ЗАТВЕРДЖЕНО
Науково-методичною радою
Державного університету
«Житомирська політехніка»
протокол від 21 травня
2021 р. №3

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
для проведення лабораторних занять
з навчальної дисципліни

«ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНІ СИСТЕМИ»

для здобувачів вищої освіти освітнього ступеня «магістр»
спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка»
освітньо-професійна програма «Комп'ютеризовані інформаційно-вимірювальні системи»
факультет комп'ютерно-інтегрованих технологій, мехатроніки і робототехніки
кафедра метрології та інформаційно-вимірювальної техніки

Розглянуто і рекомендовано на
засіданні кафедри метрології
та інформаційно-вимірювальної
техніки
протокол 05.03.2021 р.,
№ 3

Укладач: доцент кафедри метрології та інформаційно-вимірювальної техніки к.т.н., ЧЕПЮК Ларіна.

Житомир
2021

ЗМІСТ

§1. ОСНОВНІ МОДЕЛІ ПРЕДСТАВЛЕННЯ ЗНАНЬ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ

1.1. Дані та знання

Для розробки систем штучного інтелекту - інтелектуальних систем (експертних систем, систем підтримки прийняття рішень (СППР) та інших) використовуються данні та знання предметної області для розв'язку прикладних інтелектуальних задач.

Дані — це окремі факти, що характеризують об’єкт, процеси та явища предметної галузі, а також їх властивості. При розробці, проектуванні інтелектуальних систем данні проходять етапи трансформації від більш узагальнених множин до біль вузьких, конкретизованих множин, необхідних для розв'язку прикладних задач :

1. D_1 – данні як результат вимірювань та спостережень.

2. D_2 – данні на матеріальних носіях інформації (таблиці, протоколи, довідники).

3. D_3 – моделі (структури) даних у вигляді діаграм, графіків, функцій.

4. D_4 – данні в комп’ютері на мові опису даних.

5. D_5 – бази даних на комп’ютеризованих (електронних) носіях інформації.

Знання основані на даних, що отримані емпіричним шляхом. Вони є результат інтелектуальної діяльності людини, що спрямована на узагальнення його досвіду, отриманого під час практичної діяльності та професійного досвіду.

Знання — це закономірності предметної галузі (принципи, зв’язки, закони), отримані під час практичної діяльності та професійного досвіду людини, що дозволяють фахівцям формулювати та вирішувати завдання в цієї галузі.

В даний час у штучному інтелекті не існує строго формалізованого визначення поняття «знання». Більшість фахівців, розробників інтелектуальних систем використовують таке визначення: **знання** — це добре структурована інформація, що зберігається в системі і містить усі відомості про предметну область та правила виводу, що необхідні для розв’язку безлічі завдань інтелектуальної системи. Під час проектування та розробки інтелектуальної системи знання проходять аналогічну даним трансформацію - від більш узагальнених множин до більш вузьких, конкретизованих для даної предметної області:

1. R_1 – знання в пам’яті людини як результат мислення.

2. R_2 – знання на матеріальних носіях інформації (довідники, підручники та ін.).

3. R_3 – поле знань – умовний опис основних об'єктів предметної області, їх атрибутів та закономірностей, що їх пов'язують.

4. R_4 – формалізовані знання, що описані на специфічній мові представлення знань (продукційні моделі, семантичні мережі, фрейми, логічні моделі та ін.).

5. R_5 – бази знань, як елемент прикладної інтелектуальної системи на комп'ютеризованих (електронних) носіях інформації.

Для зберігання даних використовуються бази даних, для зберігання знань – відповідно бази знань. База знань – сукупність знань, представлених за допомогою стандартизованих моделей на визначених мовах програмування, які складають основу будь якої інтелектуальної системи.

С точки зору використання знань для розв'язку прикладних задач, то їх можна розділити на дві категорії: **точні** знання та **евристики**. Перша категорія — це знання, що отримані зі спеціальної літератури, підручників, довідників і т.п. Друга категорія — це досвід фахівців в даній предметній області, що накопичений у результаті багаторічної практики. Друга категорія знань має велике значення для підвищення ефективності системи, оскільки дозволяє використовувати у системах суб'єктивні заключення експертів, їх оцінки різних альтернатив з предметної області та ін. Для формалізації представлення евристичних знань та їх обробки у інтелектуальних системах використовуються математичні методи, що основані на теорії нечітких множин. Цей напрям розвитку систем штучного інтелекту є найбільш актуальним у сучасних методах проектування баз знань. Тому, далі у посібнику ми, як правило, розглядатиме методи та моделі проектування прикладних інтелектуальних систем, що основані на математичних теоріях нечітких множин, нечіткої логіки висловлювань та ін..

Розглянемо особливості знань, що відрізняють їх від даних. Ці особливості включають такі властивості.

1) **Інтерпретованість.** У традиційних обчислювальних програмах дані, що зберігаються у пам'яті комп'ютера, можуть бути інтерпретовані тільки відповідною програмою (специфіка формату даних). Дані без програми не несуть ніякої інформації. Прагнення до того, щоб комп'ютер був спроможній інтерпретувати вміст своєї пам'яті в поняттях, зв'язаних із семантикою розв'язуваної задачі у системах штучного інтелекту, привело до відокремлення знань від даних. Для того щоб система могла "знати", що являє собою та чи інша інформаційна одиниця, що зберігається в її пам'яті, необхідно забезпечити ці одиниці пояснювальними описами. Ці описи передбачають присвоєння даній одиниці знань системи імен. Система включає індивідуальне ім'я даної інформаційної одиниці плюс пос-

лідовність імен тих множин чи класів, до яких ця одиниця входить. Ця властивість притаманна реляційним базам даних. Наявність цих надлишкових імен дозволяє системі знати, що зберігається в її БЗ.

2) **Рекурсивна структурованість.** Для знань виконується «принцип матрьошки», тобто рекурсивне вкладення одних інформаційних одиниць в інші. Між певними одиницями знань можна установити такі відносини, як «елемент — множина», «тип — підтип», «ситуація — підситуація», що відбивають характер їхнього взаємозв'язку. Це дозволяє ефективно використовувати принцип наслідування, тобто в одному екземплярі зберігати інформацію, однакову для елементів класу. Це дає також можливість робити агрегування і декомпозицію інформаційних одиниць через відносини «елемент — множина», «тип — підтип», «рід — вид» і ін.

3) **Зв'язаність.** У БЗ між інформаційними одиницями встановлюються зв'язки різного типу, що характеризують відносини між об'єктами, явищами і т.п. Наприклад, зв'язки типу «причина — наслідок», «бути справедливим для», «сприяти» і т.п. Зв'язки дозволяють будувати процедури аналізу знань на сумісність, несуперечність і т.п., що важко реалізувати при збереженні традиційних масивів даних.

4) **Активність.** Поділ інформаційних одиниць, прийнятий у традиційному програмуванні, на дані і програми (команди) привело до того, що дані пасивні, а команди активні. У системах, заснованих на знаннях, знання можуть ініціювати дії. Поява в інформаційній базі нових фактів, установлення нових зв'язків може стати джерелом активності системи. Знання, що представлені у інтелектуальній системі - це деяка модель, формалізоване представлення предметної області. У той же час дані не виконують задачу змістовного моделювання.

Відокремлюють три типи знань у інтелектуальних системах:

- фактуальні (декларативні) знання (понятійна чи об'єктна модель предметної області). Одиниця знань — *факт* (без явної вказівки, як і коли цей факт використовується).
- процедурні знання — функціональна модель предметної області. Одиниця знань — *правило* (дає спосіб обробки знань).
- керуючі знання — стратегія виводу, порядок застосування правил (процедури логічного виводу).

1.2. Моделі представлення знань

Однією з найбільш важливих проблем, характерних для систем, заснованих на знаннях, є проблема представлення знань. Це пояснюється тим, що форма представлення знань впливає на характеристики і властивості системи. Центральним питанням побудови систем, заснованих на знаннях, є вибір форми (моделі, мови, методу) представлення знань.

Представлення знань — це безліч синтаксичних і семантичних угод, що роблять можливим формальне вираження знань про ПО в комп'ютерно - інтерпретований формі. Найбільш розповсюдженими є моделі представлення знань, що відображені на рис.1.1.

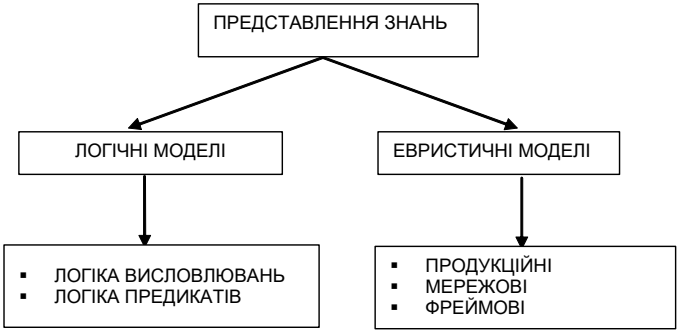

Рис.1.1. Моделі представлення знань.

Оскільки логічні моделі складають основу учбових курсів з математичної логіки і виходять за межи даного посібника, то далі будемо розглядати тільки евристичні моделі представлення знань.

1.3. Продукційні системи представлення знань

Продукційна система, або система правил продукцій – деяка узгоджена безліч окремих продукцій вигляду:

$$(i): Q;P;A \Rightarrow B;N$$

де (i) - ім'я продукції; Q - сфера застосування продукції; P - умова застосування ядра продукції; $A \Rightarrow B$ - ядро продукції, у якому A - умова ядра (антецедент), B - заключення ядра (консеквент); $\Rightarrow$ - знак логічної секвенції ("ЯКЩО A ТО B"); N - після умова продукції.

Ім'я продукції (i) - унікальний ідентифікатор, що надається сукупністю букв або дозволених мовою представлення знань символів та дозволяє єдиним образом визначати продукцію у системі. Найбільш часто продукція задається за допомогою ідентифікаційного номеру.

Умова застосування ядра продукції P - логічний вираз (як правило, предикат), за допомогою якого активізується ядро продукції : якщо P - істина ядро активізується, у протилежному випадку – ні. У багатьох випадках P відсутня у продукції, або об'єднується з ядром продукції.

Ядро продукції $A \Rightarrow B$ – центральний компонент продукції. Як правило, ядро продукції має вигляд речення-правила "ЯКЩО A ТО B", де A, B - деякі логічні вирази. Знак логічної секвенції має зміст логічного випливання B із істинного A. Якщо A не істинно, то про істинність B не можна зробити ніяких висновків. У базах знань інтелектуальних систем умова ядра A виступає також як деяке речення-зразок, логічний вираз, за яким здійснюється пошук у базі знань. Заключення ядра B виступає як дія, процедура, яка виконується при успішному завершенні пошуку.

Після умова продукції N містить опис процедур, які необхідно виконати у разі реалізації ядра продукції (тобто при істинності B). В нечітких продукційних системах представлення знань кожне з правил продукцій може додатково мати параметризовану кількісну оцінку ступеня істиності правила, яка формально знаходиться в N. Приклади побудови нечітких продукційних систем будемо розглядати далі у наступних розділах за допомогою пакету програм Fuzzy Logic Toolbox середовища MATLAB.

1.4. Семантичні мережі

Семантична мережа – це орієнтований граф, вершини якого – поняття, а дуги – відносини між ними.

Семантичні мережі (СМ) відносяться до об'єктно-орієнтованих методів представлення знань. На відміну від логічних, чисто аналітичних моделей баз знань, СМ представляють клас підходів, для яких загальним є використання графічних схем з вузлами (які позначають основні поняття, сутності ПО) і дугами, що з'єднують ці вузли. Останні відображають відносини між вузловими елементами.

Своїм походженням СМ зобов'язані так званим асоціативним мережам (АМ). Вони використовувалися для моделювання інформаційних процесів, що відбуваються в пам'яті людини. Часто АМ і СМ розглядаються як синоніми. Термін "семантична" означає "змістовна", а семантика – наука, яка встановлює відносини між символами та об'єктами, які вони позначують. Тобто наука, яка визначає зміст символів. Поняття - це абстрактні або конкретні об'єкти, а відносини – це зв'язки типу: "це"("is", "a kind of"), "має частину" ("has part"), "належить", "полюбляє", та ін. Характерна особливість семантичних мереж – обов'язкова присутність трьох типів відносин:

- **клас** – екземпляр класу (автомобіль - BMW);
- **властивість** – значення (кузов - універсал);
- **зразок** екземпляру класу (BMW – модель X5).

Найбільш поширені в семантичних мережах типи відносин:

- зв'язки типу "частина-ціле" ("клас-підклас", "елемент-множина", та ін.);

- функціональні зв’язки (визначаються, як правило, за допомогою дієслів “впливає”, “успадковує”, та ін.);
- кількісні зв’язки (більш, менш, дорівнює, та ін.);
- зв’язки у просторі (далеко від, близько від, поза, над, та ін.);
- зв’язки за часом (на протязі, раніше, пізніше, та ін.);
- зв’язки за атрибутами (має властивість, має значення, та ін.);
- логічні зв’язки (АБО, ТА, НЕ, та ін.);
- лінгвістичні зв’язки (за правилами граматик, та ін.).

Проблема пошуку розв’язку в базі знань типу семантичної мережі процедурно зводиться до задачі пошуку фрагмента мережі. Фрагмент мережі відповідає деякій її підмножині, за допомогою якої формалізується запит до бази знань. Недоліком даної моделі представлення знань є складність організації процедури пошуку виводу для складних ієрархічних багаторівневих семантичних мереж. Зокрема, вивід на основі пошуку по перетинанню зв'язків погрожує небезпекою комбінаторного вибуху.

1.5. Фреймові системи представлення знань

Фреймові моделі (системи, мережі фреймів) у порівнянні із семантичними мережами дають більш формалізований і в той же час досить гнучкий «механізм» представлення знань. Випереджаючи строге визначення фрейму, помітимо, що фрейм можна розглядати як складний вузол в особливого виду семантичної мережі. У рамках фреймових моделей вдалося значною мірою об'єднати декларативні знання про об'єкти і процедурні знання про методи витягу і перетворення інформації для досягнення заданих цілей.

Термін фрейм (каркас, рамка) був запропонований у 1973р. М Мінським для мінімального опису явищ, понять, об'єктів. Відповідно до його визначення фрейми – це структури даних (знань), використовувані для представлення стереотипних ситуацій. У психології та філософії відоме поняття абстрактного образа чи прототипу. Наприклад, при проголошенні вголос слова «кімната» виникає її абстрактний образ (рамка чи фрейм), що включає житлове приміщення з чотирма стінами, підлогою, стелею, вікнами і дверима, площею від (приблизно) 6 до 20 кв.м. Цей опис мінімальний в тому розумінні, що з нього нічого не можна забрати. Так, забравши вікна, одержимо фрейм прикомірки чи комори і т.п. У цьому мінімальному описі опущені деталі (колір стін, кількість вікон і т.д.) Факт існування подібних деталей для конкретної кімнати враховується тим, що в згаданому описі маються слоти – вакантні клітини пам’яті. На рівні мінімального опису кімнати як прототипу деякого житлового приміщення ці слоти не заповнені. Вони заповнюються (чи прототип обрамляється) конкретними значеннями атрибутів кімнати (кількість вікон, колір стін, висота стель і ін.), якщо має місце деяка конкретна ситуація.

Абстрактний образ (прототип) деякого об'єкта з потенційною можливістю його обрамлення атрибутами називається фреймом. Він дозволяє сконцентрувати всі знання про даний об'єкт (чи клас об'єктів) у єдиній структурі даних (фреймі), а не розпорошувати їх між безліччю більш дрібних структур начебто логічних формул чи продукцій. Декларативні і процедурні знання про деяку сутність укладаються (пакуються) у єдину структуру (фрейм).

Інформаційна структура фрейму представлена на рис.1.Фрейм складається з імені, що виконує роль ідентифікатора, і слотів. Ідентифікатор, що привласнюється фрейму, повинний мати унікальне ім'я, єдине в даній фреймовій (мережній) системі. Кожен фрейм включає довільне число слотів, при цьому деякі з них визначаються самою системою, а інші задаються користувачем.

Кожен слот має визначену структуру даних, що включає:

- **Ім'я слота.** Це ідентифікатор, що привласнюється слоту. Він унікальний у межах даного фрейму.

Ім'я фрейму

Ім'я слота	**Ім'я атрибута слота**	**Значення (атрибута) слота (приєднана процедура)**	**Демон**
Слот 1			
Слот 2			
⋮			
Слот n			

Рис.1.2 Структура фрейму

- **Ім'я атрибута слота** – ідентифікатор атрибута слота.
- **Значення (атрибута) слота (приєднана процедура).** Покажчики типу даних атрибутів, значення атрибутів. Особливістю є наявність у слоті так званих приєднаних (інкапсульованих) процедур. Приєднана процедура являє собою програму процедурного типу, що запускається за повідомленнями, які одержані з інших фреймів.
- **Демон**. Демоном називається процедура, яка автоматично запускається при виконанні деякої умови. Наприклад, демон IF-NEEDED запускається, якщо в момент звертання до слоту його значення не було встановлено. Демон IF-ADDED запускається при підстановці в слот значення, демон IF-REMOVED – при стиранні значення слота. У результаті виконуються всі рутинні операції, що забезпечують підтримку бази знань в актуальному стані.

Завдяки слотам-посиланням (“a kind of”, “is a”, та ін) фреймові системи утворять ієрархічні структури, що реалізують принцип спадкування інформації. Спадкування відбувається в напрямку «суперклас-підклас», «клас-екземпляр класу». Фрейм, у якому заповнені всі значення слотів, називається фреймом - екземпляром. Існують ще зразки чи фрейми-прототипи. Вони являють собою оболонку, у якій зазначені тільки імена слотів. Інкапсульовані в слоти чи приєднані процедури є важливою особливістю фреймових мереж, що істотно відрізняють їх від мереж семантичних. Ці процедури додають фреймовій системі можливість керування виводом, таким же способом, як це робиться на основі використання об’єктно-орієнтованих мов.

§2. Лабораторна робота №1
ПРОЕКТУВАННЯ СИСТЕМ НЕЧІТКОГО ВИВОДУ НА ОСНОВІ АЛГОРИТМУ МАМДАНІ

Мета роботи: освоїти методику проектування системи нечіткого виводу на основі розробки та використання баз знань продукційних правил з використанням алгоритму Мамдані.

2.1. Основні поняття

Знання можна формалізувати у вигляді системи нечітких логічних висловлювань. Кожне висловлювання можна оцінити нечітким ступенем істинності. Наприклад, висловлювання «швидкість машини висока» може бути істинне на 80%, а висловлювання «завтра буде морозна погода» - на 100%. Кожне таке висловлювання можна описати за допомогою відношень множин лінгвістичних нечітких змінних.

Лінгвістична змінна – це кортеж наступних значень $\langle \beta, T, X, G, M \rangle$, де

β - ім’я змінної (наприклад, «швидкість автомобіля»);

T – базова множина значень її термів – значень, кожне з яких надається за допомогою нечіткої множини (наприклад, «мала», «середня», «висока», «дуже висока»);

X – множина – носій можливих конкретних значень змінної для всіх термів (наприклад, $X = [0,200]$ км/год.);

G - деяка синтетична процедура генерації нових термів з множини T (наприклад, «дуже мала»);

M – семантична процедура надання терму певної нечіткої змінної вигляду $\langle X, \mu_i(X) \rangle$, $\mu_i(X)$ - функція належності *i*-того терму з множини T.

В системі **MATLAB** існує середовище для формування систем знань нечіткого виводу. Для входу в це середовище слід ввести в командному рядку слово **fuzzy** і натиснути клавішу **<Enter>**. Побудова системи

нечіткого виводу (СНВ), яка основана на використанні алгоритму Мамдані, має наступні етапи:

1. Проектування бази правил СНВ. Кожне правило представляється у вигляді:

Якщо <умова> **тоді** <заключення> [міра вірності правила]

Для алгоритму Мамдані <умова> і <заключення> виглядають як логічні зв'язки наступних записів: **<нечітка змінна> = < значення >**

2. Введення цих правил в СНВ

3. Використання СНВ для обробки вхідної інформації у вигляді конкретних значень вхідних (нечітких) змінних. Цей етап, в свою чергу, розкладається на наступні складові:

3.1 Введення значень вхідних змінних. Тобто, деякий фактів, які вважаються істинні на 100%.

3.2 Фазифікація вхідних змінних – встановлення відповідності між конкретним значенням вхідних змінних і значенням її терму, разом з функцією належності

3.3 Агрегування складних умов, які стоять в правилах після ключового слова ЯКЩО, тобто визначення степені істинності всіх умов в усіх правилах, якщо умови надаються за допомогою складних логічних виразів. Правило активується, якщо істинність його умови більша за нуль. В базах знань процедура агрегування умов в правилах виконується за допомогою нечітких логічних операцій – нечіткої кон'юнкції, нечіткої діз'юнкції, нечіткої відмови, та ін.

3.4 Активація підзаключень – процес визначення степені істинності (належності до відповідних термів) змінних, які стоять в заключеннях активних правил, за формулою: $c_k = b_k F_k$, де c_k - степінь істинності заключення правила k, b_k - степінь істинності його умови, F_k - степінь істинності самого правила (ваговий коефіцієнт k -правила). Після визначення вектору $C = (c_1,..,c_q)$ визначаються функції належності для кожного із підзаключень для кожної вихідної лінгвістичної змінної. Припустимо, що відповідний терм вихідної лінгвістичної змінної визначається функцією належності $\mu(y)$. Тоді після процедури активації отримуємо поновлену функцію належності відповідного терму (підзаключення) $\mu'(y)$ за одним із методів нечіткої композиції:

- min – активізація: $\mu'(y) = \min\{c_i, \mu(y)\}$;
- prod-активізація: $\mu'(y) = c_i \mu(y)$;
- average-активізація: $\mu'(y) = 0.5(c_i + \mu(y))$.

Відзначимо, що різні правила підзаключень можуть містити однакові терми лінгвістичних змінних. У цьому випадку для кожного терму ми визначаємо множину різних функцій належності, які обчислюються за одним із правил нечіткої композиції по кожному правилу продукцій. Остаточна функція належності для цього терму визначається у наступному пункті.

3.5 Акумуляція заключень, тобто, визначення значення функцій належності для термів всіх вихідних змінних. Якщо для одного терму визначена множина функцій належності $\mu_1'(y),\ldots,\mu_p'$, то акумуляція виконується за одним із правил об'єднання нечітких множин:

- об'єднання: $\mu'(y) = \max\{\mu_1'(y), \mu_2'\}$;
- алгебраїчне об'єднання: $\mu'(y) = \mu_1'(y) + \mu_2'(y) - \mu_1'(y)\mu_2'(y)$;
- граничне об'єднання: $\mu'(y) = \max\{\mu_1'(y) + \mu_2'(y) - 1, 0\}$;
- операція λ - суми: $\mu'(y) = \lambda\mu_1'(y) + (1-\lambda)\mu_2'(y)$, $\lambda \in [0,1]$.
- драстичне об'єднання: $\mu'(y) = \begin{cases} \mu_1'(y), & if\ \mu_2'(y) = 0, \\ \mu_2'(y), & if\ \mu_1'(y) = 0, \\ 1, & else. \end{cases}$

3.6 Дефазифікація вихідних змінних (визначення конкретних значень за функціями належності термів) розглядається методом центру ваги для неперервних та дискретних нечітких множин за формулами:

$$z = \frac{\int_{y_{\min}}^{y_{\max}} y\mu'(y)dy}{\int_{y_{\min}}^{y_{\max}} \mu'(y)dy}, \qquad z = \frac{\sum_{i=1}^{n} y_i\mu'(y_i)}{\sum_{i=1}^{n} \mu'(y_i)}.$$

Розглянемо принципи побудови та роботи системи нечіткого виводу на прикладі задачі візуалізації поверхні, яка задана функцією.

2.2. Проектування та використання системи нечіткого виводу

Завдання 1. За допомогою СНВ зобразити поверхню функції $y = (x_1^2 - 8)\cos(x_2)$ на множині $x_1 \in [0,4]; x_2 \in [0,4]$.

Проектування системи нечіткого виводу слід проводити на основі графічного зображення вказаної залежності. Для цього в М-файлі складемо наступну програму:

```
%Побудова графіка функції y=(x1^2-8)*cos(x2)
%в області x1є[0,4] и x2є[0,4].
n=15;
x1=0:4/(n-1):4;
x2=0:4/(n-1):4;
y=zeros(n,n);
for j=1:n
y(j,:)=(x1.^2-8)*cos(x2(j));
end
surf(x1,x2,y)
xlabel('x1')
ylabel('x2')
zlabel('y')
title('Target');
```

В результаті виконання цієї програми отримуємо графічне зображення, яке наведено на рис 2.1.

Проектування СНВ складається з наступних кроків.

Крок 1. Завантажити основний fis-редактор в (редактор нечіткого виводу) введенням в командному рядку слова **fuzzy.** Після чого з'явиться вікно редактору нечіткого виводу.

Крок 2. Ввести нову вхідну змінну. Для цього вибрати пункт **Add Input** в меню **Edit**.

Крок 3. Перейменувати першу вхідну змінну. Для цього слід зробити одне натиснення лівої кнопки миші на блоці **Input1**, ввести нове позначення **x1** в поле редагування імені поточної змінної і натиснути **<Enter>**.

Крок 4. Перейменувати другу вхідну змінну. Для цього зробити одне натиснення лівої кнопки миші на блоці **input2**, ввести нове позначення **x2** в поле редагування імені поточної змінної і натиснути **<Enter>**.

Крок 5. Перейменувати вихідну змінну. Для цього зробити одне натиснення лівої кнопки миші на блоці **output1**, ввести нове позначення **y** в поле редагування імені поточної змінної і натиснути **<Enter>**.

Крок 6. Задати ім'я системі. Для цього в меню **File** вибрати в підменю **Export to disk** і ввести ім'я файлу, наприклад, **first**.

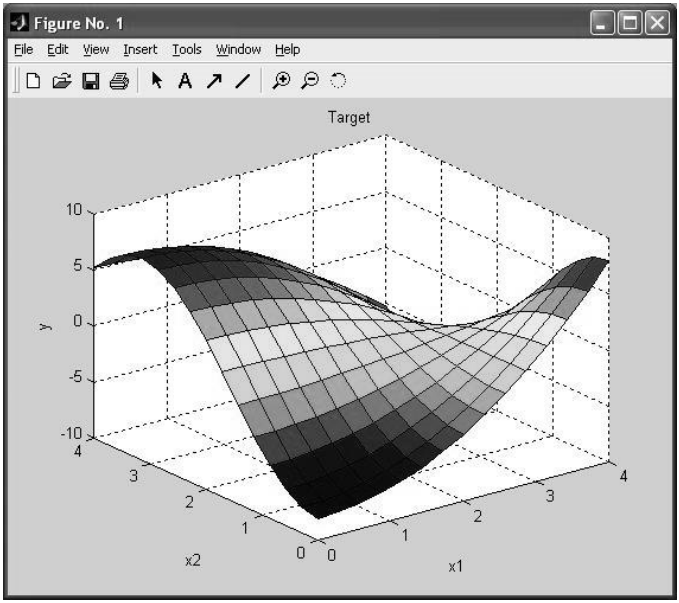

Рис 2.1. Зображення поверхні залежності

Крок 7. Перейти в редактор функцій належності. Для цього зробити швидко подвійне натиснення лівої кнопки миши на блоці **x1**.

Крок 8. Задати діапазон зміни змінної **x1**. Для цього надрукувати **0 4** в поле **Range** і натиснути **<Enter>**.

Крок 9. Задати функції належності змінної **x1**. Для лінгвістичної оцінки цієї змінної будемо використовувати 3 терми з трикутними функціями належності. Якщо в вікні немає ще функцій належності, тоді в меню **Edit** слід вибрати команду **Add MFs...** В результаті з'явиться діалогове вікно вибору типу і кількості функцій належності. За замовченням ці 3 терми мають трикутну функцію належності. Тому просто потрібно натиснути **<Enter>.**

Крок 10. Задати *найменування термів змінної* **x1**. Для цього робимо одне натиснення лівою кнопкою миші на графіку першої функції належності. (див. рис. 2.2). Потім вводимо найменування терму, наприклад, **L** (**Низький)**, в полі **Name** і натискаємо **<Enter>**. Потім робимо одне натиснення лівою кнопкою миші на графіку другої функції належності і вводимо найменування терму, наприклад, **A** (**Середній)**, в полі **Name і** натискаємо **<Enter>**. Ще раз робимо одне натиснення лівою кнопкою миші по графіку третьої функції належності і введемо найменування

терму, наприклад, **H** (**Високий)**, в полі **Name і** натискаємо **<Enter>**. В результаті отримуємо графічне вікно, яке зображено на рис. 2.2.

Крок 11. Задамо функції належності змінної **x2**. Для лінгвістичної оцінки цієї змінної будемо використовувати 5 термів з гаусовськми функціями належності. Для цього активізуємо змінну **x2** за допомогою натиснення лівої кнопки миші на блоці **x2.** Задамо діапазон змін **x2**. Для цього надрукуємо 0 **4** в полі **Range** (див. рис. 2.3) і натиснемо **<Enter>**. Потім в меню **Edit** виберемо команду **Add MFs...**. В діалоговому вікні, що зявиться, оберемо тип функції належності **gaussmf** в полі **MF type** і **5** термів в полі **Number of MFs**. Після чого натискаємо <**Enter>.**

Крок 12. За аналогією з кроком *10* задамо наступні найменування термів змінної **x2**: **L (Низький), LA (Нижче середнього), A (Середній), HA (Вище середнього), H (Високий)**. В результаті отримуємо графічне вікно, яке зображене на рис. 2.3.

Крок 13. Задамо функції належності змінної **y**. Для лінгвістичної оцінки цієї змінної будемо використовувати 5 термів з трикутними функціями належності. Для цього активуємо змінну **y** за допомогою натиснення лівої кнопки миші на блоці **y.** Задамо діапазон змін змінної **y**. Для цього надрукуємо **-10 10** в полі **Range** (див. рис. 2.4) і натиснемо **<Enter>**. Потім в меню **Edit** оберемо команду **Add MFs...**. В діалоговому вікні, що зявиться, виберемо **5** термів в полі **Number of MFs**. Після чого натискаємо **<Enter>.**

Крок 14. За аналогією з кроком *10* задамо наступні найменування термів змінної **y**: **L** (**Низький)**, **LA** (**Нижче середнього) A (середній)**, **HA (Вище середнього)**, **H (Високий)**. В результаті отримуємо графічне вікно, яке представлене на рис. 2.4.

Крок 15. Перейдемо в редактор бази знань **RuleEditor**. Для цього оберемо в меню **Edit** команду **Rules** або в меню **View** команду **Edit rules...**.

Крок 16. На основі візуального спостереження за графиком, який зображений на рис. 2.1 сформуємо наступні десять правил:

1. **Якщо** x1=Низький **і** x2=Низький, **тоді** y=Низький;
2. **Якщо** x1=Низький **і** x2=Високий, **тоді** y=Середній;
3. **Якщо** x1=Низький **і** x2=Вище середнього, **тоді** y=Високий;
4. **Якщо** x1=Високий **і** x2=Низький, **тоді** y=Вище середнього;
5. **Якщо** x1=Високий **і** x2=Високий, **тоді** y=Низький;
6. **Якщо** x1=Середній **і** x2=Середній, **тоді** y=Середній;
7. **Якщо** x1=Середній **і** x2=Вище середнього, **тоді** y=Вище середнього;
8. **Якщо** x1=Низький **і** x2=Нижче середнього, **тоді** y=Нижче середнього;
9. **Якщо** x1=Середній **і** x2=Вище середнього, **тоді** y=Середній.

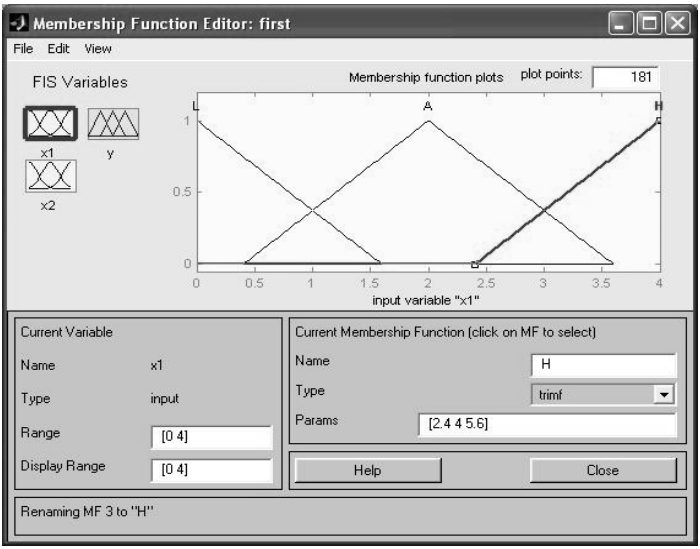

Рис. 2.2. Функція належності змінної x_1.

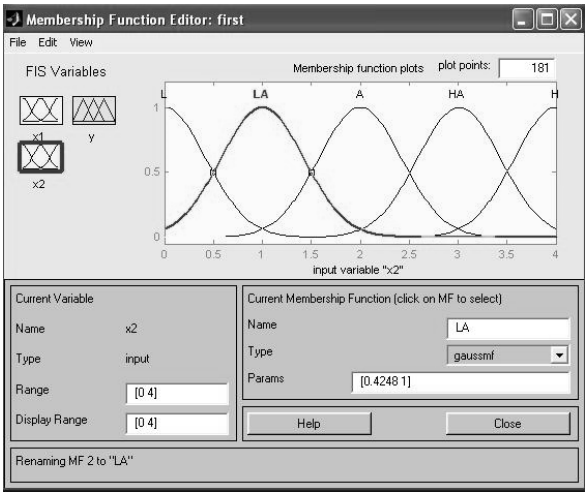

Рис 2.3. Функція належності змінної x_2

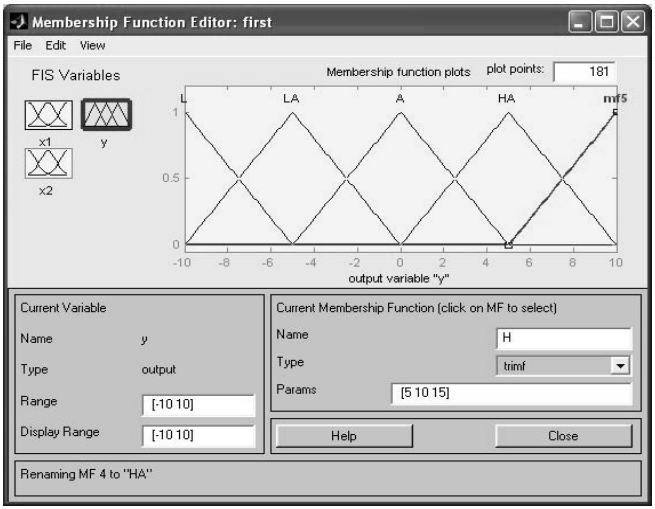

Рис. 2.4. Функції належності змінної y.

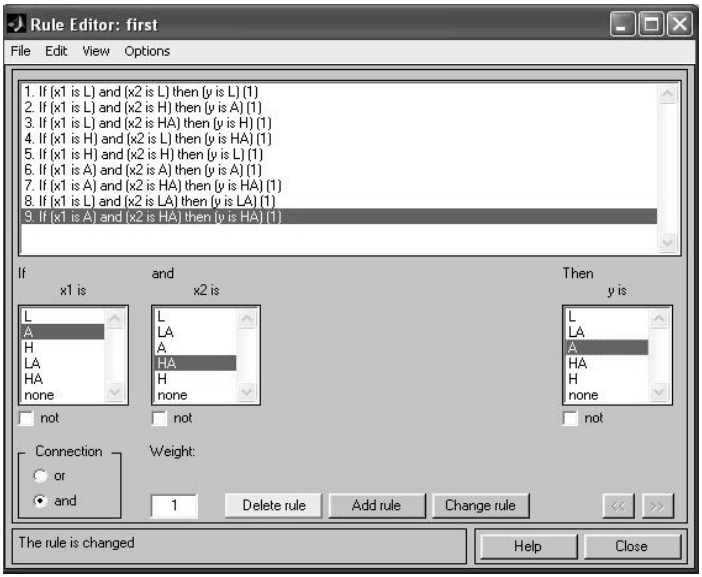

Рис. 2.5. Вікно правил

Для введення правила необхідно обрати в меню відповідну комбінацію термів і натиснути кнопку **Add rule**. На рис. 2.5 зображено вікно

редактору бази знань після введення усіх 9 правил. Число в дужках в кінці кожного правила представляє собою вагові коефіцієнти відповідного правила. Потрібно звернути увагу на параметр **Weight**, який вказує вагу нечіткої впевненості в правилі. Його можна задавати в діапазоні $[0,1]$.

