ЛАБОРАТОРНА РОБОТА № 3

**МОДЕЛЮВАННЯ ЕЛЕМЕНТІВ НЕЧІТКИХ МНОЖИН ТА ФОРМУВАННЯ НЕЧІТКИХ ПРАВИЛ**

**Мета:** дослідити можливості ППП MATLAB щодо проектування систем керування на основі алгоритмів нечіткого виводу.

Git: https://github.com/flekXD/SAI

Завдання 1.

Побудова нечіткої моделі системи керування кранами гарячої і холодної води При користуванням системою водопостачання на вхід змішувача подається холодна та гаряча вода по відповідним трубопроводам. Задача полягає у створенні моделі системи засобами Matlab Fuzzy Logic, яка б дозволила автоматизувати процес. Кран змішувача можна повертати наліво і направо (тобто, область визначення кута - це відрізок [-90;90] градусів), керуючи тим самим температурою води і її напором. Нехай, повернення будьякого крану направо - це збільшити потік води відповідної температури.

import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl

import matplotlib.pyplot as plt

temperature = ctrl.Antecedent(np.arange(0, 101, 1), 'temperature')  # Температура [0-100 °C]

pressure = ctrl.Antecedent(np.arange(0, 101, 1), 'pressure')        # Напір [0-100 %]

hot\_valve = ctrl.Consequent(np.arange(-90, 91, 1), 'hot\_valve')     # Кут крана гарячої води

cold\_valve = ctrl.Consequent(np.arange(-90, 91, 1), 'cold\_valve')   # Кут крана холодної води

# Функції належності для температури

temperature['cold'] = fuzz.trimf(temperature.universe, [0, 0, 50])

temperature['cool'] = fuzz.trimf(temperature.universe, [0, 50, 75])

temperature['warm'] = fuzz.trimf(temperature.universe, [50, 75, 100])

temperature['hot'] = fuzz.trimf(temperature.universe, [75, 100, 100])

# Функції належності для напору

pressure['low'] = fuzz.trimf(pressure.universe, [0, 0, 50])

pressure['medium'] = fuzz.trimf(pressure.universe, [25, 50, 75])

pressure['high'] = fuzz.trimf(pressure.universe, [50, 100, 100])

# Функції належності для кутів

hot\_valve['large\_left'] = fuzz.trimf(hot\_valve.universe, [-90, -90, -45])

hot\_valve['medium\_left'] = fuzz.trimf(hot\_valve.universe, [-60, -30, 0])

hot\_valve['small\_left'] = fuzz.trimf(hot\_valve.universe, [-30, 0, 30])

hot\_valve['no\_change'] = fuzz.trimf(hot\_valve.universe, [-15, 0, 15])

hot\_valve['small\_right'] = fuzz.trimf(hot\_valve.universe, [0, 30, 60])

hot\_valve['medium\_right'] = fuzz.trimf(hot\_valve.universe, [30, 60, 90])

hot\_valve['large\_right'] = fuzz.trimf(hot\_valve.universe, [45, 90, 90])

cold\_valve['large\_left'] = fuzz.trimf(cold\_valve.universe, [-90, -90, -45])

cold\_valve['medium\_left'] = fuzz.trimf(cold\_valve.universe, [-60, -30, 0])

cold\_valve['small\_left'] = fuzz.trimf(cold\_valve.universe, [-30, 0, 30])

cold\_valve['no\_change'] = fuzz.trimf(cold\_valve.universe, [-15, 0, 15])

cold\_valve['small\_right'] = fuzz.trimf(cold\_valve.universe, [0, 30, 60])

cold\_valve['medium\_right'] = fuzz.trimf(cold\_valve.universe, [30, 60, 90])

cold\_valve['large\_right'] = fuzz.trimf(cold\_valve.universe, [45, 90, 90])

