CPE 261456 Introduction to Computational Intelligence

Computer Assignment 2 (Fuzzy logic)

จัดทำโดย

นาย ณัชพล เพทายเทียมทอง

รหัสนักศึกษา 580610633

เสนอ

รศ.ดร. ศันสนีย์ เอื้อพันธ์วิริยะกุล

ภาควิชา วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเชียงใหม่ ปีการศึกษา 2561

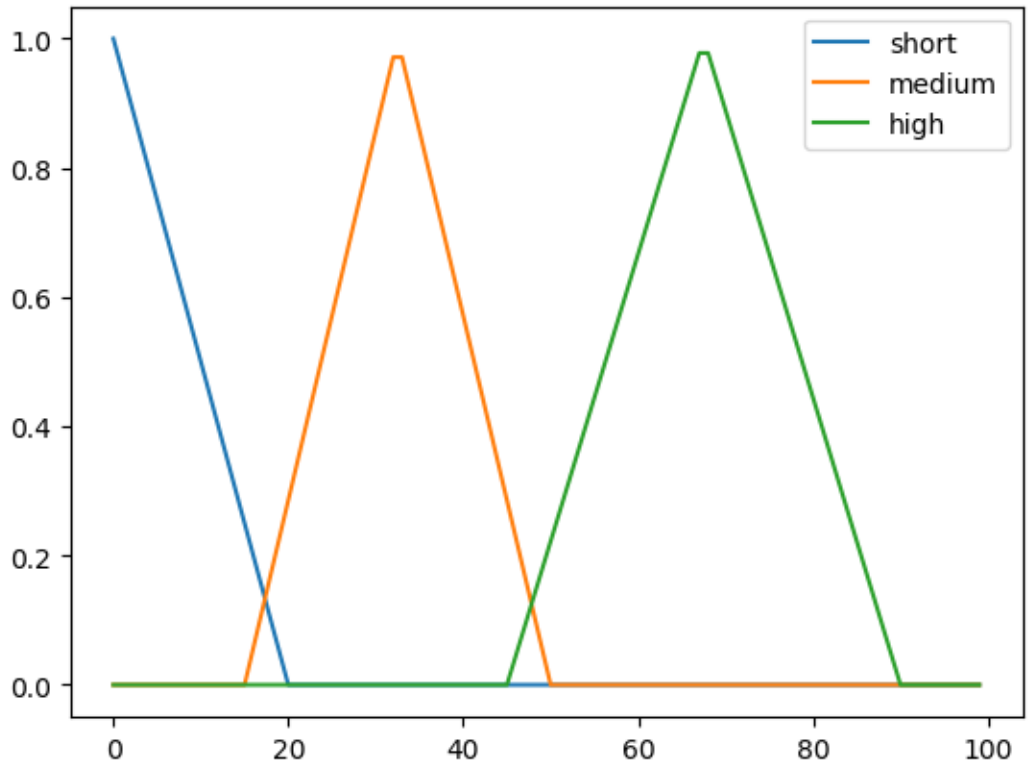
Fuzzy Logic

เตาอบ Bakery

ขั้นตอนการทำงาน

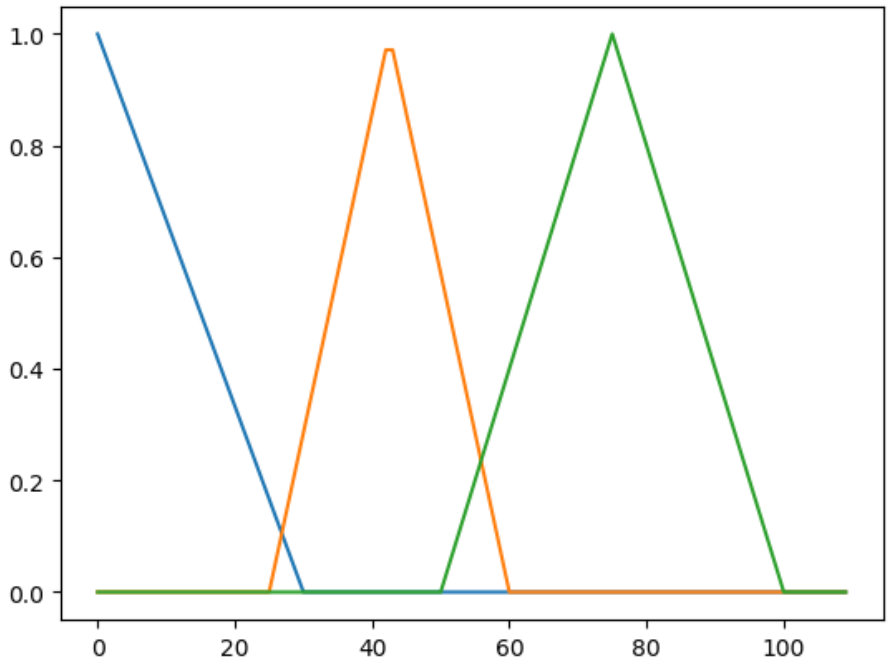
ในการทดลองนี้จะสมมติว่า เตาอบ Bakery นั้นจะพิจารณาองค์ประกอบ 3 อย่างด้วยกัน ได้แก่ ขนาดของขนมที่ต้องการอบ ชนิดของแป้งว่าขนมนั้นแป้งจะต้องมีลักษณะอย่างไร และเวลาที่ใช้ในการอบ เพื่อที่จะได้ปรับความแรงของไฟให้เหมาะสมกับองค์ประกอบของการอบขนมนั้นๆ โดย องค์ประกอบแต่ละอย่างจะแบ่งได้ดังนี้

