

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**З В І Т**

**Лабораторна робота №2  
з дисципліни  
«Комп'ютерні системи  
штучного інтелекту»**

Виконавець:

студент групи КІ-22м

Косей М.П.

Керівник:

викладач

Саяпін В.Г.

2023

## Лабораторна робота №2

**Тема:** Моделювання роботи світлофора засобами нечіткої логіки (Fuzzy Logic)

**Мета:** Одержання практичних навиків реалізації систем підтримки прийняття рішень на базі нечіткої логіки

**Завдання:** Виконати моделювання роботи світлофора на базі нечіткої логіки у відповідності із варіантами завдань

### ХІД РОБОТИ

- 1) Ознайомитись з теоретичними відомостями до лабораторної роботи.

### Визначення нечіткої логіки

Fuzzy Logic - це метод математичного моделювання, що використовується для обробки нечіткої інформації та прийняття рішень на основі нечітких даних. Він дозволяє враховувати незнання, нечіткість, сумнів та інші аспекти, які зазвичай ігноруються в традиційних методах прийняття рішень.

Fuzzy Logic знайшов застосування в багатьох галузях, таких як промисловість, транспорт, медицина та інше. Він дозволяє створювати системи, які можуть працювати з даними, які не є точними.

Наприклад, Fuzzy Logic може бути використаний в системах автоматичного керування, які потрібно приймати рішення на основі даних з сенсорів, які можуть мати певну похибку.

Крім того, Fuzzy Logic використовується в системах штучного інтелекту, таких як експертні системи та системи прийняття рішень. Він дозволяє розробляти імітаційні моделі, які можуть аналізувати складні ситуації та приймати рішення на основі нечітких даних.

Загалом, Fuzzy Logic відкриває широкі можливості для розвитку нових технологій та систем, які можуть працювати з нечіткою та незнаною інформацією.

### Архітектура системи нечіткої логіки

У архітектурі системи нечіткої логіки кожен компонент відіграє важливу роль. Архітектура складається з чотирьох різних компонентів (рис. 1), які перелічені далі

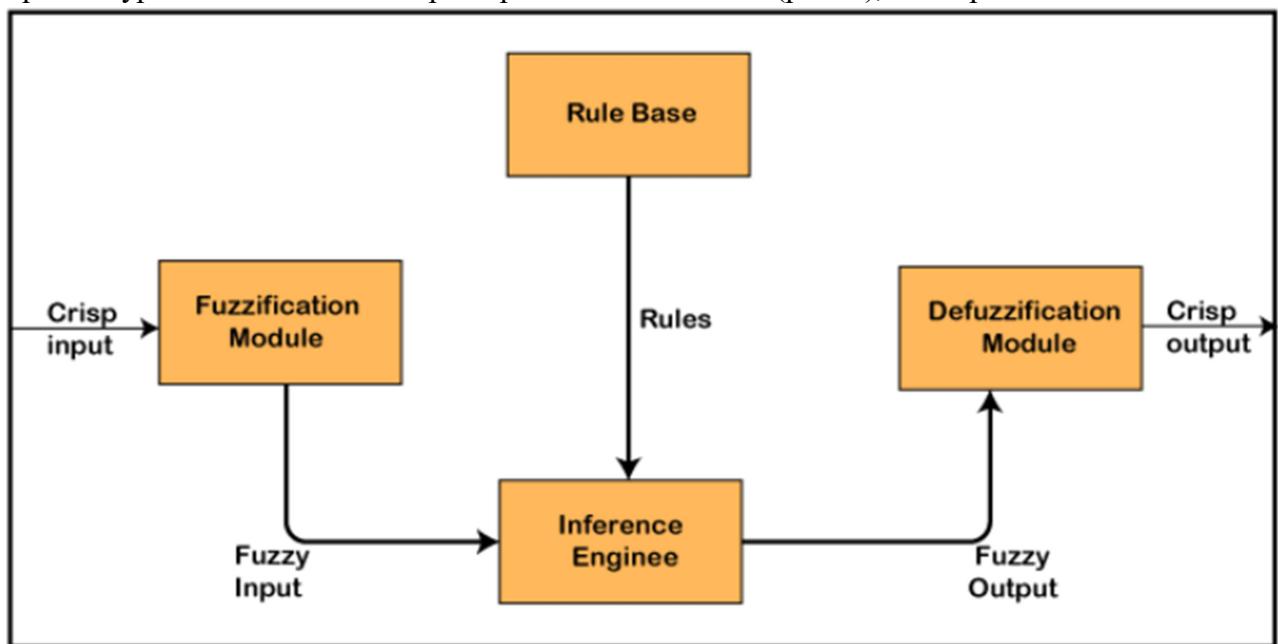


Рисунок 1. Архітектура системи нечіткої логіки

## **1.База правил (Rule Base)**

База правил – це компонент містить набір правил та умов IF-THEN, наданих експертами для управління системою прийняття рішень на основі лінгвістичної інформації. Останні розробки в теорії нечіткої логіки пропонують декілька ефективних методів для проектування та налаштування нечітких контролерів. Більшість з цих розробок зменшують кількість нечітких правил

## **2.Фузифікація (Fuzzification)**

Використовується для перетворення вхідних даних, які є чіткими числами, в нечіткі множини. Чіткі числа - це точні вхідні дані, які вимірюються датчиками та передаються в систему управління для обробки, такі як температура, тиск, оберти на хвилину тощо.

## **3.Інференційний двигун (Inference Engine)**

Цей компонент визначає ступінь відповідності поточного нечіткого введення кожному правилу та визначає, які правила мають бути запущені відповідно до вхідного поля. Потім запущені правила комбінуються, щоб створити керуючі дії.

## **4.Дефузифікація (Defuzzification)**

Використовується для перетворення нечітких множин, отриманих за допомогою інференційного двигуна, в чітке значення. Існує декілька методів дефузифікації, із яких користувач повинен вибрати найкращий для зменшення помилок в конкретній експертній системі.

## **Види систем нечіткої логіки**

Мамдані (Mamdani) та Сугено (Sugeno) - це два типи систем нечіткої логіки.

