Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»
Мегафакультет компьютерных технологий и управления
Факультет программной инженерии и компьютерной техники

Лабораторная работа № 2
по дисциплине «Теория систем»

Выполнил: студент
Чайкин Вадим Константинович
группа P3324
Принял: преподаватель
Русак Алёна Викторовна

г. Санкт-Петербург
2025

Задание лабораторной работы

Цель работы

Получить навыки построения нечетких систем.
Используя Python-библиотеку Skfuzzy построить нечёткую базу знаний для моделирования нечеткой системы в соответствии с выбранным вариантом.

Этапы выполнения работы

1. Определить входные и выходные лингвистические переменные (общее количество лингвистических переменных должно быть не менее 3).

2. Задать функции принадлежности для нечетких подмножеств, определенных на значениях выбранных лингвистических переменных (не менее трех значений для каждой лингвистической переменной).

3. Определить нечеткие правила, описывающие работу рассматриваемой системы (не менее четырех правил, при этом нечёткая база знаний должна быть полной).

4. Промоделировать работу нечеткой системы (не менее 5 прогонов с разными входными данными, заданными случайным образом).

Содержание отчета

1. Описание предметной области, в том числе описание входных данных и выходных переменных, в чем они измеряются и т.д.

2. Нечеткие правила, описывающие работу системы.

3. Графики функций принадлежности для нечетких подмножеств, определенных на значениях выбранных лингвистических переменных.

4. Результаты нечёткого вывода (не мене пяти прогонов).

Выполнение лабораторной работы

Пользовательский вариант: Построить нечеткую базу знаний для задачи управления яркостью уличного освещения (естественная освещённость, дальность видимости), произвести нечеткий вывод для конкретных значений.
Пусть лампа освещает некоторый участок улицы. Яркость каждой лампы регулируется с помощью диммера (светорегулятора), представляющего собой реостат, или переменный резистор.
Очевидно, что чем меньше освещённость участка, тем бóльшая яркость лампы требуется, чтобы его осветить. Также она должна быть тем больше, чем меньше дальность видимости. Чтобы увеличить яркость лампы, нужно уменьшить сопротивление.

1. Лингвистические переменные

В качестве входных лингвистических переменных используем естественную освещённость (β_1) и оптическую дальность (β_2), в качестве выходной переменной — сопротивление реостата (β_3). Определим терм-множества (множества значений) каждой из лингвистических переменных.

- Входная переменная β_1: Естественная освещённость, клк.
  - Терм-множество $T(\beta_1) = \{$«освещённость слабая», «освещённость средняя», «освещённость сильная»$\}$.
  - Введём следующие обозначения:
    - `illum_lo` – освещённость слабая;
    - `illum_md` – освещённость средняя;
    - `illum_hi` – освещённость сильная.
- Входная переменная β_2: Оптическая дальность, м.
  - Терм-множество $T(\beta_2) = \{$«дальность малая», «дальность средняя», «дальность большая»$\}$.
  - Введём следующие обозначения:
    - `dist_lo` – дальность малая;
    - `dist_md` – дальность средняя;
    - `dist_hi` – дальность большая.
- Выходная переменная β_3: сопротивление, Ом.
  - Терм-множество $T(\beta_3) = \{$«сопротивление очень низкое», «сопротивление низкое», «сопротивление среднее», «сопротивление высокое», «сопротивление очень высокое»$\}$.
  - Введём следующие обозначения:
    - `res_vlo` – сопротивление очень низкое;
    - `res_lo` – сопротивление низкое;
    - `res_md` – сопротивление среднее;
    - `res_hi` – сопротивление высокое;
    - `res_vhi` – сопротивление очень высокое.