1. เวลา จะแบ่งได้เป็น เวลาสั้น เวลาปานกลาง เวลานาน มีกราฟดังรูปที่ 1



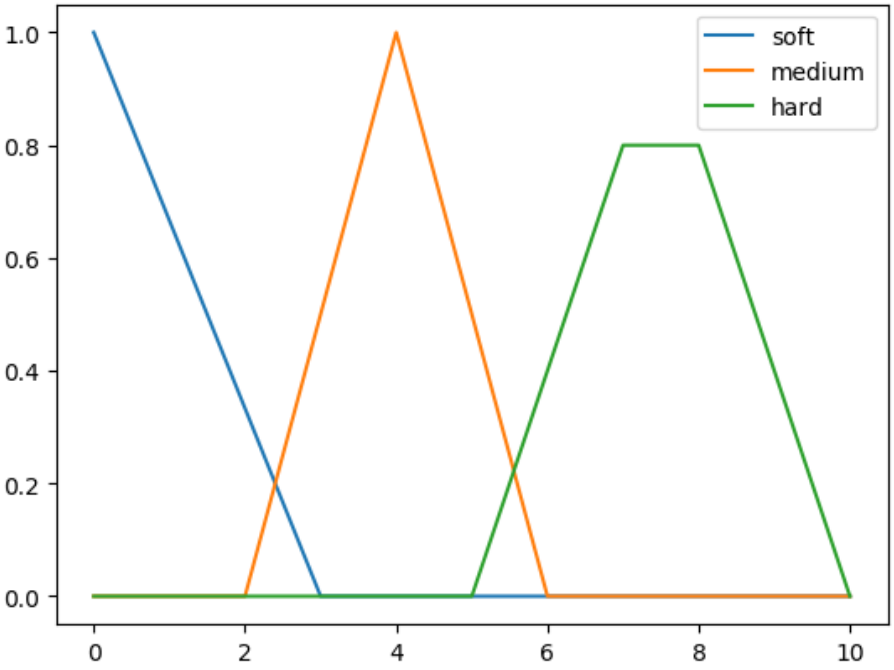
รูปที่ 1 กราฟของเวลา

2. ขนาดของขนม จะแบ่งได้เป็น ขนาดเล็ก ขนาดกลาง ขนาดใหญ่ ดังรูปที่ 2



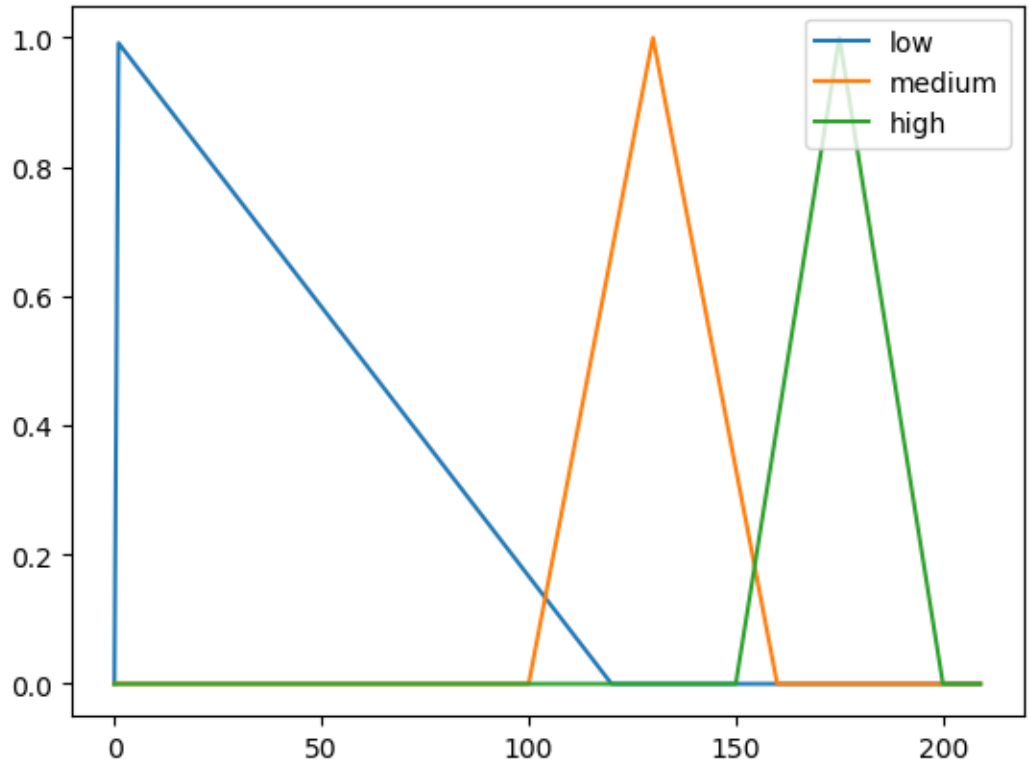
รูปที่ 2 กราฟของขนาดขนม

3. ชนิดของแป้ง จะแบ่งได้เป็น แป้งนุ่ม แป้งแข็งปานกลาง แป้งแข็ง ดังรูปที่ 3



รูปที่ 3 กราฟของชนิดแป้ง

ดังนั้นผลลัพธ์จะออกมาเป็นความแรงของไฟ เป็น ไฟเบา ไฟกลาง ไฟสูง ดังรูปที่ 4



รูปที่ 4 กราฟความแรงไฟ

ดังนั้นองค์ประกอบทั้ง 3 ที่เป็น input โดยแต่ละ input มี 3 อย่างดังนั้น Rule ที่ได้จะมี 27 Rule

ดังนั้น Rule มีดังนี้

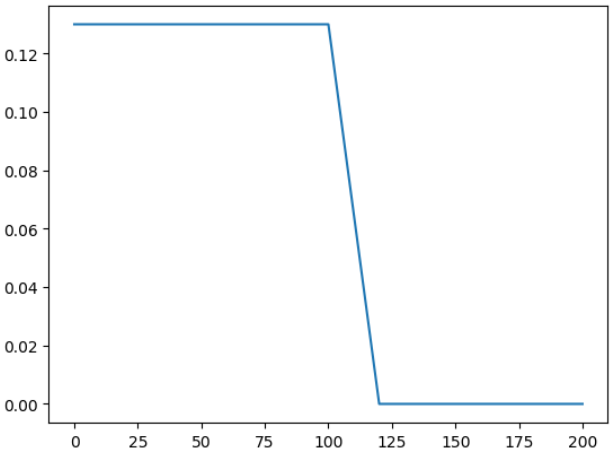
1. **IF** เวลาสั้น, ขนาดเล็ก, แป้งนุ่ม **Then** ไฟเบา
2. **IF** เวลาสั้น, ขนาดเล็ก, แป้งแข็งปานกลาง **Then** ไฟกลาง
3. **IF** เวลาสั้น, ขนาดเล็ก, แป้งแข็ง **Then** ไฟสูง
4. **IF** เวลาสั้น, ขนาดกลาง, แป้งนุ่ม **Then** ไฟกลาง

