รายงาน

เรื่อง

ระบบควบคุมสภาพอากาศในเรือนกระจกโดยใช้ Fuzzy Logic : Mamdani

โดย

น.ส. ภัทรจาริน ผดุงกิจเจริญ รหัสนักศึกษา 650610851

เสนอ

รศ.ดร. ศันสนีย์ เอื้อพันธ์วิริยะกุล

รายงานนี้เป็นส่วนหนึ่งของรายวิชา CPE 261456

Introduction to Computational Intelligence

สาขาวิชาวิศวกรรมหุ่นยนต์และปัญญาประดิษฐ์

ภาคเรียนที่ 1 ปีการศึกษา 2567

มหาวิทยาลัยเชียงใหม่

สารบัญ

ส่วนที่ 1

1.1 ลักษณะการทำงานของระบบ

ระบบควบคุมสภาพอากาศในเรือนกระจกนี้ถูกออกแบบมาเพื่อควบคุมอุณหภูมิ ความชื้น และแสงให้เหมาะสมสำหรับการเจริญเติบโตของพืช โดยใช้หลักการของ Fuzzy Logic ในการควบคุม ซึ่งช่วยปรับพัดลม ระบบพ่นหมอก และไฟ LED ตามสภาพแวดล้อมที่เปลี่ยนแปลง ระบบมีอินพุต 3 ค่า ได้แก่ : อุณหภูมิ (°C) , ความชื้น (%) , แสง (%) และเอาท์พุต 3 ค่า ได้แก่ : พัดลม: ปรับความแรงของพัดลม (0-100%) , ระบบพ่นหมอก: ปรับความเข้มของระบบพ่นหมอก (0-100%) , ไฟ LED: ปรับความเข้มของไฟ LED (0-100%) ระบบจะทำงานตามกฎที่ออกแบบไว้เพื่อตอบสนองต่อสภาพแวดล้อมที่เปลี่ยนแปลง

1.2 Fuzzy Rules

ถ้า อุณหภูมิสูง และ ความชื้นสูง และ แสงมาก ⟶ เพิ่ม พัดลม, ลด ระบบพ่นหมอก, ลด ไฟLED

ถ้า อุณหภูมิสูง และ ความชื้นสูง และ แสงน้อย ⟶ เพิ่ม พัดลม, ลด ระบบพ่นหมอก, เพิ่ม ไฟLED

ถ้า อุณหภูมิสูง และ ความชื้นต่ำ และ แสงมาก ⟶ เพิ่ม พัดลม, เพิ่ม ระบบพ่นหมอก, ลด ไฟLED

ถ้า อุณหภูมิสูง และ ความชื้นต่ำ และ แสงน้อย ⟶ เพิ่ม พัดลม, เพิ่ม ระบบพ่นหมอก, เพิ่ม ไฟLED

ถ้า อุณหภูมิต่ำ และ ความชื้นสูง และ แสงมาก ⟶ ลด พัดลม, ลด ระบบพ่นหมอก, ลด ไฟLED

ถ้า อุณหภูมิต่ำ และ ความชื้นสูง และ แสงน้อย ⟶ ลด พัดลม, ลด ระบบพ่นหมอก, เพิ่ม ไฟLED

ถ้า อุณหภูมิต่ำ และ ความชื้นต่ำ และ แสงมาก ⟶ ลด พัดลม, เพิ่ม ระบบพ่นหมอก, ลด ไฟLED

ถ้า อุณหภูมิต่ำ และ ความชื้นต่ำ และ แสงน้อย ⟶ ลด พัดลม, เพิ่ม ระบบพ่นหมอก, เพิ่ม ไฟLED

