МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
"ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

О. Ю. Заковоротний, Т. О. Орлова,

Д. В. Гриньов, В. М. Сергієнко

ОСНОВИ ОБЧИСЛЮВАЛЬНОГО ІНТЕЛЕКТУ

Лабораторний практикум
для студентів денної та заочної форм навчання за спеціальністю
123 "Комп’ютерна інженерія"

Харків
НТУ «ХПІ»
2023

УДК 004.89
ББК 32.973-02
З 19

Рецензенти:

В. Д. Ковальов, доктор технічних наук, професор, Лауреат премії Кабінету Міністрів України, ректор Донбаської державної машинобудівної академії;

О. А. Серков, доктор технічних наук, професор, заслужений винахідник України, академік Академії наук прикладної радіоелектроніки, НТУ "ХПІ"

Рекомендовано вченою радою Національного технічного університету «Харківський політехнічний інститут» для студентів за спеціальністю 123 «Комп'ютерна інженерія» (протокол № 6 від 23.09.2022 р.)

Заковоротний О. Ю. Основи обчислювального інтелекту / З 19 О. Ю. Заковоротний, Т. О. Орлова, Д. В. Гриньов, В. М. Сергієнко: лабораторний практикум. – Харків: НТУ «ХПІ», 2023. – 170 с.

ISBN 978-617-05-0438-8

Наведено теоретичні відомості про пакет MATLAB та способи побудови в ньому нечітких множин, алгоритмів нечіткого виведення й нечіткої кластеризації, нечіткого контролера, генетичних алгоритмів і нейронечітких гібридних мереж. Теоретичний матеріал підкріплений великою кількістю прикладів із використання описаних нечітких систем.

Призначено для студентів денної та заочної форм навчання за спеціальністю "Комп'ютерна інженерія".

Іл. 89. Табл. 11. Бібліог. 16 назв.

УДК 004.89
ББК 32.973-02

ISBN 978-617-05-0438-8

ЗМІСТ

ВСТУП

Теорія нечітких множин (*fuzzy sets theory*) була розроблена в 1965 р., коли професор Лотфі Заде з університету Берклі опублікував основоположну роботу Fuzzy Sets в журналі Information and Control. Прикметник "fuzzy", який можливо перекласти на українську мову як нечіткий, розмитий, введений в назву нової теорії з метою дистанціювання від традиційної чіткої математики і аристотелевої логіки, що оперують з чіткими поняттями: "належить – не належить", "правда – не правда". Концепція нечітких множин зародилася у Л. Заде "як незадоволення математичними методами класичної теорії систем, яке змушувало домагатися штучної точності, недоречної у багатьох системах реального світу, особливо в так званих гуманістичних системах, що включають людей".

Початком практичного застосування теорії нечітких множин можна вважати 1975 р., коли Е. Мамдані і С. Ассіліан побудували перший нечіткий контроллер для управління простим паровим двигуном. В 1982 р. Л. Холмблад і Д. Остергад розробили перший промисловий нечіткий контролер, який був впроваджений в управління процесом випалення цементу на заводі в Данії. Успіх першого промислового контролера, основаного на нечітких лінгвістичних правилах "Якщо – то" привів до сплеску інтересу до теорії нечітких множин серед математиків і інженерів. Дещо пізніше Бартоломеєм Коско була доведена теорема про нечітку апроксимацію (*Fuzzy Approximation Theorem*), згідно якої будь-яка математична система може бути апроксимована системою, що базується на нечіткій логікі. Іншими словами, за допомогою природно-мовних висловлювань-правил "Якщо – то", з подальшою їх формалізацією засобами теорії нечітких множин, можна скільки завгодно точно відбити довільний взаємозв'язок "входи – вихід" без використання складного апарату

диференціального і інтегрального числення, традиційно вживаного в управлінні та ідентифікації.

Системи, що базуються на нечітких множинах, розроблені і успішно впроваджені в таких областях, як управління технологічними процесами, управління транспортом, медична діагностика, технічна діагностика, фінансовий менеджмент, біржове прогнозування, розпізнавання образів і т.д. Спектр додатків дуже широкий – від відеокамер і побутових пральних машин до засобів наведення ракет ППО і управління бойовими літальними апаратами. Практичний досвід розробки систем нечіткого логічного висновку свідчить, що терміни і вартість їх проектування значно менші, ніж при використанні традиційного математичного апарату, при цьому забезпечується необхідний рівень робастності і прозорості моделей.

Лабораторна робота 1

ДОСЛІДЖЕННЯ СПОСОБІВ ФОРМУВАННЯ НЕЧІТКИХ МНОЖИН І ОПЕРАЦІЙ НАД НИМИ

Мета лабораторної роботи: отримання і закріплення знань, формування практичних навичок побудови нечітких множин з використанням різних типів функцій приналежності. Ознайомлення з найбільш поширеними логічними операціями над нечіткими множинами.

1.1. Короткі відомості з теорії

1.1.1. Функції приналежності

Трикутна функція приналежності

Інструментарій нечіткої логіки (ІНЛ) у складі пакету MATLAB містить 11 вбудованих типів функцій приналежності (ФП), що формуються на основі лінійних функцій, розподілу Гауса, сигмоїдної кривої, квадратичних і кубічних поліноміальних кривих. До найбільш простих ФП можна віднести трикутну і трапецієвидну. Найменування трикутної ФП – *trimf* (*triangle membership function*). У параметричному виді вона є не що інше, як набір трьох точок, що утворюють трикутник.

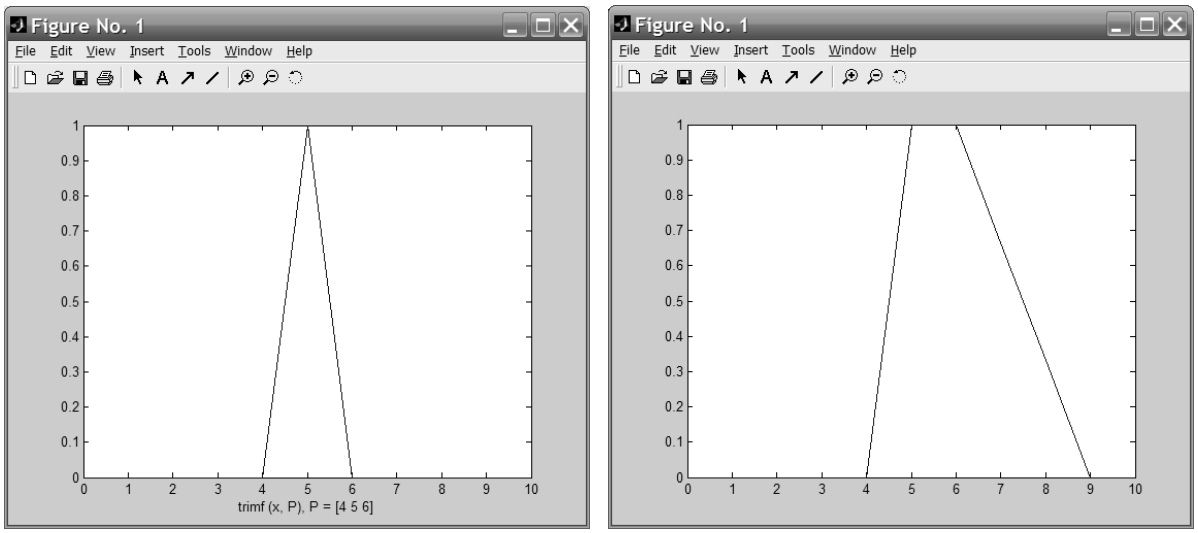
Опис функції:

$$y = trimf\,(x, [a, b, c]),$$

де вектор x – базова множина, на якій визначається ФП. Величини a і c задають основу трикутника, b – його вершину.

У аналітичному виді трикутна ФП може бути задана таким чином (рис. 1.1, *а*):

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



а *б*

Рис. 1.1. Трикутна (а) і трапецієвидна (б) функції приналежності

Далі розглянемо приклади використання різних ФП в системі.

Приклади є фрагментами програм і коментарів на мові пакету MATLAB.

Приклад 1.1. Програма використання ФП *trimf*.

```
» x = 0 : 0,1 : 10;
» y = trimf (x, [3 6 8]);
» plot (x, y);
» xlabel ('trimf (x, P), P = [3 6 8]');
```

Трапецієвидна функція приналежності

Трапецієвидна ФП – *trapmf* (*trapezoid membership function*) – відрізняється від попередньої функції лише тим, що має верхню основу.

Опис функції:

$$y = trapmf\,(x, [a, b, c, d]),$$

де параметри a і d – нижня основа трапеції; b і c – верхня основа трапеції (рис. 1.1, *б*).

Аналітичний запис трапецієвидної функції має вигляд:

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

Проста і двостороння функція приналежності Гауса

На основі функції розподілу Гауса можна побудувати ФП двох видів: просту функцію належності Гауса і двосторонню, утворену за допомогою різних функцій розподілу Гауса. Перша з них позначається *gaussmf*, а друга – *gauss2mf*.

Опис функції:

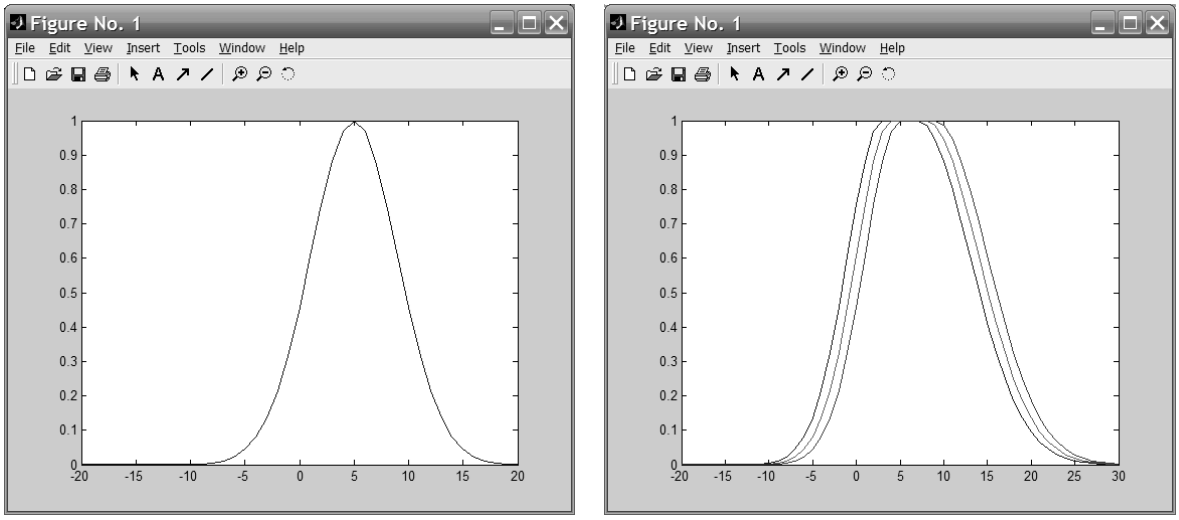
$$y = gaussmf\,(x, [\sigma, c]).$$

Приклад 1.2. Програма використання ФП *gaussmf*.

```
» X = -20 : 1 : 20;
» Y = gaussmf (x, [4 5]);
» plot (x, y);
```

Симетрична функція Гауса залежить від двох параметрів σ і c (рис. 1.2, *а*):

$$f(x,\sigma,c)=e^{\frac{-(x-c)^2}{2\sigma^2}}.$$



а *б*

Рис. 1.2. Проста (а) і двостороння (б) функції приналежності Гауса

Опис функції:

$$y = gauss2mf\,(x, [\sigma_1, c_1, \sigma_2, c_2]).$$

Останній вираз є комбінацією двох різних функцій розподілу Гауса. Перша визначається параметрами σ_1 і c_1 і задає форму лівої сторони, а друга (параметри σ_2, c_2) – правої сторони ФП.

Якщо $c_1 < c_2$, то в цьому випадку функція *gauss2mf* досягає свого максимального значення на рівні 1. Інакше – максимальне значення функції менше 1 (рис. 1.2, *б*).

Приклад 1.3. Програма використання ФП *gauss2mf*.

```
» x= [-20 : 30]';
» y1 = gauss2mf (x, [4 3 6 7]);
» y2 = gauss2mf (x, [4 4 6 8]);
» y3 = gauss2mf (x, [4 5 6 9]);
» plot (x,[y1 y2 y3]);
```

Символ «'» у рядку визначення базової множини *x* показує транспонованість базової множини.

Функція приналежності "узагальнений дзвін"

Наступною функцією, яка дозволяє представляти нечіткі суб'єктивні переваги, є ФП "узагальнений дзвін" (рис. 1.3) і позначається *gbellmf* (*generalized bell shape membership function*).

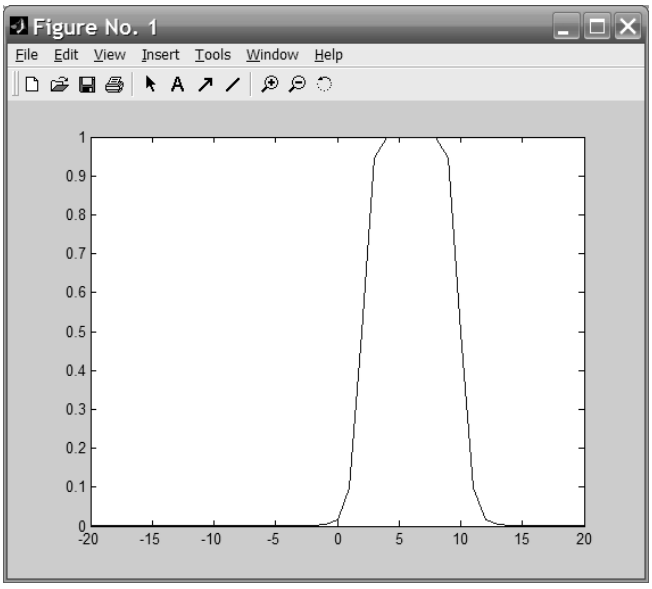


Рис. 1.3. Функція належності "узагальнений дзвін"

Її відмінність від розглянутих раніше ФП полягає в додаванні третього параметра, що дозволяє здійснювати плавний перехід між нечіткими множинами.

Опис функції:

$$y = gbellmf\,(x, [a, b, c]).$$

Функція "узагальнений дзвін" залежить від трьох параметрів і має наступний аналітичний запис:

$$f(x,a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}},$$

де c – визначає розташування центру ФП; a і b – роблять вплив на форму кривої (рис. 1.3).

Приклад 1.4. Програма використання *gbellmf*.

```
» x= [-20 : 20];
» y = gbellmf (x, [4 5 6]);
» plot (x,y);
```

Функції належності на основі функції розподілу Гауса і ФП "узагальнений дзвін" відрізняються гладкістю і простотою запису і є найбільш використовуваними при описі нечітких множин. Попри те, що гаусові і дзвіноподібні ФП мають властивість гладкості, вони не дозволяють формувати асиметричні ФП. Для цих цілей передбачений набір сигмоїдних функцій, які можуть бути відкриті або ліворуч, або праворуч залежно від типу функції. Симетричні і закриті функції синтезують з використанням двох додаткових сигмоїдів.

Сигмоїдні функції приналежності

Основна сигмоїдна ФП позначається *sigmf*, а додаткові – *dsigmf* і *psigmf*.

Опис основної сигмоїдної функції:

$$y = sigmf\,(x, [a, c]).$$

В аналітичній формі сигмоїдна функція *sigmf* записується наступним чином:

$$f(x,a,c) = \frac{1}{1+e^{-a(x-c)}}.$$

Залежно від знака параметра a ФП, що розглядається, буде відкрита або праворуч, або ліворуч (рис. 1.4, *а*), що дозволить застосовувати її при описі таких нечітких понять, як "дуже великий", "вкрай негативно" та інших.

Опис додаткової сигмоїдної функції:

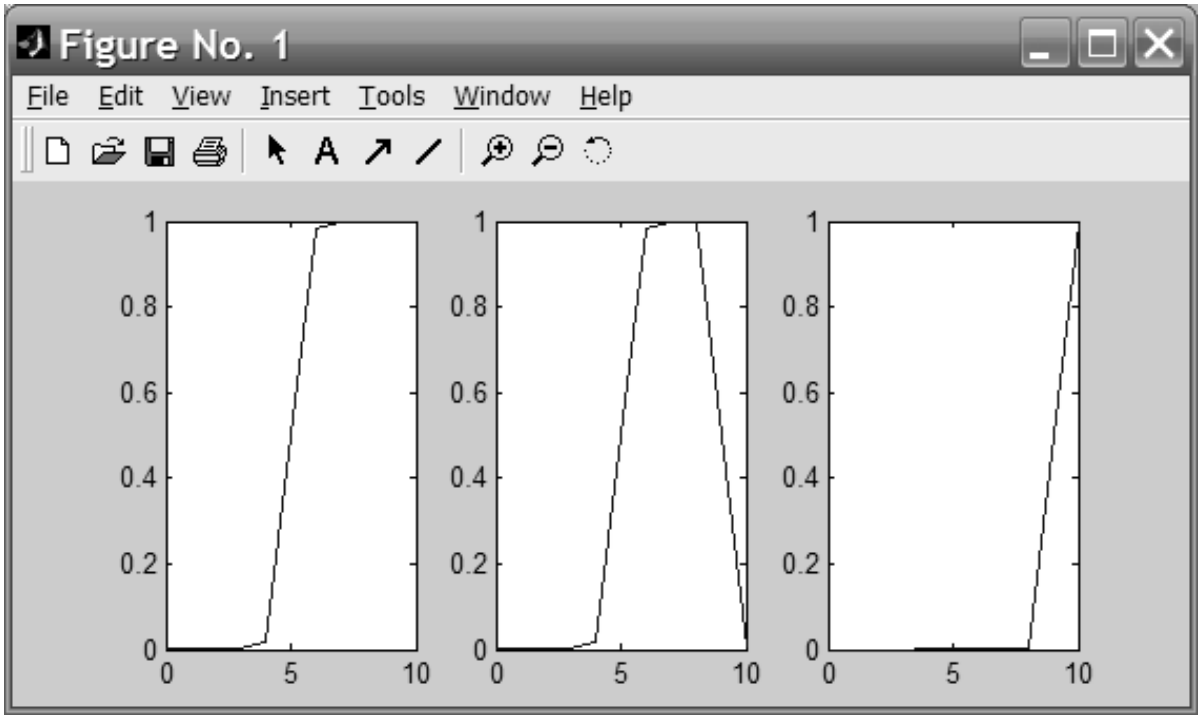
$$y = dsigmf\,(x, (a_1, c_1, a_2, c_2]) .$$

ФП *dsigmf* залежить від чотирьох параметрів a_1, c_1, a_2, c_2 та визначається як різниця двох сигмоїдних функцій: $f(x, a_1, c_1) - f(x, a_2, c_2)$ (рис. 1.4, *б*).

Опис додаткової сигмоїдної функції:

$$y = psigmf\,(x, [a_1, c_1, a_2, c_2]) .$$

ФП *psigmf*, так само, як і попередня функція, залежить від чотирьох параметрів a_1, c_1, a_2, c_2 та визначається як добуток двох сигмоїдних функцій $f(x, a_1, c_1) \cdot f(x, a_2, c_2)$ (рис. 1.4, *в*).



а *б* *в*

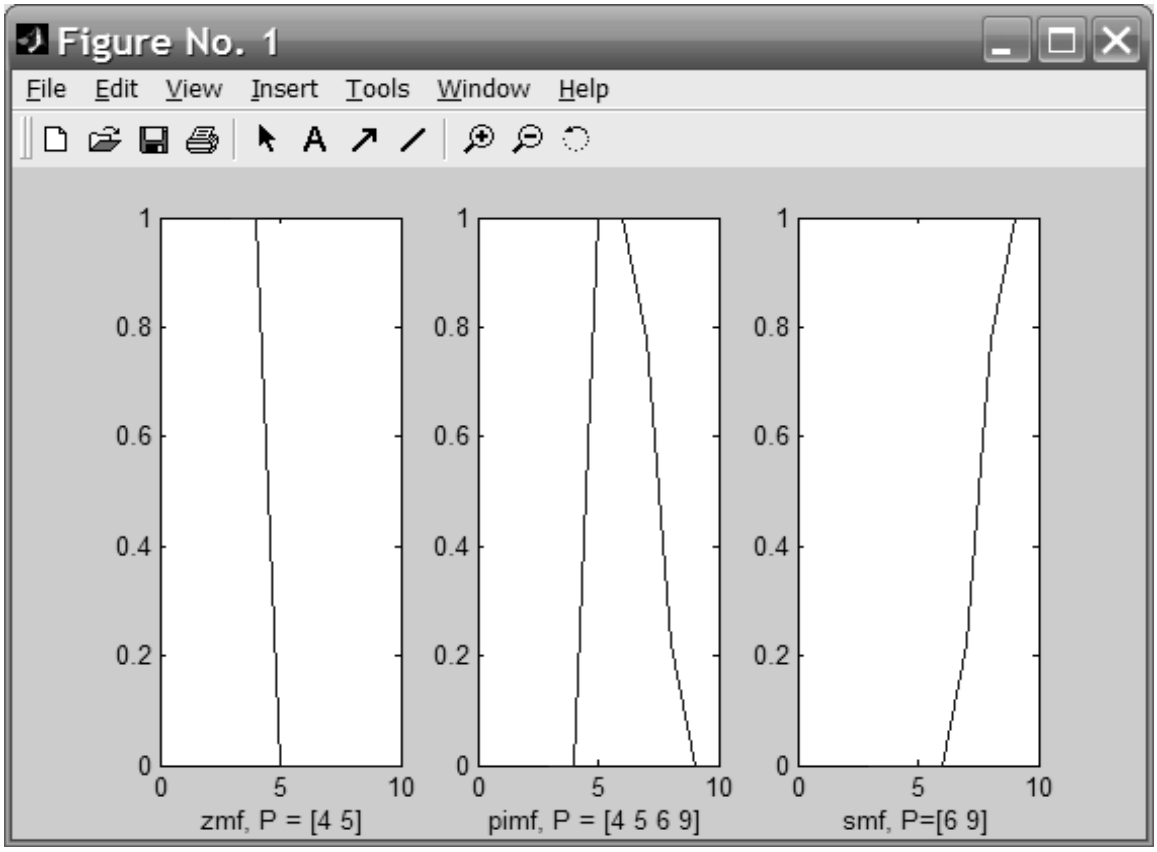
Рис. 1.4. Сигмоїдні функції належності

Приклад 1.5. Програма використання сигмоїдних функцій.

```
» x= [0 : 10];
» subplot (1, 3, 1);
» y=sigmf (x,[4 5]);
» plot (x, y);
» subplot (1, 3, 2);
» y = dsigmf (x, [4 5 6 9]);
» plot (x, y);
» subplot (1, 3, 3);
» y = psigmf (x, [4 5 6 9]);
» plot (x, y);
```

Функції приналежності на основі поліноміальних кривих

Інструментарій нечіткої логіки (*fuzzy logic toolbox*) в складі MATLAB дає можливість формування ФП на основі поліноміальних кривих. Відповідні функції називаються *Z*-функції (*zmf*), *PI*-функції (*pimf*) і *S*-функції (*smf*). Функція *zmf* являє собою асиметричну поліноміальну криву, відкриту зліва (рис. 1.5, *а*), функція *smf* – дзеркальне відображення функції *zmf* (рис. 1.5, *в*). Відповідно функція *pimf* дорівнює нулю в правій і лівій границях і набуває значення, що дорівнює одиниці, в середині деякого відрізка (рис. 1.5, *б*).



а *б* *в*

Рис. 1.5. Поліноміальні функції належності

Опис функції:

$$y = zmf(x, [a, b]).$$

Параметри *a* і *b* визначають екстремальні значення кривої (рис. 1.5, *а*).

Опис функції:

$$y = pimf\,(x, [a, b, c, d]).$$

Параметри *a* і *d* задають перехід функції в нульове значення, а параметри *b* і *c* – в одиничне (рис. 1.5, *б*).

Опис функції:

$$y = smf\,(x, [a, b]).$$

Параметри *a* і *b* визначають екстремальні значення кривої (рис. 1.5, *в*).

Приклад 1.6. Програма використання поліноміальних кривих.

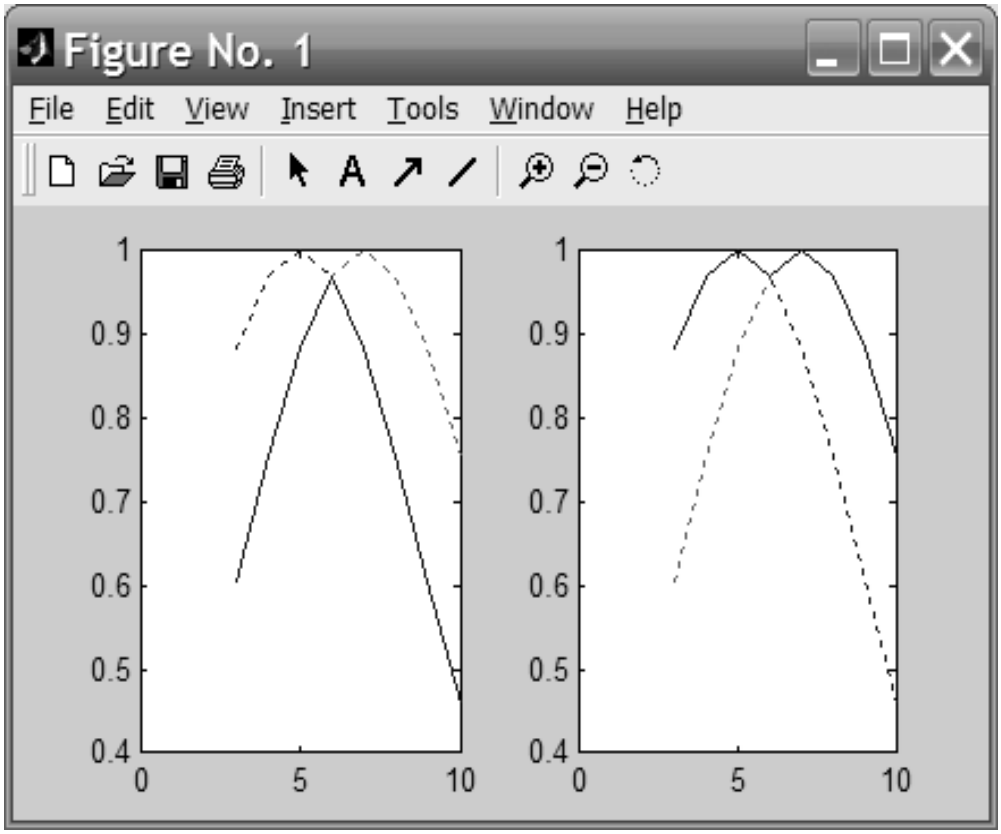
```
» x= [3 : 10];
» subplot(1, 3, 1);
» y = zmf(x, [4 5]);
» plot (x, y);
» xlabel (' zmf, P = [4 5]');
» subplot (1, 3, 2);
» y = pimf(x, [4 5 6 9]);
» plot (x, y);
» xlabel ('pimf, P = [4 5 6 9]');
» subplot (1, 3, 3);
» y = smf (x, [6 9]);
» plot(x, y);
» xlabel ('smf, P=[6 9]')
```

Крім розглянутих вище функцій, що дозволяють представляти нечіткі множини, у MATLAB є можливість формувати власні ФП або модифікувати вбудовані.

1.1.2. Операції з нечіткими множинами

Виділяють три основні логічні операції з нечіткими множинами: кон'юнкцію, диз'юнкцію та логічне заперечення. У середовищі MATLAB існує можливість визначати кон'юнктивні та диз'юнктивні оператори з погляду мінімаксної та ймовірнісної інтерпретацій.

Розглянемо мінімаксну інтерпретацію логічних операторів, де кон'юнктивний оператор представляє знаходження мінімуму – *min* (рис. 1.6, *а*), а диз'юнктивний - максимуму – *max* (рис. 1.6, *б*).



а *б*

Рис. 1.6. Перетин (*а*) та об'єднання (*б*) нечітких множин (мінімаксна інтерпретація)

Опис кон'юнктивної функції: $y = min([y_1; y_2])$.

Опис диз'юнктивної функції: $y = max([y_1; y_2])$.

Параметри y_1 і y_2 є вихідними ФП. Функція *min* працює зі списком ФП. У MATLAB список оформляється квадратними дужками, а елементи списку поділяються крапкою з комою.

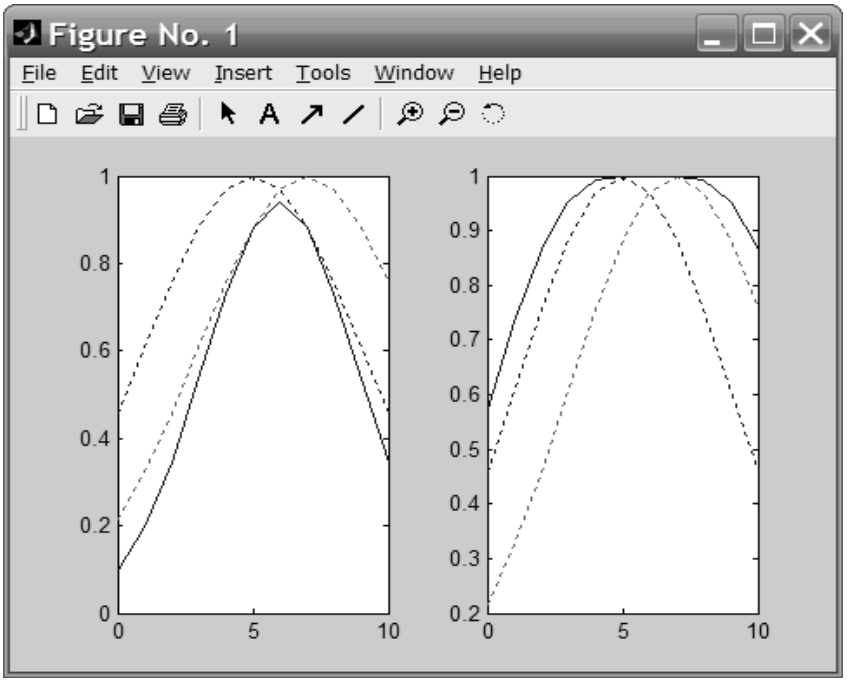
Приклад 1.7. Програма використання операцій *min* і *max*.

```
» x = 3 : 0,1 : 10;
» subplot(1, 2, 1);
» y1 = gaussmf(x, [4 5]);
» y2 = gaussmf(x, [4 7]);
» y3 = min([y1; y2]);
» plot (x, [y1; y2],':');
» hold on;
» plot(x, y3);
» hold off;
» subplot(1, 2, 2);
» y4 = max([y1; y2]);
» plot(x, [y1; y2], ':');
» hold on;
» plot(x, y4);
» hold off.
```

Пунктирною лінією на графіках зображені вихідні ФП, а суцільною лінією – результат дії логічних операторів.

Мінімаксна інтерпретація є найпоширенішою при побудові нечітких систем. Проте, практично досить часто використовується альтернативна ймовірнісна інтерпретація кон'юнктивних і диз'юнктивних операторів. MATLAB містить відповідні функції.

В рамках даної інтерпретації кон'юнктивний оператор є оператором обчислення алгебраїчного добутку – *prod* (рис. 1.7, *а*), а диз'юнктивний оператор – оператор обчислення алгебраїчної суми – *probor* (рис. 1.7, *б*).



а *б*

Рис. 1.7. Перетин (*а*) та об'єднання (*б*) нечітких множин (імовірнісна інтерпретація)

Опис функції: $y = prod([y_1; y_2])$

Опис функції: $y = probor([y_1; y_2])$.

Параметри y_1 і y_2 є вихідними ФП.

Приклад 1.8. Програма використання ймовірнісних операторів кон'юнкції та диз'юнкції.

```
» x = 0 : 0,1 : 10;
» subplot (1, 2, 1);
» y1 = gaussmf (x, [4 5]);
» y2 = gaussmf (x, [4 7]);
```

```
» y3 = prod ([y1; y2]);
» plot (x, [y1; y2],':');
» hold on;
» plot(x, y3);
» hold off;
» subplot (1, 2, 2);
» y4 = probor ([y1; y2]);
» plot (x, [y1; y2], ':');
» hold on;
» plot(x, y4);
» hold off.
```

Доповнення нечіткої множини є не що інше, як математичне подання вербального виразу «НЕ *A*» (рис. 1.8), де *A* – нечітка множина, що описує деяке розмите судження.

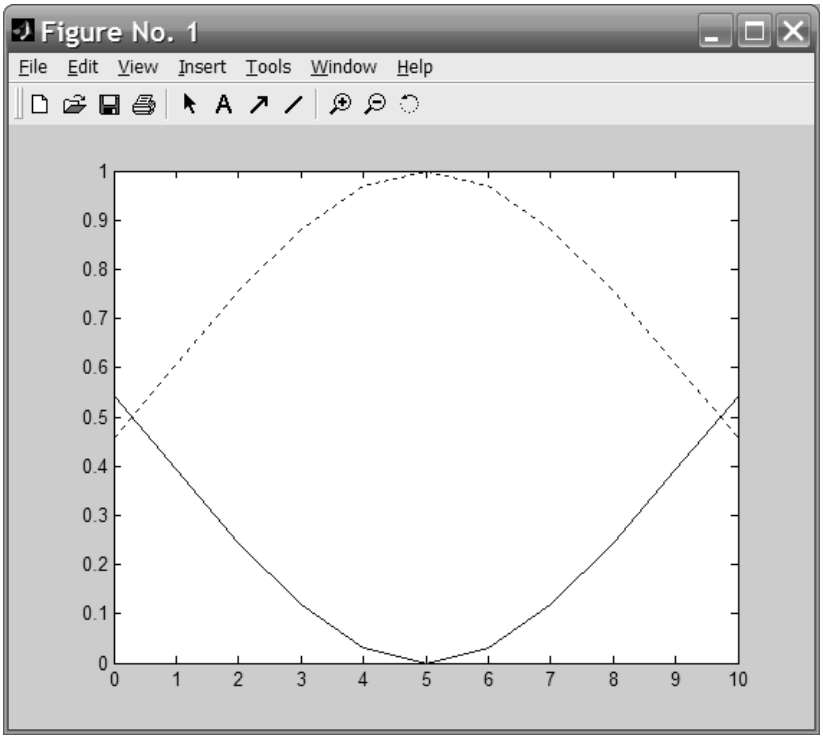


Рис. 1.8. Доповнення нечіткої множини

Опис функції доповнення: $y = 1 - y^*$, де y^* – вихідна ФП.

Приклад 1.9. Програма використання операції доповнення.

```
» x= [0 : 10];
» y1 = gaussmf (x, [3 5]);
» y= 1 - y1;
» plot (x, y1, ':');
» hold on;
» plot(x, y);
» hold off
```

1.2. Індивідуальні завдання

1. Побудувати трикутну та трапецієподібну функції приналежності.

2. Побудувати просту та двосторонню функцію приналежності Гауса, утворену за допомогою різних функцій розподілу.

3. Побудувати функцію належності "узагальнений дзвін", яка дозволяє представляти нечіткі суб'єктивні переваги.

4. Побудувати набір сигмоїдних функцій:

- основну односторонню, яка відкрита зліва або справа;
- додаткову двосторонню;
- додаткову несиметричну.

5. Побудувати набір поліноміальних функцій належності (*Z*-, *PI*- і *S*-функцій).

6. Побудувати мінімаксну інтерпретацію логічних операторів з використанням операцій пошуку мінімуму та максимуму.

7. Побудувати ймовірнісну інтерпретацію кон'юнктивних та диз'юнктивних операторів.

8. Побудувати доповнення нечіткої множини, що описує деяке розмите судження і є математичним описом вербального виразу, що заперечує цю нечітку множину.

За виконання пунктів 1 – 8 індивідуального завдання, значення змінних *a*, *b*, *c*, *d* і т.д. необхідно вибирати довільним чином.

9. Оформіть звіт з лабораторної роботи.

Лабораторна робота 2

ДОСЛІДЖЕННЯ АЛГОРИТМУ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ

Мета лабораторної роботи: отримання та закріплення знань про алгоритм нечіткої кластеризації, формування практичних навичок розв'язання задач кластеризації методами нечіткої логіки.

2.1. Короткі відомості з теорії

2.1.1. FCM-алгоритм кластеризації

Алгоритм нечіткої кластеризації називають *FCM*-алгоритмом (*Fuzzy Classifier Means*, *Fuzzy C-Means*). Метою *FCM*-алгоритма кластеризації є автоматична класифікація множин об'єктів, які задаються векторами ознак у просторі ознак. Інакше кажучи, такий алгоритм визначає кластери і класифікує об'єкти. Кластери є нечіткими множинами, і, крім того, межі між кластерами також є нечіткими.

FCM-алгоритм кластеризація передбачає, що об'єкти належать всім кластерам з певною ФП. Ступінь належності визначається відстанню від об'єкта до відповідних кластерних центрів. Даний алгоритм ітераційно (тобто через послідовні повторення чи цикли для досягнення певної мети) обчислює центри кластерів та нові ступені належності об'єктів.

Для заданого числа K вхідних векторів x_k ($k = \overline{1, K}$) і N кластерів, що виділяються c_j ($j = \overline{1, N}$) передбачається, що будь-який x_k належить любому c_j ($j = \overline{1, N}$) з належністю $\mu_{jk} \in [0,1]$, де j – номер кластера, а k – номер вхідного вектору.

Приймаються до уваги такі умови нормування для μ_{jk}:

$$\sum_{j=1}^{N} \mu_{jk} = 1, \ \forall k = 1,...,K;$$

$$0 < \sum_{k=1}^{N} \mu_{jk} \le K, \ \forall j = 1,...,N.$$

Мета алгоритму – мінімізація суми усіх зважених відстаней $\left\|x_k - c_j\right\|$

$$\sum_{j=1}^{N}\sum_{k=1}^{K}(\mu_{jk})^q\left\|x_k - c_j\right\| \to \min,$$

де q – фіксований параметр, що задається перед ітераціями.

Для досягнення вищезгаданої мети необхідно вирішити наступну систему рівнянь:

$$\partial \Big/ \partial \mu_{jk}\left(\sum_{j=1}^{N}\sum_{k=1}^{K}(\mu_{jk})^q\left\|x_k - c_j\right\|\right) = 0, \quad j=\overline{1,N},\ k=\overline{1,K},$$
$$\partial \Big/ \partial c_j\left(\sum_{j=1}^{N}\sum_{k=1}^{K}(\mu_{jk})^q\left\|x_k - c_j\right\|\right) = 0, \quad j=\overline{1,N}.$$

Спільно з умовами нормування μ_{jk} дана система рівнянь має таке рішення:

$$c_j = \frac{\sum_{j=1}^{N}(\mu_{jk})^q \cdot x_k}{\sum_{k=1}^{K}(\mu_{jk})^q},\ j=\overline{1,N}.$$

(зважений центр гравітації) та

$$\mu_{jk} = \frac{1 \Big/ \left\|x_k - c_j\right\|^{1/(q-1)}}{\sum_{j=1}^{N}\left(1 \Big/ \left\|x_k - c_j\right\|^{1/(q-1)}\right)},\ j=\overline{1,N},\ k=\overline{1,K}.$$

Алгоритм нечіткої кластеризації виконується кроками.

Крок 1. Ініціалізація.

Вибираються такі параметри:

– необхідна кількість кластерів N, $2 < N < K$;

– тип відстаней (наприклад, відстань по Евкліду);

– фіксований параметр q (зазвичай 1,5);

– начальна (на нульовій ітерації) матриця функцій належності $U^{(0)} = (\mu_{jk})^{(0)}$ об'єктів x_k $(k = \overline{1, K})$ з урахуванням заданих початкових центрів кластерів c_j ($j = \overline{1, N}$).

Крок 2. Регулювання позицій $c_j^{(t)}$ центрів кластерів.

На t-м ітераційному кроці при відомій матриці $\mu_{jk}^{(t)}$ обчислюється $c_j^{(t)}$ відповідно до вищенаведеного рішення системи рівнянь.

Крок 3. Коригування значень належності $\mu_{jk.}$

Враховуючи відомі $c_j^{(t)}$, обчислюються $\mu_{jk}^{(t)}$, якщо $x_k \neq c_j$, в іншому випадку:

$$\mu_{jk}^{(t+1)} = \begin{cases} 1, & \text{якщо } k = j, \\ 0, & \text{якщо } k \neq j. \end{cases}$$

Крок 4. Зупинення алгоритму.

Алгоритм нечіткої кластеризації зупиняється при виконанні наступної умови:

$$\left\| U^{(t+1)} - U^{(t)} \right\| \leq \varepsilon,$$

де $\| \ \|$ – матрична норма (наприклад, Евклідова норма); ε – рівень точності, що задається заздалегідь.

2.1.2. Розв'язання задач кластеризації

Існують два способи вирішення задач кластеризації у MATLAB: з використанням командного рядка або графічного інтерфейсу користувача. Розглянемо перший із зазначених способів.

Для знаходження центрів кластерів у MATLAB є вбудована функція *fcm*, опис якої наведено нижче.

Опис функції: [*center*,*U*,*obj_fcn*] = *fcm*(*data*,*cluster_n*).

Аргументами цієї функції є:

1) *data* – множина даних, що підлягають кластеризації, кожен рядок описує крапку в багатовимірному просторі характеристик;

2) *cluster_n* – кількість кластерів (більше одного).

Функцією повертаються такі параметри:

1) *center* – матриця центрів кластерів, кожен рядок якої містить координати центру окремого кластера;

2) *U* – результуюча матриця функції належності;

3) *obj_fcn* – значення цільової функції на кожній ітерації.

Приклад 2.1. Програма нечіткої кластеризації.

```
% завантаження даних, що підлягають кластеризації,
% із файлу
» load fcmdata.dat;
%визначення центру кластеризації (два кластери)
» [center, U, obj_fcm] = fcm(fcmdata, 2);
%визначення максимального ступеня належності
% окремого елемента даних кластеру
»maxU = max(U);
% розподіл рядків матриці даних між відповідними
% кластерами
»index1 = find (U(1, :) == maxU);
»index2 = find(U(2, :) = maxU);
```

```
%побудова даних, що відповідають першому кластеру
»plot (fcmdata (index1, 1), fcmdata (index1, 2),' ko',
'markersize', 5, 'LineWidth' ,1);
»hold on
%побудова даних, що відповідають другому кластеру
»plot(fcmdata (index2, 1), fcmdata(index2, 2), 'kx',
'markersize', 5, 'LineWidth', 1);
% побудова кластерних центрів
»plot(center(1, 1), center(1, 2), 'ko', 'markersize',
15, 'LineWidth', 2)
»plot (center (2, 1), center (2, 2), 'kx',
'markersize', 15, 'LineWidth', 2)
```

На рис. 2.1 представлено множину даних, що підлягають кластеризації, та знайдені центри кластерів для прикладу 2.1.

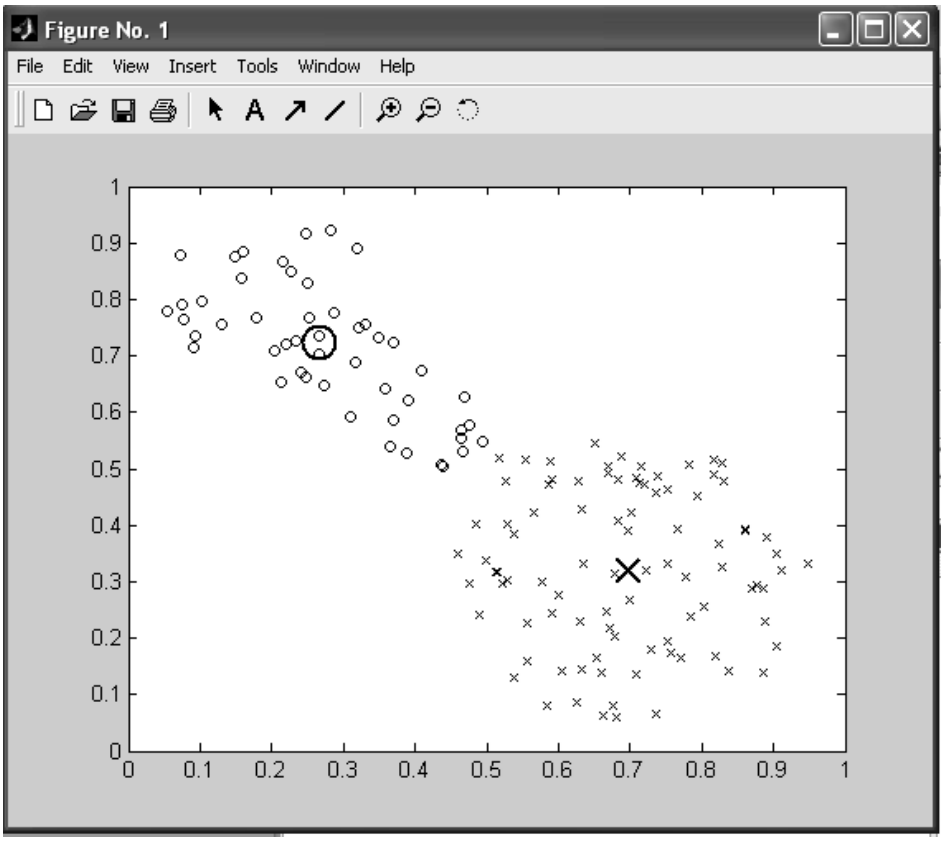


Рис. 2.1. Множиа аналізованих даних та центри кластерів

Функція *fcm* виконується ітераційно до тих пір, поки зміни цільової функції перевищують певний заданий поріг.

На кожному кроці у командному вікні MATLAB виводяться порядковий номер ітерації та відповідне поточне значення цільової функції (табл. 2.1).

Таблиця 2.1 – Зміна цільової функції

Номер ітерації	Значення цільової функції	Номер ітерації	Значення цільової функції
1	8,94	7	3,81
2	7,31	8	3,80
3	6,90	9	3,79
4	5,41	10	3,79
5	4,08	11	3,79
6	3,83	12	3,78

Для оцінки динаміки зміни значень цільової функції використовують команду побудови графіка *plot(obj_fcm)*. Результати прикладу 2.1 показані на рис. 2.2.

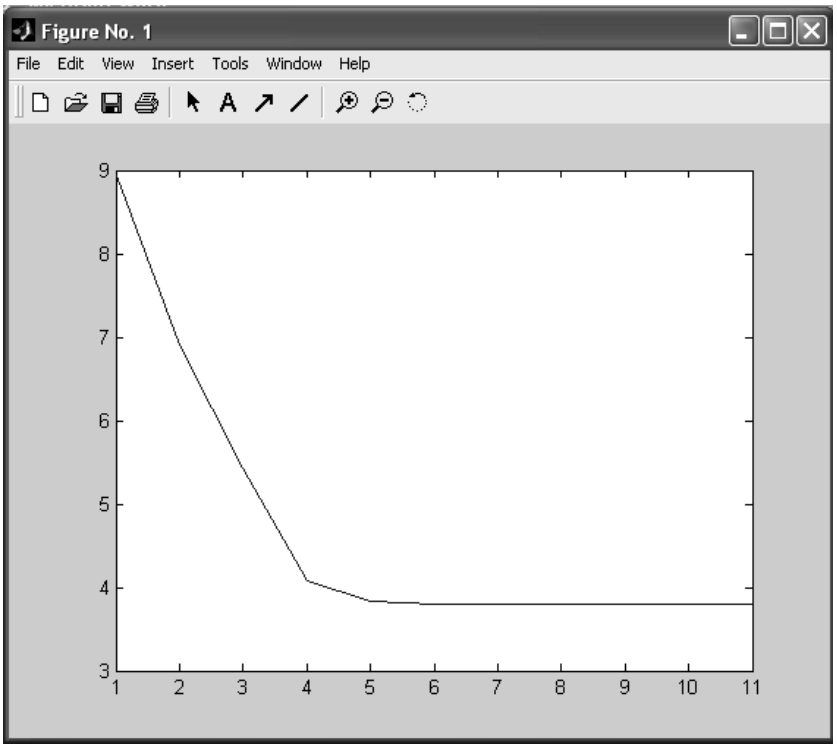


Рис. 2.2. Графік зміни значень цільової функції

Функцію кластеризації можна викликати з додатковим набором параметрів: *fcm* (*data*, *cluster_n*, *options*). Додаткові аргументи використовуються для управління процесом кластеризації:

– *options*(1) – показник ступеня для матриці *U* (за замовчуванням: 2.0);

– *options*(2) – максимальна кількість ітерацій (за замовчуванням: 100);

– *options*(3) – мінімально допустима зміна значень цільової функції (за замовчуванням: 1*e*–5);

– *options*(4) – відображення інформації на кожному кроці (за замовчуванням: 1).

Приклад визначення функції *fcm* із додатковими параметрами: [*center*, *U*, *obj_fcn*] = *fcm*(*fcmdata*, 2, [2, 100, l*e*–5, 1]).

Другий спосіб розв'язання задач кластеризації в MATLAB викликається командою *findcluster*. Головне вікно інструменту кластеризації показано на рис. 2.3.

