Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе № 2

по курсу «Основы искусственного интеллекта»

на тему: «Алгоритмы нечеткой логики»

Студент ИУ7-13М (Группа) _____ (Подпись, дата) Орду М. А. (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Строганов Ю. В. (И. О. Фамилия)

2025 г.

СОДЕРЖАНИЕ

1 Теоретическая часть

1.1 Постановка задачи

В одномерном пространстве ($X = 1$) рассматриваются два автомобиля: лидер, управляемый пользователем, и автомобиль-автопилот. Автопилот должен следовать за лидером, поддерживая постоянную дистанцию D, не имея информации о скорости лидера. Известно только текущее расстояние между автомобилями. Требуется определить необходимую скорость автопилота v_{auto} на основе нечеткого логического вывода.

Определение ускорения запрещено. Входными переменными являются:

- ошибка по расстоянию $e = D - D$;
- изменение ошибки $\Delta e = \frac{de}{dt}$.

Выходная переменная — скорость автопилота v_{auto}.

1.2 Этапы нечеткого логического вывода

Нечеткий логический вывод состоит из следующих этапов:

1. **Фаззификация** — преобразование четких входных значений e и Δe в степени принадлежности нечетким подмножествам.
2. **Применение базы правил** — вычисление степени активации каждого правила на основе входных значений.
3. **Импликация** — формирование выходных нечетких множеств в соответствии с вычисленной степенью активации α.
4. **Агрегация** — объединение всех выходных множеств.
5. **Дефаззификация** — преобразование агрегированного нечеткого множества в четкое значение v_{auto}.

1.3 Функции принадлежности

Для входных и выходных переменных были выбраны следующие функции принадлежности (рис. **??**):

- Ошибка расстояния `Negative`: `Zero`, `Positive`;
- Изменение ошибки `Negative`: `Zero`, `Positive`;
- Скорость `Slow`: `Medium`, `Fast`.

Для задания функций использовались треугольные формы:

$$\texttt{trimf}(x; a, b, c) = \begin{cases} 0, & x < a, \\ \frac{x-a}{b-a}, & a \leq x < b, \\ \frac{c-x}{c-b}, & b \leq x < c, \\ 0, & x \geq c. \end{cases}$$

1.4 База правил

Правила нечеткого вывода описывают зависимость между ошибками и требуемой скоростью:

1. ЕСЛИ distance_error = Positive И delta_distance = Negative ТО v_follower = Slow;
2. ЕСЛИ distance_error = Positive И delta_distance = Zero ТО v_follower = Slow;
3. ЕСЛИ distance_error = Positive И delta_distance = Positive ТО v_follower = Medium;
4. ЕСЛИ distance_error = Zero И delta_distance = Negative ТО v_follower = Medium;
5. ЕСЛИ distance_error = Zero И delta_distance = Zero ТО v_follower = Medium;
6. ЕСЛИ distance_error = Zero И delta_distance = Positive ТО v_follower = Fast;
7. ЕСЛИ distance_error = Negative И delta_distance = Negative ТО v_follower = Medium;

8. ЕСЛИ distance_error = Negative И delta_distance = Zero ТО v_follower = Fast;

9. ЕСЛИ distance_error = Negative И delta_distance = Positive ТО v_follower = Fast.

1.5 Алгоритм импликации Ларсена

Для варианта лабораторной работы используется импликация Ларсена:

$$\mu_{A \wedge B}(z) = \mu_A(x) \cdot \mu_B(y),$$

где:

$\mu_A(x)$ — функция принадлежности входной переменной x к множеству A;
$\mu_B(y)$ — функция принадлежности входной переменной y к множеству B.

1.6 Алгоритм дефаззификации

Для получения четкого значения скорости используется метод центра тяжести (centroid):

$$v = \frac{\int z \cdot \mu(z)\, dz}{\int \mu(z)\, dz}.$$

В качестве альтернативного метода можно применять метод среднего максимума (mean of maxima):

$$v = \frac{z_{\min} + z_{\max}}{2}, \quad z_{\min,\max} \in \{z | \mu(z) = \max(\mu)\}.$$

2 Практическая часть

2.1 Используемые инструменты

Для реализации нечеткой системы использована библиотека `scikit-fuzzy` и язык Python 3. Основные зависимости:

- `numpy` — численные вычисления;
- `matplotlib` — визуализация;
- `scikit-fuzzy` — функции принадлежности и дефаззификация.

2.2 Функции принадлежности

На рисунках 2.1-2.3 представлены функции принадлежности нечетких переменных.

