МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

МЕТОДИЧНІ ВКАЗІВКИ

до самостійної роботи студентів
"Пакети прикладних програм системного аналізу.
Комп’ютерна обробка даних в середовищі системи MatLab"

2019

УДК 681.3

Методичні вказівки до самостійної роботи студентів "Пакети прикладних програм системного аналізу. Комп’ютерна обробка даних в середовищі системи MatLab / Укладач О.В.Самощенко, 2019. - 115 с.

До збірника увійшли теми з дослідження засобів обробки даних з використанням типових функцій MATLAB та основних форм подання результатів. Розглянуті питання роботи в середовищі системи MATLAB для цифрової обробки сигналів, представлених у формі векторів та матриць.

Укладач: Самощенко О.В. – доцент, к.т.н, доцент кафедри системного аналізу

ЗМІСТ

1. ОСНОВИ РОБОТИ У СЕРЕДОВИЩІ СИСТЕМИ MATLAB

Мета: вивчення робочого середовища, методики роботи з командного рядка MATLAB, особливостей використання змінних, правил обчислення арифметичних виразів і застосування основних вбудованих функцій, вивчення можливостей вбудованого редактора та методики використання М-Файлів

1.1 МЕТОДИКА ОБЧИСЛЕНЬ У РОБОЧОМУ СЕРЕДОВИЩІ MATLAB

1.1.1 Робоче середовище MATLAB

При запуску MATLAB на екрані відкривається робоче середовище, основними елементами якого є:

- Меню;
- Панель інструментів з кнопками та списком, що розкривається;
- Вікна з вкладками Workspace і Current Directory для перегляду змінних та установки поточного каталогу;
- Вікно Command Windows, що служить для введення команд і виведення результату;
- Вікно Command History, призначене для перегляду і повторного виконання раніше введених команд;
- Рядок стану з кнопкою Start, при натисканні на яку відкривається меню для доступу до всіх основних засобів MATLAB.

У полі заголовка кожного вікна поруч із кнопкою закриття знаходиться кнопка Undock для доступу до вікна з робочого середовища, якщо воно убудовано, або кнопка Dock для вбудовування окремого вікна в робоче середовище.

1.1.2 Арифметичні обчислення і змінні

Для введення команд служить вікно Command Windows, у якому після символу запрошення >> необхідно набрати текст команди. Виконання команди (або обчислення виразу) здійснюється по натисканню на

клавішу Enter. Наприклад, якщо в командному рядку набрати 1 + 2 і нажати Enter, то в командному вікні відобразяться наступні вирази:

```
>> 1 + 2
ans =
        3
>> |
```

У випадку, коли не зазначена змінна для зберігання результату обчислень, MATLAB поміщає результат в змінну з ім'ям ans, після чого інформація про змінну ans з'являється у вікні Workspace.

Якщо потрібно виконати обчислення з використанням попереднього результату, наприклад (1+2)/4.5, у командному рядку необхідно набрати наступне:

```
>> ans/4.5
ans =
     0.6667
>> |
```

Інформація про використані змінних представлена у вікні Workspace у наступних полях:

- Name – ім'я змінної;
- Value – значення змінної;
- Bytes – об'єм пам'яті, що займається;
- Class – тип змінної (за умовчанням всі числові змінні представляються з подвійною точністю - double array);
- Size – розмірність масиву (Слід зазначити, що вміст стовпця Size наочно демонструє основний принцип роботи MATLAB, який полягає в тому, що в MATLAB усі дані представлені у вигляді масивів. У зазначеному вище прикладі змінна ans є двовимірним масивом розміру один на один).

Будь-який з перелічених вище стовпців можна сховати або відобразити за допомогою контекстного меню, що викликається правою кнопкою "миші".

За умовчанням усі змінні в списку розташовані за абеткою. Клік мишею по заголовку стовпця змінює порядок на зворотний. Аналогічно можна впорядковувати змінні за розміром займаної пам'яті й типом.

Подвійний клік по рядку зі змінною у вікні Workspace (або натискання на кнопку Open панелі інструментів вікна Workspace) призводить до відображення її вмісту в окремому вікні Array Editor, у якому можна змінювати значення елементів масиву.

Для виводу в командне вікно імен змінних, що використовуються, служить команда who, а більш докладну інформацію про змінні у вигляді таблиці, аналогічної до таблиці вікна Workspace, дозволяє отримати команда whos.

