**Computer Intelligence**

**Kunnapat Thippayapalaphonkul (590612113)**

**18 November 2019**

**รายงานผลการทดลองการบ้าน Fuzzy**

ในการบ้านนี้นั้นได้ทำการเรียกใช้ Libraries คือ 1). matplotlib 2). numpy 3). random โดยที่ matplotlib ใช้ plot กราฟ, numpy สร้าง array, matrix, random สุ่มค่าให้ Input

รายงานผลการทดลองผลการคั่วเมล็ดกาแฟเพื่อที่จะได้เมล็ดกาแฟในแต่ละระดับ ที่มีความไม่แน่นอนโดยใช้วิธี Fuzzy Mamdani โดยปัจจัยในการควบคุมการคั่วกาแฟนั้นมีอยู่มากมาย ก็คือ

* อุณหภูมิ
* เวลา
* ความชื้นในเมล็ดกาแฟ
* ชนิดของสายพันธุ์เมล็ดกาแฟ
* แหล่งการใช้พลังงานเครื่องคั่วกาแฟ
* อุณหภูมิเริ่มต้น
* อุณหภูมิจบ
* ระยะเวลาของการคั่วเมล็ดกาแฟ
* สภาพความชื้นขณะคั่วกาแฟ
* ระบบถ่ายเทความร้อนภายในเครื่องคั่วเมล็ดกาแฟ

แต่ในการบ้านนี้จะขอยกปัจจัยมาเพียง 2 ปัจจัยเท่านั้นที่จะควบคุมการคั่วเมล็ดกาแฟให้ออกมาได้ในแต่ละระดับคือ

* อุณหภูมิ โดยที่ถือว่าอุณหภูมิเริ่มต้นกับอุณหภูมิจบนั้นให้เท่ากัน
* ระยะเวลาของการคั่วเมล็ดกาแฟ

**Fuzzy Rules**

กฎของ Fuzzy ที่ผมได้สร้างขึ้นมาคือรับ Input เข้าไปในระบบ 2 ค่า คือ 1). อุณหภูมิหน่วยเป็นองศาเซลเซียส 2). เวลาหน่วยเป็นนาที

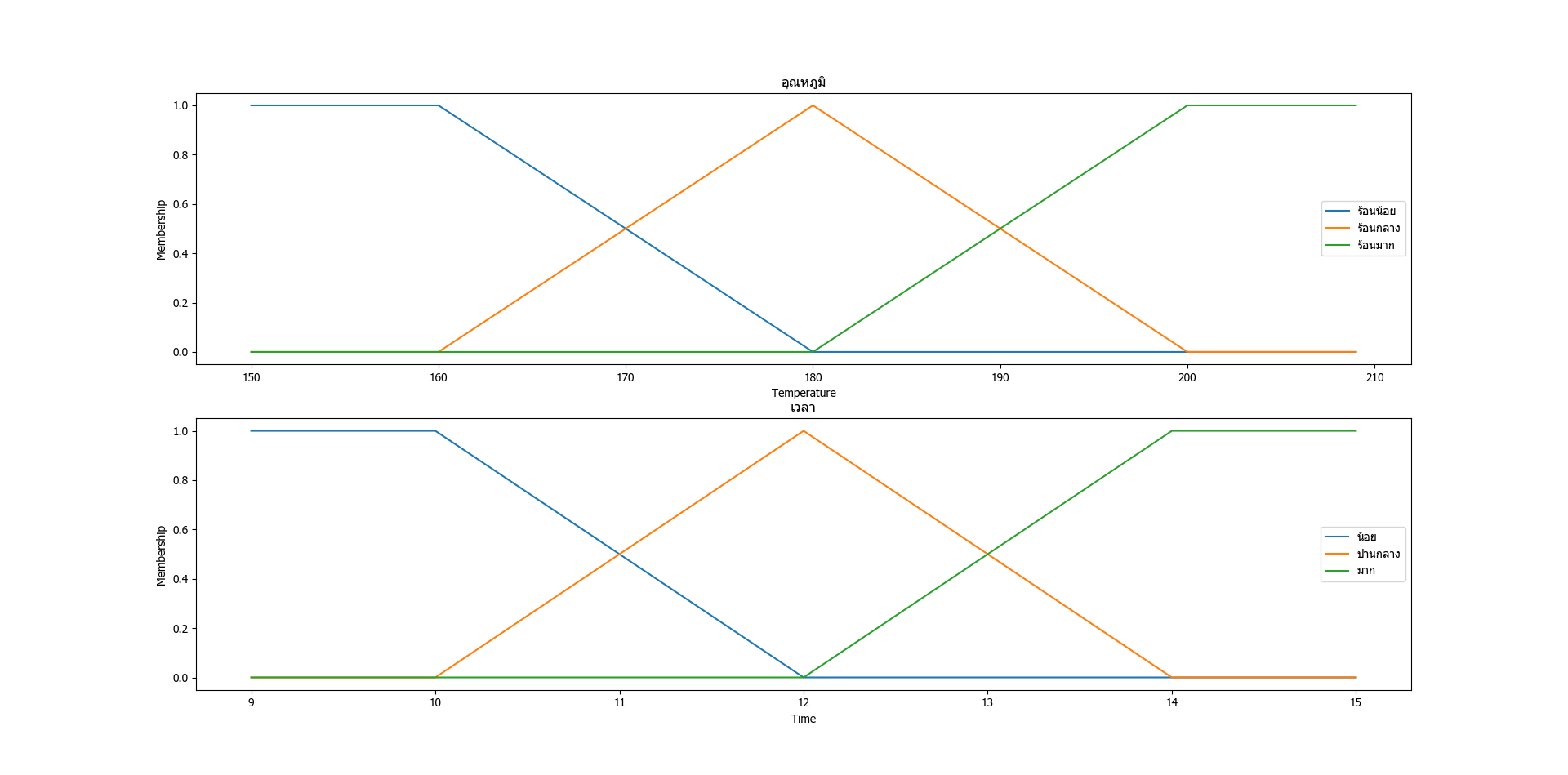
ตาราง 1 กฎแต่ละกฎใน Fuzzy Rules ที่จะได้ผลลัพธ์เป็นระดับความเข้มของเมล็ดกาแฟ