5. **IF** เวลาสั้น, ขนาดกลาง, แป้งแข็งปานกลาง **Then** ไฟกลาง
6. **IF** เวลาสั้น, ขนาดกลาง, แป้งแข็ง **Then** ไฟสูง
7. **IF** เวลาสั้น, ขนาดใหญ่, แป้งนุ่ม **Then** ไฟกลาง
8. **IF** เวลาสั้น, ขนาดใหญ่, แป้งแข็งปานกลาง **Then** ไฟสูง
9. **IF** เวลาสั้น, ขนาดใหญ่, แป้งแข็ง **Then** ไฟสูง
10. **IF** เวลาปานกลาง, ขนาดเล็ก, แป้งนุ่ม **Then** ไฟเบา
11. **IF** เวลาปานกลาง, ขนาดเล็ก, แป้งแข็งปานกลาง **Then** ไฟเบา
12. **IF** เวลาปานกลาง, ขนาดเล็ก, แป้งแข็ง **Then** ไฟกลาง
13. **IF** เวลาปานกลาง, ขนาดปานกลาง, แป้งนุ่ม **Then** ไฟกลาง
14. **IF** เวลาปานกลาง, ขนาดปานกลาง, แป้งแข็งปานกลาง **Then** ไฟกลาง
15. **IF** เวลาปานกลาง, ขนาดปานกลาง, แป้งแข็ง **Then** ไฟสูง
16. **IF** เวลาปานกลาง, ขนาดใหญ่, แป้งนุ่ม **Then** ไฟกลาง
17. **IF** เวลาปานกลาง, ขนาดใหญ่, แป้งแข็งปานกลาง **Then** ไฟสูง
18. **IF** เวลาปานกลาง, ขนาดใหญ่, แป้งแข็ง **Then** ไฟสูง
19. **IF** เวลานาน, ขนาดเล็ก, แป้งนุ่ม **Then** ไฟเบา
20. **IF** เวลานาน, ขนาดเล็ก, แป้งแข็งปานกลาง **Then** ไฟเบา
21. **IF** เวลานาน, ขนาดเล็ก, แป้งแข็ง **Then** ไฟเบา
22. **IF** เวลานาน, ขนาดปานกลาง, แป้งนุ่ม **Then** ไฟเบา
23. **IF** เวลานาน, ขนาดปานกลาง, แป้งแข็งปานกลาง **Then** ไฟกลาง
24. **IF** เวลานาน, ขนาดปานกลาง, แป้งแข็ง **Then** ไฟกลาง
25. **IF** เวลานาน, ขนาดใหญ่, แป้งนุ่ม **Then** ไฟกลาง
26. **IF** เวลานาน, ขนาดใหญ่, แป้งแข็งปานกลาง **Then** ไฟกลาง
27. **IF** เวลานาน, ขนาดใหญ่, แป้งแข็ง **Then** ไฟสูง