2. Функции принадлежности

Используем треугольные и «плечевые» функции принадлежности.
Треугольная функция имеет вид

$$\text{trimf}(x, a, b, c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a < x \le b \\ \dfrac{x-c}{b-c}, & b < x \le c \\ 0, & c < x \end{cases}, a \le b.$$

«Левое плечо» — это убывающая функция, которая имеет вид:

$$\text{left\_shoulder}(x, a, b) = \begin{cases} 1, & x \le a \\ \dfrac{x-b}{a-b}, & a < x \le b \\ 0, & b < x \end{cases}, \ a \le b.$$

«Правое плечо» — это возрастающая функция, которая имеет вид:

$$\text{right\_shoulder}(x, a, b) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a < x \le b \\ 1, & b < x \end{cases}, \ a \le b \le c \le d.$$

Зададим функции принадлежности для освещённости (E):

$$\mu_{\text{illum\_lo}}(E) = \text{left\_shoulder}(E, 1, 6).$$

$$\mu_{\text{illum\_md}}(E) = \text{trimf}(E, 1, 6, 11).$$

$$\mu_{\text{illum\_hi}}(E) = \text{right\_shoulder}(E, 6, 11).$$

Зададим функции принадлежности для дальности (d):

$$\mu_{\text{dist\_lo}}(E) = \text{left\_shoulder}(d, 0, 5).$$

$$\mu_{\text{dist\_md}}(E) = \text{trimf}(d, 0, 5, 10).$$

$$\mu_{\text{dist\_hi}}(E) = \text{right\_shoulder}(d, 5, 10).$$

Зададим функции принадлежности для сопротивления (R):

$$\mu_{\text{res\_vlo}}(R) = \text{left\_shoulder}(R, 120, 140).$$

$$\mu_{\text{res\_lo}}(R) = \text{trimf}(R, 120, 140, 160).$$

$$\mu_{\text{res\_md}}(R) = \text{trimf}(R, 140, 160, 180).$$

$$\mu_{\text{res\_hi}}(R) = \text{trimf}(R, 160, 180, 200).$$

$$\mu_{\text{res\_vhi}}(R) = \text{right\_shoulder}(R, 180, 200).$$

Поскольку функций `left_shoulder` и `right_shoulder` не существует в библиотеке `skfuzzy`, мы можем самостоятельно её реализовать.

```
import numpy as np

def left_shoulder(x, a, b):
    # Ensure x is a NumPy array for vectorized operations
    x = np.asarray(x)

    # Initialize output array
    y = np.zeros_like(x, dtype=float)

    # Region 1: x > b -> 0 (already zero-initialized)
    # Region 2: b >= x >= a -> linear ramp
    in_ramp = (x >= a) & (x <= b)
    y[in_ramp] = (x[in_ramp] - b) / (a - b)

    # Region 3: x < a -> 1
    below_a = x < a
    y[below_a] = 1.0

    return y

def right_shoulder(x, a, b):
    # Ensure x is a NumPy array for vectorized operations
    x = np.asarray(x)

    # Initialize output array
    y = np.zeros_like(x, dtype=float)

    # Region 1: x < a -> 0 (already zero-initialized)
    # Region 2: a <= x <= b -> linear ramp
    in_ramp = (x >= a) & (x <= b)
    y[in_ramp] = (x[in_ramp] - a) / (b - a)

    # Region 3: x > b -> 1
    above_b = x > b
    y[above_b] = 1.0

    return y
```