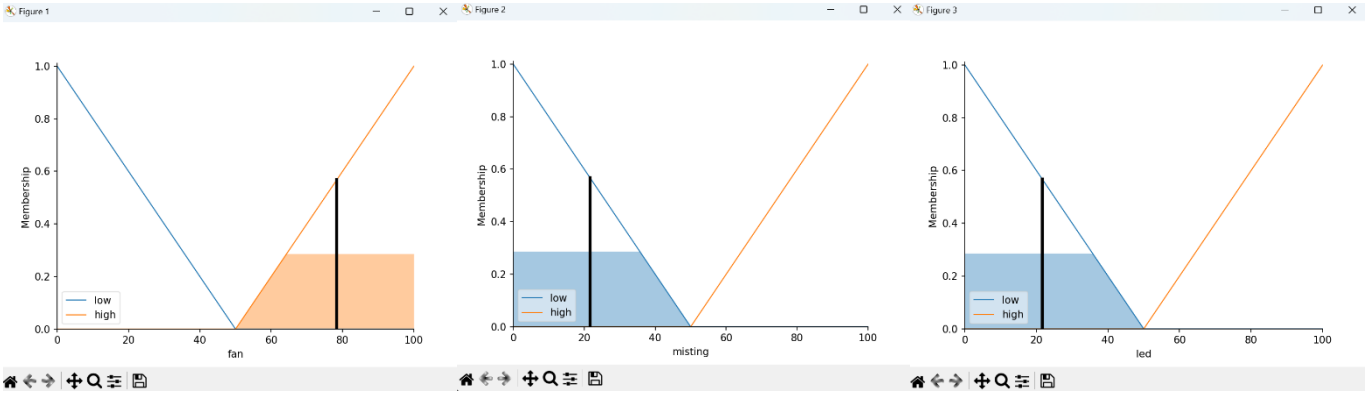
ส่วนที่ 2

2.1 simulation ของระบบ ผลการทดลอง และวิเคราะห์

ตัวอย่าง [ถ้า อุณหภูมิสูง และ ความชื้นสูง และ แสงมาก ⟶ เพิ่ม พัดลม, ลด ระบบพ่นหมอก, ลด ไฟLED]

ใช้ค่าอินพุตตัวอย่างเพื่อทดสอบการทำงานของระบบ : อุณหภูมิ: 30°C "high", ความชื้น: 70% "high", แสง: 90% "high"

ผลลัพธ์ที่ได้รับคือ : ความเร็วพัดลม 78.37% "high", ระบบพ่นหมอก 21.63% "low", ไฟ LED 21.63% "low"



การวิเคราะห์ผลลัพธ์

ความเร็วพัดลม: การตั้งค่าความเร็วพัดลมที่สูง 78.37% เป็นไปตามการควบคุมที่ตอบสนองต่ออุณหภูมิที่สูง ในกรณีนี้ พัดลมต้องทำงานมากขึ้นเพื่อช่วยระบายความร้อนออกจากเรือนกระจก

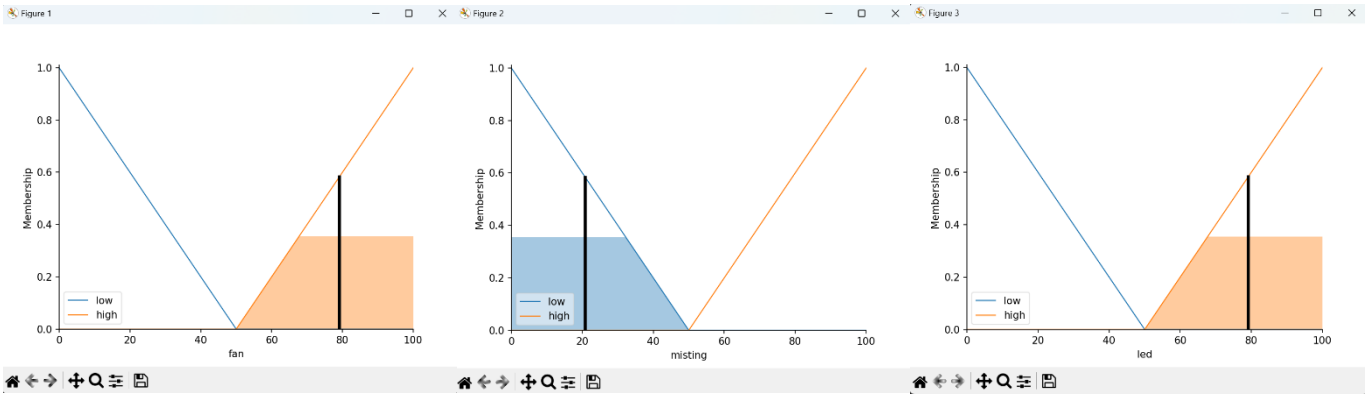
ระบบพ่นหมอก: การตั้งค่าระบบพ่นหมอกที่ 21.63% แสดงถึงความต้องการลดความชื้นในสภาพที่มีความชื้นสูง