Система нечіткої логіки Мамдані базується на нечітких правилах, які визначають зв'язок між вхідними та вихідними змінними. Вихідний сигнал генерується шляхом комбінування правил за допомогою операторів нечіткої логіки.

Система нечіткої логіки Сугено використовує лінійні функції для визначення зв'язку між вхідними та вихідними змінними. Вихідний сигнал є зваженою сумою вхідних змінних.

Система Мамдані більш підходить у випадках, коли вихідний сигнал є нелінійним і правила базуються на експертних знаннях.

Система Сугено більш підходить у випадках, коли вихідний сигнал є лінійним і правила базуються на числових даних.

Обидві системи мають свої переваги та недоліки, і вибір між ними залежить від конкретної проблеми, яку необхідно вирішити.

Наприклад, система Мамдані часто використовується у системах керування, тоді як система Сугено часто використовується у визначені закономірностей та виділенні патернів.

## **Стандарти для описання та роботи з системами нечіткої логіки**

Нечітка логіка може використовуватись для керування процесами в промисловій автоматизації за допомогою програмованих логічних контролерів (PLC). Контролери зазвичай працюють за допомогою програмного забезпечення, яке дозволяє програмувати їх для виконання певних завдань. У деяких випадках контролери можуть використовувати нечітку логіку для керування процесами, що дозволяє забезпечити більш точне та ефективне управління системою.

Наприклад, стандарт «IEC 61131-7» визначає бібліотеку перевикористовуваних програмних компонентів для використання в програмованих логічних контролерах, включаючи функціональні блоки для використання нечіткої логіки.

Крім того, існують інші стандарти та фреймворки для роботи з нечіткою логікою в промисловій автоматизації, такі як «IEEE std 1855-2016».

«IEEE STANDARD 1855-2016, IEEE Standard for Fuzzy Markup language» (FML) - це технічний стандарт, розроблений «IEEE Standards Association».

FML дозволяє моделювати систему нечіткої логіки у формі, зручній для сприйняття людиною та незалежній від апаратної частини. FML базується на мові розмітки XML

(eXtensible Markup Language).

В FML інженери систем нечіткої логіки мають уніфіковану та високорівневу методологію для опису взаємозамінних систем нечіткої логіки. IEEE STANDARD 1855-2016 використовує мову опису синтаксису та семантики програм заснованих на FML - W3C XML Schema definition language.

До введення FML спеціалісти з нечіткої логіки могли обмінюватись інформацією про свої нечіткі алгоритми, додаючи до свого програмного забезпечення можливість читання, правильного синтаксичного аналізу та збереження результатів роботи у формі, сумісній з мовою керування нечітким контролем (Fuzzy Control Language, FCL), описаної та визначененої частиною 7 стандарту IEC 61131.

FML дозволяє людям кодувати системи нечіткої логіки за допомогою збірки семантичних тегів, які моделюють компоненти класичного нечіткого контролера, таких як база знань, база правил, нечіткі змінні та нечіткі правила.

## Програмне забезпечення (ПЗ) для роботи з системами нечіткої логіки

Існує багато програмного забезпечення, яке дозволяє працювати з системами нечіткої логіки (рисунок 2).

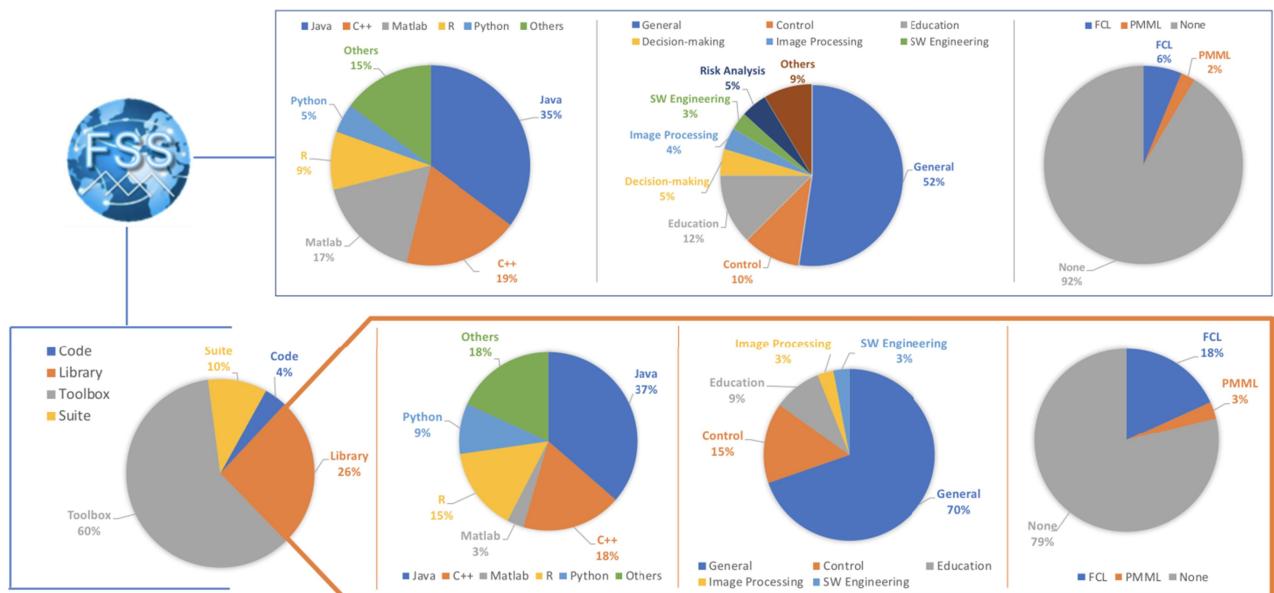


FIGURE 2. Overview on fuzzy systems software. On the top, statistics related to all types of software. On the bottom, details on libraries.