Для видалення з пам'яті всіх змінних використається команда clear без аргументів. Для видалення з пам'яті певного переліку змінних необхідно вказати їх в аргументах даної функції. Наприклад:

```
>> clear a1 a3
>> who
Your variables are:
a2
```

Довідатися про факт зайнятості змінної можна функцією exist, указавши ім'я змінної в апострофах у вхідному аргументі:

```
>> exist('d7')
ans =
     0
```

Якщо відповідь - нуль, то ім'я цієї змінної не конфліктує із зарезервованими словами MATLAB. Якщо значення, що повертається, дорівнює одиниці - така змінна вже визначена в робочому середовищі.

Вигляд, у якому виводиться результат обчислень, залежить від формату, установленого в MATLAB. Формат виводу встановлюється в опціях діалогового вікна Preferences, що викликається через меню Files/Preferences. Задавати формат виводу можна безпосередньо з командного рядка за допомогою команди format. Наприклад, для установки довго-

го із плаваючою крапкою формату виводу результатів обчислень треба в командному рядку ввести команду format long e:

```
>> format  long e
>> 1.25/3.11
ans =
        4.019292604501608e-001
```

Одержати інформацію про формати можна за допомогою команди:

>> help format.

1.1.3 Використання елементарних функцій

Для обчислення математичного виразу його слід набрати відповідно до правил MATLAB. Наприклад, необхідно обчислити значення виразу $e^{-2.5} \cdot (ln 11.3)^{0.3} - \sqrt{\frac{sin\, 2.45\pi + cos\, 3.78\pi}{tg\, 3.3}}$. У командному рядку цей вираз виглядатиме так:

```
>> exp(-2.5) * log(11.3) ^ 0.3 - sqrt((sin(2.45 * pi) + ...
cos(3.78 * pi)) / tan(3.3))
```

При записі довгого виразу для переносу його частини на інший рядок використається трикрапка (після попереднього пробілу).

По натисканні клавіші Enter відповідь виводиться в командне вікно:

```
ans =
     -3.2105
```

У наведеному виразі використані наступні убудовані функції MATLAB: обчислення експоненти, натурального логарифма, квадратного кореня й тригонометричних функцій. Аргументи функцій містяться у круглі дужках, імена функцій набираються малими літерами.

Пріоритет виконання арифметичних операцій у порядку зменшення наступний: зведення в ступінь, множення й ділення, додавання й вирахування. Для зміни цього порядку проходження необхідно використовувати круглі дужки.

Оскільки MATLAB запам'ятовує всі введені команди, то їх можна повторно занести в командний рядок без набору, використовуючи службові клавіші ↑, ↓.

При обчисленнях можливі виняткові ситуації. Ділення додатного числа на нуль формує нескінченність Inf, а ділення від'ємного числа на нуль – мінус нескінченність –Inf. При діленні нуля на нуль виходить NaN (не число). При обчисленні, наприклад, $\sqrt{-3}$ MATLAB переходить в область комплексних чисел:

```
>> sqrt(-3.0)
ans =
      0 + 1.7321i
```

При наборі комплексних чисел у командному рядку можна використати і або j, а самі числа при множенні, діленні й зведенні в ступінь необхідно писати в дужках:

```
>> (2.1 + 3.2i)*2 + (4.2 + 1.7i)^2
ans =
      18.9500 + 20.6800i
```

інакше множитися і зводитися в ступінь буде тільки уявна частина.

Для обчислення комплексно-сполученого числа застосовується апостроф, який слід набирати відразу за числом без пробілу:

```
>> 2 - 3i'
ans =
      2.0000 + 3.0000i
```

Для обчислення комплексно-сполученого виразу його необхідно взяти в круглі дужки:

```
>> ((3.2 + 1.5i)*2 + 4.2 + 7.9i)'
ans =
      10.6000 - 10.9000i
```

MATLAB дозволяє використовувати комплексні числа як аргументи убудованих елементарних функцій:

```
>> sin(2 + 3i)
ans =
      9.1545 - 4.1689i
```

Тригонометричні та зворотні до них функції (аргументи повинні бути виражені в радіанах):

- sin, cos, tan, cot – синус, косинус, тангенс, котангенс,
- sec, csc – секанс, косеканс,
- asin, acos, atan, acot – арксинус, арккосинус, арктангенс, арккотангенс,
- asec, acsc – арксеканс, арккосеканс.