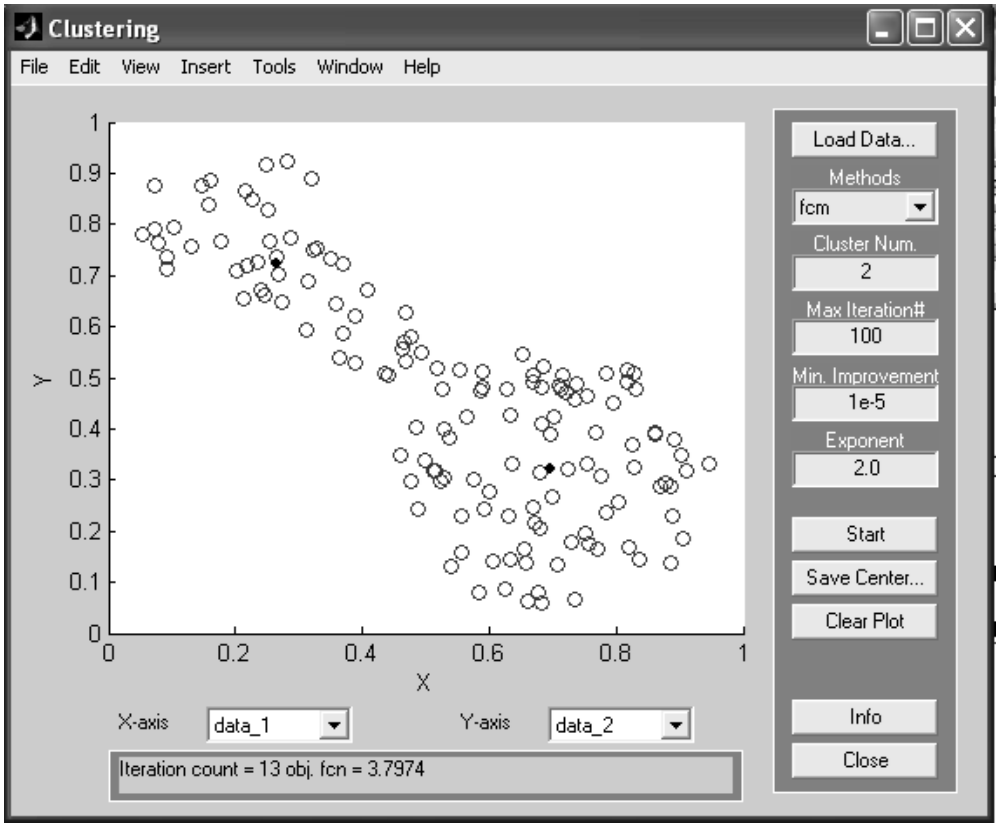


Рис. 2.3. Головне вікно кластеризації в MATLAB

Кнопка <*Load Data*> використовується для завантаження вихідних даних, що підлягають кластеризації, наступного формату: кожен рядок являє собою точку в багатовимірному просторі характеристик, кількість рядків відповідає кількості точок (елементів даних). Графічну інтерпретацію вихідних даних можна спостерігати у однойменному вікні головного вікна інструмента.

Вибір типу алгоритму кластеризації здійснюється з використанням спадного меню *Methods* (пункт меню – *fcm*). Далі визначаються параметри алгоритму кластеризації:

– кількість кластерів (рядок введення – *Cluster Num*);

– максимальна кількість ітерацій (рядок введення – *Max Iteration*);

– мінімальне значення поліпшення цільової функції (рядок введення – *Min*. *Improvement*);

– показник ступеня при матриці ФП (рядок введення – *Exponent*).

Після визначення необхідних значень зазначених параметрів здійснюється запуск алгоритму кластеризації за допомогою кнопки <*Start*>. Кількість вироблених ітерацій та значення цільової функції можна переглянути в нижній частині головного вікна інструменту кластеризації.

Координати знайдених центрів кластерів можна зберегти, натиснувши на кнопку <*Save Center...*>. Кожен рядок матриці у файлі є набором координат окремого кластера. Кількість рядків відповідає кількості кластерів.

2.2. Індивідуальні завдання

1. Необхідно сформулювати завдання в галузі комп’ютерної інженерії, для якого була б необхідна автоматична класифікація безлічі об'єктів, які задаються векторами ознак у просторі ознак.

2. Вирішити в MATLAB сформульовану задачу з використанням механізму кластеризації методами нечіткої логіки, використовуючи

командний рядок і графічний інтерфейс користувача. При використанні командного рядка функцію кластеризації необхідно викликати з додатковим набором параметрів.

3. Знайти центри кластерів та побудувати графік зміни значень цільової функції.

4. Оформіть звіт з лабораторної роботи.

Лабораторна робота 3

МОДЕЛЮВАННЯ НЕЧІТКОЇ СИСТЕМИ ЗАСОБАМИ ІНСТРУМЕНТАРІЮ НЕЧІТКОЇ ЛОГІКИ

Мета лабораторної роботи: отримання та закріплення знань, формування практичних навичок побудови нечіткої системи засобами інструментарію нечіткої логіки.

3.1. Короткі відомості з теорії

У складі MATLAB є п'ять основних засобів графічного інтерфейсу користувача (ГІК), які забезпечують доступ до інструментарію нечіткої логіки (ІНЛ): редактори системи нечіткого виведення (СНВ), функції приналежності, правила виведення, а також засоби перегляду правил та поверхні виведення. Ці засоби пов'язані між собою динамічно, і зміни, що відбуваються в одному з них спричиняють зміни в інших.

Редактор СНВ надає можливість формування проектованої системи на високому рівні абстракції: кількість вхідних та вихідних змінних, найменування змінних.

Редактор функцій приналежності (ФП) використовується для визначення форми ФП, асоційованих з кожною змінною.

Редактор правил виводу застосовується для редагування списку правил, які визначають поведінку системи, що проектується.

Засіб перегляду правил виведення використовується з метою діагностики і може показувати, наприклад, активність правил або форму впливу окремих ФП на результат нечіткого виводу.

Засіб перегляду поверхні виводу використовується для відображення залежності виходу системи від одного або двох входів. Іншими словами, він генерує та виводить карту поверхні виведення розробленої СНВ.

Редактор СНВ. Побудова нечітких систем по Мамдані. Для побудови системи в командному рядку основного вікна MATLAB необхідно набрати команду *fuzzy*. Вікно редактора нової СНВ містить вхідну, позначену *input1* та вихідну – *output1* змінні. За замовчуванням ІНЛ пропонує створювати СНВ типу Мамдані.

Щоб додати нову змінну, необхідно вибрати в меню *Edit* відповідний пункт (для вхідної змінної – *Add input*, для вихідної – *Add output*). Зміна назви змінної відбувається за кроками.

Крок 1. Відзначається змінна, яку потрібно перейменувати.

Крок 2. У полі редагування змінюється найменування змінної за промовчанням на ім'я, запропоноване користувачем.

Збереження проектованої системи у робочий простір середовища MATLAB (у змінну) проводиться за допомогою пункту меню *File – Save to workspace as...* . В цьому випадку дані зберігаються до закінчення сеансу роботи з MATLAB. Для збереження даних на диску після закінчення сеансу роботи застосовується відповідний пункт того самого меню – *Save to disk as...* .

Редактор ФП. Наступним кроком у побудові нечіткої моделі засобами ІНЛ є асоціювання набору ФП з кожною вхідною та вихідною змінною. Ця операція проводиться у редакторі ФП трьома способами, активізувати який можна:

– вибором у меню *View* пункту *Edit Membership Functions...*;

– подвійним клацанням миші на зображенні відповідної змінної (вхідний або вихідний));

– набором у командному рядку оператора *mfedit*.

За допомогою редактора ФП можна відображати та редагувати будь-які ФП, асоційовані (пов'язані) з усіма вхідними та вихідними змінними СНВ.

Зв'язування ФП з ім'ям змінної відбувається так:

– вибирається змінна по імені з набору графічних об'єктів вікна редактора ФП;

– вказується діапазон зміни значень для базової та видимий діапазон для поточної змінних;

– в меню *Edit* вибирається пункт *Add MFs...* . У вікні, що з'явилось вибирають вид ФП та їх кількість.

Редагують ФП поточної змінної двома способами: використовуючи графічне вікно ФП або змінюючи характеристики ФП (найменування, тип та числові параметри). При виборі необхідної ФП у графічному вікні допускається плавна зміна кривої за допомогою миші.

Таким чином, при побудові СНВ необхідно за допомогою редактора ФП визначити відповідні функції для кожної з вхідних та вихідних змінних.

Редактор правил виведення. Після того як зазначено кількість вхідних та вихідних змінних, визначено їх найменування та побудовано відповідні ФП, в СНВ необхідно увімкнути правила виведення. Для цього в меню *View* вибирається пункт *Edit Rules...* або у командному рядку MATLAB набирається команда *ruleedit*.

Ґрунтуючись на описах вхідних та вихідних змінних, визначених у редакторі ФП, редактор правил виведення формує структуру правила автоматично. Від користувача потрібно лише зв'язати значення вхідних та вихідних змінних, вибираючи зі списку заданих раніше ФП та визначити логічні зв'язки між ними. Також допускається використання логічного заперечення (НЕ) та зміна ваг правил у діапазоні від 0 до 1.

Правила виводу можуть відображатися у вікні у різних форматах, які визначаються шляхом вибору відповідного пункту підменю *Format* меню *Options*. За замовчуванням використовується розширений формат відображення правил виводу (*verbose form*):

If (*input_1 is*[*not*] *mf_1j1*) <*and, or*>...(*input_i is*[*not*] *mf_iji*)...<*and,or*>

(*input_n is*[*not*] *mf__njn*) *then*

(*output_1 is*[*not*] *mf_n + 1jn+1*) <*and, or*>...

(*output_k is*[*not*] *mf_k + njk+n*) <*and, or*>...

(*output_m is*[*not*] *mf_m + njm+n*) (*w*),

де *i* – номер вхідної змінної; *ji* – номер ФП *i*-й змінної; *k* – номер вихідної змінної; *n* – кількість вхідних змінних; *m* – кількість вихідних змінних; *w* – вага правила.

Круглі дужки містять обов'язкові параметри, квадратні – необов'язкові, а кутові – альтернативні параметри (один на вибір).

Крім формату за замовчуванням, існують ще два види форматів відображення правил: символьний (*symbolic form*) та індексний (*indexed form*). Символьний формат має такий вигляд:

(*input_1*<~=,==>*mf_1j1*)<&, | >...

(*input_i*<~=,==>*mf_iji*)…<&, | >

(*input_n*<~=,==>*mf_njn*)=>

(*output_1*<~=,==>*mf_n + 1jn+1*)...<&, | >

(*output_k*<~=,==>*mf_k + njk+n*) <&, | >...

(*output_m*<~=,==>*mf_m + njm≠n*) (*W*)

Відмінність символьного формату від розширеного полягає в тому, що замість словесної інтерпретації зв'язок використовується символьна (символи «&» і «|» – відповідно визначають логічне І та логічне АБО, символ «~» – логічне заперечення, а символ « => » є роздільником умовної та заключної частин правила (антецедента та консеквенту).

Загальний опис правила виведення в індексному форматі може бути поданий у такому вигляді:

[–]1*j*1…[–]*iji*…[–]*njn*[–]*n* + 1*jn* + 1…[–]*k* + *njk* +1…[–]*m* + *njm* + *n*(*w*):<1,2>.

Тут порядок слідування чисел відповідає черговості змінних, причому символ « , » поділяє правило на умовну і заключну частини. До двокрапки записується порядковий номер відповідної ФП, після двокрапки – вид логічного зв'язування (< 1 > – логічне І, < 2 > – логічне АБО). Логічне заперечення задається символом < – >.

Після визначення правил виведення в однойменному редакторі можна стверджувати, що СНВ повністю створено.

Приклад 3.1. Створення СНВ.

Розглянемо таку ситуацію. Необхідно оцінити ступінь інвестиційної привабливості конкретного бізнес-проекту на підставі даних про ставку дисконтування та період рентабельності.

Крок 1. Викликаємо редактор для створення СНВ, набираючи у командному рядку *fuzzy*. Додаємо вхідну змінну за допомогою вибору меню *Edit* пункту *Add input*. В результаті отримуємо наступну структуру СНВ: два входи, механізм нечіткого виведення по Мамдані, один вихід. Оголошуємо першу змінну як *discont*, а другу – *period*, які відповідно представлятимуть ставку дисконтування та період рентабельності бізнес-проекту. Найменування вихідної змінної, на підставі якої приймається рішення про рівень інвестиційної привабливості бізнес-проекту, задається як *rate*. Збережемо створювану модель під назвою *Invest*. На рис. 3.1 представлено поточний стан вікна редактора СНВ.

Крок 2. Кожний вхідний та вихідний змінної поставимо у відповідність набір ФП. Ця процедура реалізується у редакторі ФП. Для *discont* визначаємо діапазон базової змінної (*Range*) від 5 до 50 (одиниця виміру – відсотки). Такий діапазон вибираємо для її відображення (*Display Range*). Додамо три ФП, тип яких – *trimf*. Послідовно виділяючи мишею окремі ФП, надамо найменування – *small, middle, big* відповідно невеликій, середній та великій ставці дисконтування. Вікно редактора ФП показано на рис. 3.2. Змінної *period* діапазон базової змінної визначено рівним [3, 36]

(одиниця виміру – місяці), поставлені у відповідність три ФП типу *gaussmfc* найменуваннями – *short*, *normal*, *long*. Таким чином, змінна терміну рентабельності бізнес-проекту прийматиме наступні значення: короткий, звичайний та тривалий термін рентабельності. Нарешті, для змінної *rate* визначаємо: базова змінна змінює значення в діапазоні [0, 1], семантика описується трьома ФП типу *trimf*' з найменуваннями: *bad*, *normal*, *good*.

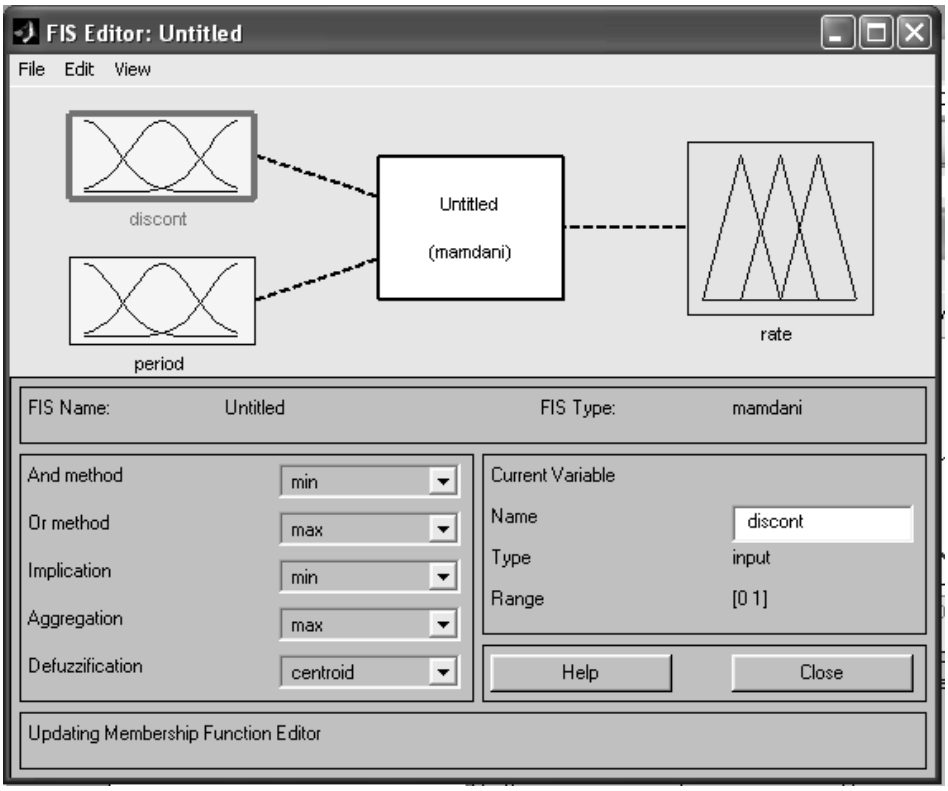


Рис. 3.1. Вікно редактора системи нечіткого виводу

Крок 3. Заключним етапом побудови СНВ є визначення набору правил, які задають зв'язок вхідних змінних із вихідними. Для цього в редакторі правил виводу визначимо:

ЯКЩО *discont* = *small* І *period* = *short* ТО *rate* = *good*

ЯКЩО *discont* = НЕ *small* І *period* = *long* ТО *rate* = *bad*

ЯКЩО *discont* = *middle* І *period* = *normal* ТО *rate* = *normal*

ЯКЩО *discont* = *big* І *period* = *short* ТО *rate* = *normal*

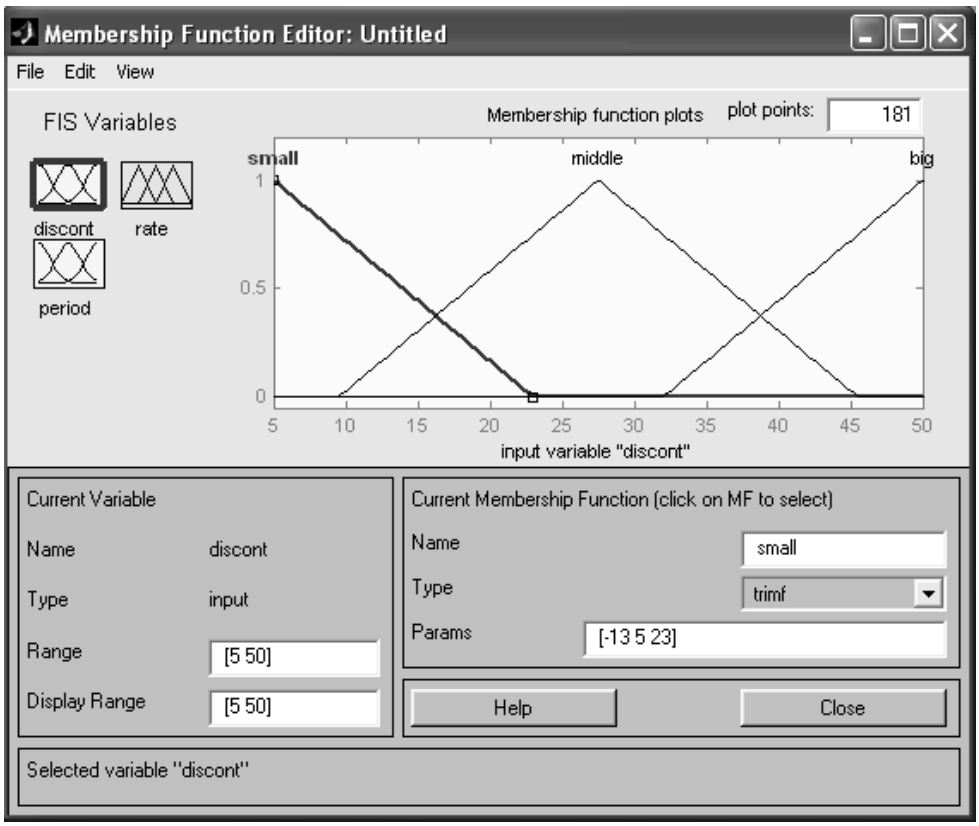


Рис. 3.2. Вікно редактора ФП

Поточний стан вікна редактора правил виведення показано на рис. 3.3. У розширеному форматі відображення зазначені правила виведення надаються наступним чином:

if(*discont is small*) *and* (*period is short*) *then* (*rate is goad*) (1)

if(*discont is not small*) *and* (*period is long*) *then* (*rate is bad*) (1)

if(*discont is middle*) *and* (*period is normal*) *then* (*rate is normal*) (1)

if(*discont is big*) *and* (*period is short*) *then* (*rate is normal*) (1)

При зміні формату на символьні правила виводу матимуть вигляд:

(*discont == small*) & (*period == short*) => (*rate == good*) (1)

(*discont ~= small*) & (*period == long*) => (*rate == bad*) (1)

(*discont == middle*) & (*period == normal*) => (*rate== normal*) (1)

(*discont == big*) & (*period == short*) => (*rate == normal*) (1)

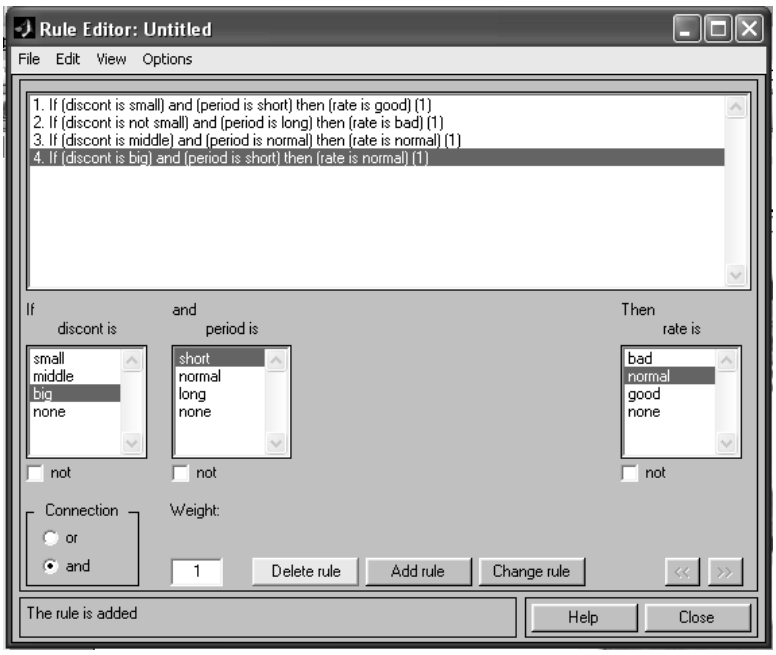


Рис. 3.3. Вікно редактора правил виводу

Засіб перегляду правил виведення. Даний засіб перегляду правил виводу дозволяє відобразити процес нечіткого виводу та отримати результат. Головне вікно засобу перегляду складається з кількох графічних вікон, що розташовуються по рядках та стовпцях. Кількість рядків відповідає числу правил нечіткого виводу, а кількість стовпців – числу вхідних і вихідних змінних, заданих у СНВ, що розробляється. Додаткове графічне вікно служить для відображення результату нечіткого виведення та операції дефазифікації. У кожному вікні відображається відповідна ФП, рівень її зрізу (для вхідних змінних) та вклад окремої ФП у загальний результат (для вихідних змінних).

У нижній частині головного вікна можна відобразити номери правил виводу в різних форматах виводу, позначаючи їх мишею. Для зміни формату меню *Options* вибирається пункт *Rule display format*.

Зміна значень вхідних змінних допустима двома способами:

1) шляхом введення в поле *Input* запису вхідного вектору, розмірність якого дорівнює кількості вхідних змінних;

2) клацанням миші у будь-якому графічному вікні, яке відноситься до вхідної змінної.

Вхідний вектор у кожному з цих варіантів визначення вихідних даних буде задавати набір червоних вертикальних прямих.

Для СНВ, розглянутої у прикладі 3.1, при вхідному векторі [15 10] (ставка дисконтування – 15 %, період рентабельності бізнес-проекту – 10 місяців) результат (ступінь інвестиційної привабливості) становитиме 0,639 (рис. 3.4).

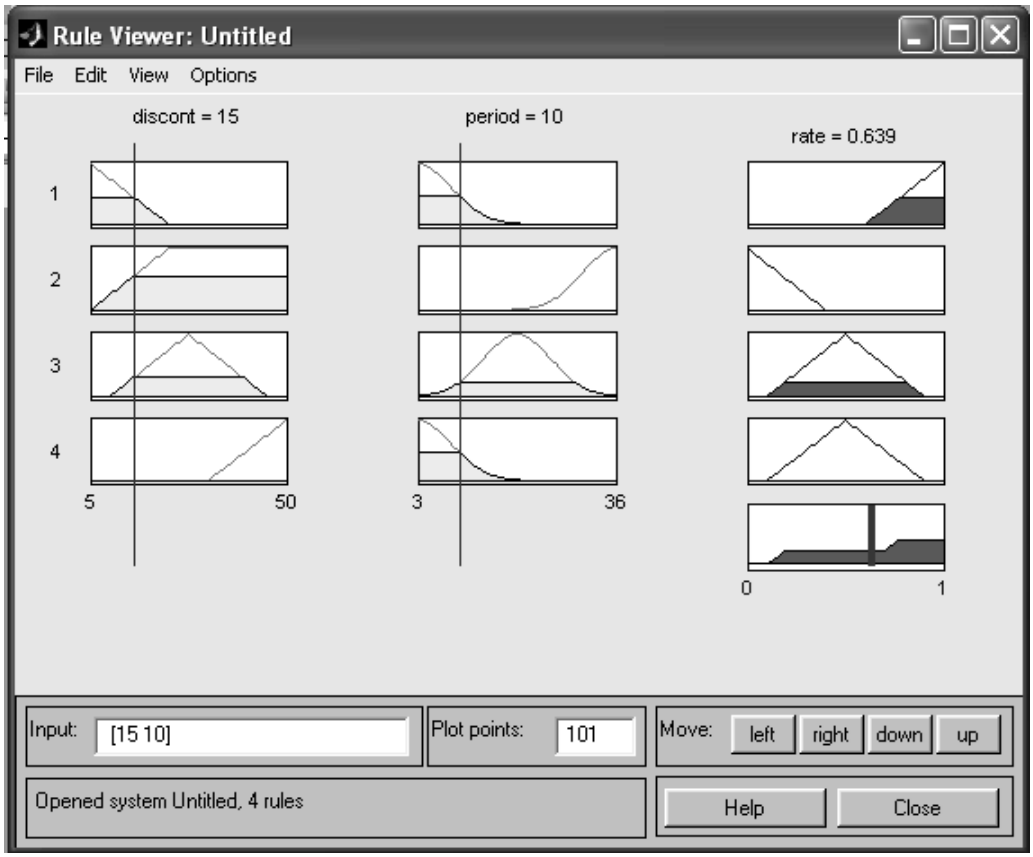


Рис. 3.4. Вікно засобу перегляду правил виводу

Засіб перегляду поверхні виводу. Засіб перегляду поверхні виводу дозволяє будувати тривимірну поверхню як залежність однієї із вихідних змінних від двох вхідних. Вибір вхідних і вихідних змінних здійснюється за допомогою випадаючого меню головного вікна програмного засобу, що розглядається. Кількість ліній, що виводяться по осях *X* і *Y* визначається в полях введення *X grids*, *Y grids*. Поверхня виведення, що відповідає правилам виведення прикладу 3.1, показана на рис. 3.5.

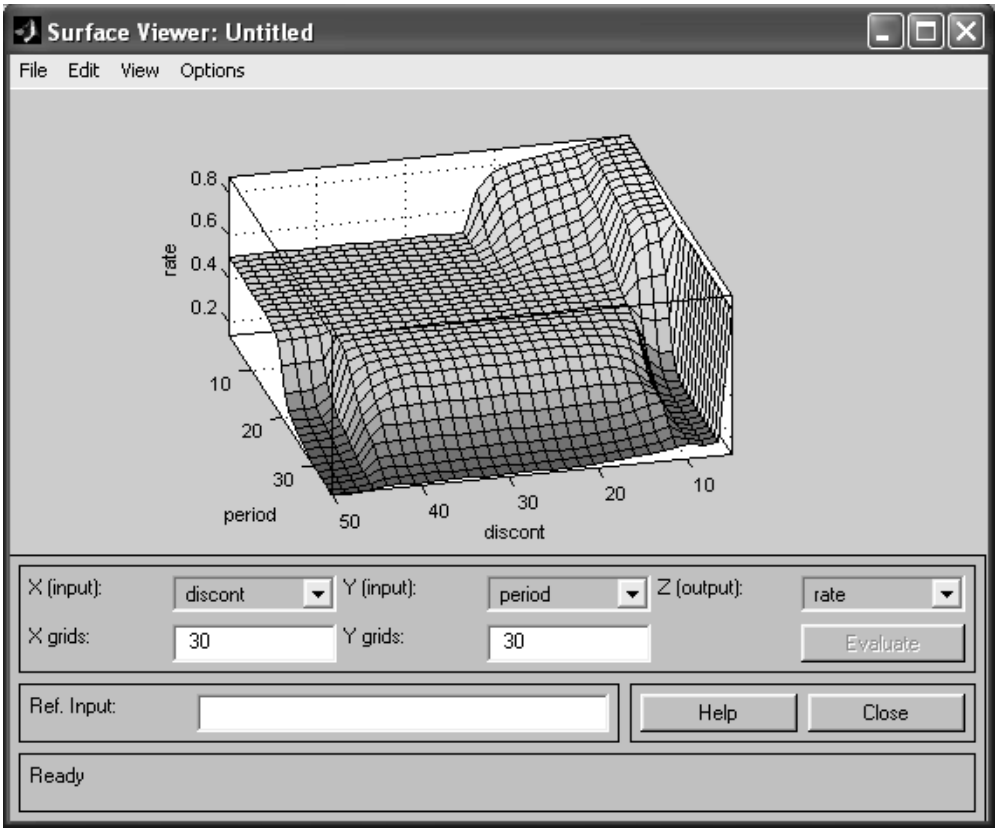


Рис. 3.5. Вікно перегляду поверхні рішень

Побудова нечітких систем типу Сугено. Розглянемо побудову СНВ двома редакторами – СНВ та ФП. Для побудови СНВ типу Сугено необхідно у меню *File* вибрати пункт *New FIS Sugeno*. Кількість вхідних та вихідних змінних визначається так само, як і при побудові СНВ типу Мамдані.

Редактор ФП. Для СНВ типу Сугено зміни стосуються лише схеми визначення ФП для вихідних змінних. ІНЛ у середовищі MATLAB дозволяє розробляти два види нечітких моделей. Перша модель – це нечітка модель Сугено нульового порядку. Нечітке правило виведення має такий вигляд:

$$if\ x\ is\ A\ and\ y\ is\ B\ then\ z = k,$$

де A і B – нечіткі множини антецедента; k – чітко задана константа консеквенту.

Для побудови такої моделі при додаванні ФП необхідно вибрати тип константи (*constant*) і задати як параметр ФП чисельне значення відповідної

константи. Друга модель – нечітка модель Сугено першого порядку. Для неї нечітке правило виведення записується в такий спосіб:

$$if\ x\ as\ A\ and\ y\ is\ B\ then\ z = p \cdot x + q \cdot y + r,$$

де p, q і r – константи.

У разі тип ФП – лінійна залежність (*linear*). Для визначення параметрів ФП необхідно ввести вектор, елементи якого відповідають чисельним значення констант консеквента.

Робота з редактором правил виведення, а також із засобами перегляду правил та поверхні виведення виконується аналогічно випадку побудови СНВ по Мамдані.

Приклад нечіткого виведення Сугено з використанням нечіткої моделі нульового порядку і правил виведення, визначених вище, представлений на рис. 3.6 (вихідна змінна має три значення: *bad*, *normal*, *good*, які задаються відповідно трьома константами – 0, 0.5, 1).

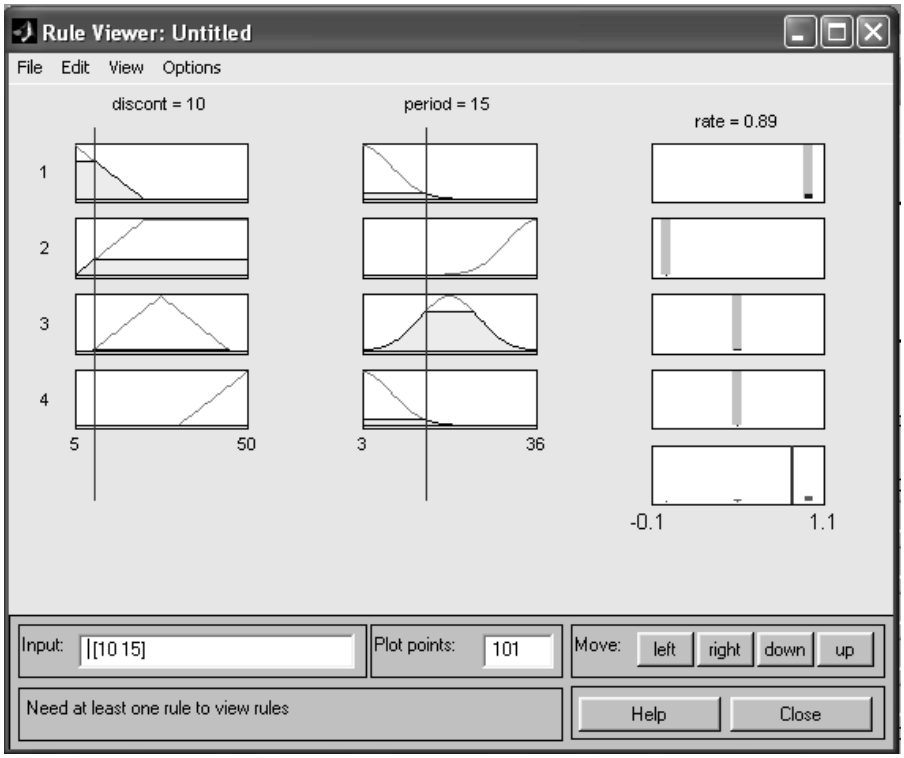


Рис. 3.6. Вікно перегляду правил виведення (висновок по Сугено)

3.2. Індивідуальні завдання

1. Необхідно сформулювати абстрактну ситуацію в галузі комп’ютерної інженерії та побудувати для неї нечітку систему з використанням графічного інтерфейсу користувача, який забезпечує доступ до інструментарію нечіткої логіки та редактора системи нечіткого виведення. При цьому побудова нечіткої системи, в першому випадку, має ґрунтуватися на принципі Мамдані, а в другому, ґрунтуватися на принципі Сугено.

2. При виконанні пункту 1 індивідуального завдання задатися різними діапазонами зміни вхідних та вихідних змінних нечіткої системи, а також різними типами ФП.

3. Побудувати графічне відображення правил виведення та поверхні рішень, сформульованої абстрактної ситуації.

4. Виконати порівняльний аналіз двох нечітких систем, побудованих на принципі Мамдані та Сугено.

5. Оформіть звіт з лабораторної роботи.

Лабораторна робота 4

НЕЧІТКЕ УПРАВЛІННЯ ДИНАМІЧНИМИ ПРОЦЕСАМИ

Мета лабораторної роботи: отримання та закріплення знань про методи проектування нечітких контролерів, формування практичних навичок роботи при управлінні складними об'єктами чи динамічними процесами.

4.1. Короткі відомості з теорії

4.1.1. Введення в теорію нечіткого управління

Методи нечіткого виведення, крім завдань побудови експертних систем, широко використовуються в розробці нечітких контролерів. Основне призначення контролера – це управління об'єктом, у якому поведінка керованого об'єкта описується нечіткими правилами.

Нехай Y – основний параметр об'єкта;

Y^* – бажане значення параметра;

$E = [Y^* - Y]$ – помилка поточного стану;

U – керуючий вплив, що виробляється контролером;

$U(K)$ – значення керуючого впливу;

$$U(K) = F(U(K-1), ..., U(K-\tau), E(K-1), ..., E(K-\tau));$$

τ – глибина розгляду передісторії.

Зазвичай нечіткий контролер працює із глибиною $\tau = 1$:

$$U(K) = F(U(K-1), E(K-1)); \text{ або } U(K) = F(U(K-1), \Delta E(K-1));$$

де ΔE – зміна помилки, а правило формування керуючого впливу

визначається за допомогою бази нечітких правил. Зазвичай керуючий вплив враховує попередній вплив та останню зміну помилки.

Процес управління описується набором змінних, які стосуються класу лінгвістичних змінних. Сутності керованого процесу можна класифікувати як "вхідні змінні" та "вихідні змінні" процесу. У загальному випадку вхідні змінні визначають залежні від часу потоки, тоді як вихідні описують моментальний стан процесу. Вхідні змінні, якими можуть керувати люди або комп'ютери, називають керованими змінними процесу. Управління у широкому сенсі означає маніпулювання процесом у контурі зворотнього зв'язку із єдиною метою отримання найкращого результату, тобто, отримання оптимального значення певних вихідних змінних.

Для керування необхідно:

1) визначити поточні значення вихідних змінних;

2) порівняти поточні вихідні значення із заданими цільовими значеннями;

3) підібрати значення керованих змінних так, щоб досягти цільових значень;

4) повторювати пункт 1 – 3 доти, доки не будуть отримані цільові значення.

Розглянемо нечіткий контролер, в якому керований процес має три вхідні змінні з двома керованими змінними (вхід 2 і вхід 3) та однією випадковою змінною (вхід 1), а також три вихідні змінні. Нечіткий контролер згідно з певним алгоритмом перетворює три вихідні змінні на дві керуючі змінні, які потім повертаються на об'єкт управління. Порівняння вихідних значень із цільовими значеннями виконується самим контролером.

На відміну від більшості стандартних контролерів, які використовують математичні обчислення, нечіткий контролер застосовує так звані правила, подібно до людини-оператора.

Приклад 4.1. Правила керування:

– ЯКЩО тиск *низький*, ТО відкрити кран (процес хімічного виробництва).

– ЯКЩО відсоток за кредитом *високий*, ТО скоротити запозичення (процес фінансового планування).

– ЯКЩО рівень споживання енергії *низький*, ТО скоротити вихідну потужність генератора (процес управління потужністю).

Терміни, набрані прямим шрифтом, – це імена об'єктних змінних, слова, набрані курсивом, – нечіткі значення відповідних змінних. Подібні правила використовуються людиною у процесі управління. Досвідчений оператор розуміє значення нечітких термінів: "тиск низький" або "трохи відкрити кран", у той час як пристрій автоматичного контролю працює з чіткими значеннями: вихідні змінні багатьох технічних процесів обробляються аналого-цифровими перетворювачами, керуючі змінні – цифро-аналоговими перетворювачами; обидва види пристроїв зводяться до обробки цифрових значень.

Отже, щоб реалізувати нечіткий контролер, необхідно:

– перетворити чіткі значення вихідних змінних процесу на нечіткі;

– використовувати правила перетворення нечітких вихідних змінних на нечіткі значення керуючих;

– перетворити нечіткі значення керуючих змінних на чіткі.

Для кожної змінної процесу A визначимо функцію приналежності $\mu(a, f)$, яка пов'язує будь-яке чітке значення змінної A з нечітким.

4.1.2. Правила та імплікація

Правила

Правила у системі нечіткого контролера мають вигляд ЯКЩО f_0, ТО g_0, де f_0 і g_0 – будь-які фіксовані нечіткі значення деякої вихідної змінної A та деякою керуючою змінною B відповідно.

Приклад 4.2. Нечіткий контролер.

Нехай процес містить:

– три вихідні змінні з нечіткими значеннями:

1) тиск (низький, середній, високий);

2) температура (холодно, досить тепло, тепло, гаряче);

3) вологість (сухо, волого, сиро);

– дві керуючі змінні з нечіткими значеннями:

1) подача води (відсутня, середня, максимальна);

2) інтенсивність нагріву (відсутня, низька, середня, максимальна);

– два правила:

1) ЯКЩО тиск *низький* ТО вентиль наполовину відкритий.

Значення f_0 – тиск низький означає, що нечітке значення вихідний змінної "тиск" низьке. Значення g_0 – вентиль наполовину відкритий означає, що "вентиль" відкритий наполовину.

2) ЯКЩО температура *холодно* І вологість *сиро* ТО нагрівач, максимальний.

Посилання правила f_0 – температура *холодно* І вологість *сиро* комбінує нечіткі значення вихідних змінних "температура" та "вологість". Значення g_0 – нагрівач, максимальний означає, що нечітке значення керуючої змінної "нагрівач" максимально.

Значення f_0 в умовній частині правила може бути або простим нечітким значенням однієї вихідної змінної, або комбінацією нечітких значень кількох вихідних змінних. Нечітке значення g_0 у заключній частині правила майже завжди представляє просте нечітке значення лише однієї керуючої змінної.

Імплікація

Правила, подібні " ЯКЩО f_0, ТО g_0", у булевій логіці називаються імплікацією, де f_0 і g_0 – булевські сутності зі значеннями істинності 0 чи 1.

Формально імплікацію записують як $f_0 \to g_0$ кожної пари значень, вона повертає результат згідно табл. 4.1.

Таблиця 4.1 – Таблиця істинності

f_0	g_0	$f_0 \to g_0$	НЕ $f_0 \vee g_0$
0	0	1	1
0	1	1	1
1	1	1	1
1	0	0	0

З табл. 4.1 видно, що операція $f_0 \to g_0$ еквівалентна (НЕ f_0) $\vee$ g_0.

У нечіткій логіці імплікація нечітких значень f_0 і g_0, $f_0 \to g_0$ визначається як нове комбіноване нечітке значення.

Наприклад, нечітка імплікація: (НЕ f_0) АБО g_0.

Функція належності:

$$\mu(a, b, f_0 \to g_0) = \mu[a, b, (\text{НЕ } f_0) \text{ АБО } g_0] = S[1 - \mu(a, f_0), n(b, g_0)],$$

де a і b – будь-які два базові значення об'єктних змінних A і B, відповідно $f_0 \in A$ і $g_0 \in B$, $\mu(a, f_0)$ і $n(b, g_0)$ – часткові функції належності A і B.

Зіставлення стану процесу та правил нечіткого контролера

Розглянемо процес із вихідною змінною A, керуючою змінною B і правилом, що пов'язує нечіткі значення f_0 сутності A і g_0 сутності B (табл. 4.2).

Таблиця 4.2 – Характеристики керованого процесу

Значення	Вхідна змінна A	Керуюча змінна B
Базове значення	$\{a\}$	$\{b\}$
Нечітке значення	$\{f\}$	$\{g\}$
Функція належності	$\mu(a, f)$	$n(b, g)$
Вибране нечітке значення	$f_0 \in \{f\}$	$b_0 \in \{g\}$
Що представляють чіткі значення	$\alpha(t) \in \{a\}$	$\beta \in \{b\}$
Правило	ЯКЩО f_0 ТО b_0	Не існує

В будь-який час t чітке значення $\alpha(t)$ вихідної змінної A зіставляється з нечітким значенням f_0, тобто, з функцією належності $\mu(\alpha, f_0)$, і в будь-який час t, і для будь-якого чіткого значення b керуючої змінної B пара (α, b) зіставляється з правилом $f_0 \rightarrow g_0 =$ (НЕ f_0) АБО g_0 з належністю

$$\mu(\alpha, b, f_0 \rightarrow g_0) = S[1 - \mu(\alpha, f_0), n(b, g_0)].$$

Відповідно до умови необхідно розглянути кон'юнкцію нечіткого значення та правила, а саме f_0 I $[f_0 \rightarrow g_0] = f_0$ I [(НЕ f_0) АБО g_0].

В будь-який час t і для будь-якого чіткого значення b від множини B пара (α, b) належить новій нечіткій множині з належністю

$$\rho(\alpha, b, f_0 \text{ I } [(\text{НЕ } f_0) \text{ АБО } g_0]) = T\{\mu(\alpha, f_0), S[1 - \mu(\alpha, f_0), n(b, g_0)]\}.$$

Для фіксованого часу t та фіксованого значення $\alpha(t)$ вказана функція залежить від b.

Вибір чіткого значення керуючої змінної

Для будь-якого фіксованого $t = t_0$ чітке значення $b = \beta(t_0)$. Керуюча змінна B вибирається так, щоб значення належності пари $[\alpha(t_0), b]$, тобто значення виразу

$$\rho(b) = f_0 \text{ I } [(\text{НЕ } f_0) \text{ АБО } g_0] = T\{\mu(\alpha, f_0), S[1 - \mu(\alpha, f_0), n(b, g_0)]\}$$

було максимально для $b = \beta(t) = b_{\max}$.

Розглянемо використання функції $\rho(b)$ у ситуації, коли вихід процесу відповідає умовній частині правила.

Допустимо, що вихідна змінна має необхідне значення з високою достовірністю. Це означає, що $\mu \approx 1$, і, отже, $\rho(b) \approx T\{1, S[0, n(b, g_0)]\} = T\{1, n(b, g_0)]) = n(b, g_0)$. Відповідно до закону управління значення b має

бути вибрано так, щоб належність $n(b, g_0)$ була максимальною.

Розглянемо ситуацію, в якій вихід процесу більше або менше відповідає правилу.

Припустимо, що значення вихідних змінних процесу більш або менш прийнятні ($\mu \leq 0{,}5$), тоді

$$S[1 - \mu, n(b, g_0)] \geq \max[1 - \mu, n(b, g_0)] \geq 0{,}5 \geq \mu$$

і, отже,

$$\rho(b) = T\{\mu, S[1 - \mu, n(b, g_0)]\} \leq min\{\mu, S[1 - \mu, n(b, g_0)]\} \leq \mu.$$

Чим менше значення μ, тим нижчий ступінь належності $\rho(b)$ для будь-якого b, тому вибір b у меншій мірі обґрунтований, отже, правило визначення змінної, що управляє, тільки більше або менш прийнятно.

Розглянемо ситуацію - вихід процесу незадовільний ($\mu \approx 0$).

Функція $\rho(b) \approx T\{0, S[1, n(b, g_0)]\} = T\{0, 1\} = 0$ незалежно від b, і вибір b не підходить для будь-якого значення. Згідно з щоденним досвідом правила, умови яких не задовольняються, не використовують.

4.1.3. Комбінування умов

Правила з комбінуванням умов подібні наступним

ЯКЩО тиск *низький* АБО тиск с*ередній* ТО вентиль відкритий

або

ЯКЩО температура *холодно* І вологість *сиро* ТО нагрівач, максимальний

обробляються аналогічно. Не має значення, чи комбінуються умови, що посилаються на одну або більше вихідних змінних. У усіх випадках для чітких вихідних значень $\alpha_1(t), ..., \alpha_N(t)$ під час t визначається міра належності $\mu(\alpha_1(t), \ldots, \alpha_N(t), f_0)$ до комбінованого нечіткого значення f_0. Далі обчислення

проводяться таким чином: визначається функція $\rho(b) = T\{\mu(\alpha_1(t), \ldots, \alpha_N(t), f_0), n(b, g)\}$, і знаходиться оптимальне значення b.

4.1.4. Накопичення результатів та дефазифікація

Агрегація результатів кількох правил

Розглянемо множину правил керування автомобілем.

Приклад 4.3. Агрегація результатів правил:

ЯКЩО до перешкоди *близько* ТО швидкість *повільно*.

ЯКЩО час *мало* ТО швидкість *швидко*.

ЯКЩО дата *кінець тижня* ТО швидкість *швидко*.

ЯКЩО паливо *порожнє* ТО швидкість *стоп*.

ЯКЩО ресурси *мало* ТО швидкість *економія*.

Всі правила містять вихідну змінну "швидкість" з нечіткими значеннями "стоп", "повільно", "економія" та "швидко".

Виконання правил може призвести до неузгоджених або суперечливих висновків, які залежать від того, які умови працювали у певній ситуації. Якщо, наприклад, час і гроші малі одночасно, чи повинен автомобіль їхати "швидко" та "економно"? Якщо використовується кілька правил одночасно, необхідно вирішити, яке з них виконується. Операція АБО у цьому контексті означає "виключне АБО".

Правило з умовою f_0 та висновком g_0 еквівалентно нечіткому значенню (f_0 I g_0). В будь-який час t і за певного b чітке вихідне значення процесу виходить із функцій належності $\rho(b) = T\{\mu(\alpha_1(t), \ldots, \alpha_N(t), f_0), n(b, g)\}$ для f_0 I g_0. Наведемо кілька правил, які виконуються разом та еквівалентні комбінації нечітких значень:

$$(f_1 \text{ I } g_1) \text{ АБО } (f_2 \text{ I } g_2) \text{ АБО } \ldots \text{ АБО } (f_N \text{ I } g_N)$$

з функціями належності $\rho(b) = S\{\rho_1(b), \rho_2(b), ... \rho_N(b)\}$, де

$$\rho_1(b) = T\{\mu(\alpha_1(t), \ldots, \alpha_N(t), f_0), n(b, g_1)\};$$
$$\rho_2(b) = T\{\mu(\alpha_1(t), \ldots, \alpha_N(t), f_0), n(b, g_2)\};$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\rho_N(b) = T\{\mu(\alpha_1(t), \ldots, \alpha_N(t), f_0), n(b, g_N)\}.$$

У розглянутому вище виразі $\rho(b) = S\{\rho_1(b), \rho_2(b), ... \rho_N(b)\}$ у будь-який фіксований час t функція b відрізняється від оптимального значення $b = \beta$.

Дефазифікація

Далі розглянемо, як отримати чітке оптимальне значення β керуючої змінної b. Ця процедура називається дефазифікацією і є останнім кроком кожного керуючого циклу в нечіткій системі. У загальному випадку змінна $b = \beta$ має часткову функцію належності

$$\rho(b) = S\{\rho_1(b), \rho_2(b), ... \rho_N(b)\} = S\{T[\mu(\alpha_1(t), n(b, g_1)], T[\mu(\alpha_2(t), n(b, g_2)], \ldots, T[\mu(\alpha_N(t), n(b, g_N)]\}.$$

Для дефазифікації нечіткого результату практично використовують три методи, які розглянемо нижче.

Метод максимальної висоти. Застосовується, якщо функція $\rho(b)$ має певний абсолютний максимум, як певне значення $b = b_{max}$, при якому $\rho(b_{max}) > \rho(b)$ для всіх b. Перевага методу полягає у простоті застосування, а недолік полягає в тому, що функція $\rho(b)$ не розглядається у всій області визначення, а лише у максимальних позиціях, що може призвести до неприйнятних результатів.

Метод середнього максимуму. Якщо функція $\rho(b)$ має кілька відносних максимумів, можна взяти зважене середнє. Враховуючи загальну

форму вираження ρ(*b*), цей метод кращий за попередній. Він також застосовується, якщо ρ(*b*) постійна на певних інтервалах *b* (що часто трапляється на практиці), тому що можна взяти центри кожного такого інтервалу як "відносні локальні максимуми" і потім усереднити ці значення.

Метод центру гравітації. Графічне представлення функції $\rho(b)$ в декартових координатах – це крива, яка разом з віссю абсцис *X*, верхньою і нижньою межами *b* обмежує плоску область. Абсциса β центру гравітації такої фігури береться як бажаний оптимум

$$M = \int_c^d b \cdot \rho(b)\, db, \quad F = \int_c^d \rho(b)\, db, \quad b_{opt}=M/F.$$

Якщо ρ(*b*) визначається тільки для дискретних значень *b* (так звані синглтони), то обчислення центру гравітації модифікується таким чином:

$$M = \sum_n b_n \cdot \rho(b_n)\, db, \quad F = \sum_n \rho(b_n)\, db, \quad b_{opt}=M/F.$$

4.2. Нечіткі системи управління динамічними процесами

4.2.1. Моделювання кочення кулі по гойдалці

Введення команди *slbb* у вікні *Command Window* пакету MATLAB призводить до появи у вікні *Simulink* структурної схеми нечіткої системи управління (рис. 4.1). Завданням керування в даному випадку є підтримка такого стану кулі, що перекочується на гойдалках, при якій вона не міг би з них скотитися.