|  |  |  |  |
| --- | --- | --- | --- |
| Temperature (องศาเซล  เซียส)  Time (นาที) | ร้อนน้อย | ร้อนกลาง | ร้อนมาก |
| สั้น | คั่วอ่อน | คั่วกลาง | คั่วกลาง |
| ปานกลาง | คั่วอ่อน | คั่วกลาง | คั่วกลาง |
| นาน | คั่วกลาง | คั่วเข้ม | คั่วเข้ม |

**Rule**

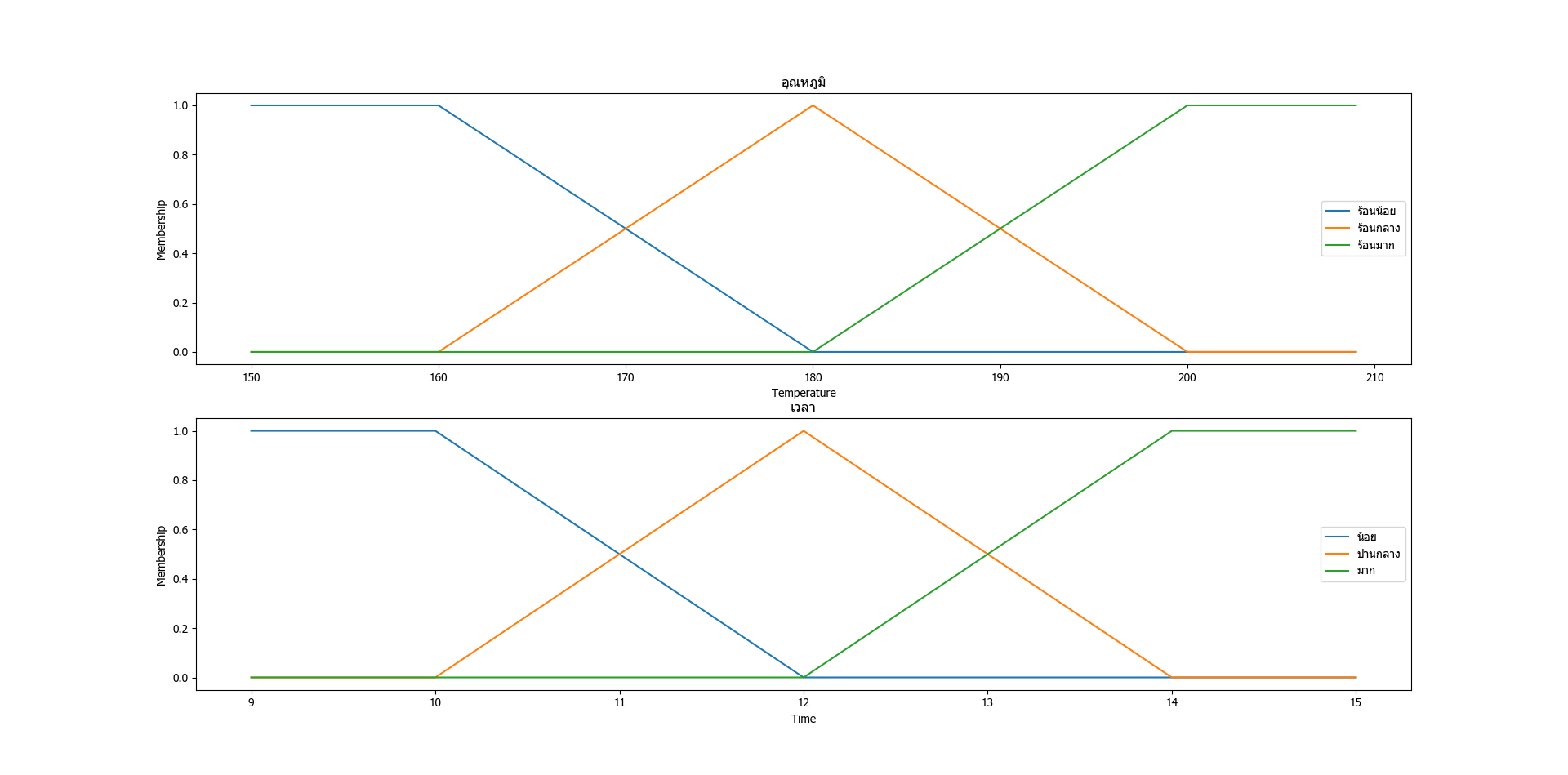
1. ถ้า Time is สั้น and Temperature is ร้อนน้อย then result is คั่วอ่อน
2. ถ้า Time is ปานกลาง and Temperature is ร้อนน้อย then result is คั่วอ่อน
3. ถ้า Time is สั้น and Temperature is ร้อนกลาง then result is คั่วกลาง
4. ถ้า Time is สั้น and Temperature is ร้อนมาก then result is คั่วกลาง
5. ถ้า Time is ปานกลาง and Temperature is ร้อนกลาง then result is คั่วกลาง
6. ถ้า Time is ปานกลาง and Temperature is ร้อนมาก then result is คั่วกลาง
7. ถ้า Time is นาน and Temperature is ร้อนน้อย then result is คั่วกลาง
8. ถ้า Time is นาน and Temperature is ร้อนปานกลาง then result is คั่วเข้ม
9. ถ้า Time is นาน and Temperature is ร้อนมาก then result is คั่วเข้ม