Крім тригонометричних, реалізовані:

- exp – експонентна функція,
- log – натуральний логарифм,
- log10 – десятковий логарифм,
- log2 – логарифм за підставою 2,
- pow2 – зведення числа 2 у ступінь,
- sqrt – квадратний корінь,
- nextpow2 – ступінь, у яку треба звести число 2, щоб одержати найближче число.

Функції для роботи з комплексними числами:

- abs, angle – модуль і фаза (у радіанах) комплексного числа,
- complex – формування комплексного числа по його дійсній і мнимій частині, наприклад,

```
>> complex(2.3, 5.8)
ans = 2.3000 + 5.8000i
```

- conj – повертає комплексно-сполучене число,
- imag, real – мнима і дійсна частина комплексного числа.

Округлення та залишок від ділення:

- fix – округлення до найближчого цілого в напрямку до нуля,
- floor, ceil – округлення до найближчого цілого в напрямку до мінус нескінченності або плюс нескінченності,
- round – округлення до найближчого цілого,
- roundn – округлення із заданою точністю,
- mod – залишок від цілечисельного ділення (зі знаком другого аргументу),
- rem – залишок від цілечисельного ділення (зі знаком першого аргуме-

нту),

- sign – знак числа.

Одержати інформацію про вбудовані елементарні та спеціальні математичні функції можна відповідно за допомогою команд:

>> help elfun та >> help specfun

1.1.4 Використання змінних

В MATLAB в якості оператора присвоювання використається знак рівності. При цьому користувач може не піклуватися про те, які значення буде приймати змінна (комплексні, дійсні або цілі). Для присвоювання значення досить написати:

```
>> z = 1.45
```

По натисканні клавіші Enter MATLAB виведе значення z :

```
z =
    1.4500
```

Вивід значення можна придушувати завершенням командного рядка крапкою з комою:

```
>> z = 1.45;
```

Ім'ям змінної може бути будь-яка послідовність букв і цифр без пробілу, що починається з букви. Малі й прописні букви розрізняються. Припустима кількість символів в імені змінної становить 63.

Наприклад, вираз $\dfrac{\dfrac{sin1.3\pi}{ln3.4}+\sqrt{\dfrac{tg2.75}{th2.75}}}{\dfrac{sin1.3\pi}{ln3.4}-\sqrt{\dfrac{tg2.75}{th2.75}}}$ може бути обчислений з використанням змінних:

```
>> x = sin(1.3*pi) / log(3.4);
>> y = sqrt(tan(2.75) / tanh(2.75));
>> z = (x+y) / (x-y)
```

Результат виводиться на екран:

```
z =
    0.0243 - 0.9997i
```

а введені і збережені у пам’яті змінні відразу з'являються у вікні Workspace. Ці змінні можна використати в інших формулах. Наприклад, якщо потім необхідно обчислити вираз $\left[\frac{sin1.3\pi}{ln3.4}-\sqrt{\frac{tg2.75}{th2.75}}\right]^{3/2}$, то досить ввести команду

```
>> (x - y) ^ (3/2)
ans =
      -0.8139 + 0.3547i
```

1.1.5 Збереження і відновлення робочого середовища

Зберегти значення змінних з метою їхнього подальшого використання в наступних сеансах роботи з MATLAB можна за допомогою опції File/Save Workspace As. При цьому з'являється діалогове вікно Save to MAT-File, у якому треба вказати каталог та ім'я файлу. За умовчанням файл пропонується зберегти в поточному каталозі (підкаталог work основного каталогу MATLAB). Зберігати можна як всі, так і частину змінних, попередньо виділивши "мишею" необхідні.

Змінні у файлах з розширенням mat зберігаються у двійковому вигляді. Перегляд цих файлів у будь-якому текстовому редакторі не дає ніякої інформації про змінні та їхні значення.

Для відновлення значень змінних слід відкрити файл за допомогою опції File/Open.

Змінні робочого середовища зберігаються і відновлюються також за допомогою кнопок Load data file та Save на панелі інструментів вікна Workspace (для приховування або відображення панелі інструментів вікна треба викликати контекстне меню, клацнувши правою кнопкою на заголовку). При відновленні змінних після вибору файлу для їхнього завантаження виникає діалогове вікно, що дозволяє переглянути значення змінних і відзначити прапорцями змінні, що підлягають відновленню.

