Code report

ANN 04 assignment – Question 4

Hesam Koucheki

4037236987

```
pip install scikit-fuzzy
```

- **`pip`** is the Python package installer, which allows you to download and install external libraries.
- **`scikit-fuzzy`** is the library that provides tools for working with fuzzy logic systems, including membership functions, inference systems, and defuzzification.

Once the installation is complete, you'll be able to import and use `scikit-fuzzy` in your Python scripts to create and manipulate fuzzy logic systems.

```
# Define the universe of discourse for each variable
temperature_range = np.linspace(-40, 40, 1000)  # Temperature: -40 to 40
degrees
humidity_range = np.linspace(10, 100, 1000)     # Humidity: 10% to 100%
speed_range = np.linspace(0, 20, 1000)          # Speed: 0 to 20 (motor
speed)
```

In this part of the code, you're defining the **universe of discourse** for each variable. The universe of discourse is the range over which each variable is defined, and it represents all possible input values that the fuzzy system can handle.

Let's break down each line:

```
python
Copy code
temperature_range = np.linspace(-40, 40, 1000)
```

- **Purpose**: This line defines the range of possible temperature values for the system.
- **`np.linspace(-40, 40, 1000)`**: This creates an array of 1000 evenly spaced values between **-40** and **40**. These values represent the possible temperatures that the system can handle. The range **-40 to 40** covers the entire range of temperatures we're considering for the system (from very cold to very hot).
- **Why 1000 values?**: This ensures the system has a fine resolution of 1000 points between -40 and 40, making the fuzzification process more accurate and continuous.

```
python
Copy code
humidity_range = np.linspace(10, 100, 1000)
```

- **Purpose**: This line defines the range of possible humidity values.

- **`np.linspace(10, 100, 1000)`**: This creates an array of 1000 evenly spaced values between **10%** and **100%**, representing the possible humidity levels in the system.
- **Why 1000 values?**: Similar to temperature, using 1000 values allows for a more precise fuzzification process.

```
python
Copy code
speed_range = np.linspace(0, 20, 1000)
```

- **Purpose**: This line defines the range of possible motor speed values.
- **`np.linspace(0, 20, 1000)`**: This creates an array of 1000 evenly spaced values between **0** and **20**, representing motor speeds ranging from no speed (0) to full speed (20).
- **Why 1000 values?**: This provides a high resolution of possible motor speeds for more accurate fuzzy logic operations and smoother transitions during defuzzification.

Summary:

- **`temperature_range`**, **`humidity_range`**, and **`speed_range`** define the continuous possible values (the universe of discourse) for the temperature, humidity, and motor speed variables, respectively.
- **`np.linspace(start, end, num_points)`** is used to generate these ranges, where `start` is the beginning of the range, `end` is the end of the range, and `num_points` is the number of values you want to generate in that range.

These ranges are used throughout the fuzzy logic system to fuzzify inputs, apply fuzzy rules, and perform defuzzification. The more points you use, the smoother and more precise the results can be in a continuous system like this one.

```
# Fuzzification of Temperature (T)
T_L = fuzz.trimf(temperature_range, [-40, -20, 0])   # Low temperature
T_M = fuzz.trimf(temperature_range, [0, 20, 40])     # Moderate temperature
T_H = fuzz.trimf(temperature_range, [20, 30, 40])    # High temperature
```

In this part of the code, we are **fuzzifying the temperature variable** by defining **membership functions** for the different fuzzy sets: Low, Moderate, and High temperature. Fuzzification is the process of converting crisp input values (like a specific temperature) into fuzzy values that represent degrees of membership in fuzzy sets.

Fuzzification of Temperature

The `fuzz.trimf()` function is used to create **triangular membership functions**. A triangular membership function is a simple and effective way to represent fuzzy sets. It has a peak at a particular value and decreases linearly on both sides.

```
python
Copy code
T_L = fuzz.trimf(temperature_range, [-40, -20, 0])  # Low temperature
```

- **Purpose**: This defines the **Low Temperature** fuzzy set.
- **`fuzz.trimf(temperature_range, [-40, -20, 0])`**: This creates a triangular membership function on the `temperature_range` (from -40 to 40). The parameters `[-40, -20, 0]` define the points of the triangle:
  - The **left point** at -40 indicates that the membership degree for Low Temperature is 1 (full membership) at -40.
  - The **middle point** at -20 represents the peak of the triangle, where the membership value starts to decrease.
  - The **right point** at 0 indicates that the membership degree for Low Temperature reaches 0 (no membership) at 0°C.