**Fuzzy Membership Function**

สมการ Membership Function ของอุณหภูมิดัง รูปที่ 1

รูปที่ 1 Membership Function ของอุณหภูมิ

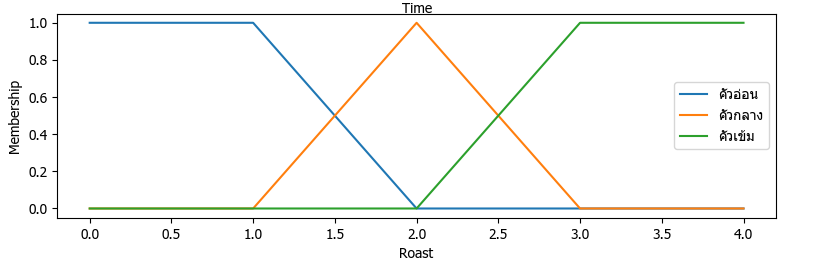
* ร้อนน้อย :
  + if x < 160 :
  + else if x >= 160 and x < 180 :
  + else :
* ร้อนกลาง :
  + if x == 180 :
  + else if x > 160 and x < 180 :
  + else if x > 180 and x < 200 :
  + else :
* ร้อนมาก :
  + if x > 200 :
  + else if x > 180 and x <= 200 :
  + else :

สมการ Membership Function ของเวลาดัง รูปที่ 2

รูปที่ 2 Membership Function ของเวลา

* น้อย :
  + if x < 10 :
  + else if x >= 10 and x < 12 :
  + else :
* ปานกลาง :
  + if x == 12 :
  + else if x > 10 and x < 12 :
  + else if x > 12 and x < 14 :
  + else :
* มาก :
  + if x < 12 :
  + else if x >= 12 and x < 14 :
  + else :

สมการ Membership Function ของระดับการคั่วกาแฟดัง รูปที่ 3



รูปที่ 3 Membership Function ของระดับการคั่วกาแฟ

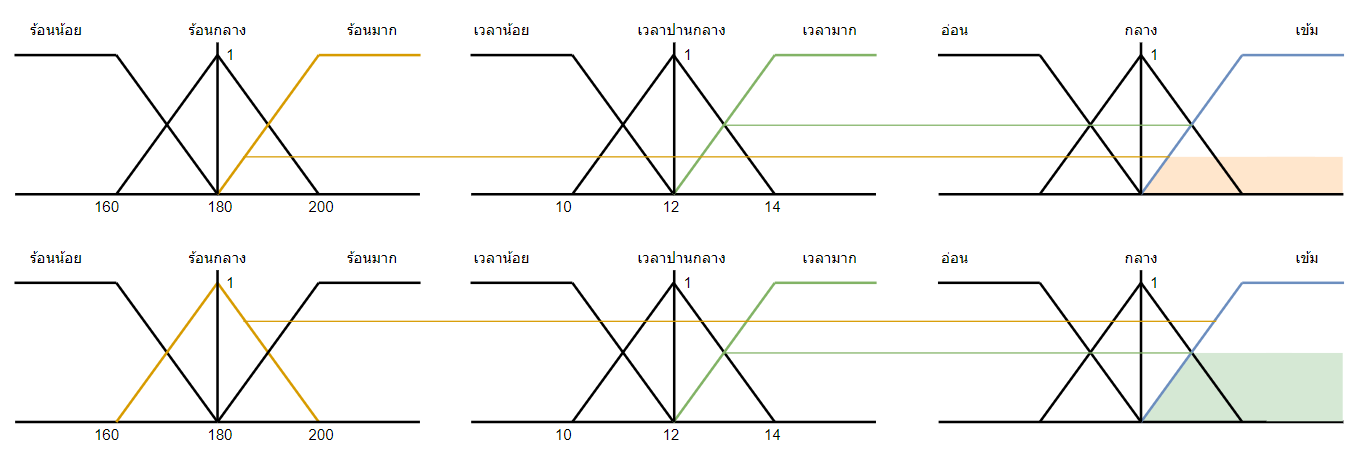
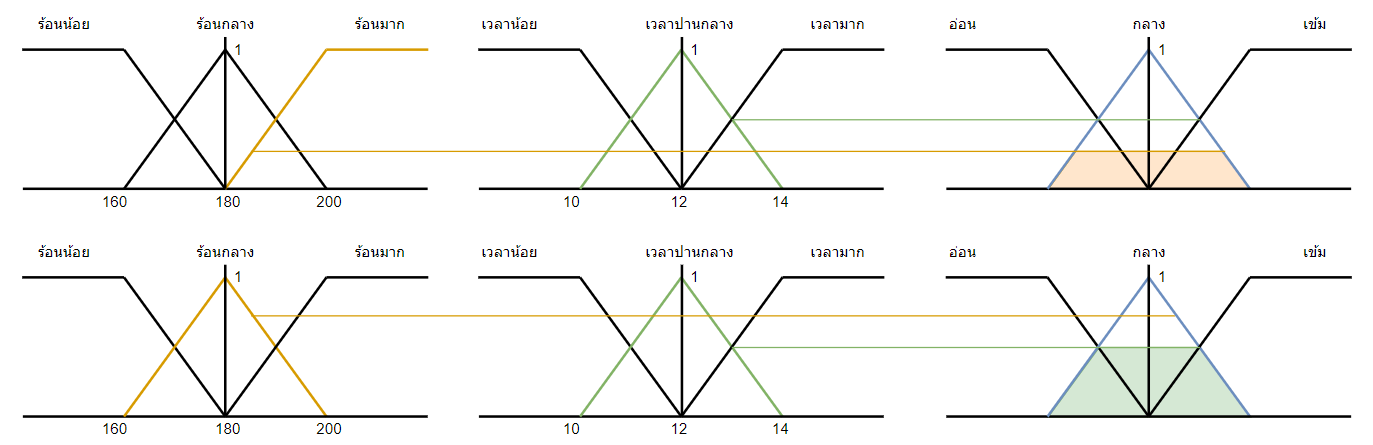
* คั่วอ่อน :
  + if x <= 1 :
  + else if x >= 1 and x < 2 :
  + else :
* ปานกลาง :
  + if x == 2 :
  + else if x > 2 and x < 3 or x < 2 and x > 1:
  + else :
* มาก :
  + if x <= 2 :
  + else if x > 2 and x < 3 :
  + else :