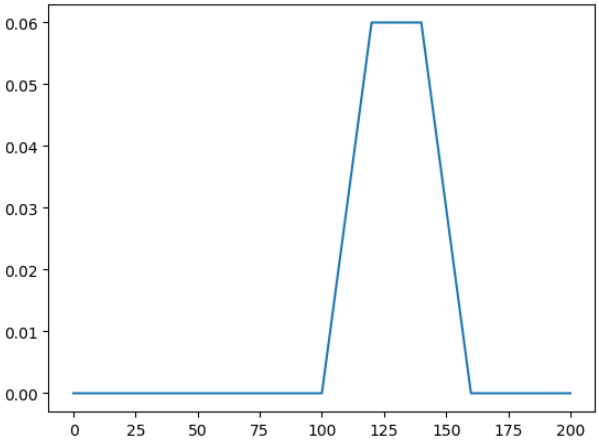
การทดลอง

การทดลองที่ 1 เวลา = 60, ขนาด = 26, แป้ง = 6

คำตอบ ต้องเปิดไฟ 60.67 องศาเซลเซียส และ ตรงกฎที่ 21 และ กฎที่ 24

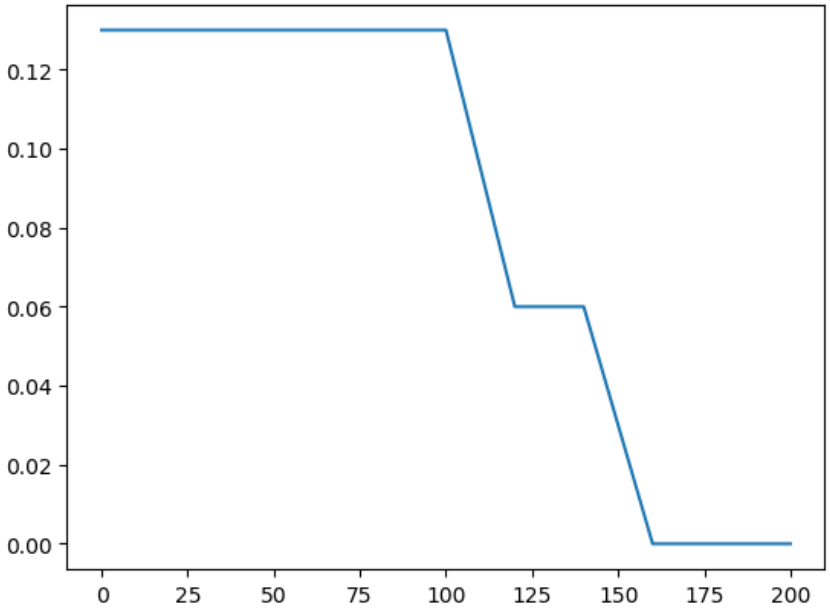


กฎที่ 21



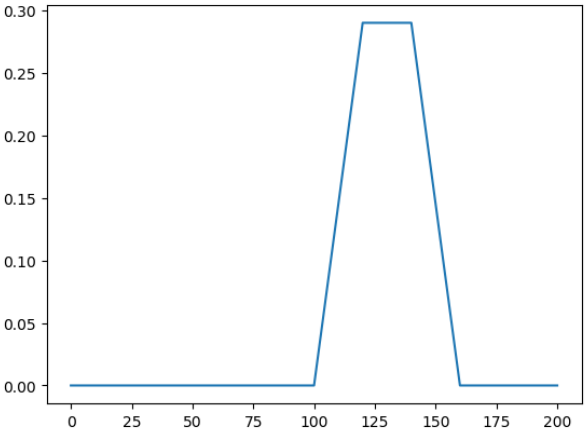
กฎที่ 24

กราฟผลลัพธ์

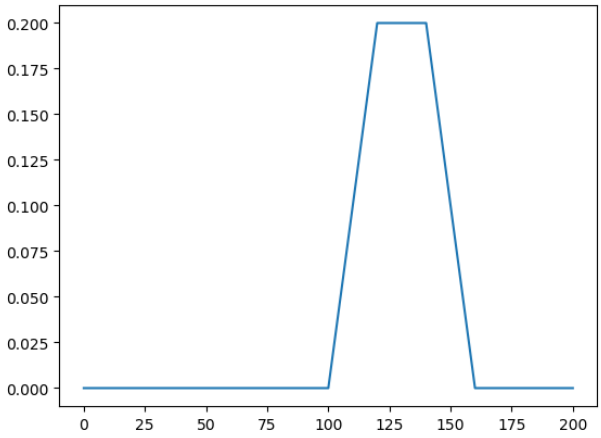


การทดลองที่ 2 เวลา = 80, ขนาด = 55, แป้ง = 3

คำตอบ ต้องเปิดไฟ 130 องศาเซลเซียส และ ตรงกฎที่ 23 และ กฎที่ 26