# Правила нечіткої логіки

rules = [

    ctrl.Rule(temperature['hot'] & pressure['high'],

              (hot\_valve['medium\_left'], cold\_valve['medium\_right'])),

    ctrl.Rule(temperature['hot'] & pressure['medium'],

              cold\_valve['medium\_right']),

    ctrl.Rule(temperature['warm'] & pressure['high'],

              hot\_valve['small\_left']),

    ctrl.Rule(temperature['warm'] & pressure['low'],

              (hot\_valve['small\_right'], cold\_valve['small\_right'])),

    ctrl.Rule(temperature['cool'] & pressure['medium'],

              (hot\_valve['no\_change'], cold\_valve['no\_change'])),

    ctrl.Rule(temperature['cool'] & pressure['high'],

              (hot\_valve['medium\_right'], cold\_valve['medium\_left'])),

    ctrl.Rule(temperature['cool'] & pressure['low'],

              (hot\_valve['medium\_right'], cold\_valve['small\_left'])),

    ctrl.Rule(temperature['cold'] & pressure['low'],

              hot\_valve['large\_right']),

    ctrl.Rule(temperature['cold'] & pressure['high'],

              (hot\_valve['medium\_left'], cold\_valve['medium\_right'])),

    ctrl.Rule(temperature['warm'] & pressure['high'],

              (hot\_valve['small\_left'], cold\_valve['small\_left'])),

    ctrl.Rule(temperature['warm'] & pressure['low'],

              (hot\_valve['small\_right'], cold\_valve['small\_right']))

]

# Система керування

control\_system = ctrl.ControlSystem(rules)

mixer = ctrl.ControlSystemSimulation(control\_system)

# Візуалізація

def visualize\_memberships():

    temperature.view()

    pressure.view()

    hot\_valve.view()

    cold\_valve.view()

def simulate\_and\_visualize(temp, pres):

    mixer.input['temperature'] = temp

    mixer.input['pressure'] = pres

    mixer.compute()

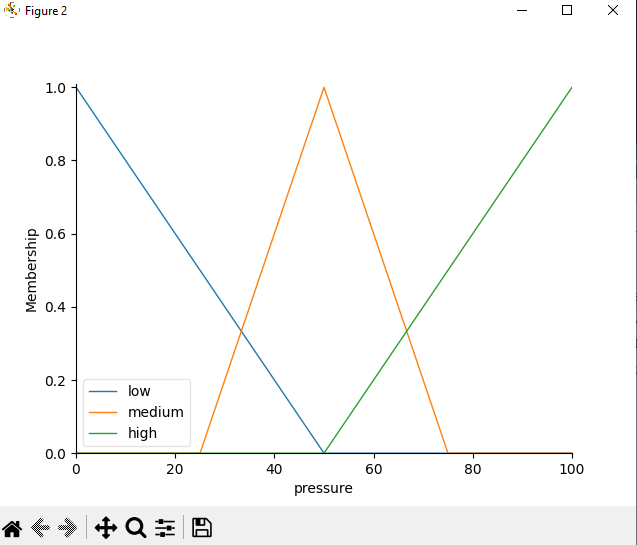
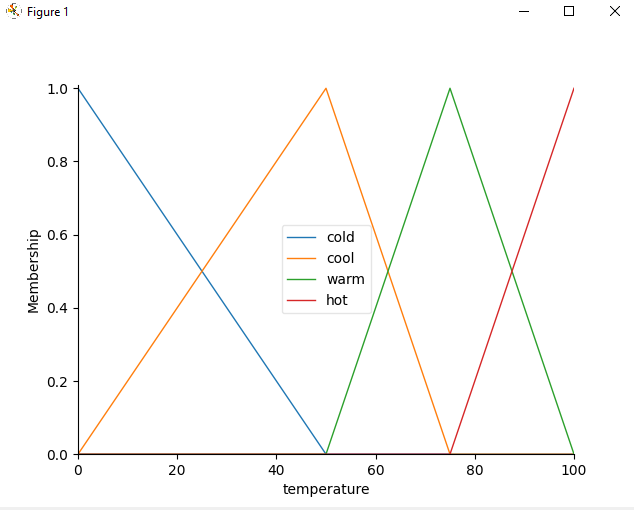
    hot\_valve.view(sim=mixer)