Зададим переменные и функции принадлежности. Построим графики функций принадлежности.

```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

# Generate universe variables
#   * Illuminance on range 0..12 klm
#   * Distance on range 0..10 m
#   * Resistance on range 0..220 Om
x_illum = np.linspace(0., 12., num=120 + 1)
x_dist = np.linspace(0., 10., num=100 + 1)
x_res  = np.linspace(100., 220., num=120 + 1)

# Generate fuzzy membership functions
illum_lo = left_shoulder(x_illum, 1, 6)
```

```
illum_md = fuzz.trimf(x_illum, [1, 6, 11])
illum_hi = right_shoulder(x_illum, 6, 11)
dist_lo = left_shoulder(x_dist, 0, 5)
dist_md = fuzz.trimf(x_dist, [0, 5, 10])
dist_hi = right_shoulder(x_dist, 5, 10)
res_vlo = left_shoulder(x_res, 120, 140)
res_lo = fuzz.trimf(x_res, [120, 140, 160])
res_md = fuzz.trimf(x_res, [140, 160, 180])
res_hi = fuzz.trimf(x_res, [160, 180, 200])
res_vhi = right_shoulder(x_res, 180, 200)

# Visualize these universes and membership functions
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(8, 9))

ax0.plot(x_illum, illum_lo, 'b', linewidth=1.5, label='Low')
ax0.plot(x_illum, illum_md, 'g', linewidth=1.5, label='Medium')
ax0.plot(x_illum, illum_hi, 'r', linewidth=1.5, label='High')
ax0.set_title('Illuminance (klm)')
ax0.legend()

ax1.plot(x_dist, dist_lo, 'b', linewidth=1.5, label='Low')
ax1.plot(x_dist, dist_md, 'g', linewidth=1.5, label='Medium')
ax1.plot(x_dist, dist_hi, 'r', linewidth=1.5, label='High')
ax1.set_title('Distance (m)')
ax1.legend()

ax2.plot(x_res, res_vlo, 'm', linewidth=1.5, label='Very Low')
ax2.plot(x_res, res_lo, 'b', linewidth=1.5, label='Low')
ax2.plot(x_res, res_md, 'g', linewidth=1.5, label='Medium')
ax2.plot(x_res, res_hi, 'r', linewidth=1.5, label='High')
ax2.plot(x_res, res_vhi, 'k', linewidth=1.5, label='Very High')
ax2.set_title('Resistance (Om)')
ax2.legend()

# Turn off top/right axes
for ax in (ax0, ax1, ax2):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()

plt.tight_layout()
```

Полный код блокнота приведён по ссылке.

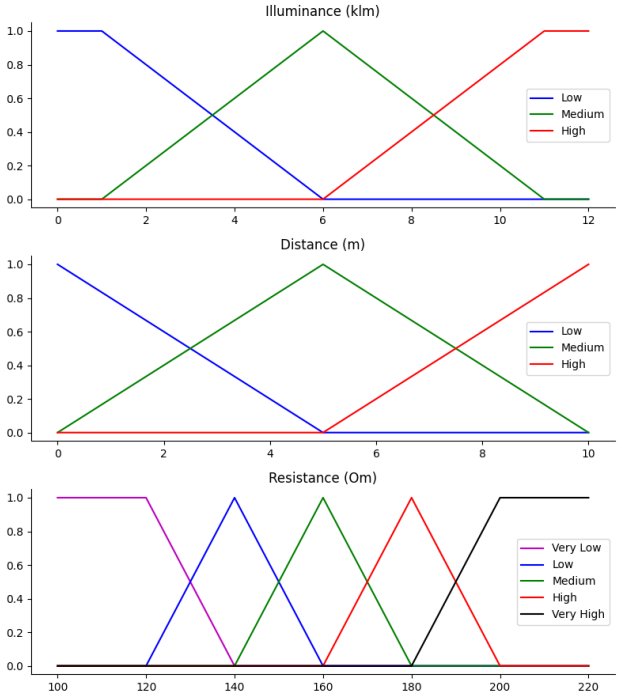

Рис. 1: Графики функций принадлежности

3. Определение нечётких правил

Определим базу нечётких правил на основе логики управления освещённостью. Принципы работы системы:

1. Чем меньше естественная освещённость (E), тем выше должна быть яркость лампы, что соответствует уменьшению сопротивления реостата (R).
2. Чем меньше дальность видимости (d), тем выше должна быть яркость лампы, что также означает уменьшение сопротивления (R).
3. Если и освещённость слабая, и дальность малая, то яркость должна быть максимальной (наименьшее R).
4. Если и освещённость высокая, и дальность большая, то яркость должна быть минимальной (наибольшее R).