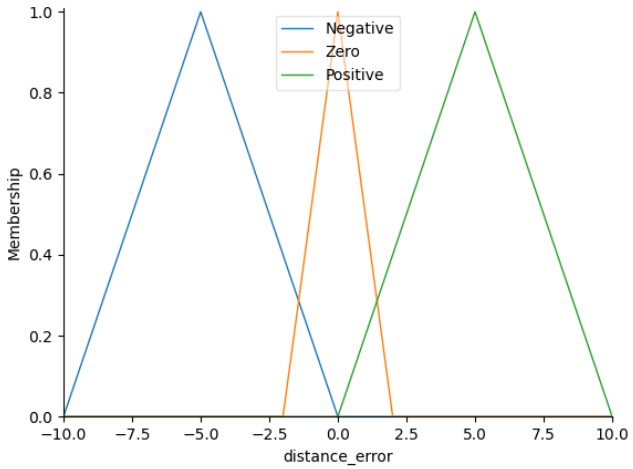

Рисунок 2.1 – Функция принадлежности расстояния между автомобилями

2.3 Реализация алгоритма Ларсена

В библиотеке scikit-fuzzy в качестве алгоритма логического вывода применяется алгоритм Мамдани, без возможности выбора альтернативы. Одним из вариантов реализации алгоритма Ларсена с использованием бибилиотеки scikit-fuzzy, является замена строк кода в исходном коде.

Листинг 2.1 – Исходный код библиотеки scikit-fuzzy

```
1 def __init__(self, antecedent=None, consequent=None, label=None,
    and_func=np.fmin, or_func=np.fmax):
```

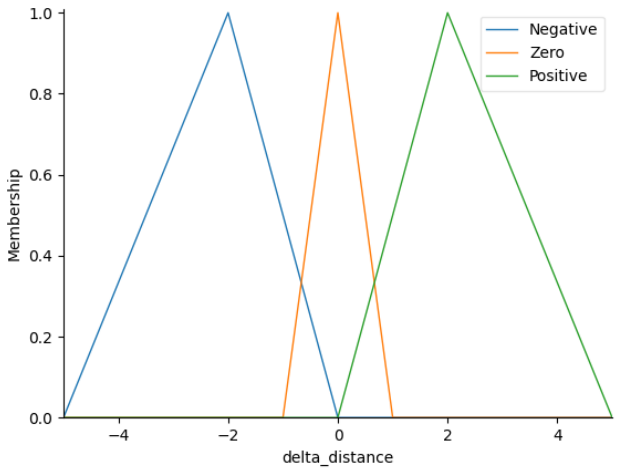

Рисунок 2.2 – Функция принадлежности изменения расстояния

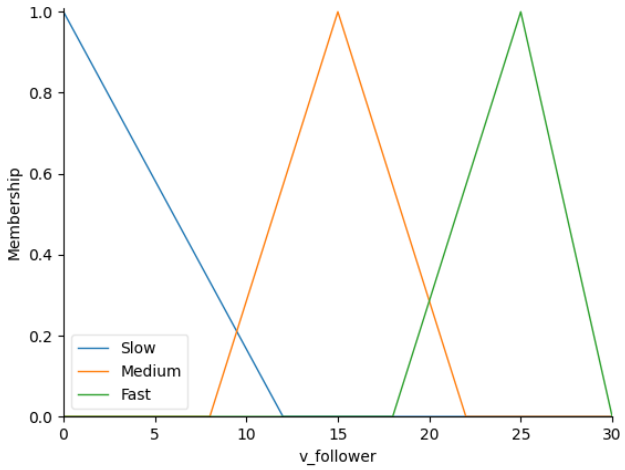

Рисунок 2.3 – Функция принадлежности скорости автопилота

Как видно из листинга 2.1, выходом функции логического И является минимальное из двух значений `and_func = np.fmin` (по Мамдани). Заменив этот кусок кода на `and_func=np.multiply`, получим алгоритм вывода Ларсена.

2.4 Результаты

На рисунке ?? показана динамика движения автомобилей. Автопилот плавно регулирует скорость, удерживая требуемую дистанцию от лидера даже при изменении его скорости.

Среднеквадратичная ошибка поддержания дистанции составила:

$$MSE = 0.153.$$

Время моделирования при шаге интегрирования $dt = 0.1$ и $T = 100\,c$ составило менее 1 секунды.

2.5 Особенности реализации

- Импликация Ларсена реализована вручную как масштабирование функции принадлежности по вертикали.
- Используется произведение степеней истинности для расчета силы активации ($\alpha = \mu_1 \times \mu_2$).
- Дефаззификация выполняется методом центра тяжести (`fuzz.defuzz(..., 'centroid')`).
- Система легко расширяется на $X = 3$ при добавлении новых входных переменных (например, бокового смещения).