ไฟ LED: การตั้งค่าไฟ LED ที่ 21.63% แสดงถึงการลดแสงสว่าง ในกรณีที่แสงในเรือนกระจกสูง

ตัวอย่าง [ถ้า อุณหภูมิสูง และ ความชื้นสูง และ แสงน้อย ⟶ เพิ่ม พัดลม, ลด ระบบพ่นหมอก, เพิ่ม ไฟLED]

ใช้ค่าอินพุตตัวอย่างเพื่อทดสอบการทำงานของระบบ : อุณหภูมิ: 35°C "high", ความชื้น: 70% "high", แสง: 30% "low"

ผลลัพธ์ที่ได้รับคือ : ความเร็วพัดลม 79.12% "high", ระบบพ่นหมอก 20.88% "low", ไฟ LED 79.12% "high"



การวิเคราะห์ผลลัพธ์

ความเร็วพัดลม: การทำงานของพัดลมที่เพิ่มขึ้นถึง 79.12% มีเป้าหมายเพื่อช่วยระบายความร้อน เนื่องจากอุณหภูมิในเรือนกระจกสูง

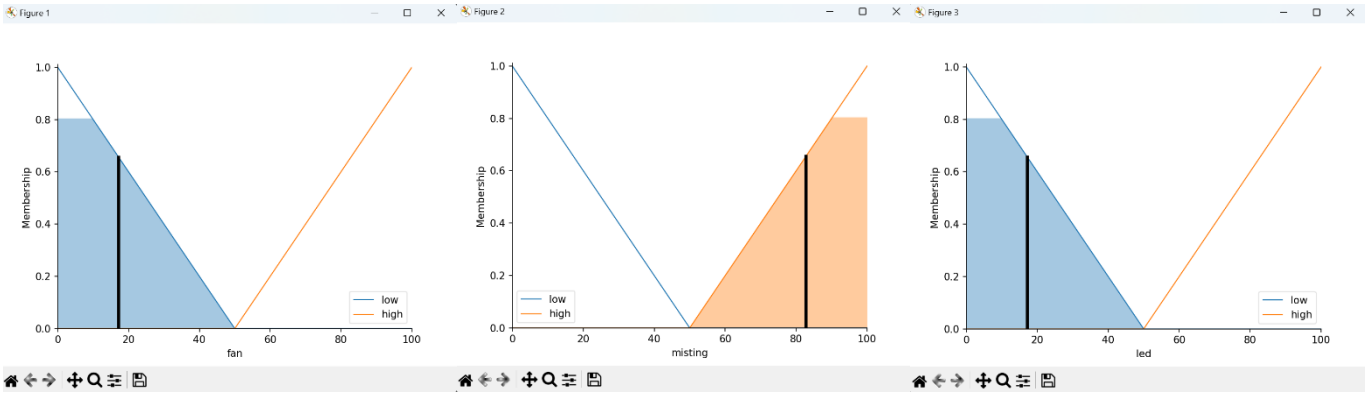
ระบบพ่นหมอก: ระบบพ่นหมอกลดลงเหลือ 20.88% เนื่องจากความชื้นในเรือนกระจกอยู่ในระดับสูงและไม่จำเป็นต้องเพิ่มความชื้นมากขึ้น

ไฟ LED: ไฟ LED ทำงานถึง 79.12% เพื่อเพิ่มแสง เนื่องจากระดับแสงที่มียังไม่เพียงพอต่อการเจริญเติบโตของพืช