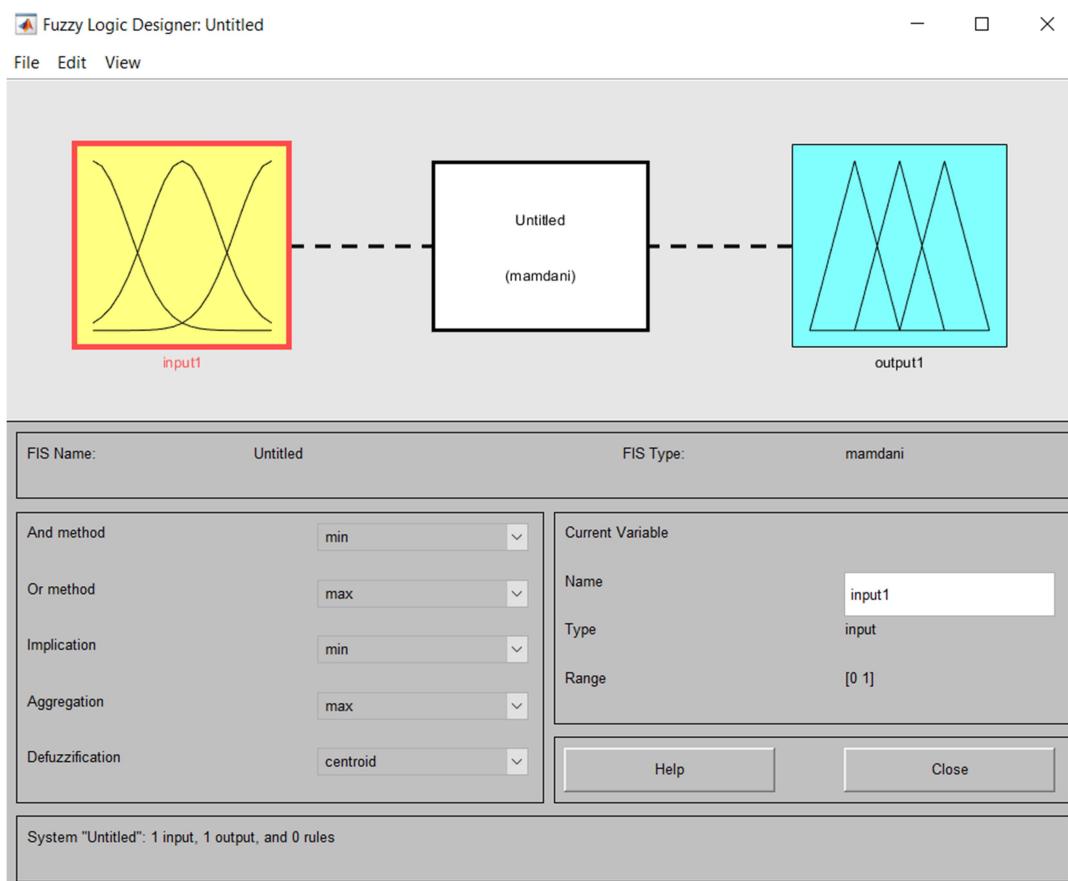
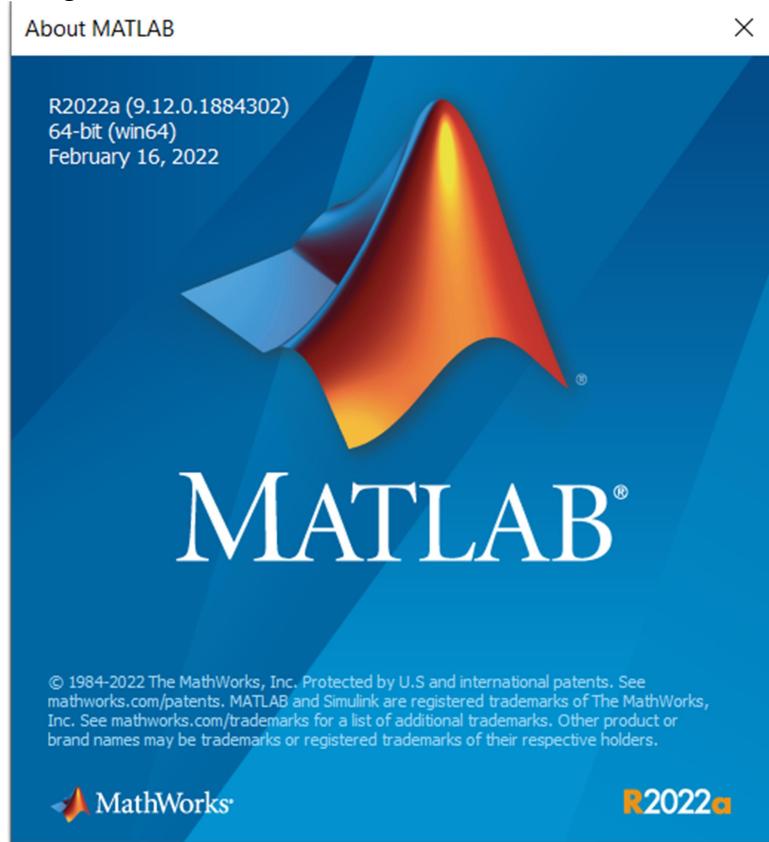
Рисунок 2. Огляд програмного забезпечення для роботи з системами нечіткої логіки (2018 рік, [http://www.uco.es/JFML/paper//2018\\_JFML.pdf](http://www.uco.es/JFML/paper//2018_JFML.pdf))

