vh6gkypev

December 6, 2024

1 Name: Ramesh Chandra Soren

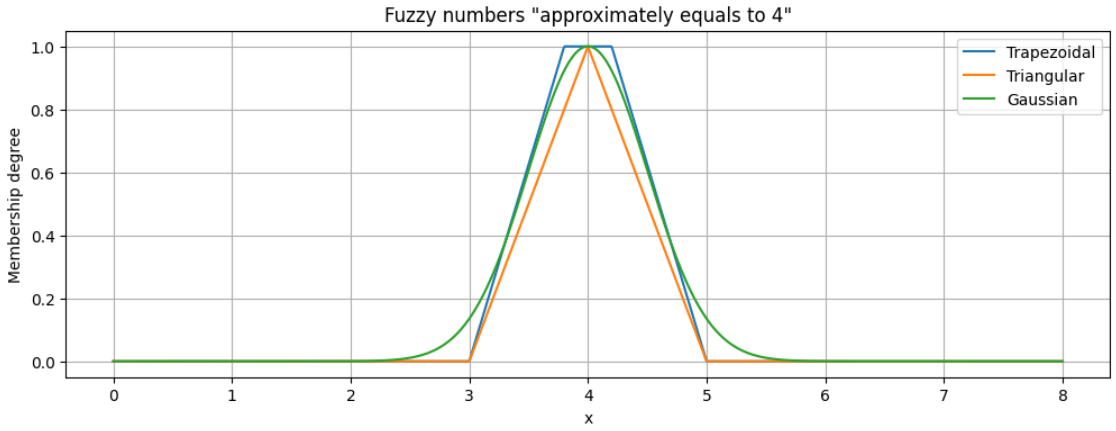
2 Enrollment No: 2022CSB086

2.1 Design Mamdani Fuzzy Inference System to control the FAN-SPEED of a furnace by inputting TEMPERATURE of a thermostat of a household.

2.2 Frame the If-Then rules using Linguistic variables input TEMPERATURE and output FAN-SPEED.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

def trapezoidal_mf(x, a, b, c, d):
    return np.maximum(np.minimum(np.minimum((x-a)/(b-a), 1), (d-x)/(d-c)), 0)

def triangular_mf(x, a, b, c):
    return np.maximum(np.minimum((x-a)/(b-a), (c-x)/(c-b)), 0)

def gaussian_mf(x, mean, sigma):
    return np.exp(-((x - mean)**2) / (2 * sigma**2))

x = np.linspace(0, 8, 1000)

# Trapezoidal
trap_mf = trapezoidal_mf(x, 3, 3.8, 4.2, 5)

# Triangular
tri_mf = triangular_mf(x, 3, 4, 5)

# Gaussian
gauss_mf = gaussian_mf(x, 4, 0.5)

plt.figure(figsize=(12, 4))
plt.plot(x, trap_mf, label='Trapezoidal')
plt.plot(x, tri_mf, label='Triangular')
plt.plot(x, gauss_mf, label='Gaussian')
plt.title('Fuzzy numbers "approximately equals to 4"')
```

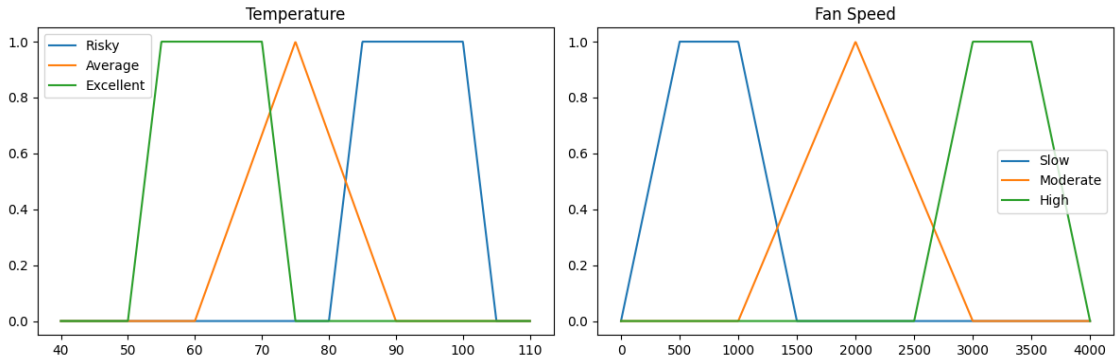
```
plt.legend()
plt.xlabel('x')
plt.ylabel('Membership degree')
plt.grid(True)
plt.show()
```



3 Define membership functions

[ ]:
```
def temp_risky(x):
    return trapezoidal_mf(x, 80, 85, 100, 105)

def temp_average(x):
    return triangular_mf(x, 60, 75, 90)

def temp_excellent(x):
    return trapezoidal_mf(x, 50, 55, 70, 75)

def fan_speed_slow(x):
    return trapezoidal_mf(x, 0, 500, 1000, 1500)

