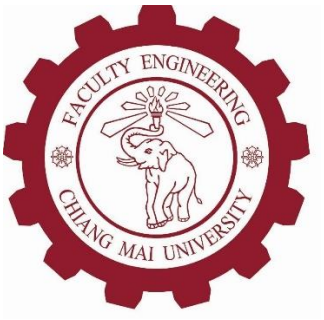


รายงานการทดลอง

Computer Assignment 2

261456 (Introduction to Computational Intelligence)

โดย

ปิยะนันท์ ปิยะวรรณ์โณ 650610845

เสนอ

รศ.ดร. ศันสนีย์ เอื้อพันธ์วิริยะกุล

ภาคเรียนที่ 1 ปีการศึกษา 2567

มหาวิทยาลัยเชียงใหม่

การทดลองทำระบบการเปลี่ยนเกียร์อัตโนมัติของรถ โดยใช้ Fuzzy logic

ลักษณะการทำงาน

เป็นโปรแกรมแบบ 2 Input 1 output โดยมี input คือ Speed (ความเร็วของรถ) และ throttle (%การเปิดของคันเร่ง) แล มี output เป็น gear (เกียร์ที่ใช้) โดยมี Membership ดังนี้

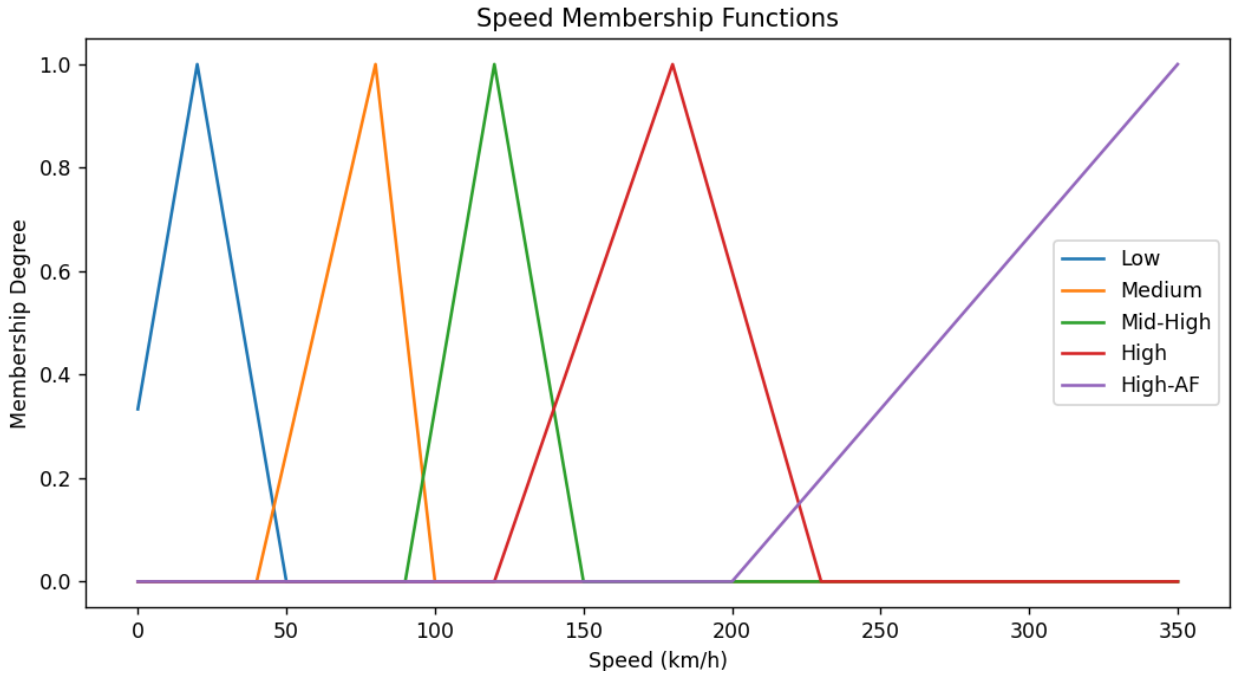
Speed = { low , medium , mid-high , high , high-af }