Крок 17. Збережемо побудовану систему. Для цього в меню **File** в підменю **Export** оберемо команду **To disk**.

На рис. 2.6 приведено вікно візуалізацій нечіткого логічного виводу. Це вікно активується командою **View rules...** меню **View.** В полі **Input** вказуються значення вхідних змінних, для яких виконується логічний вивід. Тобто, обраховується за алгоритмом Мамдані значення вихідної змінної.

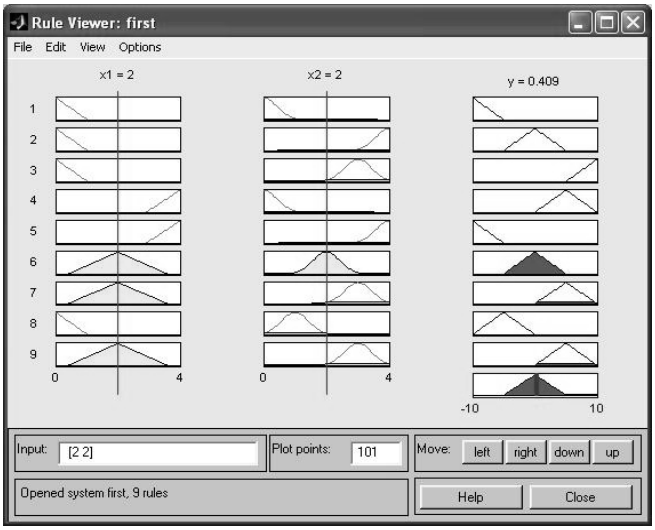

Рис 2.6. Візуалізація нечіткого виводу

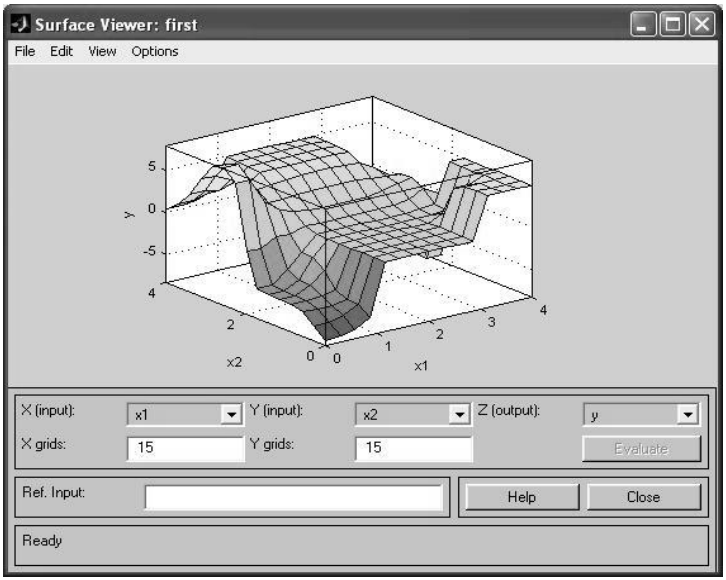

Рис 2.7. Поверхня системи «вхід-вихід».

На рис. 2.7 приведена поверхня “входи-вихід”, яка відповідає синтезованій системі логічного виводу. Для виводу цього вікна необхідно використати команду **View surface...** меню **View.** Порівнюючи поверхні на рис 2.1. і рис. 2.7. , можна зробити висновок, що нечіткі правила досить добре описують складну нелінійну залежність.

2.3 Завдання для самостійної роботи.

1. Створити систему нечіткого виводу, яка моделює залежність $y = x_1^2 \sin(x_2 - 1)$ при $x_1 \in [-7,3]$; $x_2 \in [-4.5,2]$.

2. Створити систему нечіткого виводу, яка відтворює поверхню $y = \ln(x_1 + 1)\cos(x_2)$ при $x_1 \in [1,5]$; $x_2 \in [0,2]$

3. Створити систему нечіткого виводу, яка відтворює поверхню $y = \ln(x_1 + 1)\frac{1}{1 + x_2}$ при $x_1 \in [1,5]$; $x_2 \in [0,2]$

§3. Лабораторна робота №2
ПРОЕКТУВАННЯ СИСТЕМ НЕЧІТКОГО ВИВОДУ СУГЕНО

Мета роботи: освоїти методику проектування системи нечіткого виводу на основі розробки та використання баз знань продукційних пра-

вил з використанням алгоритму Сугено. Провести порівняльний аналіз алгоритмів Мамдані і Сугено.

3.1 Основні поняття

Розглянемо основні етапи проектування систем нечіткого виводу за алгоритмом Сугено на прикладі задачі візуалізації поверхні, яка реалізує залежність $y = (x_1^2 - 8)\cos x_2$ на відрізку $x_1 \in [0,4], x_2 \in [0,4]$ (рис.2.1). Відмінність СНВ алгоритму Сугено полягає у проектуванні вихідних змінних. Формування бази правил систем нечіткого виводу наступного формату:

Правило <#>: **Якщо** «змінна 1=значення А» **і** «змінна 2=В» **Тоді** « $y = k_1 A + k_2 B + k_0$ »

або

Правило <#>: **Якщо** «змінна 1 = значення А» **і** «змінна 2 = В» **Тоді** « $y = C$ »

3.2 Метод проектування та використання систем за алгоритмом Сугено

Моделювання заданої поверхні будемо реалізовувати за допомогою наступних правил бази знань:

1. Якщо x1 та x2 низькі, тоді $y = -10$
2. Якщо x1 низьке, тоді $y = 3{,}75x_2 - 10$
3. Якщо x1 низьке і x2 вище середнього, тоді $y = 7$
4. Якщо x1 та x2 високі, тоді $y = -10$
5. Якщо x2 низьке, тоді $y = 4x_1 - 10$
6. Якщо x1 високе, тоді $y = 15 - 3{,}75x_2$
7. Якщо x2 високе, тоді $y = 15 - 3{,}75x_1$
8. Якщо x1 низьке і x2 високе, тоді $y = 7$
9. Якщо x1 середнє і x2 середнє, тоді $y = 7$
10. Тоді проектування системи нечіткого виводу типу Сугено лежить у виконанні наступної послідовності кроків.

Крок 1. Для завантаження основного fis-редактору надрукуємо слово **fuzzy** у командному рядку Mathlab.

Крок 2. Оберемо тип системи. Для цього в меню **File** в підменю **New fis…** оберемо команду **Sugeno**.

Крок 3. Додамо другу вхідну змінну. Для цього в меню **Edit** оберемо команду **Add input**.

Крок 4. Перейменуємо першу вхідну змінну. Для цього зробимо одне натиснення лівої кнопки миші на блоці **input1**, введемо нове позначення **x1** в полі редагування імені поточної змінної і натиснемо **<Enter>**.

Крок 5. Перейменуємо другу вхідну змінну. Для цього зробимо одне натиснення лівою кнопкою миші на блоці **input2**, введемо нове позначення **x2** в полі редагування імені поточної змінної і натиснемо **<Enter>**.

Крок 6. Перейменуємо вихідну змінну. Для цього зробимо одне натиснення лівою кнопкою миші на блоці **output1**, введемо нове позначення **y** в полі редагування імені поточної змінної і натиснемо **<Enter>**.

Крок 7. Задамо ім'я системи. Для цього в меню **File** в підменю **Export** оберемо команду **To disk** і введемо імя файла, наприклад, **FirstSugeno**.

Крок 8. Перейдемо в редактор функций належності. Для цього зробимо швидке подвійне натиснення лівої кнопки миші на блоці **x1**.

Крок 9. Зададимо діапазон змін змінної **x1**. Для цього надрукуємо **0 4** в полі **Range** і натиснемо **<Enter>** (див. рис.3.1).

Крок 10. Задамо функції належності змінної x_1. Для лінгвістичної оцінки цієї змінної будемо використовувати 3 терми з трикутними функціями належності, які встановлені за замовченням. Задамо найменування термів змінної x_1. Для цього робимо одне натиснення лівою кнопкою миші по графіку першої функції належності.

Крок 11. *Аналогічно з*адамо функції належності змінної x_2. Для лінгвістичної оцінки цієї змінної будемо використовувати 4 терми з трикутними функціями належності. Для цього активізуємо змінну x_2 за допомогою натиснення лівої кнопки миші на блоці x_2**.** Задамо діапазон змін x_2. Для цього надрукуємо 0 **4** в полі **Range** і натиснемо **<Enter>**. Задамо найменування 4 термів (наприклад, **L Низький**, **A Середній**, **HA Вище середнього**, **H Високий**).

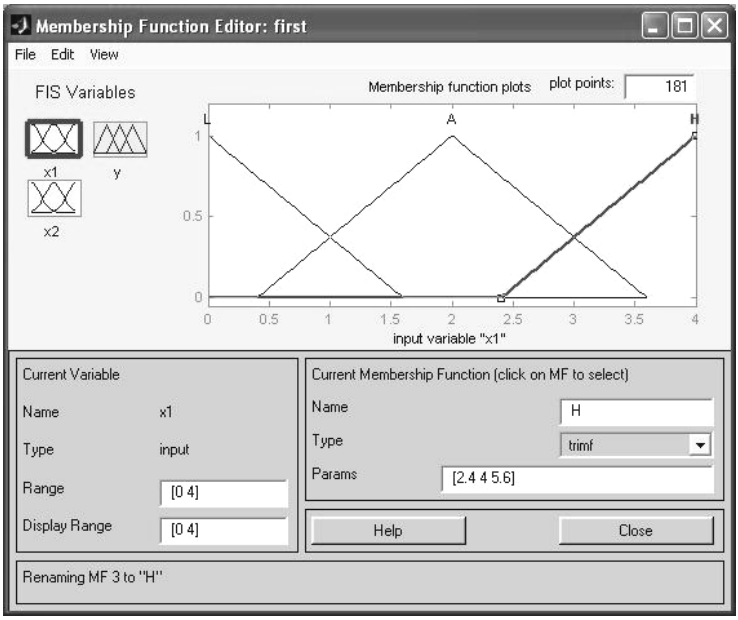

Рис 3.1. Функції належності змінної x_1

Крок 12. Задамо лінійні залежності між входами і виходом, яке наведене в базі знань. Для цього активуємо змінну **y** за допомогою натиснення лівої кнопки на блоці **y.** В правому верхньому куті можуть з'явитися позначення функцій належності, кожна з яких відповідає однієї лінійної залежності між входами і виходом. В базі знань, яка наведена на початку файлу, вказані 6 різних залежностей: y=-10; y=7; y=3,75 x_1 -10; y=4 x_1 -10;

y=-3,75 x_1 +15; y=-3,75 x_2 +15. Додамо ще необхідну кількість функцій залежності шляхом обирання команди **Add Mfs…** меню **Edit.**

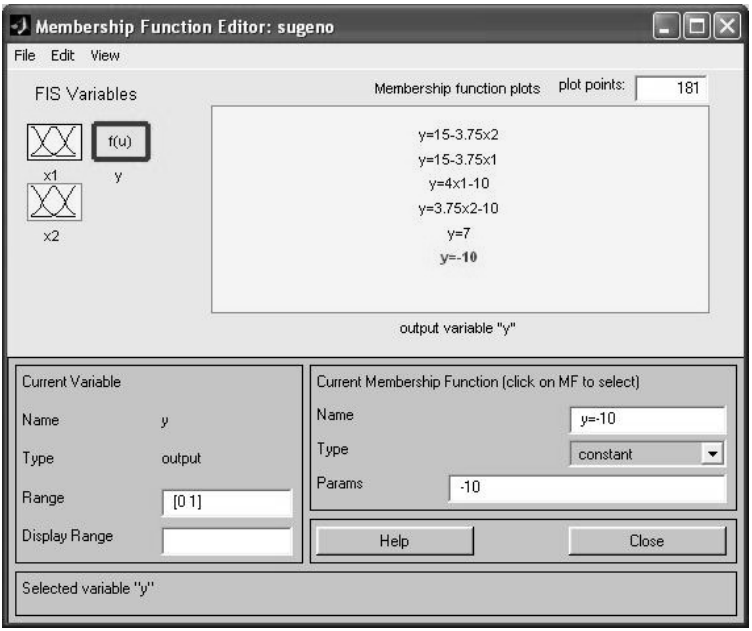

Рис 3.2. Вікно лінійних залежностей «входи-вихід»

Крок 13. Задамо найменування і параметри цих залежностей. Для цього робимо одне натиснення лівою кнопкою миші по імені першої залежності **mf1**. Потім друкуємо назву залежності, наприклад **y=-10**, в полі **Name**, і встановлюємо тип залежності – константа шляхом обирання опції **Constant** в меню **Type**. Після цього вводимо значення параметру – **-10** в полі **Params**. Аналогічну процедуру робимо для другої змінної **y=7.**

Для третьої функції **mf3** введемо найменування, наприклад, **y=3.75x1-10**. Потім вкажемо лінійний тип залежності шляхом вибору опції **Linear** в меню **Type** і введемо параметри залежності **3.75 0 -10** в полі **Params**. Для лінійної залежності порядок параметрів наступний: перший параметр – коєфіцієнт при першій змінній, другий – при дрцгій і т.д., останній параметр – вільний член залежності. Таким ж чином введемо назви і параметри для всіх 6 функцій належності змінної **y**.

В результаті отримуємо графічне вікно, яке представлено на рис. 3.2.

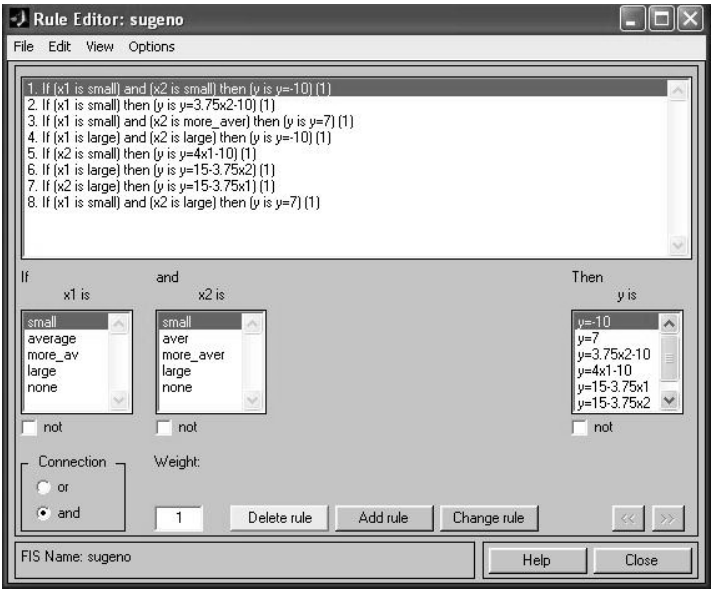

Рис 3.3. Нечітка база знань для системи типу Сугено

Крок 14. Перейдемо в редактор бази знань **RuleEditor**. Для цього оберемо в меню **Edit** команду **Edit rules...**. і введемо правила. Для вводу правила необхідно обрати відповідну комбінацію термів і залежностей і натиснути кнопку **Add rule**. На рис. 3.3 зображене вікно редактору бази знань після введення усіх 6 правил.

На рис. 3.4 приведено вікно візуалізації нечіткого логічного виводу. Це вікно активується командою **View rules...** меню **View.** В полі **Input** вказуються значення вхідних змінних, для яких виконується логичний вивід. Як можна побачити з рисунку, значення вихідної змінної, розраховується як середнє зважене значення результатів виходу за кожним правилом.

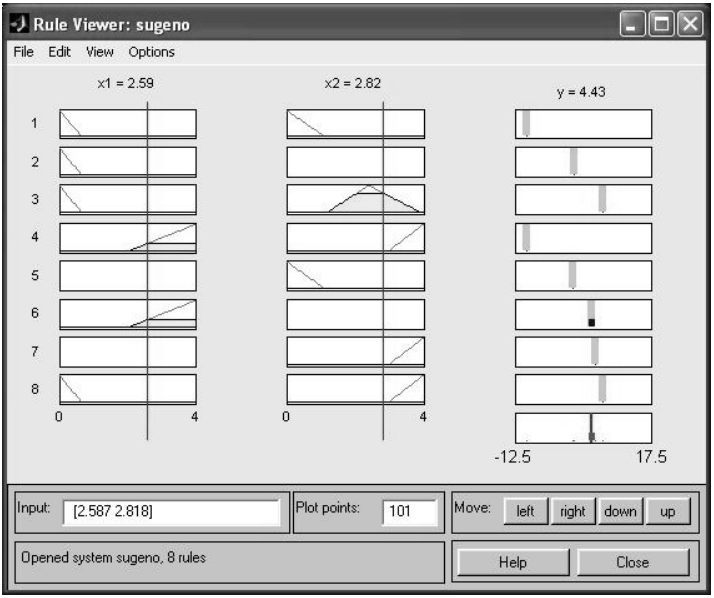

Рис. 3.4 Візуалізація нечіткого логічного виводу для системи типу Сугено

На рис. 3.5 приведена поверхня “входи-вихід”, яка відповідає синтезованій нечіткій системі. Для виведення цього вікна необхідно використати команду **View surface...** меню **View.** Порівнюючи цю поверхню і поверхню на рис. 1, можна зробити висновок, що нечіткі правила достатньо добре описують складну лінійну залежність. При цьому, модель типу Сугено більш точна. Перевага моделей типу Мамдані полягає у тому, що правила бази знань є прозорі і інтуїтивно зрозумілі, таді як для моделей типу Сугэно не завжди ясно які лінійні залежності «входи-вихід» необхідно використовувати.

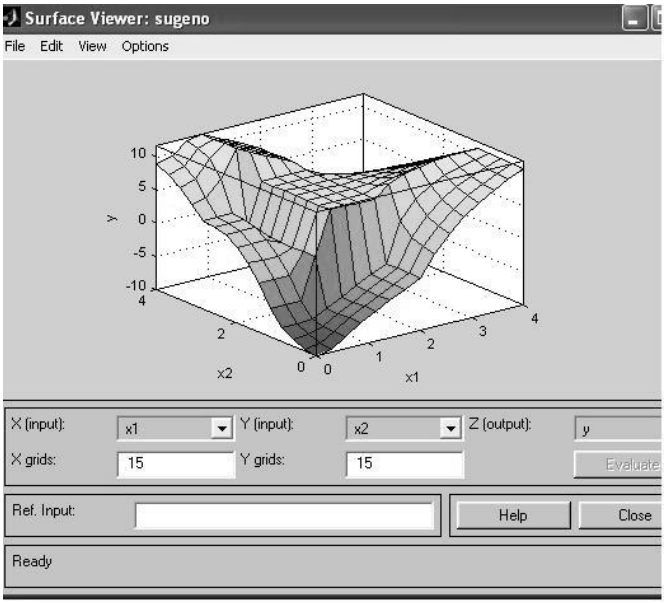

Рис 3.5. Поверхня "вхід-вихід" для системи алгоритму Сугено

3.3 Завдання для самостійної роботи.

1. Створити систему нечіткого виводу Сугено, яка моделює залежність $y = x_1^2 \sin(x_2 - 1)$ при $x_1 \in [-7{,}3]$; $x_2 \in [-4.5{,}2]$.

2. Створити систему нечіткого виводу Сугено, яка відтворює поверхню $y = \ln(x_1 + 1)\cos(x_2)$ при $x_1 \in [1{,}5]$; $x_2 \in [0{,}2]$

3. Створити систему нечіткого виводу Сугено, яка відтворює поверхню $y = \ln(x_1 + 1)\dfrac{1}{1 + x_2}$ при $x_1 \in [1{,}5]$; $x_2 \in [0{,}2]$

§4. Лабораторна робота №3
ПРОЕКТУВАННЯ СИСТЕМ КЕРУВАННЯ НА ОСНОВІ АЛГОРИТМІВ НЕЧІТКОГО ВИВОДУ ТА БАЗ ЗНАНЬ НЕЧІТКИХ ПРОДУКЦІЙ

4.1 Основні поняття

Одним з основних напрямів використання СНВ є розв'язування задач керування. Під **системою керування** [5] будемо розуміти з'єднання N елементів $e = \overline{1, N}$ виду

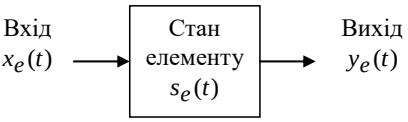

Рис. 4.1 Функціональний блок системи керування

в єдину конфігурацію, яка забезпечує їм необхідну поведінку для досягнення поставленої цілі. Зв'язок між входом (вектор-функцією $x_e(t)$) і виходом (вектор-функцією $y_e(t)$) в елементі системи (функціональному блоці) – це перетворення одного сигналу (причини) в інший (наслідок). Вся система керування (СК) теж може бути представлення аналогічною схемою:

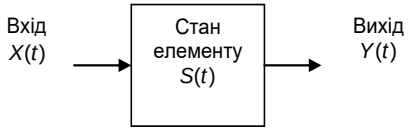

Рис. 4.2. Загальна схема системи керування

СК поділяють на розімкнені і замкнені. В розімкненій системі існує елемент – виконуючий пристрій, який визначає необхідне значення вхідного сигналу $X(t)$ («приймає рішення» про необхідний вплив на систему) в залежності від бажаного (необхідного) значення виходу Y^0:

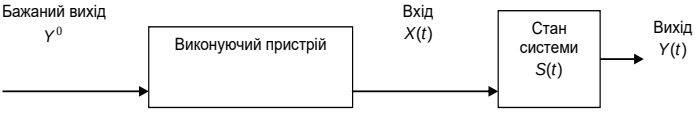

Рис. 4.3. Загальна схема розімкнутої системи керування

В СК елемент, який «приймає рішення» будемо називати активним. Якщо система представляє собою конфігурацію активних елементів, її називають **організаційною** системою управління (ОСУ). Тоді в систему може входити елемент – **координатор**, який визначає вхідні сигнали активних елементів.

В замкненій системі існує **зворотній зв'язок.** Окремі елементи вхідного вектора $X(t)$, які можна назвати **керуючими змінними**, залежать від різниці між бажаним значенням вихідних величин і їх реальним значенням $Y^0 - Y(t)$ (негативний зворотній зв'язок) або від суми цих значень $Y^0 + Y(t)$ (позитивний зворотній зв'язок):

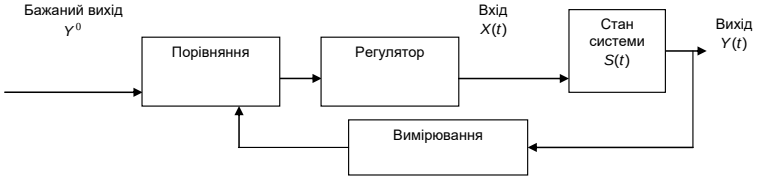

Рис. 4.4. Система керування зі зворотнім зв'язком

В якості прикладу розімкненої СУ може служити електронагрівальний пристрій, який керується кнопками завдання бажаної інтенсивності нагріву кімнати. Тоді вхідним сигналом є натиснення відповідної кнопки, а виходом – температура нагріву.

Як замкнену систему управління можна розглянути керування автомобілем на дорозі. Тут в якості вхідних елементів виступають кут повороту руля та сила натискання педалей, в якості вихідних елементів виступають швидкість та напрямок руху автомобіля, а в якості вимірювання в зворотньому зв'язку виступають органи зору водія, які спостерігають напрямок траси і дозволені швидкості.

Під **керуванням** (управлінням**)** системою будемо розуміти таке визначення вхідного сигналу $X(t)$, яке формує необхідну поведінку (зміну станів $S(t)$) системи для досягнення бажаних значень Y^0.

Побудова **нечіткої моделі 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. Вихідні змінні можуть знову поступати на вхід системи, утворюючи зворотній зв'язок, як показано на рис.4.5.

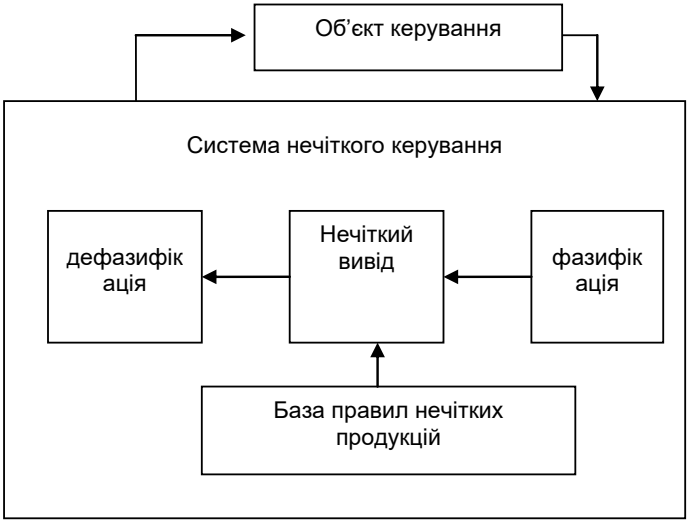

Рис 4.5. Архітектура компонентів процесу нечіткого керування

Таким чином, нечітка модель системи керування може бути описана за допомогою апарату системи нечіткого виводу.

4.2 Методика розробки та використання системи нечіткого виводу для розв’язування задачі керування

Розглянемо етапи побудови системи нечіткого виводу на основі прикладу побудови системи нечіткого керування процесом прийому на роботу у фірму нових співробітників. Нехай, у фірми є дві вакансії: програміст і системний адміністратор. Тому СНВ має дві вихідні лінгвістичні змінні y_1, y_2 з множинами-носіями у вигляді відрізків $[0,1]$, які інтерпретується 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 досіду роботи можуть бути задані наступні терми: «немає», «невеликий», «високий»

Крок 2. Фазифікація 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 $View \rightarrow Rules$ і переглянути можливі значення вихідної змінної прийняття рішення про прийом на роботу в залежності від зміни значень вхідних змінних.

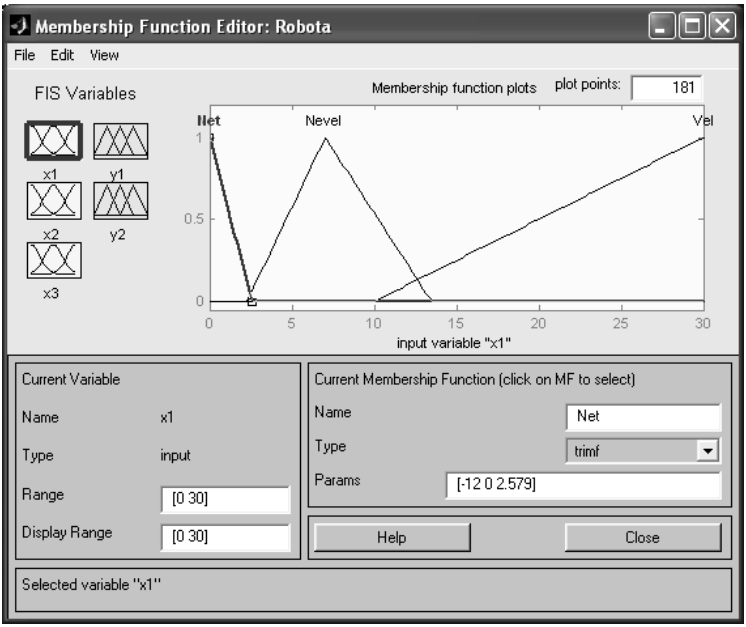

Рис. 4.6. Приклад термів змінної x_1 - досвіду роботи

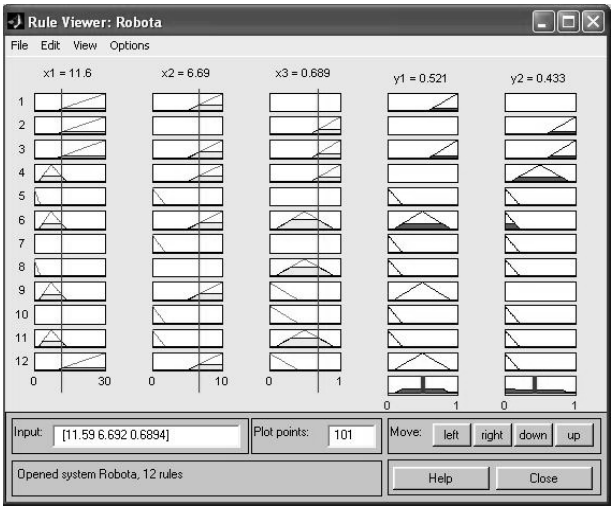

Рис 4.7. Вікно дії правил

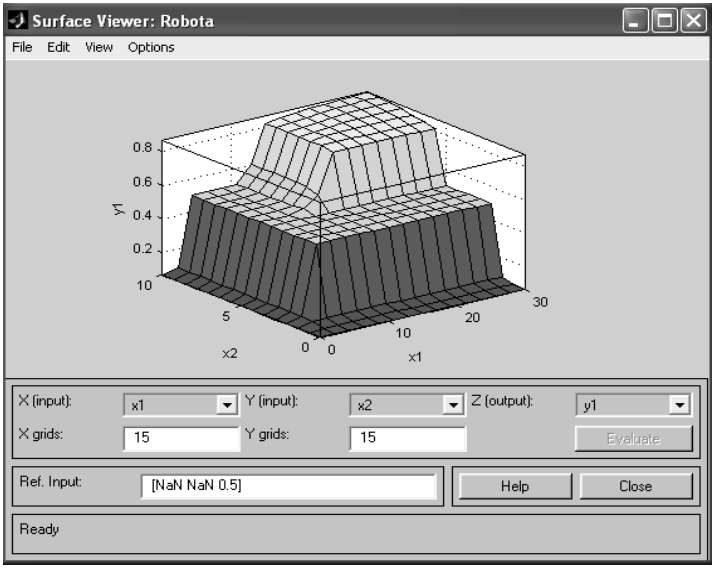

Рис 4.8 Поверхня СНВ задачі прийняття на роботу особи.

4.3 Завдання для самостійної роботи

Завдання 1. Побудова системи керування діями офіцера дорожньої служби

За допомогою СНВ алгоритму Мамдані побудувати базу знань, яка б консультувала офіцера дорожньої міліції що робити з машиною, яка наближається до контрольного пункту (пропустити, зупинити, перевірити документи, штрафувати або забирати права) в залежності від швидкості автомобіля та погодних умов, які можна задати за допомогою лінгвістичних змінних.

Завдання 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 СНВ, 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. Фазіфікація 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.

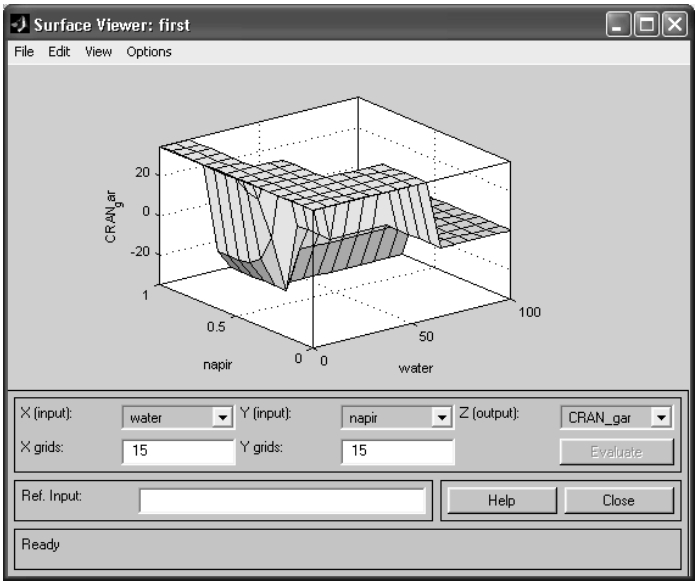

Рис. 4.9. Можливий вигляд поверхні задачі про змішувач води в душі

Завдання 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 СНВ, 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 додатня, то потрібно включити режим «холод», повернувши регулятор кондиціонеру на великий кут вліво.
2. Якщо температура повітря дуже тепла, а швидкість зміни температури від'ємна, тоді необхідно включити режим «холод», повернувши регулятор кондиціонеру на невеликий кут вліво.
3. Якщо температура повітря тепла, а швидкість зміни температури додатня, тоді потрібно включити режим «холод», повернувши регулятор кондиціонеру на великий кут вліво.
4. Якщо температура повітря тепла, а швидкість зміни температури від'ємна, тоді потрібно включити режим «холод», повернувши регулятор кондиціонеру слід вимкнути.
5. Якщо температура повітря дуже холодна, а швидкість зміни температури від'ємна, тоді потрібно включити режим «тепло», повернувши регулятор кондиціонеру на великий кут вправо.
6. Якщо температура повітря дуже холодна, а швидкість зміни температури додатня, тоді потрібно включити режим «тепло», повернувши регулятор кондиціонеру на невеликий кут вправо.
7. Якщо температура повітря холодна, а швидкість зміни температури від'ємна, тоді потрібно включити режим «тепло», повернувши регулятор кондиціонеру на великий кут вліво.
8. Якщо температура повітря холодна, а швидкість зміни температури додатня, 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 додатня, тоді потрібно включити режим «холод», повернувши регулятор кондиціонеру на невеликий кут вліво.
14. Якщо температура повітря в нормі, а швидкість зміни температури від'ємна, 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. Фазифікація вхідних змінних. На цьому кроці слід задати функції належності 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 належності 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.

Завдання 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

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 підніманя 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. Фазіфікація вхідних змінних. На цьому кроці слід задати функції належності 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.

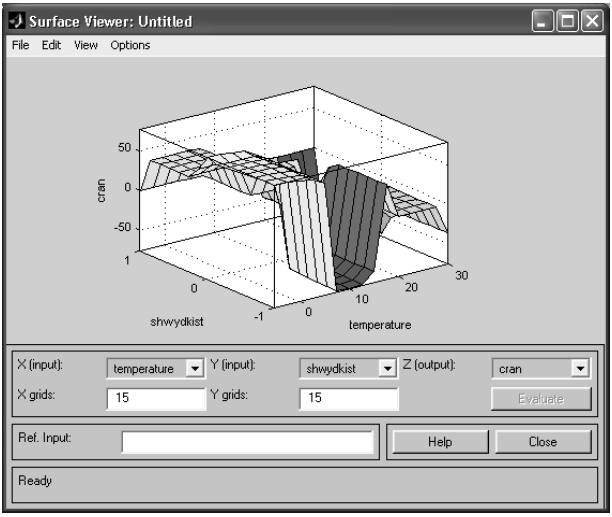

Рис 4.10. Наближений вигляд поверхні задачі про кондиціонер

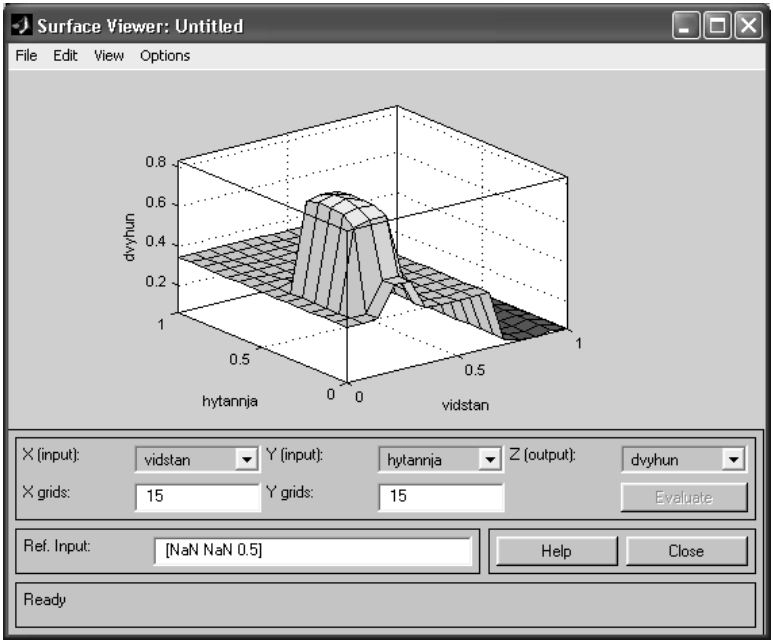

Рис 4.11. Наближений вигляд поверхні задачі контейнера

§5. Лабораторна робота № 4
РОЗРОБКА НЕЧІТКИХ МОДЕЛЕЙ СИСТЕМ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ НА ОСНОВІ БАЗ ЗНАНЬ НЕЧІТКИХ ПРОДУКЦІЙ

Мета роботи: Освоїти методику проектування і побудови систем підтримки прийняття рішень на основі моделі нечіткого виводу та бази знань нечітких продукцій, на прикладі системи оцінюваннія фінансової заможності клієнтів з боку банків при видачі довгострокових кредитів. На цьому прикладі познайомитися з особливостями розробки практично ефективних додатків - програмних засобів нечіткого моделювання.

