AIDS LAB EXPERIMENT 08

Name : Deepa Behrani
Class : D20B
RollNo : 05

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

Theory :

A **fuzzy control system** is a control method based on **fuzzy logic**, which allows handling uncertain or imprecise information by using degrees of truth rather than binary true/false values. Unlike traditional controllers, fuzzy controllers use **fuzzy sets** and **linguistic rules** to model complex or nonlinear systems where exact mathematical models are difficult to obtain.

The design process includes:

- **Fuzzification:** Converting crisp inputs into fuzzy values.
- **Rule Evaluation:** Applying a set of IF-THEN rules to infer fuzzy outputs.
- **Defuzzification:** Transforming fuzzy outputs into crisp control signals.

Using a **fuzzy tool/library** simplifies this process by providing functions to create membership functions, define rules, and simulate the controller, enabling efficient design and testing of fuzzy control systems.

Designing a Fuzzy Controller
Steps include:

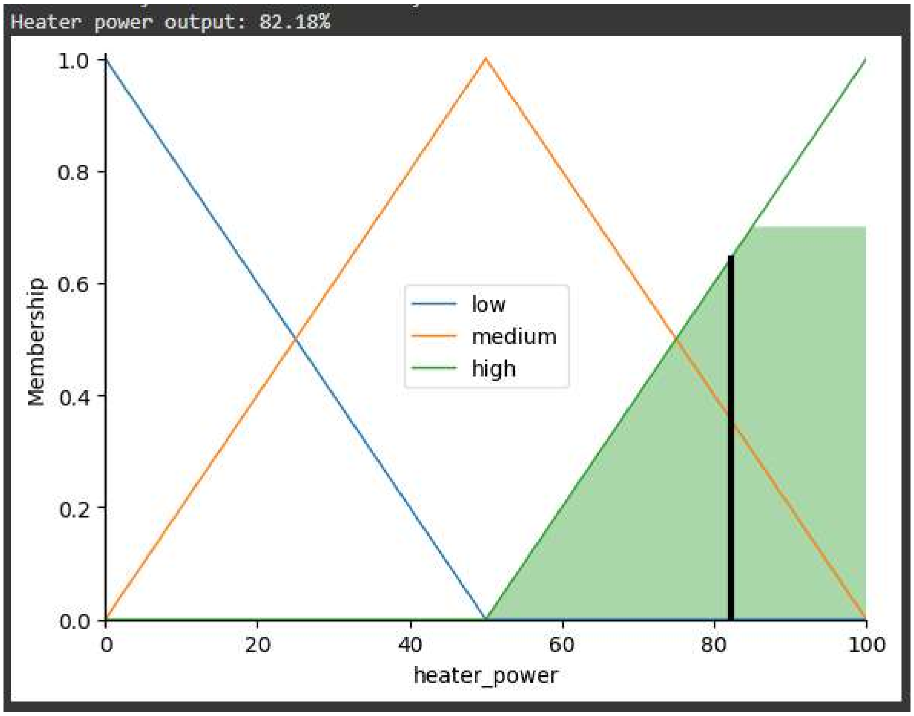
- **Selection of input and output variables.**
- **Defining membership functions** for inputs and outputs.
- **Constructing fuzzy rules** based on expert knowledge or system behavior.
- **Implementation and simulation** using a fuzzy tool/library.
- **Tuning and validation** to ensure desired control performance

Advantages of Fuzzy Control Systems
- Handles nonlinear and complex systems effectively
- Robust to noise and uncertainties.
- Incorporates human expert knowledge through linguistic rules.
- No precise mathematical model required.

Code :
```
# Install scikit-fuzzy
!pip install scikit-fuzzy

import numpy as np
import skfuzzy as fuzz
```

```
from skfuzzy import control as ctrl

# Define fuzzy variables

# Input: Temperature difference (error) - from -10 to 10 degrees
error = ctrl.Antecedent(np.arange(-10, 11, 1), 'error')

# Output: Heater power - from 0 to 100 percent
heater_power = ctrl.Consequent(np.arange(0, 101, 1), 'heater_power')

# Define membership functions for input 'error'
error['cold'] = fuzz.trimf(error.universe, [-10, -10, 0])
error['comfortable'] = fuzz.trimf(error.universe, [-5, 0, 5])
error['hot'] = fuzz.trimf(error.universe, [0, 10, 10])

# Define membership functions for output 'heater_power'
heater_power['low'] = fuzz.trimf(heater_power.universe, [0, 0, 50])
heater_power['medium'] = fuzz.trimf(heater_power.universe, [0, 50, 100])
heater_power['high'] = fuzz.trimf(heater_power.universe, [50, 100, 100])

# Define fuzzy rules
rule1 = ctrl.Rule(error['cold'], heater_power['high'])
rule2 = ctrl.Rule(error['comfortable'], heater_power['medium'])
rule3 = ctrl.Rule(error['hot'], heater_power['low'])

# Create control system and simulation
heater_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
heater_sim = ctrl.ControlSystemSimulation(heater_ctrl)

# Input value (e.g. error = -7 means too cold)
heater_sim.input['error'] = -7

# Compute the fuzzy output
heater_sim.compute()

print(f"Heater power output: {heater_sim.output['heater_power']:.2f}%")

# Optional: visualize the output membership
heater_power.view(sim=heater_sim)
```

OUTPUT :



Conclusion : This example shows how to create a basic fuzzy control system using Python's `scikit-fuzzy` library. By defining fuzzy variables and rules, we controlled heater power based on temperature error. Fuzzy logic provides smooth and flexible decision-making for control tasks with uncertainty.