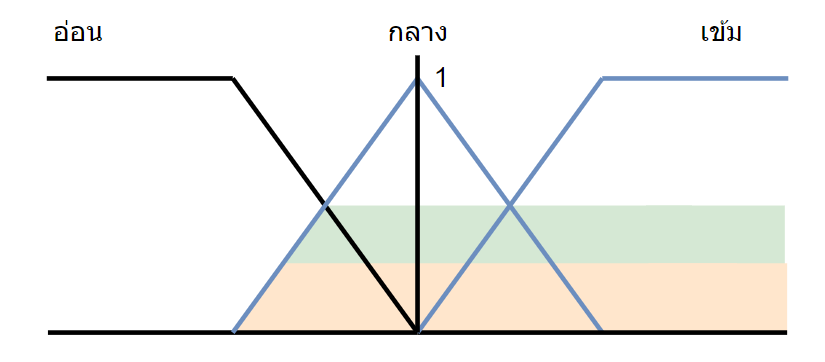
**Defuzzification**

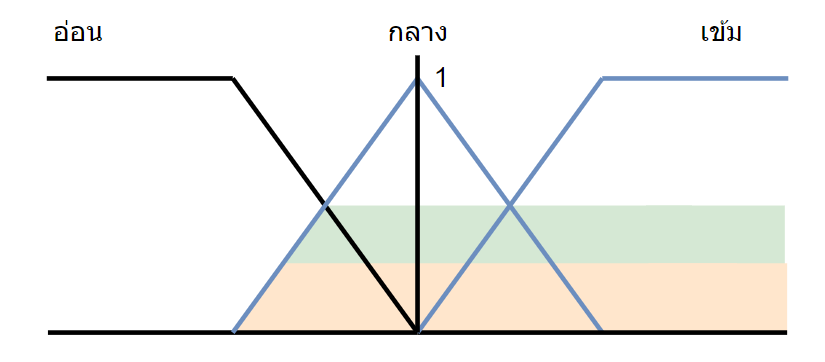
ใช้การคำนวณแบบ Center of Area (Centroid) โดยเลือกการใช้ ฐานนิยม (Mode) ในการคำนวณหา Defuzzification

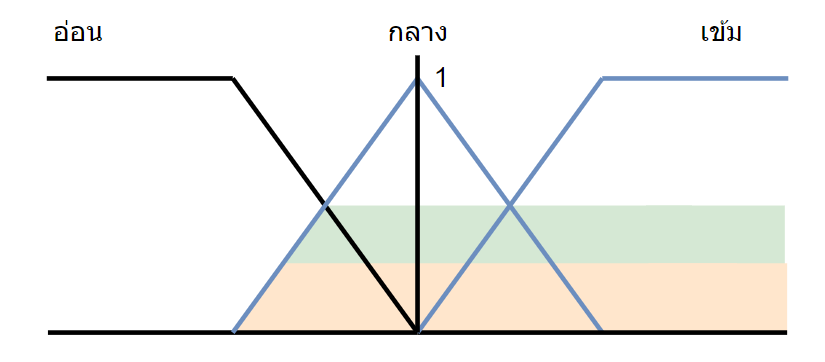
**การทดลอง**

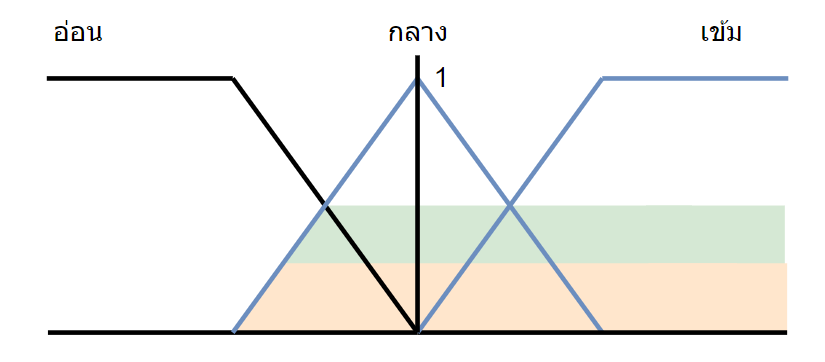
การทดลองนี้จะลองคำนวณบางส่วนด้วยการลองทำเอง เพื่อดูว่าผลจากโปรแกรมนั้นถูกต้องหรือเปล่า โดยที่ให้ Input อุณหภูมิเท่ากับ 187 และเวลาเท่ากับ 13