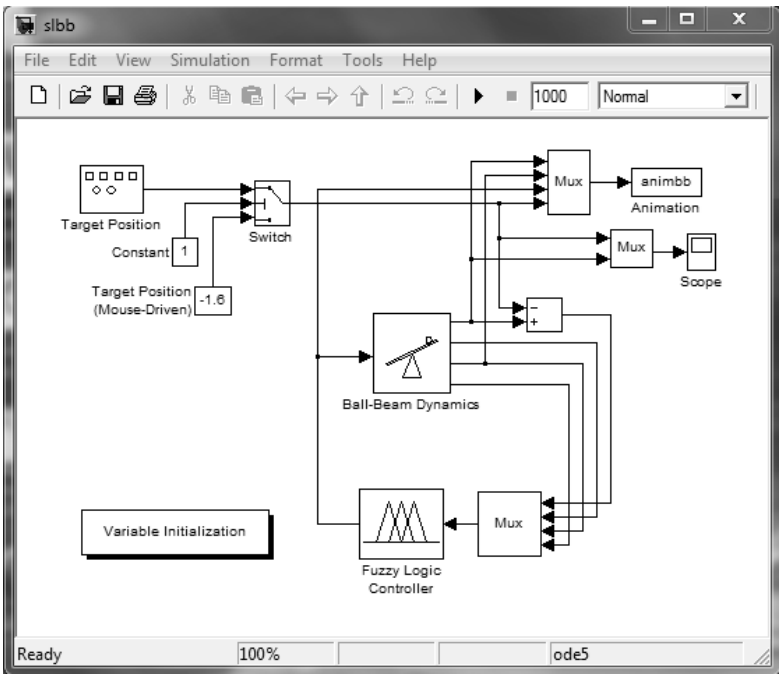


Рис. 4.1. Структурна схема нечіткої системи управління

Запуск моделювання призводить до появи анімаційної картинки (рис. 4.2), що ілюструє як представлена система, виробляючи керуючі впливи, що "підштовхують" то ліву, то праву половину гойдалки, не дає кульці скотитися з них. Точка поштовху відображається вершиною трикутника, що переміщається вздовж коромисла гойдалок.

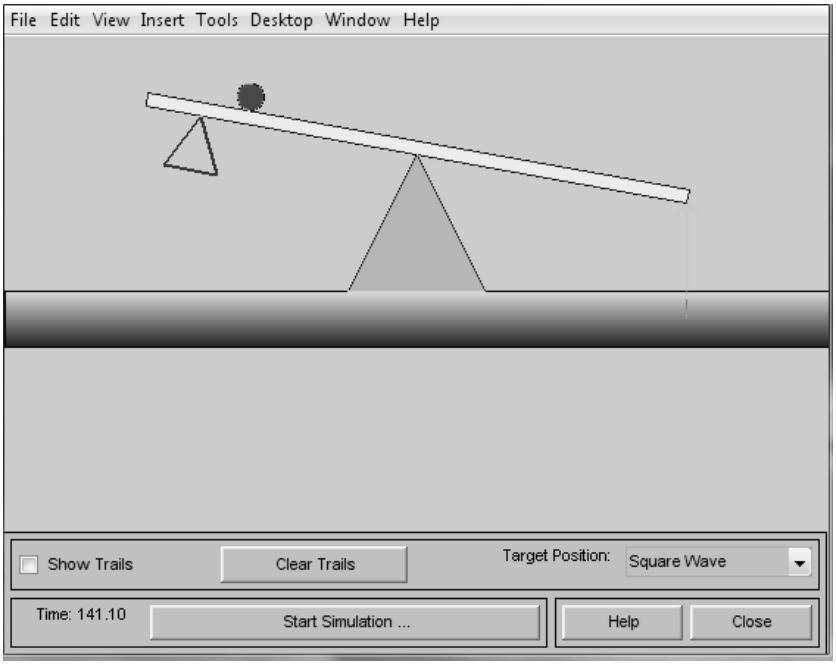


Рис. 4.2. Динамічна система "куля на гойдалці"

За бажанням користувач може переглянути основні сигнали системи (рис. 4.3).

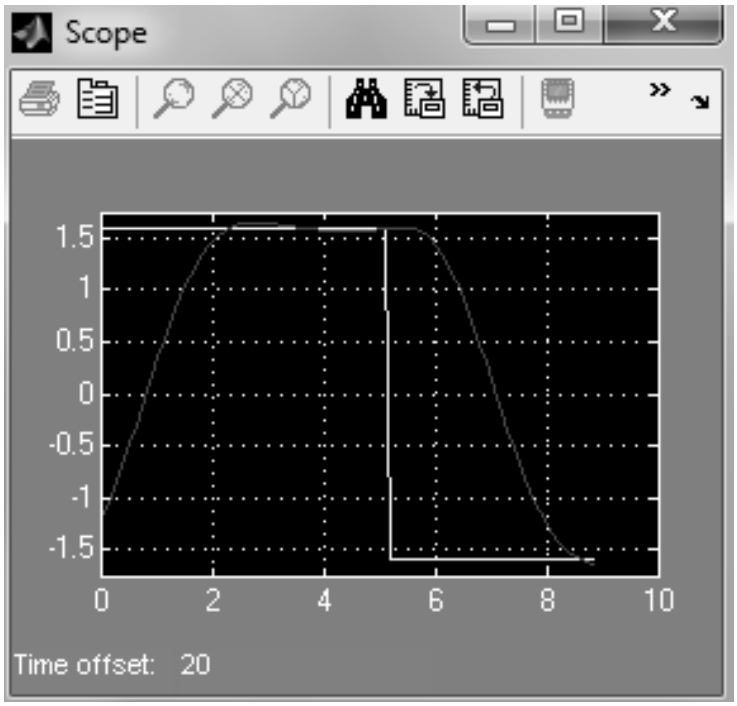


Рис. 4.3. Основні сигнали системи – значення необхідного та поточного розташування кулі на гойдалці

Подвійне клацання мишею на блоці діаграми моделі дозволяє вивести вікно з параметрами відповідного блоку. Так, на рис. 4.4 показано таке вікно блоку нечіткої логіки. Неважко помітити, що він характеризується єдиним параметром - ім'ям файлу, що задає структуру блоку.

Рис. 4.4. Вікно блоку нечіткої логіки

Список *Target Position* (рис. 4.2) дозволяє вибрати один із чотирьох видів впливів:

– *Sinusoid Wave* – синусоїдальна хвиля;

– *Square Wave* – прямокутна хвиля;

– *Saw Wave* – трикутна хвиля;

– *Mouse-Driver* – вплив, що задається мишею.

Опція *Show Trails*, що дозволяє накопичувати зміни положення кулі, коромисла гойдалок і точки на них. Завдяки цьому можна отримати чіткіше уявлення про динаміку роботи аналізованої системи. Приклад накопичення інформації про положення кулі, коромисла гойдалок та точок впливу наведено на рис. 4.5.

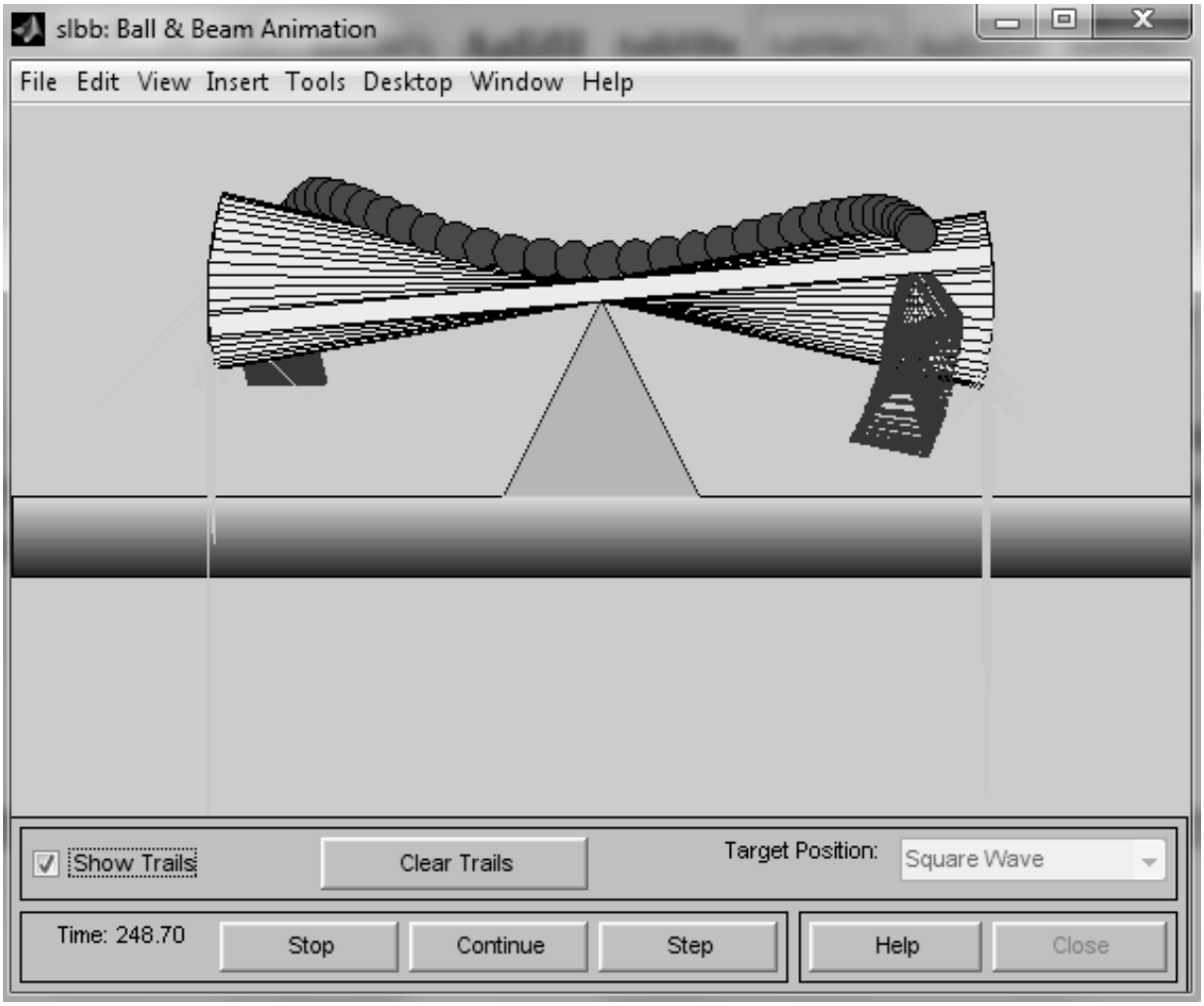


Рис. 4.5. Ілюстрація керованого об'єкта в режимі накопичення

4.2.2. Моделювання відскоків кулі від гойдалок

Схожий на описаний вище приклад з назвою *juggler* імітує складніший рух кулі (рис. 4.6). Тут куля відскакує від коромисла гойдалок, які переміщуються до точки падіння кулі. При цьому вихідне положення коромисла змінюється таким чином, щоб шар при можливості відскакував вертикально вгору. Викид кулі за межі області переміщення гойдалок виключається з урахуванням відскоків від лівої чи правої стінок.

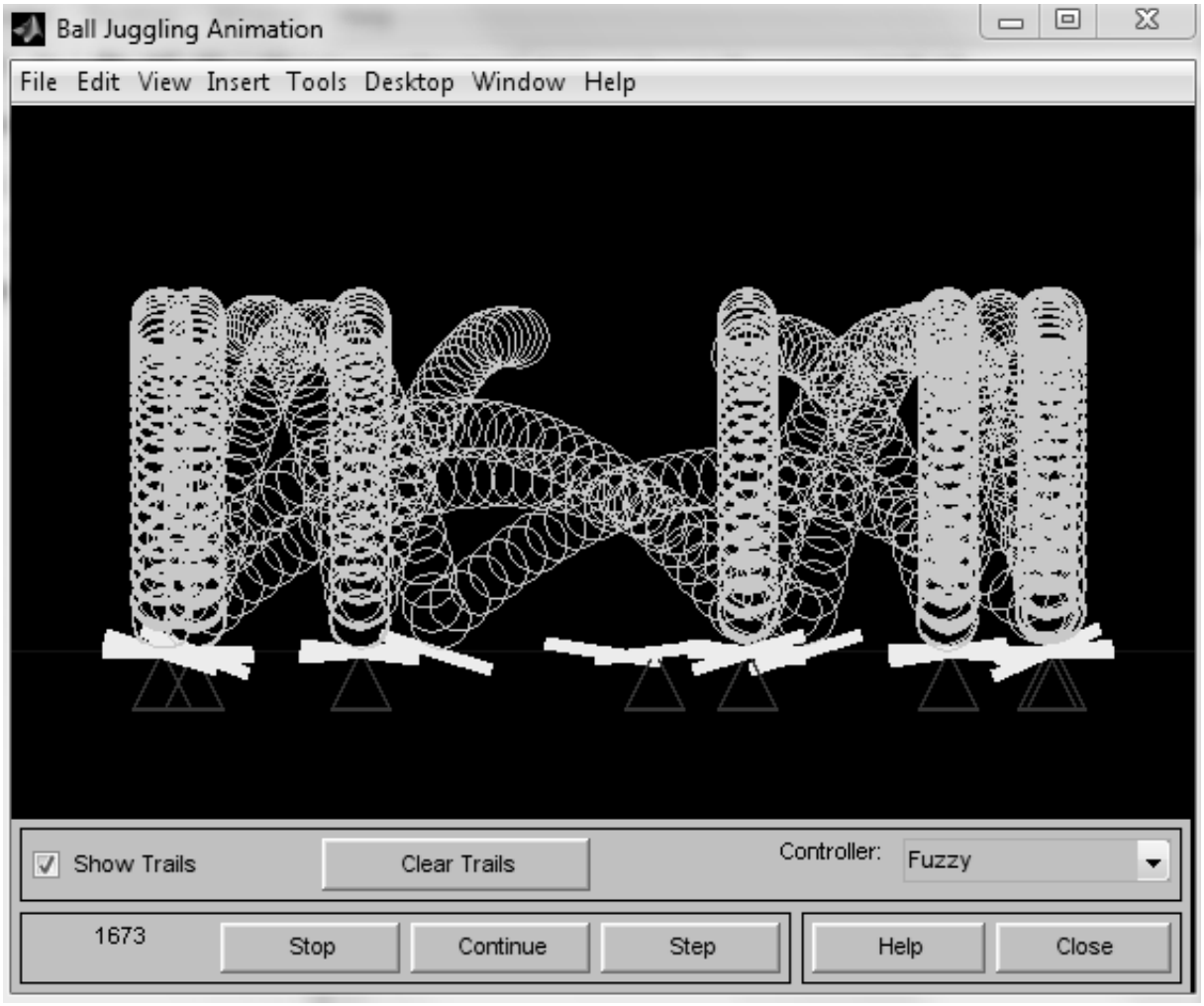


Рис. 4.6. Динаміка відскоків кулі при використанні нечіткої логіки

У цьому прикладі особливо цінною є можливість моделювання у двох режимах відскоку кульки від коромисла гойдалок – з використанням нечіткої логіки та математичної моделі. Останній випадок подано на рис. 4.7. Неважко помітити, що в цьому випадку взагалі не вдається

домогтися відскоку кульки строго вертикально вгору - діє правило "кут відбиття дорівнює куту падіння".

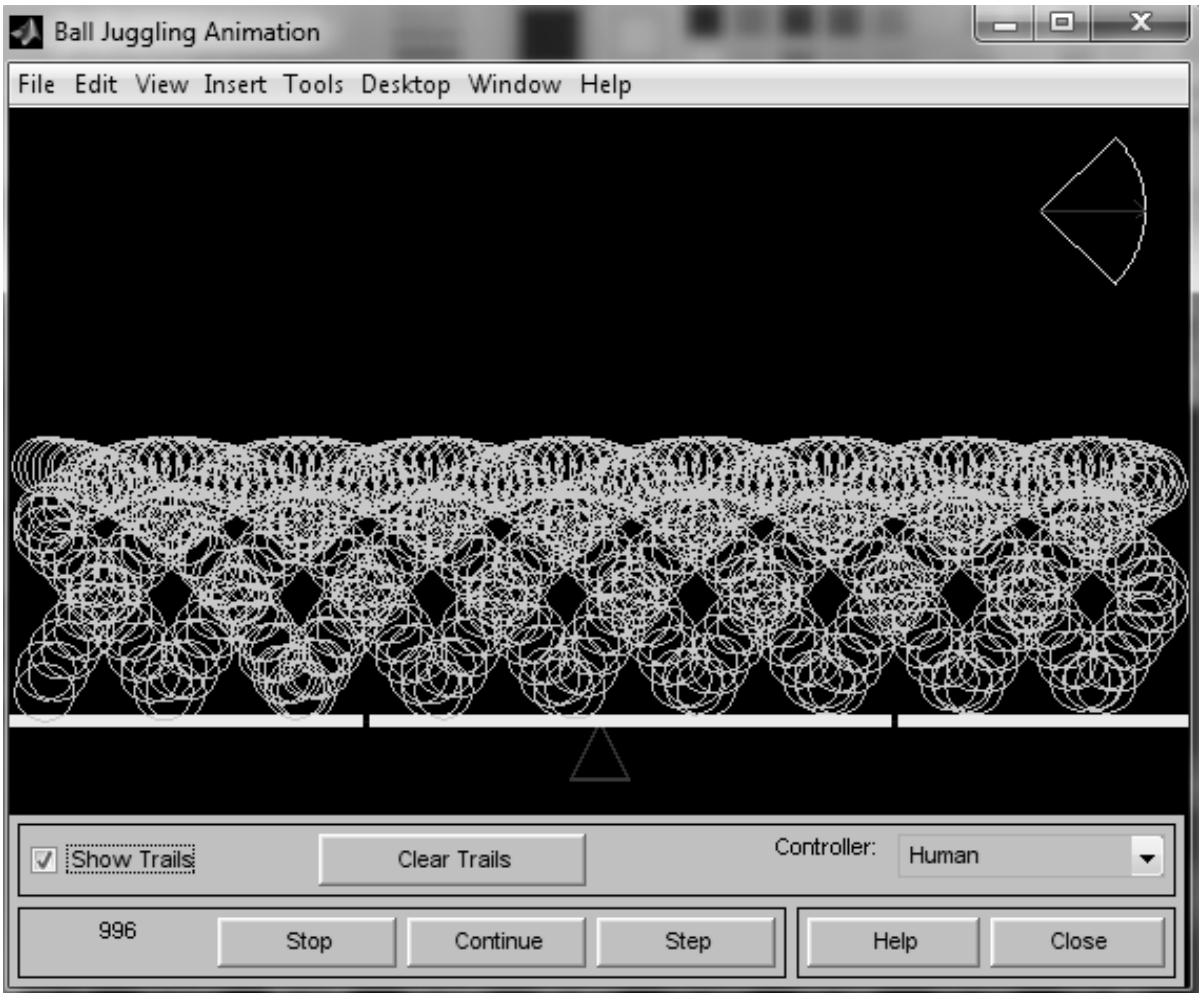


Рис. 4.7. Динаміка відскоків кулі при використанні математичної моделі

Поряд із вікном моделі, у цьому прикладі виводиться вікно контролю *Rule Viewer*, показане на рис. 4.8. Воно дозволяє уточнити параметри відскоків кульки.

Цей приклад реалізований без застосування *Simulink*. Код програми у MATLAB, що реалізує його, можна переглянути, активізувавши мишею гіперпосилання *View code for juggler* у вікні демонстрації цього прикладу. Програма займає трохи більше 700 рядків і переглядається у редакторі програм системи MATLAB.

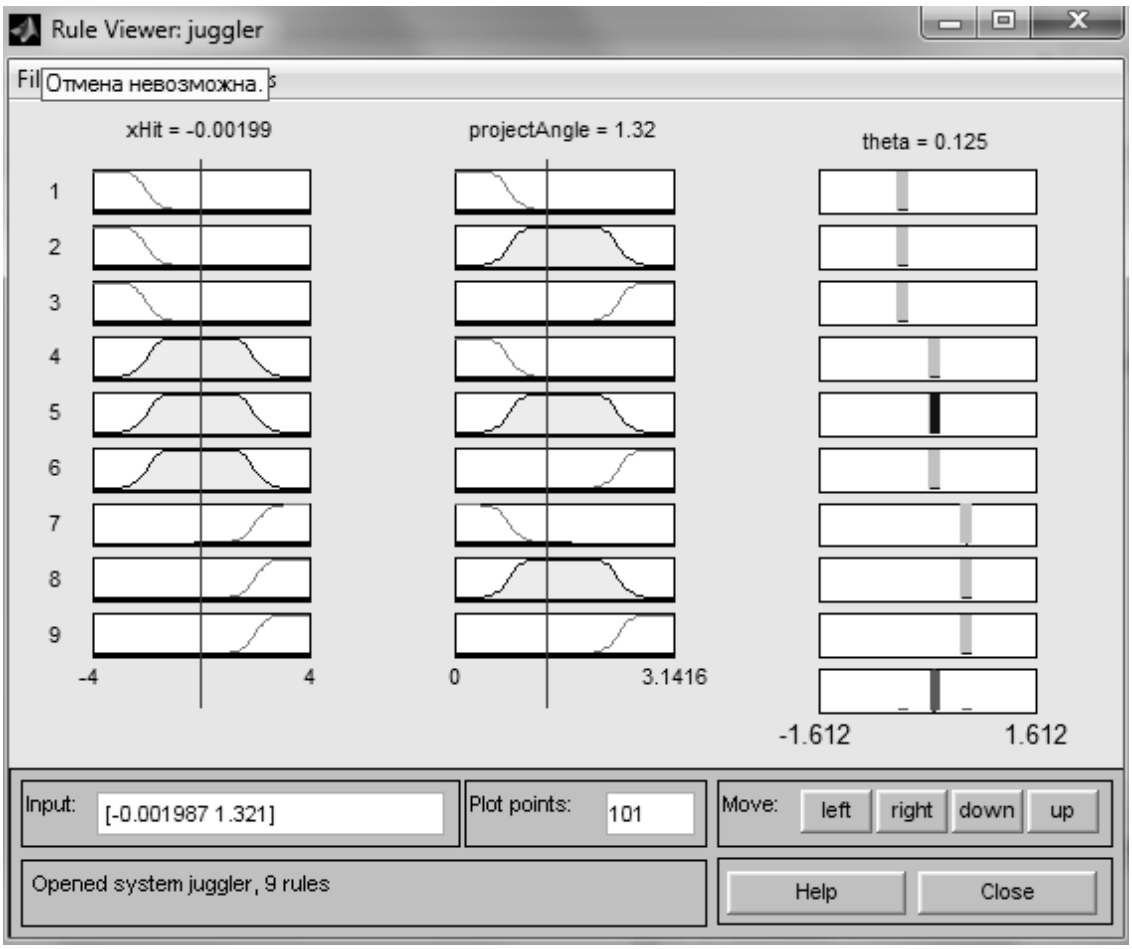


Рис. 4.8. Вікно контролю *Rule Viewer*

4.2.3. Система керування змішувачем води

Введення команди *shower* у вікні *Command Window* пакету MATLAB призводить до появи у вікні *Simulink* структурної схеми нечіткої системи керування змішувачем холодної та гарячої води.

З роботою змішувача холодної та гарячої води ми стикаємося практично щодня. Якщо напір води постійний, то особливої необхідності в керуванні цим простим пристроєм немає – достатньо ручку змішувача встановити в потрібне положення та проконтролювати рукою температуру води. Але, якщо натиск води постійно змінюється, то без автоматичного регулятора температури не обійтися. У демонстраційному прикладі розглядається замкнута система регулювання температури з використанням нечіткої логіки. *Simulink*-діаграми системи наведено на рис. 4.9.

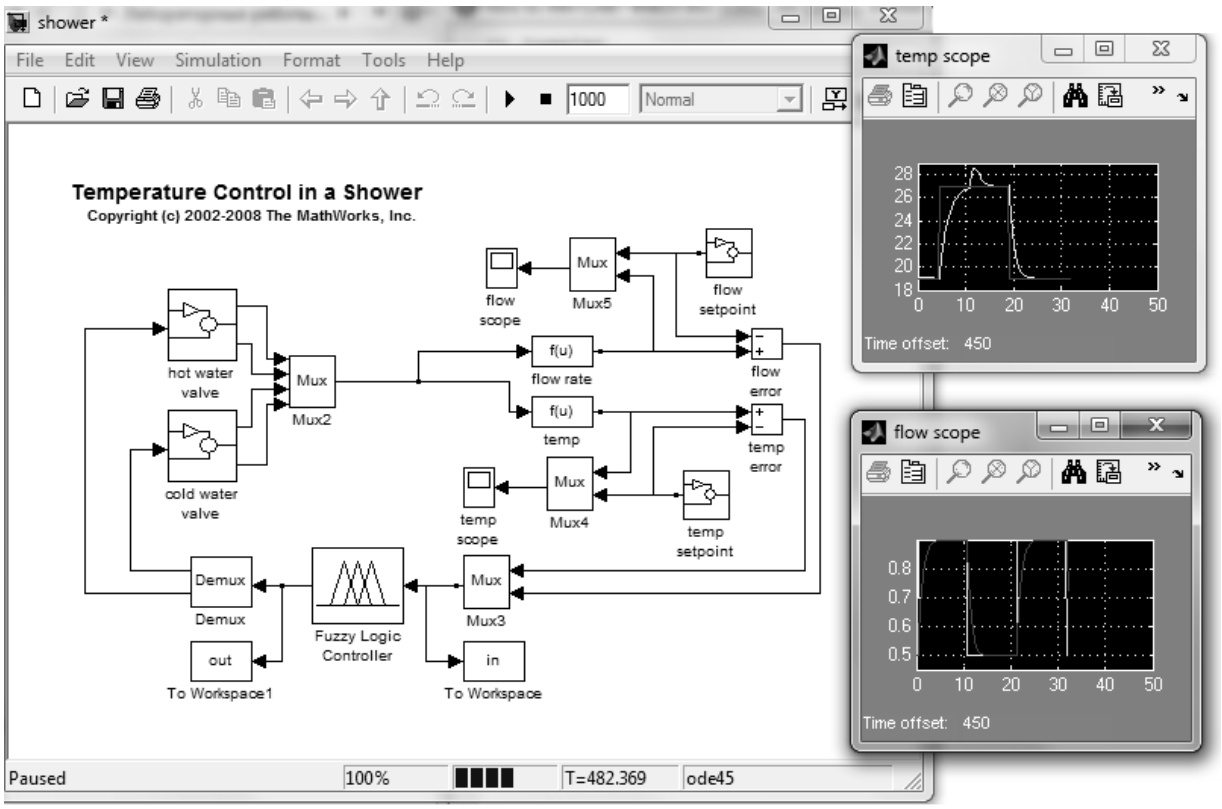


Рис. 4.9. *Simulink*-діаграми системи керування змішувачем води

В системі передбачено контроль температури води, як при ручному керуванні, так і в умовах зміни тиску гарячої та холодної води після заданої ручної установки. Діаграми потоку та температури води виводяться на віртуальні осцилографи.

4.2.4. Система управління перевернутим маятником.

Введення команди *slcp1* у вікні *Command Window* пакету MATLAB призводить до появи у вікні *Simulink* структурної схеми нечіткої системи керування перевернутим маятником (рис. 4.10).

Запуск моделювання призводить до появи анімаційної картинки (рис. 4.11), що ілюструє переміщення в просторі динамічної системи "перевернутий маятник".

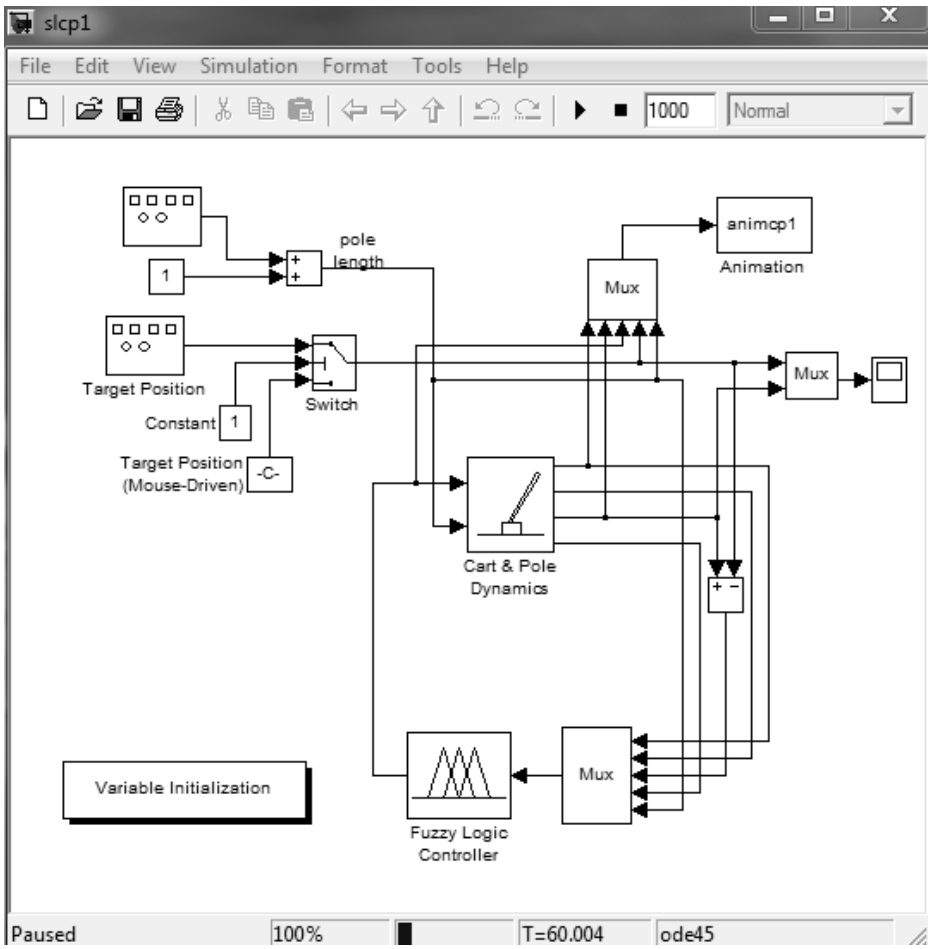


Рис. 4.10. Структурна схема нечіткої системи управління "перевернутий маятник"

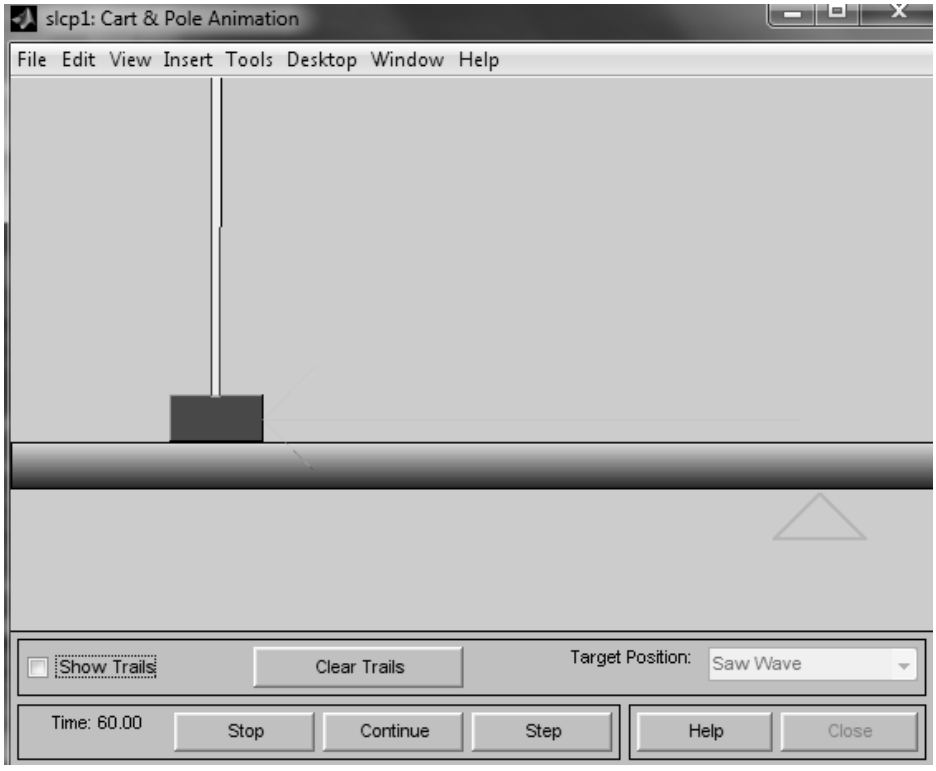


Рис. 4.11. Динамічна система "перевернутий маятник "

4.2.5. Система керування двома перевернутими маятниками

Введення команди *slcpp1* у вікні *Command Window* пакету MATLAB призводить до появи у вікні *Simulink* структурної схеми нечіткої системи управління двома перевернутими маятниками (рис. 4.12).

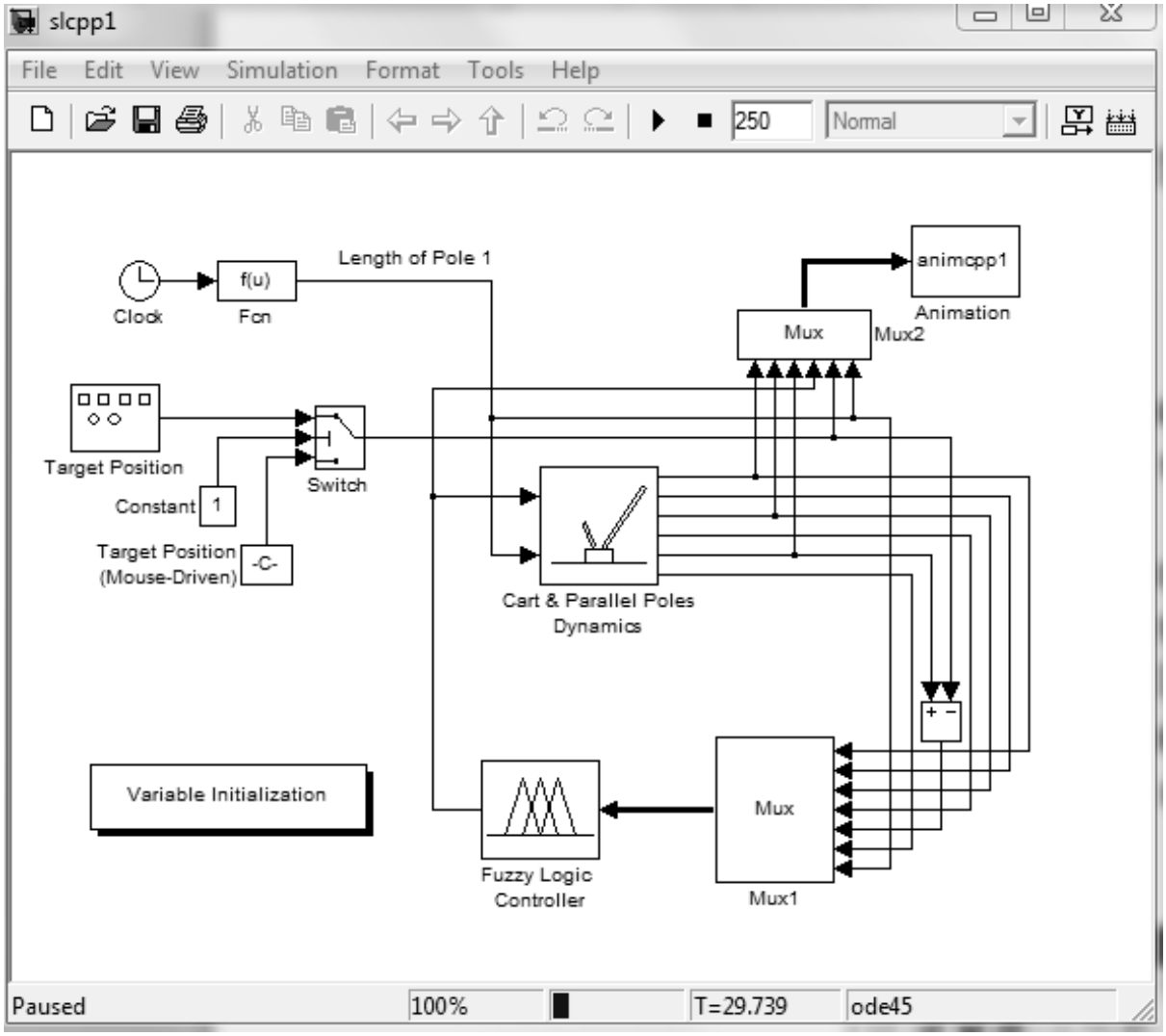


Рис. 4.12. Структурна схема нечіткої системи керування

Запуск моделювання призводить до появи анімаційної картинки (рис. 4.13), що ілюструє переміщення в просторі динамічної системи "Перевернуті маятники".

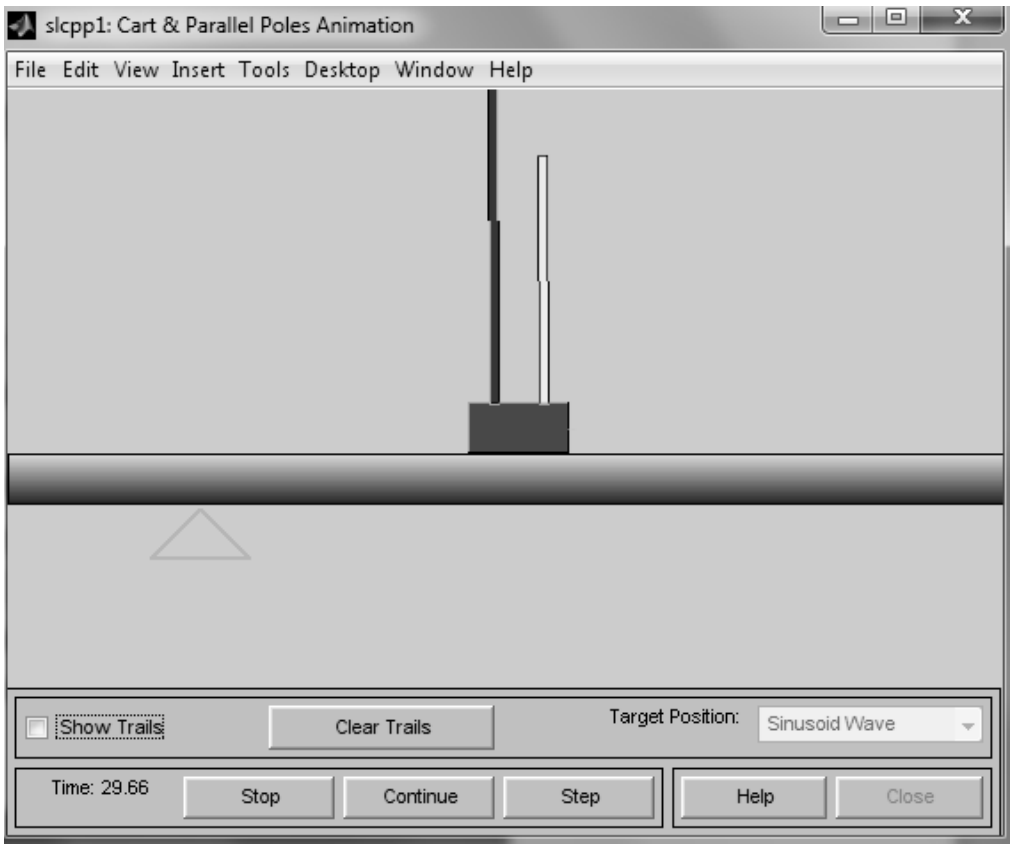


Рис. 4.13. Динамічна система "перевернуті маятники"

4.3. Індивідуальні завдання

1. Необхідно вивчити методи побудови нечітких контролерів засобами нечіткої логіки і блоків моделювання *Simulink*.

2. Промоделювати в *Simulink* усі п'ять нечітких контролерів, які керують різними динамічними процесами.

3. Отримати і привести в звіті словесний опис і графічне відображення правил нечіткого виведення розглянутих нечітких регуляторів.

4. Описати умови і визначити параметри системи, при яких відбувається "зрив" управління усіх п'яти нечітких контролерів, які керують різними динамічними процесами.

5. Оформіть звіт по лабораторній роботі.

Лабораторна робота 5

РЕГУЛЮВАННЯ З ВИКОРИСТАННЯМ НЕЧІТКОГО КОНТРОЛЕРА

Мета лабораторної роботи: отримання та закріплення знань, формування практичних навичок роботи з методами побудови нечітких контролерів засобами інструментарію нечіткої логіки (*Fuzzy Logic*) і блоків моделювання *Simulink* пакету MATLAB.

5.1. Короткі відомості з теорії

Пакет нечіткої логіки для системи MATLAB – це пакет прикладних програм, що належать до теорії нечітких множин і дозволяють конструювати так звані нечіткі експертні та керуючі системи. Основні можливості пакету:

1. Побудова систем нечіткого виведення (експертних систем, регуляторів, апроксиматорів залежностей).

2. Побудова адаптивних нечітких систем (гібридних нейронних мереж).

3. Інтерактивне динамічне моделювання в *Simulink*.

До складу програмних засобів *Fuzzy Logic Toolbox* входять такі основні програми, що мають графічний інтерфейс, редактор нечіткої системи виводу *Fuzzy Inference System Editor* (*FIS-редактор*) разом із допоміжними програмами: редактором функцій належності (*Membership Function Editor*), редактором правил (*Rule editor*), переглядачем правил (*Rule Viewer*) та переглядачем поверхні відгуку (*Surface Viewer*). FIS-редактор запускається з командного рядка командою *Fuzzy*.

Системи нечіткого висновку, створені тим чи іншим чином за допомогою пакету *Fuzzy Logic Toolbox*, допускають інтеграцію з

інструментами пакета *Simulink*, що дозволяє виконувати моделювання систем у рамках останнього.

Розглянемо приклад побудови нечіткого контролера засобами інструментарію нечіткої логіки (ІНЛ) *Fuzzy Logic* та блоків моделювання *Simulink*, який здійснює контроль рівня води у баку. На рис. 5.1 зображено об'єкт керування у вигляді бака з водою, до якого підходять дві труби: через одну трубу, з краном, вода затікає в бак, через іншу – витікає.

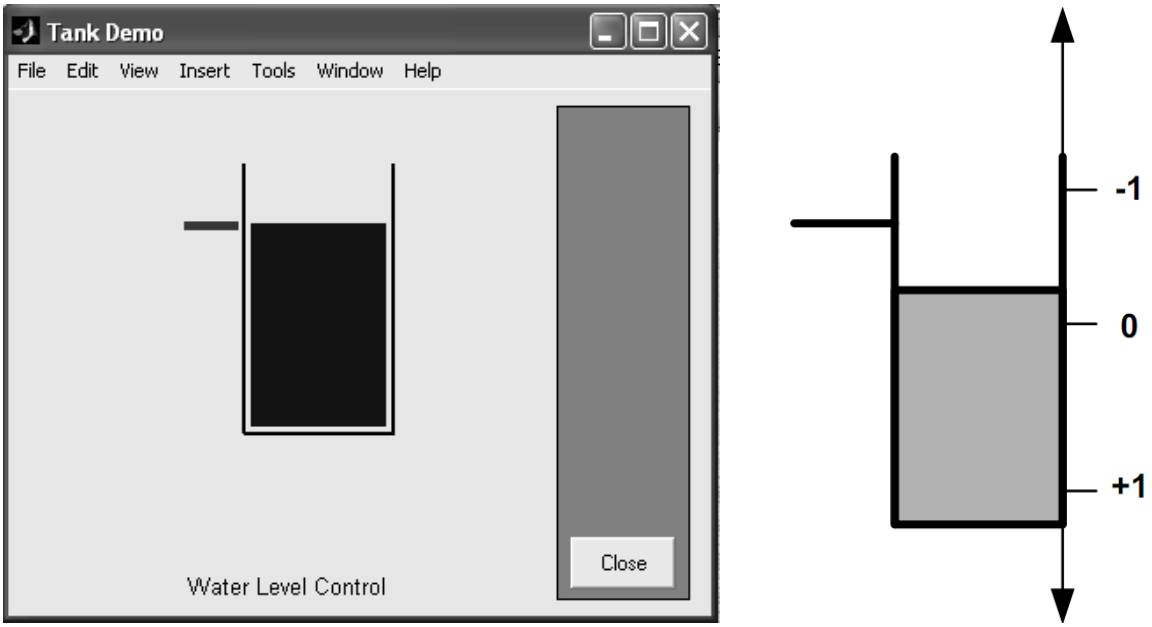


Рис. 5.1. Схематичне подання об'єкта управління (бак із водою)

Подачу води в бак можна регулювати, більше або менше відкриваючи кран. Витрата води неконтрольована і залежить від діаметра вихідної труби (він фіксований) і від поточного рівня води в баку. Якщо розуміти під вихідною (регульованою) змінною рівень води, а під регулюючим елементом – кран, можна відзначити, що подібний об'єкт регулювання, з погляду його математичного опису, є динамічним і нелінійним.

Визначимо мету управління тут як встановлення рівня води в баку на необхідному (змінюваному) рівні та спробуємо вирішити відповідне завдання управління засобами нечіткої логіки.

Очевидно, що на регулятор, який забезпечує досягнення мети управління, повинна надходити інформація про невідповідність (різницю) необхідного і фактичного рівнів води, при цьому даний регулятор повинен виробляти керуючий сигнал на регулюючий елемент (кран).

Порядок побудови нечіткого контролера:

1. Задати для кожної з вхідних та вихідних змінних функції приналежності.

2. Розробити базу правил для реалізованої нечіткої системи.

3. Вибрати алгоритм нечіткого логічного виводу (Мамдані або Сугено) та виконати аналіз результатів роботи створеної системи.

У першому наближенні функціонування регулятора можна описати набором таких правил:

1. *If* (*level is okay*) *then* (*valve is no_change*);

2. *If* (*level is low*) *then* (*valve is open_fast*);

3. *If* (*level is high*) *then* (*valve is close_fast*);

4. *If* (*level is okay*) *and* (*rate is positive*) *then* (*valve is close_slow*);

5. *If* (*level is okay and* (*rate is negative*) *then* (*valve is open_slow*),

що в перекладі означає:

1. ЯКЩО рівень відповідає *заданому*, ТО кран без зміни;

2. ЯКЩО рівень *низький*, ТО кран швидко відкрити;

3. ЯКЩО рівень *високий*, ТО кран швидко закрити;

4. ЯКЩО рівень відповідає заданому І його приріст – позитивний, ТО кран треба повільно закривати;

5. ЯКЩО рівень відповідає заданому І його приріст – негативний, ТО кран треба повільно відкривати.

5.1.1. Формування функції власності

Командою *Fuzzy* запустити FIS-редактор. За промовчанням пропонується алгоритм виведення типу Мамдані. Оскільки в системі має бути два входи, то через пункт меню *Edit/Add input* треба додати систему

другий вхід (у вікні редактора з'являється другий жовтий блок з ім'ям *input2*). Роблячи далі одноразове натискання на блоці *input1*, змінити його ім'я на "*level*", завершуючи введення нового імені натисканням клавіші <*Enter*>. Аналогічно встановити ім'я "*rate*" блоку *input2* і "*valve*" – вихідному блоку (праворуч вгорі) *output1*. Присвоїти ім'я всій системі, наприклад "*LevelControl*", виконавши це через пункт меню *File/Save* to *Workspace*. Вигляд вікна редактора після вказаних дій наведено на рис. 5.2.