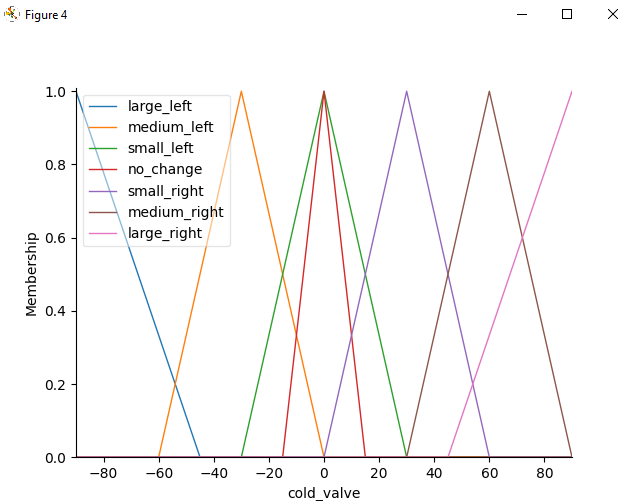
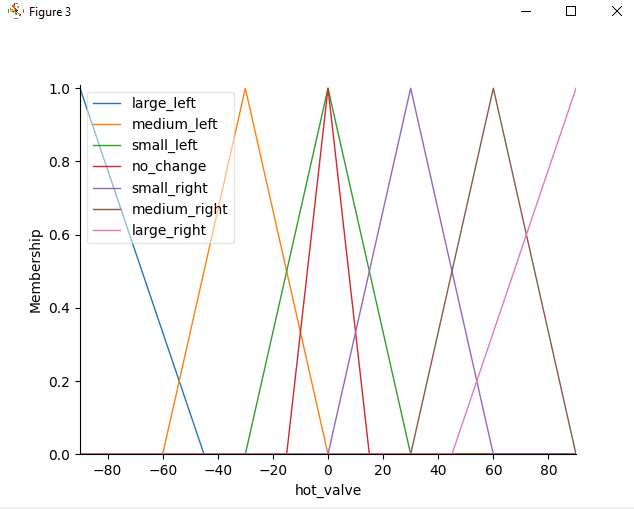
    cold\_valve.view(sim=mixer)

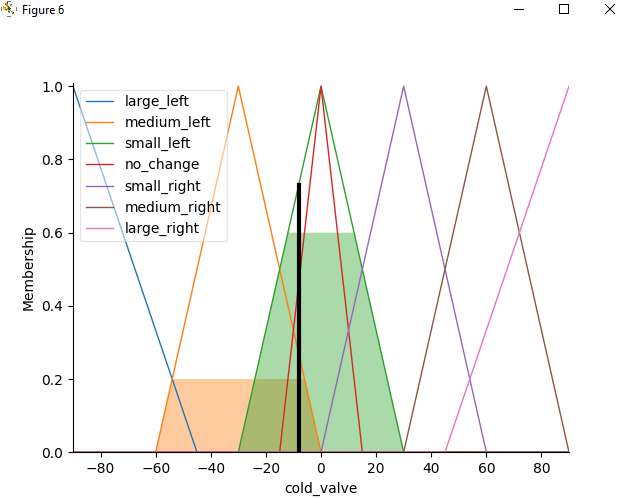
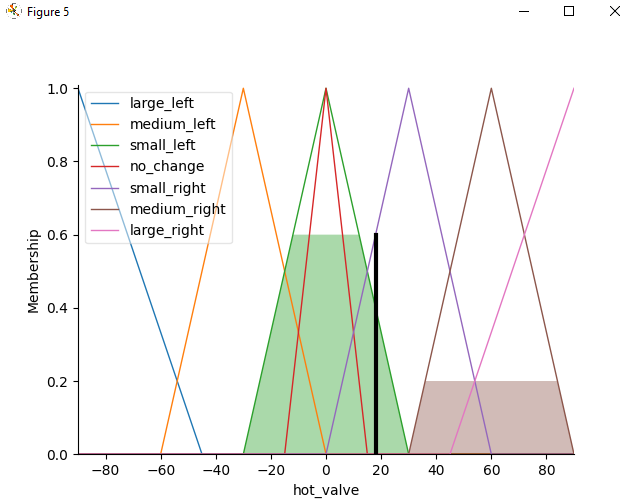
visualize\_memberships()

simulate\_and\_visualize(70, 80)

plt.show()







Модель дозволяє врахувати комбіновану дію двох факторів (температури та напору) для точного регулювання кранів.

Використання нечіткої логіки забезпечує плавність переходу між станами та можливість адаптації до різних ситуацій.

Завдання 2.