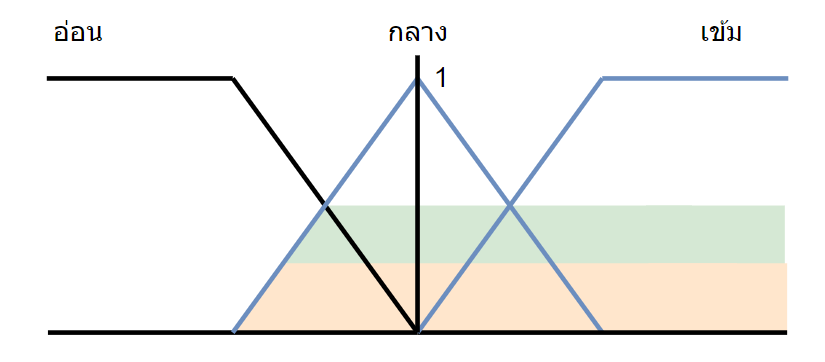


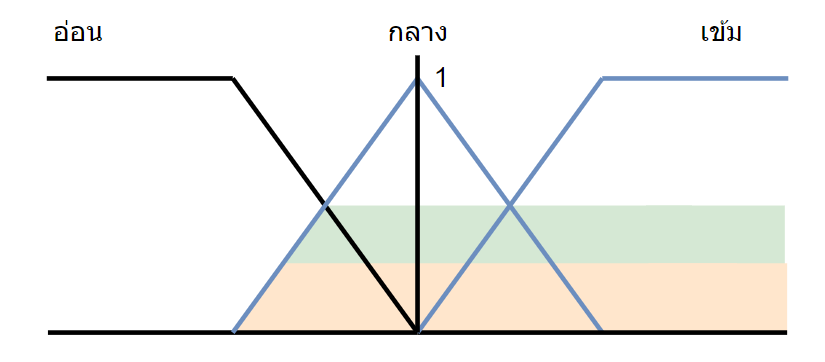
3 

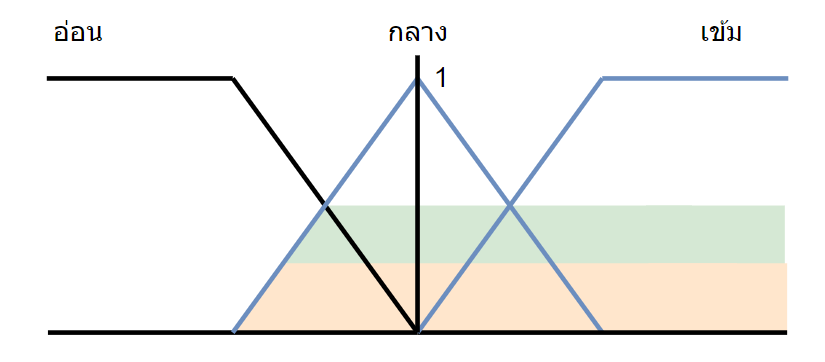
1 

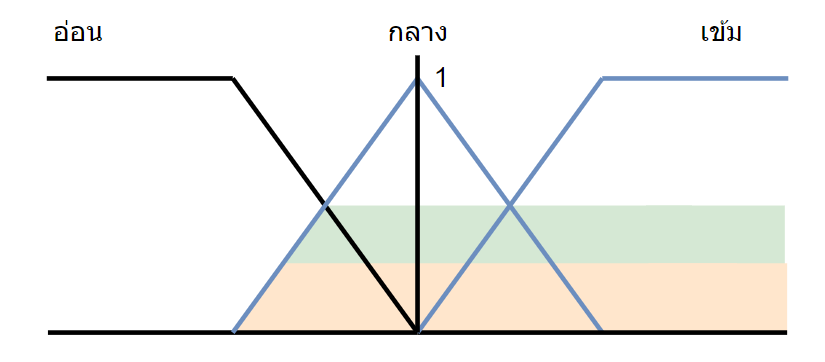
2 

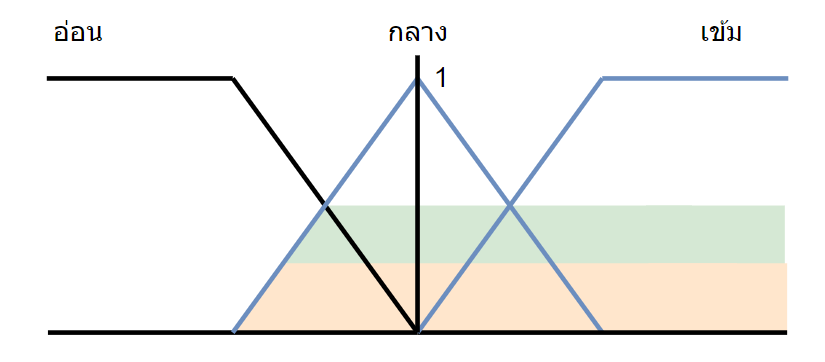
2 

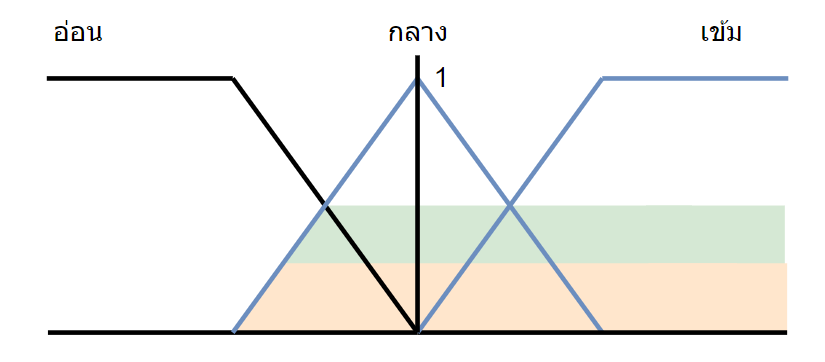
3 

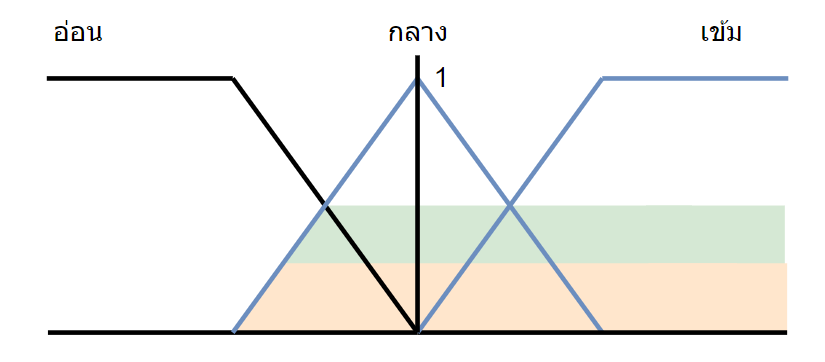
1 

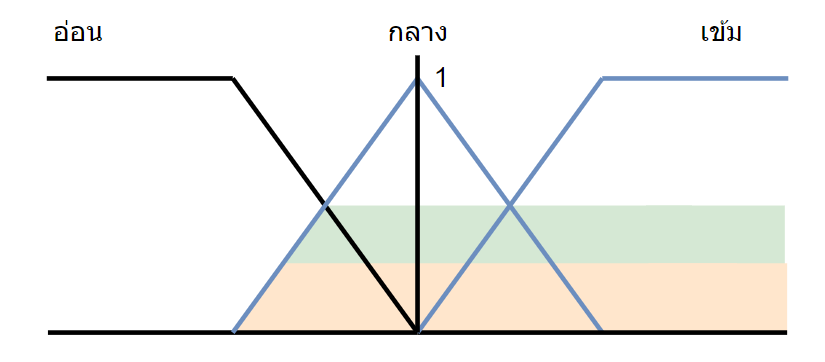
1 

3 

2 

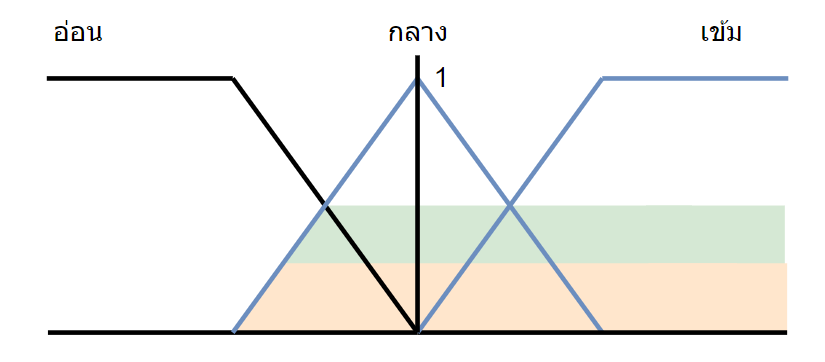
3 

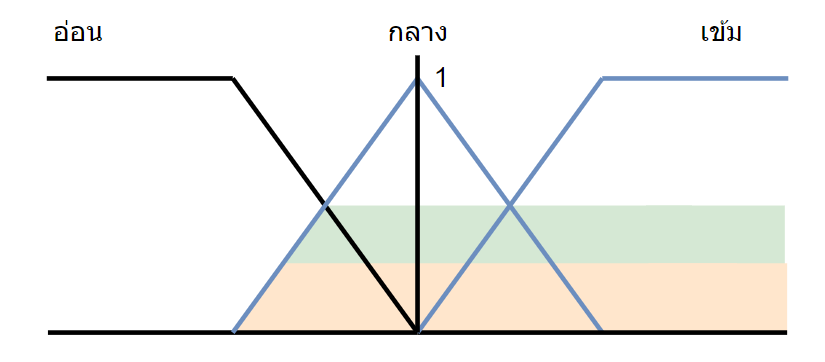
2 

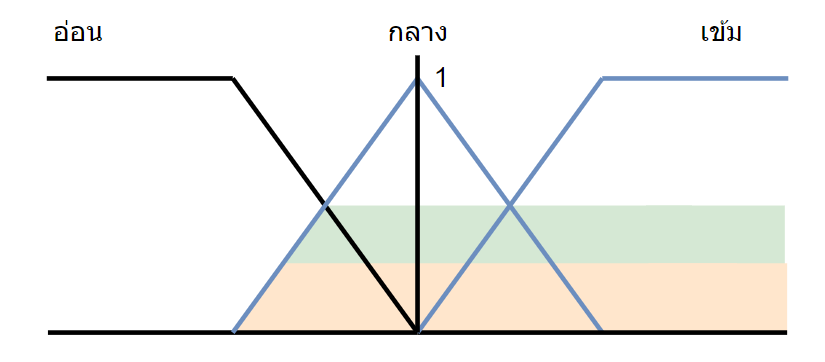
1 

รูปที่ 4 แสดงผลการหา min หรือ ในรูปจากการคำนวณด้วยตัวเอง

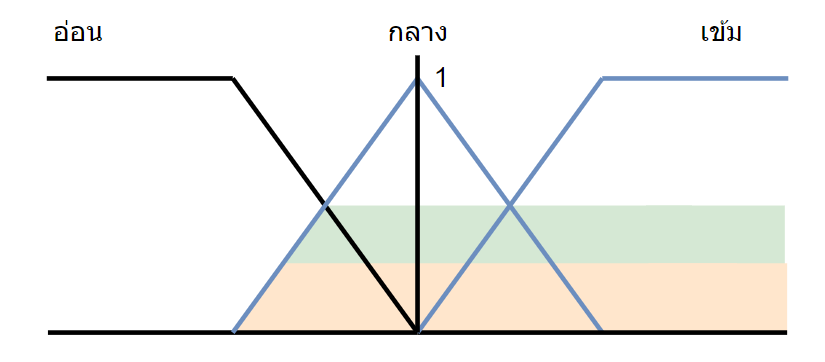
โดยจาก รูปที่ 4 จะเห็นได้ว่าจะเข้ากฎที่ 5, 6, 8, 9 จะให้ผลลัพธ์การหา จะได้ผลลัพธ์ดัง รูปที่ 4 และหลังนำแต่ละ Rule มารวมกันก็จะได้ดัง รูปที่ 5 ดังนั้นผลลัพธ์จะอยู่ในช่วง กลางจนถึงเข้ม แต่เราก็ไม่รู้ว่าจริงๆ แล้วต้องเป็นเท่าไหร่ ดังนั้นจึงต้องทำการ Defuzzification เพื่อจะให้รู้ว่าจะตอบอะไรกันแน่



3 

1 

2



รูปที่ 5 จากนำผลลัพธ์ของแต่ละ Rule มารวมกัน

ในส่วนต่อไปนี้ จะทำการ Defuzzification เพื่อหาคำตอบของระบบนี้ โดยใช้โปรแกรมและได้ใช้การ Defuzzification แบบ Centroid เลือกคำนวณแบบ Mode