5.1 Змістовна постановка задачі оцінювання фінансової заможності клієнтів

Суть розглянутої задачі полягає в наступному. При видачі довгострокових кредитів на будівництво будинків чи котеджів під заставу нерухомості для оцінки заможності клієнтів банками традиційно використовується метод експертних оцінок. При цьому метою банків є одержання максимального прибутку від укладених угод по наданню кредитів і виключення можливості фінансових утрат. Тому інтереси банків зосереджені, з одного боку, на збільшенні кількості успішних угод, а з іншого боку, на запобіганні невдалих угод, коли клієнт не повертає виданий кредит чи повертає його не вчасно.

Традиційно підставою для прийняття рішень по наданню кредитів у майбутньому служить досвід успішних угод, зроблених у минулому. Керівництво банку Home&Sevings Bank, в інтересах якого виконувалося відповідне дослідження, хотіло б узагальнити правила надання кредитів з метою максимально повно використовувати досвід експертів. При цьому необхідно виключити можливі помилки суб'єктивного характеру з боку окремих менеджерів у випадку неадекватного оцінювання фінансової заможності клієнтів.

Аналіз стратегії надання кредитів на будівництво будинків показує, що для оцінювання фінансової заможності клієнтів можуть бути використані різні характеристики, такі як місцезнаходження споруджуваного будинку, якість передбачуваного виконання опоряджувальних робіт, оцінка активів потенційного клієнта, оцінка доходу потенційного клієнта за винятком фіксованих витрат, величина підлягаючих сплаті відсотків з кредиту. При цьому власне фінансова заможність клієнта оцінюється його кредитоспроможністю.

Однією з перших формальних моделей, запропонованих для розв’язку даної задачі, була статистична модель, заснована на імовірнісній інтерпретації кількісної оцінки позитивного рішення про надання кредиту. Однак більш детальний аналіз цієї моделі згодом показав її

неадекватність, пов'язану з недостатнім обсягом статистичної вибірки й умовами надання кредитів, що змінюються з плином часу.

Саме з цієї причини була запропонована ідея розробки нечіткої моделі для оцінювання фінансової заможності клієнтів з метою прийняття рішень про надання довгострокових кредитів. При цьому як нечітка модель використовується система нечіткого виводу з наступними вхідними і вихідними змінними.

5.2 Описання вхідних і вихідних змінних задачі, що розглядається

Змістовна інтерпретація нечіткої моделі припускає вибір і специфікацію вхідних і вихідних змінних відповідної системи нечіткого виводу. При цьому в нечіткій моделі передбачається використовувати 5 вхідних змінних і 1 вихідну змінну.

У якості першої вхідної змінної використовується оцінка місця розташування споруджуваного будинку, що безпосередньо оцінює проект споруджуваного будинку, приймаючи до уваги розміщення будинку в тому чи іншому конкретному районі міста чи регіоні передмістя. Очевидно, чим вище ця оцінка, тим більше ліквідним представляється проект у випадку його реалізації на ринку нерухомості.

У якості другої вхідної змінної використовується якість передбачуваного виконання опоряджувальних робіт відповідно до архітектурного проекту споруджуваного будинку. Ця змінна вносить додатковий елемент в оцінку вартості споруджуваного будинку.

У якості третьої вхідної змінної використовується оцінка активів, що використовується для оцінки майна чи авуарів у випадку неспроможності потенційного клієнта при неповерненні їм узятого кредиту. Дійсно, величина наданого кредиту повинна ґрунтуватися не тільки на урахуванні вартості споруджуваного будинку, але і на власній капіталізації клієнта.

У якості четвертої вхідної змінної використовується оцінка доходу потенційного клієнта за винятком фіксованих витрат, яка також використовується у випадку неспроможності потенційного клієнта при неповерненні їм узятого кредиту. Чим вище значення цієї змінної, тим більше успішним є надання кредиту клієнту.

У якості п'ятої вхідної змінної використовується величина підлягаючих сплаті відсотків відповідно до передбачуваного плану виплат по узятому кредиту. Ця змінна зв'язана з терміном надання кредиту і його величиною, дозволяючи об'єднати в собі відповідні характеристики кредиту. Чим вище величина виплат по відсотках, тим більше високими повинні бути значення активів і доходів для позитивного рішення про надання кредиту потенційному клієнту.

У якості **вихідної** змінної використовується **оцінка кредитоспроможності**, що є основою для ухвалення рішення керівництвом банку по наданню кредиту потенційним клієнтам. При цьому рішення про надання кредиту керівництвом банку приймається тільки у випадку високої оцінки цієї вихідної змінної.

Аналіз надання кредитів на будівництво будинків показує, що для аналізу фінансової заможності потенційних клієнтів керівництво банків застосовує наступні евристичні правила:

1. Якщо величина доходу низька і величина виплат середня, то кредитоспроможність дуже низька.

2. Якщо величина доходу низька і величина виплат висока, то кредитоспроможність дуже низька.

3. Якщо величина доходу середня і величина виплат висока, то кредитоспроможність дуже низька.

4. Якщо активи низькі і величина доходу низька, то кредитоспроможність дуже низька.

5. Якщо активи низькі і величина доходу середня, то кредитоспроможність дуже низька.

6. Якщо активи середні і величина доходу низька, то кредитоспроможність дуже низька.

7. Якщо якість опоряджувальних робіт погана, активи низькі і величина доходу висока, то кредитоспроможність дуже низька.

8. Якщо якість опоряджувальних робіт погана, активи середні і величина доходу середня, то кредитоспроможність дуже низька.

9. Якщо якість опоряджувальних робіт погана, активи високі і величина доходу низька, то кредитоспроможність дуже низька.

10. Якщо якість опоряджувальних робіт погана, активи високі і величина доходу середня, то кредитоспроможність дуже низька.

11. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт гарна, активи низькі і величина доходу висока, то кредитоспроможність середня.

12. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт прекрасна, активи низькі і величина доходу висока, то кредитоспроможність середня.

13. Якщо місцезнаходження престижне, якість опоряджувальних робіт гарна, активи низькі і величина доходу висока, то кредитоспроможність середня.

14. Якщо місцезнаходження дуже престижне, якість опоряджувальних робіт гарна, активи низькі і величина доходу висока, то кредитоспроможність середня.

15. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт гарна, активи середні і величина доходу середня, то кредитоспроможність середня.

16. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт прекрасна, активи середні і величина доходу середня, то кредитоспроможність середня.

17. Якщо місцезнаходження престижне, якість опоряджувальних робіт гарна, активи середні і величина доходу середня, то кредитоспроможність середня.

18. Якщо місцезнаходження дуже престижне, якість опоряджувальних робіт гарна, активи середні і величина доходу середня, то кредитоспроможність середня.

19. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт гарна, активи високі і величина доходу низька, то кредитоспроможність середня.

20. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт прекрасна, активи високі і величина доходу низька, то кредитоспроможність середня.

21. Якщо місцезнаходження престижне, якість опоряджувальних робіт гарна, активи високі і величина доходу низька, то кредитоспроможність середня.

22. Якщо місцезнаходження дуже престижне, якість опоряджувальних робіт гарна, активи високі і величина доходу низька, то кредитоспроможність середня.

23. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт гарна, активи високі і величина доходу середня, то кредитоспроможність середня.

24. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт прекрасна, активи високі і величина доходу середня, то кредитоспроможність середня.

25. Якщо місцезнаходження престижне, якість опоряджувальних робіт гарна, активи високі і величина доходу середня, то кредитоспроможність середня.

26. Якщо місцезнаходження дуже престижне, якість опоряджувальних робіт гарна, активи високі і величина доходу середня, то кредитоспроможність середня.

27. Якщо місцезнаходження престижне, якість опоряджувальних робіт прекрасна, активи середні і величина доходу висока, то кредитоспроможність дуже висока.

28. Якщо місцезнаходження престижне, якість опоряджувальних робіт прекрасна, активи високі і величина доходу висока, то кредитоспроможність дуже висока.

29. Якщо місцезнаходження дуже престижне, якість опоряджувальних робіт прекрасна, активи середні і величина доходу висока, то кредитоспроможність дуже висока.

30. Якщо місцезнаходження дуже престижне, якість опоряджувальних робіт прекрасна, активи високі і величина доходу висока, то кредитоспроможність дуже висока.

31. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт гарна, активи середні і величина доходу висока, то кредитоспроможність висока.

32. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт прекрасна, активи середні і величина доходу висока, то кредитоспроможність висока.

33. Якщо місцезнаходження престижне, якість опоряджувальних робіт гарна, активи середні і величина доходу висока, то кредитоспроможність висока.

34. Якщо місцезнаходження дуже престижне, якість опоряджувальних робіт гарна, активи середні і величина доходу висока, то кредитоспроможність висока.

35. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт гарна, активи високі і величина доходу висока, то кредитоспроможність висока.

36. Якщо місцезнаходження непрестижне, якість опоряджувальних робіт прекрасна, активи високі і величина доходу висока, то кредитоспроможність висока.

37. Якщо місцезнаходження престижне, якість опоряджувальних робіт гарна, активи високі і величина доходу висока, то кредитоспроможність висока.

38. Якщо місцезнаходження дуже престижне, якість опоряджувальних робіт гарна, активи високі і величина доходу висока, то кредитоспроможність висока.

39. Якщо місцезнаходження престижне, якість опоряджувальних робіт прекрасна, то кредитоспроможність висока.

40. Якщо місцезнаходження дуже престижне, якість опоряджувальних робіт прекрасна, то кредитоспроможність висока.

Після розгляду змістовної постановки задачі можна приступити до побудови ее нечіткої моделі у формі відповідної системи нечіткого виводу. Для цієї мети скористаємося розглянутими раніше графічними засобами пакета Fuzzy Logic Toolbox системи MATLAB.

5.3 Розробка нечіткої моделі оцінювання фінансової заможності клієнтів

При побудові нечіткої моделі оцінки фінансової заможності потенційних клієнтів було зроблене припущення про те, що всі розглянуті змінні виміряються в балах в інтервалі дійсних чисел від 0 до 10. При цьому найнижча оцінка значення кожної із змінних є 0, а найвища — 10.

Фазифікація вхідних і вихідних змінних

Як терм-множину першої вхідної змінної *"Місцезнаходження"* (Location) будемо використовувати множину T1={*"непрестижне", "престижне", "дуже престижне"*} або в символічному виді T1={PS, PM, PB}.

Як терм-множину другої вхідної змінної *"Опоряджування"* (Workmanship) будемо використовувати аналогічну множину T2={*"погане", "гарне", "прекрасне"*} або в символічному виді T2={PS, PM, PB}.

Як терм-множину третьої лінгвістичної змінної *"Активи"* (Asset) будемо використовувати множину T3={*"низькі", "середні", "високі"*} чи в символічному виді T3={PS, PM, PB}.

Як терм-множину четвертої лінгвістичної змінної *"Доход"* (Income) будемо використовувати аналогічну множину T4={*"низький", "середній", "високий"*} чи в символічному виді T4={PS, PM, PB}.

Як терм-множину п'ятої лінгвістичної змінної *"Виплати"* (Interest) будемо використовувати аналогічну множину T5= {*"низькі", "середні", "високі"}* чи в символічному виді T5={PS, PM, PB}.

Як терм-множину вихідної лінгвістичної змінної *"Кредитоспроможність"* (Credit) будемо використовувати множину T6={*"дуже низька", "низька", "середня", "висока", "дуже висока"*} чи в символічному виді T6={NB, NS, Z, PS, PB}.

5.4 Формування бази правил систем нечіткого виводу

Наступним етапом побудови моделі є побудова бази правил. Для цієї мети будемо використовувати 40 правил нечітких продукцій, які зручно представити у виді наступної таблиці 5.1.

Таблиця 5.1. *Правила нечітких продукций для розглянутої системи нечіткого виводу*

№ правила	Місцезнаходження	Опоряджування	Активи	Доход	Виплати	Кред

Як схему нечіткого виводу будемо використовувати метод Мамдані, тому методом активації буде MIN. Далі необхідно визначити методи агрегувания підумов. Оскільки у всіх правилах 1 — 40 як логічне зв'язування для підумов застосовується тільки нечітка кон'юнкція (операція "I"), то як метод агрегування будемо використовувати операцію min-кон'юнкції. Для акумуляції виводів правил будемо використовувати метод max-дизъ'юнкції, що також застосовується у випадку схеми нечіткого виводу методом Мамдані. Нарешті, як метод дефазифікації будемо використовувати метод центра ваги.

5.5 Побудова нечіткої моделі засобами Fuzzy Logic Toolbox і аналіз отриманих результатів

Розробку нечіткої моделі (назвемо її mortgage) будемо виконувати з використанням графічних засобів системи MATLAB. З цією метою в редакторі FIS визначимо 5 вхідних змінних з іменами *"місцезнаходження"* (β1), *"опоряджування"* (β2), *"активи"* (β3), *"доход"* (β4), *"виплати"* (β5) і одну вихідну змінну з ім'ям *"кредитоспроможність"* (β6). Вид графічного інтерфейсу редактора FIS для цих змінних зображений на рис.5.1.

Для вирішення поставленої задачі нечіткого моделювання будемо використовувати систему нечіткого виводу типу Мамдані. Залишимо без зміни параметри розроблювальної нечіткої моделі, запропоновані системою MATLAB за замовчуванням, а саме, логічні операції (**min** — для нечіткого логічного І, **max** — для нечіткого логічного ЧИ), метод імплікації (**min**), метод агрегування (**max**) і метод дефазифікації (**centroid**).

Далі варто визначити функції належності термів для кожної з 5 вхідних і єдиної вихідної змінних розглянутої системи нечіткого виводу. Для цієї мети скористаємося редактором функцій належності системи

MATLAB. Графічний інтерфейс редактора функцій належності для вихідної змінної *"кредитоспроможність"* зображений на рис.5.2.

Далі задамо 40 правил для розроблювальної системи нечіткого виводу (табл. 5.1). Для цієї мети скористаємося редактором правил системи MATLAB. Вид графічного інтерфейсу редактора правил після завдання всіх 40 правил нечіткого виводу зображений на рис.5.6. Оскільки в робочому вікні відображаються не всі змінні нечіткої моделі, для керування режимом відображення змінних правил варто скористатися спеціальними кнопками, розташованими в нижній правій частині редактора правил.

Тепер можна виконати аналіз побудованої системи нечіткого виводу для розглянутої задачі оцінки фінансової заможності клієнтів. З цією метою відкриємо вікно перегляду правил системи MATLAB і введемо значення вхідних змінних для частинного випадку, коли значення вхідної змінної *"місцезнаходження"* оцінюється в 8 балів, значення вхідної змінної *"опоряджування"* також оцінюється в 8 балів, значення вхідної змінної *"активи"* оцінюється в 9 балів, значення вхідної змінної *"дохід"* оцінюється в 9 балів, і, нарешті, значення вхідної змінної *"виплати"* оцінюється в 5 балів. Це досить високі оцінки вхідних змінних, котрі навіть на інтуїтивному рівні свідчать на користь відповідного клієнта.

Процедура нечіткого виводу, виконана системою MATLAB для розробленої нечіткої моделі, видає в результаті значення вихідної змінної *"кредитоспроможність"*, рівне 7.75 бала (рис.5.4). Це досить висока оцінка фінансової заможності потенційного клієнта, що може служити підставою для позитивного рішення з боку банку про надання кредиту під заставу. Як можна заключити, даний висновок цілком узгоджується з раніше висловленими інтуїтивними розуміннями.

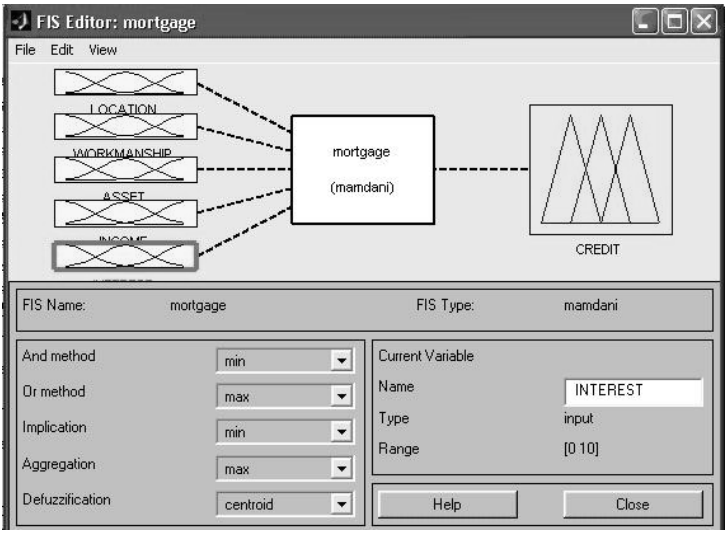

Рис. 5.1. Графічний інтерфейс редактора FIS після визначення вхідних і вихідної змінних системи нечіткого виводу mortgage

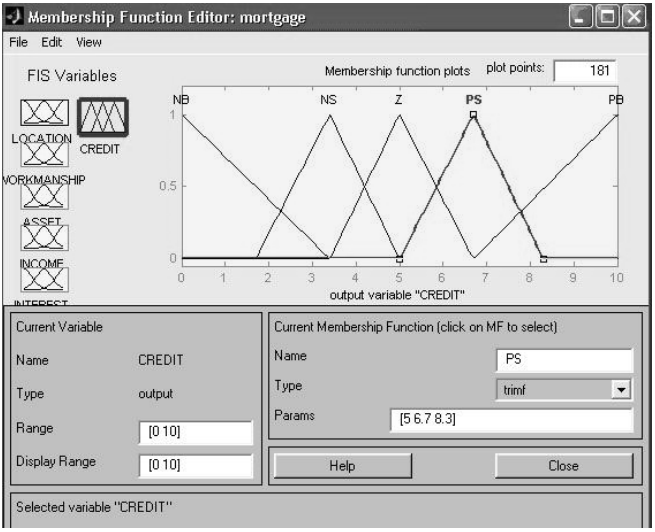

Рис. 5.2. Графічний інтерфейс редактора функцій належності для

вихідної змінної ***"кредитоспроможність"***

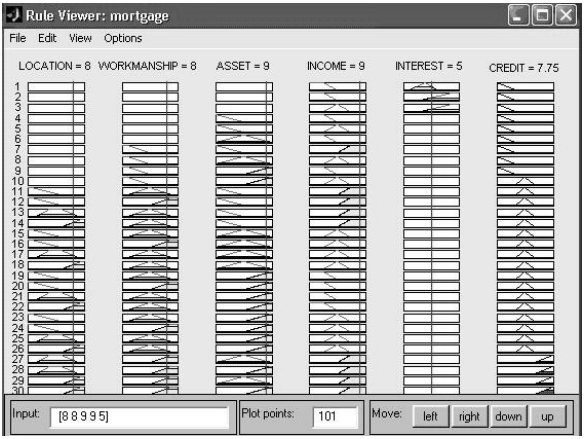

Рис. 5.3. Графічний інтерфейс програми перегляду правил після виконання процедури нечіткого виводу 1 варіанту
Ї

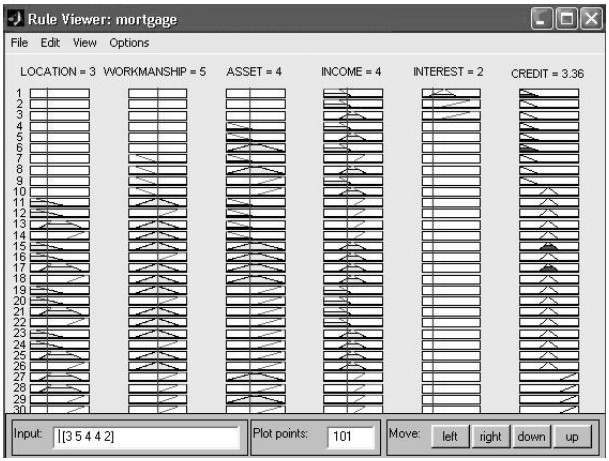

Рис. 5.4. Графічний інтерфейс програми перегляду правил після виконання процедури нечіткого виводу 2 варіанту

Далі виконаємо аналіз побудованої системи нечіткого виводу для другого варіанта вихідних даних з більш низькими оцінками значень вхідних змінних. З цією метою змінимо значення вхідних змінних: значення вхідної змінної *"місцезнаходження"* оцінимо в 3 бали, значення вхідної змінної *"опоряджування"* оцінимо в 5 балів, значення вхідної змінної *"активи"* оцінимо в 4 бали, значення вхідної змінної *"доход"* оцінимо в 4 бали, і, нарешті, значення вхідної змінної *"виплати"* оцінимо в 2 бали.

Процедура нечіткого виводу, виконана системою MATLAB, видає в результаті значення вихідної змінної *"кредитоспроможність",* рівне 3.42 бала. Це досить низька оцінка фінансової заможності потенційного клієнта, що може служити підставою для негативного рішення з боку банку про надання кредиту під заставу. Як можна заключити в цьому випадку, даний висновок також узгоджується з інтуїтивними розуміннями на цей рахунок.

Порівняння результатів нечіткого виводу для двох розглянутих варіантів значень вхідних змінних показує, що граничне значення вихідної змінної *"кредитоспроможність",* що впливає на рішення про надання кредиту, може бути обране в межах 5 балів.

Примітка

Для більш тонкого настроювання побудованої нечіткої моделі необхідно доповнити її конкретними методиками бальної оцінки окремих кількісних значень вхідних і вихідних лінгвістичних змінних. Оскільки такі методики в значній мірі залежать від розглянутої проблемної області, від сформованої на даний момент ринкової кон'юнктури і мають приватний характер для конкретного банку, тут вони не розглядаються.

Для загального аналізу розробленої нечіткої моделі також може виявитися корисною візуалізація відповідної поверхні нечіткого виводу (рис.5.5).

Дана поверхня нечіткого виводу дозволяє установити залежність значень вихідної змінної від значень окремих вхідних змінних нечіткої моделі. Аналіз цих залежностей може служити підставою для зміни функцій належності вхідних змінних чи нечітких правил з метою підвищення адекватності системи нечіткого виводу для конкретних стратегій банків.

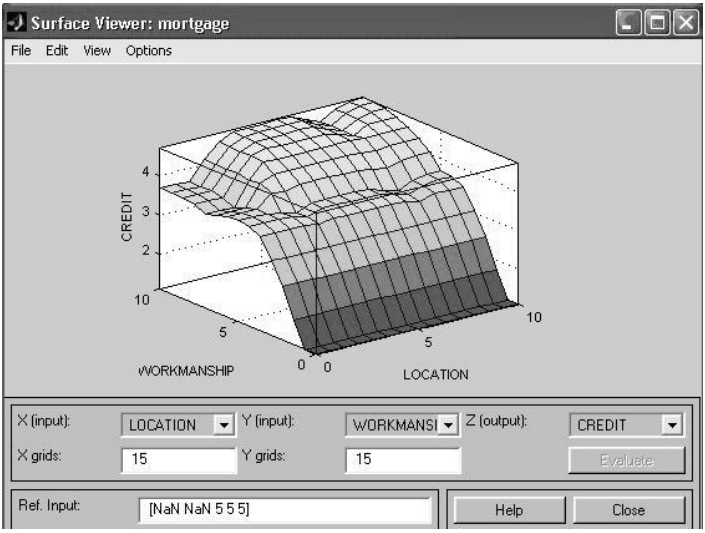

Рис. 5.5. Візуалізація поверхні нечіткого виводу розглянутої моделі для вхідних змінних *"місцезнаходження"* і *"обробка"*

Слід зазначити також ту обставину, що розроблювачі даної нечіткої моделі відзначають її декілька спрощений характер у порівнянні з реально використовуваною в процесі прийняття рішень керівництвом банку. У той же час розглянута нечітка модель має досить високу адекватність, що обумовлює її успішне застосування в практиці фінансових операцій деяких банків.

Таблиця 5.2 Встановлення ваги правил

№ правила	Вага	№ правила	Вага
1	0.51	11	0.28
2	0.75	12	0.97
3	0.24	13	0.32
4	0.14	14	0.71
5	0.77	15	0.47
6	0.33	16	0.69
7	0.79	17	0.96
8	0.16	18	0.74
9	0.86	19	0.86
10	0.98	20	0.11

№ правила	Вага	№ правила	Вага
21	0.53	31	0.05
22	0.09	32	0.03
23	0.81	33	0.49
24	0.70	34	0.68
25	0.06	35	0.27
26	0.58	36	0.40
27	0.78	37	0.59
28	0.91	38	0.14
29	0.17	39	0.75
30	0.34	40	0.59

§6. Лабораторна робота №5
РОЗРОБКА СИСТЕМ АНАЛІЗУ ДАНИХ МЕТОДАМИ НЕЧІТКОЇ КЛАСТЕРІЗАЦІЇ

Мета роботи: Освоїти методику проектування і побудови систем інтелектуального аналізу даних на основі методів нечіткої кластеризації.

6.1. Основні поняття

Кластерний аналіз – це сукупність методів, підходів і процедур, які розробляються для розв'язування проблеми формування класів – сукупностей даних, однорідних за заданими ознаками.

Кластерний аналіз (автоматична класифікація сукупності даних) займає одно з центральних місць серед методів аналізу даних і являє собою сукупність підходів та алгоритмів знаходження деякого розбиття досліджуваної сукупності об'єктів на підмножини відносно схожих між собою елементів. Такі підмножини отримали назву кластерів.

Виділення кластерів серед сукупності даних має відповідати наступним вимогам:

1. кожний кластер представляє собою сукупність об'єктів, які схожі між собою значеннями деяких властивостей або ознак;
2. сукупність всіх кластерів має бути вичерпаною, тобто всі об'єкти досліджуваної сукупності мають належить до деякого кластеру;
3. кластери мають бути взаємно-виключні; тобто, жоден з об'єктів не має належить до двох різних кластерів.

Формально, під задачею кластерного аналізу розуміється задача знаходження деякого теоретико-множинного розбиття початкової множити об'єктів на підмножини, які не перетинаються, таким чином, щоб

елементи, які відносяться до однієї підмножини відрізнялися між собою в значно меншій степені, ніж об'єкти з різних підмножин.

Концептуальний зв'язок між кластер ним аналізом і теорією нечітких множин оснований на тому, що при розв'язуванні задач структуризації складних систем більшість класів, що формуються, «розмиті» за своєю природою. Тому, найбільш адекватну відповідь слід шукати не на питання «Чи належить елемент до того чи іншого класу?», а на питання «В якій степені даний елемент належить класу, що розглядається?».

Методи нечіткої кластеризації вводять до розгляду нечіткі кластери і відповідні їм функції належності, які приймають значення з інтервалу $[0,1]$.

Таким чином, задача нечіткої кластеризації полягає у тому, що необхідно знайти нечітке розбиття або нечітке покриття множини елементів сукупності, що досліджується. Задача зводиться до знаходження степенем належності елементів множини нечітким кластерам (класам).

6.2. Постановка задачі

Нехай початкова (досліджувана) сукупність даних представляє собою скінчену множину елементів $A = \{a_1, a_2, ..., a_n\}$, яке ще називається множиною об'єктів кластеризації. Вводиться також скінчена множина ознак або атрибутів об'єктів $P = \{p_1, p_2, ..., p_q\}$, кожний з яких являє собою деяку характеристику елементів множини A.

Далі, пропонується, що для всіх елементів множини об'єктів кластеризації виміряли всі ознаки множини P, і кожен елемент множини $a_i \in A$ представлений вектором $x_i = \{x_i^1, x_i^2, ..., x_i^q\}$, де $x_i^j \in R$ - дійсне значення ознаки $p_j \in P$ для об'єкту $a_i \in A$.

Взагалі, проблема кількісного вимірювання ознак кожного об'єкта з сукупності – нетривіальна і самостійна задача. Процес вимірювання ознак може бути реалізований в різних шкалах, кожна з яких характеризується допустимим перетворенням даних. В зв'язку з цим, визначають різні типи шкал:

- шкала найменувань: об'єкту ставиться у відповідність деякий символ або номер, який лише відокремлює одне значення ознаки від іншого; прикладом таких ознак є стать людини –(м, ж) або найменування міст (Київ, Житомир, Луганськ,...); допустимим відображенням множини об'єктів у множину символів є бієктивне відображення;
- порядкова шкала: разом з відповідною множиною символьних ознак об'єктів ця шкала дозволяє встановити відношення порядку відносно цієї ознаки; тоді об'єкту ставиться у відповідність деяке число, яке грає роль його оцінки в балах; допусти-

мим відображенням є монотонне зростаюче відображення або функція між двома множинами значень ознак; приклад – оцінки на іспитах;

- інтервальна шкала: крім порядку елементів по ознакам ця шкала встановлює рівність інтервалів значень цієї ознаки; об'єкту, як правило, ставиться у відповідність число, яке дорівнює значенню цієї ознаки; допустимим перетворенням тут є довільна лінійна зростаюча функція між двома множинами значень ознак; характерною ознакою такої шкали є відсутність абсолютного нуля; приклад – температура в шкалах Цельсія;
- шкала відношень: в доповнення до рівності інтервалів додає ще рівність відношень значень ознаки, що розглядається; об'єкту ставиться у відповідність деяке число, яке дорівнює значенню цієї ознаки; допустимим відображенням є довільна лінійна зростаюча функція, яка проходить через нуль; приклад – відстань в метрах, швидкість в км/ч.

Множину ознак слід обирати таким чином, щоб всі $x_i^j \in R$ були виміряні в шкалах відношень чи інтервалів. Саме в такому випадку результати нечіткої кластеризації мають змістовну інтерпретацію, яка адекватна проблемі знаходження нечітких кластерів.

Вектори значень ознак $x_i = \left\{x_i^1, x_i^2, ..., x_i^q\right\}$ зручно представляти у вигляді матриці даних D розмірності (n*q), кожний рядок якої представляє собою значення вектору x_i .

Отже, **задача нечіткого кластерного аналізу формулюється наступним чином**: на основі даних матриці D визначити таке нечітке розбиття $R(A) = \left\{A_k \mid A_k \subseteq A\right\}$ або нечітке покриття $J(A) = \left\{A_k \mid A_k \subseteq A\right\}$ множини A на задане число нечітких кластерів $A_k (k \in \left\{2,..., c\right\})$, яке доставляє екстремум деякій цільовій функції $F(R(A))$ серед всіх нечітких розбиттів чи екстремум цільової функції $F(J(A))$ серед всіх можливих нечітких покриттів.

Для конкретизації задачі ще слід уточнити вигляд цільової функції та тип шуканих нечітких кластерів.

Одним з видів конкретизації цієї задачі є використання спеціальної функції fcm системи MATLAB, який оснований на алгоритмі розв'язування методом нечітких с-середніх.

Для уточнення вигляду цільової функції $F(J(A))$ вводяться деякі додаткові поняття. По-перше, пропонується, що шукані нечіткі кластери представляють собою нечіткі множини $A_k (k \in \left\{2,..., c\right\})$, які є нечітким

покриттям початкової множини об'єктів кластеризації A, для якої має місце наступна умова:

$$\sum_{k=1}^{c} \mu_{A_k}(a_i) = 1, (\forall a_i \in A), \qquad (6.1)$$

де c – загальна кількість нечітких кластерів $A_k (k \in \{2,..., c\})$, яке вважається попередньо заданим.

Далі для кожного кластеру вводяться так звані типові представники або **центри** v_k шуканих нечітких кластерів $A_k (k \in \{2,..., c\})$, які розраховуються за наступною формулою:

$$v_j^k = \frac{\sum_{i=1}^{n} (\mu_{A_k}(a_i))^m x_i^j}{\sum_{i=1}^{n} (\mu_{A_k}(a_i))^m}, (\forall k \in \{2,..., c\}, \forall p_j \in P), \qquad (6.2)$$

де m – деякий параметр, який має назву **експоненційна вага** і дорівнює деякому дійсному числу (m>1). Кожний з центрів кластерів є вектором $v_k = (v_k^1, v_k^2, ..., v_k^q)$ в деякому q-вимірному нормованому просторі ознак, який ізоморфний R^q, якщо всі ознаки виміряні по шкалі відношень.

В якості цільової функції будемо розглядати суму квадратів зважених відхилень координат об'єктів кластеризації від центрів нечітких кластерів:

$$F(A_k, v_k^j) = \sum_{i=1}^{n} \sum_{k=1}^{c} (\mu_{A_k}(a_i))^m \sum_{j=1}^{q} (x_i^j - v_k^j)^2 . \qquad (6.3)$$

Чим більше елементів містить множина A, тим менше значення слід вибирати для m>1.

Тоді задача нечіткої кластеризації полягає у наступному: для заданої матриці даних D, кількості нечітких кластерів $c \in N, c > 1$, параметра m, визначити матрицю U значень функції належності об'єктів кластерізації $a_i \in A$ нечітким кластерам $A_k (k \in \{2,..., c\})$, які доставляють мінімум цільової функції (6.3) і задовольняють обмеженням (6.1)-(6.2), а також додатковим обмеженням:

$$\sum_{i=1}^{n} \mu_{A_k}(a_i) > 0, (\forall k = \{2,..., c\})$$
$$\mu_{A_k}(a_i) \geq 0 (\forall k = \{2,..., c\}, a_i \in A) \qquad (6.4).$$

Умови (4) виключають появу пустих нечітких кластерів в шуканій нечіткій кластеризації. Таким чином, мінімізація цільової функції (6.3) мінімізує відхилення всіх об'єктів кластеризації від центрів нечітких кластерів пропорційно значенням функцій належності цих об'єктів відповідним нечітким кластерам.

Ця функція не є випуклою, а тому задача кластеризації в загальному випадку відноситься до багатоекстремальних задач нелінійного програмування.

6.3. Алгоритм розв'язування задачі нечіткої кластеризації

Основні ідеї алгоритму для розв'язування задачі нечіткої кластеризації були запропоновані Дж.К.Данном у 1974р. Цей алгоритм спочатку отримав назву нечіткого алгоритму fuzzyISODATA. У 1980 році Дж.К. Беджек теоретично довів збіжність цього алгоритму, потім він же узагальнив цей алгоритм на випадок довільних нечітких множин даних і запропонував для цього алгоритму назву нечітких середніх FCM, Fuzzy-C-Means. Саме під такою назвою алгоритм реалізований в системі MATLAB.

Алгоритм FCM має ітеративний характер послідовного покращення деякого початкового нечіткого розбиття $R(A) = \{A_k \mid A_k \subseteq A\}$, яке задається користувачем або формується автоматично за деяким евристичним правилом. На кожному кроці ітерації прораховуються значення функцій належності нечітких кластерів і їх типових представників.

Алгоритм FCM завершує роботу у випадку, коли відбудеться наперед задане число ітерацій, або, коли мінімальна абсолютна різниця між значеннями функцій належності на двох послідовних ітераціях не стане менше деякого наперед заданого значення.