ตัวอย่าง [ถ้า อุณหภูมิต่ำ และ ความชื้นต่ำ และ แสงมาก ⟶ ลด พัดลม, เพิ่ม ระบบพ่นหมอก, ลด ไฟLED]

ใช้ค่าอินพุตตัวอย่างเพื่อทดสอบการทำงานของระบบ : อุณหภูมิ: 17°C "low", ความชื้น: 35% "low", แสง: 90% "high"

ผลลัพธ์ที่ได้รับคือ : ความเร็วพัดลม 17.20% "low", ระบบพ่นหมอก 82.80% "high", ไฟ LED 17.20% "low"



การวิเคราะห์ผลลัพธ์

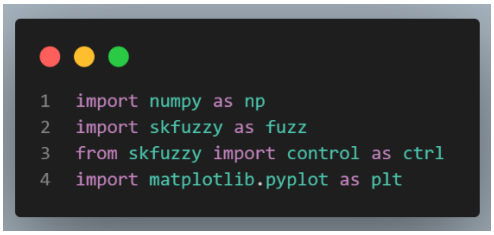
ความเร็วพัดลม: เนื่องจากอุณหภูมิอยู่ที่ระดับต่ำ ระบบจึงลดการทำงานของพัดลมลงมาเพียง 17.20%

ระบบพ่นหมอก: ระบบพ่นหมอกถูกเพิ่มขึ้นถึง 82.80% เพื่อเพิ่มความชื้นในอากาศ เนื่องจากความชื้นในปัจจุบันต่ำ

ไฟ LED: การทำงานของไฟ LED ถูกลดลงถึง 17.20% เนื่องจากแสงจากธรรมชาติมีมากเพียงพอแล้ว

ส่วนที่ 3

GITHUB : https://github.com/Pattharajrin/261456_CI_HW2_Y3-1.git



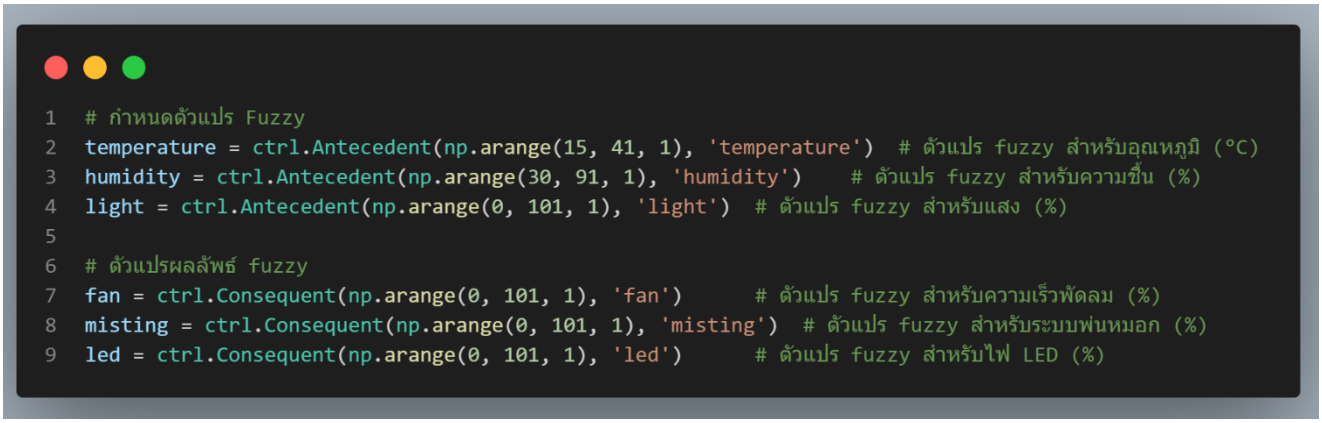
```python
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
```