Input = [187, 13]

Medium Roast: 41.18 %

รูปที่ 6 แสดงค่า Input ที่ใส่เข้าไปในระบบและได้ผลลัพธ์

จะเห็นได้ว่าผลลัพธ์ของระบบนั้นอยู่ในช่วงที่ถูกต้องที่คาดการณ์ไว้คือช่วงกลางถึงเข้ม หรือ 0 ถึง 2

การทดลองต่อไปนี้ จะทำการสุ่มค่าของ Input โดยที่ให้ค่าอุณหภูมินั้นอยู่ในช่วง 150 ถึง 250 องศาเซลเซียส และให้ เวลานั้น อยู่ในช่วง 8 ถึง 10 นาที

Input = [154, 14]

Medium Roast

--------------------

Input = [208, 17]

Dark Roast

--------------------

Input = [183, 13]

Medium Roast: 23.08 %

--------------------

Input = [236, 15]

Dark Roast

--------------------

Input = [157, 14]

Medium Roast

--------------------

Input = [154, 8]

Light Roast

--------------------

Input = [162, 12]

Light Roast: 10.00 %

--------------------

Input = [196, 10]

Medium Roast

--------------------

Input = [199, 11]

Medium Roast: 45.45 %

--------------------

Input = [176, 13]

Light Roast: 85.71 %

--------------------

รูปที่ 7 ผลลัพธ์ของการสุ่มค่าอุณหภูมิ, เวลาและผลลัพธ์หลังจากการนำไปเข้าระบบ