Формально алгоритм FCM представляється у вигляді наступних кроків:

1. попередньо необхідно задати наступні значення: кількість шуканих нечітких кластерів c, максимальну кількість ітерацій алгоритмів $s \in N$, параметр збіжності алгоритму $\varepsilon \in R_+$, а також екпоненційну вагу для цільової функції і центрів кластерів m>1. В якості початкового розбиття на першій ітерації алгоритму для матриці даних D задати деяке нечітке розбиття $R(A) = \{A_k \mid A_k \subseteq A\}$ на c непустих нечітких кластерів, які описуються сукупністю функцій належності $\mu_k(a_i), \forall k \in \{2,..., c\}, \forall a_i \in A$.
2. для поточного нечіткого розбиття $R(A) = \{A_k \mid A_k \subseteq A\}$ за формулою (6.2) розрахувати центри нечітких кластерів

$v_k^j, (\forall k = \{2,..., c\}, \forall p_j \in P$ і значення цільової функції (6.3). Кількість виконаних ітерацій покласти 1.

3. сформувати нове нечітке розбиття $R'(A) = \{A_k \mid A_k \subseteq A\}$ множини об'єктів кластеризації A на c непусті нечіткі кластери, які характеризуються сукупністю функцій належності $\mu_k^1(a_i), \forall k \in \{2,..., c\}, \forall a_i \in A$, що визначаються за формулою:

$$\mu_k^1(a_i) = \left(\sum_{l=1}^{c} \left(\frac{\left(\sum_{j=1}^{q} (x_i^j - v_k^j)^2 \right)^{\frac{1}{2}}}{\left(\sum_{j=1}^{q} (x_i^j - v_l^j)^2 \right)^{\frac{1}{2}}} \right)^{\frac{2}{m-1}} \right)^{-1}, \forall k \in \{2,..., c\}, \forall a_i \in A$$

4. При цьому, якщо для деякого $k \in \{2,..., c\}$ і деякого $a_i \in A$ значення $\sum_{j=1}^{q} (x_i^j - v_k^j)^2 = 0$, тоді для відповідного нечіткого кластеру $\mathrm{A_k}$ беремо $\mu_k^1(a_i) = 1$, а для інших кластерів $A_l (\forall l = \{2,..., c\}, l \neq k)$ беремо $\mu_l^1(a_i) = 0$. Якщо ж таких значень $k \in \{2,..., c\}$ виявиться декілька для $a_i \in A$, тоді евристично беремо $\mu_{\mathrm{k}}^1(\mathrm{a_i}) = 1$ для меншого з них, а для інших $\mu_l^1(a_i) = 0$.
5. Для нового нечіткого розбиття $R'(A) = \{A_k \mid A_k \subseteq A\}$ за формулою (6.2) розраховуємо центри нечітких кластерів і значення цільової функції (6.3).
6. Якщо кількість виконаних ітерацій більше за s або модуль різниці між попереднім і новим значенням цільової функції менше за $\varepsilon \in R_+$, тоді в якості результату прийняти нечітке розбиття $R'(A) = \{A_k \mid A_k \subseteq A\}$ і завершити виконання алгоритму. Інакше, вважати поточним розбиттям $R(A) = R'(A)$ і перейти на крок 3, збільшивши на 1 кількість виконаних ітерацій.

В результаті виконання алгоритм зводиться до деякого локально-оптимального розбиття $R^*(A)$, яке описується сукупністю функцій належності $\mu_k(a_i)$, а також центрами нечітких класів $v_k = (v_k^1, v_k^2, ..., v_k^q)$.

6.4. Виконання алгоритму FCM в системі MATLAB

Функція fcm може бути викликана в одному з наступних форматів:
$[center, U, obj\_fcn] = fcm(data, cluster\_n)$
або
$[center, U, obj\_fcn] = fcm(data, cluster\_n, options)$.
Вхідними аргументами цієї функції є

- data: матриця початкових даних D, і-тий рядок якої являє собою інформацію про об'єкт нечіткої кластеризації $a_i \in A$ у формі вектора $x_i = \{x_i^1, x_i^2, ..., x_i^q\}$;
- cluster_n: число шуканих кластерів $c \in N, c > 1$.

Вихідними аргументами цієї функції є

- center: матриця центрів шуканих нечітких кластерів, кожний рядок якої являє собою координати центру одного з нечітких кластерів в формі вектора $v_k = (v_k^1, v_k^2, ..., v_k^q)$;
- U: матриця значень функцій належності шуканого нечіткого розбиття $\mu_k(a_i), \forall k \in \{2, ..., c\}, \forall a_i \in A$;
- obj_fun: значення цільової функції (3) на кожній з ітерацій роботи алгоритму.

Функція fcm() може бути викликана з додатковими аргументами options, які введенні для управління процесом кластеризації, а також для зміни критерію останова роботи алгоритму і/або відображення інформації на екрані монітора.

Ці додаткові аргументи мають наступні значення:

-option (1): експоненційна вага m для розрахунків матриці нечіткого розбиття U (за замовченням $m = 2$);

-option (2): максимальне число ітерацій s (за замовченням це значення дорівнює 100);

-option (3): параметр збіжності алгоритму ε (за замовченням це значення дорівнює 0.00001);

-option (4): інформація про поточну ітерацію, яка відображається на екрані монітора (за замовченням, це значення 1).

Якщо будь-яке зі значень додаткових аргументів дорівнює NaN (не число), тоді для цього аргументу використовується значення за замовченням.

6.5. Приклад реалізації алгоритму

Завдання 1. В якості прикладу застосування нечіткої кластеризації розглянемо множину даних, які містяться в системі MATLAB і використовуються в якості текстової сукупності об'єктів нечіткої кластеризації. Ці дані являють собою матрицю D розмірності 140x2 і містяться у файлі fcmdata.dat, який поставляється разом зі MATLAB В даному випадку матриця D відповідає 140 об'єктам, для кожного з який виконане вимірювання за двома ознаками, що є дуже зручним для візуалізації результатів нечіткої кластеризації в двовимірному просторі на площині.

1. Для візуалізації цих даних слід виконати наступні команди:

Load fcmdata.dat

plot(fcmdata(:,1), fcmdata(:,2), 'o')

На екрані з'явиться графічне зображення, яке представлене на рис. 6.1.

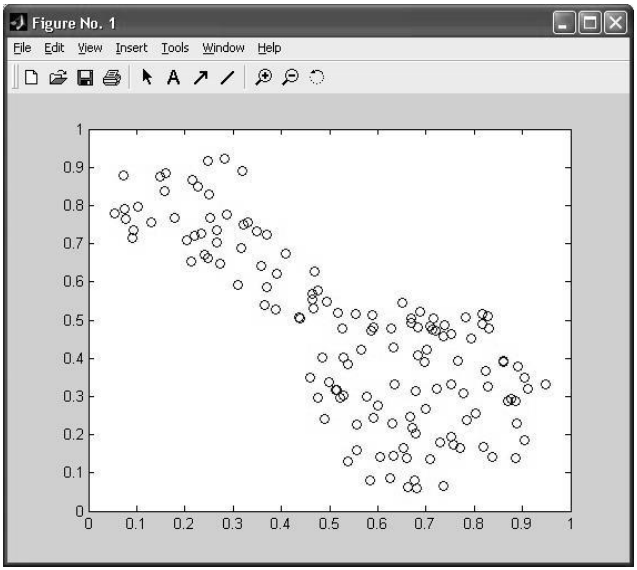

Рис.6.1 Зображення точок матриці D з файлу fcmdata.dat

2. Далі слід викликати функцію fcm, наприклад, з наступним форматом:

[center, U, obj_fcn]=fcm(fcmdata, 2)

Потім слід подивитись результати виконання процедури нечіткої кластеризації:

- координати центрів класів, тобто, матрицю center;
- належність кожної сукупності даних до класів – матрицю U;
- значення функції цілі – obj_fcn.

Взагалі, процедуру нечіткої кластеризації можна записати у вигляді командного М-файла з наступним текстом:

```
load fcmdata.dat
plot(fcmdata(:,1), fcmdata(:,2),'o')
[center, U, obj_fcn]= fcm(fcmdata, 2);
maxU=max(U);
index1 = find(U(1,:)== maxU);
index2 = find(U(2,:)== maxU);
line(fcmdata(index1,1), fcmdata(index1,2),'linestyle','none',…
'marker', 'x', 'color', 'g');
line(fcmdata(index2,1), fcmdata(index2,2),'linestyle','none',…
'marker', 'x', 'color', 'r');
hold on
plot( center(1,1), center(1,2),'ko', 'markersize',10, 'LineWidth', 2)
plot( center(2,1), center(2,2),'ko', 'markersize',10, 'LineWidth', 2)
```

Результатом цієї програми буде розбиття даних на два кластера, візуалізація якого зображена на рис.6.2.

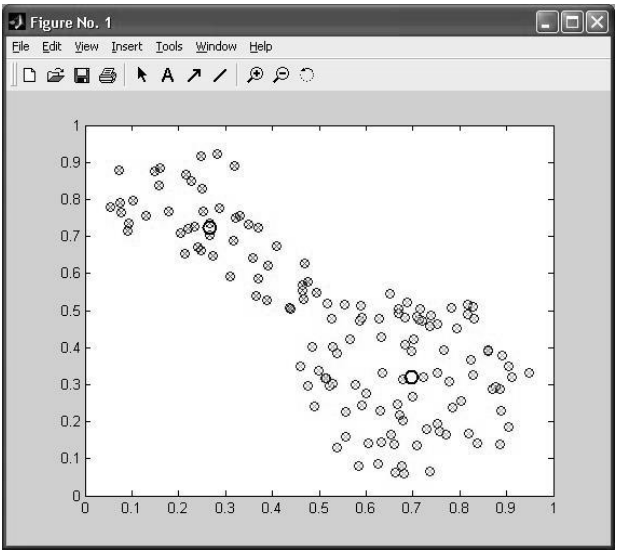

Рис. 6.2. Результат роботи програми нечіткої кластерізації

3. Після роботи програми в системі MATLAB можна перевірити значення матриць center та U, набравши їх назву в командному рядку і натиснувши Enter.

4. Крім того, в програмі можна використати наступний формат запису:

[center, U, obj_fcn]= fcm(fcmdata, 2, [2.5 1000 0.000001 1]);

В цьому випадку експоненційна вага 2.5, максимальне число ітерацій 1000, параметр збіжності $\varepsilon = 0.000001$. Порівняльний аналіз показує практичну ідентичність графіків – результатів використання обох форматів функції fcm, що дозволяє зробити висновок про відповідність отриманих результатів нечіткої кластеризації.

5. Для розв'язування задачі нечіткої кластеризації в системі MATLAB можна використовувати графічний інтерфейс, який викликається за допомогою команди **findcluster**. Ця програма може використовувати або метод с-середніх або метод субтрактивної кластеризації (substractive clustering, який викликається і окремо за допомогою команди **subclust**). Останній використовується тоді, коли не можна заздалегідь встановити число кластерів с на кроці 6.

Формат виклику графічного інтерфейсу: **findcluster або findc?uster('file.dat')**.

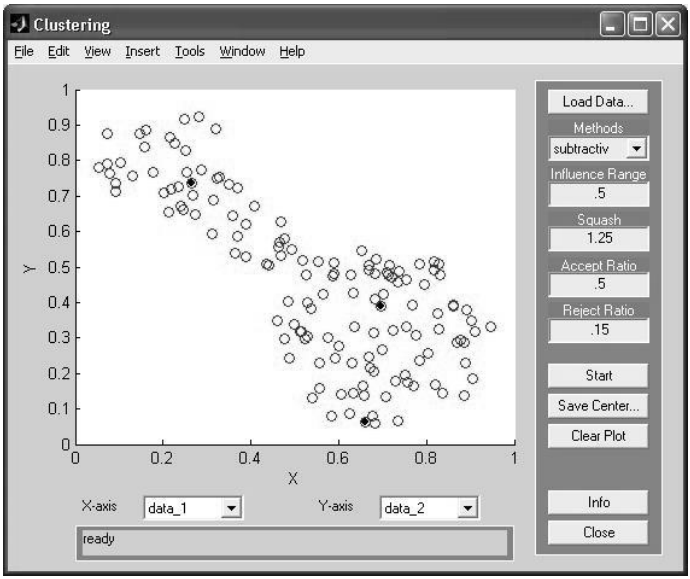

Рис 6.3. Вікно роботи графічного інтерфейсу нечіткої кластеризації для алгоритму субтрактивної кластеризації

В даному вікні можна завантажити файл даних Load Data, обрати метод кластеризації Methods, обрати необхідні значення параметрів і натиснути кнопку Start.

6. Нехай, заздалегідь невідома кількість кластерів с. Тоді слід використати метод субтрактивної кластеризації. Ідея цього методу полягає у тому, що кожна точка даних пропонується в якості центра потенційного кластеру. Далі слід вирахувати деяку міру можливості кожної точки даних представляти центр кластеру. Ця кількісна міра основана на оцінці густини точок навколо відповідного центра кластера.

Цей алгоритм, який є узагальненням методу кластеризації Р. Ягера, заснований на виконанні наступних кроків:

1) вибрати точку даних з максимальним потенціалом для представлення центру першого кластеру
2) забрати всі точки даних в околі центру даного кластеру, величина якої задається параметром **radii**, щоб визначити наступний нечіткий кластер і координати його центру.

Далі, ці дві процедури повторюються до тих пір, допоки всі точки даних не будуть лежати в границях околів радіуса **radii** навколого шуканих кластерів.

Функція командного рядка

[C, S] = subclust (X, radii, xBounds, options)

знаходить центри таких кластерів.

При цьому матриця X містить об'єкти кластеризації, кожний рядок якої відповідає координатам окремої точки даних. Параметр radii являє собою вектор, компоненти якого приймають значення з інтервалу [0,1] і задають діапазон розрахунку центрів кластерів по кожній з розказ вимірювань об'єктів. Робиться припущення, що всі дані знаходяться в деякому гіперкубі. Взагалі, малі значення параметрів radii призводять до знаходження малого числа великих по кількістю точок кластерів. Найкращих результатів можна очікувати при значенні radii між 0.2 і 0.5.

Аргумент xBounds являє собою матрицю розміру (2xq), яка визначає засіб відображення матриці даних X в деякому одиничному гіперкубі. Тут q – кількість ознак. Цей аргумент є необов'язковим, якщо матриця X вже нормована. Перший рядок цієї матриці містить мінімальні значення інтервалу вимірювання кожної ознаки, а другий рядок – максимальне значення вимірювання кожної ознаки.

Для зміни значень, які встановлені по замовченню, можна використати параметр options, компоненти вектора якого можуть приймати наступні значення:

- options(1) = guashFactor – параметр, який використовується в якості коефіцієнту для множення значень radii з ціллю зменшення впливу потенціалу граничних точок, які розглядаються як частина даного кластеру (за замовченням це значення дорівнює 1.25);
- options(2) = acceptRatio – параметр, який встановлює потенціал як частину потенціалу першого кластеру, вище якого інша точка даних не

може розглядатся в якості центра іншого кластеру (за замовченням це значення 0.5);

- option(3) = rejectRation - параметр, який встановлює потенціал як частину потенціалу першого кластеру, нижче якого інша точка даних не може розглядатся в якості центра іншого кластеру (за замовченням це значення 0.15);
- options(4) = verbose – якщо значення цього параметра не дорівнює 0, тоді на екран монітору виводиться інформація про виконання процесу кластеризації (за замовченням це значення 0).

Функція subclust повертає матрицю C значень координат центрів нечітких кластерів. При цьому кожний рядок цієї матриці містить координати одного центру кластеру. Вектор S містить σ-значень, які визначають діапазон впливу центра кластеру по кожній з розглянутих ознак. При цьому, всі центра кластерів мають однакову множину σ-значень.

Для **прикладу** розглянемо наступну послідовність команд:

Load fcmdata.dat

[C, S] = subclust(fcmdata, [0.5 0.5], [], [1.25 0.5 0.15 1])

Для кожної ознаки вводяться радіуси околів – 0.5 і 0.5.

Як видно з рис. 3. для даної сукупності даних функція suclust має знайти три кластери.

Для розв'язування задачі субтрактивної кластеризації може використовуватися і розглянутий раніше графічний інтерфейс, який викликається командою **findcluster**.

Таким чином, система MATLAB дозволяє розв'язувати задачі нечіткої кластеризації двома способами: за допомогою функцій командного рядка і за допомогою графічного інтерфейсу кластеризації. Результати нечіткої кластеризації мають наближений характер і мають служити для попередньої структуризація вхідної (початкової) інформації про систему, що вивчається. Тобто, провівши кластерізацію, можна отримати і зберегти знання про систему у вигляді структуризації початкової інформації.

6.6. Завдання для самостійної роботи

1. Виконати нечітку кластеризації набору точок в двовимірному просторі ознак: (1,2); (2,1); (1, 2); (3,2); (2,3); (1,6); (2,5); (3,6); (5,5); (5,6); (7,5); (7,6). Розділити їх на 3 кластери. Відповідь можна подивитися на рис 6.4.

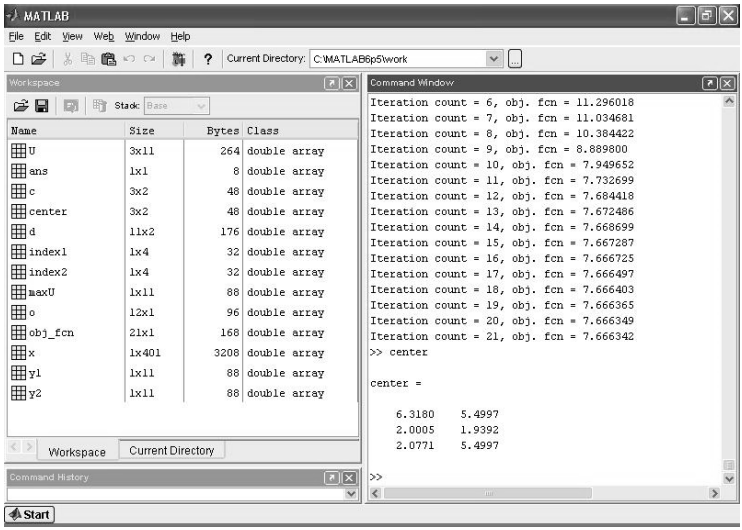

Рис 6.4. Точки і центри кластерів завдання 1

2. Виконати нечітку кластерізацію 10 фірм, які можуть в майбутньому опинитися в різних зонах своєї діяльності на ринку (2, 3 або 4 зони). Для цього слід використовувати дві ознаки стану фірм – середній дохід і середнє-квадратичне відхилення можливого доходу, тобто, ризикованість. Відомі наступні дані про фірми:

Фірма	Показники					
1	Можливий дохід	100	120	300	250	200
	Імовірність	0,1	0,3	0,2	0,2	0,2
2	Можливий дохід	120	400	200	150	300
	Імовірність	0,2	0,4	0,1	0,2	0,1
3	Можливий дохід	200	250	260	150	300
	Імовірність	0,3	0,3	0,2	0,1	0,1
4	Можливий дохід	120	100	200	150	130
	Імовірність	0,2	0,4	0,2	0,2	0,2
5	Можливий дохід	520	400	250	450	300
	Імовірність	0,6	0,1	0,1	0,1	0,1
6	Можливий	520	400	200	450	300

	дохід					
	Імовірність	0,2	0,4	0,1	0,2	0,1
7	Можливий дохід	220	300	200	150	300
	Імовірність	0,2	0,3	0,2	0,2	0,1
8	Можливий дохід	100	300	200	150	300
	Імовірність	0,2	0,2	0,3	0,1	0,2
9	Можливий дохід	230	400	200	250	200
	Імовірність	0,1	0,3	0,2	0,2	0,2
10	Можливий дохід	220	120	200	350	400
	Імовірність	0,1	0,5	0,1	0,2	0,1

Яка можлива економічна інтерпретація цих зон?

§7. Лабораторна робота №6
РОЗРОБКА ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ НА ОСНОВІ МОДЕЛЕЙ НЕЙРОНИХ МЕРЕЖ

Мета роботи: освоїти методику проектування і побудови інтелектуальних систем на основі моделей нечітких нейронних мереж.

7.1. Нейроні мережі в MATLAB

Концептуальною основою і складовою частиною штучних нейронних мереж є так званий *штучний нейрон,* що має визначену внутрішню структуру (рис. 7.1) і правила перетворення сигналів (7.1).

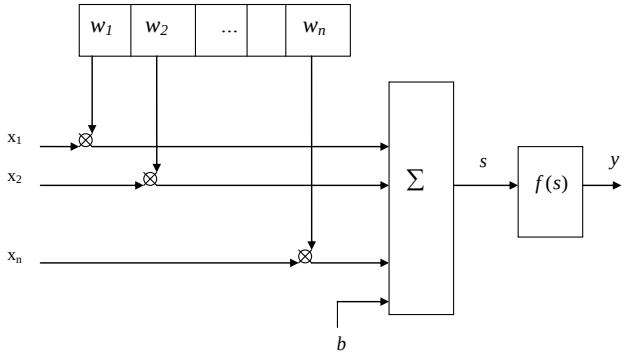

Рис. 7.1. Структура штучного нейрона

Штучний нейрон (далі просто — нейрон) складається з помножувачів (синапсів), суматора і нелінійного перетворювача. *Синапси,* зображені перекресленим кружком, призначені для зв'язку нейронів між собою і множать вхідний сигнал x_i на деяке постійне число. Це число w_i, назване *вагою синапса,* характеризує силу цього зв'язку. Суматор виконує додавання всіх сигналів, що надходять на вхід нейрона від інших нейронів, і зовнішніх вхідних сигналів. *Нелінійний перетворювач* призначений для нелінійної зміни вихідного значення суматора відповідно до деякої функції від одного аргументу. Ця функція називається *функцією активації* чи *передатною функцією* нейрона.

Правила перетворення сигналів визначаються математичною моделлю нейрона, що може бути записана у формі наступних аналітичних виразів:

$$s = \sum_{i=1}^{n} w_i \cdot x_i + b; \qquad y = f(s), \tag{7.1}$$

де w_i — вага синапса ($i \in \{1,2,\ldots,n\}$); b — значення зсуву; s — результат підсумовування; x_i — компонент вектора чи входу вхідного сигналу $(i \in \{1,2,\ldots,n\})$; y — вихідний сигнал нейрона; п — число входів нейрона; f — функція активації (передатна функція) нейрона, що представляє собою деяке нелінійне перетворення. У загальному випадку: $w_i, x_i, b \in \Re (i \in \{1,2,\ldots,n\})$.

Синаптичні зв'язки з позитивними вагами: $w_i \in \Re_+ (i \in \{1,2,\ldots,n\})$ називаються збуджуючими, а з негативними вагами $w_i \in \Re\_ (i \in \{1,2,\ldots,n\})$ гальмуючими.

Таким чином, окремо узятий штучний нейрон цілком описується своєю структурою (див. рис. 7.1) і математичною моделлю (7.1). Одержавши вектор вхідного сигналу хі, нейрон видає деяке число у на своєму виході.

Як функція активації нейрона можуть бути використані різні нелінійні перетворення (табл. 7.1).

Таблиця .7.1. Основні види функцій активації нейронів

Назва	Область значень	Формула
Лінійна	$\Re$	$f(s) = k \cdot s$
Напівлінійна	$\Re_+$	$f(s) = \begin{cases} 0, & якщо \ s \le 0 \\ k \cdot s, & якщо \ s > 0 \end{cases}$
Гранична	$\{0, 1\}$	$f(s) = \begin{cases} 0, & якщо \ s < T \\ 1, & якщо \ s \ge T \end{cases}$
Модульна	$\Re_+$	$f(s) = \lvert s \rvert$
Знакова (сигнатурна)	$\{-1, 1\}$	$f(s) = \begin{cases} -1, & якщо \ s \le T \\ 1, & якщо \ s > T \end{cases}$
Квадратична	$\Re_+$	$f(s) = s^2$
Експонентна	$\Re_+$	$f(s) = e^{-a \cdot s}$
Синусоїдальна	[-1, 1]	$f(s) = \sin(s)$
Логістична (сигмоїдальн	(0, 1)	$f(s) = \frac{1}{1 + e^{-a \cdot s}}$
Раціональна (сигмоїдал	(-1, 1)	$f(s) = \frac{s}{a + \lvert s \rvert}$
Гіперболічний тангенс (сигмоїдальна)	(-1, 1)	$f(s) = \frac{e^{a \cdot s} - e^{-a \cdot s}}{e^{a \cdot s} + e^{-a \cdot s}}$
Лінійна з насиченням (крокова)	[-1, 1]	$f(s) = \begin{cases} -1, & якщо \ s \le -1 \\ s, & якщо \ -1 < s < 1 \\ 1, & якщо \ s \ge 1 \end{cases}$

$$f(s) = \begin{cases} -1, & якщо \ s \le -1 \\ s, & якщо \ -1 < s < 1 \\ 1, & якщо \ s \ge 1 \end{cases}$$

Напівлінійна з насиченн	[0, 1)	$f(s)=\begin{cases}0, & якщо\ \ s\le 0\\ s, & якщо\ \ 0<s<1\\ 1, & якщо\ \ s\ge 1\end{cases}$
Трикутна	[0, 1]	$f(s)=\begin{cases}1-\lvert s\rvert, & якщо\ \ \lvert s\rvert\le -1\\ 0, & якщо\ \ \lvert s\rvert>1\end{cases}$
Радіальна базисна (Гау	(0, 1]	$f(s)=e^{s^2}$

Нейронна мережа являє собою сукупність окремих нейронів, взаємозв'язаних між собою деяким фіксованим чином. При цьому взаємозв'язок нейронів визначається чи задається структурою *(топологією)* нейронної мережі. З точки зору топології нейронні мережі можуть бути повнозв'язними, багатошаровими і слабозв'язними. У загальному випадку *структура* багатошарової чи багаторівневої нейронної мережі може бути зображена в такий спосіб (рис. 7.2).

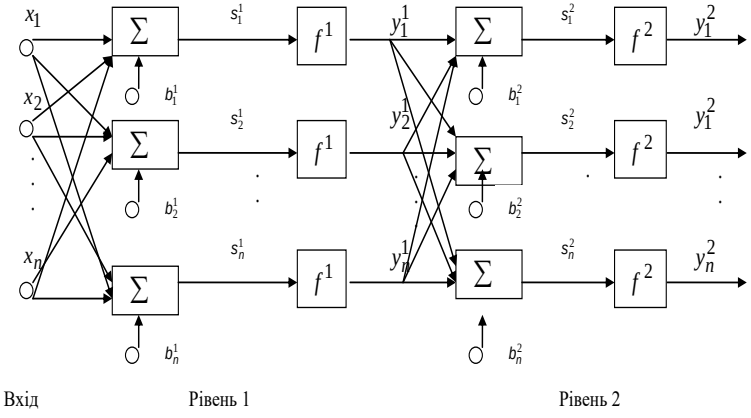

Рис. 7.2. Структура трирівневої нейронної мережі

Кожний з рівнів нейронної мережі називається її *шаром.* При цьому шар вхідного рівня називається *вхідним шаром,* шар рівня 1 і 2 — *схованими шарами,* а шар рівня 3 — *вихідним шаром.*

У свою чергу багатошарові нейронні мережі можуть бути наступних типів:

- Монотонні — кожен шар (крім вихідного) додатково розбивається на 2 блоки: збуджуючий і гальмуючий. Аналогічно розбиваються і зв'язки між блоками: на збуджуючі і гальмуючі. При цьому як функції активації можуть бути використані тільки монотонні функції (див. табл. 7.1).
- Нейронні мережі зі зворотними зв'язками — інформація з наступних шарів може передаватися на нейрони попередніх шарів.
- Нейронні мережі без зворотних зв'язків — інформація з наступних шарів не може передаватися на нейрони попередніх шарів.

Класичним варіантом багатошарової нейронної мережі є повнозв’язна мережа прямого поширення (рис. 7.2).

Процес побудови і використання нейро-мережевих моделей складається з наступних етапів:

1. Вибір типу і структури нейронної мережі для розв’язку поставленої проблеми (синтез структури нейронної мережі).

2. Навчання нейронної мережі (визначення чисельних значень ваг кожного з нейронів) на основі наявної інформації про розв’язок даної задачі експертом чи даних про розв’язок задачі в минулому.

3. Перевірка нейронної мережі на основі використання деякого контрольного приклада (необов'язковий етап).

4. Використання навченої нейронної мережі для розв’язку поставленої проблеми.

В даний час запропоновані різні схеми класифікації нейронних мереж і відповідні алгоритми їх навчання. Одним з найпоширеніших алгоритмів навчання є так званий алгоритм *зворотного поширення помилки* (back propagation). Цей алгоритм являє собою ітеративний градієнтний алгоритм мінімізації середньоквадратичного відхилення значень виходу від бажаних значень (мінімізації помилки) у багатошарових нейронних мережах.

Вибір виду і структури нейронної мережі визначається специфікою розв'язуваної задачі. При цьому для розв’язку окремих типів практичних задач розроблені оптимальні конфігурації нейронних мереж, що найбільше адекватно відбивають особливості відповідної проблемної області.

Подальшим розвитком нейронних мереж є так звані гібридні мережі, що реалізовані в пакеті Fuzzy Logic Toolbox системи MATLAB.

Гібридна мережа являє собою багатошарову нейронну мережу спеціальної структури без зворотних зв'язків, у якій використовуються звичайні (не нечіткі) сигнали, ваги і функції активації, а виконання операції підсумовування (7.1) засновано на використанні фіксованої Т-норми, Т-конорми чи деякої іншої неперервної операції. При цьому значення входів, виходів і ваг гібридної нейронної мережі являють собою дійсні числа з відрізка [0, 1].

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

У пакеті Fuzzy Logic Toolbox системи MATLAB гібридні мережі реалізовані у формі так званої адаптивної системи нейро-нечіткого виводу ANFIS. З одного боку, гібридна мережа ANFIS являє собою нейронну мережу з єдиним виходом і декількома входами, що являють собою нечіткі лінгвістичні змінні. При цьому терми вхідних лінгвістичних змінних описуються стандартними для системи MATLAB функціями належності, а терми вихідної змінної представляються лінійною чи постійною функцією належності.

З іншого боку, гібридна мережа ANFIS являє собою систему нечіткого виводу FIS типу Сугено нульового чи першого порядку, у якій кожне з правил нечітких продукцій має постійну вагу, рівну 1. У системі MATLAB користувач має можливість редагувати і налаштовувати гібридні мережі ANFIS аналогічно системам нечіткого виводу, використовуючи всі розглянуті раніше засоби пакета Fuzzy Logic Toolbox.

У пакеті Fuzzy Logic Toolbox системи MATLAB гібридні мережі реалізовані у формі адаптивних систем нейро-нечіткого виводу ANFIS. При цьому розробка і дослідження гібридних мереж виявляється можливою:

- інтерактивному режимі за допомогою спеціального графічного редактора адаптивних мереж, що одержав назву редактора ANFIS;
- режимі командного рядка за допомогою введення імен відповідних функцій з необхідними аргументами безпосередньо у вікно команд системи MATLAB. Для роботи в режимі командного рядка призначені спеціальні функції.

Редактор ANFIS дозволяє створювати чи завантажувати конкретну модель адаптивної системи нейро-нечіткого виводу, виконувати її навчання, візуалізувати її структуру, змінювати і налагоджувати її параметри, а також використовувати налагоджену мережу для одержання результатів нечіткого виводу.

Графічний інтерфейс редактора ANFIS викликається функцією **anfisedit** з командного рядка середовища Matlab.

7.2. Приклад розв'язку задачі нейро-нечіткого виводу

Для ілюстрації процесу розробки гібридної мережі в системі MATLAB розглянемо задачу побудови адаптивної системи нейро-нечіткого виводу для апроксимації деякої виробничої функції типу Кобба-Дугласа:

$$q = F(x_1, x_2) = 2\sqrt{x_1}\sqrt[4]{x_2}\ ,$$

яка описує залежність між витратами двох виробничих факторів (наприклад, грошей і праці) у кількості x_1 і x_2 та кількістю виготовленої продукції q у наперед визначених одиницях.

Загальна послідовність процесу розробки моделі гібридної мережі може бути представлена в наступному вигляді.

Крок 1. Для початку за допомогою редактора відладчика m-файлів або будь-якого текстового редактору підготуємо навчальні дані, що містять 6-ти рядків з трьох значень – два вхідних і одне вихідне значення, наприклад, наступного виду:

1	2	2.37
4	1	4.00
1	4	2.82
5	5	6.68
2	4	4.00
4	3	5.26
3	1	3.46
4	4	5.65
4	2	4.80

Числа в рядку розділяються клавішею «**Space**», а перехід на наступний рядок відбувається натисненням клавіші «**Enter**» . Слід зберегти цей файл у папці work каталогу Matlab6, наприклад, з назвою **train.txt**.

Взагалі, вхідні дані представляють собою звичайну числову матрицю розмірності *m*×(*n*+1), у якій кількість рядків *m* відповідає обсягу вибірки, перші *n* стовпців — значенням вхідних змінних моделі, а останній стовпчик — значенню вихідної змінної. Відповідно до правил системи MATLAB окремі значення матриці відокремлюються пробілами, а кожен рядок матриці завершується символом *"перевід каретки"* (клавіша <Enter>).

Хоча по кількості рядків матриці вхідних даних не існує формальних рекомендацій, прийнято вважати, що якість навчання гібридної мережі, а, отже, і точність одержуваних результатів пропорційно залежить

від обсягу навчальної вибірки. Що стосується кількості стовпців матриці вхідних даних, то слід зазначити можливі проблеми з працездатністю системи MATLAB, якщо кількість вхідних змінних перевищує 5 — 6.

Початкові дані, що завантажуються, можуть бути одного з наступних типів:

- навчальні дані **(Training)** — обов'язкові дані, що використовуються для побудови гібридної мережі;
- тестові дані **(Testing)** — необов'язкові дані, що використовуються для тестування побудованої гібридної мережі з метою перевірки якості функціонування побудованої гібридної мережі;
- перевірочні дані **(Checking)** — необов'язкові дані, що використовуються для перевірки побудованої гібридної мережі з метою з'ясування факту перенавчання мережі;
- демонстраційні дані **(Demo)** — дозволяють завантажити один з демонстраційних прикладів гібридної мережі.

Тому, створимо ще і файл тестових даних **test.txt**. За аналогією зі створенням файлу train.txt, введемо в нього наступні дані:

```
2    1    2.82
3    2    4.12
1    3    2.63
2    3    3.72
3    3    4.56
4    2    4.75
3    4    4.90
```

Крок 2. Головне меню редактора ANFIS достатньо просте і призначене для роботи з попередньо створеною системою нечіткого виводу. Основну частину графічного інтерфейсу займає вікно візуалізації даних, що розташоване нижче головного меню (рис. 7.3). Для знову створюваної гібридної мережі це вікно не містить ніяких даних. Для створення гібридної мережі необхідно завантажити дані. Для цієї мети слід скористатися кнопкою Load Data в лівій нижній частині графічного вікна. При цьому дані можуть бути завантажені з зовнішнього файлу (disk) чи з робочої області (worksp). У першому випадку необхідно попередньо створити файл із вхідними даними (файл train), що являє собою звичайний текстовий файл. Після завантаження файлу із навчальними даними в редактор ANFIS у робочому вікні редактора буде зображений графік, ордината якого відображає значення вхідної математичної функції (кількість продукції у прикладі).

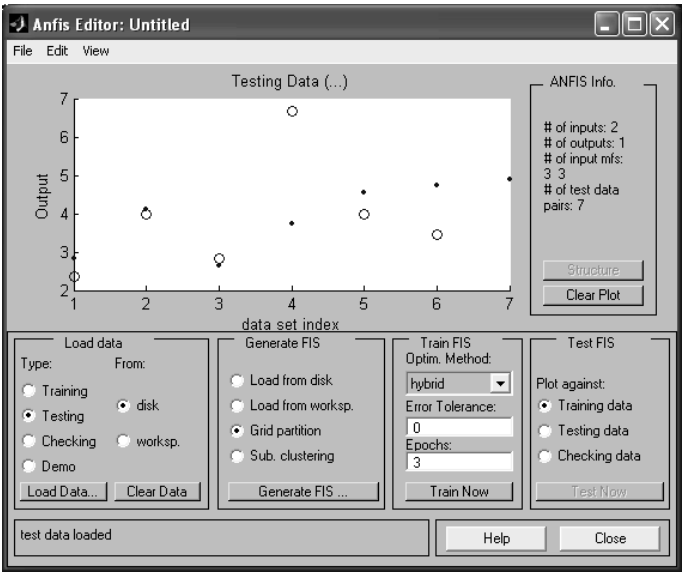

Рис. 7.3. Графічний інтерфейс редактора ANFIS після завантаження файлу function.dat з навчальними даними

Завантажимо ще і тестові дані для перевірки. Для цього в лівому нижньому кутку екрану оберемо тип **testing**, натиснемо кнопку **Load data** і завантажимо файл **test.txt**.