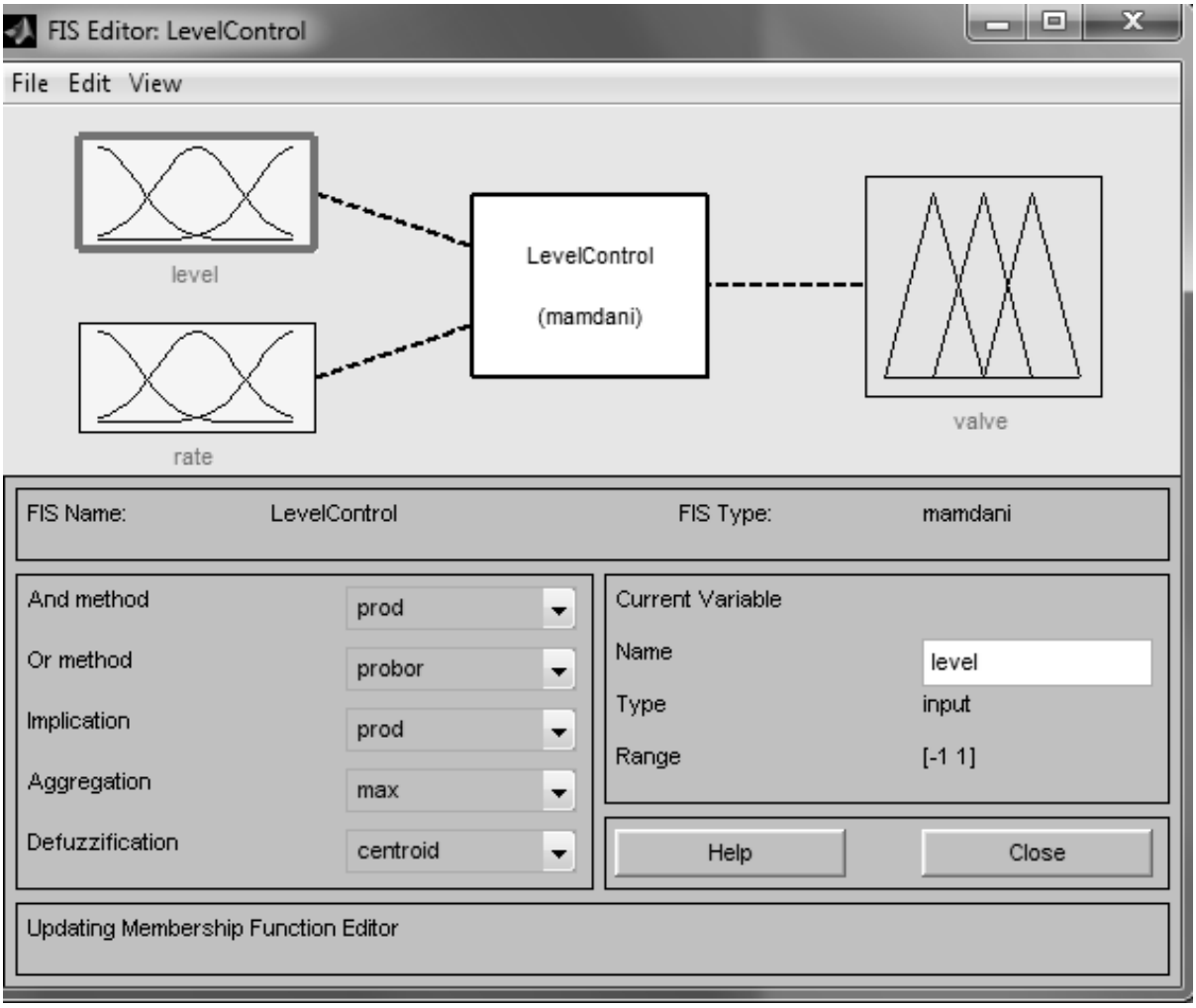


Рис. 5.2. Вигляд вікна *FIS*-редактора після завдання структури системи

Щоб сформувати функції приналежності для змінних, слід вибрати пункт меню *View/Edit membership functions* і вказати параметри системи відповідно до табл. 5.1. Тут використовуються такі функції. Гаусова функція *Gaussmf* має вигляд:

$$y = Gaussmf(x, [c, \sigma]) = \exp(-(x - c)^2 / (2\sigma)^2).$$

Функція трикутної форми *Trimf* визначається так:

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

Таблиця 5.1 – Параметри функцій належності для змінних *level*, *rate*, *valve*

Назва вхідних змінних	Назва функції належності	Тип функції належності
level	*High*	*Gaussmf* [0,3; –1]
	Okay	*Gaussmf* [0,3; 0]
	Low	*Gaussmf* [0,3; 1]
rate	*Negative*	*Gaussmf* [0,03; –0,1]
	None	*Gaussmf* [0,03; 0]
	Positive	*Gaussmf* [0,03; 0,1]
valve	*Close_fast*	*Trimf* [–1; –0,9; –0,8]
	Close_low	*Trimf* [–0,6; –0,5; –0,4]
	No_change	*Trimf* [–0,1; 0; 0,1]
	Open_low	*Trimf* [0,2; 0,3; 0,4]
	Open_fast	*Trimf* [0,8; 0,9; 1]

Сформувати правила виводу за допомогою *Rule Editor*. Для цього треба запровадити відповідні правила з підрозділу 5.1, як показано на рис. 5.3.

Відкрити через пункт меню *View/View rules* вікно перегляду правил та встановити значення змінних *level* та *rate*. Результат для виходу *valve* має бути показаний так само, як на рис. 5.4.

Відкрити через пункт меню *View/View surface* вікно перегляду графічної залежності вихідної змінної від вхідних.

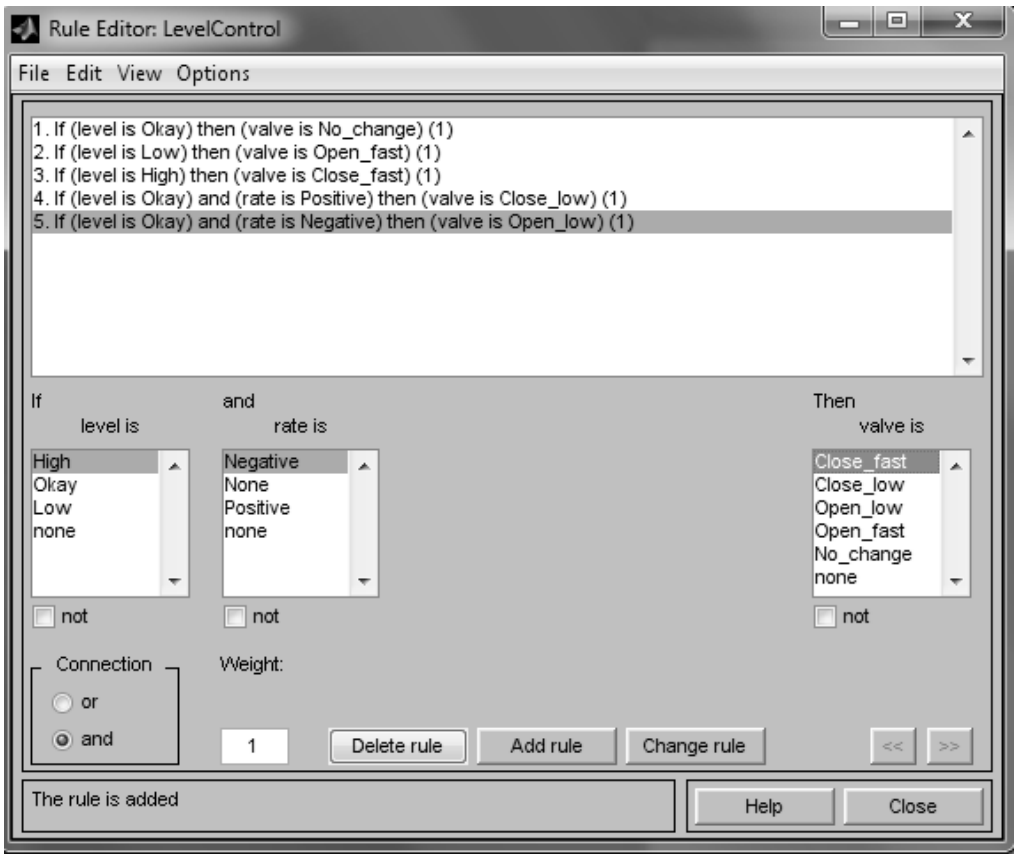


Рис. 5.3. Вікно редактора правил

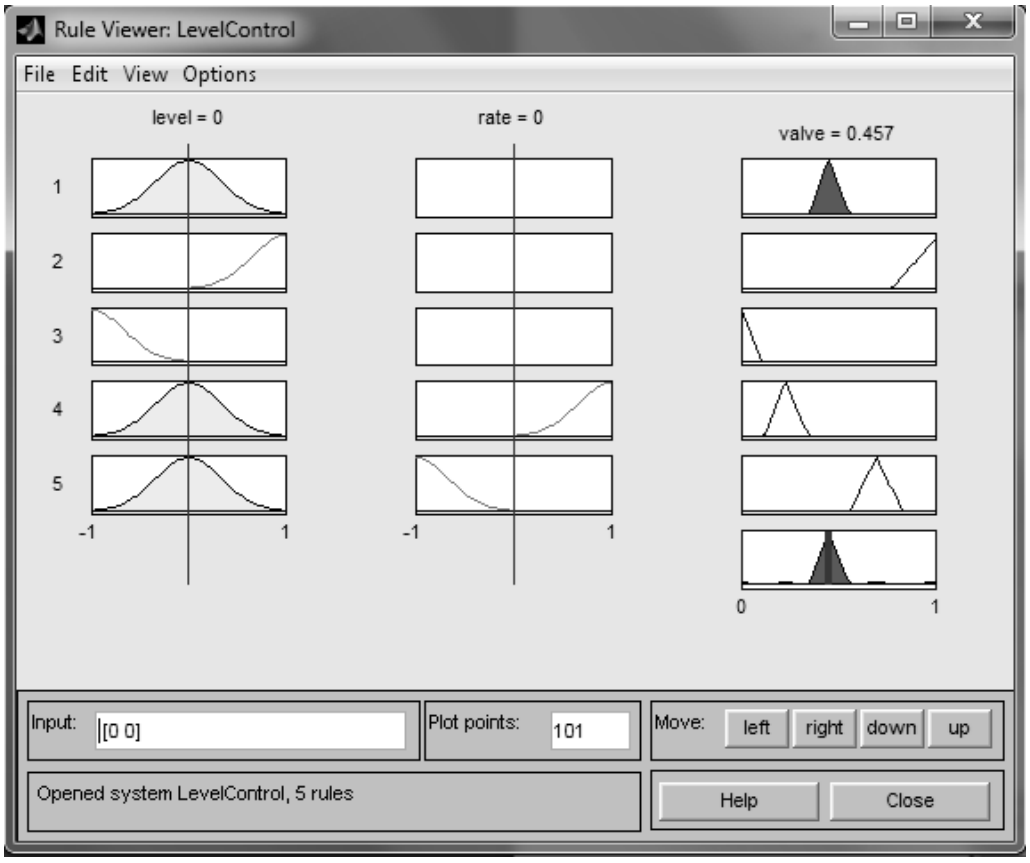


Рис. 5.4. Вікна перегляду правил у задачі при *level* = 0 і *rate* = 0

За потреби можна виконати експорт та імпорт результатів. У разі використання пунктів меню *File/Save to disk* на диску створюється текстовий (*ASCII*) файл із розширенням *fis*. Його можна переглядати, за необхідності редагувати поза системою MATLAB, а також використовувати повторно під час наступних сеансів роботи з системою.

5.1.2. Створення користувацьких функцій приналежності

Якщо жодна з вбудованих функцій приналежності не підходить для розв'язуваного завдання, можна визначити і використовувати власну функцію. Така функція має бути створена як М-файл, повертати значення в діапазоні від 0 до 1 і мати число аргументів не більше 16. Оголошення функції (наприклад, *custmf*) виконується в наступній послідовності:

1. Створити файл з ім'ям *cutmf.m.*

2. Вибрати пункт меню *Edit/Add custm MF* у меню редактора функцій належності.

3. У полі *M-File function* діалогового вікна *Add customized membership function*, що з'явилося, ввести ім'я створеного М-файлу (*custmf*).

4. У полі *Parameter List* цього вікна ввести необхідні числові параметри.

5. У полі *MF name* ввести якесь унікальне ім'я функції, наприклад, *polymf.*

6. Завершення визначення функції підтверджується натисканням кнопки <*ok*> (рис. 5.5).

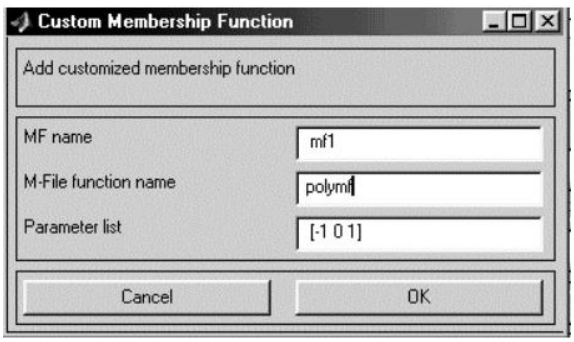


Рис. 5.5. Вікно завдання функції належності користувача

5.1.3. Модель системи керування рівнем води у баку

Модель системи керування рівнем води у баку з нечітким регулятором зображена на рис. 5.6.

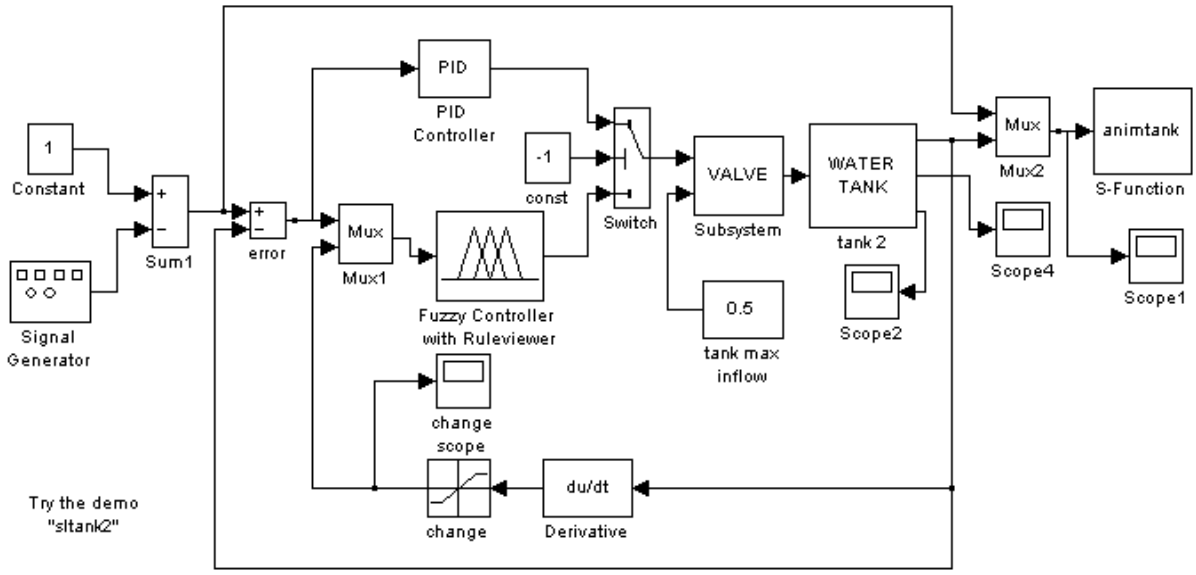


Рис. 5.6. Блок-діаграма моделі системи керування рівнем води у баку з нечітким регулятором

При цьому параметри основних блоків системи керування рівнем води, що настроюються, в баку з нечітким регулятором (рис. 5.6) мають значення, наведені на рис. 5.7.

Запуск анімації здійснюється командою *Start* меню *Simulation*. Після цього з'явиться анімаційне вікно (вікно *Tank Demo* на рис. 5.1). Поточний рівень води показано чорним кольором. Необхідний рівень води відзначений чорною лінією. Анімація запускається автоматично після відкриття вікна.

Нечіткий контролер реалізований системою нечіткого виведення з двома входами: різниця між необхідним та поточним рівнями води та швидкість зміни цієї різниці. Тимчасові діаграми необхідного та поточного рівнів води показані у вікні *Scope1* (рис. 5.8) світлою та темною лініями відповідно. У вікні change *scope* (рис. 5.9) наведено часову діаграму

швидкості зміни різниці необхідного та поточного рівнів води. Ці вікна відкриваються після натискання кнопки миші за піктограмами *Scope1* і *change scope* в *Simulink*-моделі системи "Водяний бак з нечітким контролером".

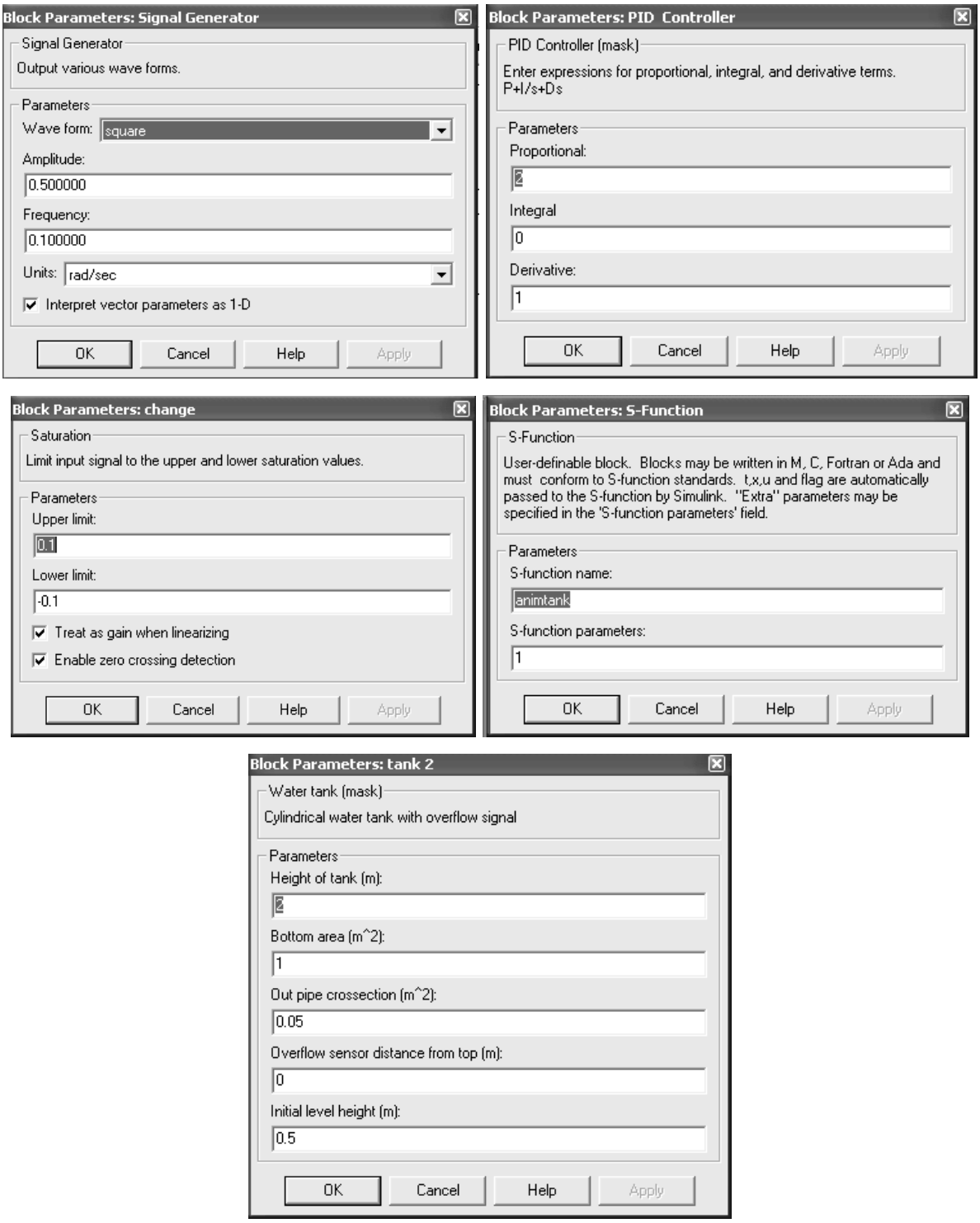


Рис. 5.7. Параметри основних блоків системи управління рівнем води в баку, що настроюються

Користувач може аналогічним чином вивести на екран часові діаграми витрати води (*Scope4*) та сигналу переповнення бака (*Scope2*).

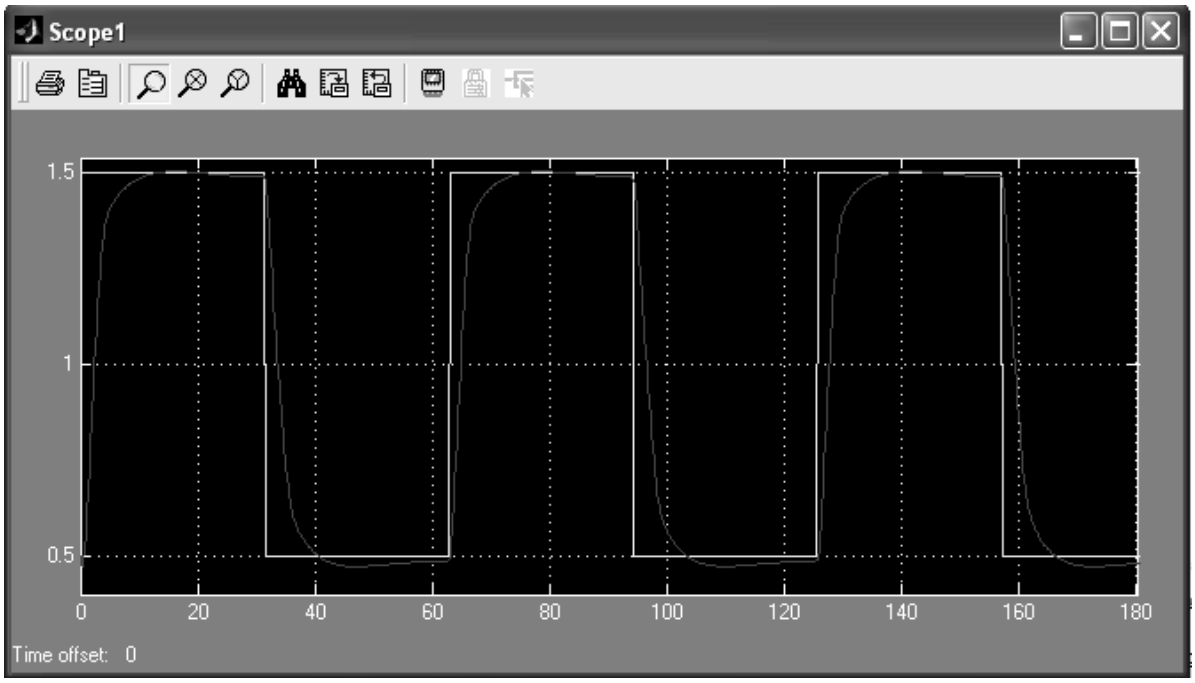


Рис. 5.8. Діаграма необхідного та поточного рівнів води

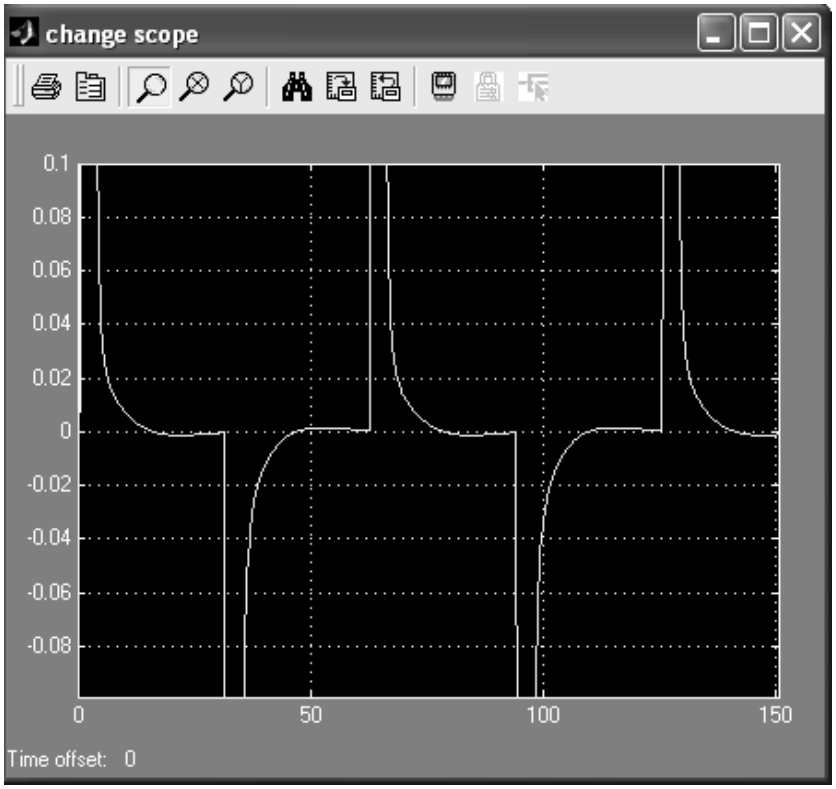


Рис. 5.9. Тимчасова діаграма швидкості зміни різниці необхідного та поточного рівнів води

Як видно з рисунка, перехідний процес у системі має аперіодичну форму і закінчується досить швидко, тобто. якість регулювання слід визнати гарною.

Нечіткий контролер використовує п'ять правил для розрахунку керуючого впливу (рис. 5.10).

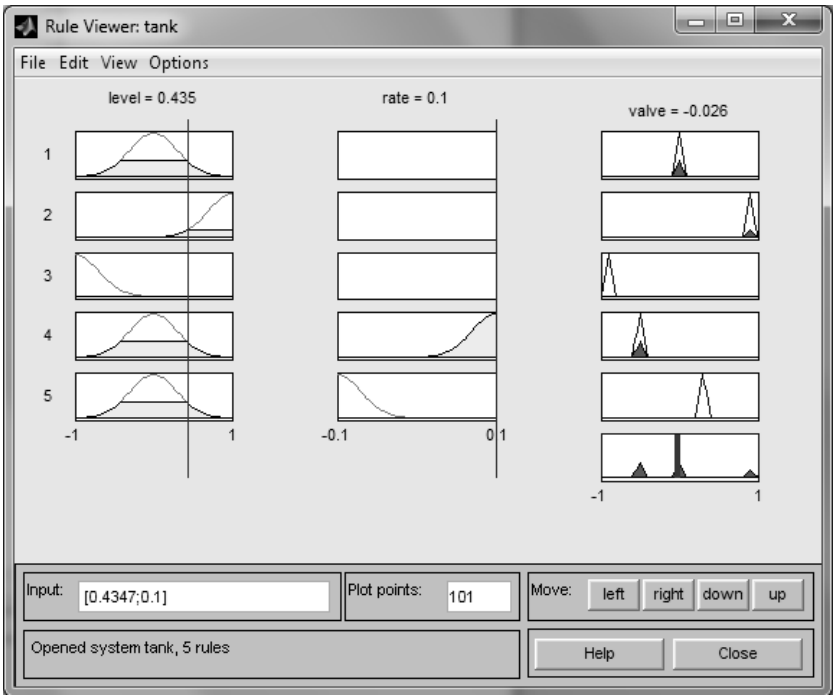


Рис. 5.10. Нечіткі правила для розрахунку керуючого впливу

Для лінгвістичної оцінки вхідної змінної *level* (різниця між необхідним та поточним рівнями води) використовуються три терми з гаусовими функціями належності, для оцінки вхідної змінної *rate* (швидкість зміни різниці між необхідним і поточним рівнями води) – два терми з гаусовими функціями *valve* (зміна положення вентиля) – п'ять термів із трикутними функціями належності. Для перегляду нечіткого контролера в *Simulink*-форматі (рис. 5.11) необхідно клацнути правою кнопкою миші по блоку *Fuzzy Controller* (див. рис. 5.6) і в меню вибрати команду *Look under mask*. Потім у новому графічному вікні *Link*: *sltankrule/Fuzzy Logic Controller with Ruleviewer* (рис. 5.12) клацнути правою кнопкою миші по блоку *Fuzzy Logic Controller* і знову в меню вибрати команду *Look under mask*. Потім у

новому графічному вікні *Link*: *sltankrule/Fuzzy Logic Controller* з *Ruleviewer/Fuzzy Logic Controller* клацнути правою кнопкою миші по блоку *FIS Wizard* і знову в меню вибрати команду *Look under mask.*

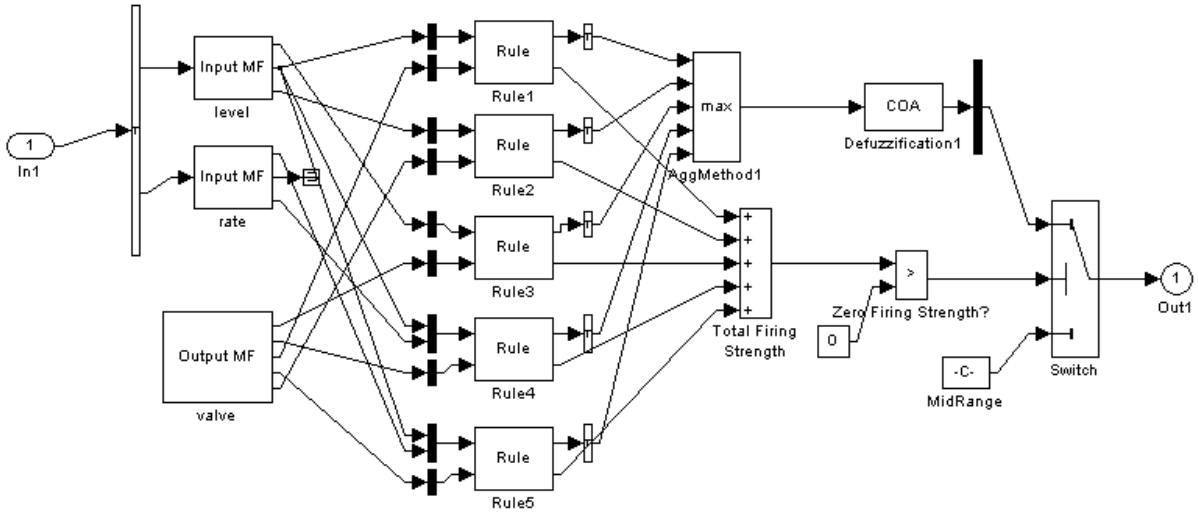


Рис. 5.11. Нечіткий контролер рівня води у баку в *Simulink*-форматі

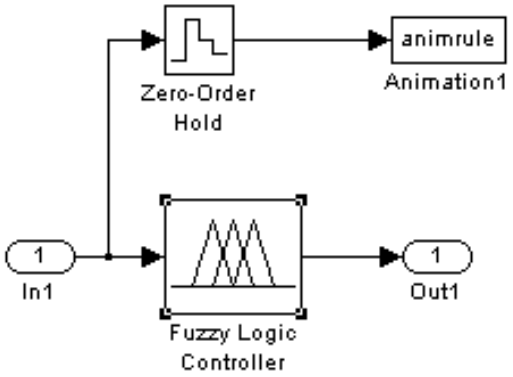


Рис. 5.12. Структура блоку *Fuzzy Logic Controller with Ruleviewer*

Можна спробувати змінити характер функціонування системи, вносячи, наприклад, за допомогою розглянутих програм з графічним інтерфейсом (типу *FIS* редактора тощо) певні зміни до системи, що задається файлом *tank.fis* (змінюючи правила, функції належності, метод приведення до чіткості та т. д.). Зрозуміло, це все краще робити, зупинивши процес моделювання.

Виконавши (в режимі командного рядка) команду *sltankrule*, можна перейти до блок-діаграми тієї ж системи управління, але з нечітким регулятором та переглядачем правил (*Fuzzy Logic Controller with Ruleviewer*).

5.2. Індивідуальні завдання

1. Необхідно вивчити методи побудови нечітких контролерів засобами інструментарію нечіткої логіки та блоків моделювання *Simulink.*

2. Промоделювати в *Simulink* нечіткий контролер, який керує рівнем води у баку.

3. Отримати та навести у звіті словесний опис та графічне відображення правил нечіткого виводу побудованого регулятора.

4. Описати принцип роботи схеми та пояснити отримані результати.

5. Внести зміни до правил нечіткого виводу та пояснити, як внесені зміни вплинуть на роботу всього пристрою.

Наступні параметри є спільними для всіх варіантів (табл. 5.2).

Індивідуальні варіанти завдань для функцій належності змінної *level* наводяться у табл. 5.3. У ній використовуються такі позначення:

$$polymf(x,A)=\frac{\exp(a_0+...+a_n x^n)}{1+\exp(a_0+...+a_n x^n)};\ A=[a_0,a_1,...,a_n].$$

Таблиця 5.2 – Параметри функцій належності для змінних *valve* і *ratre*

Назва вхідних змінних	Назва функції власності	Тип функції належності
valve	*Close_fast*	*Trimf* [–1; –0,9; –0,8]
	Close_low	*Trimf* [–0,6; –0,5; –0,4]
	No_change	*Trimf* [–0,1; 0; 0,1]
	Open_low	*Trimf* [0,2; 0,3; 0,4]
	Open_fast	*Trimf* [0,8; 0,9; 1]
rate	*Negative*	*Gaussmf* [0,03; –0,1]
	None	*Gaussmf* [0,03; 0]
	Positive	*Gaussmf* [0,03; 0,1]

Таблиця 5.3 – Параметри функції належності для змінної *level*

№	*High* *polymf* (x,A)	*Okay* *polymf* (x,A)	*Low* *polymf* (x,A)
1	2	3	4
1	[–3,535; –2,925; 4,293]	[1,604; –1,723; -9,023]	[–2,908; 2,732; 3,256]
2	[–2,591; –2,978; 3,049]	[2,284; 0,697; –8,499]	[–3,724; 3,836; 4,227]
3	[–3,252; –3,353; 3,829]	[2,104; 0,967; –9,367]	[–3,238; 1,977; 3,823]
4	[–4,149; –2,777; 5,561]	[1,638; 0,4058; –8,782]	[–2,559; 2,687; 2,4026]
5	[–3,553; –3,939; 5,266]	[1,443; –0,218; –6,373]	[–3,323; 3,4241; 2,833]
6	[–2,916; –2,883; 2,781]	[1,736; –0,234; –6,048]	[–2,696; 3,001; 2,9796]
7	[–2,241; –2,435; 2,177]	[2,094; 0,2477; –9,948]	[–1,733; 2,522; 2,3055]
8	[–3,9789; –2,825; 4,47]	[1,941; –0,189; –8,253]	[–2,441; 2,131; 2,3544]
9	[–2,503; –2,979; 2,271]	[1,968; 1,24; –10,8512]	[–2,441; 2,131; 2,3543]
10	[–2,526; –2,787; 2,974]	[2,444; 0,9028; –15,06]	[–3,326; 3,185; 4,0171]
11	[–3,691; –3,016; 3,816]	[1,038; 0,8638; –8,122]	[–2,742; 2,483; 3,4183]
12	[–2,952; –3,167; 3,316]	[1,706; –1,023; –6,489]	[–2,764; 2,392; 2,5585]
13	[–3,124; –3,482; 3,972]	[0,941; –0,3953; –5,63]	[–3,134; 2,372; 3,4613]
14	[–3,662; –3,621; 5,183]	[1,869; –0,291; –8,634]	[–2,95; 3,132; 3,24231]
15	[–3,113; –2,659; 2,983]	[2,038; –2,138; –11,24]	[–3,407; 2,763; 3,8331]

6. Оформіть звіт з лабораторної роботи.

Лабораторна робота 6

ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ НА ОСНОВІ НЕЧІТКИХ ЗНАНЬ

Мета лабораторної роботи: отримання та закріплення знань по проектуванню та розробці нечітких моделей інтелектуальних систем.

6.1. Короткі відомості з теорії

Побудова нечіткої моделі інтелектуальної системи керування базується на формальному представленні характеристик об’єкту в термінах лінгвістичних змінних. В системі керування в якості лінгвістичних змінних розглядаються змінні входу і виходу системи. Ціль керування полягає у визначенні значень керуючих змінних (вхідних) змінних, реалізація яких забезпечує бажану поведінку чи бажаний стан об’єкту керування. Таким чином, нечітка модель системи керування може бути описана за допомогою апарату системи нечіткого виводу.

Для керування системами з вбудованими елементами штучного інтелекту в умовах невизначеності необхідно побудувати людино-машинний інтерфейс на основі якісних, вербальних категорій. Такі категорії визначаються в термінах нечітких множин. Для ефективного використання нечітких систем необхідно адекватно визначити нечіткі множини величин, побудувати правила виведення, правила агрегування виходів, здійснити перетворення чітких входів у нечіткі й нечітких виходів у чіткі. Забезпечити адекватність функцій належності нечітких величин можна їх динамічним підстроюванням за допомогою адаптивних методів.

Якість прийняття рішень в основному визначається базою нечітких правил. Такі правила визначаються експертним методом і тому можуть бути суб'єктивними, неповними або суперечливими. Подолання суперечливості

правил і, відповідно, підвищення інтелектуального рівня системи нечіткого логічного виведення досягається поповненням і вдосконаленням бази правил у процесі навчання.

Розглянемо приклад того, як обробляються нечіткі правила виведення в експертній системі, що управляє вентилятором кімнатного кондиціонера. Завдання кондиціонера – підтримувати оптимальну температуру повітря в кімнаті, охолоджуючи його, коли жарко, і нагріваючи, коли холодно. Нехай, змінюючи швидкість обертання вентилятора, що проганяє повітря через охолоджуючий елемент, ми можемо змінювати температуру повітря, тоді алгоритм роботи кондиціонера може бути заданий наступними правилами:

1) ЯКЩО температура повітря в кімнаті *висока*, ТО швидкість обертання вентилятора *висока*;

2) ЯКЩО температура повітря в кімнаті *середня*, ТО швидкість обертання вентилятора *середня*;

3) ЯКЩО температура повітря в кімнаті *низьк*а, ТО швидкість обертання вентилятора *низька*.

Для того щоб система могла обробляти ці правила, треба налаштувати установки приналежності для нечітких підмножин, визначених на значеннях температури (t) і швидкості обертання вентилятора (v). Нехай температура повітря в кімнаті знаходиться в межах від 0 °С до 60 °С – в іншому випадку кондиціонер навряд чи допоможе. Функцію приналежності для нечіткого підмножини «низька», певну на інтервалі зміни температури, можна задати, наприклад, так (рис. 6.1):

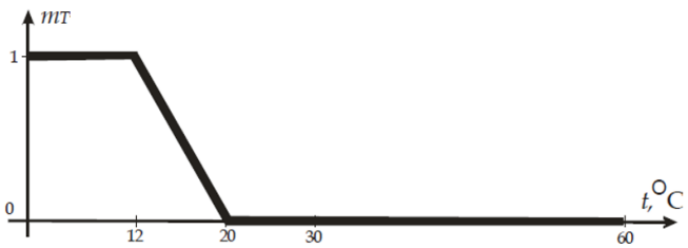


Рис. 6.1 – Нечітка множина «низька», визначена на множині значень температури

Якщо температура менше 12° С, то це – напевно низька температура для кімнати. Температуру вище 20° С ніяк не можна назвати низькою. У проміжку між цими значеннями функція приналежності лінійно убуває - зі збільшенням температури зменшується істинність твердження «температура повітря в кімнаті низька».

Аналітично виражається наступним чином:

$$m_{T\text{низька}}(t) = \begin{cases} 1, \text{при } t \leq 12 \\ \dfrac{20-t}{8}, \text{при } 12 < t < 20 \\ 0, \text{при } t \geq 29 \end{cases}$$

Подібні міркування дозволяють нам налаштувати установки приналежності для решти множин: середня (рис. 6.2) і висока (рис. 6.3).

$$m_{T\text{середня}}(t) = \begin{cases} 0, \text{при } t < 12 \text{ або } t \geq 30 \\ \dfrac{t-12}{8}, \text{при } 12 \leq t < 20 \\ \dfrac{30-t}{10}, \text{при } 20 \leq t < 30 \end{cases}$$

$$m_{T\text{висока}}(t) = \begin{cases} 0, \text{при } t < 20 \\ \dfrac{t-20}{10}, \text{при } 20 \leq t < 30 \\ 1, \text{при } t > 30 \end{cases}$$

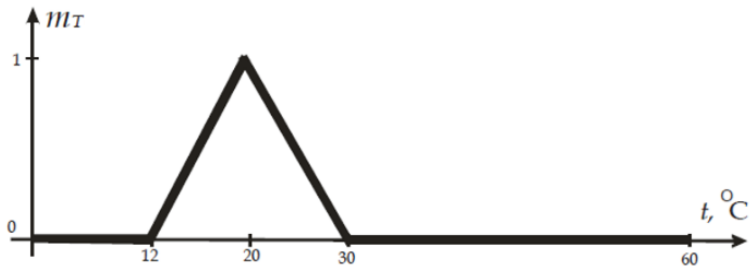


Рис. 6.2 – Нечітка множина «середня», визначена на множині значень температури

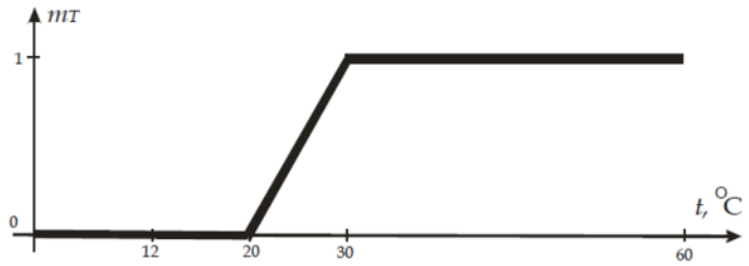
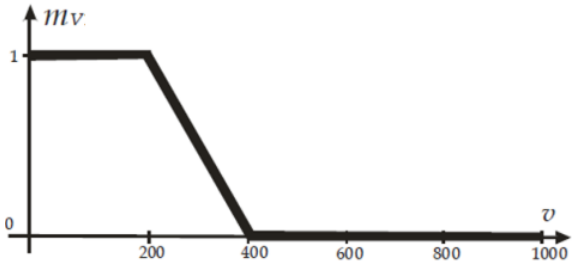


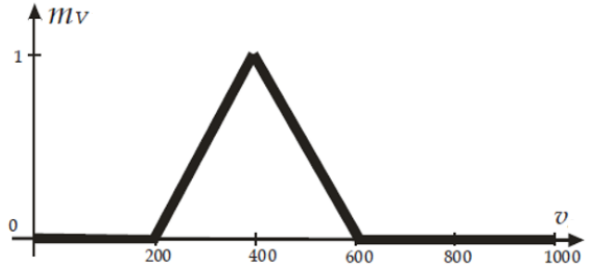
Рис. 6.3 – Нечітка множина «висока», визначена на множині значень температури

Визначимо нечіткі підмножини для швидкості обертання вентилятора. Нехай вона може змінюватися від 0 до 1000 об/хв. Цілком допустимим буде наступний варіант визначення функцій приналежності для нечітких підмножин низька, середня і висока (рис. 6.4 – 6.5).



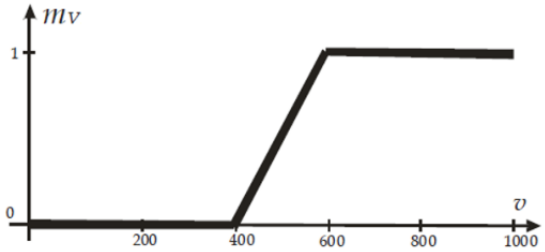
$$m_{V\ \text{низька}}(v)=\begin{cases}1, v<200\\ \dfrac{400-v}{200}, 200\le v\le 400\\ 0, v>400\end{cases}$$

Рис. 6.4 – Нечітка множина «низька», визначена на множині значень швидкості обертання вентилятора кондиціонера



$$m_{V\ \text{середня}}(v)=\begin{cases}0, v<200\ \text{чи}\ v>600\\ \dfrac{v-200}{200}, 200\le v<400\\ \dfrac{600-v}{200}, 400\le v\le 600\end{cases}$$

Рис. 6.5 – Нечітка множина «середня», визначена на множині значень швидкості обертання вентилятора кондиціонера



$$m_{V\ \text{висока}}(v)=\begin{cases}0, v<400\\ \dfrac{v-400}{200}, 400\le v<600\\ 1, v\ge 600\end{cases}$$

Рис. 6.6 – Нечітка множина «висока», визначена на множині значень швидкості обертання вентилятора кондиціонера

Розглянемо тепер, як нечітка експертна система визначає швидкість обертання вентилятора в залежності від температури повітря в кімнаті. Нехай ця температура дорівнює 22°С. Спочатку експертній системі треба визначити істинність лівих частин правил виведення при підстановці в них поточного значення температури. Для цього вона повинна знайти ступінь входження $t = 22°$ С в кожну із зазначених зліва нечітких підмножин. У лівих частинах правил вказані три підмножини, заданих на інтервалі значень температури: висока, низька і середня. Ступінь входження знаходимо, обчислюючи значення функцій приналежності кожного з підмножин від $t = 22°$ С:

$$m_{T\text{висока}}(22) = 0{,}2;$$
$$m_{T\text{середня}}(22) = 0{,}8;$$
$$m_{T\text{низька}}(22) = 0.$$

Значення істинності лівій частині кожного правила використовуються для модифікації нечіткої множини, зазначеного в його правій частині. На рис. 6.7 зображено, як трансформуються множини, які знаходяться в правих частинах правил.

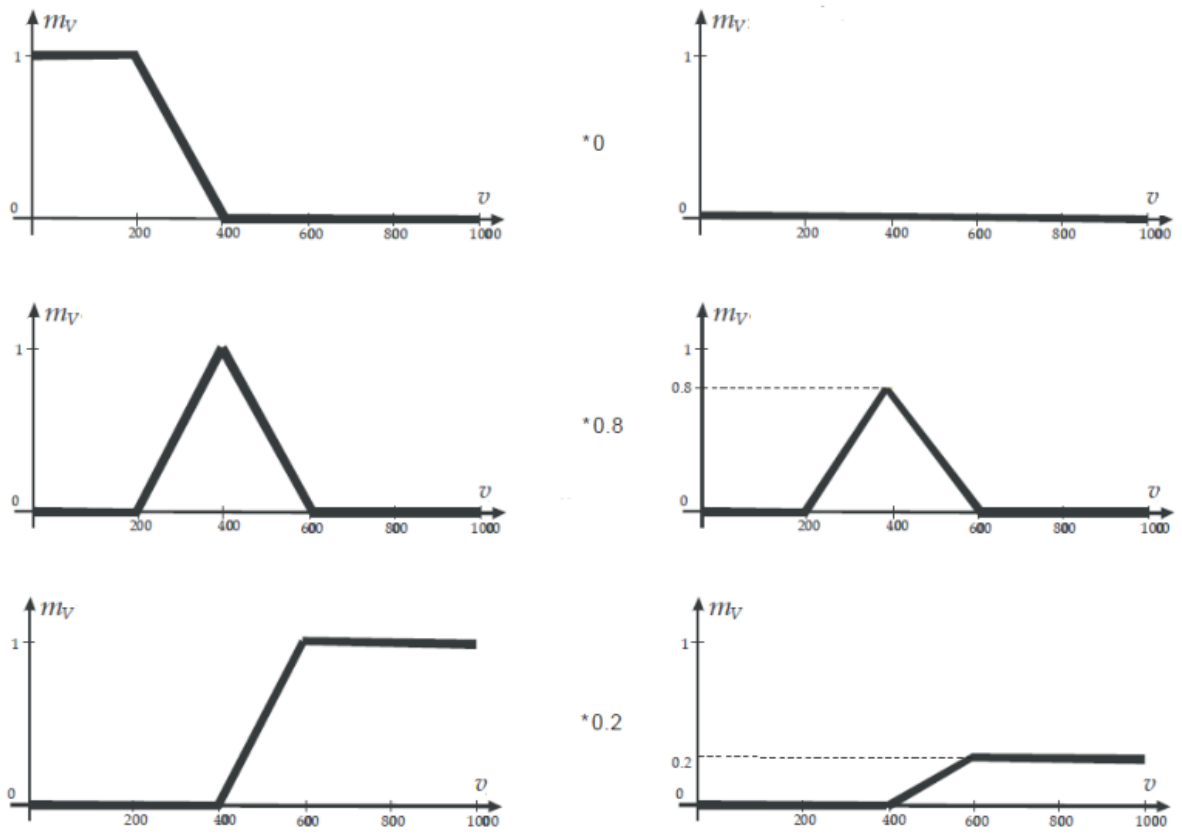


Рис. 6.7 - Модифікація нечітких підмножин, визначених на інтервалі зміни швидкості обертання вентилятора

Далі нечіткої експертної системі необхідно узагальнити результати дії всіх правил виведення, тобто зробити суперпозицію отриманих нечітких множин. Результат об'єднання нечітких множин показаний на рис. 6.8.

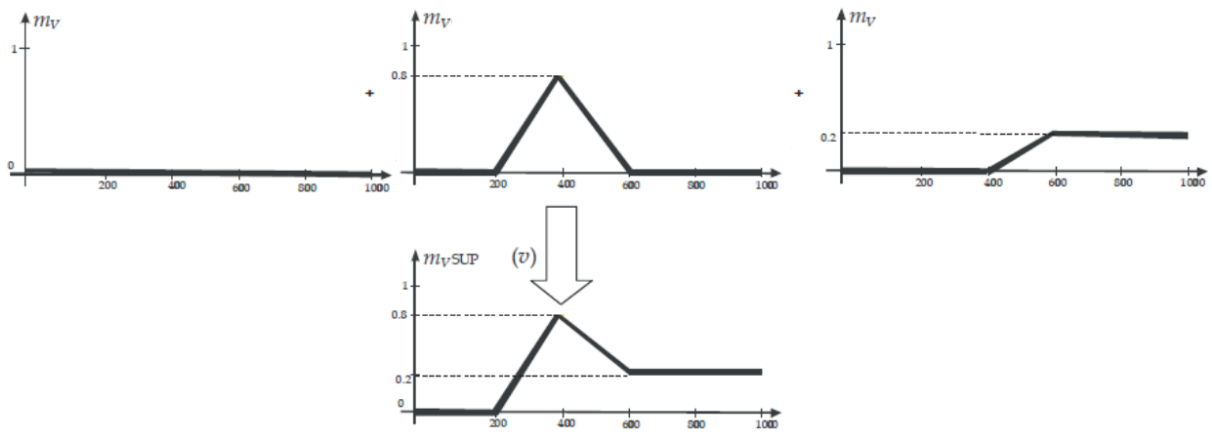


Рис. 6.8 – Результат суперпозиції нечітких множин

Тепер необхідно здійснити перехід від суперпозиції множин до скалярного значення. Скаляризації зробимо методом "центра ваги". Ілюстрація того, як виходить результат, представлена на рис. 6.9.

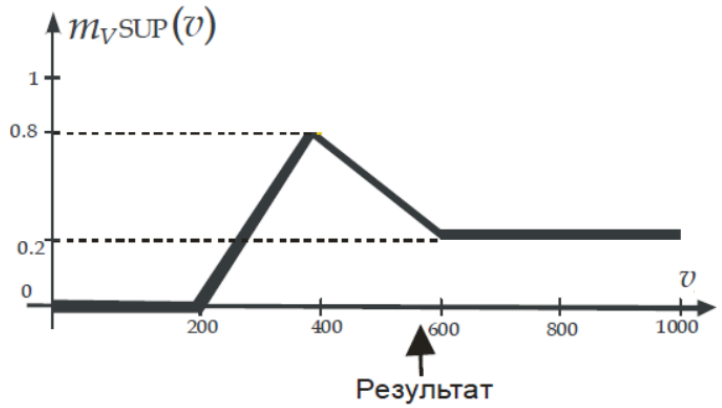


Рис. 6.9 - Отримання скалярного значення швидкості обертання вентилятора методом «центру ваги»

Центр ваги фігури на рис. 6.9 знаходиться в точці $v = 520$. Це і буде значенням швидкості обертання вентилятора, яке видає експертна система при температурі повітря в кімнаті дорівнює 22° С. При інших значеннях температури функція приналежності узагальненого результату виконання всіх правил, зображена на рис. 6.9, буде змінюватися.