Throttle = { tap , light , medium , high , full-send }

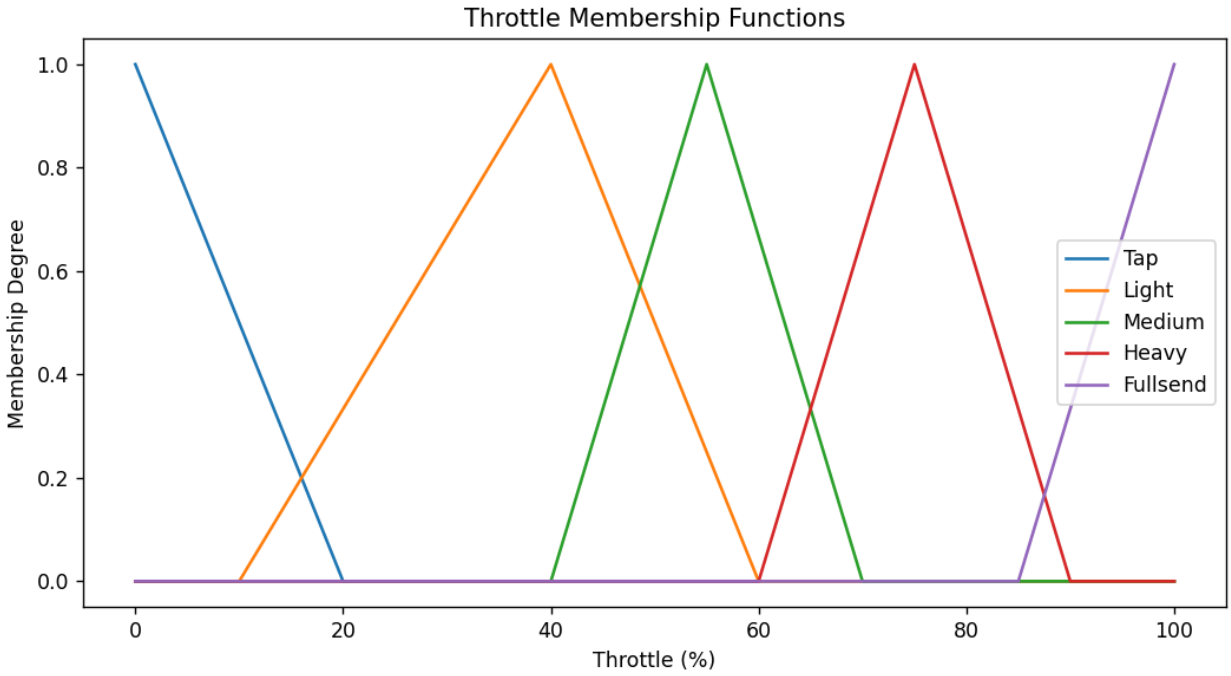
Gear = { first , second , third , fourth , fifth , sixth }