Крок 3. Приступити до генерації структури системи нечіткого виводу FIS**.** Ця структура аналогічна структурі системи нечіткого виводу типу Сугено. Тобто, має в своєму складі лінгвістичні вхідні змінні; вихідну змінну, терми якої представляються у вигляді числових констант або лінійних функцій від вхідних змінних; систему правил виводу.

Крім того, можна завантажити структуру уже створеної FIS з диска **(Load from disk),** або з робочої області **(Load from worksp).** При створенні структури нової FIS можна незалежно розбити усі вхідні змінні на області їх значень **(Grid partision)** чи скористатися процедурою субтрактивної кластеризації для попередньої розбивки значень вхідних змінних на кластери близьких значень **(Sub. clustering).**

Після натискання кнопки **Generate FIS** викликається діалогове вікно з вказівкою числа і типу функцій належності для окремих термів вхідних змінних і вихідної змінної (рис. 7.4). В цьому вікні можна вибрати тип функцій належності і кількість термів кожної вхідної змінної, а також тип вихідної змінної. Установивши параметри генерації, наприклад, як на рис.7.4, одержимо структуру FIS.

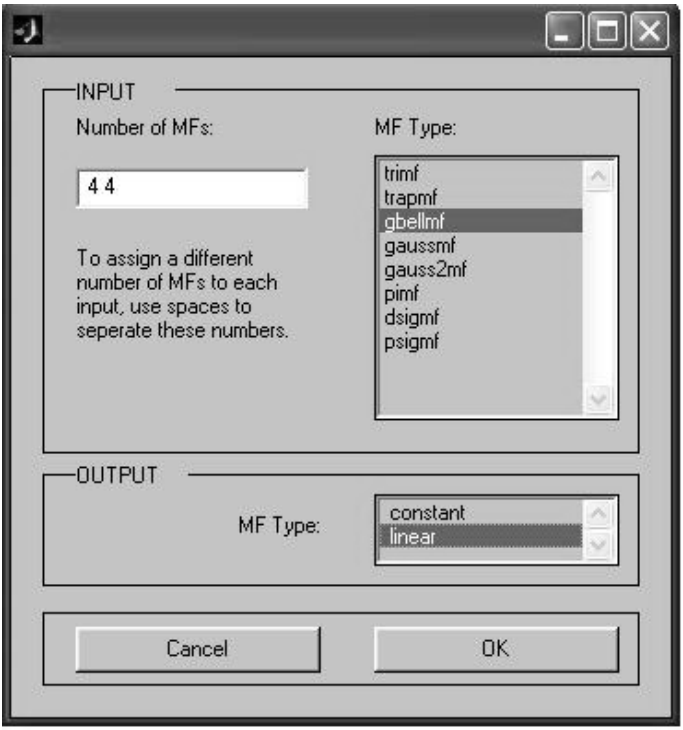

Рис. 7.4. Діалогове вікно для завдання кількості і типу функцій належності

Після генерації структури гібридної мережі можна її подивитись, натиснувши кнопку **Structure** в правій частині графічного вікна. Структура отриманої в результаті системи нечіткого виводу FIS відображається в окремому вікні і досить тривіальна по своєму виду. Проаналізуйте її зміст.

Крок 4. Перед навчанням гібридної мережі необхідно задати параметри навчання, для чого варто скористатися наступною групою опцій у правій нижній частині робочого вікна:

1. Вибрати метод навчання гібридної мережі — зворотного розповсюдження **(bacpropo)** чи гібридний **(hybrid)**, що представляє собою комбінацію методу найменших квадратів і методу спадання зворотного градієнта.

2. Установити рівень помилки навчання **(Error Tolerance)** — за замовчуванням значення 0 (змінювати не рекомендується).

3. Задати кількість циклів навчання (Epochs) — за замовчуванням значення 3 (рекомендується збільшити і для розглянутого приклада задати його значення рівним 40).

6. Для навчання мережі варто натиснути кнопку **Trian Now**. При цьому хід процесу навчання ілюструється у вікні візуалізації у формі графіка — залежність помилки від кількості циклів навчання.

7. Аналогічно можуть бути виконані додаткові етапи тестування й перевірки гібридної мережі, для яких необхідно попередньо завантажити відповідні дані.

Крок 5. Подальше настроювання параметрів побудованої і навченої гібридної мережі може бути виконане за допомогою розглянутих раніше стандартних графічних засобів пакета Fuzzy Logic Toolbox. Для цього рекомендується зберегти створену систему нечіткого виводу в зовнішньому файлі з розширенням **fis**, після чого варто завантажити цей файл у редактор систем нечіткого виводу FIS. Можна працювати зі структурою системи нечіткого виводу безпосередньо в редакторі ANFIS.

При цьому також стають доступними редактор функцій належності системи нечіткого виводу (Membership Function Editor), редактор правил системи нечіткого виводу (Rule Editor), програма перегляду правил системи нечіткого виводу (Rule Viewer) і програма перегляду поверхні системи нечіткого виводу (Surface Viewer). Проаналізуйте правила та функції належності, які були побудовані даною системою апроксимації виробничої функції. Спробуйте їх корегувати і слідкувати за зміною результатів.

Крок 6. Виконати аналіз точності побудованої нечіткої моделі гібридної мережі можна за допомогою перегляду поверхні відповідної системи нечіткого виводу. Для цього слід записати побудовану мережу у вигляді файлу з розширенням *.fis і завантажити його в системі нечіткого виводу fuzzyToolbox (рис.5).

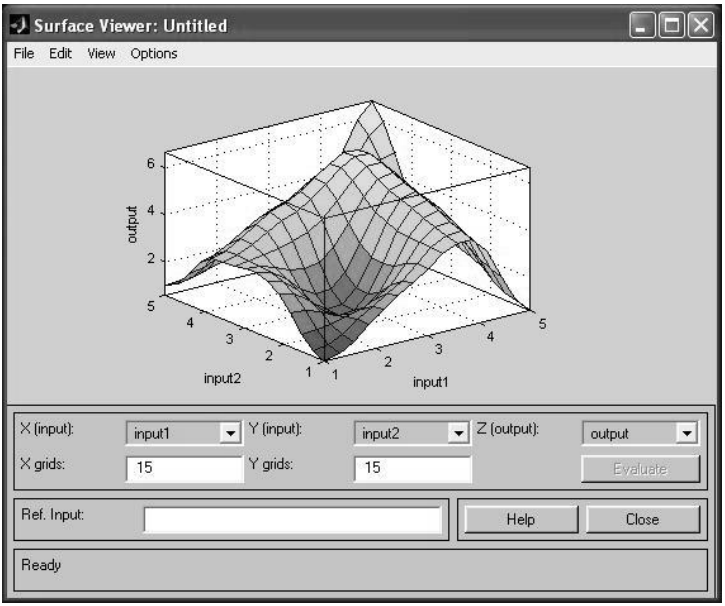

Рис. 7.5. Графічний інтерфейс пергляду поверхні згенерованої системи нечіткого виводу

Візуальний аналіз зображеного графіка з точним графіком функції $q = F(x_1, x_2) = 2\sqrt{x_1}\sqrt[4]{x_2}$ дозволяє судити про досить високий ступінь їх співпадання, що може свідчити про адекватність побудованої нечіткої моделі гібридної мережі.

Аналіз адекватності побудованої моделі можна виконати за допомогою перегляду правил відповідної системи нечіткого виводу (рис.7.6).

Перевірка побудованої моделі гібридної мережі може бути виконана для декількох значень вихідної змінної. З цією метою необхідно ввести конкретне значення в поле вводу **Input** (наприклад, значення 3 1), після натискання клавіші <Enter> за допомогою побудованої моделі буде отримане відповідне значення вихідної змінної (у даному випадку значення 3.46). Порівнюючи отримане значення з точним значенням функції 3.46, одержимо відносну помилку порядку 0%.

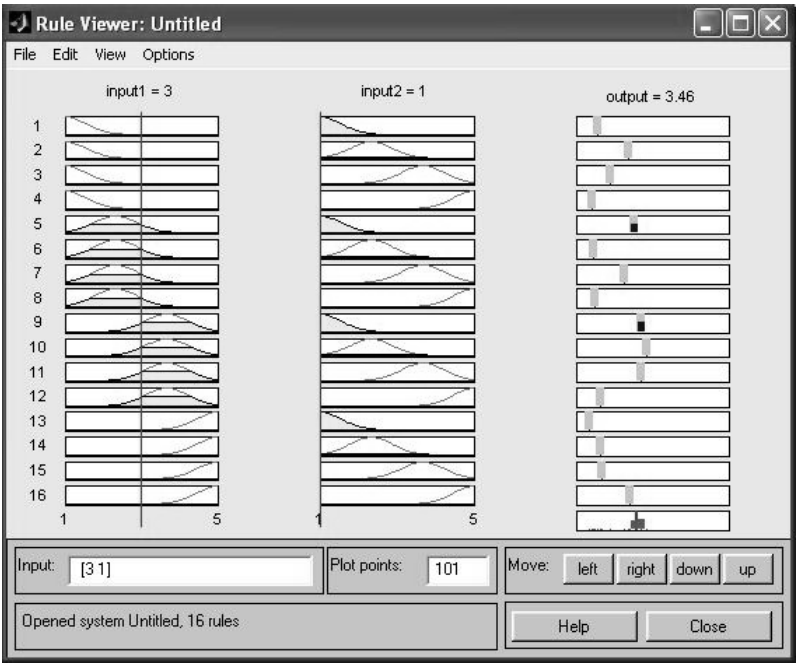

Рис. 7.6. Графічний інтерфейс перегляду правил згенерованої системи нечіткого виводу

Менш удалою виявляється перевірка для значення вхідних змінних 4 3, для якого побудована модель пропонує значення 5.45 (табличне значення 5.26). Очевидно, даний факт свідчить не на користь адекватності побудованої нечіткої моделі і вимагає її додаткового настроювання.

У загальному випадку додаткове настроювання моделі може бути виконане декількома можливими способами. Найбільш прийнятними з них представляються наступні.

1. Підготовка і завантаження більшого за обсягом вибірки файлу з навчальними вхідними даними.

2. Підготовка і завантаження додаткового файлу з перевірочними вхідними даними, сформованими для пар значень розглянутої математичної функції, відсутніх у вибірці навчальних даних.

3. Редагування типів і значень параметрів функцій належності термів вхідної і вихідної змінних за допомогою редактора функцій належності системи MATLAB.

Проілюструємо третій спосіб додаткового настроювання побудованої нечіткої моделі гібридної мережі. На перший погляд він представляється найбільш природним з погляду можливості візуального контро-

лю виконуваних змін параметрів. З цією метою відкриємо редактор функцій належності і методом підбора змінимо кількісні значення параметрів другої і третьої функції належності вхідної змінної, оскільки саме вони "працюють" при одержанні некоректного значення вихідної змінної: 5.45 для значення вхідних змінних 4 і 3. (рис.7.7).

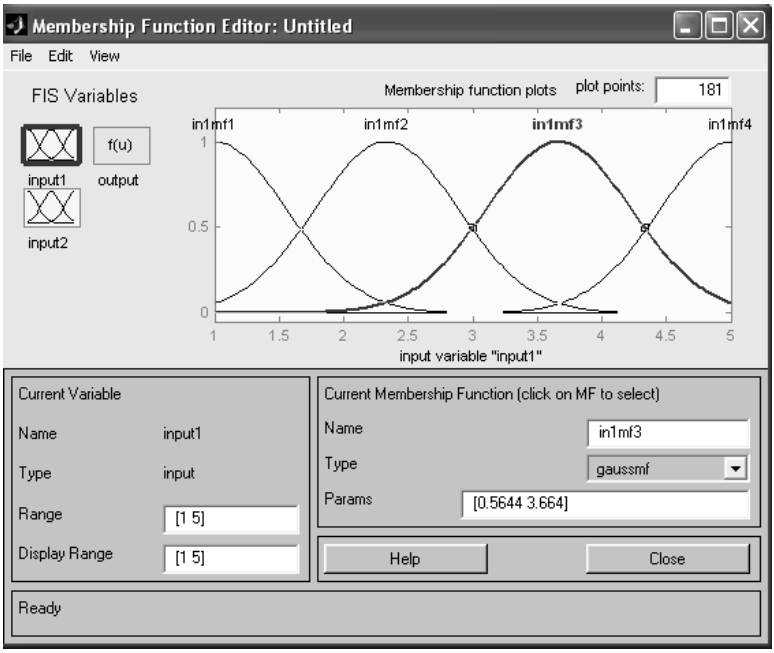

Рис. 7.7. Графічний інтерфейс редактора функцій належності побудованої системи нечіткого виводу

Після зміни параметрів у полі **Params** для третьої функції належності першої змінної на значення [0.385 3.66] одержимо практично точне значення вихідної змінної для значення вхідної змінної 5.29 (рис.7.8).

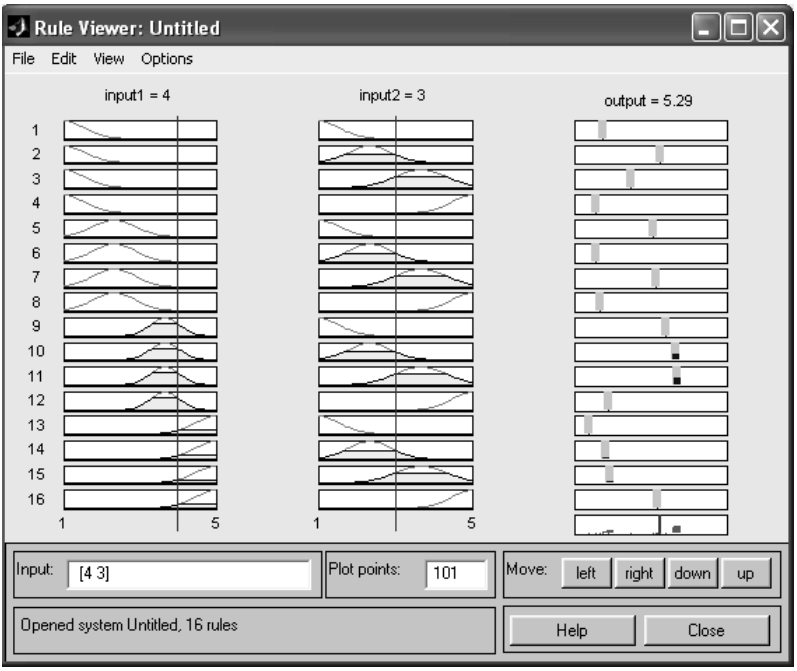

Рис. 7.8. Результат ручного настроювання параметрів функції належності вхідної змінної для розглянутої системи нечіткого виводу

Більш ефективним способом настроювання, а точніше — модифікації, даної нечіткої моделі виявляється перший. З цією метою можна збільшити обсяг навчальної вибірки до 12 пар значень, додавши ще три рядка в файл train.txt. Наприклад:

```
2    1          2.82
3    2          4.12
1    3          2.63
```

Після видалення усіх раніше завантажених даних кнопкою **Clear Data** завантажимо новий файл із навчальною вибіркою. При генерації структури нової FIS збільшимо кількість термів і, відповідно, кількість функцій належності вхідної змінної до 5, залишивши їх тип без зміни (**gbellmf**). Процес навчання виконаємо аналогічно раніше розглянутому. У результаті буде отримана нова система нечіткого виводу FIS, аналіз якої показує, що в порівнянні з першим варіантом нечіткої моделі вона більш точно описує вихідну математичну функцію. Це може бути осно-

вою для генерації нової нечіткої моделі, що пропонується виконати самостійно як вправу.

На закінчення необхідно відзначити, що навіть найпростіші розглянуті приклади відбивають творчий характер процесу побудови й аналізу моделей гібридних мереж. При цьому вибір того чи іншого способу додаткового настроювання нечітких моделей залежить не тільки від специфіки розв'язуваної задачі, але і від обсягу доступної вибірки навчальних і перевірочних даних.

У випадку недостатньої інформації навчальних даних використання гібридних мереж може виявитися взагалі недоцільним, оскільки одержати адекватну нечітку модель, а значить — і точний прогноз значень вихідної змінної не представляється можливим. Саме з цих причин необхідний попередній аналіз усіх можливостей застосовуваних нечітких моделей для рішення конкретних задач у тій чи іншій проблемній області. Подібний аналіз необхідно виконувати із системної точки зору і з обліком всіх обставин, що складаються на даний момент. Тільки всебічна і повна оцінка проблемної ситуації дозволить розробити адекватну модель розв'язку тієї чи іншої конкретної задачі нечіткого керування чи прийняття рішень.

7.3. Завдання для самостійної роботи.

1. Апроксимація поверхні.

За допомогою гібридної мережі провести апроксимацію залежності $y = x^3$, використовуючи дані, які представлені на наступному рисунку.

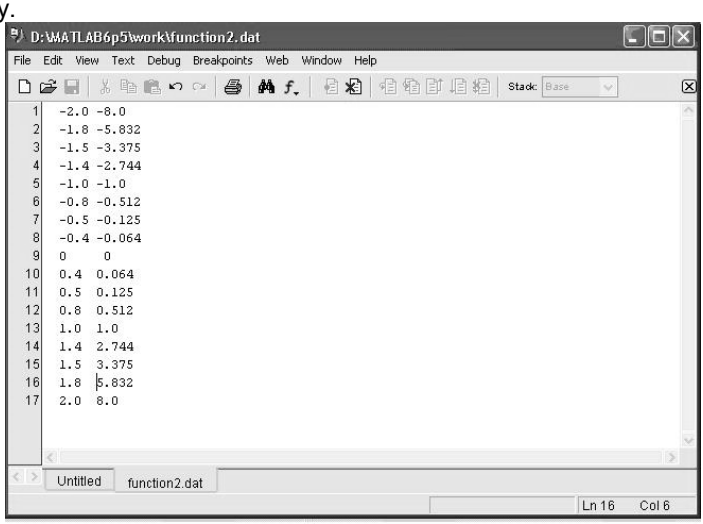

Рис. 7.9. Навчальні дані збільшеного об’єму для побу-

дови гібридної мережі ANFIS, що представляє функцію

$$y = x^3.$$

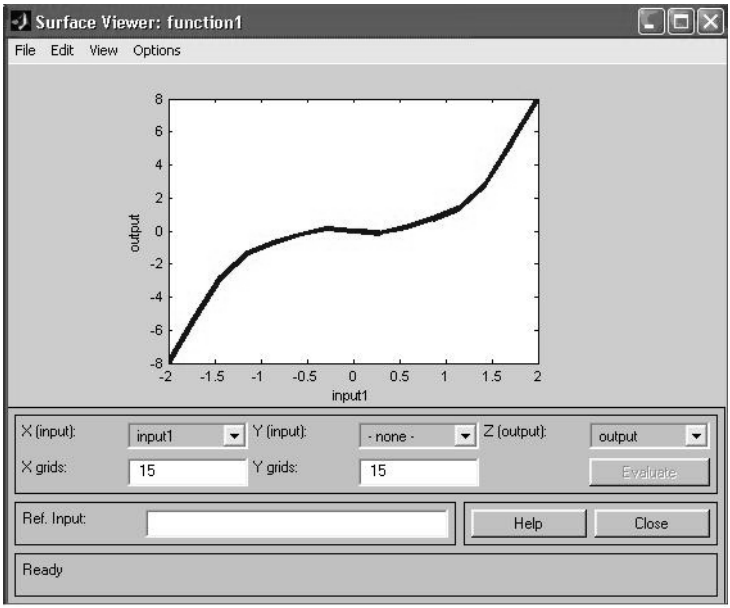

Рис 7. 10. Приблизний вигляд поверхні за результатами роботи СНВ, яка апроксимує залежність $y = x^3$

2. Аналіз і прогнозування цін на ринку житла

Як приклад побудови і використання адаптивної системи нейронечіткого виводу розглянемо процес розробки нечіткої моделі гібридної мережі для розв'язування задачі прогнозування ціни на ринку однокімнатних квартир в районі Теремків в залежності від кількості метрів та ціни на дану квартиру за попередні три місяці.

Суть даної задачі полягає в тому, щоб, знаючи динаміку зміни вартості продажу квартир за фіксований інтервал часу, прогнозувати значення їх вартості на визначений момент часу в майбутньому. При цьому характерною рисою динаміки зміни вартості (тренда) є наявність двох основних тенденцій у коливаннях відповідних цін.

З одного боку, спостерігається загальне довгострокове підвищення вартості, зв'язане з інфляцією та іншими факторами. З іншого боку,

спостерігається короткострокове коливання цін, зв'язане з цілим рядом випадкових факторів, адекватне представлення яких у тій чи іншій формальній моделі навряд чи можливо.

Традиційно для рішення даної задачі застосовуються різні моделі технічного аналізу, засновані на використанні різних індикаторів. У той же час наявність неявних тенденцій у динаміці зміни вартості житла дозволяє застосувати модель адаптивних нейро-нечітких мереж.

У якості вихідних даних можна скористатися інформацією про динаміку вартості квартир, які пропонує фірма «Благовест» за 10 останніх тижднів, що доступна в Інтернеті за адресою: **www.blagovest.com**. Дану інформацію для зручності подальшої роботи представимо в табличній формі (табл. 2)

Таблиця 2. Динаміка вартості квартир у період з липня 2005 року по квітень 2006 року.

Тиждень	Метраж	Вартість
29.01.06	33	60000
29.01.06	45	65000
5.02.06	58	59700
5.02.06	36	50000
12.02.06	28	52000
12.02.06	25	40000
19.02.06	42	60000
19.02.06	33	58000
2.03.06	25	40000
2.03.06	46	60500
19.03.06	28	52000
19.03.06	45	63000
26.03.06	36	60000
26.03.06	39	54000
2.04.06	46	60500
2.04.06	45	63000
9.04.06	28	52000
9.04.06	46	63500

16.04.06	43	63000
16.04.06	45	68000
7.05.06	45	64500
7.05.06	46	65000

Припустимо, що нечітка модель гібридної мережі буде містити 3 вхідних змінних. Це будуть значення цін 1 м^2 за попередні до поточного 3 тижня. Вихідна змінна – вартість квартири (1м^2) на поточний тиждень. Тому у файлі даних слід ввести матрицю цін, яка містить 4 стовпчики. Якщо ми розглядаємо 10 тижнів, тоді кількість рядків буде – 7.

Збережемо навчальну вибірку в зовнішньому файлі під ім'ям prise.txt. Після цього відкриємо редактор ANFIS, у який завантажимо цей файл із навчальними даними.

Перед генерацією структури системи нечіткого виводу типу Сугено після виклику діалогового вікна властивостей задамо для кожної із вхідних змінних по 3 або 4 лінгвістичних терма, а як тип їх функції належності виберемо трикутні функції (установлені системою MATLAB за замовчуванням).

Для навчання гібридної мережі скористаємося гібридним методом навчання з рівнем помилки 0, а кількість циклів навчання задамо 20. Після закінчення навчання даної гібридної мережі може бути виконаний аналіз графіка помилки навчання, що показує, що навчання практично закінчилося.

Після навчання гібридної мережі можна візуально оцінити структуру побудованої нечіткої моделі. Очевидно, графічна наочність даної моделі залишає бажати кращого, оскільки загальна кількість правил у розробленій адаптивній системі нейро-нечіткого виводу досить багато, що ускладнює їхній візуальний контроль і оцінку.

За допомогою графічних засобів системи MATLAB можна виконати контроль і настроювання параметрів функцій належності вхідних змінних і правил нечітких продукцій. Для виконання відповідних операцій можна скористатися редактором функцій належності.

Оскільки точність кількісних значень, забезпечувана графічними засобами пакета Fuzzy Logic Toolbox, є недостатньою для рішення даної задачі, скористаємося функцією командного рядка evalfis. Як аргументи цієї функції вкажемо вектор значень вартості квартири на поточний і 3 попередніх тижня. Повний формат виклику цієї функції буде наступним:

```
out=evalfis ([x1  x2        x3],       prise)
```

де out — умовне ім'я вихідної змінної; x1 — значення вартості квартири за попередній тиждень; x2— значення вартості квартири за два тиждня; x3— значення вартості квартири за три тиждня; prise — ім'я структури FIS, попередньо завантаженої в робочу область системи MATLAB.

Після виконання цієї команди за допомогою розробленої нечіткої моделі буде отримане значення вихідної змінної для наступного тижня буде отримано деяке значення. Порівнюючи отримане значення з відповідним значенням з табл. 2, можна констатувати порівняти прогноз цін квартир.

У цьому випадку нечіткі моделі адаптивних систем нейро-нечіткого виводу можуть вважатися новим і конструктивним інструментом технічного аналізу фінансових ринків.

Розглянутий підхід є перспективним напрямком для побудови і використання відповідних нечітких моделей прогнозування цін інших фінансових інструментів, таких як курси валют, акцій компаній, фьючерсів і опціонів. Дійсно, загальним для всіх цих інструментів з позицій технічного аналізу є відсутність апріорних припущень про динаміку коливань відповідних курсів цін, що цілком узгоджується з вихідними передумовами побудови нечітких моделей адаптивних систем нейро-нечіткого виводу.

ДОДАТОК
ОСНОВИ РОБОТИ В СЕРЕДОВИЩІ MATLAB

§ 1. Обчислення арифметичних виразів

Арифметичні вирази в MatLab складаються, як і в більшості мов, з чисел, знаків арифметичних операцій, знака ^ (в степені), круглих дужок, змінних та вбудованих функцій [7]. Десяткова частина числа відділяється точкою. Для обчислення найпростішого виразу потрібно набрати його в командному рядку і натиснути <Enter>. Відповідь записується в спеціальну змінну ans і її значення виводиться на екран:

```
>> 1.5+2.9
ans =
4.4000
```

Вигляд результату залежить від встановленого формату (про формати дивись нижче). Після обчислення наступного виразу значення змінної ans відповідно зміниться. Тому для збереження результатів проміжних обчислень потрібно записувати їх в спеціальні змінні з додержанням наступних правил:

- ім’я змінної може складатися з символів латинського алфавіту, знаку підкреслення та цифр, але не обов’язково починатися з символу алфавіту;
- великі і малі літери різняться;
- пропуск не входить в ім’я змінної.

В якості знаку присвоювання використовується «=», наприклад:

```
>> a=3.25*(0.7-3.3/5.1)+2.3^3
```

Результат відразу виводиться на екран. Якщо це не потрібно, слід завершити вираз з оператором присвоювання крапкою з копою «;».Символ e використовується для запису числа в експоненціальній формі: числа 0.00125 и 1.25e–3 еквівалентні. Комплексні числа вводяться за допомогою символу «i»:

```
>>b=5*(2.2+3.9i)+0.8
b =
11.8000 +19.5000i
```

В MatLab є великий набір вбудованих математичних функцій. Деякі з них наведені в табл. Д1.1. Під час виклику математичної функції аргумент записується в круглі дужки. Повний перелік всіх вбудованих елементарних математичних функцій можна отримати, якщо набрати в командному рядку help elfun. Команда help відображає список розділів довідкової системи. Для отримання змісту розділу необхідно вказати через пропуск його назву після help, а для виводу детальної інформації про будь-яку функцію, потрібно ввести в рядку разом з командою help ім’я функції.

Таблица Д1.1 Основні математичні функції

sin, cos, tan, cot	Синус, косинус, тангенс и котангенс (аргумент задається в радіанах)
sec, csc	Секанс, косеканс
asin, acos, atan, acot	Арксинус, арккосинус, арктангенс и арккотангенс
asec, acsc	Арксеканс, арккосеканс
sinh, cosh, tanh, coth	Гіперболічні синус, косинус, тангенс и котангенс
sech, csch	Гіперболічні секанс и косеканс

Asinh, acosh, atanh, acoth	Гіперболічні арксинус, арккосинус, арктангенс и арккотангенс;
Exp	Експоненціальна функція
log, log2, log10	Натуральний логарифм, логарифми за основою 2 і 10
Sqrt	Квадратний корінь
abs, sign	Модуль и знак числа
conj, imag, real	Комплексно-спряженні, уявна і дійсна частина

Нехай, наприклад, потрібно найти значення виразів при $x = 0.2$ и $y = -3.9$:

$$c = \sqrt{\frac{\sin\left(\frac{4}{3}\pi x\right) + e^{0.1y}}{\cos\left(\frac{4}{3}\pi x\right) + e^{0.1y}}} + \sqrt[3]{\frac{\sin\left(\frac{4}{3}\pi x\right) + e^{0.1y}}{\cos\left(\frac{4}{3}\pi x\right) + e^{0.1y}}}$$

Простіше всього вирішити поставлену задачу, використовуючи проміжні змінні:

```
>> x=0.2;
>> y=-3.9;
>> a=sin(4/3*pi*x)+exp(0.1*y);
>> b=cos(4/3*pi*x)+exp(0.1*y);
>> c=sqrt(a/b)+(a/b)^(1/3)
c =
2.0451
```

Зверніть увагу на кілька важливих особливостей. Всі оператори присвоювання, крім останнього, закінченні точкою з комою для подавлені виводу результату. Не обов'язково набирати виразів для b, подібні на тільки що введення для a. Після введення третього рядка натисніть клавішу <↑>. В командному рядку появиться попередній вираз, внесіть в нього необхідні зміни, а саме, поміняйте sin на cos, и натисніть <Enter>. Клавіші <↑> и <↓> служать для переходу по історії команд, т. ін. для занесення раніше набраних команд в командний рядок, а <←>, <→>, <Home>, <End> — для переміщення в межах командного рядка. Пересування по екрану (тільки для перегляду команд, а не для редагування) виконується клавішами <PageUp>, <PageDown> чи вертикальної полос-

ки скролінга. Починаючи з версії 6.0 в середовище MatLab включено вікно Command History для швидкого переходу по історії команд. В любий момент можна вивести значення змінної в командне вікно, для чого слід набрати ім'я змінної в командному рядку і натиснути <Enter>, або викликати функцію disp:

```
>> disp(c)
2.0451
```

Виникаючі в процесі обчислення комплексний результат не являється помилкою. MatLab автоматично переходить в область комплексних чисел, продовжуючи обчислення. Найдіть, наприклад, квадратний корінь із –1. Більш того, допустимі операції ділення на нуль, котрі призводять до стандартних змінних Inf чи –Inf. Результат ділення нуля на нуль є NaN (Not a Number — не число). Переповнення чи втрата точності в MatLab при виконання операцій з числами з плаваючою точкою не викликає зупинення обчислений.

Перегляд поточний змінних робочого середовища відбувається при допомозі команди whos. Припустимо, що попереднім змінним a і b були присвоєнні значення:

```
>> a=-1.34;
>> b=2.98+3.86i;
```

Викличте команду whos, вказав через пропуски імена змінних

```
>> whos a b
```

У командне вікно виводиться таблиця, приведена нижче. В стовпчику Class вказаний тип змінної, в Bytes — число байт, виділених під зберігання значення, а Size містить інформацію про розміри. Після таблиці розміщена стрічка з вказаним сумарного об'єму пам'яті в байтах.

```
Name Size Bytes Class
a 1x1 8 double array
b 1x1 16 double array (complex)
Grand total is 2 elements using 24 bytes
```

Зверніть увагу, що числові змінні в MatLab являються двомірними масивами розміру один на один. Представлення всіх даних в MatLab у виді масивів виявляється дуже корисним, про що детальніше буде сказано в наступних параграфах додатку.

Користувач має можливість встановлювати підходящі формати виводу. Існує два способи надання форматів. Вибір пункту Preferences… в меню File робочого середовища призводить до появлення діалогового вікна з одно-іменною назвою. В MatLab 5.3 панель Numeric Format вкладки General містить переключатиль, а в версії 6.x на панелі Text display розміщенні відкриваючі списки Numeric Format и Numeric display (при вибраному пункті Command Window в списку лівої панелі вікна). Можливо встановить один із наступних форматів.

- short (default) — короткий формат с плаваючою крапкою з чотирма цифрами після десяткової точки (використовується по замовчування).
- long — довгий формат з плаваючою крапкою з чотирнадцятьма цифрами після десяткової точки.
- short e — експоненціальний формат з чотирма цифрами після десяткової крапки.
- long e — експоненціальний формат з п'ятнадцятьма цифрами після десяткової крапки.
- short g — найкращі представлення числа або в форматі short, або в short є (аналогічно long g).
- hex — шістнадцятирічного представлення числа.
- + — додаткові і від'ємні числа відображаються знаками "+" і "–", а нульові — пропусками.
- bank — формат для виводу грошових сум з двома знаками після десяткової крапки.
- rat — дійсні числа приближено представляються відношенням двох не великих цілих чисел.

Інді виникають ситуації, коли MatLab не може вкластися в встановлений формат при виводі результату. Нехай, наприклад, встановлений формат short. При обчисленні 1/3333 результат відображається в форматі short e, однак, автоматичною зміни формату для всіх наступних обчислений не відбувається.

MatLab надає два способи виводу в командне вікно.

- compact — рядки з результатами виводяться підряд.
- loose — рядки з результатами розділяються пустим рядком.

Команда format служить для установки формату із командної рядка. До прикладу, звернення

```
>> format short e
```

аналогічно вибору короткого експоненціального формату в діалоговому вікні Preferences. Поза залежності від встановленого формату всі обчислення відбувається з подвійною точністю, як наслідок, після зміни формату з short на long не потребується повторно находить значення змінних. Достатньо знову вивести їх значення в командному вікні.

Завдання для самостійної роботи

У всіх завданнях потрібно занести в деяку змінну значення виразів при заданих і, відобразити результат в різних форматах і вивчити інформацію змінних за допомогою команди whos.

1. $$F = \left(\frac{e^x \sin y + 2^x \cos y}{200x + y} \right)^{2.3} + \ln\left|\sin y\right| - \sqrt{\frac{e^x \sin y + 2^x \cos y}{200x + y}}$$

2. $$Z = \arctan \frac{\sqrt[3]{x - \sin(y)}}{\sqrt{1 - x^2}} - \frac{|x|\sqrt{1 - x^2}}{\sqrt[3]{x - \sin(y)}}$$

3. $$T = \frac{(\sin y + \sin 2y + \sin 3y)^4}{1 + \dfrac{\sin y + \sin 2y + \sin 3y}{e^x}} + \sqrt{1 + \frac{\sin y + \sin 2y + \sin 3y}{e^x}}$$

4. $$W = \left(1 + \frac{\ln y}{x + \tan y} \right)^{1 + \frac{x + \tan y}{\ln y}}$$

5. $$R = sh \frac{(x + \ln y)^3}{\sqrt{|x - \ln y|}} \cdot ch\left[(x + \ln y)\sqrt{|x - \ln y|} \right]$$

6. $$H = \frac{\sqrt{\cos 2y + \sin 4y + \sqrt{e^x + e^{-x}}}}{(e^{-x} + e^x)^3 (\sin 4y + \cos 2y - 2)^2}$$

7. $$Q = \sqrt{e^x \sin y + e^{-x} \cos y + \sqrt{1 + \frac{e^x \sin y + e^{-x} \cos y}{\tan y}}}$$

8. $$A = \sqrt[5]{x(1 + x)^2 (1 + 2x)^3 + \sqrt[3]{\frac{x(1 + x)^2 (1 + 2x)^3}{\ln\left|\cot \tan \quad y\right|}}}$$

9.

$$S = \arctan\sqrt{\left|\frac{x-\sin y}{x+\sin y}+\frac{x+\sin y}{x-\sin y}\right|}+e^{(x-\sin y)(x+\sin y)}$$

10.

$$B=\frac{1+\arcsin(\cos 2y)}{2^x+3^{-x}}+\left(\frac{2^x+3^{-x}-1}{x+\arcsin(\cos 2y)}\right)^2$$

§ 2. Вектор-рядки і вектор-стовпці

До особливостей роботи з масивами в MatLab відноситься те, що одновимірний масив може бути вектор-рядком або вектор-стовпця. Якщо спосіб представлення масиву важливий, то ми підкреслюватимемо, про рядок або про стовпець йдеться. Якщо ж це несуттєво, то говоритимемо про вектор-рядки і вектор-стовпців просто як про вектори або одновимірні масиви (одновимірний масив в MatLab є двовимірний, у якого один з розмірів рівний одиниці).