การนำเข้าไลบรารี : numpy: ใช้สำหรับจัดการอาร์เรย์และการคำนวณทางคณิตศาสตร์

skfuzzy: ไลบรารีสำหรับทำงานกับฟัซซี่ลอจิก

control as ctrl: สำหรับสร้างระบบควบคุมฟัซซี่ลอจิก

matplotlib.pyplot: ใช้สำหรับการแสดงผลกราฟ



```python
# กำหนดตัวแปร Fuzzy
temperature = ctrl.Antecedent(np.arange(15, 41, 1), 'temperature')  # ตัวแปร fuzzy สำหรับอุณหภูมิ (°C)
humidity = ctrl.Antecedent(np.arange(30, 91, 1), 'humidity')    # ตัวแปร fuzzy สำหรับความชื้น (%)
light = ctrl.Antecedent(np.arange(0, 101, 1), 'light')  # ตัวแปร fuzzy สำหรับแสง (%)

# ตัวแปรผลลัพธ์ fuzzy
fan = ctrl.Consequent(np.arange(0, 101, 1), 'fan')      # ตัวแปร fuzzy สำหรับความเร็วพัดลม (%)
misting = ctrl.Consequent(np.arange(0, 101, 1), 'misting')  # ตัวแปร fuzzy สำหรับระบบพ่นหมอก (%)
led = ctrl.Consequent(np.arange(0, 101, 1), 'led')      # ตัวแปร fuzzy สำหรับไฟ LED (%)
```

การกำหนดตัวแปร Fuzzy (Antecedents และ Consequents) :

กำหนดตัวแปร Antecedents (อินพุต): temperature (อุณหภูมิ), humidity (ความชื้น), light (แสง)

กำหนดตัวแปร Consequents (ผลลัพธ์): fan (พัดลม), misting (ระบบพ่นหมอก), led (ไฟ LED)

```python
# ฟังก์ชันสมาชิก (Membership Functions) สำหรับแต่ละตัวแปร
temperature['low'] = fuzz.trimf(temperature.universe, [15, 15, 28])    # ฟังก์ชันสมาชิก "low" ของอุณหภูมิ
temperature['high'] = fuzz.trimf(temperature.universe, [26, 40, 40])   # ฟังก์ชันสมาชิก "high" ของอุณหภูมิ

humidity['low'] = fuzz.trimf(humidity.universe, [30, 30, 61])      # ฟังก์ชันสมาชิก "low" ของความชื้น
humidity['high'] = fuzz.trimf(humidity.universe, [59, 90, 90])      # ฟังก์ชันสมาชิก "high" ของความชื้น

light['low'] = fuzz.trimf(light.universe, [0, 0, 51])            # ฟังก์ชันสมาชิก "low" ของแสง
light['high'] = fuzz.trimf(light.universe, [49, 100, 100])        # ฟังก์ชันสมาชิก "high" ของแสง

fan['low'] = fuzz.trimf(fan.universe, [0, 0, 50])                # ฟังก์ชันสมาชิก "low" ของความเร็วพัดลม
fan['high'] = fuzz.trimf(fan.universe, [50, 100, 100])            # ฟังก์ชันสมาชิก "high" ของความเร็วพัดลม

misting['low'] = fuzz.trimf(misting.universe, [0, 0, 50])        # ฟังก์ชันสมาชิก "low" ของระบบพ่นหมอก
misting['high'] = fuzz.trimf(misting.universe, [50, 100, 100])    # ฟังก์ชันสมาชิก "high" ของระบบพ่นหมอก

led['low'] = fuzz.trimf(led.universe, [0, 0, 50])                # ฟังก์ชันสมาชิก "low" ของไฟ LED
led['high'] = fuzz.trimf(led.universe, [50, 100, 100])            # ฟังก์ชันสมาชิก "high" ของไฟ LED
```