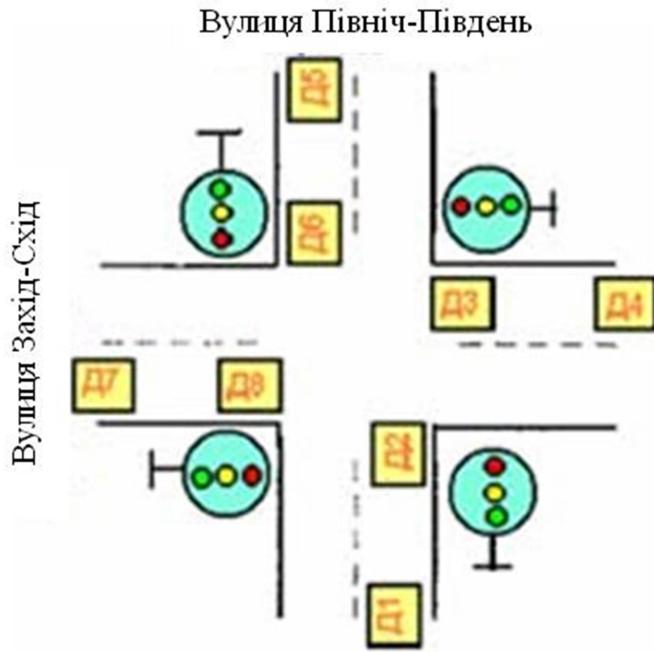
Ось декілька прикладів:

- MATLAB Fuzzy Logic Toolbox - це інструментарій MATLAB для розробки та реалізації систем нечіткої логіки.
- FuzzyLite - це безкоштовна бібліотека C++, яка дозволяє розробляти та реалізувати системи нечіткої логіки.
- JavaFuzzy - це фреймворк Java для розробки та реалізації систем нечіткої логіки.
- Soft Computing Toolkit - це набір інструментів для MATLAB, який містить бібліотеку функцій для роботи з нечіткою логікою та нейронними мережами.

**2) Моделювання роботи світлофора на базі нечіткої логіки за допомогою пакета «Matlab R2022a».**

Моделювання роботи світлофора виконується за допомогою ПЗ «Matlab R2022a» в додатку «Fuzzy Logic Designer».





Світлофор використовує різницю показань чотирьох пар датчиків: (Д1-Д2), (Д 3-Д4), (Д 5-Д6) і (Д 7-Д8).

Таким чином, якщо для вулиці ПнПв горить зелене світло, машини проїжджають перехрестя й показання двох пар датчиків рівні:  $D1=D2$ ,  $D5=D6$ , а, отже, їхня різниця дорівнює нулю.

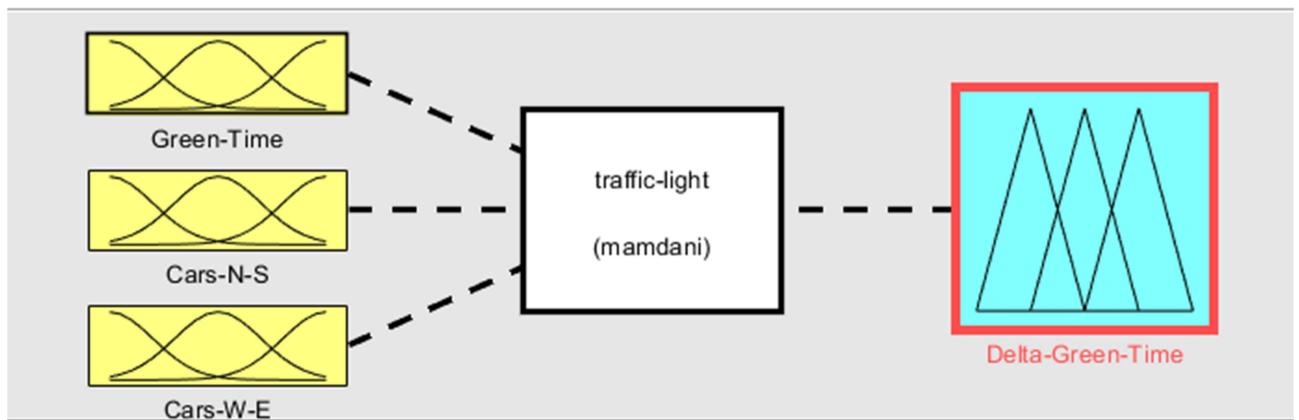
У цей же час на вулиці ЗС перед світлофором зупиняються машини, які встигли проїхати тільки  $D4$  і  $D7$ . У результаті можна розрахувати сумарну кількість автомобілів на цій вулиці в такий спосіб:

$$(D4 - D3) + (D7 - D8) = (D4 - 0) + (D7 - 0) = D4 + D7.$$

Оскільки робота світлофора залежить від числа машин на обох вулицях і поточному часі зеленого світла, для нашої підпрограмми пропонується використовувати 3 входи (вхідні змінні):

- число машин на вулиці ПнПв по закінченню чергового циклу (**Cars-N-S**);
- число машин на вулиці ЗС по закінченню циклу (**Cars-W-E**);
- час зеленого світла нечіткого світлофора (**Green-Time**).

Так як суттю роботи світлофора є зміна часу роботи зеленого світла, то як вихідний параметр (вихідна змінна) пропонується використовувати величину цієї зміни (**Delta-Green-Time**).