และมี Membership function ดังนี้

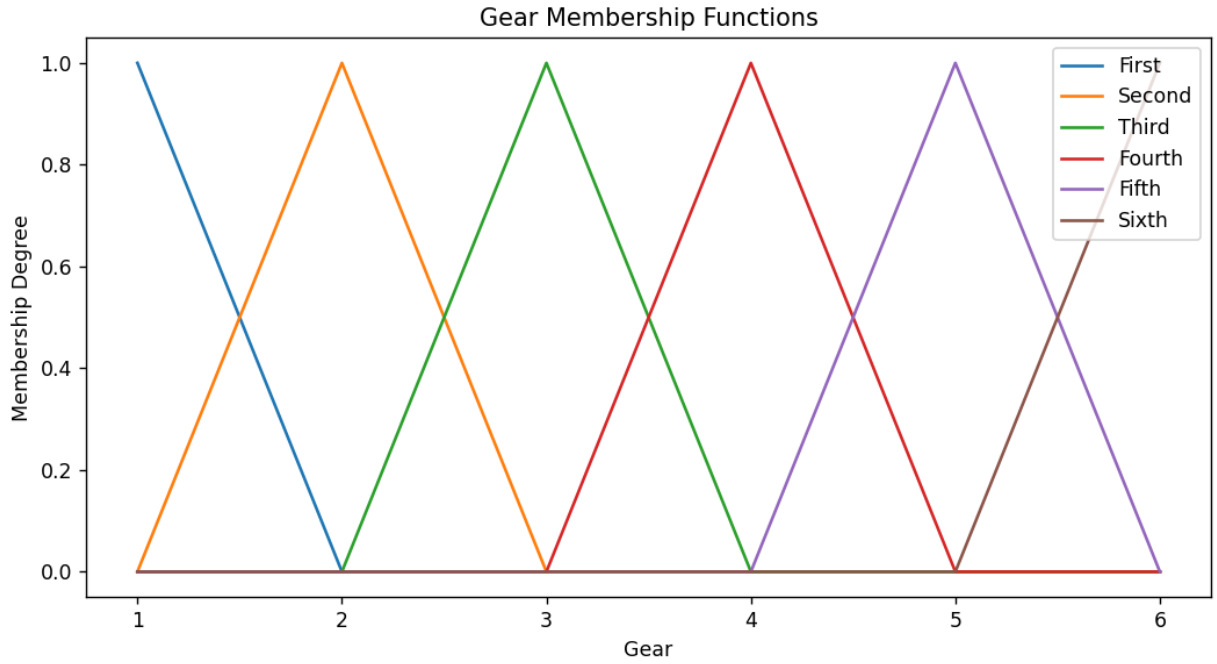
- Membership function ของ Speed



- Membership function ของ Throttle



- Membership function ของ Gear



และมี Rule คือ

- Fuzzy rule table for Gear Shift

	Tap	Light	Medium	Heavy	Fullsend
Low	First Gear	First Gear	First Gear	First Gear	First Gear
Medium	Second Gear	Second Gear	Third Gear	Second Gear	Second Gear
Mid-High	Third Gear	Third Gear	Third Gear	Fourth Gear	Third Gear
High	Sixth Gear	Fourth Gear	Fifth Gear	Fourth Gear	Fifth Gear
High-AF	Sixth Gear	Sixth Gear	Sixth Gear	Sixth Gear	Fifth Gear

การทดลอง

ทดลองโดยมี test case และ expect output ดังนี้

Test Case Table and Gear Shift Outputs

Speed (km/h)	Throttle (%)	Expected Gear
20	40	First
120	80	Fourth
180	100	Fifth
60	30	Second
10	10	First
130	40	Third
350	100	Fifth
50	60	Third
90	20	Second
200	85	Fourth
-10	50	First
150	-5	Sixth
400	50	Sixth
130	120	Fourth

ได้ output จากโปรแกรมเป็น ดังนี้

```
Test 1: Speed: 20 km/h, Throttle: 40% => Gear: First
Test 2: Speed: 120 km/h, Throttle: 80% => Gear: Fourth
Test 3: Speed: 180 km/h, Throttle: 100% => Gear: Fifth
Test 4: Speed: 60 km/h, Throttle: 30% => Gear: Second
Test 5: Speed: 10 km/h, Throttle: 10% => Gear: First
Test 6: Speed: 130 km/h, Throttle: 40% => Gear: Third
Test 7: Speed: 350 km/h, Throttle: 100% => Gear: Fifth
Test 8: Speed: 50 km/h, Throttle: 60% => Gear: Third
Test 9: Speed: 90 km/h, Throttle: 20% => Gear: Second
Test 10: Speed: 200 km/h, Throttle: 85% => Gear: Fourth
Test 11: Speed: -10 km/h, Throttle: 50% => Gear: First
Test 12: Speed: 150 km/h, Throttle: -5% => Gear: Sixth
Test 13: Speed: 400 km/h, Throttle: 50% => Gear: Sixth
Test 14: Speed: 130 km/h, Throttle: 120% => Gear: Fourth
```