Для введення вектора використовуються квадратні дужки, елементи вектора відокремлюються один від одного:

- крапкою з комою, якщо потрібно отримати вектор-стовбець;
- пропуском або комою, якщо необхідно розмістити елементи у вектор-рядку.

Наприклад, щоб занести вектор-стовпці і вектор-рядки

$$a=\begin{bmatrix}0.2\\-3.9\\4.6\end{bmatrix}\qquad b=\begin{bmatrix}7.6\\0.1\\2.5\end{bmatrix}\qquad u=\begin{bmatrix}0.1 & 0.5 & -3.7 & 8.1\end{bmatrix}$$

$$v=\begin{bmatrix}5.2 & 9{,}7 & 3.4 & -0.2\end{bmatrix}$$

у відповідні масиви, слід набрати в командному рядку:

```
>> a=[0.2; -3.9; 4.6];
>> b=[7.6; 0.1; 2.5];
>> u=[0.1 0.5 -3.7 8.1];
>> v=[5.2 9.7 3.4 –0.2];
```

Крапка з комою в кінці кожного рядка поставлена для придушення висновку на екран, вона ніяк не пов'язана з крапкою з комою, яка є розділником елементів у вектор-стовпцях. Виведіть в командне вікно значення змінних a, b, u, v і побачите, як MatLab відображає вміст вектор-рядків і вектор-стовпців. Отримаєте інформацію про змінних за допомогою команди whos.

Для отримання довжини вектора призначена функція length, вектор вказується як її вхідний аргумент:

```
>> L=length(a)
L =
3
```

Вектор-стовпці з однаковим числом елементів можна складати і віднімати один з одного за допомогою знаків "+" і "–". Аналогічне вірно і для вектор-рядків:

```
>> c=a+b;
>> w=u-v;
```

Складання і віднімання вектор-рядка і вектор-стовпця або векторів різних розмірів приводить до помилки. Операція * призначена для множення векторів за правилом матричного множення. Оскільки MatLab розрізняє вектор-рядки і вектор стовпці, то допустимо або множення вектор-рядка на такій же по довжині вектор-стовпець (скалярний добуток), або множення вектор-стовпця на вектор-рядок (зовнішній добуток, в результаті якого виходить прямокутна матриця). Скалярний добуток двох векторів повертає функція dot, а векторне — cross:

```
>> s=dot(a,b)
>> c=cross(a,b)
```

Зрозуміло, векторний добуток визначений тільки для векторів з трьох елементів.

Для операції транспонування зарезервований апостроф '. Якщо вектор містить комплексні числа, то операція ' призводить до комплексно-зв'язаного вектора. При обчисленні скалярного і векторного добутків функціями cross і dot не обов'язково стежити за тим, щоб обидва вектори були або стовпцями, або рядками. Результат виходить вірний, наприклад, при зверненні c=cross(a,b'), лише із стає вектор-рядком.

MatLab підтримує по-елементні операції з векторами. Разом з множенням за правилом матричного множення, існує операція по-елементного множення .* (крапка із зірочкою). Дана операція застосовується до векторів однакової довжини і приводить до вектора тієї ж довжини, що вихідний, елементи якого рівні добутку відповідних елементів початкових векторів. Наприклад, для векторів a і b, введених вище, по-елементне множення дає слідуючий результат:

```
>> c=a.*b
c =
1.5200
-0.3900
11.5000
```

Аналогічним чином працює по-елементне ділення ./ (крапка з косою межею). Крім того, операція .\ (крапка із зворотною косою межею) здійснює зворотне по-елементне ділення, тобто вирази a./b і b.\a еквівалентні. Зведення елементів вектора a в ступені, рівні відповідним елементам b,

проводиться з використанням .^. Для транспонування вектор-рядків або вектор-стовпців призначено поєднання .' (крапка з апострофом). Операції ' і .' для речовинних векторів приводять до однакових результатів. Не обов'язково застосовувати по-елементні операції при множенні вектора на число і числа на вектор, діленні вектора на число, складанні і відніманні вектора і числа. При виконанні, наприклад, операції a*2, результат є вектор того ж розміру, що і a, з подвоєними елементами.

Вектори можуть бути аргументами вбудованих математичних функцій, таких, як sin, cos і т.д. В результаті виходить вектор з елементами, рівними значенню функції, що викликається, від відповідних елементів початкового вектора, наприклад:

```
>> q=sin([0 pi/2 pi])
q =
0 1.0000 0.0000
```

Однак для обчислення складнішої функції від вектора значень, скажімо

$$\frac{v\sin v + v^2}{v+1}$$

слід записати

```
>> f=(v.*sin(v)+v.^2)./(v+1)
```

Часто потрібно обчислити функцію від вектора значень аргументу, що відрізняються один від одного на постійний крок. Для створення таких вектор-рядків передбачена двокрапка. Послідовність команд

```
>> x=-1.2:0.5:1.8;
>> f=(x.*sin(x)+x.^2)./(x+1);
```

Призводить до заповнення наступних векторів:

```
>> x
x =
-1.2000 -0.7000 -0.2000 0.3000 0.8000 1.3000 1.8000
>> f
f =
-12.7922 3.1365 0.0997 0.1374 0.6744 1.2794 1.7832
```

Крок може бути негативним, в цьому випадку початкове значення повинне бути більше, або рівно кінцевому для отримання не порожнього вектора. Якщо крок рівний одиниці, то його можна не вказувати, наприклад:

```
>> n=-3:4
n =
-3 -2 -1 0 1 2 3 4
```

Ясно, що для заповнення вектор-стовпця елементами з постійним кроком слід транспонувати вектор-рядок. Створення векторів за допомогою двокрапки і уміння проводити по-елементні операції необхідне для візуалізації масивів даних, про що буде сказано в наступних розділах.

MatLab володіє великим набором вбудованих функцій для обробки векторних даних, частина з них приведена в табл. 2.1. Повний список наявних функцій виводиться в командне вікно за допомогою help datafun. Звернете увагу на те, що ряд функцій допускає звернення до них як з одним, так і з двома вихідними аргументами. У разі декількох вихідних аргументів вони полягають в квадратні дужки і відділяються один від одного коми.

Дуже часто потрібно обробити тільки частину вектора, або звернутися до деяких його елементам. Розберемо правила MatLab, по яких проводиться індексація векторних даних. Для доступу до елементу вектора необхідно вказати його номер в круглих дужках відразу після імені змінної, в якій міститься вектор. Наприклад, сума першого і третього елементів вектора v знаходиться за допомогою виразу

```
>> s=v(1)+v(3);
```

Звернення до останнього елементу вектора можна провести з використанням end, тобто v(end) і v(length(v)) приводять до однакових результатів.

Таблиця. Д2.1 Функції обробки даних

Функції	Призначення
s=sum(a)	Сума всіх елементів вектора a
p=prod(a)	Добуток всіх елементів вектора a
m=max(a)	Знаходження максимального значення серед елем
[m,k]=max(a)	Другий вихідний аргумент k містить номер максим векторі a
m=min(a)	Знаходження мінімального значення серед елемен
[m,k]=min(a)	Другий вихідний аргумент k містить номер мінім векторі a
m=mean(a)	Обчислення середнього арифметичного елементів
a1=sort(a)	Впорядкування елементів вектора за збільшенням
[a1,ind]=sort(a)	Другий вихідний аргумент ind є вектором з ці length(a), який відповідає виконаним перестановка

Вказівку номерів елементів вектора можна використовувати і при введенні векторів, послідовно додаючи нові елементи (не обов'язково в порядку зростання їх номерів). Команди:

```
>> h=10;
```

```
>> h(2)=20;
>> h(4)=40;
```
приводять до утворення вектора:
```
>> h
h =
10 20 0 40
```
Зауважте, що для введення першого елементу h не обов'язково вказувати його індекс, оскільки при виконанні оператора h=1 створюється вектор (масив розміру один на один). Наступні оператори привласнення приводять до автоматичного збільшення довжини вектора h, а пропущені елементи (у нашому випадку h(3)) набувають значення нуль.

Індексація двокрапкою дозволяє виділити що йдуть підряд елементи в новий вектор. Початковий і кінцевий номери вказуються в круглих дужках через двокрапку, наприклад:
```
>> z=[0.2 -3.8 7.9 4.5 7.2 -8.1 3.4];
>> znew=z(3:6)
znew =
7.9000 4.5000 7.2000 -8.1000
```
Індексація вектором служить для виділення елементів із заданими індексами в новий вектор. Індексний вектор повинен містити номери необхідних елементів, наприклад:
```
>> ind=[3 5 7];
>> znew=z(ind)
znew =
7.9000 7.2000 3.4000
```
Конструювання нових векторів з елементів наявних векторів проводиться за допомогою квадратних дужок. Наступний оператор призводить до утворення вектора, в якому пропущений п'ятий елемент вектора z
```
>> znew=[z(1:4) z(6:end)]
znew =
0.2000 -3.8000 7.9000 4.5000 -8.1000 3.4000
```

Завдання для самостійної роботи

Для заданих векторів *a* і *b* довжини *n*:

1. обчислити їх суму, різницю і скалярний добуток;
2. утворити вектор $c = [a_1, a_2, ..., a_n, b_1, b_2, ..., b_n]$, визначити його максимальний і мінімальний елементи і поміняти їх місцями;
3. упорядкувати вектор за збільшенням і убуванню;
4. переставити елементи вектора в зворотному порядку і записати результат в новий вектор;
5. знайти векторний добуток $u = [a_1, a_3, a_4]$ і $v = [b_2, b_3, b_4]$.

Варіанти

a = [0.5 3.7 6.0 -4.3 1.2 -2.7 2.4 2.2]; *b* = [3.6 7.0 7.0 5.4

a = [-4.8 -1.3 -1.0 0.7 4.0 5.8 4.3 -8.0]; *b* = [-1.1 -1.9 7.1 -2
a = [1.0 -3.9 -2.3 -3.3 -1.7 2.2 -0.6 1.8]; *b* = [2.7 -2.7 -2.2 4.
a = [-2.4 3.3 -0.1 3.6 7.4 -2.8 0.3 2.2]; *b* = [6.3 0.6 4.3 -3.7
a = [8.4 -5.9 -6.5 -0.9 6.9 -1.7 1.7 0.8]; *b* = [-0.0 2.0 -1.5 7.
a = [5.3 6.8 -7.1 6.8 -4.0 -2.3 -4.4 -0.2]; *b* = [7.5 -1.5 -4.9 -4
a = [1.2 -4.1 -0.8 -0.7 -2.2 1.7 3.3 -6.1]; *b* = [-1.5 2.2 1.0 -4.
a = [6.6 -5.0 -2.7 8.3 3.8 1.9 1.1 2.7]; *b* = [-1.0 3.2 4.2 -6.
a = [-1.9 0.4 1.8 4.2 -3.8 -4.7 4.0 -2.1]; *b* = [-8.7 -4.2 -1.4 2
a = [0.9 1.7 -3.2 -3.8 7.3 6.0 -0.2 8.6]; *b* = [0.6 -0.4 -6.9 -2

Обчислити значення функції на відрізку в заданому числі *N*, рівномірно віддалених один від одного точок.

Варіанти

1. $y(x)=\dfrac{\sin x\cos x}{x^2+1}$ $[0,2\pi]$

2. $y(x)=\ln(x+1)\sqrt{e^x+e^{-x}}$ $[-0.2,4]$

3. $y(x)=x^2\tan\sqrt{\arcsin x}$ $\left[0,\dfrac{1}{3}\right]$

4. $y(x)=x\sin x+x^3\dfrac{e^x}{x+1}$ $[0,1]$

5. $y(x)=\dfrac{1}{1+\dfrac{x}{\sqrt{1+x}}}$ $[0,3]$

6. $y(x)=\dfrac{e^{\sin x}+e^{\cos x}}{x^2}$ $[\pi,3\pi]$

7. $y(x)=\cot\tan(x^2+1)\cdot(\sin 2x+\cos 2x)$ $[-1,1]$

8. $y(x)=\log_2\left(x^2+1\right)\cdot\sin\dfrac{1}{x^2+1}$ $[-1,1]$

9. $y(x)=\left|x^3+2x^2-3\right|\sin\pi x$ $[-2,2]$

10 $y(x)=\dfrac{\sqrt[3]{x+1}}{\sqrt{|x|+\dfrac{1}{2}}}\cdot\dfrac{\sin x+1}{\cos x+2}$ $[-2,2]$

§ 3. Матриці

Матриці невеликих розмірів зручно вводити з командного рядка. Існує три способи введення матриць. Наприклад, матрицю

$$A = \begin{bmatrix} 0.7 & -2.5 & 9.1 \\ 8.4 & 0.3 & 1.7 \\ -3.5 & 6.2 & 4.7 \end{bmatrix}$$

можна ввести таким чином: набрати в командному рядку (розділяючи елементи рядка матриці пропусками): A=[0.7 2.5 9.1 і натиснути <Enter>. Курсор переміщається в наступний рядок (символ запрошення командного рядка >> відсутній). Елементи кожного наступного рядка матриці набираються через пропуск, а введення рядка завершується натисненням на <Enter>. При введенні останнього рядка в кінці ставиться закриваюча квадратна дужка:

```
>> A=[0.7 -2.5 9.1
8.4 0.3 1.7
-3.5 6.2 4.7]
```

Якщо після закриваючої квадратної дужки не ставити крапку з комою для придушення висновку в командне вікно, то матриця виведеться у вигляді таблиці.

Інший спосіб введення матриці заснований на тому, що матрицю можна розглядати як вектор-стовпець, кожен елемент якого є рядком матриці. Оскільки крапка з комою використовується для розділення елементів вектор-стовпця, то введення, наприклад, матриці

$$B = \begin{bmatrix} 6.1 & 0.3 \\ -7.9 & 4.4 \\ 2.5 & -8.1 \end{bmatrix}$$

здійснюється оператором привласнення:

```
>> B=[6.1 0.3; -7.9 4.4; 2.5 -8.1];
```

Введіть матрицю *B* і відобразіть її вміст в командному вікні, набравши в командному рядку B і натиснувши <Enter>.

Очевидно, що допустиме таке трактування матриці, при якій вона вважається вектор-рядком, кожен елемент якого є стовпцем матриці. Отже, для введення матриці

$$C = \begin{bmatrix} 0.4 & -7.2 & 5.3 \\ 0.1 & -2.1 & -9.5 \end{bmatrix}$$

досить скористатися командою:

```
>> C=[[0.4; 0.1] [-7.2; -2.1] [5.3; -9.5]]
```

Звернете увагу, що внутрішні квадратні дужки дійсно потрібні. Оператор C=[0.4; 0.1 -7.2; -2.1 5.3; -9.5] є неприпустимим і приводить до повідомлення про помилку, оскільки виявляється, що в першому рядку матриці міститься тільки один елемент, в другій і третій по два, а в четвертій — знову один.

Скористайтеся командою whos для отримання інформації про змінних A, B і C робочого середовища. У командне вікно виводиться таблиця з інформацією про розміри масивів, пам'ять, необхідну для зберігання кожного з масивів, і тип — double array.

Функція size дозволяє встановити розміри масивів, вона повертає результат у вигляді вектора, перший елемент якого рівний числу рядків, а другою стовпців:

```
>> s=size(B)
s =
3 2
```

Складання і віднімання матриць однакових розмірів проводиться з використанням знаків +, -. Зірочка * служить для обчислення матричного добутку, причому відповідні розміри матриць повинні збігатися, наприклад:

```
>> P=A*B
P =
46.7700 -84.5000
53.1200 -9.9300
-58.5800 -11.8400
```

Допустиме множення матриці на число і числа на матрицю, при цьому відбувається множення кожного елементу матриці на число і результатом є матриця тих же розмірів, що і вихідна. Апостроф ' призначений для транспонування речовинної матриці або знаходження зв'язаною до комплексної матриці. Для піднесення квадратної матриці до ступеня застосовується знак ^.

Наприклад, обчислення матричного виразу $R = (A - BC)^3 + ABC$ здійснюється командою:

```
>> R=(A-B*C)^3+A*B*C
R =
1.0e+006 *
-0.0454 0.1661 -0.6579
0.0812 -0.2770 1.2906
-0.0426 0.1274 -0.7871
```

MatLab володіє різноманіттям різних функцій і способів для роботи з матричними даними. Для звернення до елементу двовимірного масиву слід вказати його рядковий і стовпцеві індекси в круглих дужках після імені масиву, наприклад:

```
>> C(1,2)
```

ans =
-7.2000

Індексація двокрапкою дозволяє отримати частину матриці — рядок, стовпець або блок, наприклад:

```
>> c1=A(2:3,2)
c1 =
0.3000
6.2000
>> r1=A(1,1:3)
r1 =
0.7000 -2.5000 9.1000
```

Для звернення до всього рядка або всього стовпця не обов'язково указувати через двокрапку початковий (перший) і кінцевий індекси, тобто оператори r1=A(1,1:3) і r1=A(1:) еквівалентні. Для доступу до елементів рядка або стовпця від заданого до останнього можна використовувати end, так само як і для векторів: A(1,2:end). Виділення блоку, що складається з декількох рядків і стовпців, вимагає індексації двокрапкою як по першому вимірюванню, так і по другому. Нехай в масиві T зберігається матриця:

$$T = \begin{bmatrix} 1 & 7 & -3 & 2 & 4 & 9 \\ 0 & -5 & -6 & 3 & -8 & 7 \\ 2 & 4 & 5 & -1 & 0 & 3 \\ -6 & -4 & 7 & 2 & 6 & 1 \end{bmatrix}$$

Для виділення її елементів з другого рядка по третю і з другого стовпця по четвертий досить використовувати оператор:

```
>> T1=T(2:3,2:4)
T1 =
-5 -6 3
4 5 -1
```

Індексація двокрапкою так само дуже корисна при різних перестановках в масивах. Зокрема, для перестановки першого і останнього рядків в довільній матриці, що зберігається в масиві A, підійде послідовність команд:

```
>> s=A(1:);
>> A(1:)=A(end:);
>> A(end:)=s;
```

MatLab підтримує таку операцію, як викреслювання рядків або стовпців з матриці. Блоку, що достатньо віддаляється, привласнити порожній масив, що задається квадратними дужками. Наприклад, викреслювання другого і третього рядка з масиву T, запровадженого вище, проводиться наступною командою:

```
>> T(2:3:)=[]
T =
```

1 7 -3 2 4 9
-6 -4 7 2 6 1

Індексація двокрапкою спрощує заповнення матриць, що мають певну структуру. Припустимо, що потрібно створити матрицю

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Перший крок полягає у визначенні нульової матриці розміру п'ять на п'ять, потім заповнюються перший і останній рядки і перший і останній стовпці:

```
>> W(1:5,1:5)=0;
>> W(1:)=1;
>> W(end:)=1;
>> W(:,1)=1;
>> W(:,end)=1;
```

Перевірте, що в результаті виходить необхідна матриця. Низка вбудованих функцій, приведених в табл.3.1, дозволяє ввести стандартні матриці заданих розмірів. Звернете увагу, що у всіх функціях, крім diag, допустимо вказувати розміри матриці наступними способами:

- числами через кому (у двох вхідних аргументах);
- одним числом, результат — квадратна матриця;
- вектором з двох елементів, рівних числу рядків і стовпців.

Останній варіант дуже зручний, коли потрібно створити стандартну матрицю тих же розмірів, що і деяка наявна матриця. Якщо, наприклад, A була визначена раніше, то команда I=eye(size(A)) призводить до появи по-одиночній матриці, розміри якої збігатися з розмірами A, оскільки функція size повертає розміри матриці у векторі.

Розберемо, як отримати трьох-діагональну матрицю розміру сім на сім, приведену нижче, з використанням функцій MatLab.

$$T = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 5 & 3 & -3 & 0 & 0 & 0 \\ 0 & 0 & 5 & 4 & -4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 5 & -5 & 0 \\ 0 & 0 & 0 & 0 & 5 & 6 & -6 \\ 0 & 0 & 0 & 0 & 0 & 5 & 7 \end{bmatrix}$$

Введемо вектор v з цілими числами від одного до семи і використаємо його для створення діагональної матриці і матриці із зміщеною на одиницю вгору діагоналлю. Вектор довжини шість, що містить п'ятірки, заповнюється, наприклад, так: 5*ones(1,6). Цей вектор вкажіть в першому аргументі функції diag, а мінус одиницю — в другому і отримаєте третю допоміжну матрицю. Тепер досить відняти з першої матриці другу і скласти з третьою:

```
>> T=diag(v) -diag(v(1:6),1)+diag(5*ones(1,6)-1)
```

Таблиця. 3.1 Функції для створення стандартних матриць

Функція	Результат і приклади викликів
zeros	Нульова матриця F=zeros(4,5) F=zeros(3) F=zeros([3 4])
Eye	Одинична прямокутна матриця (одиниці розташовані на голов I=eye(5,8) I=eye(5) I=eye([5 8])
Ones	Матриця, що цілком складається з одиниць E=ones(3,5) E=ones(6) E=ones([2 5])
Rand	Матриця, елементи якої — випадкові числа, рівномірно роз (0,1) R=rand(5,7) R=rand(6) R=rand([3 5])
Randn	Матриця, елементи якої — випадкові числа, розподілені по н нульовим середнім і дисперсією рівній одиниці N=randn(5,3) N=randn(9) N=randn([2 4])
Diag	1) діагональна матриця, елементи якої задаються у вхідному а D=diag(v) 2) діагональна матриця із зміщеною на до позицій діагоналлю вгору, негативні — вниз), результатом є квадратна length(v)+abs(k) D=diag(v,k) 3) виділення головної діагоналі з матриці у вектор d=diag(A) 4) виділення k-ої діагоналі з матриці у вектор d=diag(A, до)

По-елементні обчислення з матрицями проводяться практично аналогічно як і для векторів з необхідністю, зрозуміло, стеження за збігом розмірів матриць:

A.*B, A./B— по-елементне множення і ділення;

A.^p — по-елементне піднесення до ступеня, p — число;

A.^B — піднесення елементів матриці A до ступенів, рівних відповідним елементам матриці B;

A.' — транспонування матриці (для речовинних матриць A' і A.' приводять до однакових результатів);

Іноді потрібно не просто транспонувати матрицю, але і "розвернути" її. Розворот матриці на 90^o проти годинникової стрілки здійснює функція rot90:

```
>> Q=[1 2;3 4]
Q =
1 2
3 4
>> R=rot90(Q)
R =
2 4
1 3
```

Допустимо записувати суму і різницю матриці і числа, при цьому складання або віднімання застосовується, відповідно, до всіх елементів матриці. Виклик математичної функції від матриці призводить до матриці того ж розміру, на відповідних позиціях якої коштують значення функції від елементів початкової матриці.

У MatLab визначені і матричні функції, наприклад, sqrtm призначена для обчислення квадратного кореня. Знайдіть квадратний корінь з матриці

$$K = \begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix}$$

і перевірте отриманий результат, звівши його в квадрат (за правилом матричного множення, а не по-елементно):

```
>> K=[3 2; 1 4];
>> S=sqrtm(K)
S =
1.6882 0.5479
0.2740 1.9621
>> S*S
ans =
3.0000 2.0000
1.0000 4.0000
```

Матрична експонента обчислюється з використанням expm. Спеціальна функція funm служить для обчислення довільної матричної функції.

Всі функції обробки даних, приведені в табл. 2.1, можуть бути застосовані і до двовимірних масивів. Основна відмінність від обробки

векторних даних полягає в тому, що ці функції працюють з двовимірними масивами по стовпцях, наприклад, функція sum підсумовує елементи кожного із стовпців і повертає вектор-рядок, довжина якого рівна числу стовпців початкової матриці:

```
>> M=[1 2 3; 4 5 6; 7 8 9]
M =
1 2 3
4 5 6
7 8 9
>> s=sum(M)
s =
12 15 18
```

Якщо як другий вхідний аргумент sum вказати 2, то підсумовування відбудеться по рядках. Для обчислення суми всіх елементів матриці потрібно двічі застосувати sum:

```
>> s=sum(sum(M))
s =
45
```

Дуже зручною можливістю MatLab є конструювання матриці з матриць менших розмірів. Нехай задані матриці:

$$M_1=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\ M_2=\begin{bmatrix}-2 & -3 & -5\\ -1 & -5 & -6\end{bmatrix}\ M_3=\begin{bmatrix}9 & 8\\ -7 & -5\\ 1 & 2\end{bmatrix}\ M_4=\begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$$

Потрібно скласти блокову матрицю *M*: $M=\left[\begin{array}{c|c} M_1 & M_2\\ \hline M_3 & M_4\end{array}\right]$

Можна вважати, що має розміри два на два, а кожен елемент є, відповідно, матрицею, або M_4. Отже, для отримання в робочому середовищі MatLab масиву *M* з матрицею потрібно використовувати оператора:

```
>> M=[M1 M2; M3 M4]
```

Завдання для самостійної роботи

1. Введіть матриці

$$A=\begin{bmatrix}-9.8 & 4.4 & 1.3\\ -5.7 & 0.1 & 0.8\\ 2.4 & 4.4 & 8.6\end{bmatrix}\ B=\begin{bmatrix}1 & 0 & 2\\ 3 & 0 & -1\\ 5 & 2 & 2\\ 8 & 9 & 0\end{bmatrix}\ C=\begin{bmatrix}0.1 & 0.2 & -1.3 & 0.7\\ -0.2 & 0.3 & 2.2 & 0.8\\ 1.9 & 2.3 & 6.5 & 4.9\end{bmatrix}$$

і знайдіть значення наступних виразів.

Варіанти

1. $\left(A^3+CB\right)\left(A^2-3CB\right)^T$

2. $A^4 + 2A^3 - ACB$

3. $BAC - 4C^T B^T$

4. $3BA^3C - BAC + 2BC$

5. $-3C^T AC - BB^T$

6. $\left(BCB - 4C^T\right)A^4$

7. $\left(AB^T - C\right)\left(C + AB^T\right)^T - 3A$

8. $\left(AB^T B\right)^4 - 2A^3 + CC^T$

9. $C\left(BB^T + C^T C\right)C^T - 8A$

10. $2AA^T - (CB)^2 + 4A$

2. За допомогою вбудованих функцій для заповнення стандартних матриць, індексації двокрапкою і, можливо, повороту, транспонування або викреслювання отримаєте наступні матриці:

Варіанти

1. $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 0 & 7 & 1 \\ 1 & 0 & 2 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 2 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

2. $\begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 5 \\ -1 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 5 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 7 & 0 \\ 5 & 0 & 0 & 0 & 0 & -1 & 8 \end{bmatrix}$

3. $\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

4. $\begin{bmatrix} 4 & 3 & 3 & 3 & 3 & 3 & 9 \\ 3 & 4 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 4 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 4 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 4 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 4 & 3 \\ 9 & 3 & 3 & 3 & 3 & 3 & 4 \end{bmatrix}$

5. $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 1 \\ 2 & 0 & 0 & 0 & 7 & 0 & 1 \\ 3 & 0 & 0 & 7 & 0 & 7 & 1 \\ 4 & 0 & 7 & 0 & 7 & 0 & 1 \\ 5 & 7 & 0 & 7 & 0 & 0 & 1 \\ 6 & 0 & 7 & 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 1 \end{bmatrix}$

6. $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

7.

8. $\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

9. $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$

10. $\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$

3. Обчислити значення функції для всіх елементів матриці і записати результат в матрицю того ж розміру, що і вихідна.

Варіанти

1. $f(x) = x^3 - 2x^2 + \sin x - 4;$	$A = \begin{bmatrix} 9.33 & -4.01 & 8.19 & 2.64 \\ 0.55 & 3.81 & 3.32 & 5.07 \end{bmatrix}$
2. $f(x) = \dfrac{e^x - x}{e^x + x};$	$A = \begin{bmatrix} 9.32 & 0.21 & -9.89 & 3.11 \\ 0.54 & 4.99 & 5.01 & -0.03 \end{bmatrix}$
3. $f(x) = \sqrt{1 + \sqrt{\lvert x \rvert^3 + 1}};$	$A = \begin{bmatrix} -1.54 & 0.49 & 3.11 & 2.99 \\ 4.05 & -5.85 & 3.72 & 0.11 \end{bmatrix}$
	$A = \begin{bmatrix} -9.04 & 3.36 & 3.09 & -2.49 \\ -4.33 & -5.09 & 9.74 & 1.65 \end{bmatrix}$

4. $f(x)=e^{x}\sin x-e^{-x}\cos x;$	$A=\begin{bmatrix}0.33 & 0.95 & 7.12 & -9.22\\ -0.64 & 3.76 & 1.34 & -0.03\end{bmatrix}$
5. $f(x)=\ln(\lvert x\rvert)\sin\pi x;$	$A=\begin{bmatrix}-4.53 & -2.12 & -6.54 & -3.21\\ 3.43 & 7.43 & -0.25 & 1.64\end{bmatrix}$
6. $f(x)=e^{x^2+x+1};$	$A=\begin{bmatrix}0.23 & 3.89 & -4.23 & -7.25\\ 5.84 & 5.13 & -0.89 & 3.55\end{bmatrix}$
7. $f(x)=\dfrac{\sqrt[3]{x^2-1}}{\lvert x\rvert+3};$	$A=\begin{bmatrix}-5.84 & 9.84 & 0.23 & 1.59\\ -9.25 & -0.25 & 1.54 & 0.43\end{bmatrix}$
8. $f(x)=\dfrac{1}{1+\dfrac{1+x}{1-x^2}};$	$A=\begin{bmatrix}0.64 & 6.34 & 0.32 & -4.23\\ 1.19 & 3.23 & 1.54 & 0.43\end{bmatrix}$
9. $f(x)=\dfrac{x^3+\sin x}{x^3\cos x}\sqrt{e^{x}+1};$	$A=\begin{bmatrix}\pi & 2.2\pi & -2\pi & 0.3\pi\\ 3\pi & -\pi & 0.1\pi & 5\pi\end{bmatrix}$
10. $f(x)=\arcsin(\cos x^2);$	

4. Сконструювати блокові матриці (використовуючи функції для заповнення стандартних матриць) і застосувати функції обробки даних і по-елементні операції для знаходження заданих величин.

Варіанти

1. $A=\begin{bmatrix}1 & 0 & 0 & 4 & 0 & 0\\ 0 & 1 & 0 & 0 & 4 & 0\\ 0 & 0 & 1 & 0 & 0 & 4\\ 2 & 2 & 2 & 3 & 3 & 3\\ 2 & 2 & 2 & 3 & 3 & 3\\ 2 & 2 & 2 & 3 & 3 & 3\end{bmatrix}$ $m=\max_{j=1..6}\left\{\sum_{i=1}^{6}a_{ij}{}^{2}\right\}$

2. $A = \begin{bmatrix} 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \end{bmatrix}$ $s = \sum_{i=1}^{6} \sum_{j=1}^{6} \left| a_{ij} \right|$

3. $A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ -1 & -2 & -3 & -4 & -5 & -6 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 \end{bmatrix}$ $m = \min_{i,j=1..6} a_{ij}^{3}$

4. $A = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 3 & 0 & 0 \\ 2 & 2 & 2 & 0 & 3 & 0 \\ 2 & 2 & 2 & 0 & 0 & 3 \end{bmatrix}$ $s = \sum_{k=1}^{6} a_{kk}^{3}$

5. $A = \begin{bmatrix} 1 & 1 & -3 & -3 & -3 & -3 \\ 1 & 1 & -3 & -3 & -3 & -3 \\ -3 & -3 & 2 & 0 & 0 & 0 \\ -3 & -3 & 0 & 2 & 0 & 0 \\ -3 & -3 & 0 & 0 & 2 & 0 \\ -3 & -3 & 0 & 0 & 0 & 2 \end{bmatrix}$ $s = \sum_{i=1}^{5} a_{ii+1}$

6. $A = \begin{bmatrix} -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 4 & 4 \end{bmatrix}$ $s = \sum_{i=1}^{6} a_{ii} + \sum_{i=1}^{5} a_{ii+1}$

7. $A = \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 & 0 & 4 \\ 3 & 0 & 0 & 0 & 0 & 4 \\ 1 & 1 & 1 & 1 & 1 & -2 \end{bmatrix}$ $s = \sum_{i=1}^{6} \sum_{j=1}^{6} \sin\left(\frac{\pi}{6} a_{ij}^2\right)$

8. $A = \begin{bmatrix} 1 & 2 & 3 & -1 & 0 & 0 \\ 1 & 2 & 3 & 0 & -1 & 0 \\ 1 & 2 & 3 & 0 & 0 & -1 \\ 0 & 0 & 7 & 2 & 2 & 2 \\ 0 & 7 & 0 & 2 & 2 & 2 \\ 7 & 0 & 0 & 2 & 2 & 2 \end{bmatrix}$ $m = \max_{i=1..6} \min_{j=1..7} a_{ij}$

9. $A = \begin{bmatrix} 1 & 0 & 0 & 1 & 3 & 3 \\ 0 & 1 & 1 & 0 & 3 & 3 \\ 0 & 1 & 1 & 0 & 3 & 3 \\ 1 & 0 & 0 & 1 & 3 & 3 \\ -2 & -2 & -2 & -2 & 4 & 4 \\ -2 & -2 & -2 & -2 & 4 & 4 \end{bmatrix}$ $s = \sum_{i=1}^{6} \max_{j=1..6} \left(a_{ij} + a_{ji}\right)$

10. $A = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ -1 & -1 & -1 & 4 & 0 & 0 \\ -1 & -1 & -1 & 0 & 4 & 0 \\ -1 & -1 & -1 & 0 & 0 & 4 \end{bmatrix}$ $p = \prod_{i=1}^{6} \sum_{j=1}^{6} \left(a_{ij}\right)^{a_{ij}}$

.

§ 4. Графіка і візуалізація даних

MatLab володіє широким набором засобів для побудови графіків функцій одній і два змінних і відображення різних типів даних. Всі графіки виводяться в графічні вікна з своїми меню і панелями інструментів. Вид графіків визначається аргументами графічних команд і потім може бути змінений за допомогою інструментів графічного вікна. Важливо розуміти, що для побудови графіків функцій на деякій області зміни аргументів слід обчислити значення функції в точках області, часто для отримання хороших графіків слід використовувати достатньо багато точок.

Розберемо спочатку, як отримати графік функції однієї змінної, наприклад:

$$f(x) = e^x \sin \pi x + x^2$$

на відрізку [-2,2]. Для побудови графіка функції використовується одна з графічних функцій MatLab. Достатньо універсальною графічною функцією є plot. У найпростішому випадку вона викликається з двома вхідними аргументами — парою x і f (т. е. plot виводить залежність елементів одного вектора від елементів іншого). Послідовність команд, записана нижче, приводить до появи графічного вікна Figure No.1 з графіком функції.

```
>> x=[-2:0.05:2];
>> f=exp(x).*sin(pi*x)+x.^2;
>> plot(x,f)
```

Тип лінії, колір і маркери визначаються значенням третього додаткового аргументу функції plot. Цей аргумент вказується в апострофах, наприклад, виклик plot(x,f,'ro:') призводить до побудови графіка червоною пунктирною лінією, розміченою круглими маркерами. Звернете увагу, що абсциси маркерів визначаються значеннями елементів вектора x. Всього в додатковому аргументі може бути заповнена три позиції, відповідні квітну, типу маркерів і стилю лінії. Позначення для них приведені в табл. 4.1. Порядок позицій може бути довільний, допустимо вказувати тільки один або два параметри, наприклад, колір і тип маркерів. Подивіться на результат виконання наступних команд: plot(x,f,'g'), plot(x,f,'ko'), plot(x,f,':').