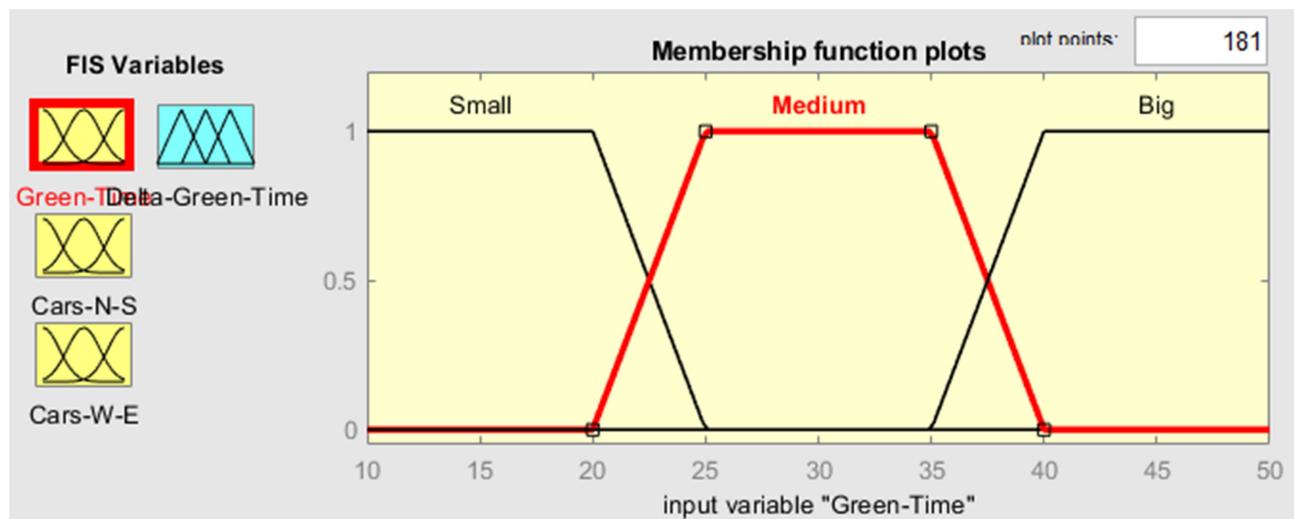


Так, для змінної **Green-Time** пропонується використовувати три терми:

- мале (10-25 сек.) (**Small**);
- середнє (20-40 сек.) (**Medium**);

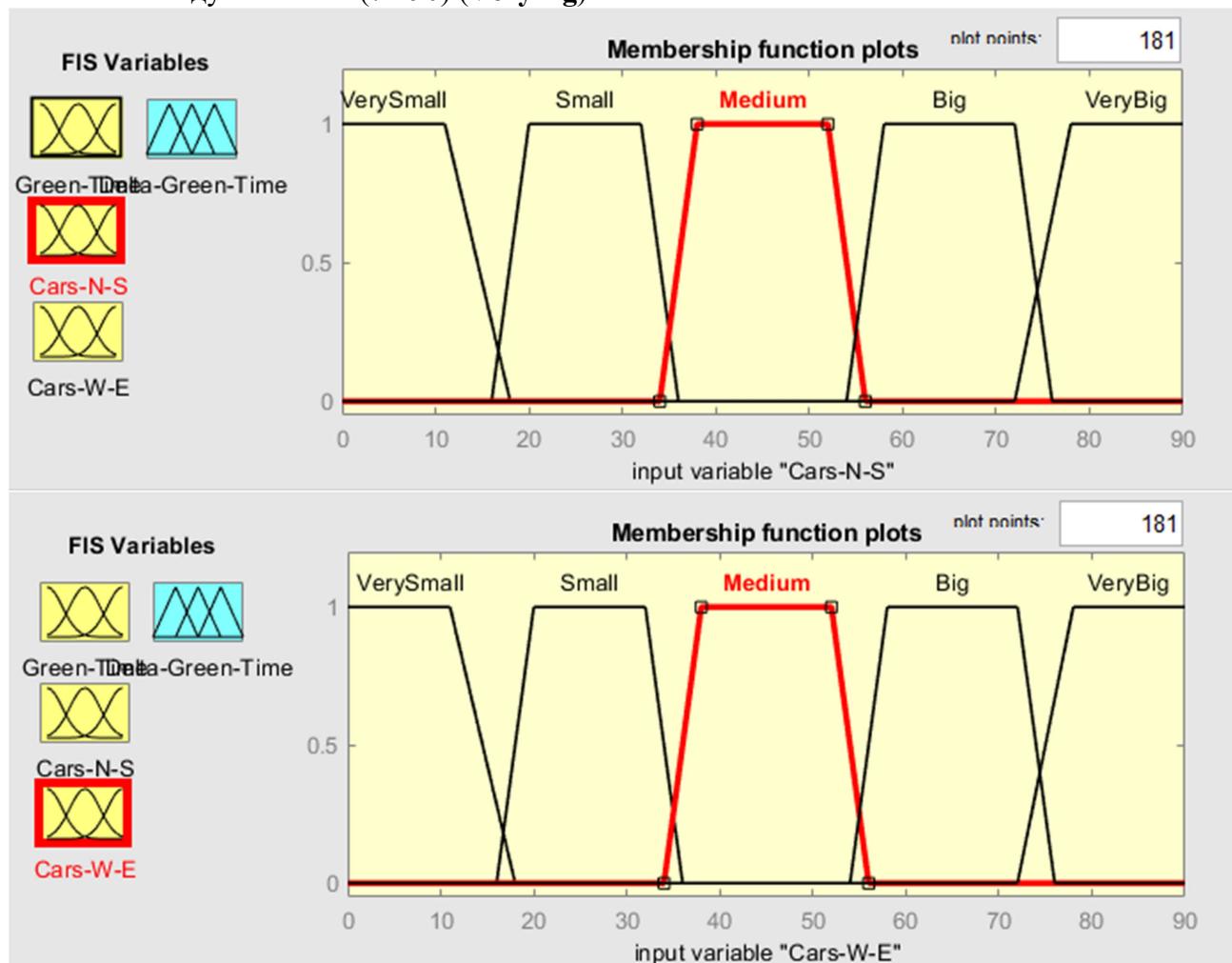
➤ велике(35-50сек.) (**Big**).

Ступінь приналежності чітких значень термам задається за допомогою функцій приналежності (у нашому випадку ці функції мають форму трапеції).