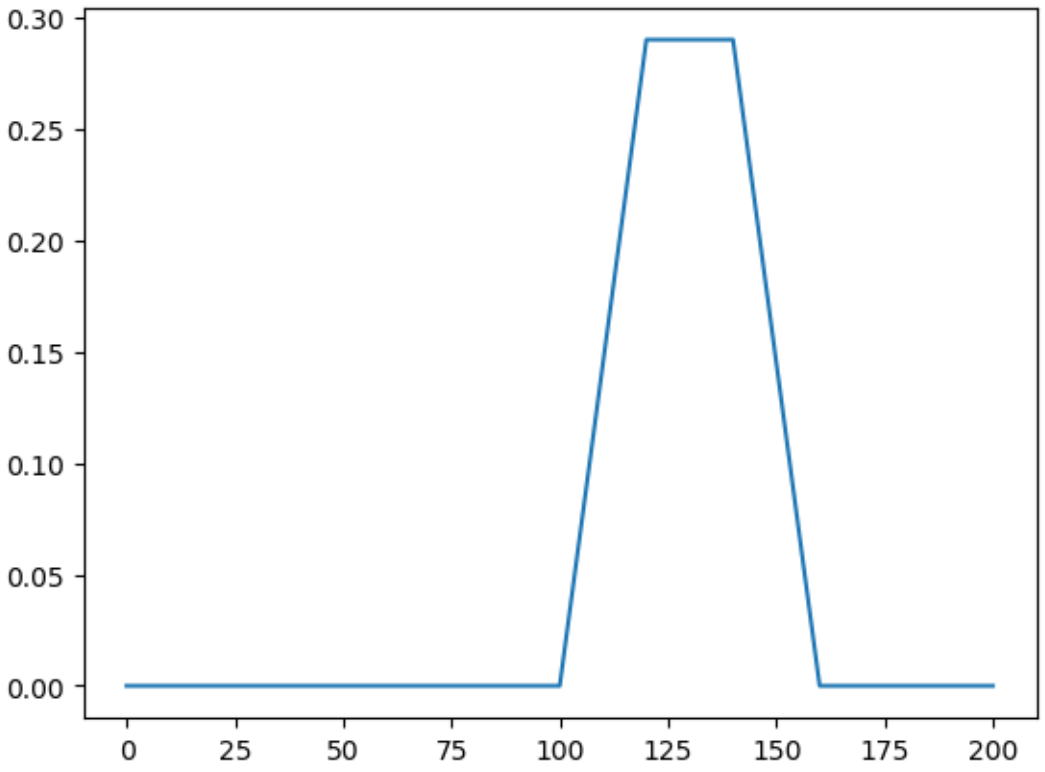


กฎที่ 23



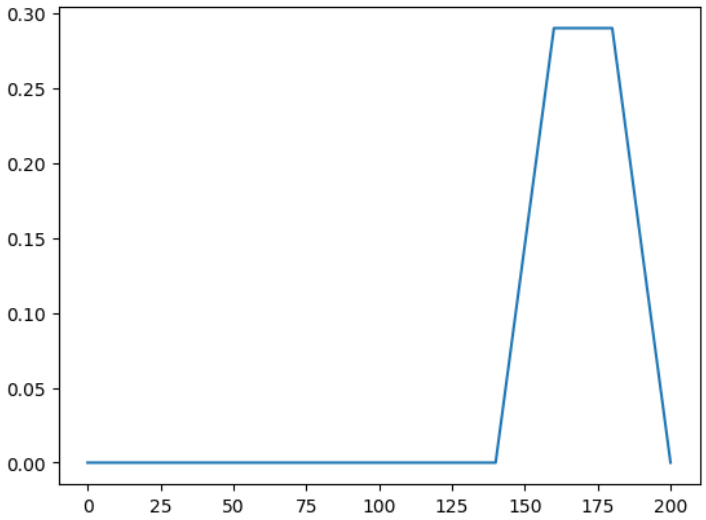
กฎที่ 26

กราฟผลลัพธ์



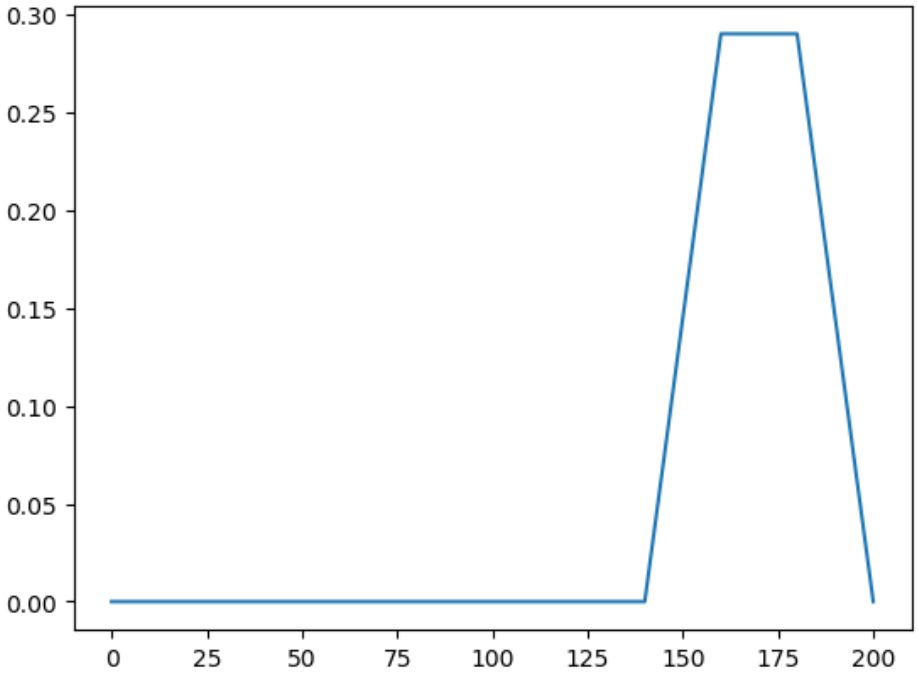
การทดลองที่ 3 เวลา = 20, ขนาด = 45, แป้ง = 8

คำตอบ ต้องเปิดไฟ 170 องศาเซลเซียส และ ตรงกฎที่ 15



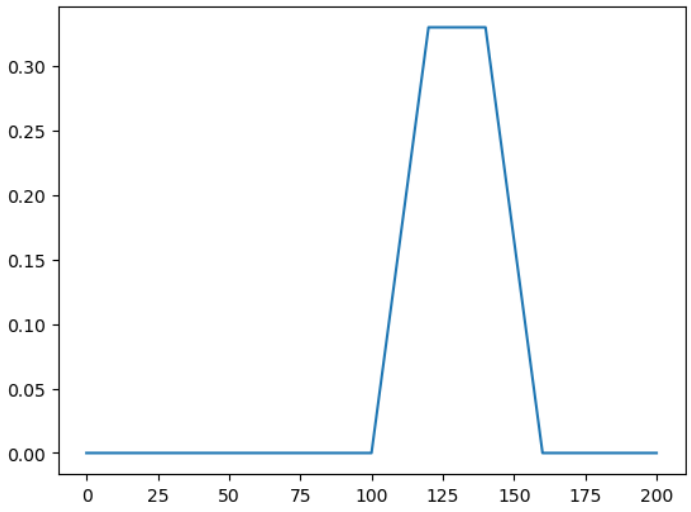
กฎที่ 15

กราฟผลลัพธ์



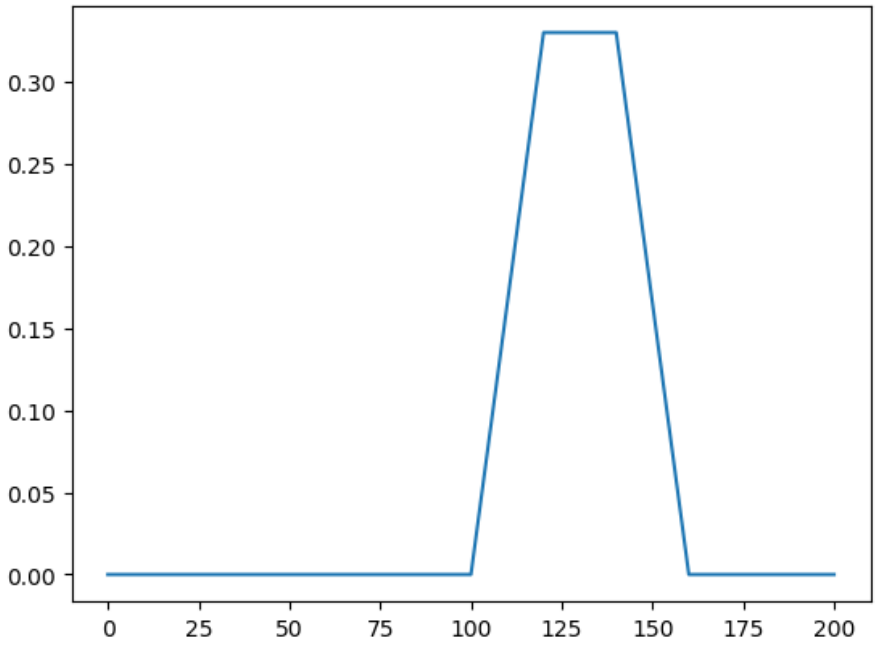
การทดลองที่ 4 เวลา = 60, ขนาด = 70, แป้ง = 2