การกำหนดฟังก์ชันสมาชิก (Membership Functions) : กำหนด ฟังก์ชันสมาชิก สำหรับตัวแปรแต่ละตัวในรูปแบบของฟังก์ชันแบบสามเหลี่ยม (Triangular Membership Function: trimf) โดยระบุค่าเริ่มต้น, ค่ากลาง, และค่าสูงสุด

```python
# กำหนดกฎควบคุม (Rules)
# ตัวอย่างกฎการควบคุมระบบ Fuzzy สำหรับสภาพอากาศในเรือนกระจก
rule1 = ctrl.Rule(temperature['high'] & humidity['high'] & light['high'], (fan['high'], misting['low'], led['low']))
rule2 = ctrl.Rule(temperature['high'] & humidity['high'] & light['low'], (fan['high'], misting['low'], led['high']))
rule3 = ctrl.Rule(temperature['high'] & humidity['low'] & light['high'], (fan['high'], misting['high'], led['low']))
rule4 = ctrl.Rule(temperature['high'] & humidity['low'] & light['low'], (fan['high'], misting['high'], led['high']))
rule5 = ctrl.Rule(temperature['low'] & humidity['high'] & light['high'], (fan['low'], misting['low'], led['low']))
rule6 = ctrl.Rule(temperature['low'] & humidity['high'] & light['low'], (fan['low'], misting['low'], led['high']))
rule7 = ctrl.Rule(temperature['low'] & humidity['low'] & light['high'], (fan['low'], misting['high'], led['low']))
rule8 = ctrl.Rule(temperature['low'] & humidity['low'] & light['low'], (fan['low'], misting['high'], led['high']))
```

การสร้างกฎควบคุม (Fuzzy Rules) : สร้างกฎควบคุมฟัซซี่เพื่อกำหนดการทำงานของพัดลม, ระบบพ่นหมอก, และไฟ LED โดยอ้างอิงจากสภาวะของอุณหภูมิ, ความชื้น, และแสง

```python
# สร้างระบบควบคุมและการจำลอง
control_system = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8])
simulation = ctrl.ControlSystemSimulation(control_system)
```

การสร้างระบบควบคุมและการจำลอง : สร้างระบบควบคุมฟัซซี่ (ControlSystem) จากกฎที่กำหนดและสร้างการจำลอง (ControlSystemSimulation) เพื่อตรวจสอบผลลัพธ์

```
# ทดสอบระบบควบคุมด้วยค่าอินพุต
simulation.input['temperature'] = 30  # ตั้งค่าอุณหภูมิ (°C) เพื่อทดสอบ (15-40 °C)
simulation.input['humidity'] = 70     # ตั้งค่าความชื้น (%) เพื่อทดสอบ (30-90 %)
simulation.input['light'] = 90        # ตั้งค่าแสง (%) เพื่อทดสอบ (0-100 %)
```

การทดสอบระบบควบคุมด้วยค่าอินพุต : ตั้งค่าค่าของอุณหภูมิ, ความชื้น, และแสงเพื่อตรวจสอบการทำงานของระบบ

```
# คำนวณผลลัพธ์จากการทดสอบ
simulation.compute()

# แสดงผลลัพธ์
print(f"Fan Speed: {simulation.output['fan']:.2f}%")      # แสดงผลความเร็วพัดลม
print(f"Misting System: {simulation.output['misting']:.2f}%")  # แสดงผลระบบพ่นหมอก
print(f"LED Lights: {simulation.output['led']:.2f}%")      # แสดงผลไฟ LED
```

การคำนวณผลลัพธ์และแสดงผล : คำนวณผลลัพธ์ตามอินพุตที่ตั้งไว้ และแสดงผลความเร็วพัดลม, ระบบพ่นหมอก, และไฟ LED

```
# แสดงกราฟฟังก์ชันสมาชิก
fan.view(simulation)          # แสดงกราฟฟังก์ชันสมาชิกของพัดลม
misting.view(simulation)      # แสดงกราฟฟังก์ชันสมาชิกของระบบพ่นหมอก
led.view(simulation)          # แสดงกราฟฟังก์ชันสมาชิกของไฟ LED

# แสดงผลลัพธ์กราฟฟังก์ชันสมาชิก
plt.show()
```

การแสดงกราฟฟังก์ชันสมาชิก : แสดงกราฟฟังก์ชันสมาชิกของตัวแปรผลลัพธ์ (พัดลม, ระบบพ่นหมอก, ไฟ LED) และแสดงผลด้วย plt.show()