def fan_speed_moderate(x):
    return triangular_mf(x, 1000, 2000, 3000)

def fan_speed_high(x):
    return trapezoidal_mf(x, 2500, 3000, 3500, 4000)

temp = np.linspace(40, 110, 1000)
fan_speed = np.linspace(0, 4000, 1000)

plt.figure(figsize=(12, 4))
plt.subplot(121)
```

```
plt.plot(temp, temp_risky(temp), label='Risky')
plt.plot(temp, temp_average(temp), label='Average')
plt.plot(temp, temp_excellent(temp), label='Excellent')
plt.title('Temperature')
plt.legend()

plt.subplot(122)
plt.plot(fan_speed, fan_speed_slow(fan_speed), label='Slow')
plt.plot(fan_speed, fan_speed_moderate(fan_speed), label='Moderate')
plt.plot(fan_speed, fan_speed_high(fan_speed), label='High')
plt.title('Fan Speed')
plt.legend()

plt.tight_layout()
plt.show()
```

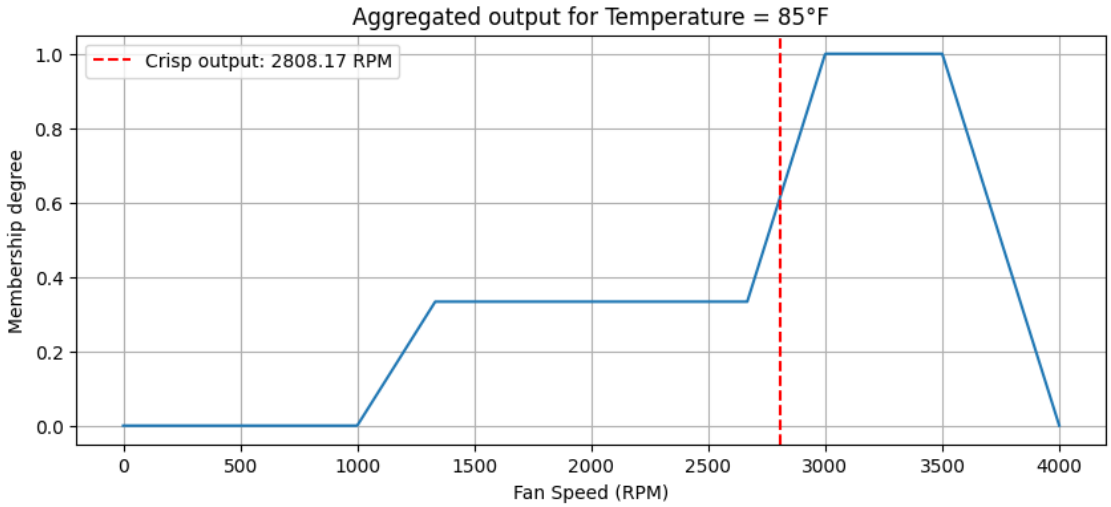


3.1 This script performs the following steps:

1. Fuzzification: Convert the crisp input temperature to fuzzy membership degrees in each temperature set (Risky, Average, Excellent).
2. Rule Evaluation: Apply each rule using the minimum operator between the input membership degree and the corresponding output fuzzy set.
3. Aggregation: Combine the outputs of all rules using the maximum operator.
4. Defuzzification: Calculate the centroid of the aggregated output to get a crisp fan speed value.

[ ]:
```
def mamdani_inference(temp_input):
    # Rule evaluation
    rule1 = np.minimum(temp_risky(temp_input), fan_speed_high(fan_speed))
    rule2 = np.minimum(temp_average(temp_input), fan_speed_moderate(fan_speed))
    rule3 = np.minimum(temp_excellent(temp_input), fan_speed_slow(fan_speed))

    # Aggregation
```

```
    aggregated = np.maximum.reduce([rule1, rule2, rule3])

    # Defuzzification (Centroid method)
    numerator = np.sum(aggregated * fan_speed)
    denominator = np.sum(aggregated)

    crisp_output = numerator / denominator

    return crisp_output, aggregated

# Example usage
temp_input = 85
crisp_fan_speed, aggregated_output = mamdani_inference(temp_input)

plt.figure(figsize=(10, 4))
plt.plot(fan_speed, aggregated_output)
plt.axvline(crisp_fan_speed, color='r', linestyle='--', label=f'Crisp output:␣
 ↪{crisp_fan_speed:.2f} RPM')
plt.title(f'Aggregated output for Temperature = {temp_input}°F')
plt.xlabel('Fan Speed (RPM)')
plt.ylabel('Membership degree')
plt.legend()
plt.grid(True)
plt.show()

print(f"For input temperature {temp_input}°F, the crisp fan speed is␣
 ↪{crisp_fan_speed:.2f} RPM")
```




```
For input temperature 85°F, the crisp fan speed is 2808.17 RPM
```