คำตอบ ต้องเปิดไฟ 130 องศาเซลเซียส และ ตรงกฎที่ 25



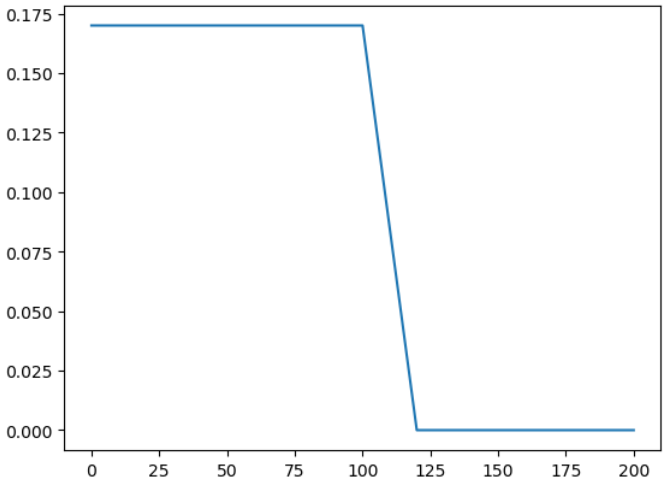
กฎที่ 25

กราฟผลลัพธ์



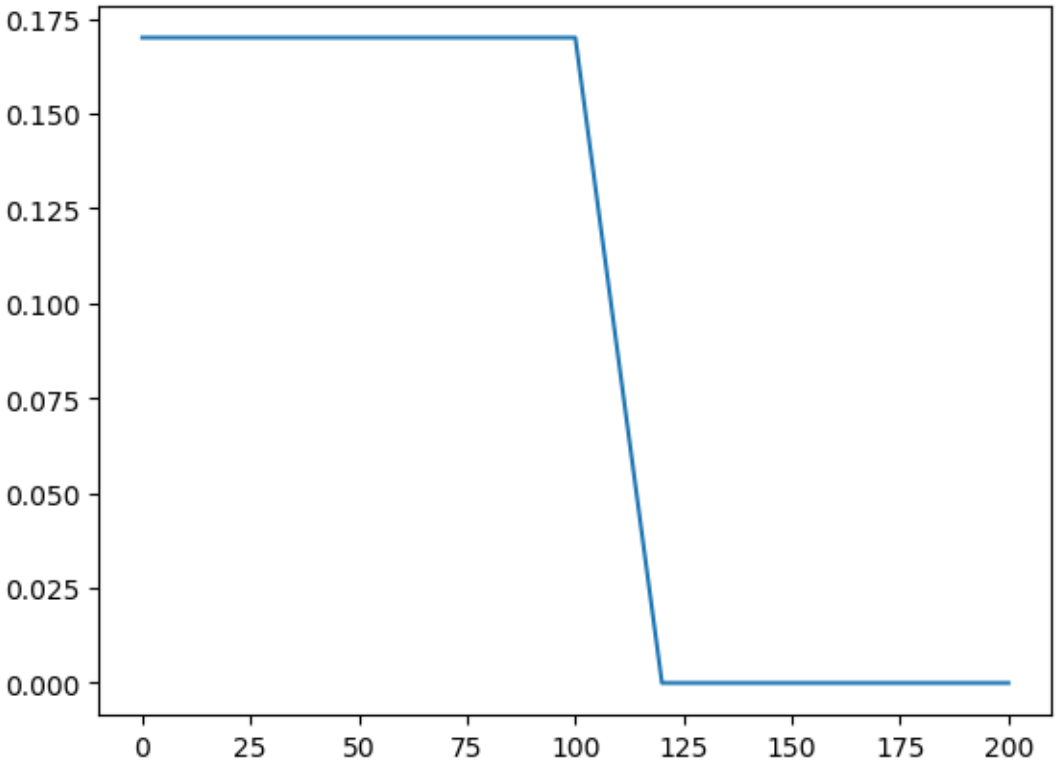
การทดลองที่ 5 เวลา = 15, ขนาด = 25, แป้ง = 2

คำตอบ ต้องเปิดไฟ 50 องศาเซลเซียส และ ตรงกฎที่ 1



กฎที่ 1

กราฟผลลัพธ์



วิเคราะห์ผลการทดลอง

จากการทดลองจะเห็นได้ว่า ความแรงของไฟนั้นมีแปรผันตรงกับขนาดของขนม ชนิดแป้งแต่แปรผกผัน กับระยะของเวลาของการอบ จากการทดลองที่ 1 จะเห็นได้ว่า เกิดการรวมกันของทั้ง 2 กฎซึ่งสามารถหาความแรงไฟของทั้ง 2 กฎได้ โดยความแรงไฟที่ได้มานั้นสมเหตุสมผลก็ความเป็นจริงซึ่ง การใช้ fuzzy โดยกฎตาม expert นั้นได้คำตอบที่เหมาะสม

ภาคผนวก

Code ส่วนของ fire.py

```
import math
import matplotlib.pyplot as plt

def low(value, acut):
    if 0 <= value <= 120:
        membership = ((-1/120) * value) + 1
        if membership > acut:
            membership = acut
    else:
        membership = 0

    return membership


def medium(value, acut):
    if 100 <= value <= 160:
        membership = 1 - math.fabs((value - 130) / 30)
        if membership > acut:
            membership = acut
    else:
        membership = 0

    return membership


def high(value, acut):
    if 150 <= value <= 200:
        membership = 1 - math.fabs((value - 175) / 25)
        if membership > acut:
            membership = acut
    else:
        membership = 0

    return membership

def union(x, y):
    out = []
    for i in range(0, len(x)):
        ax = max(x[i], y[i])
        out.append(ax)
```

```
    return out

y1 = []
y2 = []
y3 = []
x1 = []
plt.figure(90)
for x in range(0, 210):
    y1.append(low(x,x))
    y2.append(medium(x,x))
    y3.append(high(x,x))
    x1.append(x)

plt.plot(x1, y1)
plt.plot(x1, y2)
plt.plot(x1, y3)
plt.legend(('low', 'medium', 'high'), loc='upper right')
plt.savefig("fire.png", bbox_inches='tight')
```