6.2. Порядок виконання роботи

1. Завдання нечітких функцій приналежності. У прикладному пакеті Fuzzy logic toolbox програми MATLAB створимо новий проект і поставимо нечіткі функції приналежності для температури як вхідні параметри (рис. 6.10). Задамо нечіткі функції приналежності для швидкості обертання вентилятора як вхідні параметри (рис. 6.11).

2. Задання правил виведення. Правила виведення в створеній комп'ютерній моделі задаються за допомогою вкладки *Edit* → *Rules*. Вікно редактора після завдання правил виведення матиме вигляд, представлений на рис. 6.12.

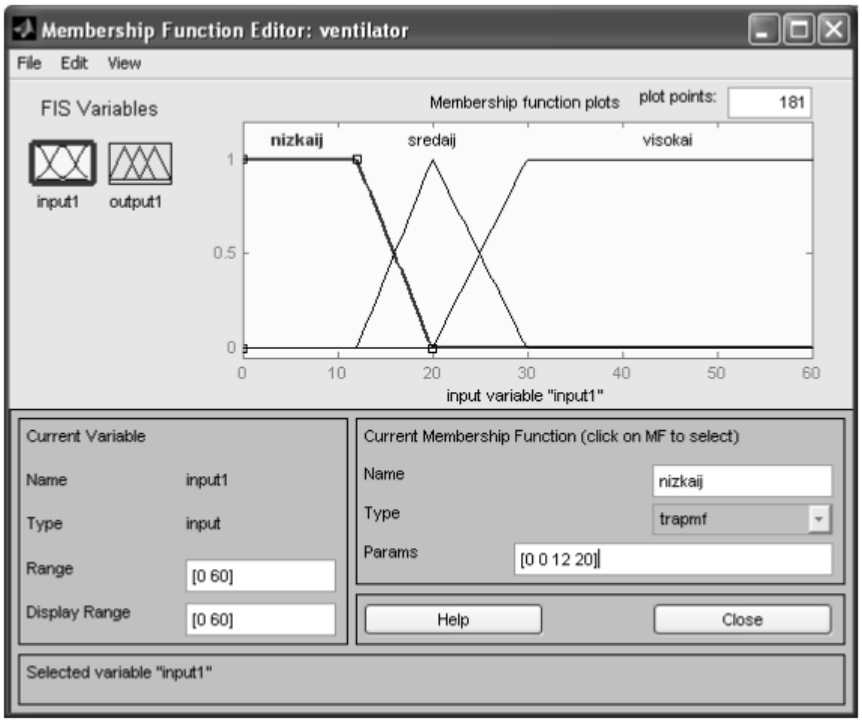


Рис. 6.10 – Вікно завдання нечітких функцій приналежності для значень температури

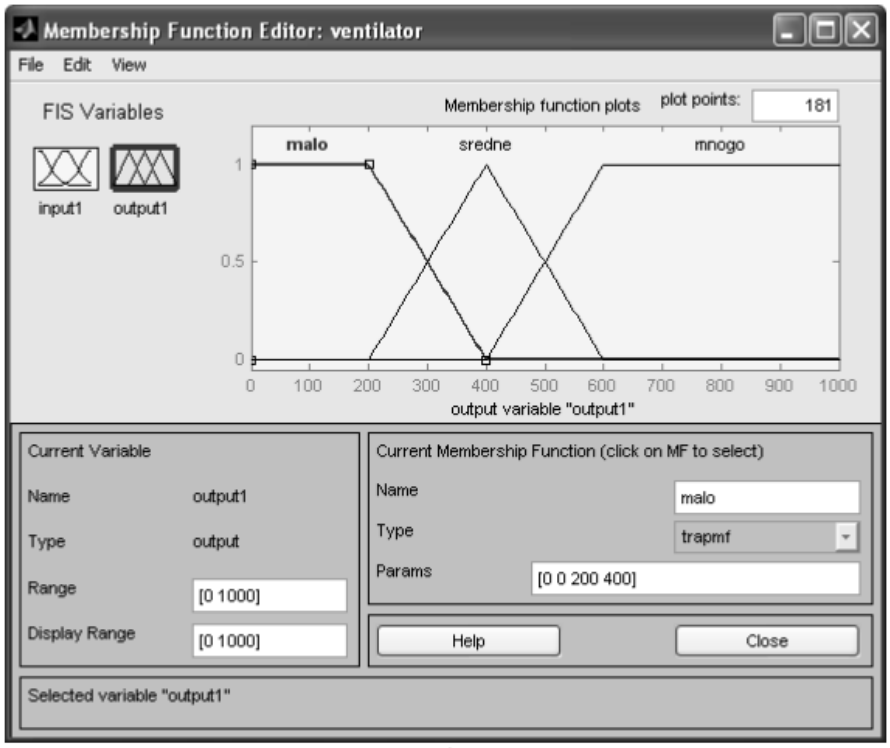


Рис. 6.11 – Вікно задання нечітких функцій приналежності для значень швидкості обертання вентилятора

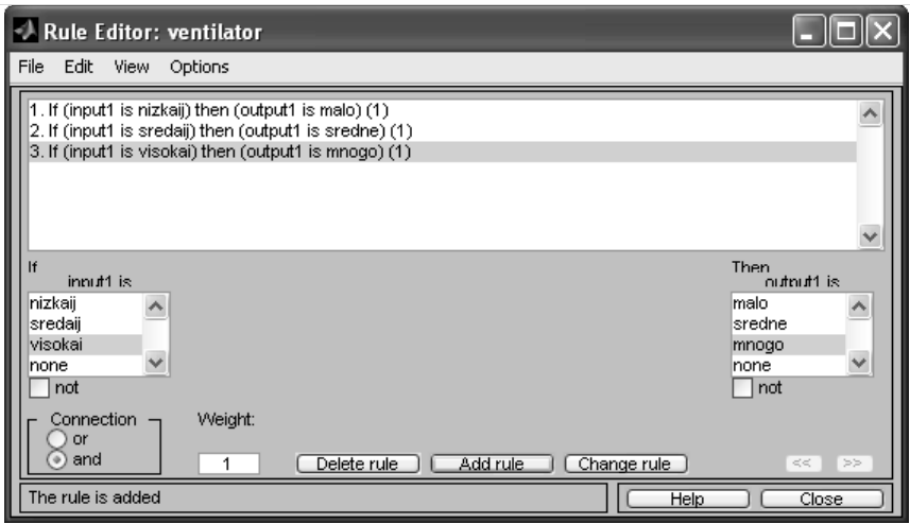


Рис. 6.12 – Задання правил виведення

3. Отримання відгуку системи. Послідовність обробки нечітких знань для конкретного значення температури можна переглянути у вікні перегляду правил *View* → *Rules*. На рис. 6.13 представлена послідовність обробки нечітких знань для температури t = 22 ° С, при цьому вентилятор кондиціонера повинен обертатися зі швидкістю v = 520 об / хв.

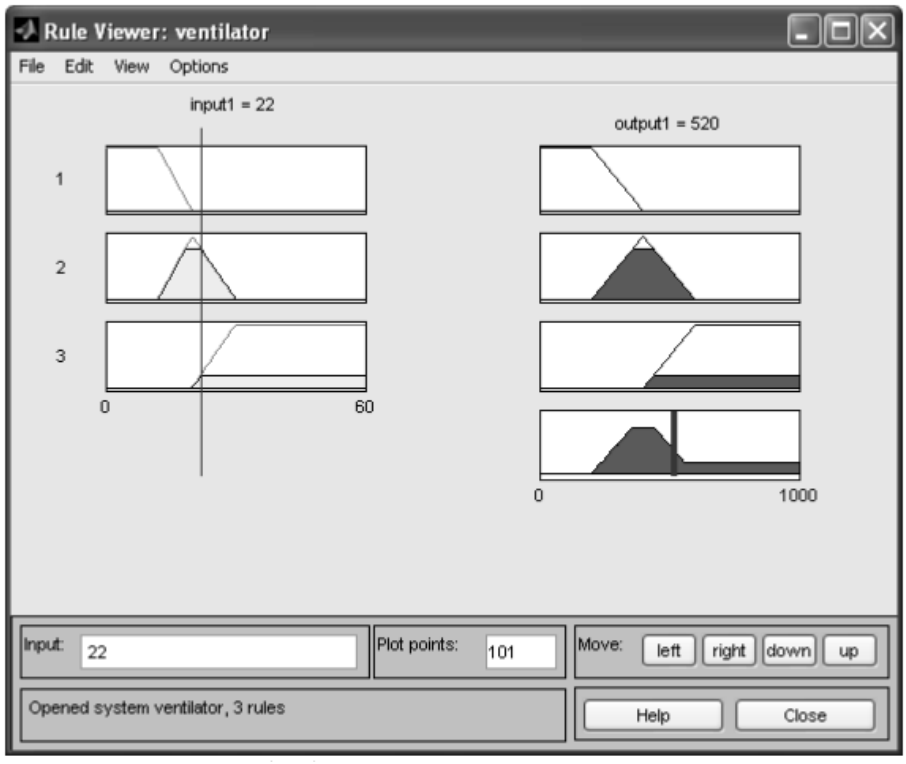


Рис. 6.13 – Обробка нечітких знань в експертній системі

Залежність швидкості обертання кондиціонера від температури для даної моделі можна переглянути за допомогою команди *View* → *Surface* (рис. 6.14).

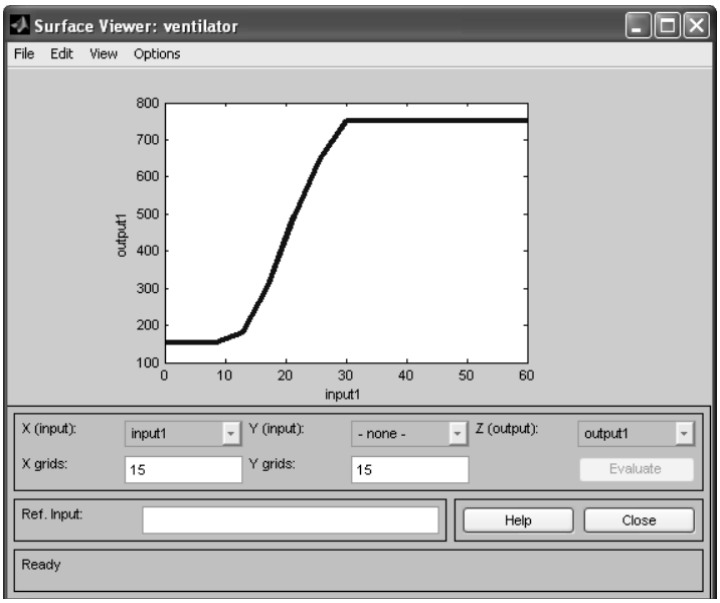


Рис. 6.14 – Відгук системи

6.3. Індивідуальні завдання

1. Відповідно до номера N за списком у журналі групи, оберіть описаний нижче варіант свого завдання за принципом:

Варіант = N, при $N \in [1, 4]$;

Варіант = (N mod 4) +1, при $N > 4$.

2. Оформіть звіт з лабораторної роботи описавши усі стадії розробки.

Варіант 1.

Побудова системи нечіткого керування процесом прийому на роботу у фірму нових співробітників.

Нехай, у фірми є дві вакансії: програміст і системний адміністратор. Тому система нечіткого виведення має дві вихідні лінгвістичні змінні y_1, y_2 з множинами-носіями у вигляді відрізків [0, 1], які інтерпретуються як

імовірність прийняття деякої особи на роботу відповідно на першу чи другу посаду.

Вхідні змінні мають характеризувати особу, яка подала резюме про себе в фірми з метою влаштуватися на роботу. Наприклад, x_1 - досвід роботи в сфері комп'ютерних технологій, x_2 - освіта, x_3 - ділові якості спілкування.

Менеджер по кадрам даної фірми розробив такі прості правила прийому на роботу:

1. Якщо досвід роботи високий і професійна освіта висока, то ймовірність прийняття на посаду програміста висока

2. Якщо досвід роботи високий і ділові якості гарні, то ймовірність прийняття на посаду системного адміністратора висока

3. Якщо досвід роботи і освіта високі, а ділові якості гарні, то ймовірність прийняття на роботу даної особи висока на обидві посади

4. Якщо досвід роботи невеликий, але освіта та ділові якості високі, тоді можливо прийняти на роботу таку особу в якості системного адміністратора

5. Якщо особа не має освіти і досвіду роботи, тоді прийняття її на роботу малоймовірне на обидві посади

6. Якщо професійна освіта висока, але досвід роботи і ділові якості невеликі тоді можна прийняти цю особу на посаду програміста і малоймовірно її прийняти на посаду системного адміністратора

7. Якщо освіти у особи немає, тоді малоймовірне її прийняття на роботу

8. Якщо особа не має досвіду роботи і її ділові якості не високі, тоді малоймовірне її прийняття на роботу на обидві посади

9. Якщо освіта особи висока, досвід роботи невеликий, то можливо прийняття її на роботу на посаду програміста

10. Якщо особа не має освіті і ділових якостей, то прийняття її на роботу малоймовірне на обидві посади

11. Якщо особа не має освіти, досвід і її ділові якості невисокі, тоді малоймовірне її прийняття на роботу на обидві посади

12. Якщо ділові якості особи невеликі, освіта професійна і досвід роботи високі, тоді малоймовірно її прийняти на посаду системного адміністратора і можливо її прийняти на посаду програміста.

Для розробки системи слід виконати наступні кроки.

Крок 1. Визначити вхідні і вихідні змінні. Очевидно, що для СНВ у якості вхідних змінних потрібно взяти

1) досвід роботи x_1 з множиною-носієм від 0 до 30 років;
2) освіту особи x_2 з множиною-носієм, наприклад, [0,10] ;
3) ділові якості x_3 з множиною-носієм [0,1].

Далі слід задати терми цих змінних. Наприклад, для вхідної змінної досвіду роботи можуть бути задані наступні терми: «немає», «невеликий», «високий»

Крок 2. Фузифікація вхідних змінних. На цьому кроці слід задати функції належності для всіх термів вхідних змінних, а в якості області визначення – їх множини-носії

Крок 3. Задати функції належності термів вихідних змінних (ймовірність прийняття на посаду)

Крок 4. Ввести правила у базу правил.

Крок 5. Використання моделі. Для цього розглянути приклад роботи системи керування при різних значеннях вхідної змінної. Для цього слід відкрити вікно правил *Rules View* → і переглянути можливі значення вихідної змінної прийняття рішення про прийом на роботу в залежності від зміни значень вхідних змінних.

Варіант 2.

Побудова нечіткої моделі керування кранами гарячої і холодної води при прийнятті душу

При прийнятті душу на вхід змішувача подається холодна та гаряча вода по відповідним трубопроводам. Для комфортного прийняття душа, користувач задає на виході бажану комфортну теплу температуру і бажаний напір води. Оскільки, під час прийняття душу спостерігається нерівномірне використання води і температура на виході змішувача буде весь час коливатися, виникає необхідність ручного керування кранами відкриття гарячої чи холодної води. Задача полягає у створенні системи нечіткого виведення, яка б дозволила автоматизувати цей 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 будь-якого крану направо – це збільшити потік води відповідної температури. Тоді досвід прийняття душа дозволяє сформувати декілька евристичних правил.

1. Якщо вода гаряча і її напір сильний, тоді необхідно повернути кран гарячої води на середній кут вліво, а кран холодної води на середній кут вправо

2. Якщо вода гаряча і її напір не дуже сильний, слід повернути кран холодної води на середній кут вправо

3. Якщо вода не дуже гаряча і її напір сильний, тоді необхідно повернути кран гарячої води на невеликий кут вліво

4. Якщо вода не дуже гаряча і її напір слабий, тоді слід повернути крани гарячої і холодної води на невеликий кут вправо

5. Якщо вода тепла і її напір не дуже сильний, тоді слід залишити кран змішувача в своєму положенні

6. Якщо вода прохолодна і її напір сильний, тоді необхідно повернути кран гарячої води на середній кут вправо, а кран холодної води на середній кут вліво

7. Якщо вода прохолодна і її напір не дуже сильний, тоді слід повернути кран гарячої води на середній кут вправо, а кран холодної води на невеликий кут вліво

8. Якщо вода холодна і її напір слабий, тоді слід повернути кран гарячої води на великий кут вправо

9. Якщо вода холодна і її напір сильний, тоді слід повернути кран гарячої води на середній кут вліво, а кран гарячої води на середній кут вправо

10. Якщо вода тепла і її напір сильний, тоді слід повернути крани гарячої і холодної води на невеликий кут вліво .

11. Якщо вода тепла і її напір слабий, тоді слід повернути крани гарячої і холодної води на невеликий кут вправо.

Для розробки системи слід виконати наступні кроки.

Крок 1. Визначити вхідні і вихідні змінні. Очевидно, що для СНВ у якості вхідних змінних потрібно взяти температуру води на виході змішувача і її напір. У якості вихідних змінної слід взяти кути повороту кранів гарячої і холодної води.

Далі слід задати терми цих змінних. Наприклад, для вхідної змінної температури – «вода гаряча», «вода не дуже гаряча», «вода тепла», «вода прохолодна», «вода холодна»

Крок 2. Фузифікація вхідних змінних. На цьому кроці слід задати функції належності для всіх термів вхідних змінних, а в якості області визначення – інтервал можливої температури води і інтервал кількісної оцінки напору. Наприклад, відповідно, в градусах Цельсія від 0 до 100 і внормований одиницях – від 0(напору немає) до 1(напір дуже сильний).

Крок 3. Задати функції належності термів вихідної змінної (кута повороту крана гарячої води) з інтервалом в області визначення від -90 до 90 гр.

Крок 4. Ввести правила у базу правил.

Крок 5. Використання моделі. Для цього розглянути приклад роботи системи керування при різних значеннях вхідної змінної.

Варіант 3.

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 системи нечіткого виведення, яка б змогла автоматизувати роботу кондиціонера при коливанні температури приміщення через різні зовнішні дестабілізуючі фактори.

Досвід використання побутових кондиціонерів показує деяку інертність в процесі нагріву чи охолодження повітря. Наприклад, після включення режиму «холод», відбувається нагнітання холодного повітря, через що температура в приміщенні поступово спадає.

При цьому, при виключенні цього режиму, температура все рівно деякий час продовжує знижуватися. Аналогічна картина спостерігається при включенні режиму «тепло». Щоб врахувати цю властивість, потрібно задати як вхідну змінну не тільки температуру приміщення, але і швидкість її зміни. В такому випадку, досвід показує адекватність наступних правил керування кондиціонеру:

1. Якщо температура повітря дуже тепла і швидкість зміни температури позитивна, то потрібно включити режим «холод», повернувши регулятор кондиціонеру на великий кут вліво.

2. Якщо температура повітря дуже тепла, а швидкість зміни температури негативна, тоді необхідно включити режим «холод», повернувши регулятор кондиціонеру на невеликий кут вліво.

3. Якщо температура повітря тепла, а швидкість зміни температури позитивна, тоді потрібно включити режим «холод», повернувши регулятор кондиціонеру на великий кут вліво.

4. Якщо температура повітря тепла, а швидкість зміни температури негативна, тоді потрібно включити режим «холод», повернувши регулятор кондиціонеру слід вимкнути.

5. Якщо температура повітря дуже холодна, а швидкість зміни температури негативна, тоді потрібно включити режим «тепло», повернувши регулятор кондиціонеру на великий кут вправо.

6. Якщо температура повітря дуже холодна, а швидкість зміни температури позитивна, тоді потрібно включити режим «тепло», повернувши регулятор кондиціонеру на невеликий кут вправо.

7. Якщо температура повітря холодна, а швидкість зміни температури негативна, тоді потрібно включити режим «тепло», повернувши регулятор кондиціонеру на великий кут вліво.

8. Якщо температура повітря холодна, а швидкість зміни температури позитивна, тоді потрібно виключити кондиціонер.

9. Якщо температура повітря дуже тепла, а швидкість зміни температури дорівнює 0, тоді потрібно включити режим «холод», повернувши регулятор кондиціонеру на великий кут вліво.

10. Якщо температура повітря тепла, а швидкість зміни температури дорівнює 0, тоді потрібно включити режим «холод», повернувши регулятор кондиціонеру на невеликий кут вліво.

11. Якщо температура повітря дуже холодна, а швидкість зміни температури дорівнює 0, тоді потрібно включити режим «тепло», повернувши регулятор кондиціонеру на великий кут вправо.

12. Якщо температура повітря холодна, а швидкість зміни температури дорівнює 0, тоді потрібно включити режим «тепло», повернувши регулятор кондиціонеру на невеликий кут вправо.

13. Якщо температура повітря в нормі, а швидкість зміни температури позитивна, тоді потрібно включити режим «холод», повернувши регулятор кондиціонеру на невеликий кут вліво.

14. Якщо температура повітря в нормі, а швидкість зміни температури негативна, тоді потрібно включити режим «тепло», повернувши регулятор кондиціонеру на невеликий кут вправо.

15. Якщо температура повітря в нормі, а швидкість зміни температури дорівнює 0, тоді потрібно виключити кондиціонер.

Тому, для розробки системи слід виконати наступні кроки.

Крок 1. Визначити вхідні і вихідні змінні. Очевидно, що для СНВ у якості вхідних змінних потрібно взяти температуру повітря в приміщенні та швидкість її зміни. У якості вихідної змінної слід взяти кут повороту регулятора кондиціонеру. Далі слід задати терми цих змінних. Наприклад, для 1 вхідної змінної – «повітря дуже холодне», «повітря холодне», «повітря в нормі», «повітря тепле», «повітря дуже тепле»

Крок 2. Фузифікація вхідних змінних. На цьому кроці слід задати функції приналежності для всіх термів вхідної змінної, а в якості області визначення – інтервал можливої температури повітря та її швидкості. Наприклад, в градусах Цельсія від - 10 до 40.

Крок 3. Задати функції приналежності термів вихідної змінної (кута повороту регулятора кондиціонеру) з інтервалом в області визначення від – 90 до 90 гр.

Крок 4. Ввести правила у базу правил.

Крок 5. Використання моделі. Для цього розглянути приклад роботи системи керування при різних значеннях вхідної змінної.

Варіант 4.

Нечітка модель керування контейнерним краном.

Контейнерні крани використовуються при виконанні вантажних робіт в портах. Крани тросом підіймають контейнер з вантажем і далі ідуть на спеціальних рейсах до місця призначення і там розвантажуються. Коли контейнер піднімається до крану і кран починає рухатися по рейсам, вантаж

починає розхитувати і відхилятися від свого строго вертикального положення під краном. Проблема полягає в тому, що допоки контейнер хитається, він не може бути опущений на платформу цілі, в якості яких використовуються деякі транспортні засоби.

Аналіз дій крановиків-операторів, які виконують дії по управлінню краном, показує, що вони в своїй роботи використовують наступні евристичні правила:

Почати роботу, потрібно, коли контейнер вже завантажено грумом і виконувати наступні правила:

1. Якщо контейнер знаходиться на горизонталі завантаження чи цілі (наприклад, це позиція 0), тоді слід підняти контейнер на рівень руху (наприклад, позиція 1).

2. Починати рух потрібно зі середньою потужністю

3. Якщо рух вже почався, кабіна знаходиться далеко від цілі і контейнер несильно хитається, тоді потрібно задати потужність руху велику

4. Якщо рух вже почався, кабіна знаходиться далеко від цілі і контейнер сильно хитається, тоді потрібно задати потужність руху невелику

5. Якщо кабіна знаходиться близько від цілі, слід зменшити швидкість руху до малої

6. Якщо контейнер знаходиться дуже близько до цілі, тоді слід виключити потужність двигуна

7. Коли контейнер знаходиться прямо над позицією цілі, тоді слід зупинити двигун.

8. Якщо контейнер знаходиться над ціллю, але хитання великі, тоді слід не опускати контейнер на ціль і чекати

9. Якщо контейнер знаходиться над ціллю, він не хитається, тоді слід опускати його

10. Якщо контейнер зустрівся з ціллю, тоді слід зупинити його опускання

Тому, для розробки системи слід виконати наступні кроки.

Крок 1. Визначити вхідні і вихідні змінні. Очевидно, що для СНВ у якості вхідних змінних потрібно взяти відстань до цілі, кут хитання контейнеру від строго вертикальної позиції під кабіною крану і вертикальну відстань контейнеру від цілі. У якості вихідних змінних слід взяти потужність двигуна крана при русі рейсами і швидкість опускання, чи піднімання крану.

Далі слід задати терми цих змінних. Наприклад, для 1 вхідної змінної – «відстань далека», «відстань середня», «відстань близька», «позиція над ціллю», для 2 вхідної змінної – «кут=0», «кут хитання малий», «кут хитання великий», для 3 вхідної змінної – «контейнер в позиції 0», «контейнер в позиції 1», для 1 вихідної змінної – «потужність середня», «потужність велика», «потужність мала», «потужність нульова», для 2 вихідної змінної – «піднімати контейнер», «опускати контейнер», «не змінювати вертикальну позицію контейнера»

Крок 2. Фузифікація вхідних змінних. На цьому кроці слід задати функції приналежності для всіх термів вхідної змінної, а в якості області визначення – інтервал можливих значень.

Крок 3. Задати функції належності термів вихідних змінних.

Крок 4. Ввести правила у базу правил.

Крок 5. Використання моделі. Для цього розглянути приклад роботи системи керування при різних значеннях вхідної змінної.

Лабораторна робота 7

ЗАСТОСУВАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ ПРИ ВИЗНАЧЕННІ ЕКСТРЕМУМІВ ФУНКЦІЙ

Мета лабораторної роботи: отримання та закріплення знань, формування практичних навичок щодо визначення екстремумів функцій за допомогою генетичних алгоритмів у пакеті MATLAB.

7.1. Короткі відомості з теорії

Генетичні алгоритми (ГА) відносяться до евристичних алгоритмів пошуку, які використовуються для вирішення задач оптимізації та моделювання шляхом випадкового підбору, комбінування та варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію. ГА є різновидом еволюційних обчислень, за допомогою яких вирішуються оптимізаційні завдання з використанням методів природної еволюції, таких як спадкування, мутації, відбір та кросовер. Відмінною особливістю ГА є використання оператора "схрещування", який здійснює операцію рекомбінації рішень-кандидатів, роль якої аналогічна ролі схрещування у живій природі.

Типовими застосуваннями ГА є такі: пошук екстремумів функцій; оптимізація запитів у базах даних; завдання на графах (завдання комівояжера, розмальовка, знаходження паросполучень); налаштування та навчання штучної нейронної мережі; завдання компонування; складання розкладів; ігрові стратегії; теорія наближень та інше.

Основною ідеєю ГА є організація "боротьби за існування" та "природний відбір" серед пробних (зразкових) розв'язків задачі. Оскільки ГА використовують біологічні аналогії, то термінологія, що застосовується, нагадує біологічну.

Як відомо, еволюційна теорія стверджує, що життя на нашій планеті виникло спочатку лише в її найпростіших формах – у вигляді одноклітинних організмів. Ці форми поступово ускладнювалися, пристосовуючись до навколишнього середовища і породжуючи нові види, і тільки через багато мільйонів років з'являлися перші тварини і люди. Можна сміливо сказати, що кожен біологічний вид з часом поліпшує свої якості для того, щоб найбільш ефективно справлятися з найважливішими завданнями виживання, самозахисту, розмноження тощо. Так виникло захисне забарвлення у багатьох риб і комах, панцир у черепахи, отрута у скорпіона та багато інших корисних пристосувань.

За допомогою еволюції природа постійно оптимізує все живе, знаходячи часом найнеординарніші рішення. З першого погляду неясно, за рахунок чого відбувається цей прогрес, проте є наукове пояснення. Дати це пояснення можна, ґрунтуючись лише на двох біологічних механізмах – природному відборі та генетичному наслідуванні.

Ключову роль еволюційної теорії грає природний відбір. Його суть полягає в тому, що найбільш пристосовані особини краще виживають і приносять більше нащадків, ніж менш пристосовані. Зауважимо, що сам собою природний відбір ще забезпечує розвитку біологічного виду. Дійсно, якщо припустити, що всі нащадки народжуються приблизно однаковими, то різні покоління відрізнятимуться тільки за чисельністю, але не за пристосованістю. Тому дуже важливо вивчити, як відбувається успадкування, тобто як властивості нащадка залежать від властивостей батьків.

Основний закон наслідування інтуїтивно зрозумілий кожному – він у тому, що нащадки схожі на батьків. Зокрема, нащадки більш пристосованих батьків будуть, найімовірніше, одними з найпристосованіших у своєму поколінні. Щоб зрозуміти, на чому заснована ця схожість, нам потрібно трохи заглибитись у будову клітини тварини – у світ генів та хромосом.

У клітинах будь-якої тварини є набір хромосом, що несуть інформацію про цю тварину. Основна частина хромосоми – нитка ДНК (молекула дезоксирибонуклеїнової кислоти), яка складається з чотирьох видів спеціальних сполук – нуклеотидів, що йдуть у певній послідовності. Нуклеотиди позначаються буквами *A, T, C* і *G*, і саме порядок їхнього слідування кодує всі генетичні властивості даного організму. Говорячи точніше, ДНК визначає, які хімічні реакції відбуватимуться у цій клітині, як розвиватиметься і які функції виконувати.

Ген – це відрізок ланцюга ДНК, що відповідає за певну властивість особини, наприклад, за колір очей, тип волосся, колір шкіри і т.д. Вся сукупність генетичних ознак людини кодується за допомогою приблизно 60 тис. генів, сумарна довжина яких становить понад 90 млн. нуклеотидів.

Розрізняють два види клітин: статеві та соматичні. У кожній соматичній клітині людини міститься 46 хромосом. Ці 46 хромосом – насправді 23 пари, причому в кожній парі одна з хромосом отримана від батька, а друга – від матері. Парні хромосоми відповідають за ті самі ознаки – наприклад, батьківська хромосома може містити ген чорного кольору очей, а парна їй материнська – ген блакитноокості. Існують певні закони, що керують участю тих або інших генів у розвитку особини. Зокрема, у нашому прикладі нащадок буде чорнооким, оскільки ген блакитних очей є "слабким" (рецесивним) і пригнічується геном будь-якого іншого кольору.

У статевих клітинах хромосом лише 23, і вони непарні. При заплідненні відбувається злиття чоловічої та жіночої клітин та утворюється клітина зародка, що містить якраз 46 хромосом. Які властивості нащадок отримає від батька, а які – від матері?

Це залежить від того, які саме клітини брали участь у заплідненні. Справа в тому, що процес вироблення статевих клітин в організмі схильний до випадковостей, завдяки яким нащадки все ж таки багато в чому відрізняються від своїх батьків. При цьому парні хромосоми соматичної клітини зближуються впритул, потім їх нитки ДНК розриваються в

декількох випадкових місцях і обмінюються хромосоми своїми частинами (рис. 7.1).

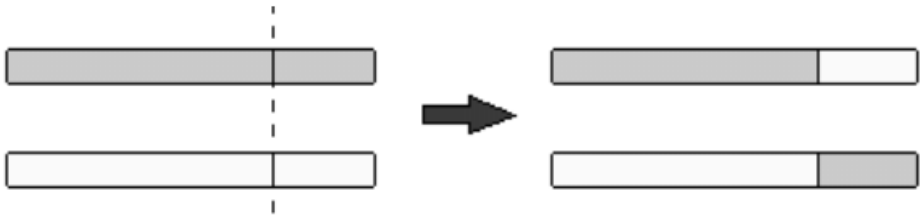


Рис. 7.1. Умовна схема схрещування (кросовера)

Цей процес забезпечує появу нових варіантів хромосом і зветься "кросовер". Кожна з хромосом, що знову з'явилися, виявиться потім усередині однієї з статевих клітин, і її генетична інформація може реалізуватися в нащадках даної особини.

Другий важливий фактор, що впливає на спадковість, – це мутації, що виражаються у зміні деяких ділянок ДНК. Мутації також випадкові і можуть бути викликані різними зовнішніми факторами, такими як, на приклад, радіоактивне опромінення. Якщо мутація відбулася в клітині, то змінений ген може передатися нащадку і проявитися у вигляді спадкової хвороби або інших нових властивостях нащадка. Вважається, що саме мутації є причиною появи нових біологічних видів, а кросовер визначає вже мінливість усередині виду (наприклад, генетичні різницю між людьми).

Як було зазначено вище, еволюція – це процес постійної оптимізації біологічних видів. Природний відбір гарантує, що найбільш пристосовані особини дадуть досить велике потомство, а завдяки генетичній спадковості ми можемо бути впевнені, що частина цього потомства не тільки збереже високу пристосованість батьків, але й матиме деякі нові властивості. Якщо ці нові властивості виявляться корисними, то з великою ймовірністю вони перейдуть і у наступне покоління. Таким чином, відбувається накопичення корисних якостей та поступове підвищення пристосованості біологічного виду в цілому. Знаючи, як вирішується завдання оптимізації видів у природі,

ми тепер застосуємо схожий метод для вирішення різних реальних технічних завдань.

Введемо позначення та наведемо кілька класичних прикладів. Як правило, у задачі оптимізації ми можемо керувати декількома параметрами (позначимо їх значення через x_1, x_2, ..., x_n, а нашою метою є максимізація (АБО мінімізація) деякої функції $f(x_1, x_2, ..., x_n)$, залежить від цих параметрів. Функція f називається цільовою функцією. Наприклад, якщо потрібно максимізувати цільову функцію "дохід компанії", то керованими параметрами будуть кількість співробітників компанії, обсяги виробництва, витрати на рекламу, ціни на кінцеві продукти і т.д. Важливо відзначити, що ці параметри пов'язані між собою – зокрема, при зменшенні кількості співробітників швидше за все впаде обсяг виробництва.

Для таких завдань було розроблено кілька методів рішень. В разі, якщо цільова функція досить гладка і має лише один локальний максимум (унімодальна), оптимальне рішення можна отримати методом градієнтного спуску. Ідея цього у тому, що оптимальне рішення виходить ітераціями. Береться випадкова початкова точка, а потім у циклі відбувається зсув цієї точки на малий крок, причому крок робиться в тому напрямку, в якому цільова функція зростає найшвидше. Недоліком градієнтного алгоритму є дуже високі вимоги до функції – практично унімодальність зустрічається вкрай рідко, а неунімодальної функції градієнтний метод часто призводить до неоптимальної відповіді. Аналогічні проблеми виникають із застосуванням інших математичних методів. У багатьох важливих завданнях параметри можуть набувати лише певних значень, причому у всіх інших точках цільова функція не визначена. Звичайно, в цьому випадку не може бути мови про її гладкість і потрібні принципово інші підходи.

Уявімо собі штучний світ, населений безліччю істот (особин), причому кожна істота – це певне рішення нашого завдання. Вважатимемо особину тим більше пристосованою, чим краще відповідне рішення (що більше значення цільової функції воно дає). Тоді завдання максимізації

цільової функції зводиться до пошуку найбільш пристосованої істоти. Звичайно, ми не можемо поселити у наш віртуальний світ усі істоти одразу, бо їх дуже багато. Натомість ми розглядатимемо багато поколінь, які змінюють одне одного. Тепер, якщо ми зуміємо ввести в дію природний відбір і генетичне успадкування, то отриманий світ підкорятиметься законам еволюції. Зауважимо, що відповідно до нашого визначення пристосованості метою цієї штучної еволюції буде саме створення найкращих рішень. Вочевидь, еволюція – нескінченний процес, під час якого пристосованість особин поступово підвищується. Примусово зупинивши цей процес через досить довгий час після його початку і вибравши найбільш пристосовану особину в поточному поколінні, ми можемо отримати відповідь близьку до оптимального, але можемо таку відповідь не отримати, оскільки це випадковий пошук. Така, стисло, ідея генетичного алгоритму. Перейдемо тепер до точних визначень та опишемо роботу генетичного алгоритму детальніше.

Для того щоб говорити про генетичне спадкування, потрібно забезпечити наші істоти хромосомами. У генетичному алгоритмі **хромосома** – це деякий числовий вектор, що відповідає параметру, що підбирається, а набір хромосом даної особини визначає рішення задачі. Які саме вектори слід розглядати у конкретній задачі, вирішує сам користувач. Кожна з позицій вектора хромосоми називається **геном**.

Визначимо тепер поняття, що відповідають мутації та кросоверу в генетичному алгоритмі.

Мутація – це перетворення хромосоми, що випадково змінює одну або кілька її позицій (генів). Найбільш поширений вид мутацій – випадкова зміна лише одного з генів хромосоми.

Кросовер (у літературі з ГА також використовується назва кросинговер і схрещування) – це операція, при якій з двох хромосом породжується одна або кілька нових хромосом. У найпростішому випадку кросовер у генетичному алгоритмі реалізується так само, як і в біології (див.

рис. 7.1). При цьому хромосоми розрізаються у випадковій точці та обмінюються частинами між собою. Наприклад, якщо хромосоми (1, 2, 3, 4, 5) і (0, 0, 0, 0, 0) розрізати між третім і четвертим генами та обміняти їх частини, то вийдуть нащадки (1, 2, 3, 0, 0) і (0, 0, 0, 4, 5).

Спочатку генерується початкова населення особин (індивідуумів), тобто деякий набір розв'язків задачі. Як правило, це робиться випадковим чином. Потім ми маємо змоделювати розмноження усередині цієї популяції. І тому випадково відбираються кілька пар індивідуумів, виробляється схрещування між хромосомами кожної парі, а отримані нові хромосоми поміщаються у популяцію нового покоління. У генетичному алгоритмі зберігається основний принцип природного відбору – чим пристосованіший індивід (що більше відповідне йому значення цільової функції), то з більшою ймовірністю він братиме участь у схрещуванні. Тепер моделюються мутації – у кількох випадково обраних особин нового покоління змінюються деякі гени. Потім стара популяція частково чи повністю знищується, і ми переходимо до розгляду наступного покоління. Населення наступного покоління у більшості реалізацій ГА містить стільки ж особин, скільки початкова, але з відбору пристосованість у ній середньому вище. Тепер описані процеси відбору, схрещування та мутації повторюються вже для цієї популяції тощо.

У кожному наступному поколінні ми будемо спостерігати виникнення нових рішень нашого завдання. Серед них будуть як погані, так і добрі, але завдяки відбору кількість добрих рішень зростатиме. Зауважимо, що у природі немає абсолютних гарантій, і навіть найпристосований тигр може загинути від пострілу з рушниці, не залишивши потомства. Імітуючи еволюцію на комп'ютері, ми можемо уникати подібних небажаних подій і завжди зберігати життя кращому з індивідуумів поточного покоління – така методика називається "стратегією елітизму".

7.2. Реалізація генетичних алгоритмів за допомогою консолі MATLAB

У MATLAB можливість використання ГА для обчислень реалізована за допомогою вкладки *Genetic Algorithm and Direct Sarch Toolbox*, що розширює можливості пакету *Optimization Toolbox* генетичними алгоритмами. Такі алгоритми найчастіше використовуються у разі, коли шукана цільова функція є розривною, суттєво нелінійною, стохастичною і не має похідних, або ці похідні є недостатньо визначеними. Працювати з генетичними алгоритмами тепер можна у двох тулбоксах.

Власне генетичні алгоритми відносяться до розділу *Genetic Algorithm* і викликаються з командного терміну за допомогою *gatool* або *ga*. Генетичні алгоритми та їх комбінації з іншими оптимізаційними методами можна знайти у розділі *Direct Search Toolbox*. Для цього у командному рядку необхідно набрати *psearchtool*. Розглянемо перший варіант роботи із ГА. Існують 4 основні функції для роботи з алгоритмом:

ga – функція знаходження мінімуму цільової функції;

gaoptimget – повертає параметри використовуваного генетичного алгоритму;

gaoptimset – встановлює параметри генетичного алгоритму;

gatool – відкриває вікно *Genetic Algorithm Tool*.

Для того, щоб застосувати ГА до поставленої функції мети, необхідно, в першу чергу, записати її в М-файл і зберегти в поточній папці.

7.2.1. Функція ga

Функція *ga* викликається в командному рядку згідно з наведеним нижче синтаксисом:

[*x fval*] = *ga*(*@fitnessfun, nvars, options*),

де *fitnessfun* – ім’я М-файлу, що містить поставлену цільову функцію, *nvars* – число незалежних змінних у цільовій функції, *options* – структура, що містить параметри використовуваного ГА. Якщо параметри не змінювати, їх значення візьмуться за замовчуванням. Результати обчислень збережуться у змінних: *fval* – остаточне значення цільової функції, *x* – точка, у якій досягнуто оптимальне значення.

Існують інші варіанти виклику функції *ga*:

```
x = ga(fitnessfun, nvars)
x = ga(fitnessfun, nvars, options)
x = ga(problem)
[x, fval]=ga(...)
[x, fval, reason]=ga(...)
[x, fval, reason, output]=ga(...)
[x, fval, reason, output, population]=ga(...)
[x, fval, reason, output, population, scores]=ga(...)
```

Опис *x* = *ga*(*fitnessfun*, *nvars*) застосовується для вирішення оптимізаційної задачі, *fitnessfun* – мінімізована цільова функція та *nvars* – довжина вектора рішень *x*, відповідного найкращої особини.

x = *ga*(*fitnessfun*, *nvars*, *options*) застосовується для вирішення оптимізаційної задачі, використовуючи параметри (*options*) алгоритму.

x = *ga*(*problem*) знаходить мінімум функції, структура якої описується трьома полями:

f*itnessfcn* – цільова функція;

nvars – число незалежних змінних цільової функції;

options – параметри структури ГА, що задаються функцією *gaoptimset*.

[*x*, *fval*] = *ga*(...) повертає *fval*, значення цільової функції по *x*.

[*x*, *fval*, *reason*] = *ga*(...) повертає *reason* – рядок, що містить параметри зупинення алгоритму.

[*x, fval, reason, output*] = *ga*(...) повертає *output* сукупність відомостей про кожне покоління та іншу інформацію про реалізацію алгоритму. Структура *output* складається з наступних полів:

Randstate або (*randnstate*) – початковий стан популяції, згенерований випадковим чином (відмінність функцій полягає у різних висновках);

generations – кількість поколінь, що обчислюються;

funccount – кількість обчислень функції;

message - Параметри зупинки алгоритму. Це повідомлення виводить кілька аргументів завершення алгоритму.

[*x, fval, reason, output, population*] = *ga*(...) повертає матрицю популяції, рядки якої відповідають особин кінцевої популяції.

[*x, fval, reason, output, population, scores*] = *ga*(...) повертає розрахунки фінальної популяції.

Зауваження 1

Для всіх оптимізаційних завдань популяція має бути представлена у вигляді речових чисел. Функція *ga* не працює для функцій із комплексними змінними. Для вирішення завдань, що включають комплексні числа, потрібно записати цільову функцію у вигляді допустимого речового вектора, відокремивши речові та комплексні частини. Така цільова функція може виглядати так:

```
[x fval, reason] = ga(@rastriginsFcn, 10)
x =
    Columns 1 through 7
    0.9977 0.9598 0.0085 0.0097 −0.0274 −0.0173 0.965
    Columns 8 through 10
    −0.0021 −0.0210 0.0065
fval = 3.7456
reason =
    generations
```

7.2.2. Функція gaoptimset

Для налаштування генетичного алгоритму використовується функція *gaoptimset*. Вона дозволяє побудувати ГА, комбінуючи оператори за бажанням користувача. Синтаксис цієї функції виглядає так:

```
options = gaoptimset
gaoptimset
options = gaoptimset('param1', value1, 'param2',
value2,...)
options = gaoptimset(oldopts,'param1',value1,...)
options = gaoptimset(oldopts,newopts)
```

Опис функції:

options = gaoptimset (тут аргументи не вводяться) за допомогою даної конструкції задається структура ГА, відмінна від структури ГА за замовчуванням.

gaoptimset не вимагає введення або виведення аргументів. В результаті сформує список параметрів та їх дійсних значень.

options = gaoptimset('*param*1',*value*1,'*param*2',*value*2,...) генерує структуру з багатьма параметрами їх значень. Для кількох неспеціальних параметрів можна використовувати значення за замовчуванням.

options = gaoptimset(*oldopts*,'*param*1',*value*1,...) створює копію *oldopts*, модифіковану вибраними спеціальними параметрами та їх значеннями.

options = gaoptimset(*oldopts*, *newopts*) поєднує параметри існуючої структури *oldopts*, з параметрами нових структурновоптів.

Деякі параметри з ненульовими значеннями в *newopts* можуть бути замінені або можуть бути присвоєні старим параметрам в *oldopts*.

7.3. Реалізація генетичних алгоритмів за допомогою діалогового вікна MATLAB

Інший найбільш наочний спосіб роботи з ГА полягає у виклик вікна за допомогою функції *gatool*. Діалогове вікно представлено на рис. 7.2.

Як видно з малюнка, вікно супроводжується довідкою по всіх компонентах, і робота користувача полягає у вигляді встановлення параметрів та натискання кнопки "*start*". Результат буде тим самим, що й у разі послідовного застосування функцій *gaoptimset* і *ga*. У розділі *plots* можна вибрати змінні, зміна яких відображатиметься графічно.

У діалоговому вікні ГА передбачено векторизацію функцій, завдяки чому обчислення відбуваються помітно швидше. Сенс цього методу полягає в тому, що як параметри функції виступають вектори, тоді для поточної популяції цільова функція буде викликатися лише один раз, обчислюючи пристосованість всіх особин. Наприклад, розглянемо функцію $f(x_1, x_2) = x_{21} - 2x_1x_2 + 6x_1 + x_2^2 - 6x_2$.

Запишемо для неї М-файл, використовуючи наступний код:

```
» z =x(:,1)./ 2 - 2*x(:,1).*x(:,2) + 6*x(:,1) +
x(:,2)./ 2-6*x(:,2);
```

Тут *x*(:, 1) є вектором, а ./ та .* – операції поелементного відповідно до зведення в ступінь та множення.

У вікні тулбоксу у списку *Vectorize option* слід встановити значення *On*.

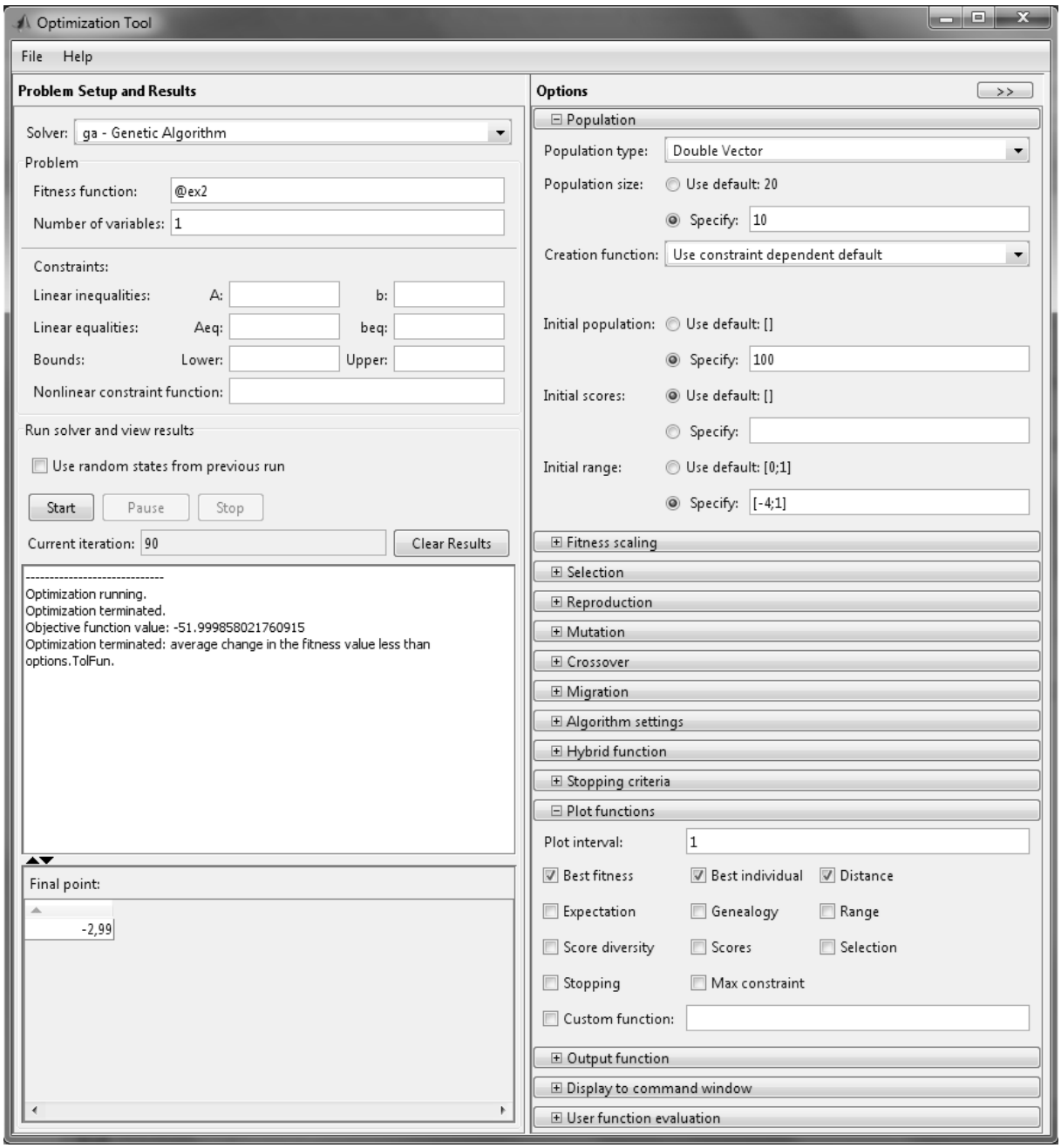


Рис. 7.2. Діалогове вікно по генетичних алгоритмах

Переконатися у ефективності векторизації можна з прикладу функції Растригіна. В командному рядку введемо такий вираз:

```
» tic;
» ga(@rastriginsfcn,20);
» toc;
```

В результаті можна дізнатися час, витрачений на обчислення:

```
Elapsed time is 4.366073 seconds.
```

Зробимо те саме, але використовуючи векторизацію функції Растригіна:

```
» options=gaoptimset('Vectorize','on');
» tic;
» ga(@rastriginsfcn,20,options);
» toc;
```

Дізнаємося час, витрачений на векторизацію:

```
Elapsed time is 0.581 seconds.
```

Як видно, метод векторизації працює набагато швидше.

