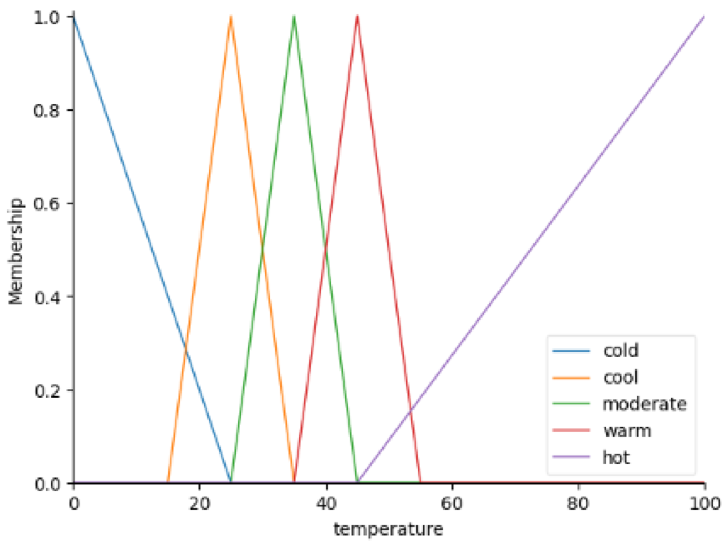
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Create fuzzy variables
temperature = ctrl.Antecedent(np.arange(0, 101, 1), 'temperature')
cooling_power = ctrl.Consequent(np.arange(0, 101, 1), 'cooling_power')

# Define membership functions for temperature
temperature['cold'] = fuzz.trimf(temperature.universe, [0, 0, 25])
```

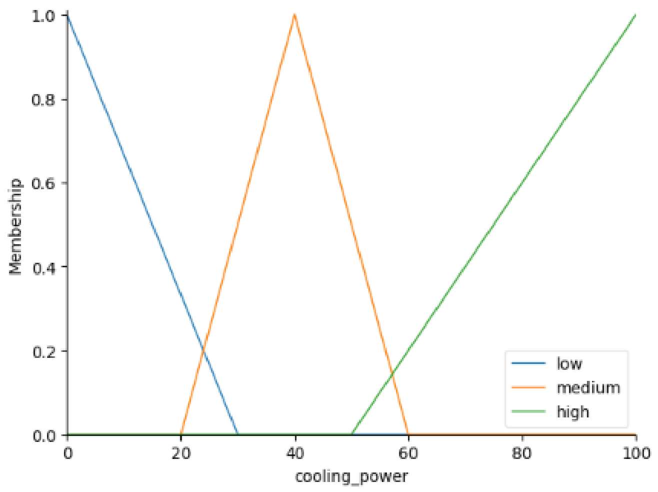
```
temperature['cool'] = fuzz.trimf(temperature.universe, [15, 25, 35])
temperature['moderate'] = fuzz.trimf(temperature.universe, [25, 35, 45])
temperature['warm'] = fuzz.trimf(temperature.universe, [35, 45, 55])
temperature['hot'] = fuzz.trimf(temperature.universe, [45, 100, 100])

# Define membership functions for cooling power
cooling_power['low'] = fuzz.trimf(cooling_power.universe, [0, 0, 30])
cooling_power['medium'] = fuzz.trimf(cooling_power.universe, [20, 40, 60])
cooling_power['high'] = fuzz.trimf(cooling_power.universe, [50, 100, 100])

# Visualize membership functions
temperature.view()
cooling_power.view()
plt.show()

# Define fuzzy rules
rule1 = ctrl.Rule(temperature['cold'], cooling_power['low'])
rule2 = ctrl.Rule(temperature['cool'], cooling_power['low'])
rule3 = ctrl.Rule(temperature['moderate'], cooling_power['medium'])
rule4 = ctrl.Rule(temperature['warm'], cooling_power['medium'])
rule5 = ctrl.Rule(temperature['hot'], cooling_power['high'])

# Create control system
cooling_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5])
cooling_sim = ctrl.ControlSystemSimulation(cooling_ctrl)

# Test with sample temperatures
test_temperatures = [15, 25, 35, 45, 60]
results = []

for temp in test_temperatures:
    cooling_sim.input['temperature'] = temp
    cooling_sim.compute()
    results.append(cooling_sim.output['cooling_power'])
            print(f"Temperature:    {temp}°C    ->    Cooling    Power:
{cooling_sim.output['cooling_power']:.2f}%")