Это позволяет задать следующие правила:

1. ЕСЛИ E слабая И d малая, ТО R очень низкое.
2. ЕСЛИ (E слабая И d средняя) ИЛИ (ЕСЛИ E средняя И d малая), ТО R низкое.
3. ЕСЛИ (E слабая И d большая) ИЛИ (E средняя И d средняя) ИЛИ (E сильная И d малая), ТО R среднее.
4. ЕСЛИ (E средняя И d большая) ИЛИ (E сильная И d средняя), ТО R высокое.
5. ЕСЛИ E сильная И d большая, ТО R очень высокое.

В коде эти правила можно задать следующим образом:

1. Используя функции принадлежности объединения и пересечения:

$$\mu_{A \cup B} = \max(\mu_A, \mu_B)$$

$$\mu_{A \cap B} = \min(\mu_A, \mu_B)$$

```
res_activation_vlo = np.fmin(active_rule1, res_vlo)  # removed
   entirely to 0

res_activation_lo = np.fmin(
    np.fmax(
        np.fmin(illum_level_lo, dist_level_md),
        np.fmin(illum_level_md, dist_level_lo)
    )
, res_lo)

active_rule3 = np.fmax(
    np.fmax(
        np.fmin(illum_level_lo, dist_level_hi),
        np.fmin(illum_level_md, dist_level_md)
    ),
    np.fmin(illum_level_hi, dist_level_lo)
)
res_activation_md = np.fmin(active_rule3, res_md)

```

```
res_activation_hi = np.fmin(
    np.fmax(
        np.fmin(illum_level_hi, dist_level_md),
        np.fmin(illum_level_md, dist_level_hi)
    )
, res_hi)

```

2. С помощью пакета `skfuzzy.control`:

```
rules = [
    ctrl.Rule(illum['low'] & dist['low'], res['very_low']),
    ctrl.Rule(
        (illum['low'] & dist['medium']) | (illum['medium'] & dist['low']),
        res['low']
    ),
    ctrl.Rule(
        (illum['low'] & dist['high']) | (illum['medium'] & dist['medium']) | (illum['high'] & dist['low']),
        res['medium']
    ),
    ctrl.Rule(
        (illum['high'] & dist['medium']) | (illum['medium'] & dist['high']),
        res['high']
    ),
    ctrl.Rule(illum['high'] & dist['high'], res['very_high']),
]

```

4. Моделирование работы системы

Сгенерируем 5 случайных пар (освещённость, дальность видимости):

```
# Generate 5 random pairs
illum_dist_pairs = [
    (
        np.random.uniform(low=0., high=12.),
        np.random.uniform(low=0., high=10.),
    ) for _ in range(5)
]
```

Каждую из пар входных значений можно «скормить» модели, которая рассчитает выходное значение с помощью нечёткой логики и выведет график выходного значения и функций принадлежности разным множествам.

```
for pair in illum_dist_pairs:
    illum_value, dist_value = pair
    print(f'illuminance: {illum_value}, distance: {dist_value}')

    # Assuming we have the computation model initialized
    resistance.input['illuminance'] = illum_value
    resistance.input['distance'] = dist_value
    resistance.compute()
    res_value = resistance.output['resistance']

    print(f'resistance: {resistance.output["resistance"]}')

    res.view(sim=resistance)
```