Функція plot має достатньо універсальний інтерфейс, вона, зокрема, дозволяє відображати графіки декількох функцій на одних осях. Нехай потрібно вивести графік не тільки *f*(x), але і на відрізку [-2, 2]. Спочатку необхідно обчислити значення *g(x):*

```
>> g=exp(-x.^2).*sin(5*pi*x);
```

а потім викликати plot, вказавши через кому пари x, f і x, g і, при бажанні, властивості кожній з ліній:

```
>> plot(x,f,'ko-', x,g,'k:')
```

Таблиця Д4.1 Скорочення для кольору, типу маркерів і стилю ліній

Колір		**Тип маркера**	
Y	Жовтий	.	точка
M	Рожевий	o	Кружок
З	Блакитною	x	Хрестик
R	Червоний	+	Знак плюс
G	Зелений	*	Зірочка

B	Синій	s	Квадрат
W	Білий	d	Ромб
До	Чорний	v	Трикутник вершиною вниз
Тип лінії		^	Трикутник вершиною вгору
-	Суцільна	<	Трикутник вершиною вліво
:	Пунктирна	>	Трикутник вершиною вправо
-.	Штріх-пунктірная	p	П'ятикутна зірка
--	Штрихова	h	Шестикінечна зірка

Допускається побудова довільного числа графіків функцій, властивості всіх ліній можуть бути різними. Крім того, області побудови кожною з функцій не обов'язково повинні збігатися, але тоді слід використовувати різних вектора для значень аргументів і обчислювати значення функцій від відповідних векторів. Для отримання графіка шматково-заданої функції:

$$y(x)=\begin{cases}\sin x & -4\pi \le x \le -\pi \\ 3(x/\pi+1)^2 & -\pi < x \le 0 \\ 3e^{-x} & 0 < x \le 5\end{cases}$$

досить виконати послідовність команд:

```
>> x1=[-4*pi:pi/10:-pi];
>> y1=sin(x1);
>> x2=[-pi:pi/30:0];
>> y2=3*(x2/pi+1).^2;
>> x3=[0:0.02:5];
>> y3=3*exp(-x3)
>> plot(x1,y1,x2,y2,x3,y3)
```

Зауважте, що графіки гілок функції відображаються різними кольорами. Можна було поступити і по-іншому, а саме: після заповнення x1, y1, x2, y2, x3 і y3 зібрати вектор x для значень аргументу і вектор для значень *y(x)* і побудувати залежність від x:

```
>> x=[x1 x2 x3];
>> y=[y1 y2 y3];
>> plot(x,y)
```

Нескладно здогадатися, як побудувати графік параметрично заданої функції, використовуючи ту обставину, що plot відображає залежність одного вектора від іншого. Нехай потрібно отримати графік астроїди: $x(t)=\cos^3 t, y(t)=\sin^3 t, t\in[0,2\pi]$. Слід задати вектор t, потім у вектори X, Y занести значення *x(t)*, *y(t)* і скористатися plot для відображення залежності Y від X:

```
>> t=[0:pi/20:2*pi];
>> x=cos(t).^3;
>> y=sin(t).^3;
>> plot(x,y)
```

Функція comet дозволяє простежити за рухом точки по траєкторії параметрично заданої лінії. Виклик comet(x,y) наводить до появи графічного вікна, на осях якого малюється переміщення точки у вигляді руху комети з хвостом. Управління швидкістю руху здійснюється зміною кроку при визначенні вектора значень параметра.

У MatLab є графічні функції, призначені для відображення графіків в логарифмічному і напівлогарифмічному масштабах:

- Loglog (логарифмічний масштаб по обох осях);
- Semilogx (логарифмічний масштаб тільки по осі абсцис);
- Semilogy (логарифмічний масштаб тільки по осі ординат).

Вхідні аргументи цих функцій задаються так само, як і при використанні plot. Для порівняння поведінки двох функцій із значеннями різних порядків зручно застосовувати plotyy. Функція plotyy викликається від двох пар вхідних аргументів (векторів) і приводить до появи двох ліній графіків, кожною з яких відповідає своя вісь ординат.

Графіки оформляються в MatLab спеціальними командами і функціями. Сітка наноситься на осі командою grid on, а забирається за допомогою grid off. Заголовок розміщується в графічному вікні за допомогою функції title, вхідним аргументом якої являється рядок, ув'язнена в апострофи:

```
>> title('Результати експерименту')
```

За наявності декількох графіків потрібно розташувати легенду звернувшись до legend. Надписи, легенди, ув'язнені, в апострофи вказуються у вхідних аргументах функції legend, їх число повинне співпадати з числом ліній графіків. Крім того, останній додатковий вхідний аргумент визначає положення легенди:

–1 — поза графіком в правому верхньому кутку графічного вікна;

0 — вибирається краще положення в межах графіка так, щоб якомога менше перекривати самі графіки;

1 — у верхньому правому кутку графіка (це положення використовується за умовчанням);

2 — у верхньому лівому кутку графіка;

3 — в нижньому лівому кутку графіка;

4 — в нижньому правому кутку графіка.

Функції xlabel і ylabel призначені для підписів до осей, їх вхідні аргументи так само полягають в апострофи.

Звернемося тепер до візуалізації векторних і матричних даних. Найпростіший спосіб відображення векторних даних полягає у використанні функції plot з вектором як вхідний аргумент. При цьому графік, що виходить у вигляді ламаної лінії, символізує залежність значень елемен-

тів вектора від їх індексів. Другий додатковий аргумент може визначати колір, стиль лінії і тип маркерів, наприклад: plot(x,'ko'). Виклик функції plot від матриці приводить до декількох графіків, їх число співпадає з числом стовпців матриці, а кожний з них є залежністю елементів стовпця від їх рядкових індексів. Колір і стиль ліній і тип маркерів відразу для всіх ліній так само визначається другим додатковим аргументом.

Наглядним способом представлення матричних і векторних даних є різноманітні діаграми. Проста стовпцева діаграма будується за допомогою функції bar:

```
>> x=[0.7 2.1 2.5 1.9 0.8 1.3];
>> bar(x)
```

Додатковий числовий аргумент bar указує на ширину стовпців (за умовчанням він рівний 0.8), а значення великі одиниці, наприклад bar(x,1.2), приводять до часткового перекриття стовпців. Вказівка матриці у вхідному аргументі bar приводить до побудови групової діаграми, число груп співпадає з числом рядків матриці, а усередині кожної групи стовпчиками відображаються значення елементів рядків.

Кругові діаграми векторних даних виходять за допомогою функції pie, яка має деякі особливості в порівнянні з bar. Розрізняються два випадки:

1. якщо сума елементів вектора більше або рівна одиниці, то виводиться повна кругова діаграма, площа кожного її сектора пропорційна величині елементу вектора;
2. якщо сума елементів вектора менше одиниці, то результатом є неповна кругова діаграма, в якій площа кожного сектора пропорційна величині елементів вектора, в припущенні що площа всього круга рівна одиниці.

Порівняєте, наприклад pie([0.1 0.2 0.3]) і pie([1 2 3]). Можна відокремити деякі сектори від всього круга діаграми, для чого слід викликати pie з другим аргументом — вектором тієї ж довжини, що і вихідний. Ненульові елементи другого вектора відповідають відокремлюваним секторам. Наступний приклад показує, як відокремити від діаграми сектор, відповідний найбільшому елементу вектора x:

```
>> x=[0.3 2 1.4 0.5 0.9];
>> [m,k]=max(x);
>> v=zeros(size(x));
>> v(k)=1;
>> pie(x,v)
```

Підписи до секторів діаграми указуються в другому додатковому вхідному аргументі, який полягає у фігурні дужки:

```
>> pie([2400 3450 1800 5100],{'Март','Апрель','Май','Июнь'})
```

Функції bar і pie мають аналоги:

- barh — побудова стовпцевої діаграми з горизонтальним розташуванням стовпців;

- bar3, pie3 — побудова об'ємних діаграм.

При обробці великих масивів векторних даних часто потрібно отримати інформацію про те, яка частина даних знаходиться в тому або іншому інтервалі. Функція hist призначена для відображення гістограми даних і знаходження числа даних в інтервалах. Вхідним аргументом hist являється вектор з даними, а вихідним — вектор, що містить кількість елементів, що потрапили в кожний з інтервалів. За умовчанням береться десять рівних інтервалів. Наприклад, виклик hist(randn(1,5000)) наводить до появи на екрані гістограми даних, розподілених по нормальному закону, а n=hist(randn(1,5000)) до заповнення вектора n довжини десять (при цьому гістограма не будується). Число інтервалів указується в другому додатковому аргументі hist. Можна задати інтервали, використавши як другий аргумент не число, а вектор, що містить центри інтервалів. Зручніше задавати інтервали не центрами, а межами. В цьому випадку потрібно спочатку визначити кількість елементів в інтервалах за допомогою функції histc, а потім застосувати bar із спеціальним аргументом 'histc', наприклад:

```
>> x=randn(1,10000);
>> int=[-2:0.5:2];
>> n=histc(x,int);
>> bar(int,n,'histc')
```

Візуалізація функцій два змінних в MatLab може бути здійснена декількома способами, але всі вони припускають однотипні попередні дії. Розглянемо тут тільки побудову графіків функцій два змінних на прямокутній області визначення. Припустимо, що потрібно отримати поверхню функції $z(x,y) = e^{-x}\sin(\pi y)$ на прямокутнику $x \in [-1,1]$ $y \in [0,2]$. Перший крок полягає в завданні сітки на прямокутнику, т. е. крапок, які використовуватимуться для обчислення значень функції. Для генерації сітки передбачена функція meshgrid, що викликається від двох вхідних аргументів — векторів, задаючи розбиття по осях *x* і *y*. Функція meshgrid повертає два вихідних аргументу, що є матрицями.

```
>> [X,Y]=meshgrid(-1:0.1:1,0:0.1:2);
```

Матриця X складається з однакових рядків, рівних першому вхідному аргументу вектору в meshgrid, а матриця Y — з однакових стовпців, співпадаючих з другим вектором в meshgrid. Такі матриці виявляються необхідними на другому кроці при заповненні матриці Z, кожен елемент якої є значенням функції z*(*x,y*)* в точках сітки. Нескладно зрозуміти, що використання по-елементних операцій при обчисленні функції z*(*x,y*)* приводить до необхідної матриці:

```
>> Z=exp(-X).*sin(pi*Y);
```

Для побудови графіка *z(x,y)* залишилося викликати відповідну графічну функцію, наприклад:

```
>> mesh(X,Y,Z)
```

На екрані з'являється графічне вікно, що містить каркасну поверхню досліджуваної функції (рис. Д4.1). Звернете увагу, що колір поверхні відповідає значенню функції.

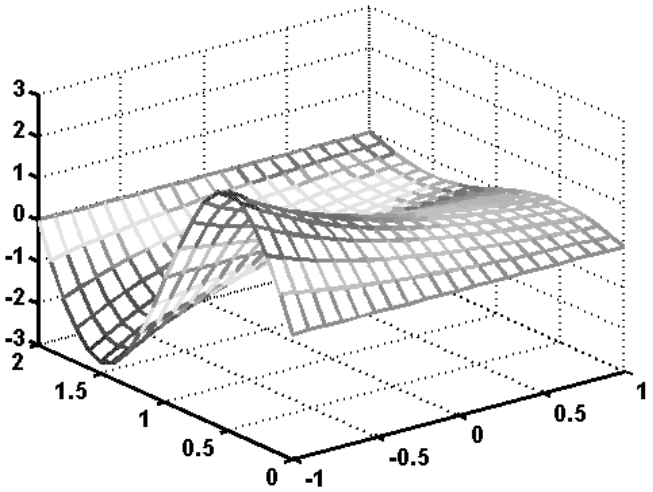

Рис. Д4.1. Приклад графічного вікна

Команда colorbar приводить до відображення в графічному вікні стовпчика, що показує співвідношення між кольором і значенням *z*(*x,y*). Колірні палітри графіка можна змінювати, користуючись функцією colormap, наприклад colormap(gray) відображає графік у відтінках сірого кольору. Деякі колірні палітри приведені нижче:

- bone — схожа на палітру gray, але з легким відтінком синього кольору;
- colorcube — кожен колір змінюється від темного до яскравого;
- cool — відтінки блакитного і пурпурного кольорів;
- copper — відтінки мідного кольору;
- hot плавна зміна: чорний –червоний –оранжевий –жовтий – білий;
- hsv — плавна зміна (як барви веселки);
- jet плавна зміна: синій –голубий –зелений –жовтий – червоний;
- spring — відтінки пурпурного і жовтого;
- summer — відтінки зеленого і жовтого;
- winter — відтінки синього і зеленого;

MatLab надає цілий набір графічних функцій для візуалізації функцій два змінних, серед них:

- surf — залита кольором каркасна поверхня;
- meshc, surfc — поверхні з лініями рівня на площині *xy*;

- contour — плоский графік з лініями рівня;
- contourf — залитий кольором плоский графік з лініями рівня;
- contour3 — поверхня, складена з ліній рівня;
- surfl — освітлена поверхня

Зупинимося докладніше на декількох питаннях. Перший з них: як змінювати установки, визначені за умовчанням, при відображенні функцій лініями рівня за допомогою contour, contourf і contour3. Число ліній рівня задається в четвертому додатковому аргументі, наприклад:

```
>> contourf(X,Y,Z,10)
```

Замість числа ліній рівня можна вказати у векторі ті значення $z(x,y)$, для яких потрібно побудувати лінії рівня:

```
>> contour(X,Y,Z[-0.51 -0.25 -0.01 0.89])
```

Дещо складніше нанести підписи з відповідним значенням $z(x,y)$ до кожної лінії рівня. Для цього доведеться викликати contour з двома вихідними аргументами, перший з них — матриця з інформацією про положення ліній рівня, а другою — вектор з покажчиками на лінії. Отримані змінні слід використовувати як вхідні аргументи функції clabel:

```
>> [CMatr, h] = contour(X, Y, Z[-0.51 -0.25 -0.01 0.89]);
>> clabel(CMatr, h)
```

Залиті кольором каркасні поверхні, побудовані за допомогою sufr і surfc, мають постійний колір в межах кожної комірки. Команда shading interp, що викликається після surf і surfc, служить для плавної зміни кольору в межах комірок і приховання ліній сітки на поверхні. Якщо бажано прибрати сітку і зберегти постійний колір комірок, то досить використовувати shading flat, а shading faceted надає графіку колишній вигляд.

Графічні функції за умовчанням розташовують поверхню так, що спостерігач бачить її частину під деяким кутом, а інша — прихована від погляду. Положення спостерігача визначається двома кутами: азимутом (AZ) і кутом піднесення (EL). Азимут відлічується від осі, протилежної *y*, а кут піднесення — від площини *xy*.

Оглянути поверхню з усіх боків дозволяє функція view. Виклик функції view з двома вихідними аргументами і без вхідних дає можливість визначити поточне положення спостерігача (кути виводяться в градусах):

```
>> [AZ, EL]=view
AZ =
-37.5000
EL =
30
```

Ці значення MatLab використовує за умовчанням при побудові тривимірних графіків. Для завдання положення спостерігача слід вказати азимут і кут піднесення (у градусах) як вхідні аргументи view, наприклад: view(0,90) показує вигляд на графік зверху. Перед поворотом графіка

доцільно розставити позначення до осей, використовуючи, як і для двовимірних графіків xlabel і ylabel, і zlabel для підпису до вертикальної осі. Функція view допускає ще декілька варіантів виклику:

- view(3) — повернення до стандартних установок;
- view([x,y,z]) — розміщення спостерігача в точку із координатами x, y і z.

Освітлена поверхня будується за допомогою функції surfl, яка дозволяє отримати наглядне уявлення про поведінку досліджуваної функції. Слід врахувати, що краще поєднувати виклик surfl з командою shading interp і колірною палітрою, що містить велику кількість відтінків (gray, copper, bone, winter і т. д.), оскільки поверхня володіє властивостями розсіювання, віддзеркалення і поглинання світла, витікаючого від деякого джерела. Положення джерела можна задавати в четвертому додатковому аргументі surfl, причому або вектором з двох елементів (азимут і кут положення джерела), або вектором з трьох елементів (положення джерела світла в системі координат осей), наприклад: surfl(X,Y,Z[20 80]) або surfl(X,Y,Z[6 8 11]).

Розберемо тепер роботу з декількома графіками. Перший виклик будь-якої графічної функції призводити до появи на екрані графічного вікна Figure No. 1, що містить осі з графіком. Проте, при подальших зверненнях до графічних функцій колишній графік пропадає, а новий виводиться в теж саме вікно. Команда figure призначена для створення порожнього графічного вікна. Якщо потрібно отримати декілька графіків в різних вікнах, то перед викликом графічних функцій слід удаватися до figure. Графічні вікна при цьому нумеруються так: Figure No. 2, Figure No. 3 і т. д.

Кожне вікно має свої осі, за наявності декількох пар осей (у одному вікні або в різних) виведення графіків проводиться в поточні осі. Остання створена пара осей є поточною. Для того, щоб вибрати поточні осі з декількох що є, досить клацнути по ним лівою кнопкою миші перед викликом графічної функції. Можлива і зворотна ситуація, коли в процесі роботи потрібно додавати графіки що вже є на деяких осях. У цій ситуації перед додаванням графіка слід виконати команду hold on. Для завершення такого режиму досить скористатися hold off.

У одному графічному вікні можна розташувати декілька осей з своїми графіками. Функція subplot призначена для розбиття вікна на частини і визначення поточною з них. Припустимо, що потрібно вивести графіки на шість пар осей в одне графічне вікно (дві по вертикалі і три по горизонталі). Створіть графічне вікно за допомогою figure і виконайте команду:

```
>> subplot(2,3,1)
```

У лівому верхньому кутку вікна з'явилися осі. Перші два аргументи в subplot вказують на загальне число пар осей по вертикалі і горизонталі, а останній аргумент означає номер даної пари осей. Нумерація йде зліва направо, зверху вниз. Використовуйте subplot(2,3,2) ., subplot(2,3,6) для

створення решти пар осей. Висновок будь-якою з графічних функцій можна направити в потрібні осі, вказавши їх за допомогою subplot(2,3,k), наприклад:

```
>> subplot(2,3,3)
>> bar([1.2 0.3 2.8 0.9])
>> subplot(2,3,6)
>> surf(X,Y,Z)
```

Завдання для самостійної роботи

1. Побудувати графіки функцій однієї змінної на вказаних інтервалах. Вивести графіки різними способами:

- у окремі графічні вікна;
- у одне вікно на одні осі;
- у одне вікно на окремі осі.

Дати заголовки, розмістити підписи до осей, легенду, використовувати різні кольори, стилі ліній і типи маркерів, нанести сітку.

Варіанти

1. $f(x)=\sin x,\quad g(x)=\sin^2 x,\quad x\in[-2\pi,3\pi]$.
   $u(x)=0.01x^2,\quad v(x)=e^{-|x|},\quad x\in[-0.2,9.4]$.
2. $f(x)=\sin x^2,\quad g(x)=\cos x^2,\quad x\in[-\pi,\pi]$.
   $u(x)=x/20,\quad v(x)=e^x,\quad x\in[-2,2]$.
3. $f(x)=x^3+2x^2+1,\quad g(x)=(x-1)^4,\quad x\in[-1,1]$.
   $u(x)=\sqrt{x},\quad v(x)=e^{-x^2},\quad x\in[0,1]$.
4. $f(x)=\ln x,\quad g(x)=x\ln x,\quad x\in[0.2,10]$.
   $u(x)=x^{1/3},\quad v(x)=\sqrt{x},\quad x\in[0,8]$.
5. $f(x)=|2x|^3,\quad g(x)=|2x|^5,\quad x\in[-0.5,0.5]$.
   $u(x)=\sqrt{|x|},\quad v(x)=x^{1/5},\quad x\in[-0.6,0.5]$.
6. $f(x)=x^2,\quad g(x)=x^3,\quad x\in[-1,1]$.
   $u(x)=x^4,\quad v(x)=x^5,\quad x\in[-1,1]$.
7. $f(x)=\arcsin x,\quad g(x)=\arccos x,\quad x\in[-1,1]$.
   $u(x)=\arctan x,\quad v(x)=\arctan 3x,\quad x\in[-1,1]$.
8. $f(x)=sh\ \ x,\quad g(x)=ch\ \ x,\quad x\in[-1,1]$.
   $u(x)=e^x,\quad v(x)=e^{-x},\quad x\in[-0.6,0.6]$.

9. $f(x)=\dfrac{\sin x}{x}$, $g(x)=e^{-x}\cos x$, $x\in[0.01,2\pi]$.

$u(x)=\sin(\ln(x+1))$, $v(x)=\cos(\ln(x+1))$, $x\in[0,2\pi]$.

10. $f(x)=x^x$, $g(x)=x^{x^x}$, $x\in[0.1,1]$.

$u(x)=\dfrac{1}{1+x}$, $v(x)=\dfrac{1}{1+\dfrac{1}{1+x}}$, $x\in[0,1]$.

2. Побудувати графіки шматково-заданої функції, відобразити галузь різними кольорами і маркерами.

1. $f(x)=\begin{cases}-1 & ,-3\le x\le -1\\ x & ,-1<x\le 1\\ e^{1-x} & ,1<x\le 3\end{cases}$ 2. $f(x)=\begin{cases}\sqrt{x} & ,0\le x\le 1\\ 1 & ,1<x\le 3\\ (x-4)^2 & ,3<x\le 5\end{cases}$

3. $f(x)=\begin{cases}\ln x & ,1\le x\le e\\ \dfrac{x}{e} & ,e<x\le 9\\ 9e^{8-x} & ,9<x\le 12\end{cases}$ 4. $f(x)=\begin{cases}\sin x & ,-2\pi\le x\le 0\\ -x^3 & ,0<x\le 1\\ \cos\pi x & ,1<x\le 3\pi\end{cases}$

5. $f(x)=\begin{cases}\arcsin x-1 & ,0\le x\le 1\\ \dfrac{\pi}{2}-x & ,1<x\le\dfrac{\pi}{2}\\ \cos x & ,\dfrac{\pi}{2}<x\le\pi\end{cases}$

6. $f(x)=\begin{cases}|x| & ,-2\le x\le 1\\ \sin\dfrac{\pi}{2}x & ,1<x\le 2\\ (2-x)^3 & ,2<x\le 3\end{cases}$ 7. $f(x)=\begin{cases}(x-1)^2 & ,-2\le x\le 1\\ \cos\dfrac{\pi}{2}x & ,1<x\le 3\\ 1-e^{3-x} & ,3<x\le 8\end{cases}$

8. $f(x)=\begin{cases}e^x & ,-2\le x\le -1\\ \dfrac{|x|}{e} & ,-1<x\le 1\\ e^{-x} & ,1<x\le 2\end{cases}$ 9. $f(x)=\begin{cases}e^{x+1} & ,-2\le x\le 1\\ x^2 & ,-1<x\le 1\\ (2-x)^3 & ,1<x\le 2\end{cases}$

10. $f(x)=\begin{cases} x^2\log_2 x & ,-2\le x\le 1 \\ \dfrac{x^3}{2} & ,-1<x\le 1 \\ \dfrac{x^x}{2} & ,1<x\le 2 \end{cases}$

3. Побудувати графік параметрично заданої функції, використовуючи plot і comet.

Варіанти

1. $x(t)=t-\sin t; y(t)=1-\cos t;$
2. $x(t)=2\sin t-\frac{2}{3}\sin 2t;\ y(t)=2\cos t-\frac{2}{3}\cos 2t;$
3. $x(t)=9\sin\frac{t}{10}-\frac{1}{2}\sin\frac{9}{10}t;\ y(t)=9\cos\frac{t}{10}-\frac{1}{2}\cos\frac{9}{10}t;$
4. $x(t)=\cos t; y(t)=\sin(\sin t);$
5. $x(t)=e^{-t}\cos t; y(t)=\sin(t);$
6. $x(t)=e^{-t}\cos t; y(t)=e^{t}\sin t;$
7. $x(t)=t(t-2\pi); y(t)=\sin t;$
8. $x(t)=\sin t(t-2\pi); y(t)=\sin t;$
9. $x(t)=\sin t(t-2\pi); y(t)=\sin t\cdot\cos t;$
10. $x(t)=\sin t+\cos^3 t; y(t)=\sin t\cdot\cos t;$

4. Віалізувати функцію два змінних на прямокутній області визначення різними способами:

- каркасною поверхнею;
- залитою кольором каркасною поверхнею;
- промаркованими лініями рівня (самостійно вибрати значення функції, що відображеними лініями рівня);
- освітленою поверхнею.

Розташувати графіки в окремих графічних вікнах, і в одному вікні із відповідним числом пар осей. Представити вид каркасної або освітленої поверхні з декількох точок огляду.

Варіанти

1. $z(x,y)=\sin x\cdot e^{-3y}, x\in[0,2\pi], y\in[0,1];$
2. $z(x,y)=\sin^2 x\cdot\ln y, x\in[0,2\pi], y\in[0,10];$

3. $z(x,y) = \sin^2(x-2y)\cdot e^{-|y|}, x \in [0,\pi], y \in [-1,1];$

4. $z(x,y) = \dfrac{x^2y^2 + 2xy - 3}{x^2 + y^2 + 1}, x \in [-2,2], y \in [-1,1];$

5. $z(x,y) = \dfrac{\sin xy}{x}, x \in [0.1,5], y \in [-\pi,\pi];$

6. $z(x,y) = (\sin x^2 + \cos y^2)^{xy}, x \in [-1,1], y \in [-1,1];$

7. $z(x,y) = \arctan(x+y)(\arccos x + \arcsin y), x \in [-1,1], y \in [-1,1];$

8. $z(x,y) = (1+xy)(3-x)(4-y), x \in [0,3], y \in [0,4];$

9. $z(x,y) = e^{-|x|}(x^5 + y^4)\sin(xy), x \in [-2,2], y \in [-3,3];$

10. $z(x,y) = (y^2 - 3)\sin\dfrac{x}{|y|+1}, x \in [-2\pi,2\pi], y \in [-3,3];$

§ 5. Файл-функції і файл-програми

Вбудована мова програмування MatLab достатньо проста, вона містить необхідний мінімум конструкцій, які описані в наступному параграфі. Перш ніж програмувати в MatLab, необхідно зрозуміти, що всі програми можуть бути або файл-функціями, або файл-програмами. Файл-програма являється текстовим файлом із розширенням m (M-файлом), у якому записані команди і оператори MatLab. Розберемо, як створити просту файл-програму.

У MatLab наявний редактор M-файлів, для запуску якого слід натиснути кнопку New M-file на панелі інструментів робочого середовища, або вибрати в меню File в пункті New підпункт M-file. На екрані появиться вікно редактора. Наберіть у ньому які-небудь команди, наприклад для побудови графіка (див. лістинг 5.1):

Лістинг 5.1. Проста файл-програма

```
x=[-1:0.01:1];
y=exp(x);
plot(x,y)
grid on
title('Експоненціальна функція')
```

Для запуску програми або її частини є декілька способів. Перший, найпростіший — виділити оператори з допомогою мишки, утримуючи ліву кнопку, або за допомогою клавіші <Shift> із стрілками, <PageUp>, <PageDown> і вибрати в меню View (у версії 5.3), або в меню Text (у версії 6.x) пункт Evaluate Selection (або натиснути <F9>). Виділені оператори виконуються послідовно, точно так, як і якби вони були набрані в командному рядку. Очевидно, що працювати в M-файлі зручніше, ніж з командного рядка, оскільки можна зберегти програму, добавити опера-

тори, виконувати окремі команди не пробігаючись по історії команд, як у випадку із командного рядка.

Після того, як програма збережена в М-файлі, приміром в myprog.m, для її запуску можна використовувати пункт Run меню Tools (у версії 5.3), або меню Debug (у версії 6.x), або набрати в командному рядку ім'я М-файлу (без розширення) і натиснути <Enter>, тобто виконати, як команду MatLab. При таких способах запуску програми слід враховувати важливу обставину шлях до каталогу із М-файлом повинен бути відомий MatLab. Зробіть каталог з файлом myprog поточним.

- У MatLab 5.3 в меню File робочого середовища перейдіть до пункту Set Path... Появиться діалогове вікно Path Browser (навігатор шляхів). У рядку введення Current Directory встановите необхідний каталог. Скористайтеся кнопкою, розташованою праворуч від рядка введення, для вибору каталогу.
- У MatLab 6.x установки поточного каталогу проводиться із вікна Current Directory робочого середовища. Якщо це вікно відсутнє, то слід вибрати пункт Current Directory меню View робочого середовища. Для вибору бажаного каталогу на диску натисніть кнопку, розташовану праворуч від списку, що розкривається.

Коли поточний каталог встановлений, то всі М-файли, що перебувають в ньому, можуть бути запущені із командного рядка, або з редактора М-файлів. Всі змінні файл-програми після її запуску доступні в робочому середовищі, т. е. являються глобальними. Переконаєтеся в цьому, виконавши команду whos. Більше того, файл-програма може використовувати змінні робочого середовища. Наприклад, якщо була введена команда:

```
>> a=[0.1 0.4 0.3 1.9 3.3];
```

то файл-програма, що містить рядок bar(a), побудує стовпцеву діаграму вектора a (зрозуміло, якщо він не був перевизначений в самій файл-програмі).

Файл-функції відрізняються від файл-програм тим, що вони можуть мати вхідні і вихідні аргументи, а всі змінні, визначені усередині файл-функції, є локальними і не показані в робочому середовищі. М-файл, утримуючий файл-функцію, повинен починатися із заголовка, потім у нього записуються оператори MatLab. Заголовок складається із слова function, списку вихідних аргументів, імені файл-функції і списку вхідних аргументів. Аргументи в списках розділяються комою. Лістинг 5.2 містить зразок простої файл-функції із двома вхідними і одним вихідним аргументами.

Лістинг 5.2. Файл-функція mysum

```
function c=mysum(a,b)
c=a+b;
```

Наберіть цей зразок в новому файлі в редакторові і збережіть його. Звернете увагу, що MatLab пропонує як ім'я М-файлу назву файл-

функції, тобто mysum.m. Завжди зберігайте файл-функцію в М-файлі, ім'я якого збігатися із ім'ям файл-функції! Переконаєтеся, що каталог із файлом mysum.m є поточним і викличте файл-функцію mysum з командного рядка:

```
>> s=mysum(2,3)
s =
5
```

При виклику файл-функції mysum відбулися наступні події:

- вхідний аргумент a набув значення 2;
- вхідний аргумент b став рівний 3;
- сума a і b записалася у вихідний аргумент C;
- значення вихідного аргументу C отримала змінна s робочого середовища і результат вивівся в командне вікно.

Зауважте, що оператор c=a+b у файл-функції mysum завершений крапкою з комою для придушення висновку локальною змінною C в командне вікно. Для поглядання значень локальних змінних при від ладці файл-функції, очевидно, не слід заглушувати висновок на екран значень необхідних змінних.

Практично всі функції MatLab являюся файл-функціями і зберігаються в однойменних М-файлах. Функція sin допускає два варіанти виклику: sin(x) і y=sin(x), в першому випадку результат записується в ans, а в другому — в змінну y. Наша функція mysum поводиться так само. Більше того, вхідними аргументами mysum можуть бути масиви однакових розмірів або масив і число.

Розберемось, тепер, як створити файл-функцію із декількома вихідними аргументами. Список вихідних аргументів в заголовку файл-функції полягає в квадратні дужки, самі аргументи відокремлюються комою. Як приклад на лістингу 5.3 приведена файл-функція quadeq, яка по заданих коефіцієнтах квадратного рівняння знаходить його коріння.

Лістинг 5.3. Файл-функція для вирішення квадратного рівняння

```
function [x1,x2]=quadeq(a,b,c)
D=b^2-4*a*c;
x1=(-b+sqrt(D))/(2*a);
x2=(-b-sqrt(D))/(2*a);
```

При виклику quadeq з командного рядка використовуйте квадратні дужки для вказівки змінних, в які будуть занесені значення коріння:

```
>> [r1,r2]=quadeq(1,3,2)
```

Зауважте, що файл-функцію quadeq можна викликати без вихідних аргументів, або лише із одним вихідним аргументом. В цьому випадку повернеться тільки перший корінь.

Файл-функція може і не мати вхідних або вихідних аргументів, заголовки таких файл-функцій приведені нижче:

function noout(a,b), function [v,u]=noin, function noarg()

Уміння писати власні файл-функції і файл-програми необхідне як при програмуванні в MatLab, так і при рішенні різних задач засобами MatLab (зокрема, пошуку коріння рівнянь, інтеграції, оптимізації). Розберемо тільки один приклад, пов'язаний з побудовою графіка функції $f(x) = e^{-x}(\sin x + 0.1\sin(100\pi x))$ на відрізку [0,1]. Запрограмуйте файл-функцію myfun для обчислення *f*(x). Використовуйте по-елементні операції (див. лістинг 5.4) для того, щоб myfun можна було викликати від вектора значень аргументу і одержати вектор відповідних значень функції.

Лістинг 5.4. Файл-функція myfun

```
function y=myfun(x);
y=exp(-x).*(sin(x) +0.1*sin(100*pi*x));
```

Графік *f(x)* можна одержати двома способами. Перший спосіб полягає в створенні вектора значень аргументу, скажемо з кроком 0.01, заповненні вектор значень функції і виклику plot:

```
>> x=[0:0.01:1];
>> y=myfun(x);
>> plot(x,y)
```

В результаті виходить графік, приведений на мал. 5.1, *а*, який, очевидно, невірний. Дійсно, при обчисленні значень функції на відрізку [0,1] з кроком 0.01 доданок $\sin(100\pi x)$ весь час звертався в нуль і plot побудувала графік не *f(x)*, а іншій функції. Непродуманий вибір кроку часто призводить до втрати істотної інформації про поведінку функції. У MatLab є вбудована функція fplot — деякий аналог plot, але з автоматичним підбором кроку при побудові графіка. Першим вхідним аргументом fplot є ім'я файл-функції, а другим — вектор, елементи якого є межі відрізків: fplot('ім'я файл-функції', [a,b]). Побудуйте тепер в новому вікні графік *f(x)* за допомогою fplot:

```
>> figure
>> fplot('myfun'[0,1])
```

Вийшов графік, що точно відображає поведінку функції (рис. Д5.1, *б*).

а) б)

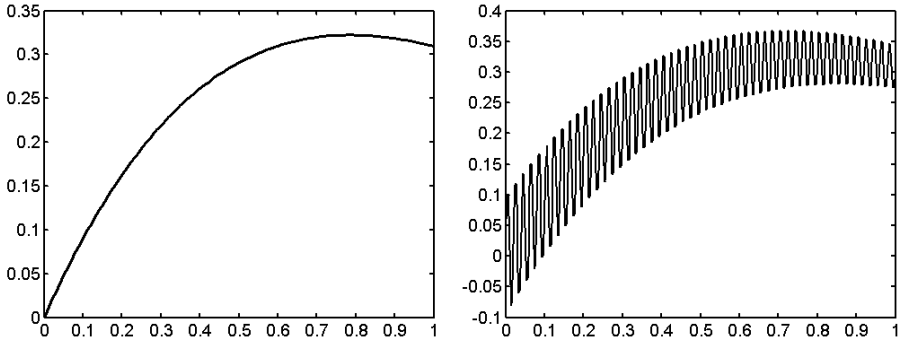

рис. Д 5.1 Графіки функцій

Завдання для самостійної роботи

1. Написати файл-функції і побудувати графіки на заданому відрізку за допомогою plot (з кроком 0.05) і fplot для наступних функцій:

Варіанти

1. $f(x)=\sin\frac{1}{x}, x\in[0.05,1];$ 2. $f(x)=e^{3x\sin 5\pi x}+e^{3x\cos\pi x}, x\in[0,1]$

3. $f(x)=\frac{10}{11-10\sin 21\pi x}, x\in[0.05,1]$ 4. $f(x)=\sqrt{\frac{|\sin 21\pi x|}{2+\sin 20\pi x}}, x\in[0,1]$

5. $f(x)=\frac{1}{\arctan\left(\frac{1}{1.1+\sin 5\pi x}\right)-\frac{3}{2}}, x\in[0,1];$ 6. $f(x)=\cos\left(\frac{1}{\frac{2\pi}{11}-\arctan x^x}\right), x\in[0,1];$

7. $f(x)=\sin\left(6\pi\left|x-\frac{2}{3}\right|x^3\right), x\in[0,1];$ 8. $f(x)=\sqrt{\left|\sqrt{1-x^3}-\frac{4}{7}\right|}, x\in[0,1];$

9. $f(x)=|\sin 20\pi x|, x\in[0.05,1];$ 10. $f(x)=\frac{1}{\sin(e^{2x}-e^{-2x})+\cos(e^{2x}-e^{-2x})-\frac{3}{2}}, x\in[-1,1];$

2. Написати файл-функцію для вирішення поставленого завдання.