So, the temperature values between -40°C and 0°C will have varying degrees of membership to the "Low Temperature" fuzzy set.

```
python
Copy code
T_M = fuzz.trimf(temperature_range, [0, 20, 40])    # Moderate temperature
```

- **Purpose**: This defines the **Moderate Temperature** fuzzy set.
- **`fuzz.trimf(temperature_range, [0, 20, 40])`**: This creates a triangular membership function with the following points:
  - The **left point** at 0 indicates full membership to "Moderate Temperature" at 0°C.
  - The **middle point** at 20 represents the peak of the triangle, where the membership degree is highest.
  - The **right point** at 40 indicates that the membership degree for "Moderate Temperature" is 0 at 40°C.

So, temperatures around 20°C will have the highest membership to "Moderate Temperature," while the membership decreases as the temperature moves away from 20°C.

```
python
Copy code
T_H = fuzz.trimf(temperature_range, [20, 30, 40])   # High temperature
```

- **Purpose**: This defines the **High Temperature** fuzzy set.
- **`fuzz.trimf(temperature_range, [20, 30, 40])`**: This creates a triangular membership function with the following points:
  - The **left point** at 20 indicates full membership to "High Temperature" at 20°C.
  - The **middle point** at 30 is the peak of the triangle.

- The **right point** at 40 indicates that the membership degree for "High Temperature" is 0 at 40°C.

So, temperatures around 40°C will have the highest membership to "High Temperature."

Summary of the Triangular Membership Functions for Temperature:

1. **Low Temperature (T_L)**:
   - Full membership at -40°C (degree = 1).
   - The membership decreases linearly until it reaches 0 at 0°C.
2. **Moderate Temperature (T_M)**:
   - Full membership at 0°C and 40°C, with the highest membership at 20°C.
3. **High Temperature (T_H)**:
   - Full membership at 20°C and 40°C, with the highest membership at 30°C.

Purpose of Fuzzification:

Fuzzification helps represent real-world values in a way that captures **uncertainty** and **imprecision**. For example, if the temperature is 18°C:

- The system can calculate partial membership values for each fuzzy set (Low, Moderate, High).
- The degree of membership for 18°C in **Moderate** would be high, but it would still have a small membership in **Low** and **High**, indicating that it's not fully one or the other.

In practice, **fuzzification** makes it easier to model complex systems like HVAC (Heating, Ventilation, and Air Conditioning), where temperature changes aren't always sharp and distinct, and decisions need to be made gradually.

```
# Fuzzification of Humidity (H)
H_L = fuzz.trimf(humidity_range, [10, 20, 40])   # Low humidity
H_M = fuzz.trimf(humidity_range, [40, 55, 70])   # Moderate humidity
H_H = fuzz.trimf(humidity_range, [70, 85, 100])  # High humidity
```

In this part of the code, we are **fuzzifying the humidity variable** by defining **membership functions** for the fuzzy sets: **Low**, **Moderate**, and **High** humidity. Similar to temperature, we use **triangular membership functions** to represent the degrees of membership for the humidity values.

Fuzzification of Humidity

We again use `fuzz.trimf()` to define the triangular membership functions for the three humidity levels. Each of the fuzzy sets (Low, Moderate, High) represents a range of humidity values, where the degree of membership decreases as the humidity value moves away from the central point of the triangle.

```
python
Copy code
H_L = fuzz.trimf(humidity_range, [10, 20, 40])   # Low humidity
```

- **Purpose**: This defines the **Low Humidity** fuzzy set.
- **`fuzz.trimf(humidity_range, [10, 20, 40])`**: This creates a triangular membership function on the `humidity_range` (from 10% to 100%). The points `[10, 20, 40]` define the shape of the triangle:
    - The **left point** at 10% represents full membership (degree = 1) to "Low Humidity."
    - The **middle point** at 20% indicates the peak of the triangle, where the membership degree starts to decrease.
    - The **right point** at 40% indicates that the membership degree for "Low Humidity" is 0, meaning 40% humidity and above no longer belong to the "Low" category.