7.4. Застосування генетичних алгоритмів для пошуку мінімуму функції

Знайдемо мінімум функції одного змінного вигляду:

$$f(x) = 8x - 16 - 12\sqrt[3]{(x+4)^2}\,.$$

Напишемо М-файл для цієї функції та збережемо його в поточній папці під ім'ям *ex2.m*.

```
function y = ex2(x)
y=8*x-16-12*((x+4)^(2/3));
```

Викличемо вікно тулбоксу за допомогою *gatool*.

У полі *fitness function* введемо ім'я цільової функції *@ex*2.

Встановимо значення параметрів ГА: кількість особин у популяції дорівнює 10, кількість поколінь дорівнює 100 (у вікні критерію зупинення алгоритму), початковий відрізок [-4; 1]. У розділі *plots* встановимо прапорці для *best fitness, best individual, distance* та натиснемо кнопку *start*.

У результаті завершення процесу у вікні *final point* з'явиться значення змінної *x*, відповідне мінімуму функції, а вікні *status and result* можна побачити знайдене мінімальне значення цільової функції.

Для цього завдання результати вийшли такі:

– мінімум функції досягається у точці $x = –2.99$;

– значення функції $f(–2.99) = –51.999858021760915$.

На рис. 7.3 відображається зміна значення цільової функції, найкраща особина та відстані між особами у поколіннях. З даних, одержаних на рис. 7.3 видно, що, починаючи приблизно з 80 популяції, алгоритм зійшовся до рішення. Особи стають однаковими (відстань по Хеммінгу дорівнює 0) в останніх 18 поколіннях.

ГА потрібно запускати кілька разів, а потім обирати оптимальне рішення. Це з тим, що початкова популяція формується з допомогою генератора випадкових чисел. Впевнитись у правильності рішення можна, побудувавши графік функції (рис. 7.4).

Те саме можна було б отримати, використовуючи функції *gaoptimset* і *ga*. Щоб переглянути М-файл, оберіть у меню *File* вікна *Genetic Algoritm Tool* команду *Generate M-file*, збережіть файл під іншим ім'ям та перегляньте код.

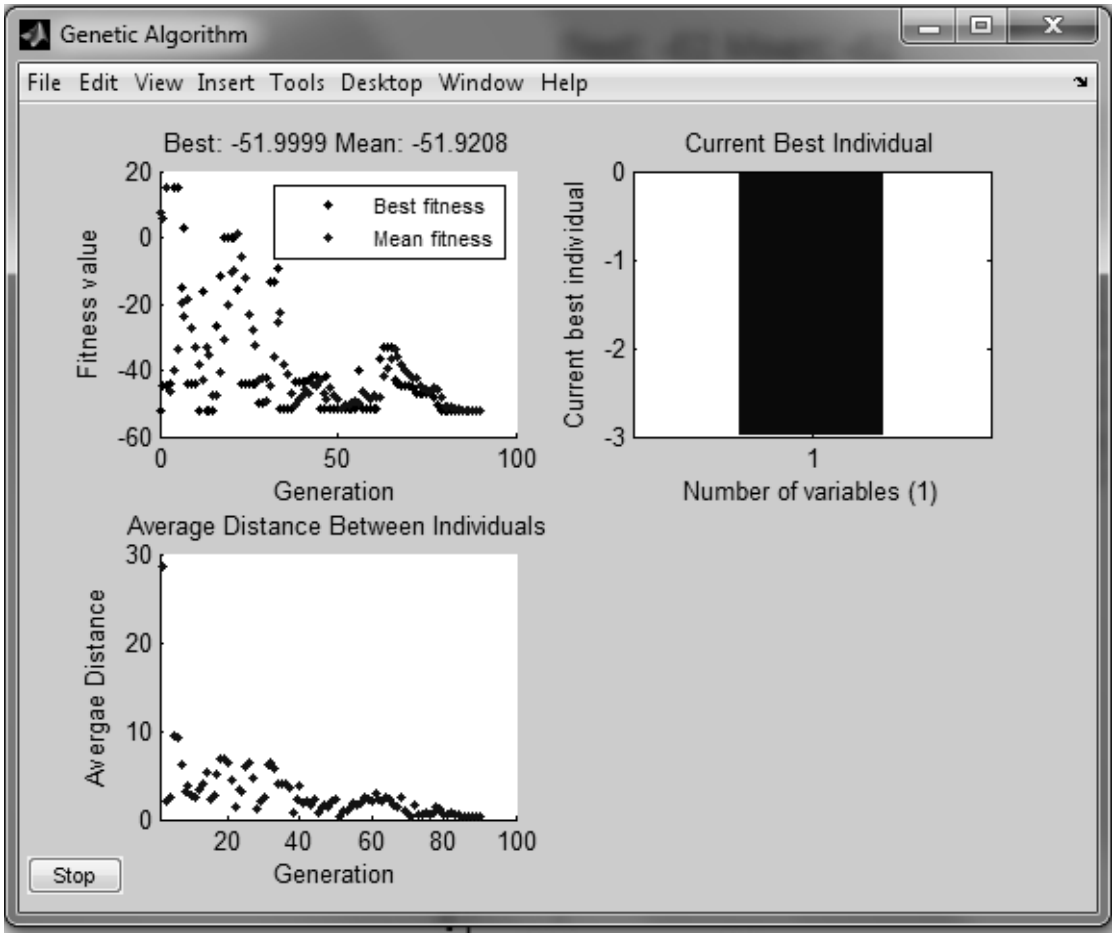


Рис. 7.3. Графічний аналіз рішення

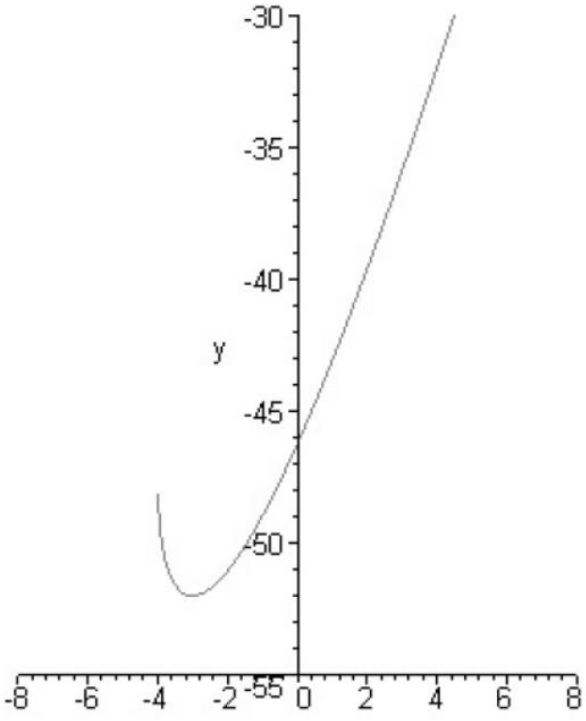


Рис. 7.4. Графік функції

Для цього завдання отримали:

```
function [x,fval,exitflag,output,population,score] =
ex2q(nvars,PopInitRange_Data,PopulationSize_Data,InitialPo
pulation_Data)
% This is an auto generated M-file from Optimization Tool.
% Start with the default options
options = gaoptimset;
% Modify options setting
options = gaoptimset(options,'PopInitRange',
PopInitRange_Data);
options = gaoptimset(options,'PopulationSize',
PopulationSize_Data);
options = gaoptimset(options,'InitialPopulation',
InitialPopulation_Data);
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', {  @gaplotbestf
@gaplotbestindiv @gaplotdistance });
[x,fval,exitflag,output,population,score] = ...
ga(@ex2,nvars,[],[],[],[],[],[],[],options);
```

7.5. Застосування генетичних алгоритмів для пошуку максимуму функції

Знайдемо максимум функції двох змінних видів:

$$z(x,y) = \exp(-x^2 - y^2) + \sin(x + y).$$

У діалоговому вікні ГА можна вирішити лише завдання мінімізації. Для знаходження максимуму функції $f(x)$ слід мінімізувати $-f(x)$. Це тим, що точка мінімуму $-f(x)$ є деякою точкою $f(x)$, у якій досягається максимум.

Напишемо М-файл для функції $z(x)=-f(x)$ і збережемо його в поточній папці під ім'ям *ex*13.*m*:

```
function z = ex13(x)
z=-(exp(-x(1)^2-x(2)^2)+sin(x(1)+x(2)));
```

Викличемо діалогове вікно за допомогою *gatool*.

У полі *fitnes function* введемо ім'я цільової функції *@ex13*. Встановимо значення параметрів ГА: кількість змінних, що дорівнює 2, кількість особин у популяції, що дорівнює 10, кількість поколінь, що дорівнює 100 (у вікні критерію зупинки алгоритму), початковий відрізок, рівний [-1; 3]. Для побудови графіків у розділі *plots* встановимо прапорці для *best fitness*, *best individual*, *distance* та натиснемо кнопку *start*.

В результаті завершення процесу у вікні *final point* з'явиться значення змінної x, що відповідає мінімуму функції, а у вікні *status and result* можна побачити знайдене мінімальне значення цільової функції $z(x)$.

Для цього завдання результати вийшли такі:

– максимум функції досягається у точці $x = 0.46419$, $y = 0.42406$;

– значення функції $f(0.46419; 0.42406) = 1.449$.

Також можна сконструювати інші ГА, моделюючи оператори (вибір батьківських пар, кросинговер, мутація, міграція, відбір особин у нову популяцію, критерії завершення алгоритму) та параметри алгоритму.

Подальше вивчення оптимізації за допомогою ГА можна продовжити, розглядаючи *Direct Search Toolbox*. Для виклику вікна можна в командному рядку просто прописати *psearchtool*.

7.6. Індивідуальні завдання

1. За номером у журналі групи з табл. 7.1 вибрати функцію для визначення екстремумів за допомогою ГА у пакеті MATLAB.

Таблиця 7.1 – Функції визначення екстремумів з допомогою генетичного алгоритму

Номер у журналі групи	Назва та вид функції для визначення екстремумів
1, 8, 15	Синусоїдальна функція: $f(x) = x_1 \cdot \sin 4x_1 + 1{,}1 \cdot x_2 \cdot \sin 2x_2$.
2, 9, 16	Функція Растригіна: $f(x) = 20 + x_1^2 + x_2^2 - 10 \cdot (\cos 2\pi x_1 + \cos 2\pi x_2)$.
3, 10, 17	Модифікована функція Растригіну: $f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos 2\pi x_i)$, для $n = 2$.
4, 11, 18	Функція Швефеля: $f(x) = 418{,}9829n - \sum_{i=1}^{n} (x_i \cdot \sin \sqrt{\lvert x_i \rvert})$, для $n = 2$.
5, 12, 19	Функція Екклі ($n = 2$): $f(x) = 20 + e - 20 \cdot \exp\left(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i \right)$.
6, 13, 20	Функція Михалевича: $f(x) = -\sum_{i=1}^{n} (\sin x_i \cdot \sin \frac{i x_i^2}{\pi})$, для $n = 2$.
7, 14, 21	Функція Негневицького: $f(x) = (x_1 - x_1^3 - x_2^3)\exp(-x_1^2 - x_2^2) - (1 - x_1)^2 \exp(-x_1^2 - (x_2 + 1)^2)$.

2. Здійснити пошук мінімуму та максимуму обраної функції за допомогою командного рядка та графічного інтерфейсу ГА пакету MATLAB, використовуючи при цьому функції *gaoptimset* та *ga*. При цьому обґрунтувати вибір усіх параметрів та критеріїв зупинення роботи ГА.

3. Прорахувати час, витрачений пошук екстремумів функції, при реалізації ГА з допомогою командного рядка пакета MATLAB.

4. Побудувати графік функції у тому, щоб переконатися у правильності знайдених з допомогою ГА екстремумів.

5. Згенерувати коди М-файлів для кожного завдання пошуку екстремумів функції за допомогою ГА.

6. Пояснити різницю між отриманими результатами при багаторазових запусках ГА під час пошуку оптимального рішення.

7. Оформіть звіт з лабораторної роботи.

Лабораторна робота 8

ЗАСТОСУВАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ У ЗАВДАННЯХ ОПТИМІЗАЦІЇ

Мета лабораторної роботи: отримання та закріплення знань, формування практичних навичок застосування генетичних алгоритмів до різних завдань оптимізації.

8.1. Короткі відомості з теорії

8.1.1. Застосування генетичних алгоритмів до завдання оптимізації обчислювальної мережі

Об'єктом генетичної оптимізації буде розгалужена обчислювальна мережа (ОМ). Основою мережі є опорне кільце, або бекбон, що є послідовно з'єднаними концентраторами. До концентраторів підключаються хости, які, своєю чергою, можуть бути маршрутизаторами для підмереж другого рівня. Кожен хост може бути підключений до будь-якого маршрутизатора. Завданням оптимізації є знаходження такої схеми підключення хостів, коли трафік на бекбоні буде мінімальним. Розрахунок трафіку на бекбоні будується на основі статистичних даних, накопичених за досить великий проміжок часу, що дозволяє оцінити обсяг трафіку між двома хостами.

Для оптимізації трафіку необхідно вирішити такі інженерні завдання*:*

– розробити та реалізувати систему зняття статистики трафіку на бекбоні, що складається з концентраторів;

– розробити систему подання вхідних даних для генетичного алгоритму (ГА);

– розробити та реалізувати генетичний алгоритм.

Модель обчислювальної мережі організації

Опишемо поточний стан ОМ з погляду оптимізації. Вважаємо, що на деякому етапі розвитку ОМ прийнято рішення про прокладання центральної магістралі, закупівлю та встановлення комунікаційного обладнання. Перелік робочих станцій (вузлів) ОМ відомий, причому для вузлів вирішено завдання розміщення, тобто відомі координати вузлів. Розміщення вузлів за умов установи підпорядковується структурі підрозділів, тому технічну оптимізацію доцільно обмежити рамками розміщення комунікаційного устаткування. Комунікаційне обладнання представлене маршрутизаторами, концентраторами та/або комутаторами. Кожен вузол (робоча станція) може бути підключений тільки до одного комутатора/концентратора (ком/конц), таким чином кожен ком/конц визначає сегмент ОМ. Розбиття вузлів по ком/конц вважаємо відомим.

Нехай у ОМ є n ком/конц і m вузлів. Позначимо вузли u_i, $i = 1, ..., m$, а комутатори/концентратори q_j, $j = 1, ..., n$. Канали зв'язку між q_l і q_k будемо позначати S_{lk}. Обмеження з погляду завдання розміщення представлені обмеженнями на допустимі довжини каналів зв'язку. Канали S_{lk} у більшості реальних ОМ з'єднані послідовно (рис. 8.1). Канали, що замикають кільце (на рис. 8.1 – точкова лінія), необхідні підвищення надійності з допомогою оперативного формування нової магістралі у разі пошкодження каналів зв'язку.

У контексті завдання існують два види каналів зв'язку:

– < вузол > – <комутатор/концентратор>;

– <комутатор/концентратор> – <комутатор/концентратор>.

У будь-якій ОМ є m каналів першого типу (за кількістю вузлів) і не більше $n \cdot (n - 1)/2$ – каналів другого типу.

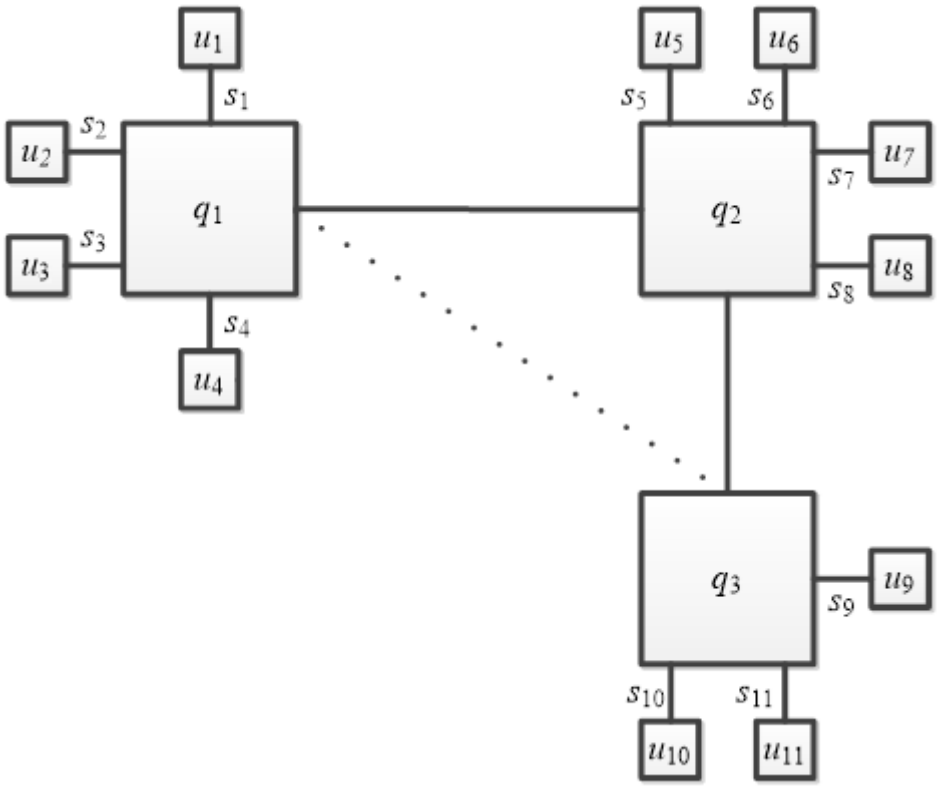


Рис. 8.1. Приклад ОМ з комутацією

У процесі модернізації ОМ зміна топології представлена такими діями: перепідключення вузлів до інших комутаторів/концентраторів або прокладання нових каналів зв'язку. Ефективність кожного нового каналу зв'язку оцінюється впливом на сумарний трафік всіх каналів зв'язку. Функцію ефективності

$$F^l = \min \| T_k - P_k \|, \text{ для } \forall\ k = 1, ..., K$$

можна багаторазово перераховувати для різних варіантів ОМ.

Змінна T_k висловлює трафік каналу k, а змінна P_k висловлює максимальну пропускну здатність каналу, K – кількість каналів зв'язку. Завдання перепідключення вузлів до інших комутаторів/концентраторів – це завдання розбиття m робочих станцій на n групп. Для кожної i-ї групи є обмеження на $n_i^{\max}$ кількості підключених станцій.

Вирішення задачі перепідключення робочих станцій – це процес спрямованого перебору варіантів підключення з метою оптимізації. Розмірність завдання велика навіть для обчислювальної мережі середнього розміру (до 1000 вузлів), тому є доцільним вирішити задачу за допомогою генетичного алгоритму.

Кодування розв'язання задачі (хромосоми) може бути наступним. Варіант розбиття вузлів сегменти, тобто, рішення зручно представляти поряд цілих чисел. Нехай всі вузли мають унікальні номери від 1 до m та упорядковані відповідно до цих номерів. Позиція і містить номер комутатора/концентратора (від 1 до n), до якого підключено вузол i. Приклад кодування топології ОМ, зображеної на рис. 8.1, наведено у табл. 8.1.

Таблиця 8.1 – Приклад кодування підключення вузлів ОМ

Комутатор/концентратор	1	1	1	1	2	2	2	2	3	3	3
Номер вузла	1	2	3	4	5	6	7	8	9	10	11

Для двійкового кодування номера q_i комутатора/концентратора необхідно: $[\log_2 n]$ біт, де [ ] – операція округлення у більшу сторону.

Для вищерозглянутого прикладу хромосома з використанням двійкового кодування наведено в табл. 8.2.

Таблиця 8.2 – Хромосома з двійковим кодуванням

Хромосома	01	01	01	01	10	10	10	10	11	11	11
Номер гена	1	2	3	4	5	6	7	8	9	10	11

При такому способі кодування і вільному механізмі мутацій теоретично можлива поява заборонених комбінацій, тому оператор мутації при використанні генетичного алгоритму потребує уточнення. Результат мутації окремого гена (позиції) є послідовністю операцій:

– випадковий вибір гена (імовірність 0,001);

– випадковий вибір біта;

– інверсія вибраного біта;

– операція по модулю n над геном, що містить результат інверсії $mod_n(x)$.

Оператор рекомбінації виконується традиційно, але межа для розрізання батьківських хромосом має бути межею генів. Найбільшу важливість має визначення функції оптимальності хромосоми, оскільки визначення впливає збіжність еволюції. Як функція оптимальності може бути взята наступна функція:

$$\max_{l} \min_{k} \left\| T_k - P_k \right\|, \text{ для } \forall\, l,\ l \in L,\ \forall\, k, k \in K,$$

де T_k – трафік каналу k; P_k – пропускна спроможність каналу; L – множина всіх варіантів вибору комунікаційного обладнання;
K – кількість каналів зв'язку.

Вигляд комунікаційного устаткування значно впливає завантаженість каналів зв'язку. Реальна ОМ часто містить як комутатори так і концентратори. Поліпшити пропускні можливості каналів зв'язку можна за рахунок оптимального вибору комутаторів або концентраторів. Завдання вибору комунікаційного устаткування задається лише на рівні каналів зв'язку, кожен канал k характеризується пропускної спроможністю – реальної P_k і максимальної P_k^{max} (біт/с). Інтенсивність взаємодії (передачі повідомлень) будь-якої пари вузлів – це величина B_{ij} (біт/с). Величина B_{ij} вимірюється протягом тривалого проміжку часу та усереднюється. Усереднення може бути представлене або обчисленням середнього значення $B_{ij}^{ср}$, або побудовою на основі гістограми розподілу ймовірностей.

Сумарний трафік, що припадає на зв'язок, залежить від типу каналу. Розглянемо тільки канали типу:

– <комутатор/концентратор> – <комутатор/концентратор>.

Тоді можна виділити такі підвиди каналів:

– <комутатор> – <комутатор>;

– концентратор> – <концентратор>;

– комутатор> – <концентратор>.

Сумарний трафік виражається по-різному трьох підвидів каналів зв'язку. Розглянемо сумарний трафік каналу типу <комутатор> – <комутатор>:

$$T_k = \sum_{j \in M_1}^{n} \sum_{j \in M_2}^{n} B_{ij}.$$

Множина вершин M_1 і множина вершин M_2 – це множини вузлів з однієї й іншої сторони від каналу зв'язку. Сумарний трафік каналу <концентратор> – <концентратор> вимірюється по-іншому, оскільки канал, утворений концентраторами, становить загальну магістраль:

$$T_k = \sum_{\substack{j \in 1 \\ i \neq j}}^{n} \sum_{\substack{j \in 1 \\ i \neq j}}^{n} B_{ij}.$$

Сумарний трафік каналу <концентратор> – <комутатор> може бути обчислений наступним чином:

$$T_k = \sum_{\substack{j \in 1 \\ i \neq j}}^{n} \sum_{j \in M_2}^{n} B_{ij}.$$

Необхідність модернізації ОМ визначається за ступенем близькості сумарного трафіку та пропускних здібностей каналів зв'язку. Для знаходження оптимального підключення використовується наступний генетичний алгоритм.

Крок 1. Генерується населення хромосом випадково з перевіркою коректності, тобто. можливості підключення, закодованого у хромосомі.

Крок 2. Для кожної хромосоми розраховується значення функції оптимальності, тобто величина трафіку на бекбоні при підключенні, закодованому у хромосомі.

Крок 3. Відповідно до значення функції оптимальності проводиться ранжування хромосом, хромосоми впорядковуються в міру зменшення ефективності.

Крок 4. Для кожної хромосоми обчислюється можливість відбору

$$P_s(a_i^t) = \frac{f(a_i^t)}{\sum_{j=1}^{\lambda} f(a_j^t)},$$

де P_s – ймовірність відбору; a_i^t – *i*-я хромосома в *t*-м поколінні; f – функція оптимальності; t – номер покоління; λ – розмір популяції.

Крок 5. Проводиться масштабування ймовірностей відбору, найбільша ймовірність приймається за одиницю, найменша – за нуль, решта пропорційно масштабується:

$$P_{sl} = \frac{P_{s_i} - P_{min}}{P_{max} - P_{min}},$$

де P_{sl} – масштабована ймовірність; P_{s_i} – ймовірність відбору; P_{min} – мінімальна ймовірність у популяції; P_{max} – максимальна ймовірність у популяції.

Крок 6. Починається формування нової популяції: виробляється елітний відбір, тобто декілька "старих" хромосом переноситься в нову популяцію.

Крок 7. Виробляється кросинговер випадково обраних хромосом. Крапка кросинговера вибирається випадково. Процес триває до того часу, поки розмір нової популяції не дорівнює розміру вихідної.

Крок 8. Перехід до кроку 2 і цикл повторюється задане число разів.

Крок 9. Зупинка.

8.1.2. Застосування генетичних алгоритмів до завдання розміщення радіоелементів у корпусі пристрою

Структура завдання розміщення різногабаритних елементів у просторі визначається наступним чином:

– обмеження простору обсягу розміщення (наприклад, габарити конкретного корпусу, визначені стандартом підприємства);

– елементи розміщення, задані своїми габаритами (обсягами).

Насправді завдання розміщення у просторі часто замінюється розміщенням на настановної площі, яку називають монтажним полем. Кожен елемент, призначений для розміщення, у разі представляється своєї настановної площею, яка грубо є прямокутником, тобто. двома габаритами: довжиною та шириною.

Припустимо, що монтажне поле теж є прямокутником. За ступенем складності розрізняють простішу завдання розміщення однакових елементів на полі з кратними габаритами і складнішу задачу розміщення елементів з різними настановними площами. Розв'яжемо завдання розміщення різногабаритних елементів на обмеженому монтажному полі.

Вихідними даними є: a, b – габарити монтажного простору; $\{(a_1, b_1), ..., (a_i, b_i), ..., (a_n, b_n)\}$ – множина елементів розміщення; C – матриця зв'язків елементів розміщення, що є матрицею зв'язності. Необхідно знайти варіант розміщення елементів на монтажному просторі

$$Z = ((x_1, y_1), ..., (x_i, y_i), ..., (x_n, y_n)),$$

де x_i, y_i – координати центру ваги настановної площі елемента розміщення i такої, щоб площа перекриття площ розміщених елементів дорівнювала нулю, а сумарна довжина з'єднань – мінімальна.

Завдання розміщення ставиться як завдання оптимізації функції, що виражає нормовану оцінку суми штрафу за перекриття площ електронних радіоелементів (ЕРЕ), що розміщуються, і загальної довжини з'єднань:

$$F = \min_{z_j \in Z}(k \cdot O(L(z_j)) + P(S_{\text{заг}}(z_j)),$$

де k – ваговий коефіцієнт; $O(L(z_j))$ – оцінка загальної довжини з'єднань, наведена до інтервалу [0, 1]; z_j – варіант розміщення; $P(S_{\text{заг}}(z_j))$ – функція штрафу за перекриття площ, що приймає значення з інтервалу [0,1]; $S_{\text{заг}}$ – загальна площа перекриття площ елементів, що розміщуються.

Загальна довжина сполук розраховується за формулою:

$$L = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \cdot c_{ij},$$

де d_{ij} – відстань між позиціями установки елементів $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$; c_{ij} – кількість зв'язків між елементами i та j з вихідної матриці суміжності C.

Нормування загальної довжини з'єднань можна провести, обчислюючи ставлення $L(z_j)$ к $L_{\max}$, де

$$L_{\max} = n^2 \cdot \sqrt{a^2 + b^2};$$
$$O(L(z_j)) = L(z_j) / L_{\max}.$$

Загальна площа перекриття обчислюється за такою формулою:

$$S_{\text{пер}} = \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij},$$

де S_{ij} – площа перекриттів елементів z_i та z_j,

$$S_{ij} = [0{,}5(a_2 + a_1) - |x_2 - x_1|][0{,}5(b_2 + b_1) - |y_2 - y_1|].$$

Функція штрафу за перекриття площ

$$P(S_{\text{общ}}) = S_{\text{общ}}/nab.$$

8.1.3. Дослідження ефективності генетичних алгоритмів для завдання розміщення радіоелементів у корпусі пристрою

Досліджуємо результативність генетичного алгоритму під час вирішення завдання розміщення елементів площині. Характерною особливістю використання стандартного генетичного алгоритму на вирішення практичних завдань є необхідність уточнення його параметрів. Для створення програмної реалізації генетичного алгоритму розміщення різногабаритних радіоелектронних елементів потрібна наступна модифікація ГА.

1. Хромосома є пов'язаним списком пар координат центрів ваги елементів розміщення. Кожна координата описується дійсним числом.

2. Як оператор мутації використовується імовірнісна зміна випадкової позиції хромосоми. Як оператор рекомбінації використовується однокрапковий варіант кросовера.

3. Умова завершення еволюції задається як визначення кількості поколінь або у вигляді умови досягнення нульового перекриття елементів.

Результати аналізу поведінки генетичного алгоритму залежно від значень ймовірності мутації наведено на рис. 8.2. Експеримент проводився для значень ймовірностей мутації в діапазоні від 0.1 до 0.6 з кроком 0.1.

Значення ймовірності понад 0,6 порушує збіжність функції оптимальності ГА.

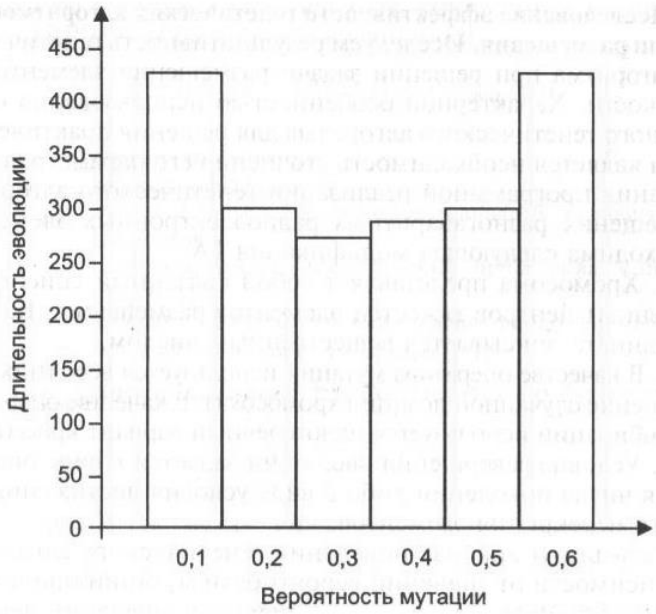


Рис. 8.2. Діаграма: тривалість еволюції залежно від значення ймовірності мутації

Крім стандартного ГА, для вирішення завдання розміщення можна використовувати еволюційні стратегії: стратегію "тільки мутація" із пропорційним оператором селекції; стратегію із рекомбінацією (m, k). Такий оператор рекомбінації передбачає m батьків та k нащадків. Параметри m і k встановлює користувач. Теоретично обмеження на параметри мають такий вигляд: $m = 1, ..., l - 1$; $k = 1, ..., m$. Параметр l – довжина хромосоми (чи у разі кількість елементів розміщення). Експериментально досліджувалися наступні поєднання батьків та нащадків: (2, 2), (3, 1), (3, 2), (3, 3), (4, 1), (4, 2), (4, 3), (4, 4). Слід зазначити, що при поєднанні (2, 1) ми отримуємо в якості оператора рекомбінації звичайний однокрапковий кросинговер, в якому беруть участь два батьки і один нащадок, що вибирається випадковим чином. На рис. 8.3 і 8.4 показано порівняння види графіків оптимальності генетичного алгоритму.

Ряд експериментів з алгоритмом, що застосовує стратегію з альтернативним оператором рекомбінації, виявив особливу результативність оператора рекомбінації (3, 3), хоча в цілому результати цього виду алгоритму гірші, ніж у випадку зі стандартним генетичним алгоритмом з однокрапковим кросинговером. Насамперед це пов'язано з тим, що значне збільшення точок розрізу хромосом веде до руйнування хороших підобластей потенційних рішень. На рис. 8.4 це помітно за численними "сплесками" на графіку.

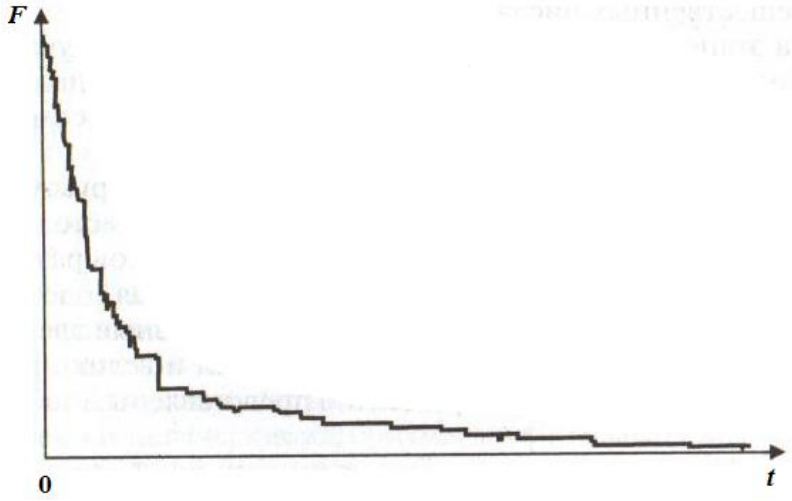


Рис. 8.3. Графік функції оптимальності ГА при одноточковому кросівері

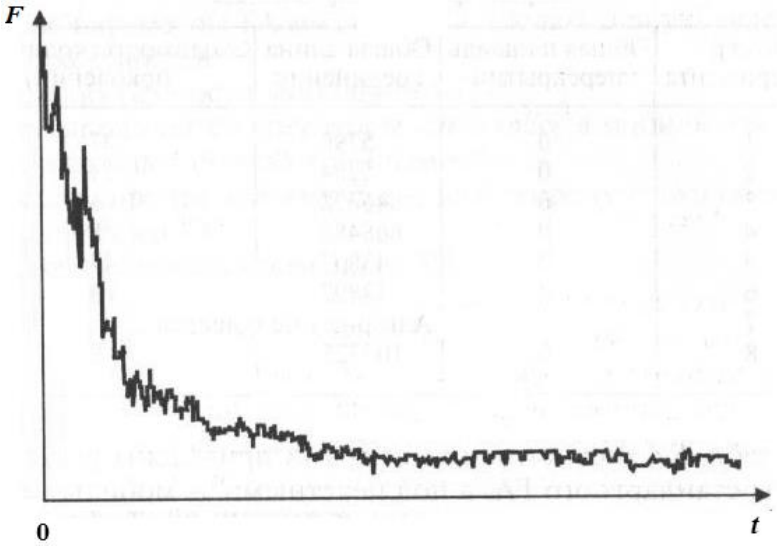


Рис. 8.4. Графік функції оптимальності ГА при стратегії рекомбінації (*m*, *k*)

При застосуванні мобільного генетичного алгоритму практично виникають проблеми, як із формулюванням початкової популяції рішень, і з функцією оптимальності. Структура хромосоми є список пар координат, упорядкованих за номером елемента: ((1, x_1, y_1), ..., (i, x_i, y_i), ..., (v, x_v, y_v)).

У мобільному генетичному алгоритмі хромосома кодується списком пар номер гена, значення гена. При вирішенні завдання розміщення номер гена збігається з номером елемента, що розміщується, а значення гена - з парою координат. Таким чином, номер гена – це ціле число, а координати – два дійсні числа.

На етапі ініціалізації генерується λ рядків випадковим чином. Параметр λ – розмір популяції, задається користувачем. Функція оптимальності кожної хромосоми обчислюється, як площа перекриття елементів.

Ефективність мобільного ГА для завдань структурного синтезу можна перевірити на наступних чотирьох тестових наборах: "велике монтажне поле та мало елементів розміщення"; "багато елементів розміщення" (висока щільність упаковки); "низька щільність упаковки, але різні елементи розміщення"; "висока щільність упаковки і велика різноманітність". Результати експериментів представлені в табл. 8.3.

Таблиця 8.3 – Параметри стандартного ГА

Номер експерименту	Загальна площа перекриттів	Загальна довжина з'єднань	Сходимість (кількість поколінь)
1	0	5796	37
2	0	5224	11
3	0	646628	6
4	0	668482	8
5	0	43835	47
6	0	38897	13
7	Алгоритм не зійшовся		
8	0	103325	8

У табл. 8.3 під парними номерами наведено результати роботи стандартного ГА, а під непарними – мобільного ГА. На підставі проведених експериментів можна зробити висновок про результативність мобільного генетичного алгоритму.

8.2. Індивідуальні завдання

1. Для студентів з парними номерами в журналі групи необхідно написати програму, будь-якою мовою високого рівня, що реалізує вирішення сформульованої задачі оптимізації обчислювальної мережі з використанням описаного генетичного алгоритму, а для студентів з непарними номерами в журналі групи – програму, що реалізує вирішення сформульованої задачі розміщення радіоелементів корпусі пристрою.

2. Обґрунтувати вибір усіх параметрів та критеріїв зупинення роботи генетичного алгоритму.

3. Оформіть звіт з лабораторної роботи.

Лабораторна робота 9

НЕЙРО-НЕЧІТКЕ МОДЕЛЮВАННЯ

Мета лабораторної роботи: отримання та закріплення знань про методи моделювання та принципи функціонування нейронечітких систем, а також формування практичних навичок щодо конструювання нейронечітких мереж у пакеті MATLAB.

9.1. Короткі відомості з теорії

Як правило, у стані нестабільної економіки, що характеризується частою зміною економічних умов, прийняття управлінських рішень здійснюється в умовах невизначеності, внаслідок чого завдання планування економічної діяльності та прогнозування її результатів є одним із найскладніших і неоднозначних.

Існує низка класичних методів прогнозування економічних показників, що базуються на апараті математичної статистики, серед яких виділяються методи аналізу та моделювання часових рядів, методи багатовимірного регресійного аналізу. Особливістю зазначених методів є необхідність чіткої специфікації моделей, що конструюються, крім того, додаткові труднощі для використання даних методів створює нестаціонарність досліджуваних економічних процесів.

Перспективним напрямом у галузі вирішення завдань прогнозування є застосування апарату штучних нейронних мереж та нейронечітких мереж.

Принцип роботи нейронної мережі у тому, що з наявному набору даних конструюється певна залежність між вхідними і вихідними змінними системи, у процесі навчання мережі налаштовуються параметри (ваги) одержуваних функціональних відносин. (Як правило, виявлення та визначення даної залежності в явному вигляді не є можливим в силу

вищевказаних причин). Модель нейронної мережі позиціонується як "чорна скринька" внаслідок того, що внутрішній алгоритм її налаштування не "прозорий", і отримані результати та взаємозв'язки складно інтерпретувати.

Нечіткі нейронні мережі або гібридні мережі покликані поєднати в собі переваги нейронних мереж та систем нечіткого виведення. З одного боку, вони дозволяють розробляти і представляти моделі систем у формі правил нечітких продукцій, які мають наочність і простоту змістовної інтерпретації. З іншого боку, для побудови правил нечітких продукцій використовуються методи нейронних мереж, що є зручнішим та менш трудомістким процесом для системних аналітиків.

У цих методичних вказівках розглядаються основні поняття нейронечітких мереж, а також засоби їх розробки, що надаються системою MATLAB.

Для прогнозування використовуємо нечітку мережу *TSK*. Узагальнену схему виведення моделі TSK під час використання M правил і N змінних x_j можна представити у вигляді:

$$IF\ (x_1\ IS\ A_1^{(1)})\ AND\ (x_2\ IS\ A_2^{(1)})\ AND\ \ldots\ AND\ (x_n\ IS\ A_n^{(1)}),$$
$$THEN\ y_1 = p_{10} + \sum_{j=1}^{N} p_{1j} x_j$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$IF\ (x_1\ IS\ A_1^{(M)})\ AND\ (x_2\ IS\ A_2^{(M)})\ \ AND\ \ldots\ AND\ (x_n\ IS\ A_n^{(M)}),$$
$$THEN\ y_M = p_{M0} + \sum_{j=1}^{N} p_{Mj} x_j\ .$$

Умова $IF\ (x_i\ IS\ A_i)$ реалізується функцією фазифікації, яка є узагальненою функцією Гауса окремо для кожної змінної x_i:

$$\mu_A(x_i) = \frac{1}{1 + \left(\frac{x_i - c_i}{\sigma_i} \right)^{2b_i}}, \qquad (9.1)$$

де $\mu_A(x_i)$ представляє оператор A_i. У нечітких мережах доцільно задавати цю умову у формі твору алгебри, з якого випливає, що для k-го правила виведення

$$\mu_A^{(k)}(x) = \prod_{j=1}^{N} \left[\frac{1}{1 + \left(\frac{x_i - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}}} \right]. \qquad (9.2)$$

При M правила агрегування вихідного результату мережі проводиться за формулою

$$y = \sum_{i=1}^{M} \frac{w_i}{\sum_{j=1}^{N} w_j} \left(p_{i0} + \sum_{j=1}^{N} p_{ij} x_j \right), \qquad (9.3)$$

яку можна представити у вигляді

$$y(x) = \frac{1}{\sum_{k=1}^{N} w_k} \left(\sum_{k=1}^{M} w_k y_k(x) \right), \qquad (9.4)$$

де $y_k(x) = p_{k0} + \sum_{j=1}^{N} p_{kj} x_j$.

Присутні у цьому виразі ваги w_k інтерпретуються як значущість компонентів $\mu_A^{(k)}(x)$, визначених формулою (9.1). За цієї умови формулою (9.4) можна зіставити багатошарову структуру мережі (рис. 9.1). У такій мережі виділяється п'ять шарів:

– перший шар виконує роздільну фазифікацію кожної змінної x_i (i = 1, 2, …, N), визначаючи для кожного k-го правила виведення значення

коефіцієнта належності $\mu_A^{(k)}(x_i)$ відповідно до застосовуваної функції фазифікації. Це параметричний шар із параметрами $c_j^{(k)}$, $\sigma_j^{(k)}$, $b_j^{(k)}$, підлягають адаптації у процесі навчання;

– другий шар (непараметричний) виконує агрегування окремих змінних x_i, визначаючи результуюче значення коефіцієнта належності $w_k = \mu_A^{(k)}(x)$ для вектору x (Рівень активізації правила виведення) відповідно до формули (9.2);

– третій шар є генератором функції *TSK*, що розраховує значення $y_k(x) = p_{k0} + \sum_{j=1}^{N} p_{kj} x_j$. У цьому шарі також здійснюється множення сигналів $y_k(x)$ на значення w_k, сформовані на попередньому шарі. Це параметричний шар, в якому адаптації підлягають лінійні ваги. p_{kj} для $k = 1, 2, \ldots, M$ і $j = 1, 2, \ldots, N$, визначальні функції слідства моделі *TSK*;

– четвертий шар (непараметричний) складають два нейрони-суматори, один з яких розраховує зважену суму сигналів $y_k(x)$, а другий визначає суму ваг $\sum_{k=1}^{M} w_k$;

– останній, п'ятий шар (нормалізуючий), складається з єдиного вихідного нейрона, в якому ваги зазнають нормалізації відповідно до формули (9.4). Вихідний сигнал *y(x)* визначається виразом, що відповідає залежності (9.3),

$$y(x) = f(x) = \frac{f_1}{f_2}. \qquad (9.5)$$

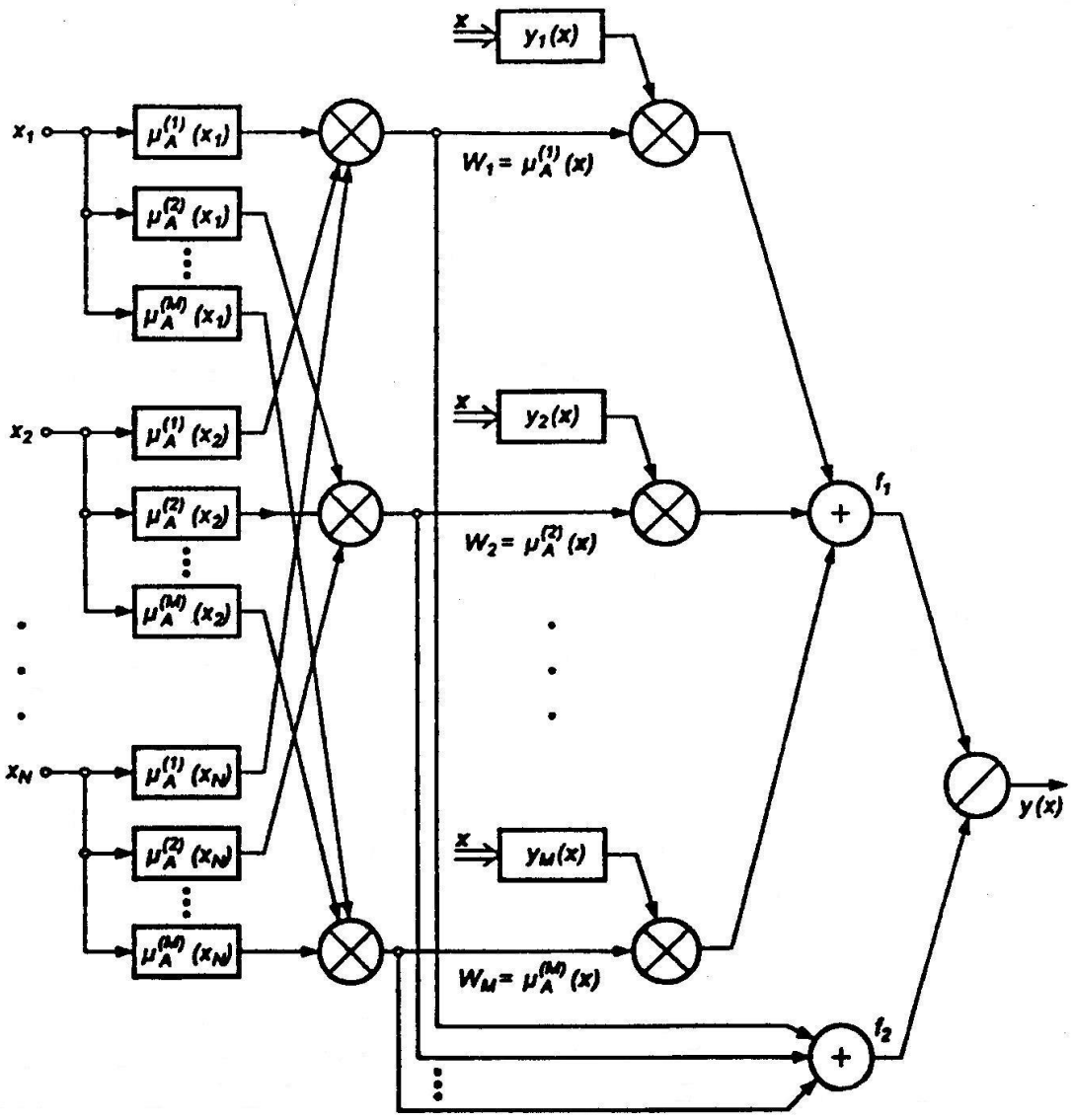


Рис. 9.1. Структура нечіткої нейронної мережі *TSK*

З наведеного опису випливає, що нечітка мережа *TSK* містить лише два параметричні шари (перший та третій), параметри яких уточнюються у процесі навчання. Параметри першого шару називатимемо нелінійними параметрами, оскільки вони належать до нелінійної функції (9.1), а параметри третього шару – лінійними вагами, т.к. вони відносяться до параметрів p_{kj} лінійної функції *TSK*.

При уточненні функціональної залежності (9.4) для мережі *TSK* отримуємо:

$$y(x) = \frac{1}{\sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]} \sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]\left[p_{k0} + \sum_{j=1}^{N} p_{kj}x_j\right]. \qquad (9.6)$$

Якщо прийняти, що в конкретний момент часу параметри умови зафіксовані, то функція $y(x)$ є лінійною щодо змінних x_i (i = 1, 2, …, N).

За наявності N вхідних змінних кожне правило формує N+1 змінних $p_j^{(k)}$ лінійної залежності *TSK*. При M правила виведення це дає $M(N+1)$ лінійних параметрів мережі. У свою чергу кожна функція належності використовує три параметри (c, s, b), що підлягають адаптації. Якщо прийняти, кожна змінна x_i характеризується своєї функцією власності, то при M правилах виведення ми отримаємо $3MN$ нелінійних параметрів. У сумі це дає $M(4N+1)$ лінійних та нелінійних параметрів, значення яких повинні підбиратися в процесі навчання мережі.

Насправді зменшення кількості адаптируемых параметрів оперують меншою кількістю незалежних функцій приналежності окремих змінних, керуючись правилами, у яких комбінуються функції приналежності різних змінних. Якщо прийняти, що кожна змінна x_i має m різних функцій приналежності, то максимальна кількість правил, які можна створити при їх комбінуванні, становитиме: $M = m^N$ (при трьох функціях приналежності, що поширюються на дві змінні, це $3^2 = 9$ правил виведення). Таким чином, сумарна кількість нелінійних параметрів мережі при M правилах виведення зменшується з $3MN$ у загальному випадку до $3NM^{1/N}$. Кількість лінійних параметрів за такої модифікації залишається без змін, тобто $M(N+1)$.

Гібридна мережа як адаптивна система нейронечіткого виведення. Гібридна мережа є багатошаровою нейронною мережею спеціальної структури без зворотних зв'язків, в якій використовуються звичайні (не нечіткі) сигнали, ваги та функції активації, а виконання операції підсумовування засноване на використанні фіксованої T-норми, S-кнорми або деякої іншої безперервної операції. При цьому значення входів, виходів

і ваг гібридної нейронної мережі є речовими числами з відрізка [0, 1].

Основна ідея, покладена в основу моделі гібридних мереж, полягає в тому, щоб використовувати існуючу вибірку даних для визначення параметрів приналежності, які найкраще відповідають деякій системі нечіткого виведення. При цьому знаходження параметрів функцій приналежності використовуються відомі процедури навчання нейронних мереж.