Зберігати і відновлювати змінні можна з командного рядка командами save і load відповідно. Розширення файлу можна не вказувати. Одержати інформацію про ці команди-функції можна по help save та help load.

В MATLAB є можливість за допомогою команди diary записувати в текстовий файл, який можна прочитати і роздрукувати, команди, що виконуються, і результати. Наприклад, наступна послідовність команд

```
>> diary  PrimBook.txt
>> a1 = 3;
>> a2 = 2.5;
>> a3 = a1 + a2
>> a3 =
      5.5000
>> save var1
>> diary off
```

створює файл PrimBook.txt і заносить у нього наступний текст:

```
a1 = 3;
a2 = 2.5;
a3 = a1 + a2
a3 =
      5.5000
save var1
diary off
```

Якщо в процесі виконання команд відбулися помилки, то повідомлення про них також будуть зареєстровані.

1.1.6 Зчитування і запис даних

Вихідні дані можуть зберігатися в текстових файлах. Для їхнього зчитування використається команда load, а для збереження результатів у текстовому файлі - команда save. Дані записуються і зчитуються по рядкам. Наступний набір команд

```
>> A = load('mtr.txt');
>> b = load('rsd.txt');
>> x = A\b;
>> save  's.txt'   x  -ascii
```

створює в каталозі work файл s.txt, у якому записаний результат обробки даних з файлів mtr.txt та rsd.txt. Параметр -ascii означає запис у текстовому форматі. Подивитися вміст файлу можна в будь-якому текстовому редакторі.

Для запису результату у файл із подвійною точністю слід використовувати команду save 's.txt' x -ascii -double.

Записати дані безпосередньо у файл можна також по команді save, наприклад,

>> save 'mr.txt' A -ascii

1.1.7 Робота з вікном Command History

Для виклику раніше уведених команд є вікно Command History, у якому зберігається історія команд. Дана історія являє собою запис команд всіх проведених сеансів роботи з MATLAB, які автоматично зберігаються у файлі history.m. На початку протоколу роботи кожного сеансу відзначені час і дата його початку.

Відключити запис команд можна в діалоговому вікні, що викликається опцією File/Preferences.

Для виконання команди з вікна Command History, необхідно виділити шукану команду (або набір команд) і натиснути на клавішу Enter (або зробити подвійний клік лівою кнопкою миші). Виділену команду можна також перетягнути у вікні Command Windows за допомогою курсору.

Виконувати дії по відношенню до команд списку можна з використанням правої кнопки миші, по натисканні на яку з'являється відповідне меню. З його допомогою можна створити ярлик для групи виділених команд, що буде розміщений на панелі робочого середовища.

Знайти необхідну команду можна за допомогою опції Find меню Edit (або комбінації клавіш Ctrl+F).

1.2 СТВОРЕННЯ І РЕДАГУВАННЯ М-ФАЙЛІВ

1.2.1 Редактор М-файлів

Користувальницькі програми в MATLAB оформляються у вигляді М-файлів, які можна запускати з робочого середовища або з редактора. Убудований в MATLAB редактор М-файлів дозволяє не тільки набирати текст програми, запускати її цілком або частинами, але і налагоджувати алгоритм.

Новий М-файл відкривається командою головного меню File/New/M-file або комбінацією клавіш Ctrl+N. Новий файл відкривається у вікні редактора М-файлів. Вигляд рядка і панелі інструментів залежить від ширини вікна. Як приклад у редакторі можна набрати наступну програму:

```
a1 = 3;
a2 = 2.5;
b1 = a1 + a2        % додавання
a3 = 2.6;
a4 = 1.7;
b2 = a3 – a4        % віднімання
```

Перед запуском програми її слід зберегти по команді головного меню File/SaveAs , наприклад, під ім'ям Prim1.m. Для запуску на виконання всіх команд, що містяться у файлі, слід вибрати пункт Run у меню Debug, або натиснути клавішу F5. Але робити попереднє збереження як окрему операцію не обов'язково. Припустимо відразу розкривати меню Debug, у якому опція Run у цьому випадку заміняється опцією Save and Run, що дозволяє запустити програму, попередньо зберігши її автоматично. Таким чином, файл можна запустити відразу після редагування (якщо не потрібно йому присвоїти нове ім'я) натисканням клавіші F5.