So, humidity values between 10% and 40% will have varying degrees of membership to the "Low Humidity" fuzzy set.

```
python
Copy code
H_M = fuzz.trimf(humidity_range, [40, 55, 70])   # Moderate humidity
```

- **Purpose**: This defines the **Moderate Humidity** fuzzy set.
- **`fuzz.trimf(humidity_range, [40, 55, 70])`**: This creates a triangular membership function on the `humidity_range` with the points `[40, 55, 70]`:
    - The **left point** at 40% indicates full membership to "Moderate Humidity."
    - The **middle point** at 55% represents the peak of the triangle, where the membership degree is highest.
    - The **right point** at 70% indicates that the membership degree for "Moderate Humidity" is 0, meaning above 70% humidity no longer belongs to the "Moderate" category.

So, humidity values between 40% and 70% will have varying degrees of membership to the "Moderate Humidity" fuzzy set.

```
python
Copy code
H_H = fuzz.trimf(humidity_range, [70, 85, 100])  # High humidity
```

- **Purpose**: This defines the **High Humidity** fuzzy set.
- **`fuzz.trimf(humidity_range, [70, 85, 100])`**: This creates a triangular membership function on the `humidity_range` with the points `[70, 85, 100]`:
    - The **left point** at 70% indicates full membership to "High Humidity."

- The **middle point** at 85% represents the peak of the triangle, where the membership degree is highest.
- The **right point** at 100% indicates that the membership degree for "High Humidity" is 0, meaning above 100% humidity doesn't belong to the "High" category.

So, humidity values between 70% and 100% will have varying degrees of membership to the "High Humidity" fuzzy set.

Summary of the Triangular Membership Functions for Humidity:

1. **Low Humidity (H_L)**:
   - Full membership at 10% (degree = 1).
   - The membership decreases as humidity increases towards 40%, at which point the membership becomes 0.
2. **Moderate Humidity (H_M)**:
   - Full membership at 40% and 70%, with the highest membership at 55%.
3. **High Humidity (H_H)**:
   - Full membership at 70% (degree = 1).
   - The membership decreases as humidity increases towards 100%, at which point the membership becomes 0.

Purpose of Fuzzification:

Just like temperature, **fuzzification** allows us to handle **uncertain** or **imprecise values**. For example, if the humidity is 60%, the system can consider it as **partially moderate** and **partially high**, depending on the degree of membership for each fuzzy set. This is useful in real-world systems where humidity values are often not sharply defined as "low," "moderate," or "high," but instead have **degrees** of belonging to each category.

By defining these fuzzy sets, we can now use **fuzzy inference rules** to make decisions based on how the temperature and humidity interact with each other (as seen in the rules you mentioned earlier in the system).

```
# Fuzzification of Motor Speed (V)
V_L = fuzz.trimf(speed_range, [0, 5, 10])    # Low speed
V_M = fuzz.trimf(speed_range, [5, 10, 15])   # Moderate speed
V_H = fuzz.trimf(speed_range, [10, 15, 20]) # High speed
```

In this section, we are **fuzzifying the motor speed variable** by defining **membership functions** for the fuzzy sets: **Low**, **Moderate**, and **High** speed. Similar to temperature and humidity, we

use **triangular membership functions** to represent the degrees of membership for motor speed values.

Fuzzification of Motor Speed

We define three triangular membership functions to represent the motor speed fuzzy sets. These sets describe different motor speed levels: Low speed, Moderate speed, and High speed.

```
python
Copy code
V_L = fuzz.trimf(speed_range, [0, 5, 10])   # Low speed
```

- **Purpose**: This defines the **Low Speed** fuzzy set.
- **`fuzz.trimf(speed_range, [0, 5, 10])`**: This creates a triangular membership function on the `speed_range` (from 0 to 20). The points `[0, 5, 10]` define the shape of the triangle:
    - The **left point** at 0 represents full membership (degree = 1) to "Low Speed."
    - The **middle point** at 5 represents the peak of the triangle, where the membership degree starts to decrease.
    - The **right point** at 10 indicates that the membership degree for "Low Speed" becomes 0 at 10.

So, motor speed values between 0 and 10 will have varying degrees of membership to the "Low Speed" fuzzy set.

```
python
Copy code
V_M = fuzz.trimf(speed_range, [5, 10, 15])  # Moderate speed
```

- **Purpose**: This defines the **Moderate Speed** fuzzy set.
- **`fuzz.trimf(speed_range, [5, 10, 15])`**: This creates a triangular membership function with the points `[5, 10, 15]`:
    - The **left point** at 5 represents full membership (degree = 1) to "Moderate Speed."
    - The **middle point** at 10 represents the peak of the triangle, where the membership degree is highest.
    - The **right point** at 15 indicates that the membership degree for "Moderate Speed" becomes 0 at 15.