Варіанти

1. Написати файл-функцію, яка по заданому вектору визначає номер його елементу з найбільшим відхиленням від середнього арифметичного всіх елементів вектора.

2. Написати файл-функцію, що повертає суму всіх елементів вектора з непарними індексами.

3. Написати файл-функцію, що обчислює максимальне значення серед діагональних елементів заданої матриці.

4. Написати файл-функцію, що переставляє перший стовпець квадратної матриці з її діагоналлю.

5. Написати файл-функцію, яка підсумовує всі не діагональні елементи заданої матриці.

6. Написати файл-функцію, замінюючи максимальний елемент вектора середнім значенням всіх його елементів.

7. Написати файл-функцію, замінюючи елемент матриці з індексами 1,1 добутком всіх елементів матриці.

8. Написати файл-функцію, яка будує багатокутник (замкнутий) по заданих векторах *x* і *y* з координатами вершин.

9. Написати файл-функцію, яка відображає елементи заданого вектора синіми маркерами, а максимальний елемент — червоним і повертає значення і номер максимального елементу.

10.Написати файл-функцію, що переводить час в секундах в години, хвилини і секунди.

§ 6. Програмування

Мова програмування MatLab достатньо проста, вона містить основний набір конструкцій: оператори галуження і цикли. Простота мови програмування окупається величезною кількістю вбудованих функцій, які дозволяють вирішувати задачі із різних областей.

Цикл for використовується для повторення операторів у разі, коли число повторень заздалегідь відоме. У циклі for використовується лічильник циклу, його початкове значення, крок і кінцеве значення указуються через двокрапку. Блок операторів, розміщений усередині циклу, повинен закінчуватися словом end. Лістинг 6.1 містить файл-програму для виведення графіків функції $f(x,\beta)=e^{\beta x}\sin x$ на відрізку, для значень параметра $\beta \in [-0.5,0.5]$.

Лістинг 6.1. Графіки функції при різних значеннях параметра

```
x=[-2:0.01:2];
for beta=-0.5:0.1:0.5
y=exp(beta*x).*sin(x);
plot(x,y)
```

```
hold on
end
hold off
```

Якщо крок рівний одиниці, то його вказувати не обов'язково. Наприклад, для підрахунку суми

$$\sum_{k=1}^{10} \frac{x^k}{k!}$$

при різних значеннях *x* буде потрібно файл-функцію, текст якої приведений на лістингу 6.2. Зверніть увагу, що sum10 може бути викликана як від числа, так і від масиву значень, завдяки застосуванню по-елементних операцій.

Лістинг 6.2. Файл-функція для підрахунок суми

```
function s=sum10(x)
s=0;
for k=1:10
s=s+x.^k/factorial(k);
end
```

Подумайте над тим, як уникнути знаходження факторіалу в кожному доданку (при підрахунок *к-го* доданку можна використовувати значення (k-1)! , знайдене на попередньому кроці циклу).

Цикл for підходить для повторення заданого числа певних дій. У тому випадку, коли число повторів заздалегідь невідоме і визначається в ході виконання блоку операторів слід організувати цикл while. Цикл while працює, поки(доки) виконана умова циклу. Файл-функція negsum (див. лістинг 6.3) знаходить суму всіх перших негативних елементів вектора.

Лістинг 6.3. Файл-функція negsum

```
function s=negsum(x)
s=0;
k=1;
while x(k) <0
s=s+x(k);
k=k+1;
end
```

Як оператори відношення використовуються символи: >, < , >=, <= == (рівно), ~= (не рівно). Файл-функцію negsum має один недолік: якщо всі елементи масиву — негативні числа, то k стає більше довжини масиву *x*, що приводить до помилки, наприклад:

```
” b=[-2 -7 -1 -9 -2 -5 -4];
” s=negsum(b)
??? Index exceeds matrix dimensions.
```

Крім перевірки значення *x*(*k*) слід потурбуватися про те, щоб значення k не перевершувало довжини вектора x. Вхід в цикл повинен здійснюватися лише при одночасному виконанні умов k<=length(x) і x(k)<0,

тобто необхідно застосувати логічного оператора "и", що позначається в MatLab символом &. Замінити умову циклу на складене: k<=length(x) & x(k) <0. Якщо перше з умов не виконується, то друга умова перевірятися не буде, саме тому вибраний такий порядок операндів. Тепер файл-функція negsum працює вірно для будь-яких векторів.

Логічний оператор "або" позначається символом вертикальної межі |, а заперечення — за допомогою тильди ~. Нижче приведені логічні операції у міру убування їх пріоритету:

- заперечення ~ ;
- оператори відношення >, < , >=, <= ==, ~= ;
- логічне "и" & ;
- логічне "або" | .

Для зміни порядку виконання логічних операторів використовуються круглі дужки.

Цикли можуть бути вкладені один в одного. Наприклад, для пошуку суми елементів матриці, розташованих вище за головну діагональ, слід використовувати два цикли for, причому початкове значення лічильника внутрішнього циклу залежить від поточного значення лічильника зовнішнього циклу (див. лістинг 6.4).

Лістинг 6.4. Використання вкладених циклів

```
function s=upsum(A)
[n m]=size(A);
s=0;
for i=1:n
for j=i+1:m
s=s+A(i,j);
end
end
```

Галуження в ході роботи програми здійснюється за допомогою конструкції if-elseif-else. Найпростіший варіант її використання (без elseif і else) реалізований у файл-функції possum (див. лістинг 6.5), яка призначена для знаходження суми всіх позитивних елементів вектора.

Лістинг 6.5. Файл-функція для підсумовування позитивних елементів вектора

```
function s=possum(x)
s=0;
for k=1:length(x)
if x(k) >0
s=s+x(k);
end
end
```

Якщо хід програми повинен змінюватися залежно від декількох умов, то слід використовувати повну конструкцію if-elseif-else. Кожна з гілок elseif в цьому випадку повинна містити умову виконання блоку опе-

раторів, розміщених після неї. Важливо розуміти, що умови перевіряються підряд, перша виконана умова приводить до роботи відповідного блоку, виходу з конструкції if-elseif-else і переходу до оператора, наступного за end. В останньої гілки else не повинне бути ніякої умови. Оператори, що знаходяться між else і end, працюють в тому випадку, якщо всі умови виявилися невиконаними. Припустимо, що потрібно написати файл-функцію для підрахунок шматково-заданої функції:

$$f(x)=\begin{cases}1-e^{-1-x} & ,x<-1\\ x^2-x-2 & ,-1\le x\le 2\\ 2-x & ,x>2\end{cases}$$

Перша умова x<-1 перевіряється в гілці if. Звернете увагу, що умова –1<=x не потрібно включати в наступну гілку elseif (див. лістинг 6.6), оскільки в цю гілку програма заходить, якщо попередня умова (x<-1) опинилася не виконано. Умову x>2 перевіряти не треба — якщо не виконано дві попередні умови, то x буде більше два.

Лістинг 6.6. Файл-функція для підрахунок шматково-заданої функції

```
function f=pwf(x)
if x<-1
f=1-exp(-1-x);
elseif x<=2
f=x^2-x-2;
else
f=2-x;
end
```

Хід роботи програми може визначатися значенням деякої змінної (перемикача). Такий альтернативний спосіб галуження програми заснований на використанні оператора перемикання switch. Змінна-перемикач поміщається після switch через пропуск. Оператор switch містить блоки, що починаються із слова case, після кожного case записується через пропуск те значення перемикача, при якому виконується даний блок. Останній блок починається із слова otherwise, його оператори працюють у тому випадку, коли жоден з блоків case не був виконаний. Якщо хоча би один з блоків case виконаний, то відбувається вихід із оператора switch і перехід до оператора, наступного за end.

Припустимо, що потрібно знайти кількість одиниць і мінус одиниць в заданому масиві і, крім того, знайти суму всіх елементів, відмінних від одиниці і мінус одиниці. Слід перебрати всі елементи масиву в циклі, причому в ролі змінної-перемикача буде виступає поточний елемент масиву. Лістинг 6.7 містить файл-функцію, яка по заданому масиву повертає число мінус одиниць в першому вихідному аргументі, число одиниць — в другому, а суму — в третьому.

Лістинг 6.7. Файл-функція mpsum

```
function [m,p,s]=mpsum(x)
m=0;
p=0;
s=0;
for i=1:length(x)
switch x(i)
case -1
m=m+1;
case 1
p=p+1;
otherwise
s=s+x(i);
end
end
```

Блок case може бути виконаний не лише при одному певному значенні перемикача, але і у тому випадку, коли перемикач приймає одне з декількох допустимих значень. В цьому випадку значення вказуються після слова case у фігурних дужках через кому, наприклад: case{1,2,3}.

Дострокове завершення циклу while або for здійснюється за допомогою оператора break. Нехай, наприклад, потрібно по заданому масиву x утворити новий масив за правилом y(k)=x(k+1)/x(k) до першого нульового елементу x(k), тобто до тих пір, поки має сенс операція ділення. Номер першого нульового елементу в масиві x заздалегідь невідомий, більш того, в масиві x може і не бути нулів. Рішення задачі полягає в послідовному обчисленні елементів масиву y і припиненні обчислень при виявленні нульового елементу в x. Файл-функція, приведена на лістингу 6.8, демонструє роботу оператора break.

Лістинг 6.8. Використання оператора break для виходу із циклу

```
function y=div(x)
for k=1:length(x)-1
if x(k)==0
break
end
Y(k)=x(k+1)/x(k);
end
```

Завдання для самостійної роботи

1. Написати файл-функцію для підрахунків шматково-заданої функції. Написати файл-функцію для вирішення поставленого завдання.

Варіанти

1. Обчислити твір елементів вектора, що не перевершують середнє арифметичне значення його елементів.
2. Підрахувати число нулів і одиниць в заданій матриці.

3. Визначити кількість позитивних елементів вектора, розташованих між його максимальним і мінімальним елементами.

4. Підсумувати негативні елементи матриці, лежачі нижче за головну діагональ.

5. Замінити позитивні елементи вектора сумою всіх його негативних елементів.

6. Заповнити квадратну матрицю *A*, кожен елемент якої визначається таким чином:

$$a_{ij} = \begin{cases} i - j & , \ i > j \\ i + j & , \ i = j \\ i^2 + j^2 & , \ i < j \end{cases}$$

7. Вичислити суму: $s(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{x^{i+j}}{(i+j)^2}$

8. Для матриці розміру *n* на *m* знайти значення виразу $\omega(x) = \sum_{i=1}^{n} \prod_{j=1}^{m} a_{ij}$

9. По заданому *x* знайти максимальне значення *n*, для якого наступна сума не перевершує 100: $s(x) = \sum_{k=1}^{n} kx^k$

10.Обчислити суму $s(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$

із заданою точністю ξ. Підсумовувати слід поки модуль відношення поточного доданку до вже накопиченої частини суми перевершує ξ. Порівняти результат з точним значенням, побудувавши графіки e^x і *s(x)* для $x \in [0,5]$

11.Задані кола, координати їх центрів містяться в масивах x і y, а радіуси в масиві r. Відомі координати деякої точки. Потрібно вивести графік, на якому маркером відмічено положення крапки , синім кольором зображені ті кола, усередині яких лежить крапка, а решта кіл намальована червоним кольором.

§ 7. Робота з рядками

Основним типом даних в MatLab є масив, пригадаєте, що навіть числа представляються масивами розміру один на один. Елементи масиву не обов'язково повинні бути числами. Якщо елементи масиву є символами, то такий масив називається масивом символів, або рядком. Змінні, що містять рядки, називатимемо рядковими змінними. Для обмеження рядка використовуються апострофи, наприклад, оператор привласнення

```
>> str='Hello, World!'
```

приводить до утворення рядкової змінної

```
str =
Hello, World!
```

Переконаєтеся, що рядкова змінна str дійсно являється масивом, звернувшись до її елементів: str(8), str(1:5). Команда whos дозволяє отримати докладну інформацію про str, зокрема, str зберігається у вигляді вектор-рядки, її довжина рівна 13. Оскільки рядкові змінні являється масивами, то до них застосовні деякі функції і операції, розглянуті нами раніше. Довжина рядкової змінної, тобто число символів в ній, знаходиться за допомогою функції length. Допустиме зчеплення рядків як вектор-рядків з використанням квадратних дужок. Створіть ще одну рядкову змінну str1, що містить текст ' My name is Igor.', і здійсните зчеплення:

```
>> strnew=[str str1]
strnew =
Hello, World! My name is Igor.
```

Звернете увагу, що для розділення злитих рядків в квадратних дужках слід використовувати пропуск (або кому). Вживання крапки з комою приведе до помилки, оскільки крапка з комою служить для утворення вектора-стовпців або відділення рядків матриць. Але в матриці всі рядки повинні бути однакової довжини, а змінні str і str1 складаються з різного числа символів. Далі розглянемо, як створювати двовимірні масиви символів (масиви рядків), а поки звернемося до найбільш важливих функцій MatLab, призначених для обробки рядків.

Зчеплення рядків може бути проведене як з використанням квадратних дужок, так і за допомогою функції strcat. Вхідними аргументами strcat є злиті рядки, їх число необмежене, а результат повертається у вихідному аргументі. Функція strcat ігнорує пропуски в кінці кожного рядка. Для рядків: s1='abc ' і s2='def' зчеплення s=[s1 s2] і s=strcat(s1,s2) приведуть до різних результатів.

Пошук позицій входження під-рядка в рядок проводиться за допомогою функції findstr. Її вхідними аргументами є рядок і під-рядок, а вихідним вектор позицій, починаючи з яких під-рядок входить в рядок, наприклад:

```
>> s='abcbcddefbcc';
>> s1='bc';
>> p=findstr(s,s1)
p =
2 4 10
```

Порядок вхідних аргументів не важливий, під-рядком завжди вважається аргумент меншого розміру.

Функція strcmp призначена для порівняння двох рядків, які указуються у вхідних аргументах: strcmp(s1,s2). Результатом є логічна одиниця або нуль. Функція strncmp порівнює лише частину рядків від першого

символу до символу, номер якого вказаний в третьому вхідному аргументі: strncmp(s1,s2,8). Дані функції мають аналоги: strcmpi і strncmpi, які проводять порівняння без урахування регістра, тобто, наприклад, символи 'A' і 'a' вважаються однаковими.

Для заміни в рядку одного під-рядка на іншу служить функція strrep. Нехай, наприклад, потрібно замінити в рядку str під-рядок s1 на s2 і записати оновлений рядок в strnew. Тоді виклик функції strrep повинен виглядати так: strnew=strrep(str,s1,s2). Тут важливий порядок аргументів.

Перетворення всіх букв рядка в рядкові (прописні) проводить функція lower (upper), наприклад: lower('MatLab') приводить до 'matlab', а upper('MatLab') до 'MATLAB'.

Те, що рядки є масивами символів, дозволяє легко писати власні файл-функції обробки рядків. Припустимо, що потрібно переставити символи в рядку в зворотному порядку. Файл-функція, приведена на лістингу 7.1, вирішує поставлену задачу.

Лістинг 7.1. Файл-функція для перестановки символів в рядку

```
function sout=strinv(s)
L=length(s);
for k=1:L
sout(L-k+1)=s(k);
end
```

Перейдемо тепер до вивчення масивів рядків. Можна вважати, що масив рядків є вектор-стовбець, кожен елемент якого є рядок, причому довжини всіх рядків однакові. В результаті виходить прямокутна матриця, що складається з символів. Наприклад, для змінних s1='March', s2='April', s3='May', операція S=[s1; s2] є допустимою, і приводить до масиву рядків:

```
S =
March
April
```

Аналогічне об'єднання трьох рядків S=[s1; s2; s3] викличе виведення повідомлення про помилку. Можна, звичайно, доповнити за допомогою зчеплення, кожен рядок пропусками справа до довжини найбільшою з рядків і потім проводити формування масиву. Функція char якраз і вирішує цю задачу, створюючи з рядків різної довжини масив рядків:

```
>> S=char(s1,s2,s3)
S =
March
April
May
```

Перевірте за допомогою команди whos, що S є масивом розміру 3 на 5. Для визначення розмірів масиву рядків, так само, як і будь-яких масивів, використовується size.

Звернення до елементів масиву рядків проводиться аналогічно зверненню до елементів числового масиву за допомогою індексування числами або двокрапкою, наприклад: S(3,2), S(2:), S(2:3,1:4). При виділенні рядка із масиву рядків в кінці можуть залишитися пропуски. Якщо вони не потрібні, то їх слід видалити, скориставшись функцією deblank. Порівняєте, до чого приводять S(3:) і deblank(S(3:)).

Пошук під-рядка в масиві рядків виконується функцією strmatch. Вхідними аргументами strmatch являється під-рядок і масив рядків, а вихідним — вектор з номерами рядків, що містять під-рядок:

```
>> p=strmatch('Ma',S)
p =
1
3
```

Якщо потрібно шукати номери рядків, точно співпадаючих з під-рядком, то слід задати третій додатковий вхідний аргумент 'exact'.

Деякі функції обробки рядків можуть працювати і з масивами рядків. Наприклад, при зчепленні масивів з однаковим числом рядків за допомогою strcat, зчеплення застосовується до відповідних рядків масивів, утворюючи новий масив рядків. Серед вхідних аргументів strcat може бути і рядок. В цьому випадку вона добавляється до всіх рядків масиву.

Серед символів рядки можуть бути цифри і точки, тобто рядок може містити числа. Важливо розуміти, що оператор привласнення b=10 приводить до утворення числової змінної b, а оператор s='10' — рядковою. Перетворення цілого числа в рядок проводиться за допомогою функції int2str, вхідним аргументом якої є число, а вихідним рядок, наприклад:

```
>> str=['May ' int2str(10)]
str =
May, 10
```

Зауважте, що нецілі числа перед перетворенням округляються. Для перекладу нецілих чисел в рядки служить функція num2str:

```
>> str=num2str(pi)
str =
3.1416
```

Додатковий другий вхідний аргумент num2str призначений для вказівки кількості цифр в рядку з результатом: str=num2str(pi,11). Можливе гнучкіше управління перетворенням за допомогою рядка спеціального вигляду, що визначає формат (експоненціальний або з плаваючою крапкою) і кількість позицій, що відводяться під число. Рядок з форматом задається як другий вхідний аргумент num2str, починається із знаку відсотка і має вид '%A.ax', де:

- A — кількість позицій, що відводиться під все число;
- a — кількість цифр після десяткової крапки;

- x — формат висновку, який може приймати, наприклад, одне з наступних значень: f (з плаваючою крапкою), e (експоненціальний) або g (автоматичний підбір якнайкращого представлення).

Порівняєте результати наступних перетворень числа в рядок:

```
>> z=198.23981
>> sf=num2str(z, '%12.5f')
>> se=num2str(z, '%12.5e')
>> sg=num2str(z, '%12.5g')
```

Якщо вхідним аргументом функцій int2str і num2str є матриця, то результатом буде масив рядків. В цьому випадку в рядку з форматом часто корисно вказати роздільник, наприклад кому і пропуск:

```
>> R=rand(5)
>> S=num2str(R,'%7.2e ')
```

Зворотне завдання, а саме, перетворення рядкової змінної, що містить число, в числову змінну вирішується за допомогою функції str2num. Вхідним аргументом str2num може бути рядок з представленням цілого, речовинного або комплексного числа відповідно до правил MatLab, наприклад: str2num('2.9e-3'), str2num('0.1'), str2num('4.6+4i').

Завдання для самостійної роботи

1. Написати файл-функцію для вирішення поставленого завдання.

Варіанти

1. Підрахувати число входжень під-рядка в рядок.
2. Знайти кількість пропусків в рядку.
3. Визначити кількість цифр в рядку.
4. Видалити що йдуть підряд однакові символи в рядку.
5. Замінити що йдуть підряд однакові символи в рядку на один.
6. Рядок є пропозицією, в якій слова розділені пропусками. Переставити перше і останнє слово.
7. Утворити рядок, що складається з перших букв рядків, що входять в масив рядків.
8. Вивести номери однакових рядків в масиві рядків.
9. Визначити кількість символів в кожному рядку масиву рядків без урахування пропусків.
10. По заданому масиву рядків утворити новий, виключивши рядки, що повторюються.
11. Замінити в рядку цифри числівниками (замість 1, 2. — один, два, три.).
12. Заданий рядок, що містить текст і числа, виділити числа в числовий масив.

§ 9. Текстові файли

MatLab пропонує достатньо універсальні способи прочитування даних з текстових файлів і запису даних в необхідному вигляді в текстовий файл. Розглянемо спочатку операцію з числовими даними, представленими матрицями і векторами. Нехай в текстовому файлі vec.dat у стовпчик записані числа відповідно до правил MatLab, тобто для відділення десяткових знаків використовується крапка, а для запису числа в експоненціальному вигляді застосовується символ e. Функція load дозволяє занести вміст файлу vec.dat у числовий масив. Ім'я файлу, поміщене в апострофи, вказується у вхідному аргументі load, а у вихідному аргументі повертається вектор-стовпець:

```
>> v=load('vec.dat');
```

Зворотна операція запис значень вектор-стовпця у файл проводиться за допомогою команди save, аргументами якої є: ім'я файлу, змінна, і додатковий параметр –ascii, що означає запис в текстовому вигляді:

```
>> v2=v.^2;
>> save 'vec2.dat' v2 –ascii
```

Якщо числа в текстовому файлі розташовані в рядок і розділені пропусками, то результатом прочитування буде вектор-рядок. Вірно і зворотне, запис значень вектор-рядка приведе до занесення їх в рядок в текстовому файлі.

Запис матриці командою save приводить до утворення текстового файлу з таблицею чисел. Текстовий файл, в кожному рядку якого знаходиться однакова кількість чисел, відокремлених один від одного пропусками, прочитується в двовимірний масив функцією load.

Команди load і save призначені тільки для простого введення-висновку числових даних. Розберемо тепер складніший випадок, коли прочитуванна або записувана інформація містить не тільки числа, але і текст. Робота з текстовими файлами складається з трьох етапів:

1. відкриття файлу;
2. прочитування або запис даних;
3. закриття файлу.

Для відкриття файлу служить функція fopen, яка викликається з двома вхідними аргументами: ім'ям файлу і рядком, заданий спосіб доступу до файлу. Вихідним аргументом fopen є ідентифікатор файлу, тобто змінна, яка згодом використовується при будь-якому зверненні до файлу. Функція fopen повертає –1, якщо при відкритті файлу виникла помилка. Існує чотири основні способи відкриття файлу:

1. f=fopen('myfile.dat','rt') відкриття текстового файлу myfile.dat тільки для читання з нього;
2. f=fopen('myfile.dat','rt+') відкриття текстового файлу myfile.dat для читання і запису даних;
3. f=fopen('myfile.dat','wt') створення порожнього текстового файлу myfile.dat тільки для запису даних;

4. f=fopen('myfile.dat','wt+') створення порожнього текстового файлу myfile.dat для запису і читання даних.

При використанні двох останніх варіантів слід дотримуватися обережності якщо файл myfile.dat вже існує, то його вміст буде знищений. Після відкриття файлу з'являється можливість прочитувати з нього інформацію, або заносити її у файл. Після закінчення роботи з файлом необхідно закрити його за допомогою fclose(f).

Відрядкове прочитування інформації із текстового файлу проводиться за допомогою функції fgetl. Вхідним аргументом fgetl є ідентифікатор файлу, а вихідним поточний рядок. Кожен виклик fgetl приводить до прочитування одного рядка і перекладу поточної позиції у файлі на початок наступного рядка. Команди, приведені на лістингу 9.1, послідовно прочитують з файлу myfile.dat перші три рядки в рядкові змінні str1, str2, str3.

Лістинг 9.1. Прочитування трьох перших рядків із текстового файлу

```
f=fopen('myfile.dat','rt');
str1=fgetl(f);
str2=fgetl(f);
str3=fgetl(f);
fclose(f);
```

При послідовному прочитуванні рано чи пізно буде досягнутий кінець файлу, при цьому функція fgetl поверне мінус 1. Краще всього перед прочитуванням перевіряти, чи не є поточна позиція у файлі останньої. Для цього призначена функція feof, яка викликається від ідентифікатора файлу і повертає одиницю, якщо поточна позиція остання і нуль, інакше. Звичайне послідовне прочитування організовується за допомогою циклу while. Лістинг 9.2 містить файл-функцію viewfile, яка прочитує рядки файлу, і виводить їх в командне вікно.

Лістинг 9.2. Файл-функція viewfile

```
function viewfile(fname)
f=fopen(fname,'rt');
while feof(f)==0
s=fgetl(f);
disp(s)
end
fclose(f);
```

Виклик viewfile('viewfile.m') приводить до відображення тексту самої файл-функції в командному вікні.

Рядки записуються в текстовий файл за допомогою функції fprintf, її першим вхідним аргументом є ідентифікатор файлу, а другим рядок, що додається. Символ \n служить для перекладу рядка. Якщо помістити його в кінець рядка, що додається, то наступна команда fprintf здійснюватиме висновок у файл з нового рядка, а якщо \n перебувати на почат-

ку, то поточна команда fprintf виведе текст з нового рядка. Наприклад, послідовність команд (лістинг 9.3) приведе до появи у файлі my.txt тексту з двох рядків (лістинг 9.4).

Лістинг 9.3. Виведення рядків в текстовий файл

```
f=fopen('my.txt','wt');
fprintf(f,'текст ');
fprintf(f,'еще текст\n');
fprintf(f,'а цей текст з нового рядка');
fclose(f);
```

Лістинг 9.4. Результат роботи операторів лістингу 9.3

```
текст ще текст
а цей текст з нового рядка
```

Аналогічного результату можна добитися, перемістивши \n з кінця рядка другої функції fprintf в початок рядка третьої функції fprintf. Аргументом fprintf може бути не тільки рядок, але і рядкова змінна. В цьому випадку для забезпечення висновку з нового рядка її слід зчепити з рядком '\n'.

Рядок, призначений для висновку в текстовий файл, може містити як текст, так і числа. Часто потрібно виділити певну кількість позицій під число і вивести його в експоненціальному вигляді або з плаваючою крапкою і із заданою кількістю цифр після десяткової крапки. Тут не обійтися без висновку формату за допомогою fprintf, звернення до якої має вигляд:

fprintf(ідентифікатор файлу, 'формати', список змінних)

У списку змінних можуть бути як числові змінні (або числа), так і рядкові змінні (або рядки). Другий аргумент є рядком спеціального вигляду з форматами, в яких виводитимуться всі елементи із списку. Кожен формат починається із знаку відсотка. Для виведення рядків використовується формат s, а для виведення чисел f (з плаваючою крапкою) або e (експоненціальний). Число перед s указує кількість позицій, що відводяться під виведення рядка. При виведенні значень числових змінних перед f або e ставиться два числа, розділених крапкою. Перше з них означає кількість позицій, що виділяються під все значення змінної, а друга кількість знаків після десяткової крапки. Таким чином, під кожний з елементів списку відводиться поле певної довжини, вирівнювання в якому за умовчанням проводиться по правому краю. Для вирівнювання по лівому краю слід після знаку відсотка поставити знак мінус. Нижче приведені варіанти виклику fprintf і результати, незаповнені позиції після висновку (пропуски) позначені символом 0.

fprintf(f,'%6.2f', pi) 003.14
fprintf(f,'%-6.2f', pi) **3.1400**
fprintf(f,'%14.4e', exp(-5)) 0006.7379e-003
fprintf(f,'%-14.4e', exp(-5)) **6.7379e-003000**
fprintf(f,'%8s', 'текст') 000текст

fprintf(f,'%-8s', 'текст') **текст000**

У разі, коли зарезервоване поле не вистачає під рядок, що виводиться, або число, MatLab автоматично збільшує довжину поля, наприклад:

fprintf(f,'%5.4e', exp(-5)) **6.7379e-003**

При одночасному виведенні чисел і тексту пропуски можуть з'являтися із-за неповністю заповнених полів, виділених як під числа, так і під рядки. Приведені нижче оператори і результат їх виконання демонструють розташування полий в рядку текстового файлу. Хвилястою лінією підкреслені поля, виділені під числа, а прямій під рядки, символ 0 як і раніше позначає пропуски.

```
x=0.55;
fprintf(f,'%-5s%6.2f%6s%20.8e','x=',x,'y=',exp(x))
```

x=○○○○○0.55○○○○y=○○○○○1.73325302e+000

Формальний вивід при формуванні файлу з таблицею результатів. Припустимо, що необхідно записати у файл f.dat таблицю значень функції $f(x) = x^2 \sin x$ для заданого числа *n* значень $x \in [a,b]$, встоявши один від одного на однакову відстань. Файл з таблицею значень повинен мати таку структуру, як показано на лістингу 9.5.

Лістинг 9.5. Текстовий файл з таблицею значень функції

```
f(0.65) =2.55691256e-001
f(0.75) =3.83421803e-001
f(0.85) =5.42800093e-001
f(0.95) =7.34107493e-001
f(1.05) =9.56334106e-001
```

Очевидно, що слід організувати цикл від початкового значення аргументу до кінцевого з кроком, відповідним заданому числу крапок, а усередині циклу викликати fprintf з відповідним списком висновку і форматами. Символ \n, призначений для перекладу рядка, поміщається в кінець рядка з форматами (див. лістинг 9.6) .

Лістинг 9.6. Файл-функція tab, що виводить таблицю значень функції

```
function tab(a,b,n)
h=(b-a)/(n-1);
f=fopen('f.dat','wt');
for x=a:h:b
fprintf(f,'%2s%4.2f%2s%15.8e\n', 'f(',x,')=',x^2*sin(x))
end
fclose(f);
```

Звернемося тепер до прочитування даних з текстового файлу. Функція fscanf здійснює зворотну дію по відношенню до fprintf. Кожен виклик fscanf приводить до занесення даних, що починаються з поточної пози-

ції, в змінну. Тип змінної визначається заданим форматом. У загальному випадку, звернення до fscanf має вигляд:

a=fscanf(ідентифікатор файлу, 'формат', число прочитуваних елементів)

Для прочитування рядків використовується формат %s, для цілих чисел %d, а для речовинних %g.

Припустимо, що файл exper.dat містить текст, приведений на лістингу 9.7. Дані відокремлені один від одного пропусками. Потрібно рахувати дату проведення експерименту (число, місяць і рік) у відповідні змінні, а результати занести в числові масиви TIME і DAT.

Лістинг 9.7. Файл з даними exper.dat

```
Результати експериментів 10 травня 2002
t= 0.1 0.2 0.3 0.4 0.5
G= 3.02 3.05 2.99 2.84 3.11
```

Прочитування різнорідних даних (числа, рядки, масиви) вимагає контролю, який досягається застосуванням форматів. Перші два елементи файлу exper.dat є рядками, отже можна відразу занести їх в одну рядкову змінну head. Очевидно, треба вказати формат %s і число 2 прочитуваних елементів. Далі потрібно рахувати ціле число 10, рядок 'травня' і знову ціле число 2002. Звернете увагу, що перед заповненням масиву TIME ще доведеться передбачити прочитування рядка 't='. Тепер все готово для занесення п'яти дійсних чисел у вектор-стовпець TIME за допомогою формату %g. Дані з останнього рядка файлу exper.dat витягуються аналогічним чином (лістинг 9.8).

Лістинг 9.8. Застосування fscanf для прочитування даних

```
f=fopen('exper.dat','rt');
head=fscanf(f,'%s',2)
data=fscanf(f,'%d',1)
month=fscanf(f,'%s',1)
year=fscanf(f,'%d',1)
str1=fscanf(f,'%s',1)
TIME=fscanf(f,'%g',5)
str2=fscanf(f,'%s',1)
DAT=fscanf(f,'%g',5)
fclose(f);
```

Інформація в текстовому файлі може бути представлена у вигляді таблиці. Для прочитування такої інформації слід використовувати масив структур з відповідним набором полий. Прочитування всієї інформації реалізується в циклі while, в умові якого проводиться перевірка на досягнення кінця файлу за допомогою функції feof.

Функція fscanf надає можливість прочитувати числа з текстового файлу не тільки у вектор-стовпець, але і масив заданих розмірів. Розташування чисел по рядках у файлі не має значення. Розміри формовано-

го масиву указуються у векторі в третьому вхідному аргументі fscanf. Розглянемо прочитування числових даних з файлу matr.txt (лістинг 9.9).

Лістинг 9.9. Текстовий файл matr.txt із матрицею

```
1.1 2.2 3.3 4.4
5.5 6.6 7.7 8.8
0.2 0.4 0.6 0.8
```

Функція fscanf формує стовпець матриці, послідовно прочитуючи числа з| файлу (т. е. відрядковий). Отже, для заповнення матриці з трьома рядками і чотирма стовпцями, необхідно рахувати дані з файлу в масив чотири на три, а потім транспонувати його (лістинг 9.10).

Лістинг 9.10. Прочитування матриці із текстового файлу

```
f=fopen('matr.txt');
M=fscanf(f,'%g' [4 3]);
M=M'
fclose(f);
```

Звернете увагу, що масив може мати розміри не тільки 3 на 4. Оскільки дані прочитуються підряд, то вони можуть бути занесені і в масив 2 на 6, і 4 на 3 і т.д.

Завдання для самостійної роботи

1. Написати файл-функцію для прочитання даних з файлу в структуру або масив структур з відповідними полями.

Варіанти

1. Алексєєв Сергій 1980 5 4 4 5 3 5
Іванов Костянтин 1981 3 4 3 4 3 5
Петров Олег 1980 5 5 5 4 4 5
2. 21 березня 2002 0.56 0.58 0.49 0.44
23 березня 2002 0.36 0.32 0.28 0.25
25 березня 2002 1.62 1.68 1.71 1.91
3. 195251 Спб Політехнічна 29
195256 Спб Науки 49
195256 Спб Науки 24
4. Результати спостережень
Time= 0.0 0.1 0.2 0.3 0.4 0.5 0.6
Mass=
2.1 2.3 2.3 1.9 1.8 2.4 2.9
0.8 0.7 0.5 1.1 3.2 0.3 0.4
5. Алексєєв Іван 121-22-04
Сидоров Микола 101-21-99
Тимофіїв Сергій 570-00-03

(номери телефонів повинні бути записані в поля структур як цілі числа).

2. Вважати матриці і вектора з файлу у відповідні за розміром масиви. Звернете увагу, що у файлах міститься поряд дві або три матриці або вектори, їх слід занести в різні масиви.

Варіанти

1. 0.1 0.2 0.3 9.91 1.9 0.4 0.1 8.01 4.7 5.1 3.9 7.16	2. 1.399 2.001 9.921 3.21 0.12 0.129 1.865 8.341 9.33 8.01 9.136 8.401 7.133 3.12 3.22
3. 1 2 3 4 99 80 5 6 7 8 33 21 15 90	4. 10 20 40 50 12 19 21 32 44 -1 -2 -3 -4 32 10
5. 1 2 3 4 100 6 7 8 9 0.1 0.2 0.3 0.4 200 0.5 0.6 0.7 0.8 300	

СПИСОК ЛІТЕРАТУРИ

1. Акіменко В.В. Прикладні задачі інтелектуального аналізу даних (DATA MINING). – К.: КНУ ім. Тараса Шевченко, 2018. – 152 с.

2. Акіменко В.В., Загородній Ю.В. Лабораторний практикум з основ проектування баз знань. К.: КНУ ім. Тараса Шевченка.–2007.– 85 с.

3. Новотарський М.А., Нестеренко Б.Б. Штучні нейронні мережі: обчислення. – К.: Інститут Математики НАН України, 2004. – 408с