จะเห็นได้ว่าโปรแกรมตาม Rule ที่ตั้งไว้ และให้ output ได้ถูกต้อง

โปรแรม

SPHSTR/Computer_Intelligence_Fuzzy

ใช้ libraly skfuzzy

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


# Define fuzzy input variables
speed = ctrl.Antecedent(np.arange(0, 351, 1), 'speed')  # Speed from 0 to 350 km/h
throttle = ctrl.Antecedent(np.arange(0, 101, 1), 'throttle')  # Throttle from 0% to 100%

# Define fuzzy output variable (Gear from 1 to 6)
gear = ctrl.Consequent(np.arange(1, 7, 1), 'gear')

# Define membership functions for speed
speed['low'] = fuzz.trimf(speed.universe, [-10, 20, 50])
speed['medium'] = fuzz.trimf(speed.universe, [40, 80, 100])
speed['midhigh'] = fuzz.trimf(speed.universe, [90, 120, 150])
speed['high'] = fuzz.trimf(speed.universe, [120, 180, 230])
speed['highaf'] = fuzz.trimf(speed.universe, [200, 350, 350])

# Define membership functions for throttle
throttle['tap'] = fuzz.trimf(throttle.universe, [0, 0, 20])
throttle['light'] = fuzz.trimf(throttle.universe, [10, 40, 60])
throttle['medium'] = fuzz.trimf(throttle.universe, [40, 55, 70])
throttle['heavy'] = fuzz.trimf(throttle.universe, [60, 75, 90])
throttle['fullsend'] = fuzz.trimf(throttle.universe, [85, 100, 100])

 # Define membership functions for gear
 gear['first'] = fuzz.trimf(gear.universe, [1, 1, 1])
 gear['second'] = fuzz.trimf(gear.universe, [2, 2, 2])
 gear['third'] = fuzz.trimf(gear.universe, [3, 3, 3])
 gear['fourth'] = fuzz.trimf(gear.universe, [4, 4, 4])
 gear['fifth'] = fuzz.trimf(gear.universe, [5, 5, 5])
 gear['sixth'] = fuzz.trimf(gear.universe, [6, 6, 6])

```

```
# Define fuzzy rules based on speed and throttle
rules = [
    ctrl.Rule(speed['low'] & throttle['tap'], gear['first']),
    ctrl.Rule(speed['low'] & throttle['light'], gear['first']),
    ctrl.Rule(speed['low'] & throttle['medium'], gear['first']),
    ctrl.Rule(speed['low'] & throttle['heavy'], gear['first']),
    ctrl.Rule(speed['low'] & throttle['fullsend'], gear['first']),
    ctrl.Rule(speed['medium'] & throttle['tap'], gear['second']),
    ctrl.Rule(speed['medium'] & throttle['light'], gear['second']),
    ctrl.Rule(speed['medium'] & throttle['heavy'], gear['second']),
    ctrl.Rule(speed['medium'] & throttle['fullsend'], gear['second']),
    ctrl.Rule(speed['medium'] & throttle['medium'], gear['third']),
    ctrl.Rule(speed['midhigh'] & throttle['light'], gear['third']),
    ctrl.Rule(speed['midhigh'] & throttle['medium'], gear['third']),
    ctrl.Rule(speed['midhigh'] & throttle['fullsend'], gear['third']),
    ctrl.Rule(speed['midhigh'] & throttle['tap'], gear['third']),
    ctrl.Rule(speed['high'] & throttle['light'], gear['fourth']),
    ctrl.Rule(speed['midhigh'] & throttle['heavy'], gear['fourth']),
    ctrl.Rule(speed['high'] & throttle['heavy'], gear['fourth']),
    ctrl.Rule(speed['high'] & throttle['fullsend'], gear['fifth']),
    ctrl.Rule(speed['highaf'] & throttle['fullsend'], gear['fifth']),
    ctrl.Rule(speed['highaf'] & throttle['medium'], gear['sixth']),
    ctrl.Rule(speed['highaf'] & throttle['tap'], gear['sixth']),
    ctrl.Rule(speed['highaf'] & throttle['light'], gear['sixth']),
    ctrl.Rule(speed['highaf'] & throttle['heavy'], gear['sixth']),
    ctrl.Rule(speed['high'] & throttle['tap'], gear['sixth']),
]