Нечітка модель керування кондиціонером повітря в приміщенні. Нехай, в приміщенні встановлений кондиціонер, який дозволяє регулювати (нагрівати чи охолоджувати) температуру. Найбільш комфортні умови складаються при встановленні деякої заданої комфортної температури. Задача полягає у розробці АСУ, яка б змогла автоматизувати роботу кондиціонера при коливанні температури приміщення через різні зовнішні дестабілізуючі фактори.

import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl

import matplotlib.pyplot as plt

temperature = ctrl.Antecedent(np.arange(0, 41, 1), 'temperature')  # Температура (°C)

temp\_change = ctrl.Antecedent(np.arange(-5, 6, 1), 'temp\_change')  # Швидкість зміни температури (°C/хв)

ac\_regulator = ctrl.Consequent(np.arange(-90, 91, 1), 'ac\_regulator')  # Кут регулятора кондиціонера

# Функції належності для температури

temperature['very\_cold'] = fuzz.trimf(temperature.universe, [0, 0, 10])

temperature['cold'] = fuzz.trimf(temperature.universe, [5, 10, 20])

temperature['normal'] = fuzz.trimf(temperature.universe, [15, 20, 25])

temperature['warm'] = fuzz.trimf(temperature.universe, [20, 30, 35])

temperature['very\_warm'] = fuzz.trimf(temperature.universe, [30, 40, 40])

# Функції належності для швидкості зміни температури

temp\_change['negative'] = fuzz.trimf(temp\_change.universe, [-5, -5, 0])

temp\_change['zero'] = fuzz.trimf(temp\_change.universe, [-1, 0, 1])

temp\_change['positive'] = fuzz.trimf(temp\_change.universe, [0, 5, 5])

# Функції належності для регулятора

ac\_regulator['large\_left'] = fuzz.trimf(ac\_regulator.universe, [-90, -90, -45])

ac\_regulator['small\_left'] = fuzz.trimf(ac\_regulator.universe, [-45, -15, 0])

ac\_regulator['no\_change'] = fuzz.trimf(ac\_regulator.universe, [-15, 0, 15])

ac\_regulator['small\_right'] = fuzz.trimf(ac\_regulator.universe, [0, 15, 45])

ac\_regulator['large\_right'] = fuzz.trimf(ac\_regulator.universe, [45, 90, 90])

# Правила нечіткої логіки

rules = [

    ctrl.Rule(temperature['very\_warm'] & temp\_change['positive'], ac\_regulator['large\_left']),

    ctrl.Rule(temperature['very\_warm'] & temp\_change['negative'], ac\_regulator['small\_left']),

    ctrl.Rule(temperature['very\_warm'] & temp\_change['zero'], ac\_regulator['large\_left']),

    ctrl.Rule(temperature['warm'] & temp\_change['positive'], ac\_regulator['large\_left']),

    ctrl.Rule(temperature['warm'] & temp\_change['negative'], ac\_regulator['no\_change']),

    ctrl.Rule(temperature['warm'] & temp\_change['zero'], ac\_regulator['small\_left']),

    ctrl.Rule(temperature['very\_cold'] & temp\_change['negative'], ac\_regulator['large\_right']),

    ctrl.Rule(temperature['very\_cold'] & temp\_change['positive'], ac\_regulator['small\_right']),

    ctrl.Rule(temperature['very\_cold'] & temp\_change['zero'], ac\_regulator['large\_right']),

    ctrl.Rule(temperature['cold'] & temp\_change['negative'], ac\_regulator['large\_right']),

    ctrl.Rule(temperature['cold'] & temp\_change['positive'], ac\_regulator['no\_change']),

    ctrl.Rule(temperature['cold'] & temp\_change['zero'], ac\_regulator['small\_right']),

    ctrl.Rule(temperature['normal'] & temp\_change['positive'], ac\_regulator['small\_left']),

    ctrl.Rule(temperature['normal'] & temp\_change['negative'], ac\_regulator['small\_right']),

    ctrl.Rule(temperature['normal'] & temp\_change['zero'], ac\_regulator['no\_change'])

]

# Система керування

control\_system = ctrl.ControlSystem(rules)

ac\_simulation = ctrl.ControlSystemSimulation(control\_system)

# Візуалізація

def visualize\_memberships():

    temperature.view()

    temp\_change.view()

    ac\_regulator.view()

def simulate\_and\_visualize(temp, temp\_rate\_change):

    ac\_simulation.input['temperature'] = temp

    ac\_simulation.input['temp\_change'] = temp\_rate\_change

    ac\_simulation.compute()

    ac\_regulator.view(sim=ac\_simulation)