So, motor speed values between 5 and 15 will have varying degrees of membership to the "Moderate Speed" fuzzy set.

```
python
Copy code
V_H = fuzz.trimf(speed_range, [10, 15, 20]) # High speed
```

- **Purpose**: This defines the **High Speed** fuzzy set.
- **`fuzz.trimf(speed_range, [10, 15, 20])`**: This creates a triangular membership function with the points `[10, 15, 20]`:

- The **left point** at 10 represents full membership (degree = 1) to "High Speed."
- The **middle point** at 15 represents the peak of the triangle, where the membership degree is highest.
- The **right point** at 20 indicates that the membership degree for "High Speed" becomes 0 at 20.

So, motor speed values between 10 and 20 will have varying degrees of membership to the "High Speed" fuzzy set.

Summary of the Triangular Membership Functions for Motor Speed:

1. **Low Speed (V_L)**:
   - Full membership at 0 (degree = 1).
   - The membership decreases linearly as speed approaches 10, at which point the membership becomes 0.
2. **Moderate Speed (V_M)**:
   - Full membership at 5 and 15, with the highest membership at 10.
3. **High Speed (V_H)**:
   - Full membership at 10 (degree = 1).
   - The membership decreases linearly as speed approaches 20, at which point the membership becomes 0.

Purpose of Fuzzification:

- By defining fuzzy sets for **motor speed**, we allow the system to make decisions based on **degrees of membership** in different speed categories.
- For example, if the fuzzy system determines that a motor speed of **13** has some degree of membership in both the **Moderate Speed** and **High Speed** sets, it means that the system is considering the motor speed to be partially moderate and partially high.

This fuzzification approach is particularly useful in **control systems** where exact values may not be appropriate, and a gradual decision-making process is required.

```
# Visualize the fuzzy membership functions for each variable
def plot_memberships():
    plt.figure(figsize=(12, 8))

    # Plot Temperature membership functions
    plt.subplot(3, 1, 1)
    plt.plot(temperature_range, T_L, label='Low Temp')
    plt.plot(temperature_range, T_M, label='Moderate Temp')
    plt.plot(temperature_range, T_H, label='High Temp')
```