У пакеті *Fuzzy Logic Toolbox* системи MATLAB гібридні мережі реалізовані у формі так званої адаптивної системи нейронечіткого виведення *ANFIS*. З одного боку, гібридна мережа *ANFIS* – це нейронна мережа з єдиним виходом та кількома входами, які є нечіткими лінгвістичними змінними. При цьому терми вхідних лінгвістичних змінних описуються стандартними для системи MATLAB функціями приналежності, а вихідні терми змінної представляються лінійною або постійною функцією приналежності.

З іншого боку, гібридна мережа *ANFIS* є системою нечіткого виведення *FIS* типу Сугено нульового або першого порядку, в якій кожне з правил нечітких продукцій має постійну вагу, рівну 1.

9.2. Моделювання та реалізація нейронечіткої мережі у середовищі MATLAB

У пакеті Fuzzy Logic Toolbox системи MATLAB гібридні мережі реалізовані у формі адаптивних систем нейронечіткого виведення *ANFIS*. При цьому розробка та дослідження гібридних мереж виявляється можливою:

– в інтерактивному режимі за допомогою спеціального графічного редактора адаптивних мереж, який отримав назву редактора *ANFIS*;

– у режимі командного рядка за допомогою введення імен відповідних функцій з необхідними аргументами безпосередньо у вікно команд системи MATLAB.

Редактор *ANFIS* дозволяє створювати або завантажувати конкретну модель адаптивної системи нейронечіткого виведення, виконувати її навчання, візуалізувати її структуру, змінювати та налаштовувати її параметри, а також використовувати налаштовану мережу для отримання результатів нечіткого виводу.

9.2.1. Опис ANFIS-редактора

ANFIS є абревіатурою *Adaptive Neuro-Fuzzy Inference System* – (адаптивна нейронечітка система). *ANFIS*-редактор дозволяє автоматично синтезувати з експериментальних даних нейронечіткі мережі. Нейронечеткую мережу можна як одну з різновидів систем нечіткого логічного висновку типу Сугено. У цьому функції приналежності синтезованих систем налаштовані (навчені) те щоб мінімізувати відхилення між результатами нечіткого моделювання і експериментальними даними. Завантаження *ANFIS*-редактора здійснюється за командою *anfisedit*. В результаті виконання цієї команди з'явиться графічне вікно (рис. 9.2), на якому розміщені функціональні області *ANFIS*-редактора.

ANFIS-редактор містить 3 верхні меню - *File, Edit* і *View*, область візуалізації, область властивостей *ANFIS*, область завантаження даних, область генерування вихідної системи нечіткого логічного висновку, область навчання, область тестування, область виведення поточної інформації, а також кнопки *Help* і *Close*, які дозволяють викликати вікно довідки та закрити *ANFIS*-редактор, відповідно.

Меню *File* і *View* однакові всім *GUI*-модулів, використовуваних із системами нечіткого логічного виводу.

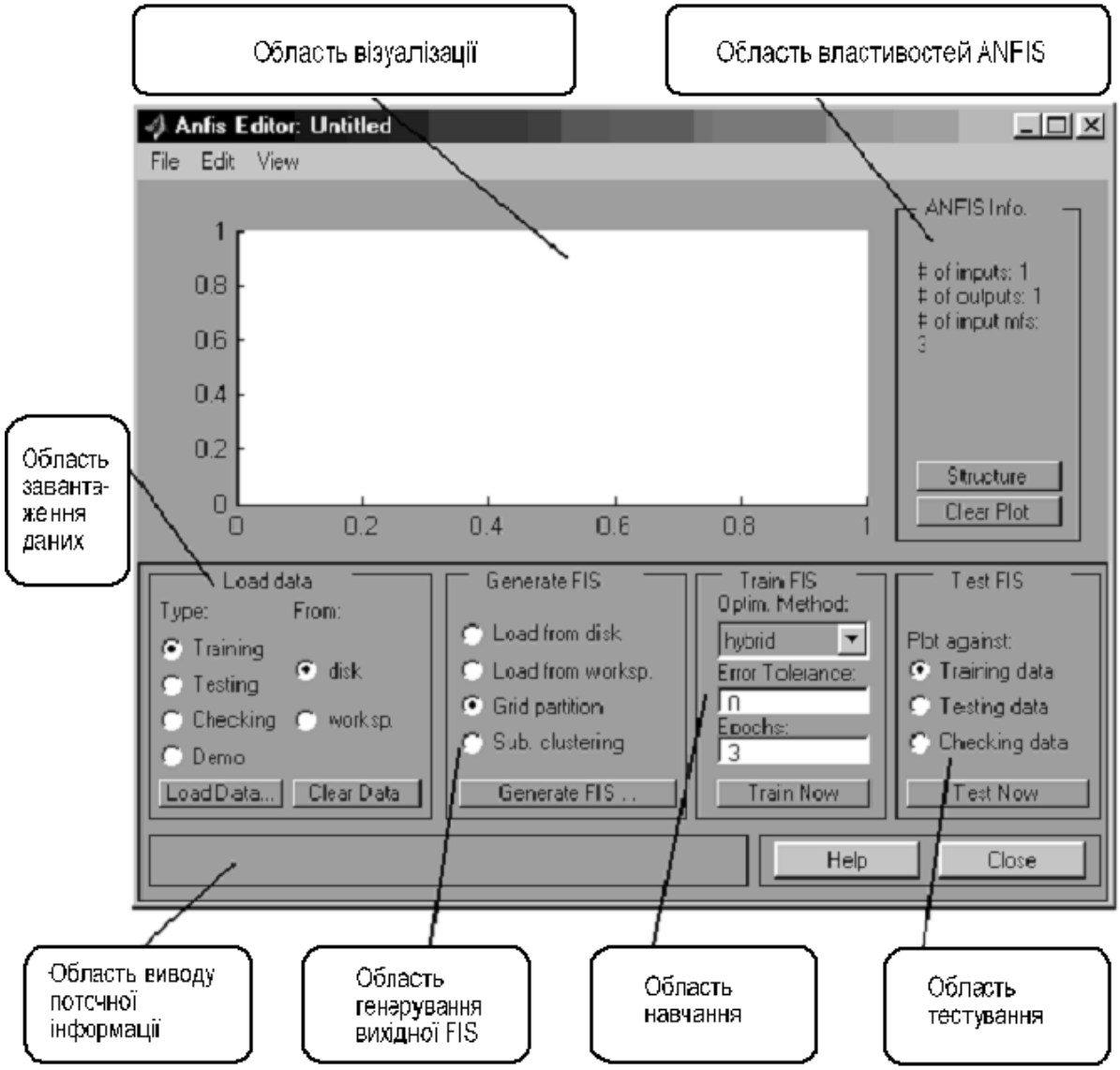


Рис. 9.2. Основне вікно ANFIS-редактора

Меню *Edit*. Загальний вигляд меню наведено на рис. 9.3.

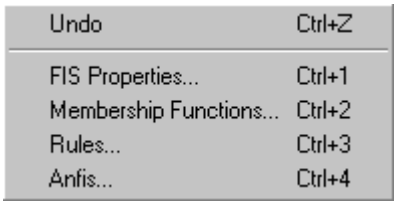


Рис. 9.3. Меню *Edit*

Команда *Undo* скасовує раніше досконалу дію. Виконується також після натискання <*Ctrl+Z*>.

Команда *FIS Properties…* відкриває *FIS*-редактор. Ця команда також може бути виконана натисканням <*Ctrl*+1>.

Команда *Membership Functions…* відкриває редактор функцій належності. Ця команда може бути виконана також натисканням <*Ctrl*+2>.

Команда *Rules…* відкриває редактор бази знань. Ця команда може бути виконана також натисканням <*Ctrl*+3>.

Команда *Anfis…* відкриває *ANFIS*-редактор. Ця команда також може бути виконана натисканням <*Ctrl*+3>. Зауважимо, що ця команда, запущена з *ANFIS*-редактора, не призводить до виконання будь-яких дій, томущо цей редактор вже відкритий. Однак у меню *Edit* інших *GUI*-модулів, що використовуються із системами нечіткого логічного висновку, додається команда *Anfis…*, що дозволяє відкрити *ANFIS*-редактор із цих модулів.

Область візуалізації. У цій області виводиться два типи інформації:

– під час навчання системи – крива навчання у вигляді графіка залежності помилки навчання від порядкового номера ітерації;

– при завантаженні даних та тестуванні системи – експериментальні дані та результати моделювання.

Експериментальні дані та результати моделювання виводяться у вигляді безлічі точок у двомірному просторі. У цьому по осі абсцис відкладається порядковий номер рядки даних у вибірці (навчальної, тестуючої чи контрольної), а, по осі ординат – значення вихідний змінної для цієї рядки вибірки. Використовуються такі маркери:

– голуба точка (.) – тестуюча вибірка;

– голубе коло (*o*) – навчальна вибірка;

– голубий плюс (+) – контрольна вибірка;

– червона зірочка (*) – результати моделювання.

Область властивостей ANFIS. В області властивостей *ANFIS* (*ANFIS info*) виводиться інформація про кількість вхідних та вихідних змінних, про кількість функцій належності для кожної вхідної змінної, а також кількість

рядків у вибірках. У цій області розташовані дві кнопки: *Structure* та *Clear Plot*.

Натискання кнопки *Structure* відкриває нове графічне вікно, у якому система нечіткого логічного виводу представляється як нейронечіткої мережі. Як ілюстрація наведена нейронечітка мережа, що містить чотири вхідні змінні та одну вихідну (рис. 9.4). У цій системі по три лінгвістичні терми використовуються для оцінки кожної з вхідних змінних та чотири терми – для вихідний.

Натискання кнопки *Clear Plot* дозволяє очистити область візуалізації.

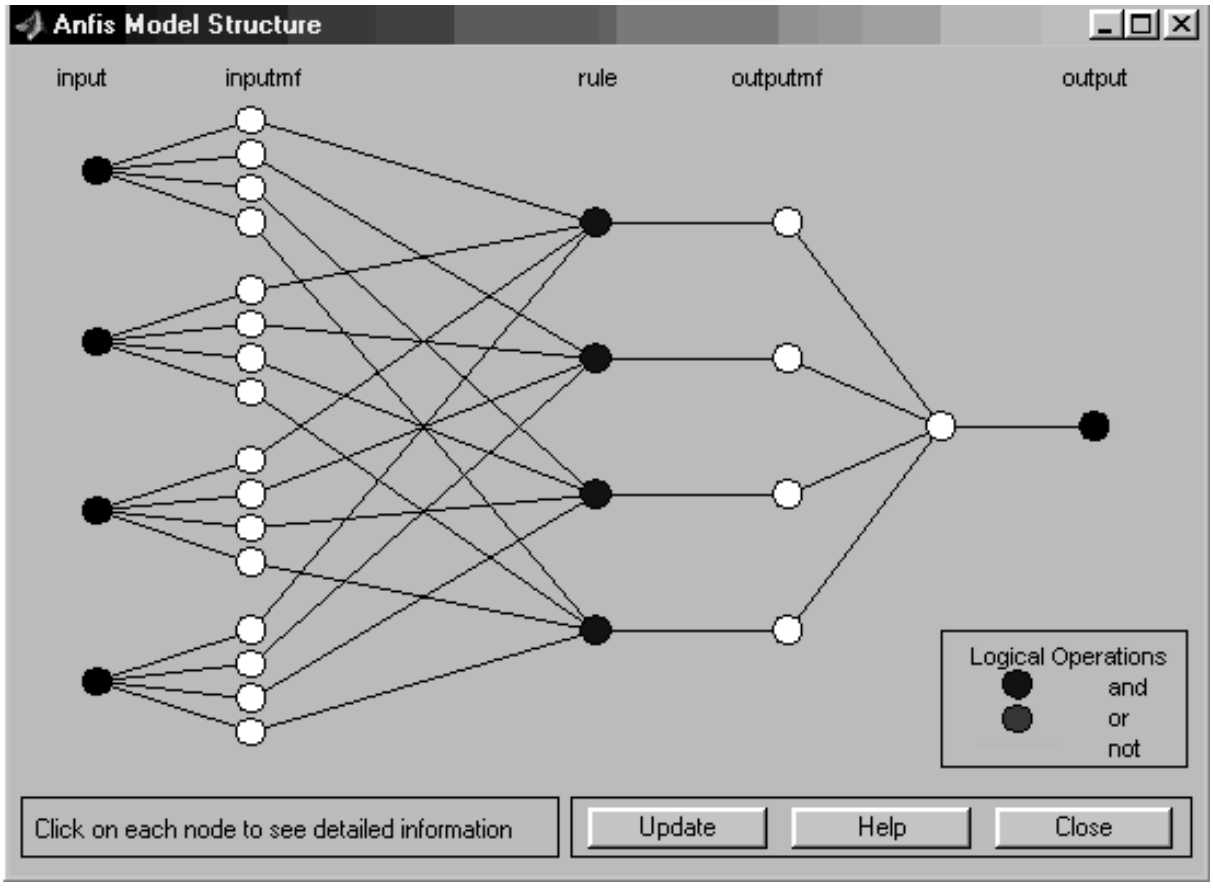


Рис. 9.4. Приклад структури нейронечіткої мережі

Область завантаження даних. В області завантаження даних *(Load data)* розташовані:

– меню вибору типу даних (*Type*), що містить альтернативи (*Traning* – навчальна вибірка; *Testing* – тестувальна вибірка; *Checking* – контрольна вибірка; *Demo* – демонстраційний приклад);

– меню вибору джерела даних (*From*), що містить альтернативи (*disk* – диск; *worksp* – робоча область MATLAB);

– кнопка завантаження даних *Load Data*…, після натискання якої з'являється діалогове вікно вибору файлу, якщо завантаження даних відбувається з диска, або вікно введення ідентифікатора вибірки, якщо завантаження даних відбувається з робочої області;

– кнопка очищення даних *Clear Data*.

Протягом сеансу роботи *ANFIS*-редактора можна завантажувати дані одного формату, тобто. кількість вхідних змінних у вибірках має бути однаковою.

Область створення вихідної системи нечіткого логічного виводу. У сфері генерування (*Generate FIS*) розташоване меню вибору способу створення вихідної системи нечіткого логічного вывода. Меню містить такі альтернативи:

– *Load from disk* – загрузка системи з диска;

– *Load from worksp* – загрузка системи з робочої області MATLAB;

– *Grid partition* – генерування системи за методом грат (без кластеризації);

– *Sub. Clustering* – генерування системи за методом субкластеризації.

В області також розташована кнопка *Generate*, після натискання якої генерується вихідна система нечіткого логічного висновку.

Під час вибору *Load from disk* з'являється стандартне діалогове вікно відкриття файлу.

Під час вибору *Load from worksp* з'являється стандартне діалогове вікно введення ідентифікатора системи нечіткого логічного виведення.

При виборі *Grid partition* з'являється вікно введення параметрів методу решітки (рис. 9.5), в якому потрібно вказати кількість термів для кожної вхідної змінної та тип функцій приладдя для вхідних та вихідних змінних.

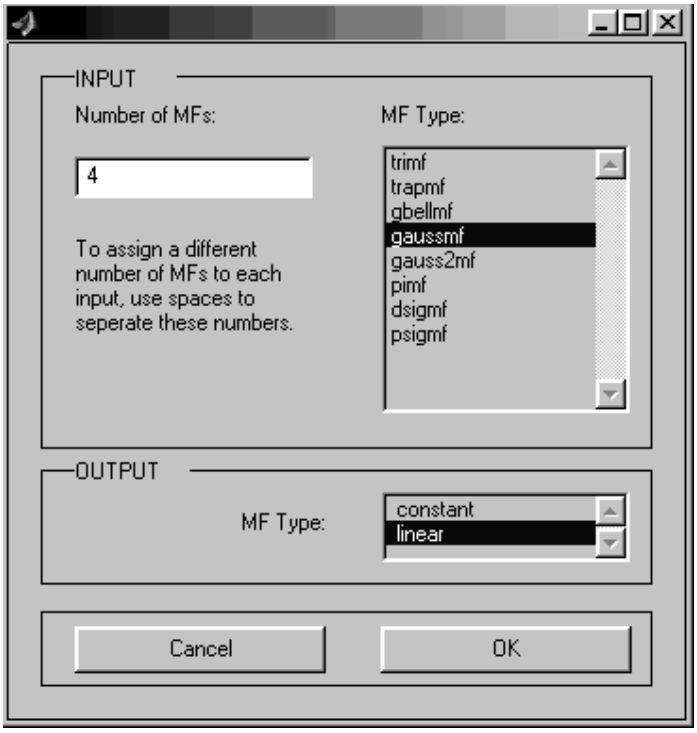


Рис. 9.5. Вікно введення параметрів для методу решітки

Під час вибору *Sub. clustering* з'являється вікно введення наступних параметрів методу субкластеризації (рис. 9.6):

– *Range of influence* – рівні впливу вхідних змінних;

– *Squash factor* – коефіцієнт придушення;

– *Accept ratio* – коефіцієнт, що встановлює у скільки разів потенціал даної точки повинен бути вищим за потенціал центру першого кластера для того, щоб центром одного з кластерів була призначена розглянута точка;

– *Reject ratio* – коефіцієнт, що встановлює у скільки разів потенціал даної точки повинен бути нижчим за потенціал центру першого кластера, щоб розглянута точка була виключена з можливих центрів кластерів.

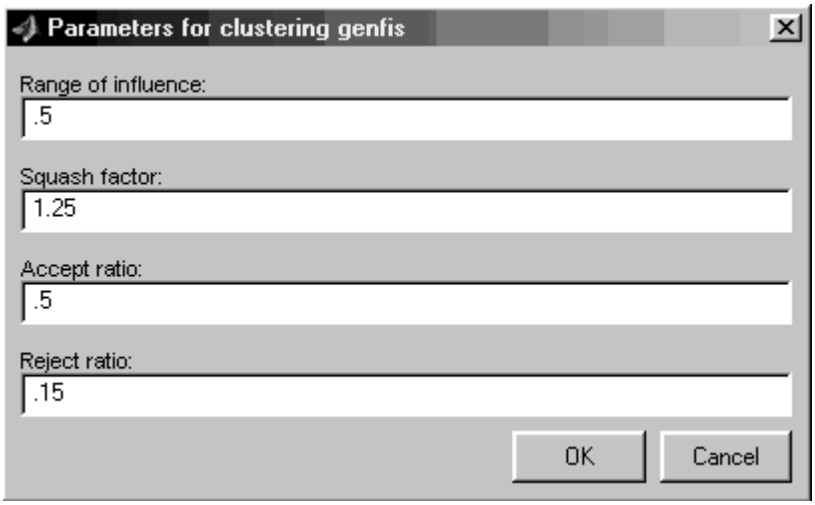


Рис. 9.6. Вікно введення параметрів для методу субкластеризації

Область навчання. В області навчання (*Train FIS*) розташовані меню вибору методу оптимізації (*Optim. method*), поле завдання необхідної точності навчання (*Error tolerance*), поле завдання кількості ітерацій навчання (*Epochs*) та кнопка *Train Now*, натискання якої запускає режим навчання. Проміжні результати навчання виводяться в область візуалізації та робочу область MATLAB. В *ANFIS*-редакторі реалізовано два методи навчання:

– *backpropa* – метод зворотного поширення помилки, заснований на ідеях методу якнайшвидшого спуску;

– *hybrid* – гібридний метод, що поєднує метод зворотного розповсюдження помилки з методом найменших квадратів.

Область тестування. В області тестування (*Test FIS*) розташовані меню вибору вибірки та кнопка *Test Now*, після натискання якої відбувається тестування нечіткої системи з виведенням результатів в область візуалізації.

Область виведення поточної інформації. У цій галузі виводиться найбільш істотна поточна інформація, наприклад повідомлення про закінчення виконання операцій, значення помилки навчання або тестування і т.п.

9.2.2. Синтез нейронечіткої мережі у середовищі MATLAB

Опис завдання: є вихідні дані зміни фінансового індексу валюти за період з 01.03.2023 до 30.04.2023. Потрібно побудувати нейронечітку мережу та спрогнозувати значення індексу на 1.05.2023.

Загальна послідовність процесу розробки моделі гібридної мережі може бути представлена у такому вигляді.

1. Підготовка файлу з навчальними даними (табл. 9.1). Доцільно користуватися редактором електронних таблиць *MS Excel*. Навчальну вибірку необхідно зберегти у зовнішньому файлі із розширенням **.dat*. Алгоритм прогнозування має на увазі те, що кожне наступне значення розраховується на основі кількох попередніх.

2. Відкрити редактор *ANFIS*. Завантажити файл із навчальними даними. Кнопка завантаження даних *Load Data*, після натискання якої з'являється діалогове вікно вибору файлу, якщо завантаження даних відбувається з диска, або вікно введення ідентифікатора вибірки, якщо завантаження даних відбувається з робочої області.

Зовнішній вигляд редактора *ANFIS* із завантаженими навчальними даними зображено на рис. 9.7.

3. Після підготовки та завантаження навчальних даних можна згенерувати структуру системи нечіткого виведення *FIS* типу Сугено, яка є моделлю гібридної мережі у системі MATLAB. Для цього потрібно скористатися кнопкою *Generate FIS* в нижній частині робочого вікна редактора. При цьому дві перші опції відносяться до попередньо створеної структури гібридної мережі, а дві останні - до форми розбиття вхідних змінних моделі.

Перед генерацією структури системи нечіткого виведення типу Сугено після виклику діалогового вікна властивостей задамо для кожної з вхідних змінних по три лінгвістичні терми, а як тип їх функцій приналежності виберемо трикутні функції.

Таблиця 9.1 – Набір даних для навчання нейронечіткої мережі

Перша вхідна змінна	Друга вхідна змінна	Третя вхідна змінна	Вихідна змінна
688,72	686,21	667,27	669,26
686,21	667,27	669,26	673,25
667,27	669,26	673,25	688,68
669,26	673,25	688,68	680,86
673,25	688,68	680,86	671,33
688,68	680,86	671,33	669,55
680,86	671,33	669,55	676,20
671,33	669,55	676,20	680,57
669,55	676,20	680,57	698,70
676,20	680,57	698,70	708,59
680,57	698,70	708,59	721,81
698,70	708,59	721,81	712,88
708,59	721,81	712,88	713,15
721,81	712,88	713,14	705,16
712,88	713,14	705,16	706,71
713,14	705,16	706,71	716,55
705,16	706,71	716,55	721,35
706,71	716,55	721,35	736,28
688,72	686,21	667,27	669,26
686,21	667,27	669,26	673,25
667,27	669,26	673,25	688,68
669,26	673,25	688,68	680,86
673,25	688,68	680,86	671,33
688,68	680,86	671,33	669,55
680,86	671,33	669,55	676,20
671,33	669,55	676,20	680,57

Після натискання кнопки *Generate FIS* викликається діалогове вікно із зазначенням числа та типу функцій приладдя для окремих термів вхідних змінних та вихідний змінної (рис. 9.8).

4. Після створення структури гібридної мережі можна візуалізувати її структуру, для чого слід натиснути кнопку *Structure* у правій частині графічного вікна (рис. 9.9).

Для прикладу, що розглядається система нечіткого виведення містить три вхідних змінних з трьома термами кожна, 27 правил нечітких продукцій, одну вихідну змінну з 27 термами.

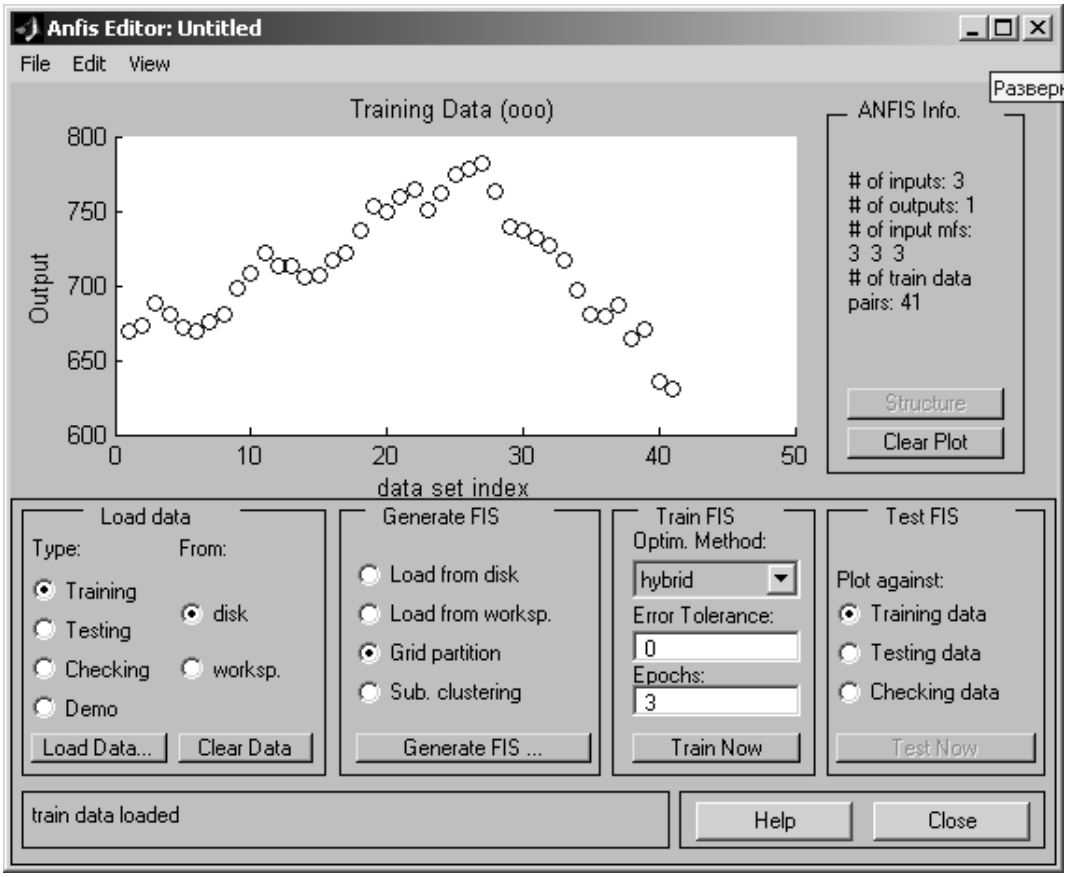


Рис. 9.7. Графічний інтерфейс редактора *ANFIS* після завантаження навчальних даних

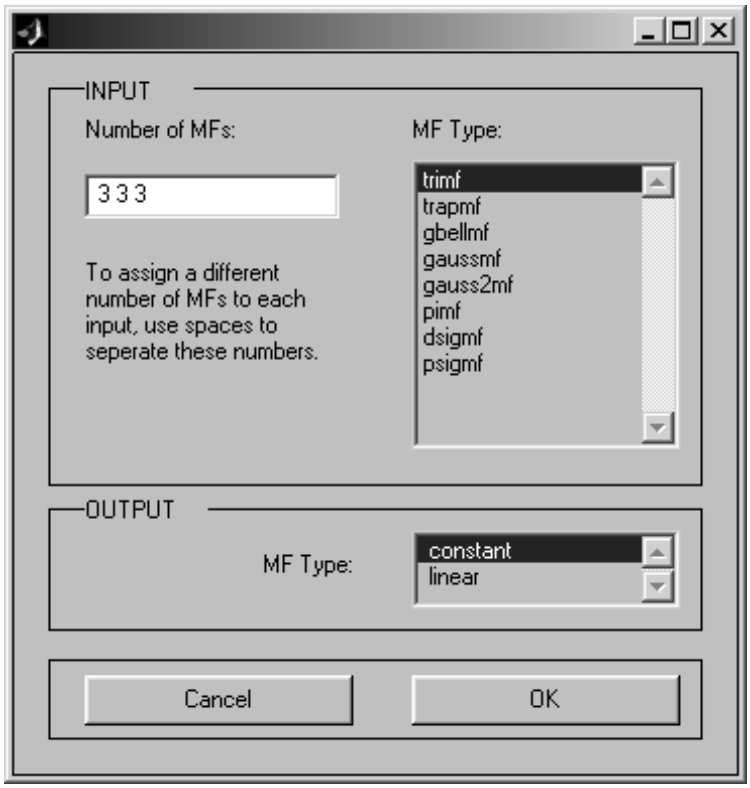


Рис. 9.8. Діалогове вікно для визначення кількості та типу функцій належності

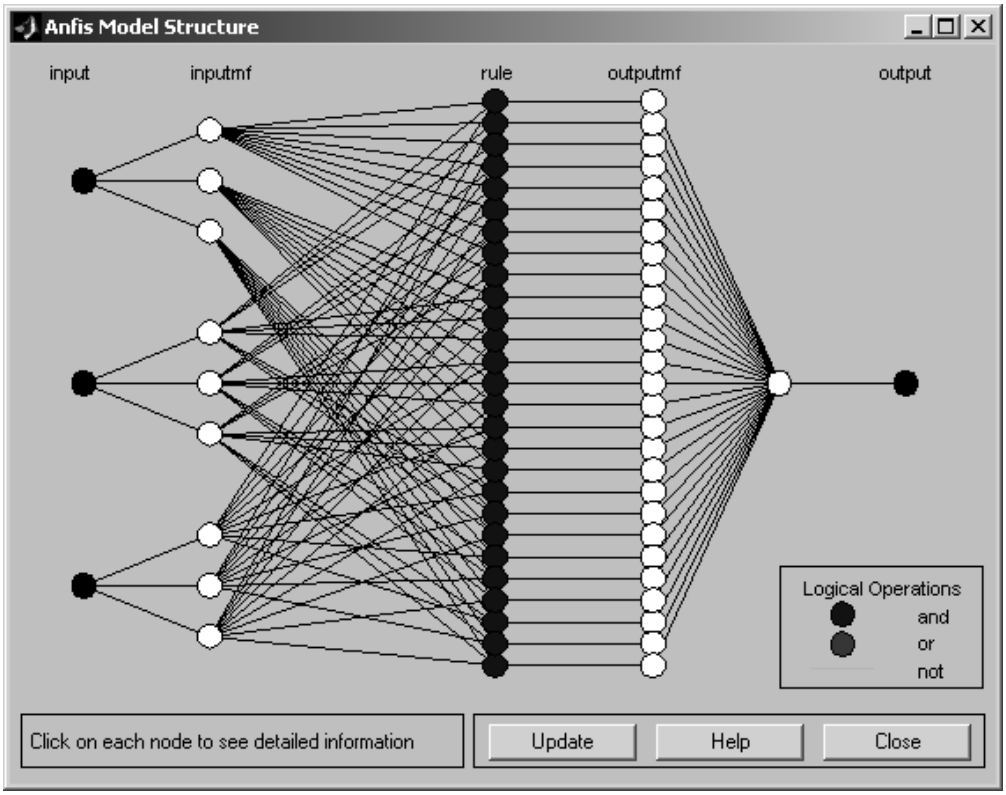


Рис. 9.9. Структура згенерованої системи нечіткого висновку

5. Перед навчанням гібридної мережі необхідно задати параметри навчання, для чого слід скористатися наступною групою опцій у нижній правій частині робочого вікна:

1. Вибрати метод навчання гібридної мережі - зворотного поширення (*backpropo*) або гібридний (*hybrid*), що є комбінацією методу найменших квадратів і методу зменшення зворотного градієнта.

2. Встановити рівень помилки навчання (*Error Tolerance*) – за промовчанням значення 0 (змінювати не рекомендується).

3. Задати кількість циклів навчання (*Epochs*) – за замовчуванням значення 3 (рекомендується збільшити для прикладу, що розглядається, задати його значення рівним 40).

Для навчання мережі потрібно натиснути кнопку *Train now*. При цьому перебіг процесу навчання ілюструється у вікні візуалізації у формі графіка залежності помилки від кількості циклів навчання (рис. 9.10).

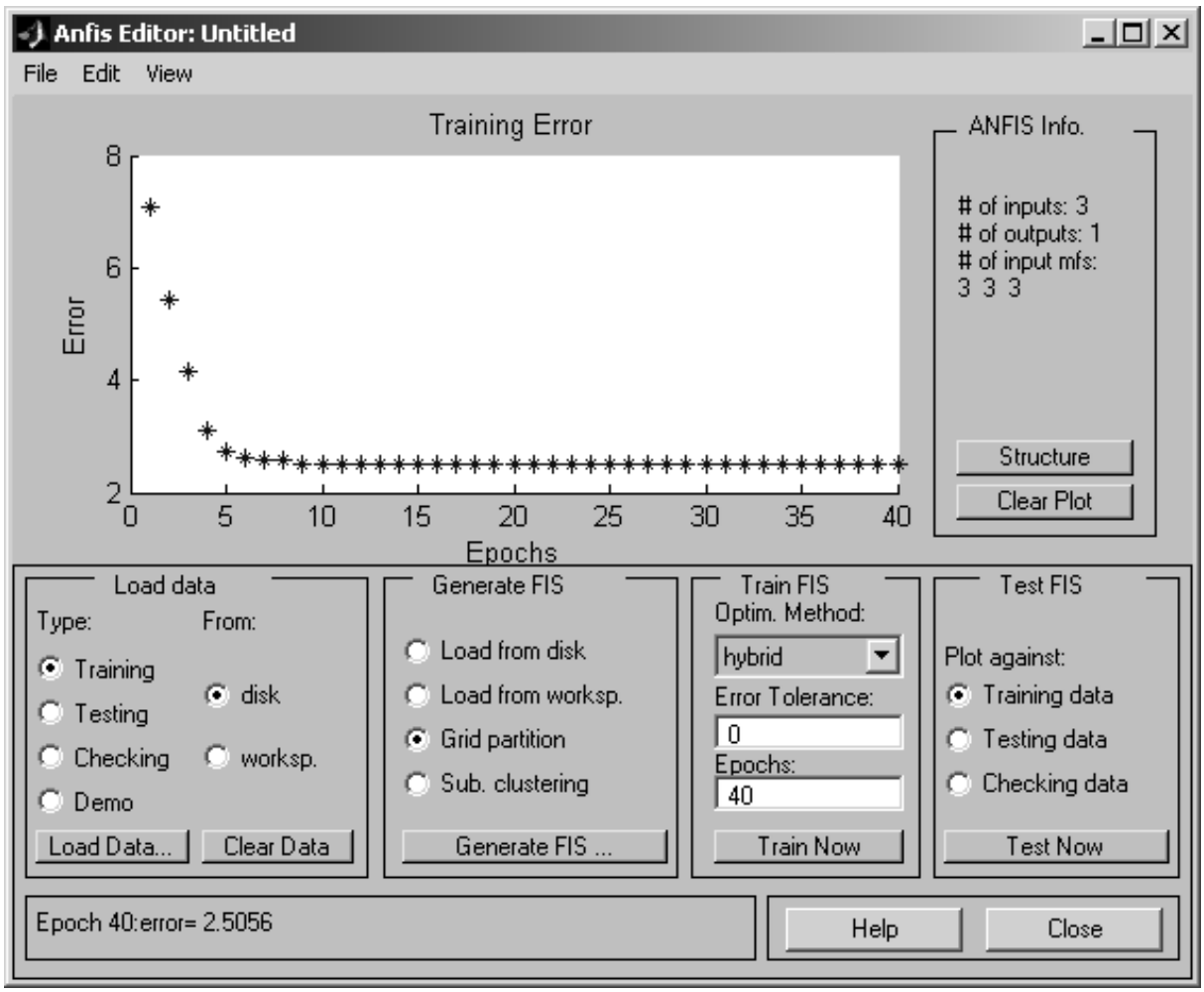


Рис. 9.10. Графік залежності помилок навчання від кількості циклів навчання

Подальше налаштування параметрів побудованої та навченої гібридної мережі може бути виконане за допомогою стандартних графічних засобів пакету *Fuzzy Logic Toolbox* (рис. 9.11 – 9.14). Для цього рекомендується зберегти створену систему нечіткого виведення у зовнішньому файлі з розширенням **.fis*, після чого слід завантажити цей файл до редактора систем нечіткого виведення *FIS*.

6. Виконаємо перевірку адекватності побудованої нечіткої моделі гібридної мережі. Для цього можна спрогнозувати курс долара на певний день. Для вирішення цього завдання необхідно скористатися функцією *evalfis*.

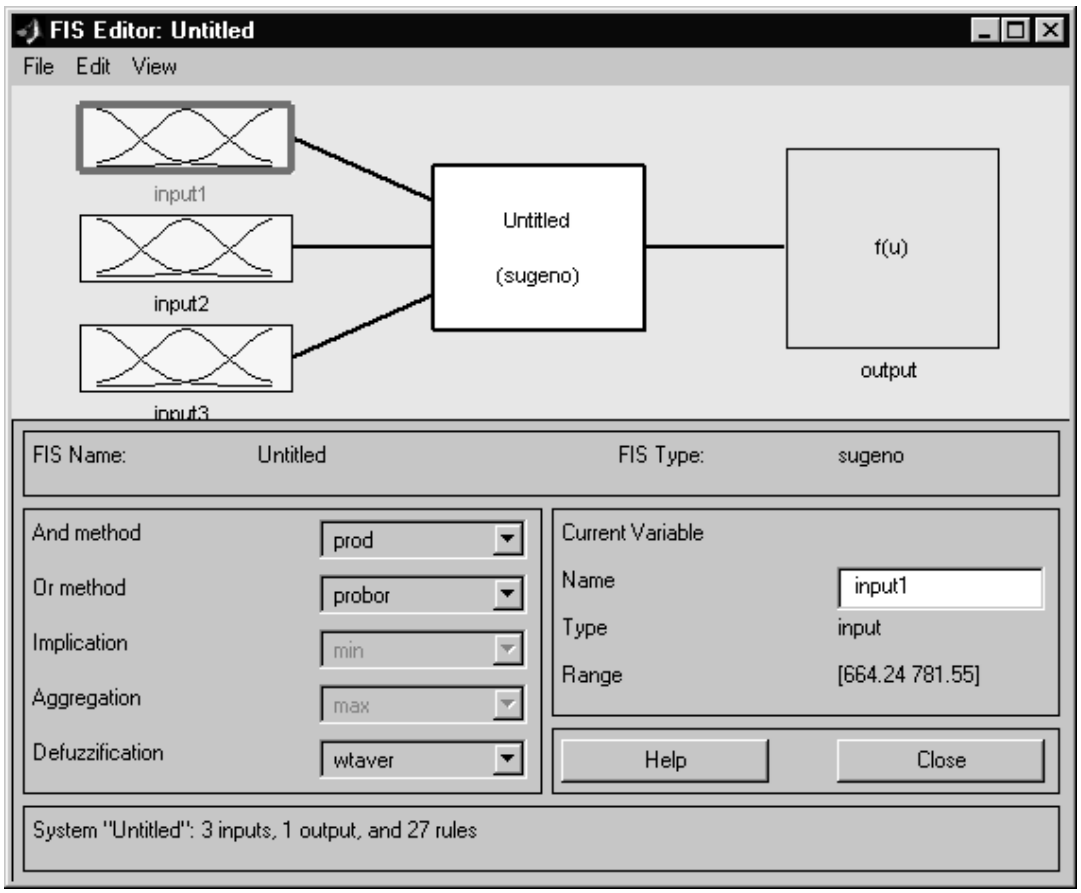


Рис. 9.11. Графічний інтерфейс редактора *FIS* для згенерованої системи нечіткого виводу

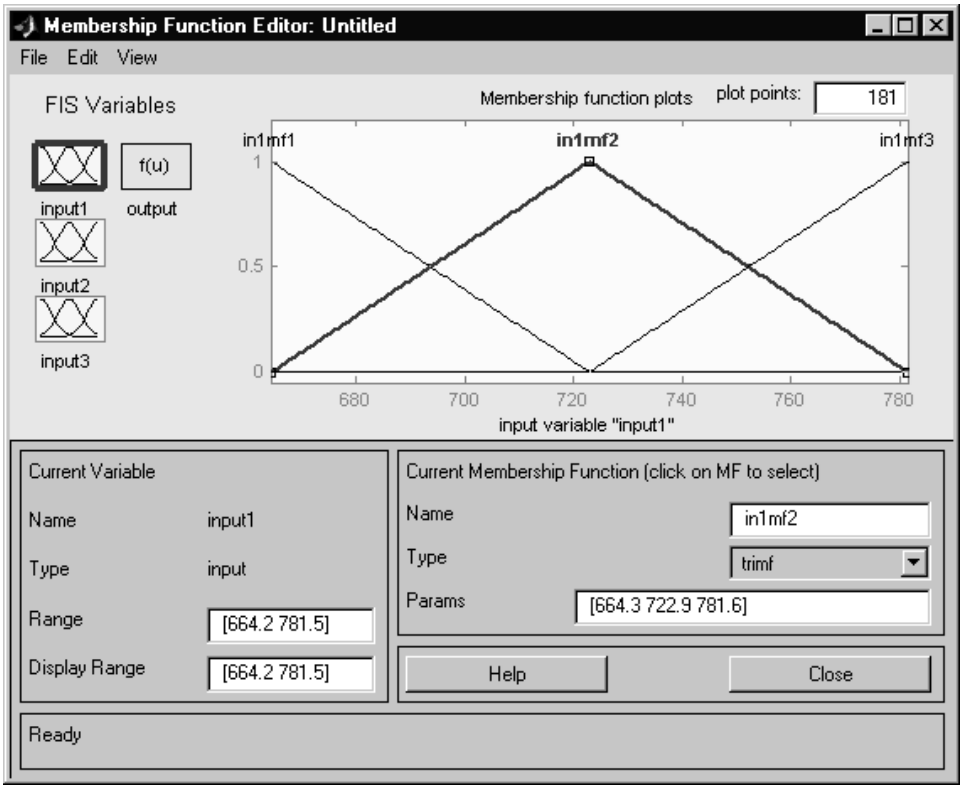


Рис. 9.12. Графічний інтерфейс редактора функцій належності побудованої системи нечіткого виводу

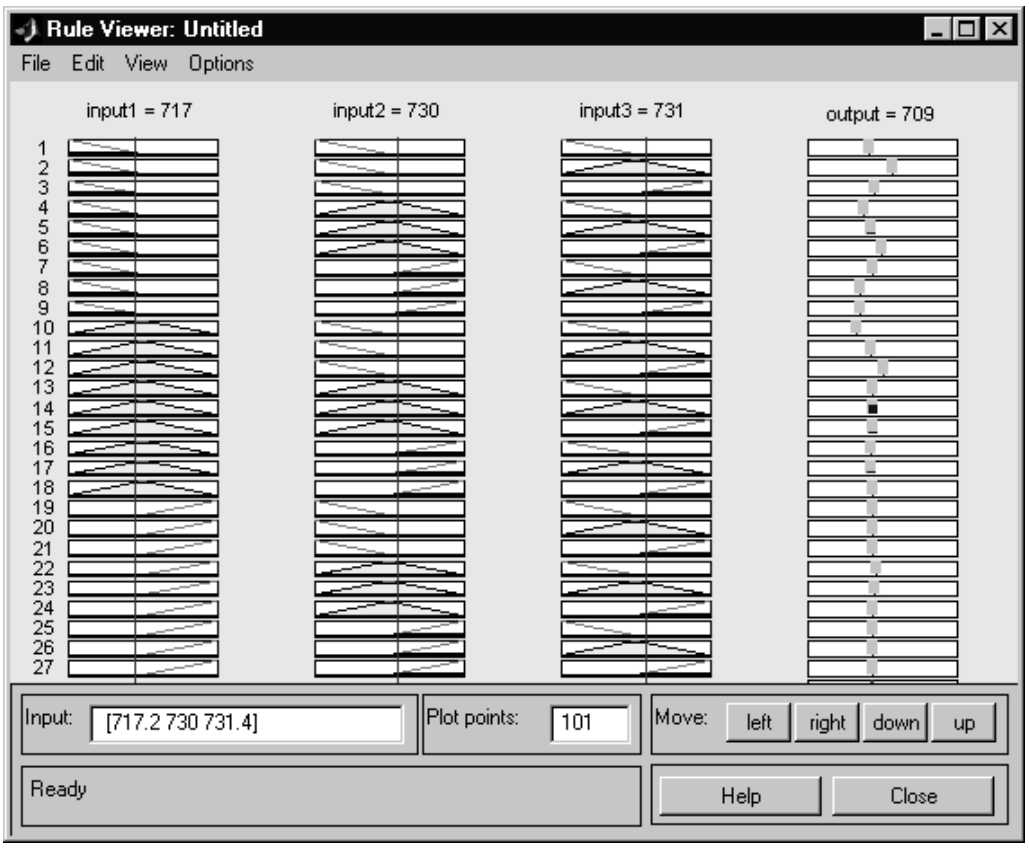


Рис. 9.13. Графічний інтерфейс перегляду правил згенерованої системи нечіткого виводу

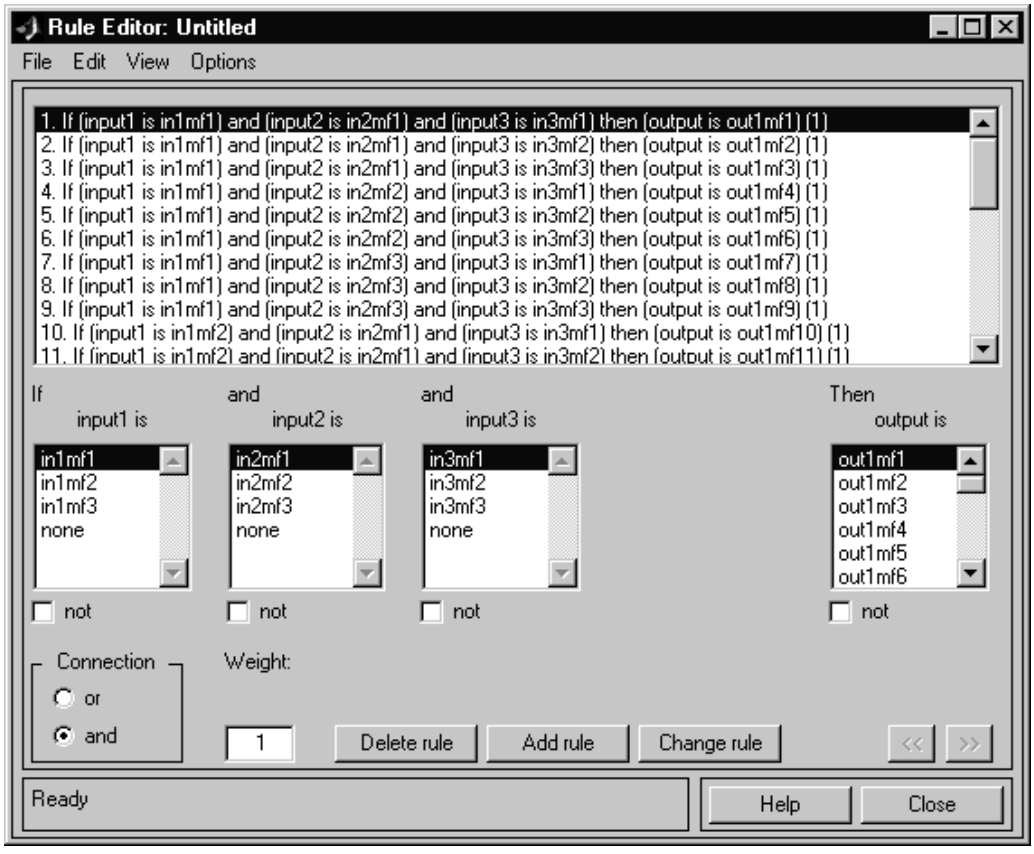


Рис. 9.14. Фрагмент бази нечітких правил

9.3. Індивідуальні завдання

1. Сформулювати завдання в галузі обчислювальної техніки, для вирішення якої було б обґрунтовано застосування гібридної нейронечіткої мережі.

2. Сформувати вибірку на навчання гібридної нейронної мережі.

3. Згенерувати та візуалізувати структуру гібридної нейронної мережі у системі MATLAB.

4. Навчити гібридну нейронну мережу, при цьому задати та обґрунтувати параметри її навчання.

5. Побудувати систему нечіткого виведення для отриманої гібридної нейронної мережі.

6. Виконати перевірку адекватності побудованої нечіткої моделі гібридної мережі.

7. Оформіть звіт з лабораторної роботи.

Лабораторна робота 10

РІШЕННЯ ЗАВДАННЯ ПРОГНОЗУВАННЯ ЗА ДОПОМОГОЮ НЕЧІТКИХ НЕЙРОННИХ МЕРЕЖ

Мета лабораторної роботи: отримання та закріплення знань, формування практичних навичок побудови нейронних мереж та гібридних нейронних мереж для прогнозування у пакеті MATLAB.

10.1. Короткі відомості з теорії

Нейронні мережі та нечітка логіка є універсальними апроксиматорами складних (нелінійних) функціональних залежностей у багатьох інтелектуальних задачах кібернетики: прогнозування, діагностики, розпізнавання образів та ін.

Перевагою нечіткої логіки є можливість використання експертних знань про структуру об'єкта як лінгвістичних висловлювань: якщо <входи>, то <вихід>. Однак апарат нечіткої логіки не містить механізмів навчання. Тому отримані з його допомогою результати сильно залежать від виду функцій приналежності, якими формалізуються нечіткі терми. Крім того, експерту необхідно визначити усі правила. Такий алгоритм багато в чому статичний. Необхідність його зміни спричинить досить трудомістку експертну процедуру.

Головною особливістю нейронних мереж є їхня здатність до навчання. Для навчання нейронної мережі не потрібно ніякої апріорної інформації про структуру функціональної залежності, що шукається. Потрібна лише навчальна вибірка у вигляді пар <входи – вихід>.

Замість визначення всіх функцій належності лінгвістичних змінних та правил необхідно створити навчальну вибірку достатнього обсягу. Одним з критеріїв "достатності" служить наступний: кількість навчальних пар

<входи - вихід> повинна перевищувати кількість параметрів мережі, що настроюються.

Перевага даного методу полягає у можливості самоналаштування параметрів мережі у процесі роботи.

Поєднання нечіткої логіки з нейронними мережами дає принципово нову якість. Отримана в результаті такого об'єднання нейронечітка мережа має дві найважливіші людські (інтелектуальні) властивості:

– лінгвістичність, тобто, використанням знань природною мовою;

– навчання в реальному масштабі часу.