# Visualize results
plt.figure(figsize=(10, 6))
plt.plot(test_temperatures, results, 'bo-')
plt.xlabel('Temperature (°C)')
plt.ylabel('Cooling Power (%)')
plt.title('Fuzzy Control System Response')
plt.grid(True)
plt.show()
```

Code Explanation:

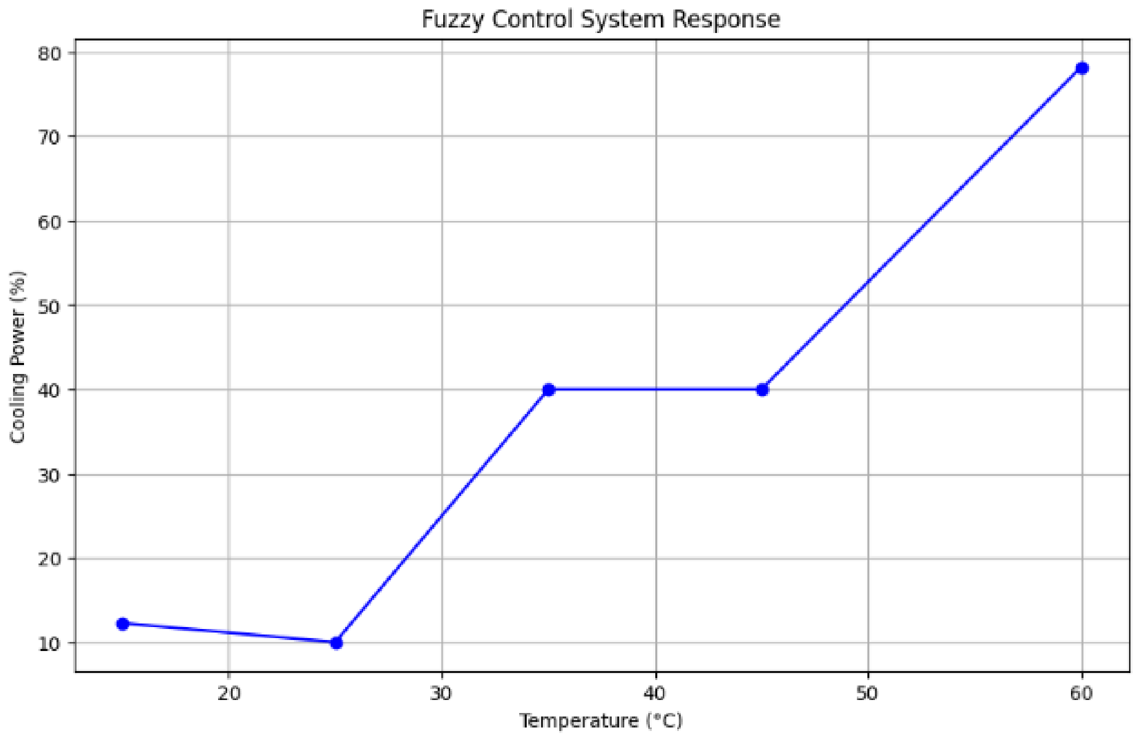
- **Fuzzy Variables:** The temperature (input) and cooling_power (output) variables are defined with a range of 0 to 100.
- **Membership Functions:** Triangular membership functions (trimf) are used to define linguistic terms for both variables, ensuring smooth transitions between states.
- **Fuzzy Rules:** Five rules map temperature states (cold, cool, moderate, warm, hot) to cooling power levels (low, medium, high).
- **Control System:** The ControlSystem combines the rules, and ControlSystemSimulation computes the output for given inputs.
- **Testing and Visualization:** Sample temperatures are tested, and the results are plotted to show the relationship between input temperature and output cooling power.

Output:





```
Temperature: 15°C -> Cooling Power: 12.25%
Temperature: 25°C -> Cooling Power: 10.00%
Temperature: 35°C -> Cooling Power: 40.00%
Temperature: 45°C -> Cooling Power: 40.00%
Temperature: 60°C -> Cooling Power: 78.23%
```



The code generates two plots:

1. **Membership Functions:** Plots showing triangular membership functions for temperature (cold, cool, moderate, warm, hot) and cooling power (low, medium, high), with appropriate overlaps.
2. **System Response:** A line plot with blue points connected by lines, showing how cooling power increases with temperature, demonstrating smooth transitions.

Conclusion:

In this experiment, we successfully designed and implemented a fuzzy logic control system for temperature regulation using the skfuzzy library. Key findings include:

- **Handling Uncertainty:** Fuzzy systems effectively manage imprecise inputs and provide gradual transitions between states.
- **Natural Modeling:** Linguistic rules (e.g., "if hot, then high cooling") mimic human reasoning, making the system intuitive.
- **Smooth Control:** The output cooling power changes gradually, avoiding abrupt transitions typical of on/off systems.

Fuzzy control systems are particularly valuable in applications where:

- Precise mathematical models are unavailable.
- Human expertise is available in linguistic form.
- Smooth control response is preferred over binary switching.

Future Work:

- Explore adaptive fuzzy systems that learn from data.
- Develop hybrid neuro-fuzzy systems combining neural networks with fuzzy logic.
- Extend to multi-input systems considering factors like humidity or occupancy.
- Implement real-time control with hardware interfaces.

This experiment demonstrates the power of fuzzy logic in building robust, intelligent control systems capable of handling real-world complexity and uncertainty effectively.