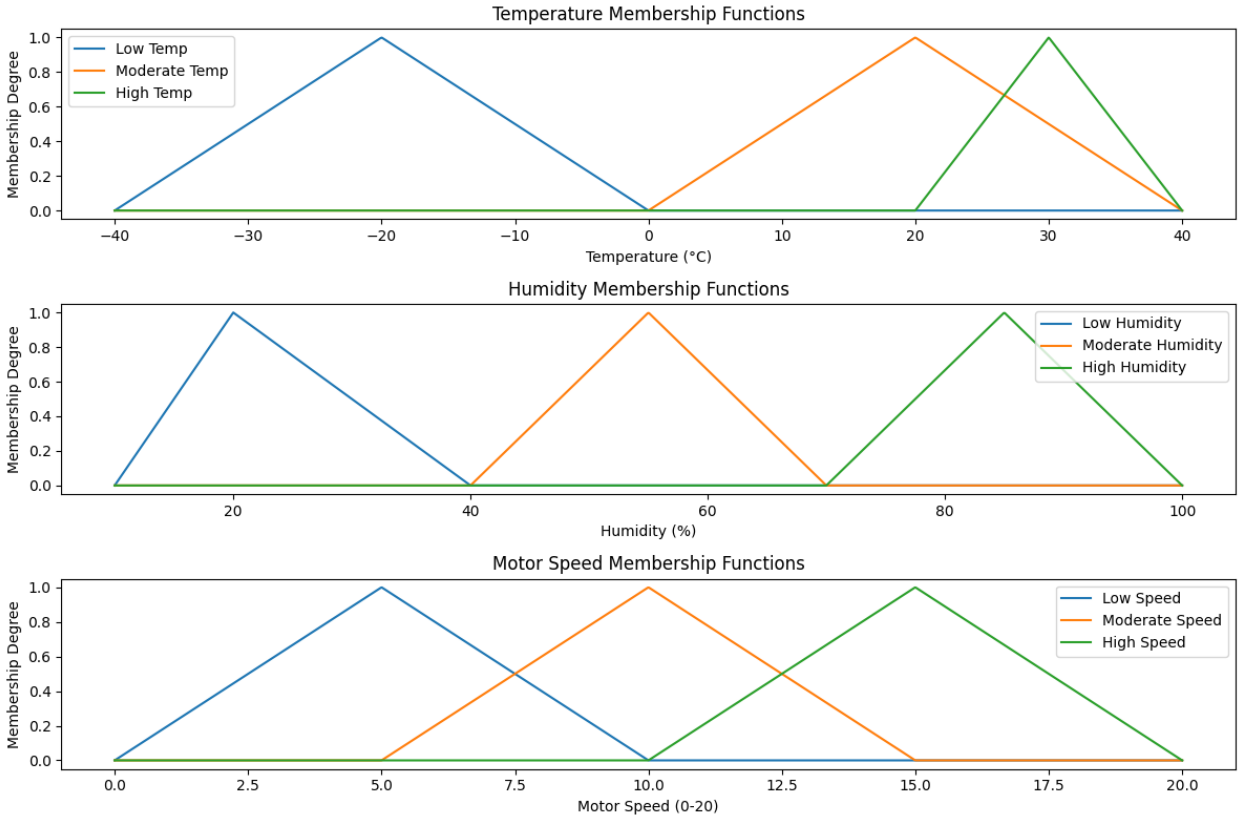
```
    plt.title('Temperature Membership Functions')
    plt.xlabel('Temperature (°C)')
    plt.ylabel('Membership Degree')
    plt.legend()

    # Plot Humidity membership functions
    plt.subplot(3, 1, 2)
    plt.plot(humidity_range, H_L, label='Low Humidity')
    plt.plot(humidity_range, H_M, label='Moderate Humidity')
    plt.plot(humidity_range, H_H, label='High Humidity')
    plt.title('Humidity Membership Functions')
    plt.xlabel('Humidity (%)')
    plt.ylabel('Membership Degree')
    plt.legend()

    # Plot Speed membership functions
    plt.subplot(3, 1, 3)
    plt.plot(speed_range, V_L, label='Low Speed')
    plt.plot(speed_range, V_M, label='Moderate Speed')
    plt.plot(speed_range, V_H, label='High Speed')
    plt.title('Motor Speed Membership Functions')
    plt.xlabel('Motor Speed (0-20)')
    plt.ylabel('Membership Degree')
    plt.legend()

    plt.tight_layout()
    plt.show()

plot_memberships()
```



This part of the code defines a function called `plot_memberships()`, which is responsible for visualizing the fuzzy membership functions for **temperature**, **humidity**, and **motor speed**. Visualizing these membership functions helps you better understand how each input variable maps to fuzzy sets and their corresponding degrees of membership.

Code Breakdown

1. **Function Definition**:

   ```
   python
   Copy code
   def plot_memberships():
   ```

   - This defines a function called `plot_memberships()` that, when called, will generate plots for the fuzzy membership functions of the three variables: temperature, humidity, and motor speed.

2. **Create the Plot Layout**:

   ```
   python
   Copy code
   plt.figure(figsize=(12, 8))
   ```

- This sets the size of the overall figure to be **12 units wide and 8 units tall**. The plots will be displayed in a grid with three subplots (one for each variable).

3. **Plot Temperature Membership Functions**:

```
python
Copy code
plt.subplot(3, 1, 1)
plt.plot(temperature_range, T_L, label='Low Temp')
plt.plot(temperature_range, T_M, label='Moderate Temp')
plt.plot(temperature_range, T_H, label='High Temp')
plt.title('Temperature Membership Functions')
plt.xlabel('Temperature (°C)')
plt.ylabel('Membership Degree')
plt.legend()
```

   - **`plt.subplot(3, 1, 1)`**: Creates the first subplot (3 rows, 1 column, position 1).
   - **`plt.plot()`**: This is used to plot each of the fuzzy membership functions for temperature (`T_L`, `T_M`, `T_H`) against the `temperature_range`.
   - **Labels and Legend**: The `label` parameter is used to name each line, which will be shown in the plot legend. The plot is titled, and the axes are labeled.