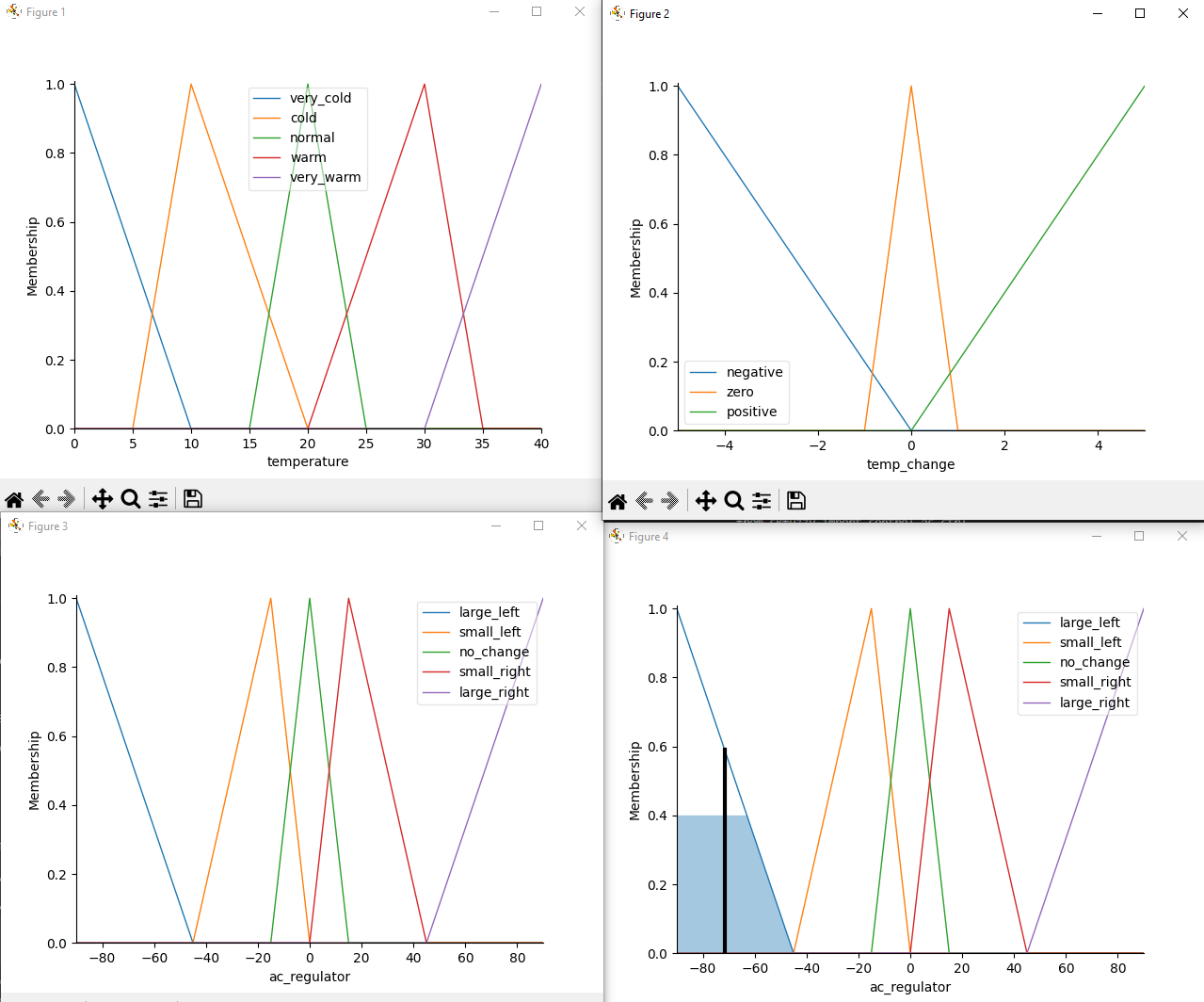
# Візуалізація функцій належності

visualize\_memberships()

# Тестування моделі з візуалізацією

simulate\_and\_visualize(28, 2)

plt.show()



Використання нечіткої логіки забезпечує плавну та адаптивну роботу системи в умовах змінних зовнішніх впливів.

Модель враховує інертність кондиціонера, що дозволяє уникнути надмірного регулювання.

Ефективне керування кондиціонером можливе навіть за складних умов, що включають зростання чи спад температури.

**Висновок:**

Використання нечіткої логіки у двох завданнях дозволяє автоматизувати процеси регулювання з урахуванням багатофакторних умов. Гнучкість нечітких моделей робить їх придатними для задач, де складно сформулювати чіткі математичні залежності. Завдяки візуалізації функцій належності та результатів можна легко інтерпретувати дії моделей. Для обох задач можна додатково покращити точність шляхом оптимізації функцій належності або додавання нових правил.