Редактор дозволяє виконувати частину команд файлу. Наприклад, можна виконати перші три команди файлу Prim1.m, виділивши їх за допомогою "миші" і вибравши опцію Evaluate Selection меню Text або на-

тиснувши клавішу F9. Аналогічно можна виконати три команди прикладу, що залишилися.

Коментарі в MATLAB починаються зі знаку відсотка і автоматично виділяються зеленим кольором (за умовчанням). Для виключення частини коду, що виконується, без його видалення або у випадку довгого коментарю, можна використовувати блок коментарів, що починається з рядка із двох символів %{ (знак відсотка та фігурна дужка, що відкривається) і закінчується рядком із двох символів %} (фігурна дужка, що закривається, та знак відсотка).

Існуючий М-файл відкривається опцією File/Open робочого середовища або редактора М-файлів. Відкрити файл у редакторі можна командою edit з командного рядка, вказавши як аргумент ім'я файлу, наприклад

```
>> edit prim1.m
```

Команда edit без аргументу відкриває редактор і створює новий файл без імені. Якщо уведено команду edit з ім'ям неіснуючого файлу, то редактор сприймає це як запит на створення нового М-файлу із зазначеним ім'ям.

У редакторі може бути одночасно відкрито кілька файлів. MATLAB дозволяє змінювати спосіб відображення файлів у редакторі. За умовчанням вікно editor редактора одне, і при відкритті кожного нового файлу воно забезпечується закладкою внизу робочої області з ім'ям файлу. Останні п'ять кнопок на панелі інструментів дозволяють вибрати спосіб розташування вікон з файлами в робочій області редактора. Наприклад, її можна розділити по горизонталі або по вертикалі. Інструменти Dock і UnDock (розташовані в правій частині рядка меню) дозволяють відкрити кожен файл у своєму вікні редактора або, навпаки, вбудувати його в інше вікно редактора.

Для зміни настроювань необхідно вибрати в меню File редактора або робочого середовища опцію Preferences. У лівій частині вікна настроювань відображені назви компонентів, частина з яких представлена

списком, що розкривається, який дозволяє перейти до необхідної групи властивостей.

Можливе використання редактора М-файлів без запуску MATLAB. Для цього розширення m в Windows повинне бути асоційоване з додатком meditor.exe. У цьому випадку файл відкриється в редакторі за подвійним натисканням мишею по імені М-файлу. Файл можна редагувати, але не виконувати.

М-файли бувають двох типів: файл-програми (Script M-Files), що містять послідовність команд, і файл-функції (Function M-Files), у яких описуються функції, визначені користувачем.

1.2.2 Файл-програми

Файл-програми не мають вхідних і вихідних аргументів і оперують змінними, існуючими в робочому середовищі, або можуть створювати нові змінні. Всі змінні, оголошені у файл-програмі, стають доступними в робочому середовищі після її виконання, що підтверджується їхньою появою в робочому середовищі Workspace. Всі створені змінні залишаються в робочому середовищі після завершення М-файлу і їх можна використовувати в інших файл-програмах, що виконуються з командного рядка.

Файл-програми запускаються з редактора М-файлів, з командного рядка або іншої файл-програми (при цьому в якості команди використовується ім'я М-файлу без розширення). MATLAB виконує наступне:

1. Перевіряє, чи є введена команда ім'ям якої-небудь із змінних, визначених у робочому середовищі. Якщо введено змінну, то виводиться її значення.
2. Якщо введено не змінну, MATLAB шукає введену команду серед убудованих функцій і, при позитивному результаті, виконує функцію.
3. Якщо введено не змінну і не убудовану функцію, то MATLAB відшукує М-файл із назвою команди і розширенням m. Якщо файл

не знайдено у поточному каталозі, то MATLAB переглядає каталоги, установлені в шляху пошуку.

Якщо жодна з перелічених вище дій не привела до успіху, то в командне вікно виводиться повідомлення про помилку.

Довідатися про факт зайнятості імені можна по команді exist.

Поточний каталог і шляхи пошуку встановлюються за допомогою команд інтерфейсу робочого середовища або з використанням команд. Уміст поточного каталогу відображається у вікні Current Directory з однойменною вкладкою. Поточний каталог установлюється вибором зі списку, що розкривається, Current Directory на панелі інструментів робочого середовища MATLAB. Якщо в списку немає потрібного каталогу, то його можна додати в діалоговому вікні Browse for Folder, що з'являється після натискання на кнопку, розташовану праворуч від списку.