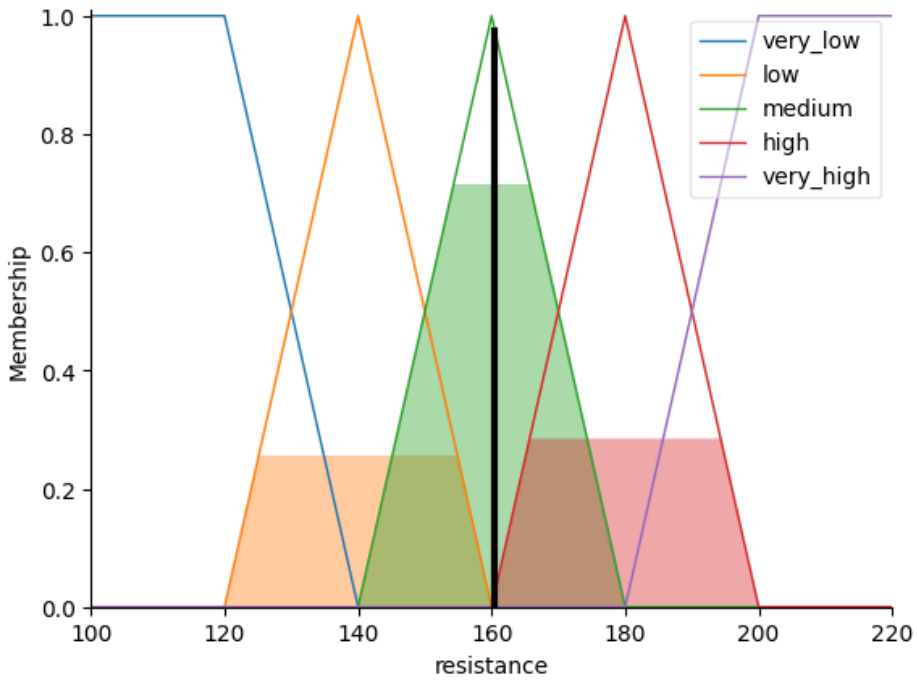

Рис. 2: Визуализация результата для $E \approx 2.42$ клк, $d \approx 8.72$ м, $R \approx 160.48$ Ом

Выводы

В ходе выполнения лабораторной работы №2 по дисциплине «Теория систем» были выполнены следующие действия:

1. Были выбраны две входные переменные — естественная освещённость и оптическая дальность, а также одна выходная переменная — сопротивление реостата (управляющее яркостью уличного освещения). Каждая из переменных измерялась в соответствующих единицах (лк для освещённости, м для дальности, Ом для сопротивления).

2. Для каждой лингвистической переменной были определены нечеткие подмножества, задаваемые треугольными и плечевыми функциями принадлежности. Таким образом, для естественной освещённости были введены терм-множества «слабая», «средняя» и «сильная», для оптической дальности — «малая», «средняя» и «большая», а для сопротивления — «очень низкое», «низкое», «среднее», «высокое» и «очень высокое». Графическое отображение функций принадлежности позволило убедиться в корректности выбранных параметров.

3. Был составлен набор из не менее чем четырех правил, охватывающих все возможные комбинации входных значений, что обеспечило полноту нечеткой базы знаний. Правила учитывали взаимосвязь между снижением естественной освещённости и уменьшением оптической дальности с целью увеличения яркости уличного освещения (снижение сопротивления реостата).

4. Система была протестирована на пяти различных наборах входных данных, заданных случайным образом. Результаты нечёткого вывода продемонстрировали адекватность работы системы: при низком уровне освещённости и малой оптической дальности требовалось минимальное сопротивление, а при высокой освещённости и большой дальности – наоборот, максимальное сопротивление. Полученные графики поверхности и распределения подтверждали логичность и предсказуемость результатов.

Исходный код блокнота доступен по ссылке.
Построенная нечеткая база знаний успешно моделирует задачу управления яркостью уличного освещения, учитывая комплексное влияние естественной освещённости и оптической дальности.
Графики функций принадлежности и 3D-график зависимости выходной переменной от входных значений продемонстрировали возможность качественной визуализации результатов нечеткого вывода, что является важным аспектом при анализе и оптимизации работы системы.
Полученные навыки могут быть использованы для разработки более сложных нечетких систем управления в других предметных областях, а также для интеграции нечетких систем с реальными системами управления.

Список использованных источников

1. Google Classroom