# Create control system and simulation
gear_ctrl = ctrl.ControlSystem(rules)
gear_shift = ctrl.ControlSystemSimulation(gear_ctrl)

# Map integer outputs to gear names
gear_names = {
    1: "First",
    2: "Second",
    3: "Third",
    4: "Fourth",
    5: "Fifth",
    6: "Sixth"
}
```

```python
# Simulate the gear shift system with output as words
def get_gear(speed_input, throttle_input):
    try:
        # Limit inputs within valid range
        speed_input = np.clip(speed_input, 0, 350)
        throttle_input = np.clip(throttle_input, 0, 100)

        # Set inputs for speed and throttle
        gear_shift.input['speed'] = speed_input
        gear_shift.input['throttle'] = throttle_input

        # Compute the result
        gear_shift.compute()

        # Get the integer gear output and map it to a word
        gear_output = int(round(gear_shift.output['gear']))
        return gear_names.get(gear_output, "Invalid Gear")
    except KeyError as e:
        print(f"Error: {e}")
        return "Error"
```

```python
# Test cases to validate the system
test_cases = [
    (20, 40),   # Low speed, light throttle
    (120, 80),  # High speed, medium throttle
    (180, 100), # High speed, full throttle
    (60, 30),   # Medium speed, light throttle
    (10, 10),   # Low speed, tap throttle
    (130, 40),  # Boundary case, high speed with light throttle
    (350, 100), # Maximum speed, full throttle
    (50, 60),   # Medium-low speed, medium throttle
    (90, 20),   # Boundary between medium and mid-high speed, tap throttle
    (200, 85),  # High-af speed, heavy throttle
    (-10, 50),  # Negative speed (edge case)
    (150, -5),  # Negative throttle (invalid input)
    (400, 50),  # Out-of-range speed (above 350 km/h)
    (130, 120)  # Throttle exceeding 100% (invalid input)
]

# Run the extended test cases
for i, (speed_input, throttle_input) in enumerate(test_cases):
    gear_selected = get_gear(speed_input, throttle_input)
    print(f"Test {i+1}: Speed: {speed_input} km/h, Throttle: {throttle_input}% => Gear: {gear_selected}")
```

```python
def plot_speed_mf():
    plt.figure(figsize=(10, 5))
    plt.plot(speed.universe, speed['low'].mf, label='Low')
    plt.plot(speed.universe, speed['medium'].mf, label='Medium')
    plt.plot(speed.universe, speed['midhigh'].mf, label='Mid-High')
    plt.plot(speed.universe, speed['high'].mf, label='High')
    plt.plot(speed.universe, speed['highaf'].mf, label='High-AF')
    plt.title('Speed Membership Functions')
    plt.xlabel('Speed (km/h)')
    plt.ylabel('Membership Degree')
    plt.legend()
    plt.show()