4. **Plot Humidity Membership Functions**:

```
python
Copy code
plt.subplot(3, 1, 2)
plt.plot(humidity_range, H_L, label='Low Humidity')
plt.plot(humidity_range, H_M, label='Moderate Humidity')
plt.plot(humidity_range, H_H, label='High Humidity')
plt.title('Humidity Membership Functions')
plt.xlabel('Humidity (%)')
plt.ylabel('Membership Degree')
plt.legend()
```

   - **`plt.subplot(3, 1, 2)`**: Creates the second subplot (3 rows, 1 column, position 2) for the humidity membership functions.
   - **`plt.plot()`**: This plots each fuzzy membership function for humidity (`H_L`, `H_M`, `H_H`) against the `humidity_range`.

5. **Plot Speed Membership Functions**:

```
python
Copy code
plt.subplot(3, 1, 3)
plt.plot(speed_range, V_L, label='Low Speed')
plt.plot(speed_range, V_M, label='Moderate Speed')
plt.plot(speed_range, V_H, label='High Speed')
plt.title('Motor Speed Membership Functions')
plt.xlabel('Motor Speed (0-20)')
plt.ylabel('Membership Degree')
plt.legend()
```

   - **`plt.subplot(3, 1, 3)`**: Creates the third subplot (3 rows, 1 column, position 3) for the motor speed membership functions.

- **`plt.plot()`**: This plots each fuzzy membership function for motor speed (`V_L`, `V_M`, `V_H`) against the `speed_range`.

6. **Layout Adjustment and Plot Display**:

```
python
Copy code
plt.tight_layout()
plt.show()
```

   - **`plt.tight_layout()`**: This adjusts the spacing between subplots to make sure everything fits without overlap.
   - **`plt.show()`**: This displays the plots.

Visual Output:

- The function will generate a figure with three subplots:
  1. **Temperature Membership Functions**: The plot will show how the temperature values are divided into Low, Moderate, and High categories, with each fuzzy set's membership degree gradually changing over the temperature range.
  2. **Humidity Membership Functions**: Similar to the temperature plot, this shows the fuzzified ranges for Low, Moderate, and High humidity.
  3. **Motor Speed Membership Functions**: This will display how motor speed values are divided into Low, Moderate, and High speed categories.

Purpose of Visualization:

Visualizing the membership functions allows you to:

- **See the shape of the fuzzy sets**: This helps you understand how each variable (temperature, humidity, and speed) is being divided into fuzzy categories.
- **Evaluate the fuzziness of the system**: You can visually assess how overlapping the fuzzy sets are for each variable, which will impact the final output of the fuzzy inference system.
- **Ensure logical behavior**: By plotting these functions, you can quickly check if the fuzzy sets make sense for your problem (e.g., that motor speed increases with higher temperature or humidity).

This part of the code implements the **fuzzy inference process**, which combines the fuzzified input values (temperature and humidity) using fuzzy rules to determine the motor speed. The inference process involves the following steps: fuzzification, applying fuzzy rules, aggregation, and defuzzification. Here's a breakdown of the code:

Function Definition

```
python
Copy code
def fuzzy_inference(temp, hum):
```

- This defines a function `fuzzy_inference` that takes two arguments:
  - `temp`: The temperature input value.
  - `hum`: The humidity input value.

Fuzzification of Inputs

```
python
Copy code
T_L_level = fuzz.interp_membership(temperature_range, T_L, temp)
T_M_level = fuzz.interp_membership(temperature_range, T_M, temp)
T_H_level = fuzz.interp_membership(temperature_range, T_H, temp)

H_L_level = fuzz.interp_membership(humidity_range, H_L, hum)
H_M_level = fuzz.interp_membership(humidity_range, H_M, hum)
H_H_level = fuzz.interp_membership(humidity_range, H_H, hum)
```

- **Fuzzify temperature and humidity**:
  - `fuzz.interp_membership()` is used to calculate the degree of membership for the input `temp` and `hum` in the respective fuzzy sets (Low, Moderate, High).
  - For temperature, the function calculates `T_L_level`, `T_M_level`, and `T_H_level`, which represent how much the temperature value belongs to the "Low," "Moderate," and "High" fuzzy sets, respectively.
  - For humidity, the function calculates `H_L_level`, `H_M_level`, and `H_H_level` in the same way.