ПРИЛОЖЕНИЕ А

Листинг А.1 – Исходный код программы

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

Distance error = ctrl.Antecedent(np.arange(-10, 10.1, 0.1),
    'Distance error')
delta_distance = ctrl.Antecedent(np.arange(-5, 5.1, 0.1),
    'delta_distance')
v_follower = ctrl.Consequent(np.arange(0, 30.1, 0.1),
    'v_follower')

Distance error['Negative'] = fuzz.trimf(Distance error.universe,
    [-10, -5, 0])
Distance error['Zero'] = fuzz.trimf(Distance error.universe,
    [-2, 0, 2])
Distance error['Positive'] = fuzz.trimf(Distance error.universe,
    [0, 5, 10])

delta_distance['Negative'] = fuzz.trimf(delta_distance.universe,
    [-5, -2, 0])
delta_distance['Zero'] = fuzz.trimf(delta_distance.universe,
    [-1, 0, 1])
delta_distance['Positive'] = fuzz.trimf(delta_distance.universe,
    [0, 2, 5])

v_follower['Slow'] = fuzz.trimf(v_follower.universe, [0, 0, 12])
v_follower['Medium'] = fuzz.trimf(v_follower.universe, [8, 15,
    22])
v_follower['Fast'] = fuzz.trimf(v_follower.universe, [18, 25,
    30])

rule1 = ctrl.Rule(Distance error['Positive'] &
    delta_distance['Negative'], v_follower['Slow'])
rule2 = ctrl.Rule(Distance error['Positive'] &
    delta_distance['Zero'], v_follower['Slow'])
rule3 = ctrl.Rule(Distance error['Positive'] &
```

```
    delta_distance['Positive'], v_follower['Medium'])

rule4 = ctrl.Rule(Distance error['Zero'] &
    delta_distance['Negative'], v_follower['Medium'])
rule5 = ctrl.Rule(Distance error['Zero'] &
    delta_distance['Zero'], v_follower['Medium'])
rule6 = ctrl.Rule(Distance error['Zero'] &
    delta_distance['Positive'], v_follower['Fast'])

rule7 = ctrl.Rule(Distance error['Negative'] &
    delta_distance['Negative'], v_follower['Medium'])
rule8 = ctrl.Rule(Distance error['Negative'] &
    delta_distance['Zero'], v_follower['Fast'])
rule9 = ctrl.Rule(Distance error['Negative'] &
    delta_distance['Positive'], v_follower['Fast'])

fuzzy_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4,
    rule5, rule6, rule7, rule8, rule9])
fuzzy_sim = ctrl.ControlSystemSimulation(fuzzy_ctrl)

dt = 0.1
t = np.arange(0, 60, dt)
d_ref = 5.0

# v_leader = 15 + 4 +(t*0) #np.sin(0.15 * t)

# --- Ступенька ---
v_leader = 13 + 3 * (t >= 20)



x_leader = np.zeros_like(t)
x_follower = np.zeros_like(t)
v_auto = np.zeros_like(t)

x_leader[0] = 0
x_follower[0] = -10
v_auto[0] = 0
error_int = 0
Ki = 0.5

```

```
for i in range(1, len(t)):
    x_leader[i] = x_leader[i-1] + v_leader[i-1] * dt
    x_follower[i] = x_follower[i-1] + v_auto[i-1] * dt

    distance = x_leader[i] - x_follower[i]
    error = distance - d_ref
    delta_error = (x_leader[i] - x_leader[i-1]) - (x_follower[i]
       - x_follower[i-1])

    error_int += error * dt
    e_fuzzy = error + Ki * error_int

    fuzzy_sim.input['Distance error'] = e_fuzzy
    fuzzy_sim.input['delta_distance'] = delta_error
    fuzzy_sim.compute()

    v_auto[i] = 0.5 * v_auto[i-1] + 1.5*
       fuzzy_sim.output['v_follower']

plt.figure(figsize=(12, 8))

plt.subplot(3, 1, 1)
plt.plot(t, [x_leader[i] - x_follower[i] - d_ref for i in
   range(len(t))])
plt.title('Ошибка по дистанции')
plt.ylabel('Ошибка (м)')
plt.grid()

plt.subplot(3, 1, 2)
plt.plot(t, x_leader, label='Лидер')
plt.plot(t, x_follower, label='Автопилот')
plt.title('Координаты автомобилей')
plt.ylabel('Позиция (м)')
plt.legend()
plt.grid()

plt.subplot(3, 1, 3)
plt.plot(t, v_leader, label='V лидера')
plt.plot(t, v_auto, label='V автопилота')
plt.title('Скорости автомобилей')
plt.ylabel('Скорость (м/с)')
```

```
96 plt.xlabel('Время (с)')
97 plt.legend()
98 plt.grid()
99
100 plt.tight_layout()
101 plt.show()
```