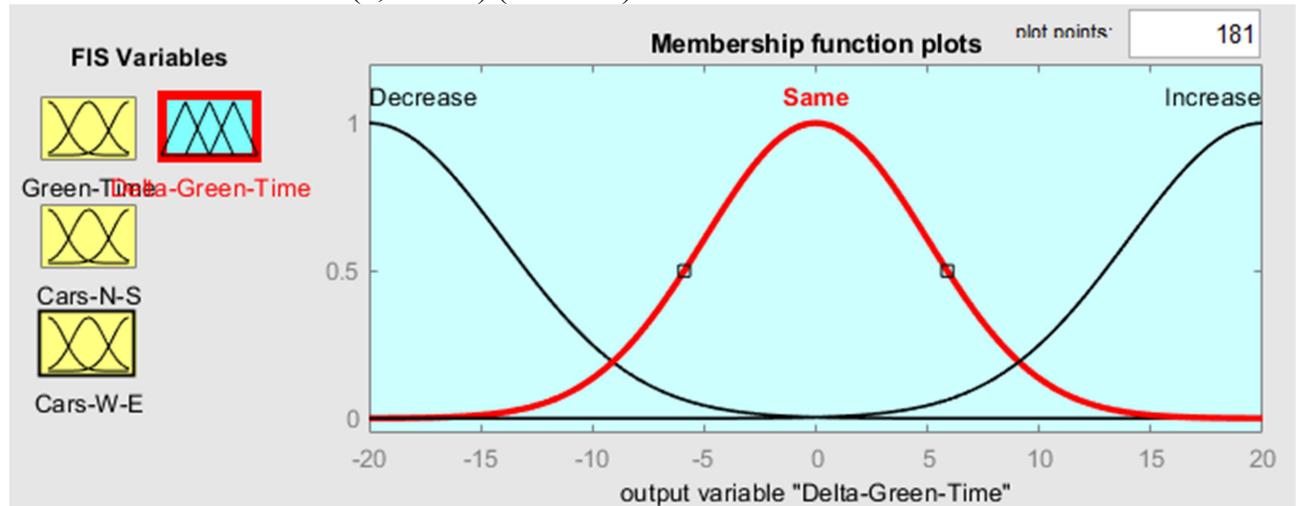
Аналогічно, терми для двох інших змінних матимуть наступні значення (функції приналежності тут також мають форму трапеції):

- дуже мале (0-18) (**VerySmall**);
- мале (16-36) (**Small**);
- середнє (34-56) (**Medium**);
- велике (54-76) (**Big**);
- дуже велике (72-90) (**VeryBig**).



Для вихідного параметру (вихідна змінна) **Delta-Green-Time** - терми будуть наступні (функції приналежності мають форму Гаусса):

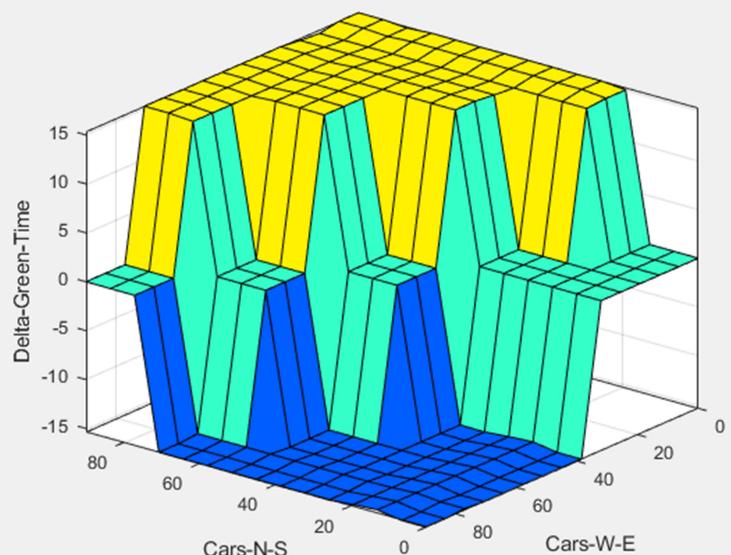
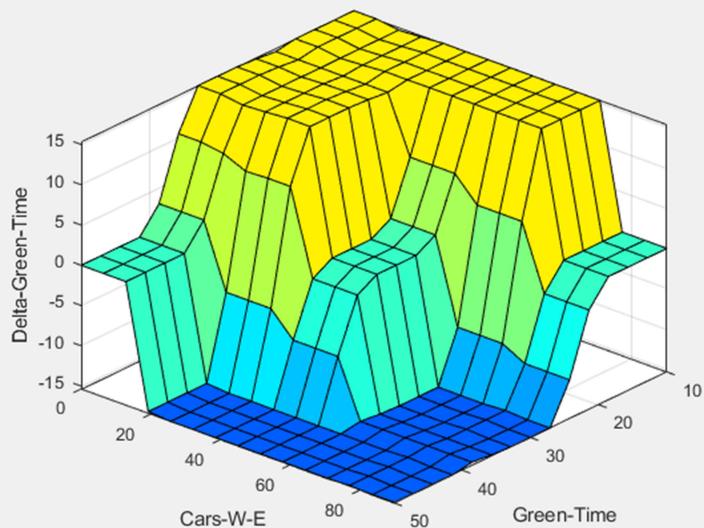
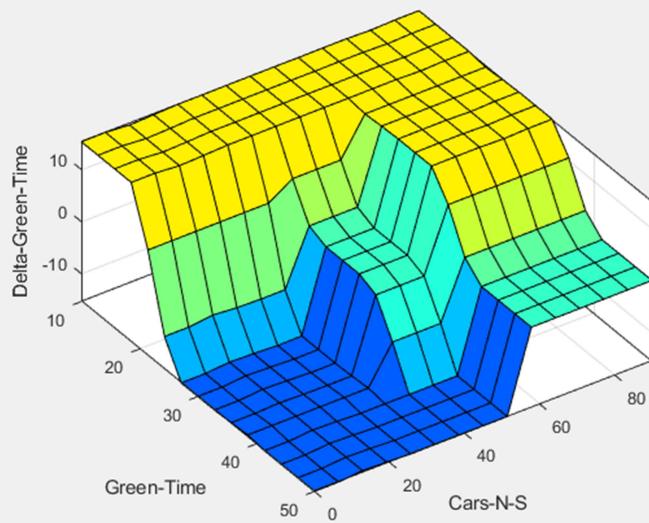
- зменшити (-20; 0 сек.) (**Decrease**);
  - не змінювати (-15; 15сек.) (**Same**);
  - збільшити (0; 20сек.) (**Increase**)