Apply Fuzzy Rules

```
python
Copy code
rule1 = min(T_H_level, H_L_level)   # High Temp and Low Humidity → Low Speed
rule2 = min(T_M_level, H_L_level)   # Moderate Temp and Low Humidity → Low
Speed
rule3 = min(T_L_level, H_L_level)   # Low Temp and Low Humidity → High Speed
rule4 = H_M_level                    # Moderate Humidity → High Speed
rule5 = min(T_L_level, H_H_level)   # Low Temp and High Humidity → High Speed
```

- **Rules based on expert knowledge**: The expert's rules are applied here:
  1. **Rule 1**: If temperature is high and humidity is low, the motor speed should be low. This is represented by `min(T_H_level, H_L_level)`, which takes the minimum of the two fuzzy memberships (because both conditions must be satisfied).
  2. **Rule 2**: If temperature is moderate and humidity is low, the motor speed should be low. This is represented by `min(T_M_level, H_L_level)`.
  3. **Rule 3**: If temperature is low and humidity is low, the motor speed should be high. This is represented by `min(T_L_level, H_L_level)`.
  4. **Rule 4**: If humidity is moderate, the motor speed should be high. This is represented by `H_M_level`.

5. **Rule 5**: If temperature is low and humidity is high, the motor speed should be high. This is represented by `min(T_L_level, H_H_level)`.

Aggregate Outputs

```
python
Copy code
aggregated_Low = max(rule1, rule2)
aggregated_High = max(rule3, rule4, rule5)
```

- **Aggregation**:
  - The **OR operator** is used for aggregation. In fuzzy logic, **OR** means the **maximum** value of the rule outputs:
    - `aggregated_Low` is the maximum of **Rule 1** and **Rule 2**, which determines how much the motor speed should be low.
    - `aggregated_High` is the maximum of **Rule 3**, **Rule 4**, and **Rule 5**, which determines how much the motor speed should be high.

Defuzzification

```
python
Copy code
V_L_output = fuzz.defuzz(speed_range, V_L, 'centroid') * aggregated_Low
V_H_output = fuzz.defuzz(speed_range, V_H, 'centroid') * aggregated_High
```

- **Defuzzification**:
  - `fuzz.defuzz()` is used to **defuzzify** the fuzzy outputs. The `'centroid'` method is used, which finds the **center of gravity** (the weighted average of the speed values based on the membership functions).
  - `V_L_output` and `V_H_output` represent the **crisp values** for **low speed** and **high speed**, respectively, after multiplying by the aggregated membership degrees (`aggregated_Low` and `aggregated_High`).

Combine the Outputs

```
python
Copy code
motor_speed = V_L_output + V_H_output
```

- The **final motor speed** is obtained by **adding** the defuzzified outputs for low and high speeds. This step combines the fuzzy outputs and calculates the final motor speed by using the **max rule** for aggregation.

Return the Result

```
python
Copy code
return motor_speed
```

- The function returns the final motor speed, which is a **crisp value** representing the motor speed based on the given temperature and humidity inputs.

Example of How It Works:

Let's walk through an example to understand the process:

1. **Input**: Suppose the temperature is `20°C` and the humidity is `60%`.
2. **Fuzzification**:
   - The temperature `20°C` will have some membership in the **Moderate Temperature** fuzzy set, and some membership in the **High Temperature** fuzzy set.
   - The humidity `60%` will have some membership in the **Moderate Humidity** fuzzy set.
3. **Rule Application**:
   - Based on the fuzzified values of temperature and humidity, we apply the fuzzy rules.
   - For example, if the temperature is moderate and the humidity is moderate, rules like "Moderate Temp and Moderate Humidity → Moderate Speed" may be triggered.
4. **Aggregation**:
   - The results from different rules are aggregated using the **max** operator.
5. **Defuzzification**:
   - The **defuzzified outputs** for low and high speeds are calculated using the **centroid method**, and they are weighted by the aggregated membership values.
6. **Final Output**:
   - The final motor speed is returned, which is a single **crisp value** representing the optimal speed based on the given inputs.

```
# Test with inputs: Temperature = 20°C, Humidity = 90%
temp_input = 20
hum_input = 90

motor_speed_output = fuzzy_inference(temp_input, hum_input)

print(f'Motor Speed for Temperature = {temp_input}°C and Humidity =
{hum_input}%: {motor_speed_output:.2f}')
```

In this section, we are testing the fuzzy inference system by providing **specific input values** for **temperature** and **humidity**. The goal is to determine the corresponding **motor speed** based on the fuzzy inference system that was previously defined.

