Experiment 08 : Design a fuzzy controller.

Learning Objective : Design a fuzzy controller.

Tools : Python

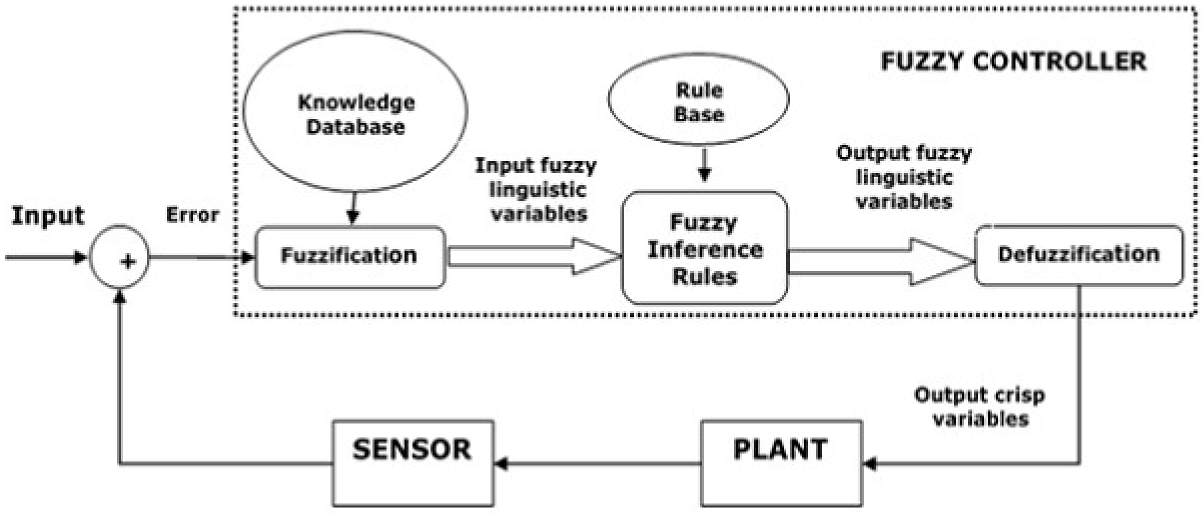
Theory :

A fuzzy controller is a type of control system that uses fuzzy logic to make decisions based on imprecise or uncertain data. Fuzzy logic is a mathematical framework for dealing with uncertain or ambiguous data, and is particularly useful when precise measurements or numerical values are to be obtained.

A fuzzy controller consists of three main components: a fuzzifier, a rules engine, and a defuzzifier. The fuzzifier converts input data into fuzzy sets, which are sets of values that represent a degree of membership in a particular category. The rules engine uses a set of fuzzy if-then rules to make decisions based on the input data. Finally, the defuzzifier converts the output of the rules engine back into a numerical value that can be used to control a system.

One example of a fuzzy controller in action is in controlling the speed of a car. The input variables might include the speed of the car, the distance to the car in front, and the road conditions. These variables can be fuzzified into fuzzy sets, which represent a degree of membership in categories such as "safe", "moderate", or "dangerous". The rules engine uses a set of fuzzy if-then rules to determine the appropriate speed for the car based on the input variables. Finally, the defuzzifier converts the output of the rule's engine into a numerical value that is used to control the speed of the car.

E.g : Washing Machine Controller



Designing a fuzzy controller involves several steps, including the following:

Identify the input and output variables: The first step in designing a fuzzy controller is to identify the input and output variables for the system you want to control. These might include physical quantities such as temperature, pressure, or speed, or more abstract variables such as customer satisfaction or risk.

Fuzzify the input variables: Once you have identified the input variables, you need to fuzzify them into fuzzy sets. This involves defining the membership functions that describe how each variable is related to different categories or ranges of values.

Define the rules: The next step is to define a set of rules that describe how the output variable should change based on the input variables. These rules are typically expressed in the form of "if X is A and Y is B, then Z is C," where X, Y, and Z are variables and A, B, and C are fuzzy sets.

Apply inference methods: Once you have defined the rules, you need to apply inference methods to determine the appropriate output value for the system based on the input variables. There are several inference methods to choose from, including the maximum rule, minimum rule, and product rule.