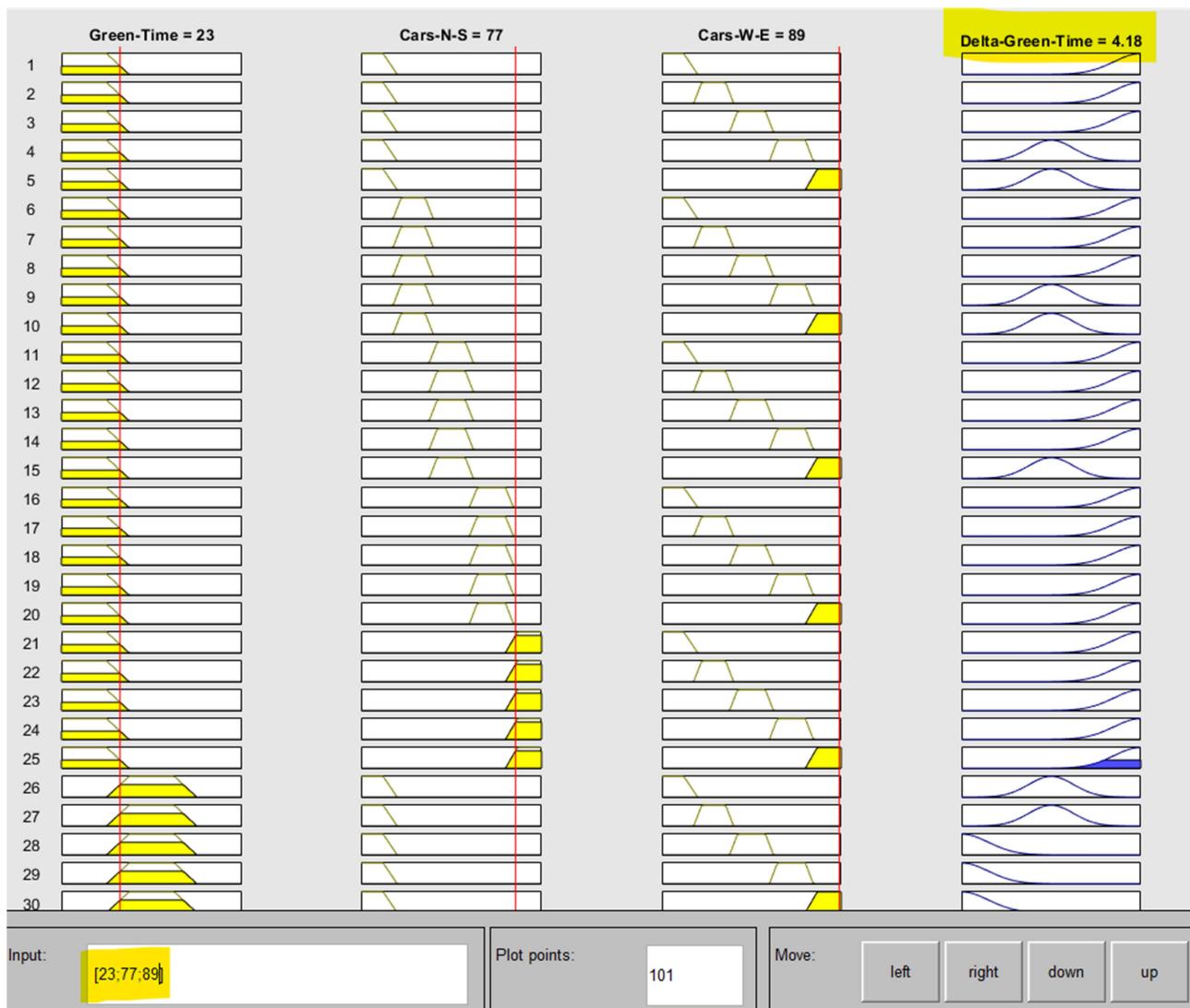
Додаємо правила до бази правил (в кількості  $75=3\times 5\times 5$ ).

The screenshot shows the Rule Editor interface with the title "Rule Editor: traffic-light". The menu bar includes File, Edit, View, and Options. The main area displays a large list of 75 rules, each consisting of an "If" clause followed by an "and" connector, another "If" clause, a "Then" connector, and a "Delta-Green-Time is" clause. The "If" clauses involve variables like Green-Time, Cars-N-S, and Cars-W-E with operators like is, is Small, is Medium, is Big, or is VeryBig. The "Then" clauses specify the Delta-Green-Time as Decrease, Same, Increase, or none. Below the list is a configuration panel with dropdown menus for Green-Time, Cars-N-S, Cars-W-E, and Delta-Green-Time, each with options for Small, Medium, Big, VeryBig, and none. There are also checkboxes for "not" under each variable and a "not" checkbox under Delta-Green-Time. At the bottom, there are buttons for Connection (radio buttons for "or" and "and"), Weight (set to 1), Delete rule, Add rule, Change rule, and navigation buttons << and >>. A status message "The rule is added" is visible at the bottom left, and help and close buttons are at the bottom right.

## Будуємо поверхні



Змодулюємо роботу світлофора для варіанта завдання №8 для значень вхідних змінних: **Green-Time** - 23, **Cars-N-S** - 77, **Cars-W-E** – 89; визначаємо значення вихідної змінної **Delta-Green-Time** – 4.18.



### 3) Реалізувати програмно роботу нечіткого світлофора .

Для програмної реалізації роботи нечіткого світлофора обираємо фреймворк **jfuzzylite™ 6.0** (<https://www.fuzzylite.com/java/>).

Описуємо інференційний двигун.

```
fuzzyLogicEngine.setName("traffic-light");
fuzzyLogicEngine.setDescription("traffic-light");
```