Відповідно до аналізу, проведеного у ряді робіт, для класу завдань, яких застосовні нейронечеткие мережі, нейронні мережі під час навчання вимагають значно більше ітерацій. Оскільки будь-яка ітерація займає певний машинний час, то й загальний час навчання НМ більше.

Існує два основні різновиди нейронечітких мереж: Сугено-Такагі-Канга та Ванга-Менделя. Структура мережі Ванга-Менделя наведена на рис. 10.1.

Шар I складається з нейронів-фазифікаторів та виконує роздільну фазифікацію кожної змінної за функціями приналежності (*FP*). Виходи даного шару розраховуються за допомогою узагальненої функції Гауса:

$$FP[i \cdot M + j] = \exp\left[\left(-\frac{(x - C[i \cdot M + j])^2}{2D[i \cdot M + j]^2}\right)^B\right], \qquad (10.1)$$

представленої у раціональній формі:

$$FP[i \cdot M + j] = \frac{1}{1 + \left(\frac{X[i] - C[i \cdot M + j]}{D[i \cdot M + j]}\right)^{2B[i \cdot M + j]}}, \qquad (10.2)$$

де $i = \overline{0, N-1}$, $j = \overline{0, M-1}$; M – кількість функцій приладдя кожної змінної; N – число вхідних змінних; $C[iM + j]$ (зміщення); $D[iM + j]$ (масштаб); $B[iM + j]$ (форма) – це нелінійні параметри мережі, що описують функцію Гауса (підлягають налаштуванню під час навчання).

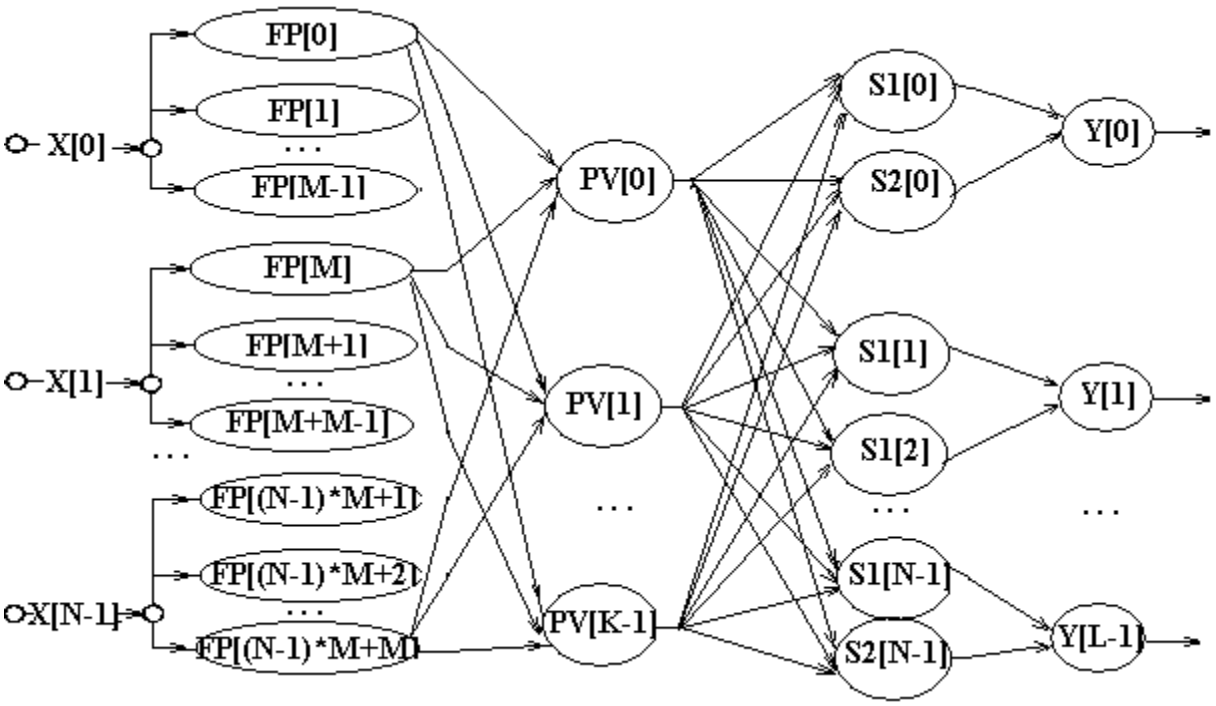


Рис. 10.1. Структура ННМ Ванга-Менделя

Шар II складається з нейронів-правил та визначає рівні активації правил виведення (PV). Виходи даного шару розраховуються за формулою:

$$PV[i] = \prod_{j=0}^{N-1} FP[j], \qquad (10.3)$$

де $i = \overline{0, K-1}$, а $K = M^N$.

Шар III складається з двох типів нейронів-суматорів, один з яких (S_1) здійснює агрегування правил виведення, а другий (S_2) – агрегування виважених правил виведення. Виходи нейронів розраховуються за формулами:

$$S_1[j] = \sum_{i=0}^{K-1} W[j, i] \cdot PV[i], \tag{10.4}$$

$$S_2[j] = \sum_{i=0}^{K-1} PV[i], \tag{10.5}$$

де $j = \overline{0, L-1}$, L – кількість виходів мережі; $W[j, i]$ – лінійні параметри мережі, що визначають вагові коефіцієнти застосовуваного правила (підлягають налаштуванню під час навчання).

Шар IV містить нейрони-нормалізатори та нормалізує вихідні змінні мережі Y. Його вихід розраховується за формулою:

$$Y[i] = S_1[i] / S_2[i], \tag{10.6}$$

де $i = \overline{0, K-1}$.

Ця мережа містить два параметричні шари – перший і третій. У першому шарі кожен нейрон зберігає три нелінійні параметри мережі C, D та B, які визначають функцію приналежності Гауса. У третьому шарі зберігаються лінійні параметри мережі W, що визначають вагові коефіцієнти правил та значення вихідних змінних.

10.2. Побудова нейронних мереж та гібридних нейронних мереж для прогнозування в пакеті MATLAB

Однією з основних можливостей нейронних мереж є здатність до екстраполяції. У процесі навчання на наявній статистиці (наприклад, про деякий час) НМ здатна знаходити залежності (в т.ч. приховані навіть для досвідченого експерта) між даними та ситуаціями, які в подальшому навченої НМ дозволяють будувати прогноз поведінки деякого функціоналу.

Зокрема, НМ досить широко застосовуються для прогнозування курсу валют на певний період. Наприклад, для прогнозування курсу на п'ятий банківський день за наявними даними про чотири попередні або прогноз на місяць за даними за півроку.

Для цих цілей можливе застосування як звичайних НМ прямого поширення, так і нечітких нейронних мереж.

10.2.1. Побудова нейронної мережі

Побудуємо нейронну мережу, яка за даними про курс валюти за чотири банківські дні передбачає курс на п'ятий день (екстраполює). Використовуються дані з 01.01.2012 по 10.03.2012 про курс долара США. Навчальні дані – з 01.01.2012 до 01.03.2012 р.

Створимо матрицю навчальних даних:

```
» obych=[31.5673  31.6053  31.6117  31.6113  31.6123
         31.6053  31.6117  31.6113  31.6123  31.6268
         31.6117  31.6113  31.6123  31.6268  31.6302
         31.6113  31.6123  31.6268  31.6302  31.6409
         ..........................................................
         31.8382  31.8423  31.8445  31.8424  31.844];
```

Створимо матрицю вхідних значень:

```
» vxod=obych(:,1:4);
```

Створимо матрицю очікуваних вихідних значень:

```
» vixod=obych(:,5);
```

Створимо нейронну мережу для прогнозування:

```
» prognoz=newff(minmax(vxod'),[15 1],{'logsig'
'purelin'});
```

Навчимо нейронну мережу:

```
» prognoz=train(prognoz,vxod',vixod');
```

Внаслідок навчання нейронної мережі отримаємо вікно, наведене на рис. 10.2.

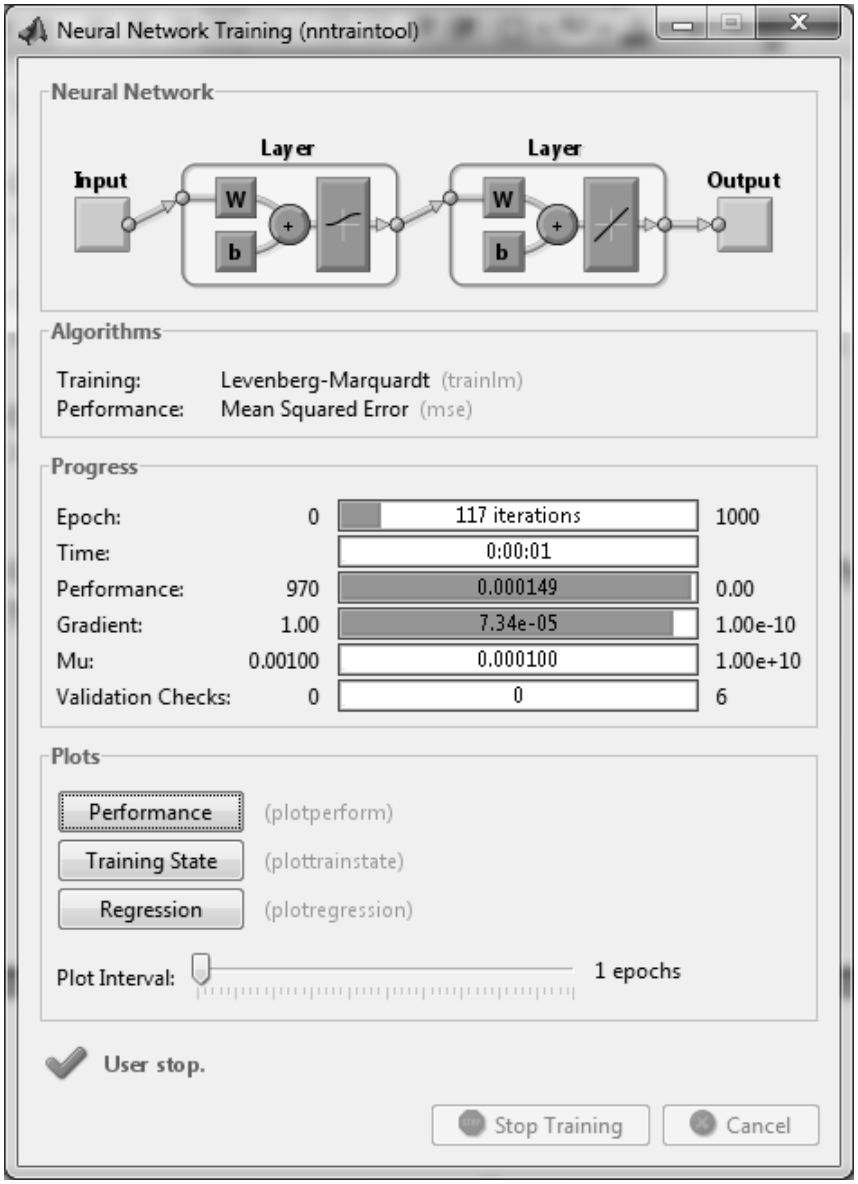


Рис. 10.2. Результати навчання НМ

Створимо тестову вибірку для перевірки результатів навчання:

```
test =
    31.8400  31.8400  31.8424  31.8547  31.8584
    31.8400  31.8424  31.8547  31.8584  31.8596
    31.8424  31.8547  31.8584  31.8596  31.8578
    31.8547  31.8584  31.8596  31.8578  31.8600
» test_vxod=test(:,1:4);
» test_vixod=test(:,5);
```

Проведемо моделювання навченої мережі на тестових даних:

```
» sim(prognoz,test_vxod)
ans =
    31.8455  31.8467  31.8467  31.8463
» test_vixod
test_vixod=
    31.8584
    31.8596
    31.8578
    31.8600
```

Результати моделювання та справжні значення курсу валют відрізняються достатньою мірою.

10.2.2. Побудова гібридної нейронної мережі

Побудуємо гібридну нейронну мережу, яка за даними про курс валюти за чотири банківські дні передбачає курс на п'ятий день (екстраполює). Використовуються дані з 01.01.2022 р. по 10.03.2022 р. про курс долара

США. Навчальні дані – з 01.01.2022 р. до 01.03.2022 р., тестуючі – з 01.01.2022 р. до 09.01.2022 р., перевірочні – на 10.01.2022 р.

Створимо файли даних: *dat: training.gat, testing.dat, cheking.dat.* Вони будуть міститися дані, необхідні навчання і перевірки нейронної мережі. Дані в них – матриці, у кожній з яких по 5 стовпців – 4 банківські дні (вхід) та 1 день (вихід). У першому файлі – 40 рядків, у другому – 4, у третьому – 1.

Файл *Training.dat*

```
31.5673  31.6053  31.6117  31.6113  31.6123
31.6053  31.6117  31.6113  31.6123  31.6268
31.6117  31.6113  31.6123  31.6268  31.6302
31.6113  31.6123  31.6268  31.6302  31.6409
.............................................................
31.8382  31.8423  31.8445  31.8424  31.8424
```

Файл *Testing.dat*

```
31.8422  31.8445  31.8424  31.8547  31.8584
31.8423  31.8424  31.8547  31.8584  31.8596
31.8424  31.8547  31.8584  31.8596  31.8578
31.8547  31.8584  31.8596  31.8578  31.8634
```

Файл *Cheking.dat*

```
31.8584  31.8596  31.8578  31.8623  31.8597
```

Для побудови нечітких нейронних мереж у MATLAB використовується редактор *Anfis Editor* (рис. 10.3).

Запустимо редактор командою *anfisedit*.

В меню *Load data* слід вибрати *Training* та *From disk*, натиснути кнопку *load data*. У вікні, що з’явилось, слід вибрати створений раніше *training.dat*.

В меню *Load data* слід вибрати *Testing* та *From disk*, натиснути кнопку *load data*. У вікні, що з’явилось, слід вибрати створений раніше *testing.dat*.

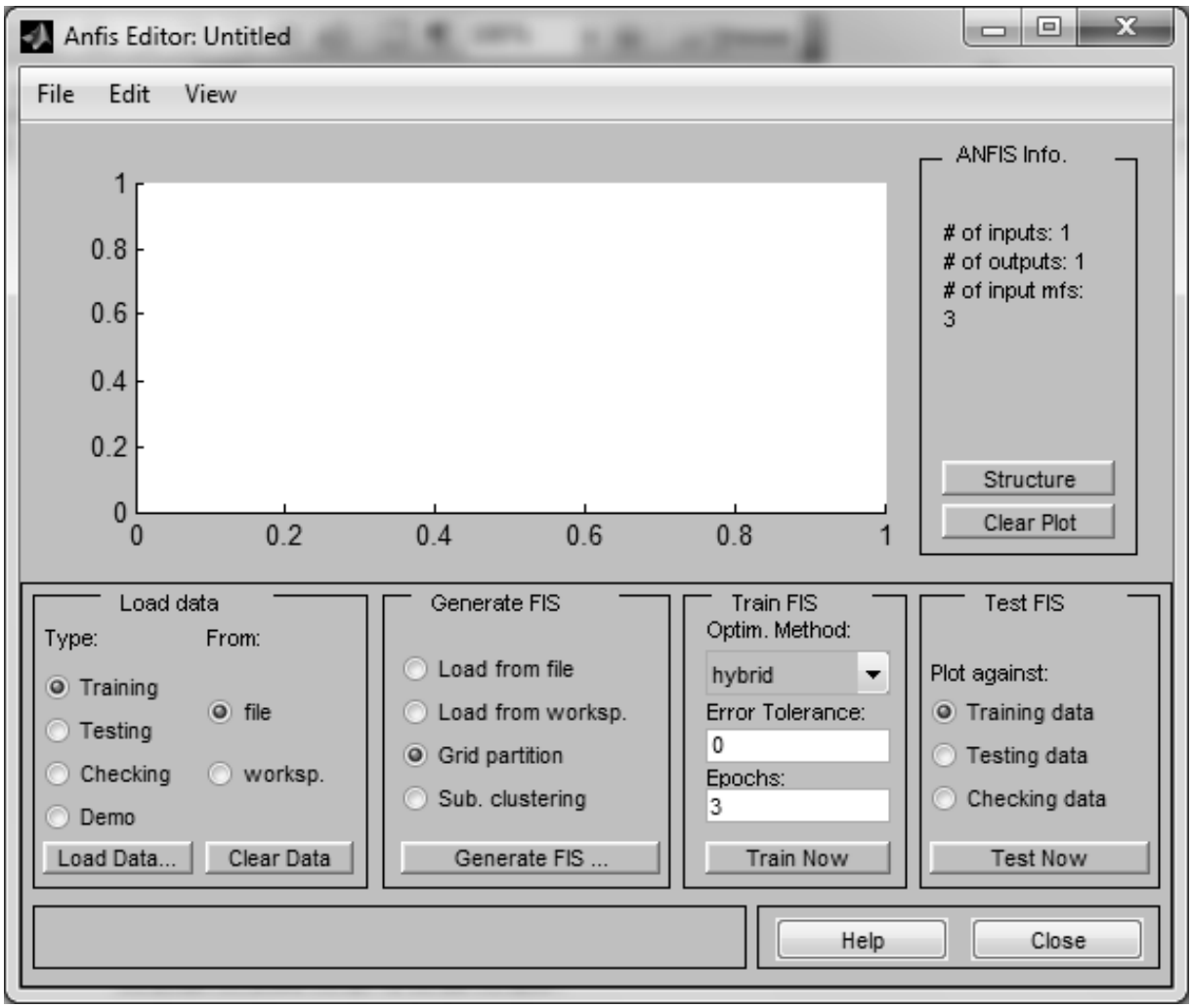


Рис. 10.3. Головне вікно *Anfis editor*

В меню *Load data* слід вибрати *Cheking* и *From disk*, натиснути кнопку *load data.* У вікні, що з'явилось, слід вибрати створений раніше *cheking.dat.*

Дані для навчання та перевірки завантажені (рис. 10.4).

Далі, встановивши перемикач меню *Generate FIS* у положення *Grid partition*, слід натиснути кнопку *Generate FIS* (рис. 10.5).

В даному випадку в моделі 4 вхідних змінних, кожній з яких відповідають по 3 терми типу *gaussmf*. Вихідна змінна задається лінійною функцією.

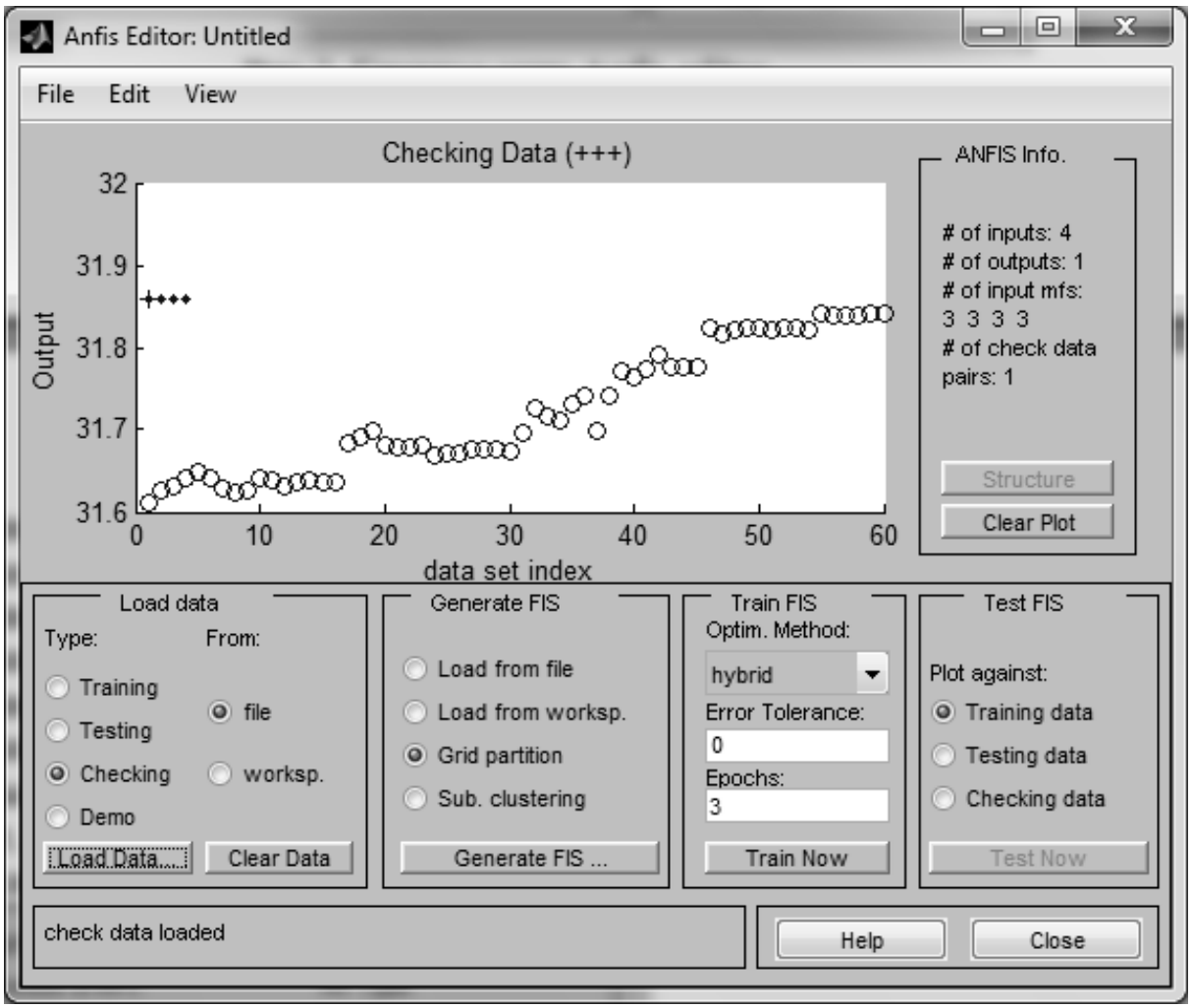


Рис. 10.4. Вигляд головного вікна редактора після завантаження навчальних даних

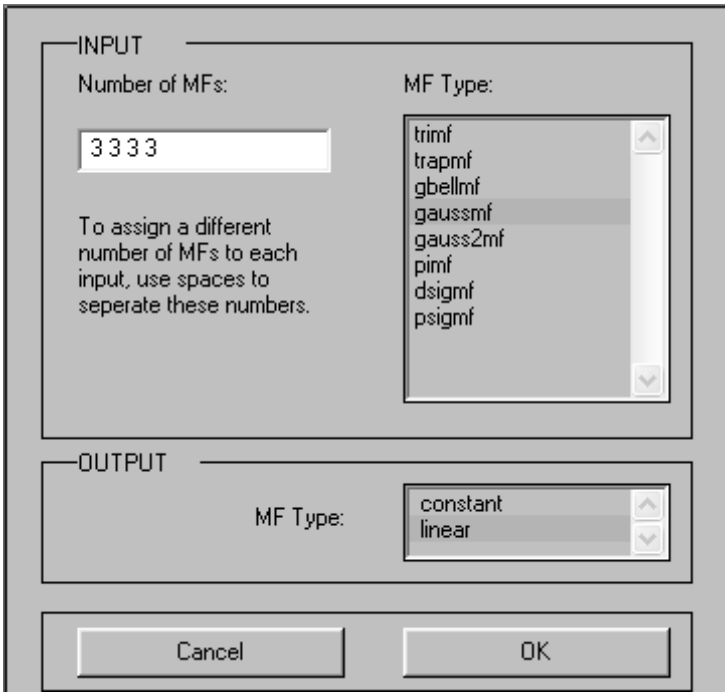


Рис. 10.5. Параметри генерованої мережі

Натиснувши кнопку *Structure* меню *Anfis info*, можна побачити структуру мережі (рис. 10.6).

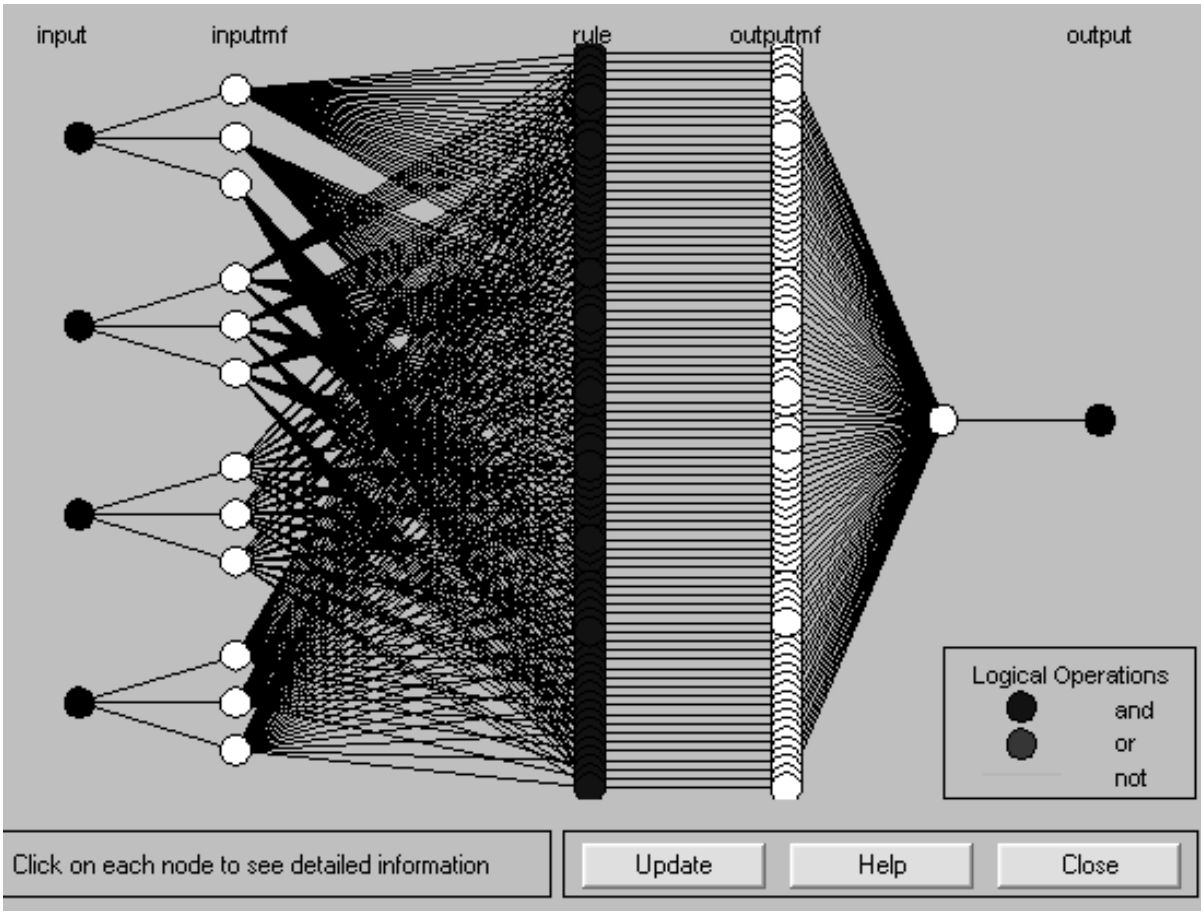


Рис. 10.6. Структура нечіткої нейронної мережі

Далі слід вибрати гібридний метод навчання, потрібну помилку навчання – 0 та кількість циклів навчання – 10, як показано на рис. 10.7. Потім слід натиснути кнопку *Train Now*.

Помилка навчання встановилася на рівні 0.00010501.

Проведемо тестування мережі. Для цього у розділі Test FIS головного вікна слід вибрати відповідну вибірку та натиснути кнопку Nest Now. У результаті для повчальних даних помилка встановилася на рівні 0.00014247 (рис. 10.8), для тестуючих даних помилка встановилася на рівні 0.020429 (рис. 10.9), а для перевірочних (контрольних) даних помилка встановилася на рівні 0.002094 (рис. 10.10).

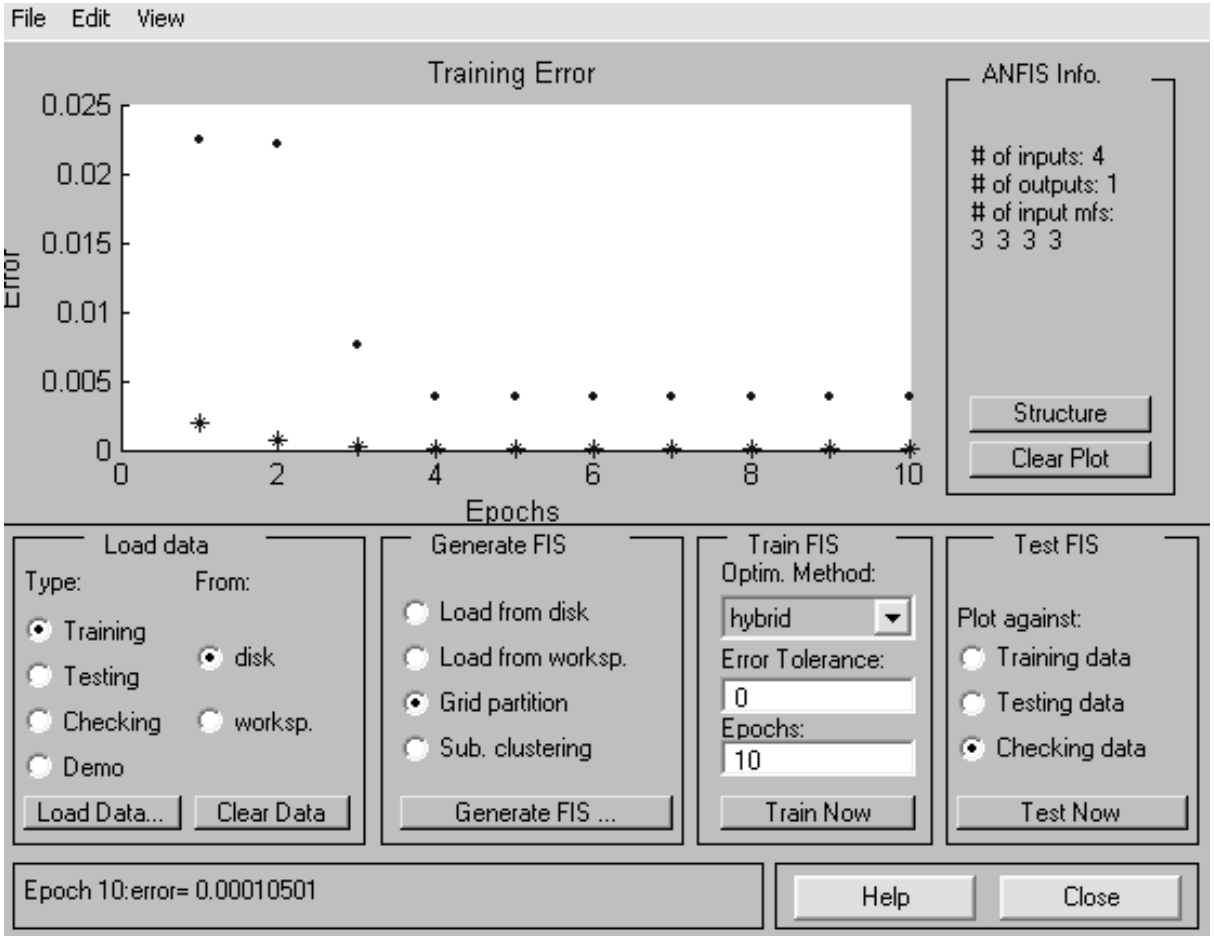


Рис. 10.7. Результати навчання ННМ

За допомогою команди *FIS Properties* меню *Edit* можна переглянути отриману нечітку нейронну мережу як систему нечіткого логічного висновку.

Далі експортуємо результати до робочої області *Export->To workspace*.

Скористайтеся командою *evalfis* для точного визначення значення прогнозу:

```
» out=evalfis([31.8584 31.8596 31.8578 31.86], anfis)
out= 31.8568
```

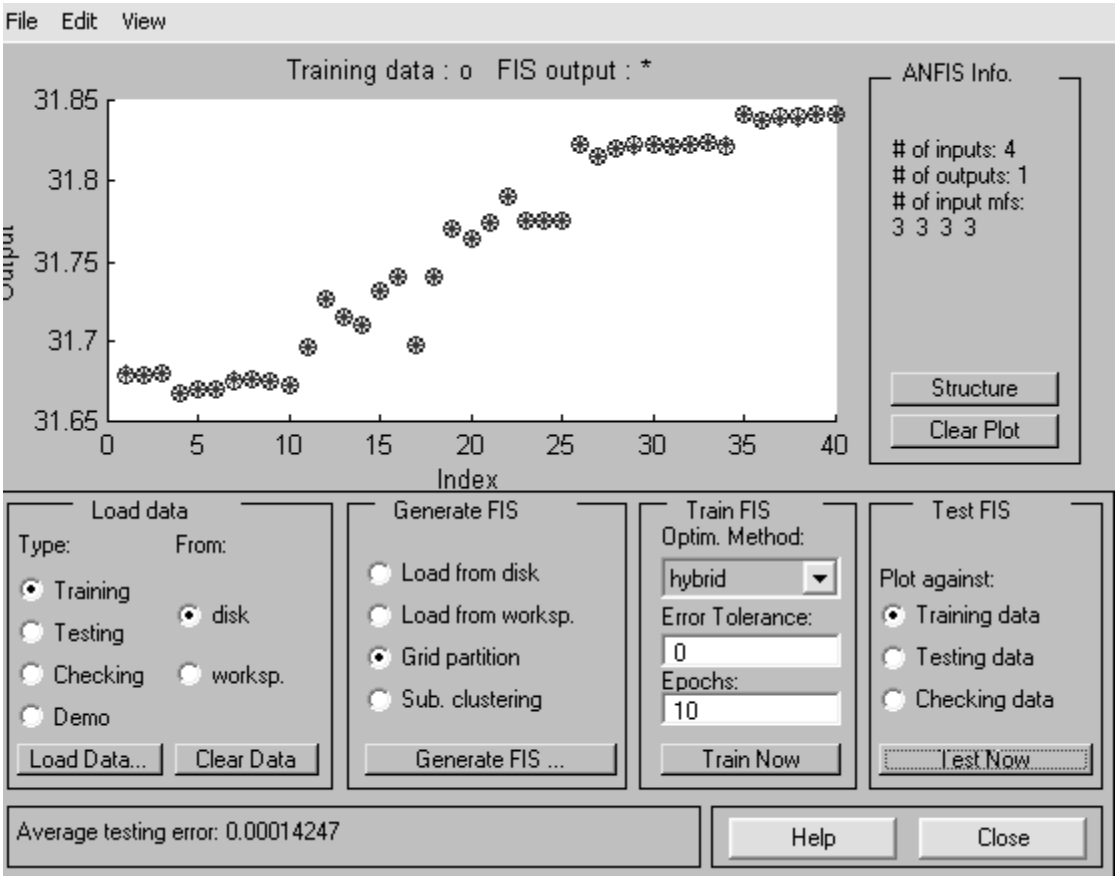


Рис. 10.8. Результати моделювання ННМ для навчальних даних

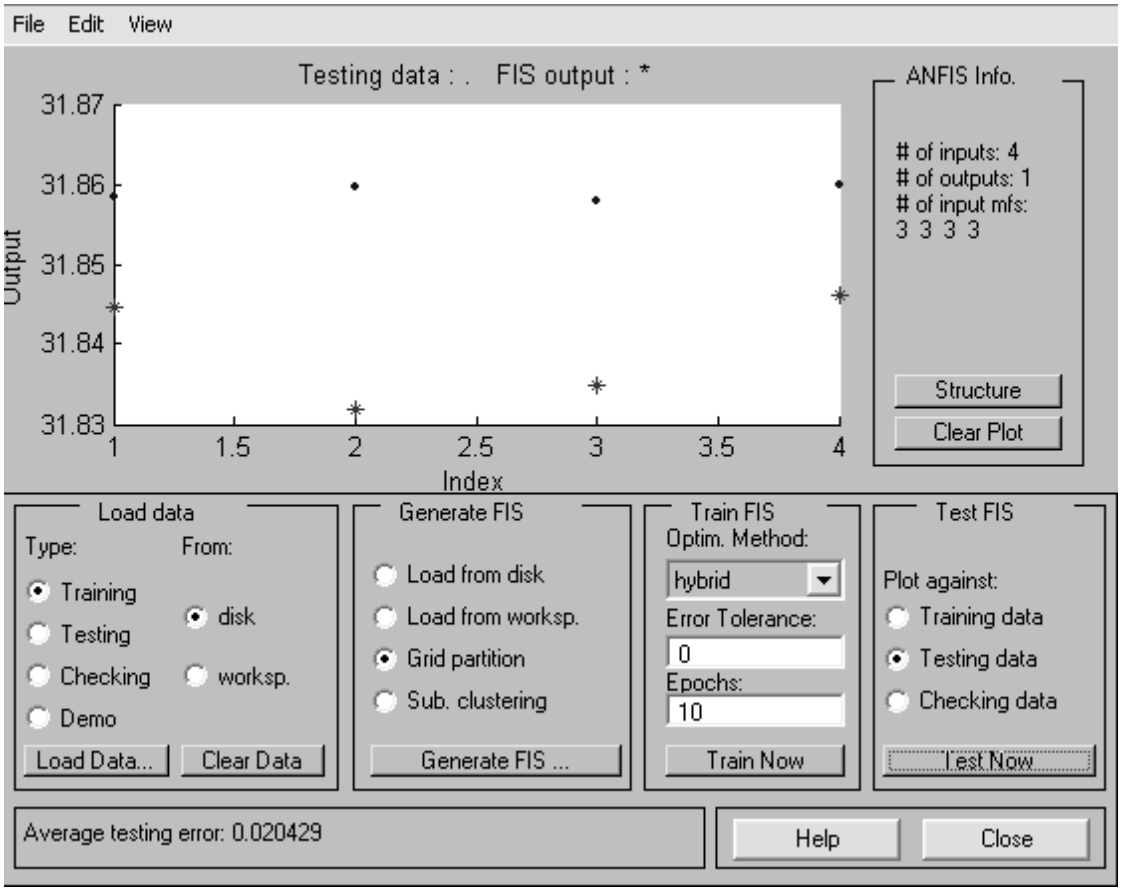


Рис. 10.9. Результати моделювання ННМ для тестуючих даних

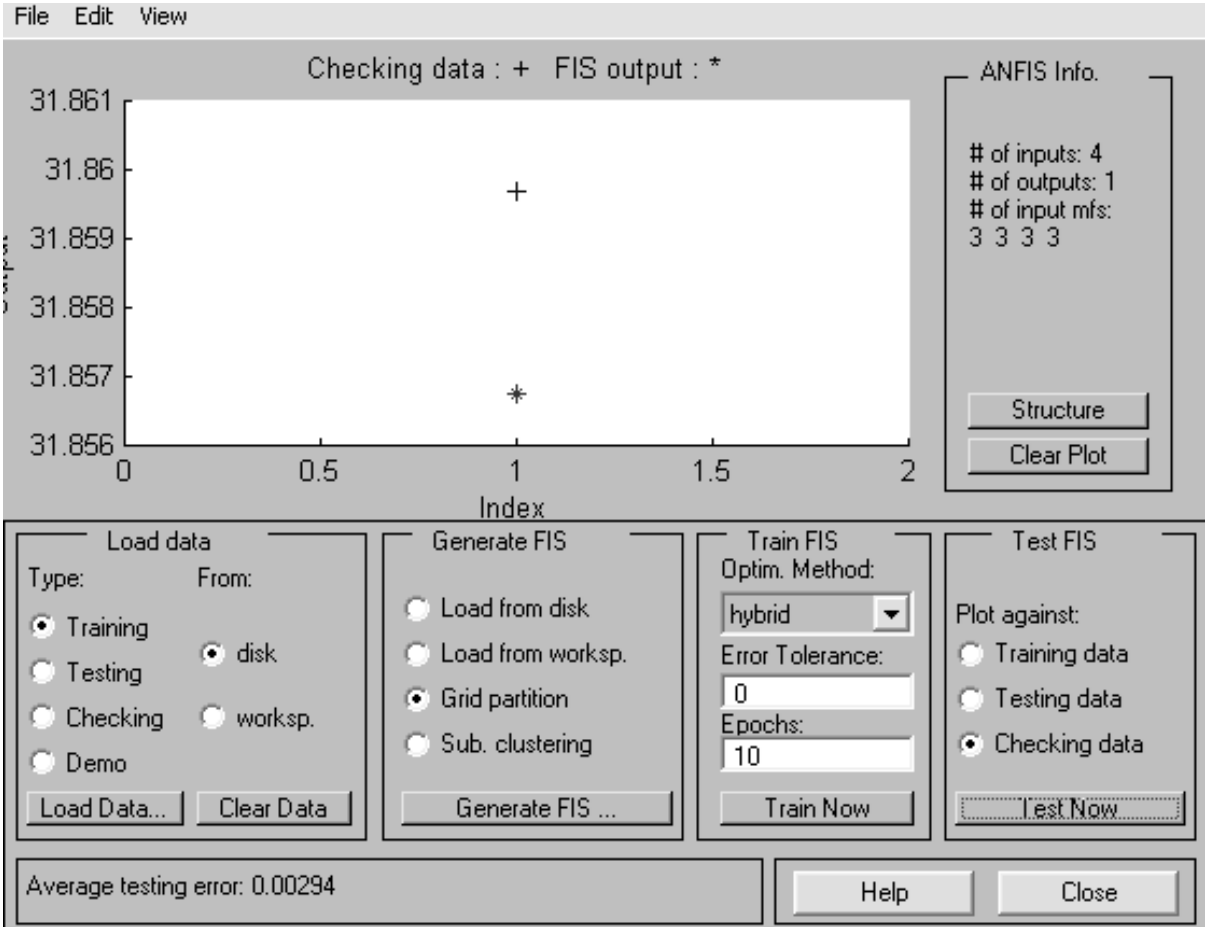


Рис. 10.10. Результати моделювання ННМ для контрольних даних

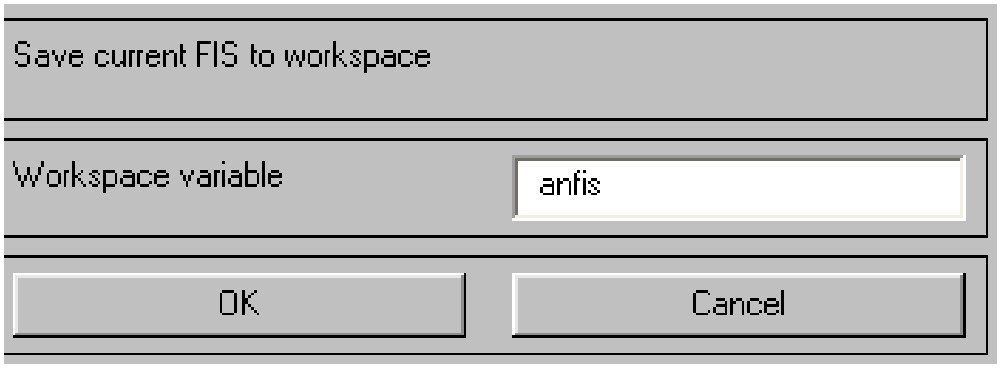


Рис. 10.11. Експорт результатів

Справжнє значення прогнозу дорівнює 31.8597.

У цьому випадку, порівняно із звичайною нейронною мережею прямого поширення, отриманий результат ближчий до реального значення курсу валют.

10.3. Індивідуальне завдання

1. Сформулювати завдання прогнозування в галузі обчислювальної техніки, для вирішення якої було б обґрунтовано застосування НМ та ННМ.

2. Сформувати навчальну вибірку для НМ та ННМ.

3. Побудувати НМ і, експериментуючи з кількістю нейронів у вхідних і прихованих шарах, функціями активації, методами навчання, досягти найкращого результату прогнозування.

4. Візуалізувати отриману структуру НМ у системі MATLAB.

5. Побудувати ННМ і, експериментуючи з методами навчання, кількістю функцій власності у вхідному шарі, досягти найкращого результату прогнозування. Кількість входів взяти так само, як і в п. 3 індивідуального завдання.

6. Візуалізувати отриману структуру ННМ у системі MATLAB.

7. Побудувати систему нечіткого логічного висновку для отриманої ННМ.

8. Порівняти отримані за допомогою НМ та ННМ результати (чисельно) та зробити висновки.

9. Оформіть звіт з лабораторної роботи.

ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТІВ

Звіти оформлюються українською мовою у редакторі *Microsoft Word* шрифтом *Times New Roman* без перенесення, розмір 14 pt. Міжрядковий інтервал – 1,5 рядки, абзацний відступ – 1,25 см. Обсяг звіту – не більше ніж 20 друкованих сторінок. Звіт готується на аркушах формату А4, орієнтація – книжкова. Поля: верхнє, нижнє та праве – по 2,0 см, ліве – 3,0 см. Розміри колонтитулів: верхнього – 0 см, нижнього – 2,0 см. Нумерація виконується внизу сторінки, вирівнювання по центру.

Перший рядок містить: прізвище, ініціали студента, номер академічної групи та варіанту (шрифт напівжирний, курсив, вирівнювання праворуч).

Другий рядок: друкується номер роботи великими літерами (шрифт – прямий, напівжирний, вирівнювання по центру).

Третій рядок: тема роботи великими літерами через один порожній рядок після номера роботи (шрифт – прямий, жирний, вирівнювання по центру).

Потім друкується мета роботи через один порожній рядок після теми роботи (відступ 1,25 пт, вирівнювання – по ширині).

Далі друкуються результати виконання всіх пунктів індивідуального завдання (абзацний відступ – 1,25 см, вирівнювання по ширині).

Наприкінці наводяться висновки щодо роботи. Вони друкуються через один порожній рядок після результатів виконання роботи (відступ 1,25 пт, вирівнювання – по ширині).

СПИСОК ЛІТЕРАТУРИ

1. Fuzzy Information Processing // *Kelly Cohen, Nicholas Ernest, Barnabas Bede, Vladik Kreinovich* //·Springer, 2023. 358 p.

2. Recent Trends on Type-2 Fuzzy Logic Systems: Theory, Methodology and Applications / *Anupam Kumar, Oscar Castillo* // Springer International Publishing, 2023. 265 p.

3. Fuzzy Logic Applications in Computer Science and Mathematics / *Ashok Kumar Shaw, Biswadip Basu Mallik, Dac-Nhuong Le, Gunjan Mukherjee, Rahul Kar* // Wiley, 2023. 304 p.

4. Fuzzy Sets, Fuzzy Logic and Their Applications 2021 / *Michael Voskoglou* // MDPI AG, 2023. 282 p.

5. Computational Methods with MATLAB / *Erik Cuevas, Alberto Luque, Héctor Escobar* // Springer Nature Switzerland, 2023. 212 p.

6. Fundamentals of Computational Intelligence / *O. Zakovorotniy, O. Lipchanska* // Laboratory workshop. Part 1. Kharkiv: NTU "KhPI", 2022. 160 p.

7. Fundamentals of Computational Intelligence / *O. Zakovorotniy, O. Lipchanska* // Laboratory workshop. Part 2. Kharkiv: NTU "KhPI", 2022. 152 p.

8. Neural Networks and Learning Algorithms in MATLAB / *Ardashir Mohammadazadeh, Mohammad Hosein Sabzalian, Oscar Castillo, Rathinasamy Sakthivel, Fayez F. M. El-Sousy, Saleh Mobayen* // Springer International Publishing, 2022. 117 p.

9. Genetic Algorithms / *Jose M. Moyano, José Luna, Sebastian Ventura* // IntechOpen, 2022. 176 p.

10. Computational Methods Using MATLAB / *P.K. Thiruvikraman* // IOP Publishing, 2022. 300 p.

11. Fuzzy Logic. Theory and Applications / *Constantin Volosencu* // IntechOpen, 2022. 280 p.

12. Dynamic System Modelling and Analysis with MATLAB and Python: For Control Engineers / *Jongrae Kim* // Wiley, 2022. 336 p.

13. Control Systems: An Introduction / *D. Sundararajan* // Springer International Publishing, 2022. 312 p.

14. Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control / *Steven L. Brunton, J. Nathan Kutz* // Cambridge University Press, 2022. 590 p.

15. Beginning MATLAB and Simulink: From Beginner to Pro / *Sulaymon Eshkabilov* // Apress, 2022. 605 p.

16. Learning Genetic Algorithms with Python. Empower the Performance of Machine Learning and AI Models with the Capabilities of a Powerful Search Algorithm / *Ivan Gridin* // BPB publications, 2021. 270 p.

17. Introduction to Fuzzy Logic / *James K. Peckol* // Wiley, 2021. 304 p.

18. Genetic Algorithms in Elixir / *Sean Moriarity* //·Pragmatic Bookshelf, 2021. 244 p.

19. DNA Computing Based Genetic Algorithm. Applications in Industrial Process Modeling and Control / *Jili Tao, Ridong Zhang, Yong Zhu* //·Springer Nature Singapore, 2021. 274 p.

20. Fuzzy Systems and Data Mining VII. Proceedings of FSDM 2021 / *C. Shen* // IOS Press, 2021. 492 p.

Навчальне видання

ЗАКОВОРОТНИЙ Олександр Юрійович
ОРЛОВА Тетяна Олександрівна
ГРИНЬОВ Денис Валерійович
СЕРГІЄНКО Віталіна Миколаївна

ОСНОВИ ОБЧИСЛЮВАЛЬНОГО ІНТЕЛЕКТУ

Лабораторний практикум
для студентів денної та заочної форм навчання за спеціальністю
123 "Комп’ютерна інженерія"

Роботу до видання рекомендував *М. Й. Заполовський*
Відповідальний за випуск *С. Ю. Леонов*
У авторській редакції.

План 2022 р.

Підп. до друку 25.01.2023. Гарнітура Times New Roman. Обл.-вид. арк. 4,6

Видавничий центр НТУ «ХПІ»
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2.