Defuzzify the output variable: The final step in designing a fuzzy controller is to defuzzify the output variable to obtain a crisp value that can be used to control the system. This involves applying a defuzzification method, such as the centroid method or the maximum membership method, to determine the appropriate output value based on the fuzzy sets and the inference method.

Implementation :

```
[1]: !pip install -U scikit-fuzzy
```

```
Requirement already satisfied: scikit-fuzzy in /usr/local/lib/python3.10/dist-
packages (0.4.2)
Requirement already satisfied: numpy>=1.6.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-fuzzy) (1.25.2)
Requirement already satisfied: scipy>=0.9.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-fuzzy) (1.11.4)
Requirement already satisfied: networkx>=1.9.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-fuzzy) (3.2.1)
```

```
[2]: import numpy as np
     import skfuzzy as fuzz
     from skfuzzy import control as ctrl
```

```
[3]: # Define the fuzzy variables
     distance_to_obstacle = ctrl.Antecedent(np.arange(0, 11, 1),␣
      ↪'distance_to_obstacle')
     speed = ctrl.Antecedent(np.arange(0, 11, 1), 'speed')
     turn_direction = ctrl.Consequent(np.arange(-180, 181, 1), 'turn_direction')
```

[4]:
```
# Define custom membership functions for distance_to_obstacle
distance_to_obstacle['close'] = fuzz.trimf(distance_to_obstacle.universe, [0,␣
 ↪0, 5])
distance_to_obstacle['medium'] = fuzz.trimf(distance_to_obstacle.universe, [0,␣
 ↪5, 10])
distance_to_obstacle['far'] = fuzz.trimf(distance_to_obstacle.universe, [5, 10,␣
 ↪10])
```

[5]:
```
# Define custom membership functions for speed
speed['slow'] = fuzz.trimf(speed.universe, [0, 0, 5])
speed['medium'] = fuzz.trimf(speed.universe, [0, 5, 10])
speed['fast'] = fuzz.trimf(speed.universe, [5, 10, 10])
```

[6]:
```
# Custom membership functions for the turn_direction
turn_direction['left'] = fuzz.trimf(turn_direction.universe, [-180, -90, 0])
turn_direction['straight'] = fuzz.trimf(turn_direction.universe, [-90, 0, 90])
turn_direction['right'] = fuzz.trimf(turn_direction.universe, [0, 90, 180])
```

[7]:
```
# Define the fuzzy rules
rule1 = ctrl.Rule(distance_to_obstacle['close'] & speed['fast'],␣
 ↪turn_direction['left'])
rule2 = ctrl.Rule(distance_to_obstacle['medium'] & speed['medium'],␣
 ↪turn_direction['straight'])
rule3 = ctrl.Rule(distance_to_obstacle['far'] & speed['slow'],␣
 ↪turn_direction['right'])
```

[8]:
```
# Create the control system
navigation_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])

# Create a simulation of the control system
navigation = ctrl.ControlSystemSimulation(navigation_ctrl)
```

[9]:
```
# Pass inputs to the ControlSystem
navigation.input['distance_to_obstacle'] = 6.5
navigation.input['speed'] = 9.8
```

[10]:
```
# Compute the output
navigation.compute()

# Determine the category with the highest degree of membership
max_membership = np.argmax(navigation.output['turn_direction'])
```

[11]:
```
# Map the index to the corresponding category
categories = ['left', 'straight', 'right']
turn_direction_output = categories[max_membership]

# Print the output
print(turn_direction_output)
```

```
left
```

Result and Discussion :

Learning Outcomes : Students should have the ability to

LO 8.1: Ability to understand the application of fuzzy controllers and how it works.
LO 8.2: Ability to implement the washing machine or any controller.

Course Outcomes :

CO : Understand and design fuzzy controller.

Conclusion :

____

____

Viva Questions :

Q1. What is a fuzzy controller, and how does it differ from a traditional control system?
Q2. What is the role of the rules engine in a fuzzy controller, and how are fuzzy rules typically defined?

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	Total
Marks Obtained				