# Plot throttle membership functions
def plot_throttle_mf():
    plt.figure(figsize=(10, 5))
    plt.plot(throttle.universe, throttle['tap'].mf, label='Tap')
    plt.plot(throttle.universe, throttle['light'].mf, label='Light')
    plt.plot(throttle.universe, throttle['medium'].mf, label='Medium')
    plt.plot(throttle.universe, throttle['heavy'].mf, label='Heavy')
    plt.plot(throttle.universe, throttle['fullsend'].mf, label='Fullsend')
    plt.title('Throttle Membership Functions')
    plt.xlabel('Throttle (%)')
    plt.ylabel('Membership Degree')
    plt.legend()
    plt.show()
# Plot gear membership functions
def plot_gear_mf():
    plt.figure(figsize=(10, 5))
    plt.plot(gear.universe, gear['first'].mf, label='First')
    plt.plot(gear.universe, gear['second'].mf, label='Second')
    plt.plot(gear.universe, gear['third'].mf, label='Third')
    plt.plot(gear.universe, gear['fourth'].mf, label='Fourth')
    plt.plot(gear.universe, gear['fifth'].mf, label='Fifth')
    plt.plot(gear.universe, gear['sixth'].mf, label='Sixth')
    plt.title('Gear Membership Functions')
    plt.xlabel('Gear')
    plt.ylabel('Membership Degree')
    plt.legend()
    plt.show()

# Call the plotting functions
plot_speed_mf()
plot_throttle_mf()
plot_gear_mf()

```

```python
plt.title("Fuzzy Logic Rule Table for Gear Shift", fontsize=16)
plt.show()

output_gears = []
```

```python
# Function to simulate gear shift
for speed_input, throttle_input in test_cases:
    gear = get_gear(speed_input, throttle_input)  # Using the `get_gear` function from your code
    output_gears.append(gear)

# Create a DataFrame for the Test Case Table
df_test_cases = pd.DataFrame(test_cases, columns=['Speed (km/h)', 'Throttle (%)'])
df_test_cases['Expected Gear'] = output_gears  # Add the computed gear column

# Plot Test Case Table using Matplotlib
fig, ax = plt.subplots(figsize=(10, 4))
ax.axis('off')  # Hide axis

# Display test case table
table = ax.table(
    cellText=df_test_cases.values,
    colLabels=df_test_cases.columns,
    cellLoc='center',
    loc='center',
)

# Formatting
table.auto_set_font_size(False)
table.set_fontsize(12)
table.scale(1.2, 1.2)
```

```python
data = [
    ['First Gear', 'First Gear', 'First Gear', 'First Gear', 'First Gear'],  # Low Speed
    ['Second Gear', 'Second Gear', 'Third Gear', 'Second Gear', 'Second Gear'],  # Medium Speed
    ['Third Gear', 'Third Gear', 'Third Gear', 'Fourth Gear', 'Third Gear'],  # Mid-High Speed
    ['Sixth Gear', 'Fourth Gear', 'Fifth Gear', 'Fourth Gear', 'Fifth Gear'],  # High Speed
    ['Sixth Gear', 'Sixth Gear', 'Sixth Gear', 'Sixth Gear', 'Fifth Gear'],  # High-AF Speed
]

# Labels for rows and columns
row_labels = ['Low', 'Medium', 'Mid-High', 'High', 'High-AF']
col_labels = ['Tap', 'Light', 'Medium', 'Heavy', 'Fullsend']

# Create a DataFrame to display the rule table
df = pd.DataFrame(data, index=row_labels, columns=col_labels)

# Plotting the table using Matplotlib
fig, ax = plt.subplots(figsize=(10, 4))
ax.axis('off')  # Turn off the axes

# Create the table with pandas DataFrame
table = ax.table(
    cellText=df.values,
    rowLabels=df.index,
    colLabels=df.columns,
    cellLoc='center',
    loc='center',
)

# Formatting the table
table.auto_set_font_size(False)
table.set_fontsize(12)
table.scale(1.2, 1.2)  # Adjust the table size
```

```
239    # Set title and show plot
240    plt.title("Test Case Table and Gear Shift Outputs", fontsize=16)
241    plt.show()
242
```