จาก รูปที่ 7 จะเห็นได้ว่า Input ที่ทำการสุ่มขึ้นมานั้นตรงกับกฎที่ได้ตั้งเอาไว้ จึงได้ผลลัพธ์ที่ถูกต้องตามที่คาดหวังไว้

**สรุปผลการทดลอง**

การทดลองการเขียนโปรแกรมจำลองการคั่วเมล็ดกาแฟ โดยการใช้ Fuzzy logic โดยใช้วิธ๊ Mamdani ให้การทำโดยมีกฎทั้งหมด 9 ข้อ ที่ได้สร้างขึ้นมาเพื่อควบคุมระบบนี้ โดยการคั่วเมล็ดกาแฟนี้จะแบ่งให้มี 3 ระดับคือ 1). คั่วอ่อน 2). คั่วกลาง 3). คั่วเข้ม ซึ่งระบบที่ได้ทำขึ้นมานั้น ให้ผลลัพธ์ที่ถูกต้องตามกฎที่ได้ตั้งไว้

**ภาคผนวก**

import matplotlib.pyplot as plt

import numpy as np

import random

plt.rcParams['font.family'] = 'Tahoma'

#### Input

inpu = [187, 13]

xlist = range(150, 250)

tlist = range(9, 16)

graph = 0

def main():

    if graph :

        graphrule()

    for i in range(10):

        inpu = []

        inpu.append(random.randrange(150, 250))

        inpu.append(random.randrange(8, 20))

        print("Input =", inpu)

        minrule = minOfrule(inpu)

        # print("Min Rule =", minrule)

        mamdani(minrule)

        print("--------------------")