code ส่วนของ baketime.py

```
import math
import matplotlib.pyplot as plt

def short(value):
    if 0 <= value <= 20:
        membership = ((-1/20) * value) + 1
    else:
        membership = 0

    return membership


def medium(value):
    if 15 <= value <= 50:
        membership = 1 - math.fabs((value - 32.5) / 17.5)
    else:
        membership = 0

    return membership


def high(value):
    if 45 <= value <= 90:
        membership = 1 - math.fabs((value - 67.5) / 22.5)
    else:
        membership = 0

    return membership
```

```
y1 = []
y2 = []
y3 = []
x1 = []
plt.figure(91)
for x in range(0, 100):
    y1.append(short(x))
    y2.append(medium(x))
    y3.append(high(x))
    x1.append(x)

plt.plot(x1, y1)
plt.plot(x1, y2)
plt.plot(x1, y3)
plt.legend(('short', 'medium', 'high'), loc='upper right')
plt.savefig("baketime.png", bbox_inches='tight')
```

code ส่วนของ size.py

```
import math
import matplotlib.pyplot as plt

def small(value):
    if 0 <= value <= 30:
        membership = ((-1 / 30) * value) + 1
    else:
        membership = 0

    return membership


def medium(value):
    if 25 <= value <= 60:
        membership = 1 - math.fabs((value - 42.5) / 17.5)

    else:
        membership = 0

    return membership


def high(value):
    if 50 <= value <= 100:
        membership = 1 - math.fabs((value - 75) / 25)
    else:
        membership = 0

    return membership
```

```
y1 = []
y2 = []
y3 = []
x1 = []
plt.figure(92)
for x in range(0, 110):
    y1.append(small(x))
    y2.append(medium(x))
    y3.append(high(x))
    x1.append(x)

plt.plot(x1, y1)
plt.plot(x1, y2)
plt.plot(x1, y3)
plt.savefig("size.png", bbox_inches='tight')
```

code ส่วนของ flour.py

```
import math
import matplotlib.pyplot as plt

def soft(value):
    if 0 <= value <= 3:
        membership = ((-1 / 3) * value) + 1
    else:
        membership = 0

    return membership


def medium(value):
    if 2 <= value <= 6:
        membership = 1 - math.fabs((value - 4) / 2)
    else:
        membership = 0

    return membership


def hard(value):
    if 5 <= value <= 10:
        membership = 1 - math.fabs((value - 7.5) / 2.5)
    else:
        membership = 0

    return membership
```

```
y1 = []
y2 = []
y3 = []
x1 = []
plt.figure(93)
for x in range(0, 11):
    y1.append(soft(x))
    y2.append(medium(x))
    y3.append(hard(x))
    x1.append(x)

plt.plot(x1, y1)
plt.plot(x1, y2)
plt.plot(x1, y3)
plt.legend(('soft', 'medium', 'hard'), loc='upper right')
plt.savefig("flour.png", bbox_inches='tight')
```