Code Breakdown:

1. **Input Values**:

```
python
Copy code
temp_input = 20
hum_input = 90
```

   - We are testing with a **temperature of 20°C** and **humidity of 90%**. These values are realistic conditions, as 20°C is a moderate temperature and 90% humidity is quite high.

2. **Function Call**:

```
python
Copy code
motor_speed_output = fuzzy_inference(temp_input, hum_input)
```

   - The function `fuzzy_inference()` is called with the test inputs (`temp_input` and `hum_input`), which returns the **motor speed** as a crisp value.

3. **Output**:

```
python
Copy code
print(f'Motor Speed for Temperature = {temp_input}°C and Humidity =
{hum_input}%: {motor_speed_output:.2f}')
```

   - The result is printed to the console with a formatted string that rounds the motor speed to two decimal places.

Expected Output:

Given the inputs:

- **Temperature = 20°C** (Moderate temperature, partially high and moderate membership).
- **Humidity = 90%** (High humidity, with full membership in the "High Humidity" fuzzy set).

The fuzzy inference system will likely apply the following rules:

- **Rule 1**: High Temperature and Low Humidity → Low Speed (not likely to be triggered because humidity is high).
- **Rule 2**: Moderate Temperature and Low Humidity → Low Speed (not triggered because humidity is high).
- **Rule 3**: Low Temperature and Low Humidity → High Speed (not triggered because temperature is not low).
- **Rule 4**: Moderate Humidity → High Speed (this rule is likely triggered because the humidity is 90%).
- **Rule 5**: Low Temperature and High Humidity → High Speed (this rule is also likely triggered because the humidity is high).

Since the aggregation of rules 4 and 5 results in a high motor speed, the final motor speed will be a **high value** within the 0-20 range.

```
# Test with other inputs
inputs = [(10, 50), (-20, 20), (35, 80)]
for temp, hum in inputs:
    motor_speed = fuzzy_inference(temp, hum)
    print(f'Motor Speed for Temperature = {temp}°C and Humidity = {hum}%:
{motor_speed:.2f}')
```

This part of the code tests the fuzzy inference system with multiple input combinations for temperature and humidity. We will evaluate how the system reacts to different environmental conditions and whether the resulting motor speed is reasonable based on the fuzzy rules.

Code Breakdown:

1. **Test Inputs**:

   ```
   python
   Copy code
   inputs = [(10, 50), (-20, 20), (35, 80)]
   ```

   - Three different pairs of temperature and humidity are provided for testing:
     - **(10, 50)**: Moderate temperature (10°C) and moderate humidity (50%).
     - **(-20, 20)**: Low temperature (-20°C) and low humidity (20%).
     - **(35, 80)**: High temperature (35°C) and high humidity (80%).
2. **Loop through Inputs**:

   ```
   python
   Copy code
   for temp, hum in inputs:
       motor_speed = fuzzy_inference(temp, hum)
       print(f'Motor Speed for Temperature = {temp}°C and Humidity =
   {hum}%: {motor_speed:.2f}')
   ```

   - This loop goes through each test case, calls the `fuzzy_inference()` function, and prints the resulting motor speed with the input temperature and humidity values.

Expected Results:

1. **For Input (10°C, 50%)**:
   - **Temperature**: 10°C is a moderate temperature.
   - **Humidity**: 50% is moderate humidity.

    - The system is likely to trigger **Rule 4** (Moderate Humidity → High Speed), but the exact output will depend on the aggregation of other rules.
2. **For Input (-20°C, 20%)**:
    - **Temperature**: -20°C is a low temperature.
    - **Humidity**: 20% is low humidity.
    - **Rule 3**: Low Temperature and Low Humidity → High Speed will likely be triggered, resulting in a **high speed**.
3. **For Input (35°C, 80%)**:
    - **Temperature**: 35°C is a high temperature.
    - **Humidity**: 80% is high humidity.
    - **Rule 5**: Low Temperature and High Humidity → High Speed will likely be triggered, and the system will likely assign a **high speed**.

Example of the Output:

- For **(10, 50)**: Moderate conditions could lead to a **moderate motor speed**.
- For **(-20, 20)**: Low temperature and low humidity should result in a **high motor speed**.
- For **(35, 80)**: High temperature and high humidity should also result in a **high motor speed**.