##### Rule

def temperature10(i, down = 160):

    ran = 20

    if i < down :

        membership = 1

    elif i >= down and i < down + ran :

        membership = i \*(-1/ran) + (1 + down/ran)

    else :

        membership = 0

    return membership

def temperature11(i, mid = 180):

    ran = 20

    if i == mid :

        membership = 1

    elif i > mid - ran and i < mid :

        membership = i \*(1/ran) + (1 - mid/ran)

    elif i > mid and i < mid + ran  :

        membership = i \*(-1/ran) + (1 + mid/ran)

    else :

        membership = 0

    return membership

def temperature12(i, up = 200):

    ran = 20

    if i > up :

        membership = 1

    elif i > up - ran and i <= up :

        membership = i \*(1/ran) + (1 - up/ran)

    else :

        membership = 0

    return membership

def time10(i, down = 10):

    ran = 2

    if i < down :

        y = 1

    elif i >= down and i < down + ran :

        y = (down + ran - i)\*(1/ran)

    else :

        y = 0

    return y

def time11(i, mid = 12):

    ran = 2

    if i == mid :

        y = 1

    elif i > mid - ran and i < mid :

        y = i \* (1/ran) - mid/ran + 1

    elif i > mid and i < (mid + ran) :

        y = i \* (1/ran) - mid/ran

    else :

        y = 0

    return y

def time12(i, up = 14):

    ran = 2

    if i < up - ran :

        y = 0

    elif i >= up - ran and i < up :

        y = i/(ran) - (up/ran - 1)

    else :

        y = 1

    return y

def roast00(i, down = 1):

    ran = 1

    if i <= down :

        membership = 1

    elif i > down and i < down + ran :

        membership = ((1)/(-ran))\*i

    else :

        membership = 0

    return membership

def roast01(i, mid = 2):

    ran = 1

    if i == mid :

        membership = 1

    elif i > mid and i < mid + ran or i < mid and i > mid - 1 :

        membership = ((1)/(-ran))\*i

    else :

        membership = 0

    return membership

def roast02(i, up = 2):

    ran = 1

    if i <= up :

        membership = 0

    elif i > up and i < up + ran :

        membership = ((1)/(up - ran))\*i

    else :

        membership = 1

    return membership

#### Find min

def minOfrule(inpu): # Find Alpha cut return listmin

    Listmin = []

    L1 = min(temperature10(inpu[0]), time10(inpu[1]))

    L2 = min(temperature10(inpu[0]), time11(inpu[1]))

    M1 = min(temperature11(inpu[0]), time10(inpu[1]))

    M2 = min(temperature12(inpu[0]), time10(inpu[1]))

    M3 = min(temperature11(inpu[0]), time11(inpu[1]))

    M4 = min(temperature10(inpu[0]), time12(inpu[1]))

    M5 = min(temperature11(inpu[0]), time12(inpu[1]))

    D1 = min(temperature12(inpu[0]), time11(inpu[1]))

    D2 = min(temperature12(inpu[0]), time12(inpu[1]))

    Listmin.append([L1, L2])

    Listmin.append([M1, M2, M3, M4, M5])

    Listmin.append([D1, D2])

    return Listmin

#### Mamdani

def mamdani(minOfrule):

    alpha = minOfrule

    maxlevel = []

    # print(alpha)

    for i in range(len(alpha)):

        maxlevel.append(max(alpha[i]))

    # print(maxlevel)

    ###### Simplified Centroid

    defuzzi = np.zeros(2)

    # print(len(alpha))

    for i in range(len(alpha)):

        for j in range(len(alpha[i])):

            defuzzi[0] += alpha[i][j]\*(i+1)

            defuzzi[1] += alpha[i][j]

    defuzzify = defuzzi[0]/defuzzi[1]

    if ( defuzzify >= 1 and defuzzify < 2):

        if defuzzify - 1 != 0 :

            print("Light Roast: {:.2f} %" .format((defuzzify - 1)\*100))

        else :

            print("Light Roast")

    elif ( defuzzify >= 2 and defuzzify < 3 ) :

        if defuzzify - 2 != 0 :

            print("Medium Roast: {:.2f} %" .format((defuzzify - 2)\*100))

        else :

            print("Medium Roast")

    elif ( defuzzify >= 3 ) :

        if defuzzify - 3 != 0 :

            print("Dark Roast: {:.2f} %" .format((defuzzify - 3)\*100))

        else :

            print("Dark Roast")

##### Graph Rule

def graphrule():

    tem1 = [temperature10(x) for x in xlist]

    tem2 = [temperature11(x) for x in xlist]

    tem3 = [temperature12(x) for x in xlist]

    tim = [time10(t, 10) for t in tlist]

    tim1 = [time11(t, 12) for t in tlist]

    tim2 = [time12(t, 14) for t in tlist]

    roaslist = range(0,5)

    roas = [roast00(x) for x in roaslist]

    roas1 = [roast01(x) for x in roaslist]

    roas2 = [roast02(x) for x in roaslist]

    # roas = [roast00(x) for t in roaslist]

    fig, axs = plt.subplots(2, 1)

    axs[0].set\_title(label=u"อุณหภูมิ")

    axs[0].plot(xlist, tem1, label=u'ร้อนน้อย')

    axs[0].plot(xlist, tem2, label=u'ร้อนกลาง')

    axs[0].plot(xlist, tem3, label=u'ร้อนมาก')

    axs[0].set\_xlabel('Temperature')

    axs[0].set\_ylabel('Membership')

    axs[0].legend()

    axs[1].set\_title(label=u"เวลา")

    axs[1].plot(tlist, tim, label=u'น้อย')

    axs[1].plot(tlist, tim1, label=u'ปานกลาง')

    axs[1].plot(tlist, tim2, label=u'มาก')

    axs[1].set\_xlabel('Time')

    axs[1].set\_ylabel('Membership')

    axs[1].legend()

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    main()