code ส่วนของ rule.py

```
import size as s
import baketime as t
import flour as f
import fire as fr
import matplotlib.pyplot as plt
import numpy as np

def Graph(x, y, rule,f):
    plt.figure(f)
    plt.plot(x, y)
    plt.savefig(rule, bbox_inches='tight')

#time is short, size is small, flour is soft so fire is low
def rule1(time, size, flour):
    mem_time = t.short(time)
    mem_size = s.small(size)
    mem_flour = f.soft(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0,len(x)):
        y.append(round(fr.low(x[i], acut), 2))

    Graph(x, y, "rule1", 1)
    return y
```

```
def rule2(time, size, flour):
    mem_time = t.short(time)
    mem_size = s.small(size)
    mem_flour = f.medium(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule2", 2)
    return y


def rule3(time, size, flour):
    mem_time = t.short(time)
    mem_size = s.small(size)
    mem_flour = f.hard(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.high(x[i], acut), 2))

    Graph(x, y, "rule3", 3)
    return y

def rule4(time, size, flour):
    mem_time = t.short(time)
    mem_size = s.medium(size)
    mem_flour = f.soft(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule4", 4)
    return y

def rule5(time, size, flour):
    mem_time = t.short(time)
    mem_size = s.medium(size)
```

```
    mem_flour = f.medium(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule5", 5)
    return y


def rule6(time, size, flour):
    mem_time = t.short(time)
    mem_size = s.medium(size)
    mem_flour = f.hard(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.high(x[i], acut), 2))

    Graph(x, y, "rule6", 6)
    return y

def rule7(time, size, flour):
    mem_time = t.short(time)
    mem_size = s.high(size)
    mem_flour = f.soft(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule7", 7)
    return y

def rule8(time, size, flour):
    mem_time = t.short(time)
    mem_size = s.high(size)
    mem_flour = f.medium(flour)

    x = np.arange(0, 210, 20)
```

```
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.high(x[i], acut), 2))

    Graph(x, y, "rule8", 8)
    return y

def rule9(time, size, flour):
    mem_time = t.short(time)
    mem_size = s.high(size)
    mem_flour = f.hard(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.high(x[i], acut), 2))

    Graph(x, y, "rule9", 9)
    return y

def rule10(time, size, flour):
    mem_time = t.medium(time)
    mem_size = s.small(size)
    mem_flour = f.soft(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.low(x[i], acut), 2))

    Graph(x, y, "rule10", 10)
    return y

def rule11(time, size, flour):
    mem_time = t.medium(time)
    mem_size = s.small(size)
    mem_flour = f.medium(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)
```

```
    for i in range(0, len(x)):
        y.append(round(fr.low(x[i], acut), 2))

    Graph(x, y, "rule11", 11)
    return y

def rule12(time, size, flour):
    mem_time = t.medium(time)
    mem_size = s.small(size)
    mem_flour = f.hard(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule12", 12)
    return y

def rule13(time, size, flour):
    mem_time = t.medium(time)
    mem_size = s.medium(size)
    mem_flour = f.soft(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule13", 13)
    return y

def rule14(time, size, flour):
    mem_time = t.medium(time)
    mem_size = s.medium(size)
    mem_flour = f.medium(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule14", 14)
```

```
    return y

def rule15(time, size, flour):
    mem_time = t.medium(time)
    mem_size = s.medium(size)
    mem_flour = f.hard(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.high(x[i], acut), 2))

    Graph(x, y, "rule15", 15)
    return y

def rule16(time, size, flour):
    mem_time = t.medium(time)
    mem_size = s.high(size)
    mem_flour = f.soft(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule16", 16)
    return y

def rule17(time, size, flour):
    mem_time = t.medium(time)
    mem_size = s.high(size)
    mem_flour = f.medium(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.high(x[i], acut), 2))

    Graph(x, y, "rule17", 17)
    return y

def rule18(time, size, flour):
    mem_time = t.medium(time)
```

```
    mem_size = s.high(size)
    mem_flour = f.hard(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.high(x[i], acut), 2))

    Graph(x, y, "rule18", 18)
    return y

def rule19(time, size, flour):
    mem_time = t.high(time)
    mem_size = s.small(size)
    mem_flour = f.soft(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.low(x[i], acut), 2))

    Graph(x, y, "rule19", 19)
    return y

def rule20(time, size, flour):
    mem_time = t.high(time)
    mem_size = s.small(size)
    mem_flour = f.medium(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.low(x[i], acut), 2))

    Graph(x, y, "rule20", 20)
    return y

def rule21(time, size, flour):
    mem_time = t.high(time)
    mem_size = s.small(size)
    mem_flour = f.hard(flour)

    x = np.arange(0, 210, 20)
```

```
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.low(x[i], acut), 2))

    Graph(x, y, "rule21", 21)
    return y

def rule22(time, size, flour):
    mem_time = t.high(time)
    mem_size = s.medium(size)
    mem_flour = f.soft(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.low(x[i], acut), 2))

    Graph(x, y, "rule22", 22)
    return y

def rule23(time, size, flour):
    mem_time = t.high(time)
    mem_size = s.medium(size)
    mem_flour = f.medium(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule23", 23)
    return y

def rule24(time, size, flour):
    mem_time = t.high(time)
    mem_size = s.medium(size)
    mem_flour = f.hard(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)
```

```
    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule24", 24)
    return y

def rule25(time, size, flour):
    mem_time = t.high(time)
    mem_size = s.high(size)
    mem_flour = f.soft(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule25", 25)
    return y

def rule26(time, size, flour):
    mem_time = t.high(time)
    mem_size = s.high(size)
    mem_flour = f.medium(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.medium(x[i], acut), 2))

    Graph(x, y, "rule26", 26)
    return y

def rule27(time, size, flour):
    mem_time = t.high(time)
    mem_size = s.high(size)
    mem_flour = f.hard(flour)

    x = np.arange(0, 210, 20)
    y = []

    acut = min(mem_time, mem_size, mem_flour)

    for i in range(0, len(x)):
        y.append(round(fr.high(x[i], acut), 2))
```

```
    Graph(x, y, "rule27", 27)
    return y
```

code ของ main.py

```
import matplotlib.pyplot as plt
import numpy as np
import rule as r
import fire as fr
plt.rcParams.update({'figure.max_open_warning': 0})
x = np.arange(0, 210, 20)
y = 0
yx = 0

time = float(input("time: "))
size = float(input("size: "))
flour = float(input("flour: "))

output1 = r.rule1(time, size, flour)
output2 = r.rule2(time, size, flour)
output3 = r.rule3(time, size, flour)
output4 = r.rule4(time, size, flour)
output5 = r.rule5(time, size, flour)
output6 = r.rule6(time, size, flour)
output7 = r.rule7(time, size, flour)
output8 = r.rule8(time, size, flour)
output9 = r.rule9(time, size, flour)
output10 = r.rule10(time, size, flour)
output11 = r.rule11(time, size, flour)
output12 = r.rule12(time, size, flour)
output13 = r.rule13(time, size, flour)
output14 = r.rule14(time, size, flour)
output15 = r.rule15(time, size, flour)
output16 = r.rule16(time, size, flour)
output17 = r.rule17(time, size, flour)
output18 = r.rule18(time, size, flour)
output19 = r.rule19(time, size, flour)
output20 = r.rule20(time, size, flour)
output21 = r.rule21(time, size, flour)
output22 = r.rule22(time, size, flour)
output23 = r.rule23(time, size, flour)
output24 = r.rule24(time, size, flour)
output25 = r.rule25(time, size, flour)
output26 = r.rule26(time, size, flour)
output27 = r.rule27(time, size, flour)
```

```
output = fr.union(output1, output2)
output = fr.union(output, output3)
output = fr.union(output, output4)
output = fr.union(output, output5)
output = fr.union(output, output6)
output = fr.union(output, output7)
output = fr.union(output, output8)
output = fr.union(output, output9)
output = fr.union(output, output10)
output = fr.union(output, output11)
output = fr.union(output, output12)
output = fr.union(output, output13)
output = fr.union(output, output14)
output = fr.union(output, output15)
output = fr.union(output, output16)
output = fr.union(output, output17)
output = fr.union(output, output18)
output = fr.union(output, output19)
output = fr.union(output, output20)
output = fr.union(output, output21)
output = fr.union(output, output22)
output = fr.union(output, output23)
output = fr.union(output, output24)
output = fr.union(output, output25)
output = fr.union(output, output26)
output = fr.union(output, output27)

print("output: ", output)
print("x: ", x)

for i in range(0, len(output)):
    y += x[i] * output[i]
    yx += output[i]
centroid = y / yx

print("Centroid: ", centroid)
plt.figure(0)
plt.plot(x, output)
plt.savefig("output.png", bbox_inches='tight')
```