

```

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

# Define the fuzzy variables
temperature = ctrl.Antecedent(np.arange(0, 41, 1), 'temperature') # Current temperature in °C
desired_temp = ctrl.Antecedent(np.arange(0, 41, 1), 'desired_temp') # Desired temperature in °C
ac_power = ctrl.Consequent(np.arange(0, 101, 1), 'ac_power') # AC power output in %

# Fuzzy sets for temperature
temperature['cold'] = fuzz.trapmf(temperature.universe, [0, 0, 18, 22])
temperature['comfortable'] = fuzz.trimf(temperature.universe, [20, 24, 28])
temperature['hot'] = fuzz.trapmf(temperature.universe, [25, 30, 40, 40])

# Fuzzy sets for desired temperature
desired_temp['cold'] = fuzz.trapmf(desired_temp.universe, [0, 0, 18, 22])
desired_temp['comfortable'] = fuzz.trimf(desired_temp.universe, [20, 24, 28])
desired_temp['hot'] = fuzz.trapmf(desired_temp.universe, [25, 30, 40, 40])

# Fuzzy sets for AC power
ac_power['off'] = fuzz.trimf(ac_power.universe, [0, 0, 0])
ac_power['low'] = fuzz.trimf(ac_power.universe, [0, 25, 50])
ac_power['high'] = fuzz.trimf(ac_power.universe, [50, 100, 100])

# Define the fuzzy rules
rule1 = ctrl.Rule(temperature['hot'] & desired_temp['cold'], ac_power['high'])
rule2 = ctrl.Rule(temperature['hot'] & desired_temp['comfortable'], ac_power['high'])
rule3 = ctrl.Rule(temperature['comfortable'] & desired_temp['cold'], ac_power['low'])
rule4 = ctrl.Rule(temperature['comfortable'] & desired_temp['hot'], ac_power['low'])
rule5 = ctrl.Rule(temperature['cold'] & desired_temp['hot'], ac_power['off'])


# Create the control system
ac_control = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5])
ac_simulation = ctrl.ControlSystemSimulation(ac_control)

# Example usage
def control_ac(current_temp, desired_temp_value):
    ac_simulation.input['temperature'] = current_temp
    ac_simulation.input['desired_temp'] = desired_temp_value
    ac_simulation.compute()
    return ac_simulation.output['ac_power']

if __name__ == "__main__":
    current_temp = 30 # Current temperature in °C
    desired_temp_value = 22 # Desired temperature in °C

    ac_power_output = control_ac(current_temp, desired_temp_value)
    print(f"AC Power Output: {ac_power_output:.2f}%")

```

 AC Power Output: 80.56%

```

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

# Define fuzzy variables
current_angle = ctrl.Antecedent(np.arange(0, 181, 1), 'current_angle') # Angle in degrees
desired_angle = ctrl.Antecedent(np.arange(0, 181, 1), 'desired_angle') # Angle in degrees
motor_power = ctrl.Consequent(np.arange(0, 101, 1), 'motor_power') # Motor power in percentage

# Fuzzy sets for current angle
current_angle['too_left'] = fuzz.trapmf(current_angle.universe, [0, 0, 30, 60])
current_angle['straight'] = fuzz.trimf(current_angle.universe, [30, 90, 150])
current_angle['too_right'] = fuzz.trapmf(current_angle.universe, [120, 150, 180, 180])

# Fuzzy sets for desired angle
desired_angle['left'] = fuzz.trapmf(desired_angle.universe, [0, 0, 30, 60])
desired_angle['center'] = fuzz.trimf(desired_angle.universe, [30, 90, 150])
desired_angle['right'] = fuzz.trapmf(desired_angle.universe, [120, 150, 180, 180])

# Fuzzy sets for motor power
motor_power['low'] = fuzz.trimf(motor_power.universe, [0, 25, 50])
motor_power['medium'] = fuzz.trimf(motor_power.universe, [25, 50, 75])

```

```

motor_power['high'] = fuzz.trimf(motor_power.universe, [50, 75, 100])

# Define the fuzzy rules
rule1 = ctrl.Rule(current_angle['too_left'] & desired_angle['right'], motor_power['high'])
rule2 = ctrl.Rule(current_angle['too_left'] & desired_angle['center'], motor_power['medium'])
rule3 = ctrl.Rule(current_angle['straight'] & desired_angle['right'], motor_power['low'])
rule4 = ctrl.Rule(current_angle['straight'] & desired_angle['left'], motor_power['low'])
rule5 = ctrl.Rule(current_angle['too_right'] & desired_angle['left'], motor_power['high'])
rule6 = ctrl.Rule(current_angle['too_right'] & desired_angle['center'], motor_power['medium'])
rule7 = ctrl.Rule(current_angle['straight'] & desired_angle['center'], motor_power['low'])

# Create the control system
robotic_arm_control = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7])
robotic_arm_simulation = ctrl.ControlSystemSimulation(robotic_arm_control)

# Function to control the robotic arm
def control_robotic_arm(current, desired):
    robotic_arm_simulation.input['current_angle'] = current
    robotic_arm_simulation.input['desired_angle'] = desired
    robotic_arm_simulation.compute()
    return robotic_arm_simulation.output['motor_power']

# Example usage
if __name__ == "__main__":
    # Simulate a few scenarios
    current_angles = [10, 90, 150, 80]
    desired_angles = [60, 90, 120, 30]

    for current, desired in zip(current_angles, desired_angles):
        power_output = control_robotic_arm(current, desired)
        print(f"Current Angle: {current}°, Desired Angle: {desired}° -> Motor Power: {power_output:.2f}%")

➡ Current Angle: 10°, Desired Angle: 60° -> Motor Power: 50.00%
Current Angle: 90°, Desired Angle: 90° -> Motor Power: 25.00%
Current Angle: 150°, Desired Angle: 120° -> Motor Power: 50.00%
Current Angle: 80°, Desired Angle: 30° -> Motor Power: 25.00%

```