Додаємо вхідні та вихідні змінні.

```
private void addDeltaGreenTimeVariable() { 1 usage new *
    OutputVariable DeltaGreenTime = new OutputVariable();
    DeltaGreenTime.setName("DeltaGreenTime");
    DeltaGreenTime.setDescription("DeltaGreenTime");
    DeltaGreenTime.setEnabled(true);
    DeltaGreenTime.setRange(-20.000, 20.000);
    DeltaGreenTime.setLockValueInRange(false);
    DeltaGreenTime.setAggregation(new Maximum());
    DeltaGreenTime.setDefuzzifier(new Centroid(100));
    DeltaGreenTime.setDefaultValue(Double.NaN);
    DeltaGreenTime.setLockPreviousValue(false);
    DeltaGreenTime.addTerm(new Gaussian("Decrease", -20.000, 5.997));
    DeltaGreenTime.addTerm(new Gaussian("Same", 0.000, 5.000));
    DeltaGreenTime.addTerm(new Gaussian("Increase", 20.000, 6.000));
    fuzzyLogicEngine.addOutputVariable(DeltaGreenTime);
}

private void addCarsWEVariable() { 1 usage new *
    InputVariable CarsWE = new InputVariable();
    CarsWE.setName("CarsWE");
    CarsWE.setDescription("CarsWE");
    CarsWE.setEnabled(true);
    CarsWE.setRange(0.000, 90.000);
    CarsWE.setLockValueInRange(false);
    CarsWE.addTerm(new Trapezoid("VerySmall", 0.000, 0.000, 11.000, 18.000));
    CarsWE.addTerm(new Trapezoid("Small", 16.000, 20.000, 32.000, 36.000));
    CarsWE.addTerm(new Trapezoid("Medium", 34.000, 38.000, 52.000, 56.000));
    CarsWE.addTerm(new Trapezoid("Big", 54.000, 58.000, 72.000, 76.000));
    CarsWE.addTerm(new Trapezoid("VeryBig", 72.000, 78.000, 90.000, 90.000));
    fuzzyLogicEngine.addInputVariable(CarsWE);
}
```

```

private void addGreenTimeVariable() { 1 usage new *
    InputVariable GreenTime = new InputVariable();
    GreenTime.setName("GreenTime");
    GreenTime.setDescription("");
    GreenTime.setEnabled(true);
    GreenTime.setRange(10.000, 50.000);
    GreenTime.setLockValueInRange(false);
    GreenTime.addTerm(new Trapezoid("Small", 10.000, 10.000, 20.000, 25.000));
    GreenTime.addTerm(new Trapezoid("Medium", 20.000, 25.000, 35.000, 40.000));
    GreenTime.addTerm(new Trapezoid("Big", 35.000, 40.000, 50.000, 50.000));
    fuzzyLogicEngine.addInputVariable(GreenTime);
}

private void addCarsNSVariable() { 1 usage new *
    InputVariable CarsNS = new InputVariable();
    CarsNS.setName("CarsNS");
    CarsNS.setDescription("CarsNS");
    CarsNS.setEnabled(true);
    CarsNS.setRange(0.000, 90.000);
    CarsNS.setLockValueInRange(false);
    CarsNS.addTerm(new Trapezoid("VerySmall", 0.000, 0.000, 11.000, 18.000));
    CarsNS.addTerm(new Trapezoid("Small", 16.000, 20.000, 32.000, 36.000));
    CarsNS.addTerm(new Trapezoid("Medium", 34.000, 38.000, 52.000, 56.000));
    CarsNS.addTerm(new Trapezoid("Big", 54.000, 58.000, 72.000, 76.000));
    CarsNS.addTerm(new Trapezoid("VeryBig", 72.000, 78.000, 90.000, 90.000));
    fuzzyLogicEngine.addInputVariable(CarsNS);
}

}

```

Додаємо правила Мамдані

```

private void addMamdaniRules() { 1 usage new *
    RuleBlock ruleBlock = new RuleBlock();
    ruleBlock.setName("mamdani");
    ruleBlock.setDescription("Mamdani inference");
    ruleBlock.setEnabled(true);
    ruleBlock.setConjunction(new Minimum());
    ruleBlock.setDisjunction(new Maximum());
    ruleBlock.setImplication(new Minimum());
    ruleBlock.setActivation(new General());
    ruleBlock.addRule(Rule.parse("if GreenTime is Small and CarsNS is VerySmall and CarsWE is VerySmall " +
        "then DeltaGreenTime is Increase", fuzzyLogicEngine));
    ruleBlock.addRule(Rule.parse("if GreenTime is Small and CarsNS is VerySmall and CarsWE is Small " +
        "then DeltaGreenTime is Increase", fuzzyLogicEngine));

    ruleBlock.addRule(Rule.parse("if GreenTime is Big and CarsNS is VeryBig and CarsWE is Big " +
        "then DeltaGreenTime is Decrease", fuzzyLogicEngine));
    ruleBlock.addRule(Rule.parse("if GreenTime is Big and CarsNS is VeryBig and CarsWE is VeryBig " +
        "then DeltaGreenTime is Decrease", fuzzyLogicEngine));
    fuzzyLogicEngine.addRuleBlock(ruleBlock);
}

}

```

Змодулюємо роботу світлофора для варіанта завдання №8 для значень вхідних змінних: **Green-Time** - 23, **Cars-N-S** - 77, **Cars-W-E** – 89.

```
public static void main( String[] args ){    no usages  new *  
  
    TrafficLight trafficLight = new TrafficLight();  
    System.out.println("GreenTime - 23");  
    trafficLight.setGreenTime(23);  
  
    System.out.println("CarsNS - 77");  
    trafficLight.setCarsNS(77);  
  
    System.out.println("CarsWS - 89");  
    trafficLight.setCarsWE(89);  
  
    System.out.println("DeltaGreenTime - " + trafficLight.getDeltaGreenTime());  
  
}  
C:\Users\Max11mus\.jdks\azul-11.0.12\bin\java.exe -agentlib:jdwp=transport=dt_socket  
Connected to the target VM, address: '127.0.0.1:55933', transport: 'socket'  
GreenTime - 23  
CarsNS - 77  
CarsWS - 89  
DeltaGreenTime - 4.0862062228234635  
Disconnected from the target VM, address: '127.0.0.1:55933', transport: 'socket'  
  
Process finished with exit code 0
```

Значення вихідної змінної **Delta-Green-Time** – 4.09 ( **MatLab** - 4.18).

## **ВИСНОВКИ**

**В результаті виконання лабораторної роботи було розглянуто поняття фреймів, та написана програма реалізації фреймів на мові програмування JAVA.**

**Усі матеріали викладені у репозіторії GitHub, за посиланням <https://github.com/Max11mus/Artifition-Intelect-Lab2>.**