Директорії для пошуку файлів визначаються в діалоговому вікні Set Patch навігатора шляхів (File/Set Patch). Рекомендується зберігати власні файли поза підкаталогом toolbox основного каталогу MATLAB, тому що ці файли можуть бути знищені при переустановці MATLAB, а також через особливості використання файлів у цьому підкаталозі.

Дії з установки шляхів дублюються командами. Поточний каталог встановлюється командою cd, наприклад

>> cd c:\students\donchanka або >> cd('c:\students\donchanka').

Для додавання каталогів до шляху пошуку служить команда addpath, що за умовчанням поміщає каталог у початок списку пошуку, наприклад

>> addpath c:\magisters

Для додавання каталогу в кінець списку необхідно використовувати параметр

–end:

>> addpath c:\specialists -end

Результат виконання можна перевірити за допомогою команди path, яка повертає список каталогів, що входять до шляхів пошуку.

Для видалення каталогу зі списку шляхів пошуку призначена функція rmpath:

>> rmpath c:\bacalavrs або >> rmpath ('c:\ bacalavrs').

1.2.3 Файл-функції

Файл-функції виконують необхідні дії із вхідними аргументами і повертають результат у вихідних аргументах. Ідентифікація файлу як функції виконується установкою слова function у першому рядку, тобто в заголовку функції:

```
function  f = myfun(x)
f = exp(-x) * sqrt((x^2 + 1) / (x^4 + 0.1));
```

В заголовку функції також розміщується ім'я функції і списки вхідних і вихідних аргументів. Вхідні аргументи записуються в круглих дужках після імені функції (у прикладі – це x). Вихідний аргумент указується ліворуч від знака рівності в заголовку функції (у прикладі – це f).

Після заголовка розміщується тіло файл-функції - один або кілька операторів, які реалізовують алгоритм одержання значення вихідних значень із вхідних.

Використовувати файл-функцію можна як і убудовані функції, наприклад

```
>> y = myfun(1.3)
y =
    0.2600
```

Каталог, у якому містяться файл-функції, повинен бути поточним, або шлях до нього повинен бути доданий до шляхів пошуку.

Використання файл-функцій спрощує візуалізацію результатів математичних дій. Наприклад, потрібно визначити значення функції myfun на відрізку [0, 4]. З командного рядка це виглядатиме так:

```
>> x = 0:0.5:4;
>> y = myfun(x);
```

>> plot(x, y)

Результат виконання показаний на рис. 1.

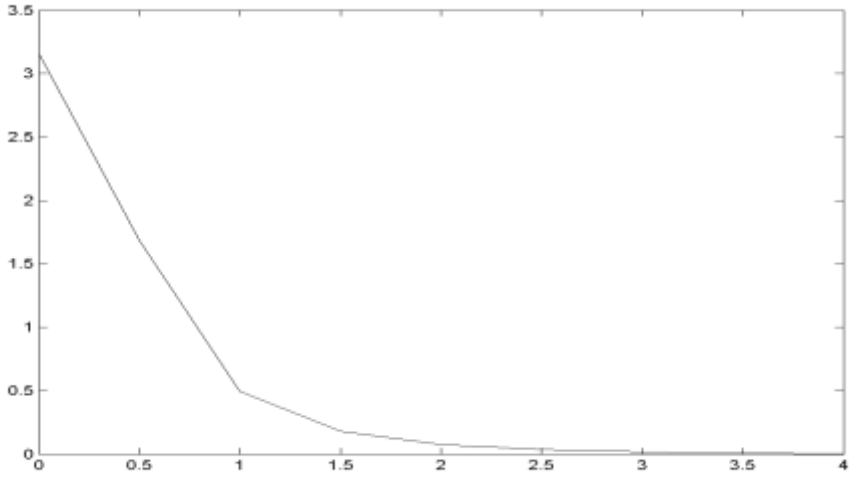


Рис. 1 Приклад використання plot

Аналогічну задачу можна вирішити за допомогою функції fplot, якій потрібно вказати ім'я файлу-функції myfun або вказівник на неї (з оператором @ перед ім'ям функції) і границі відрізка для побудови графіка:

>> fplot('myfun', [0 4]) або >> fplot(@myfun, [0 4]).

Результат виконання показаний на рис. 2.

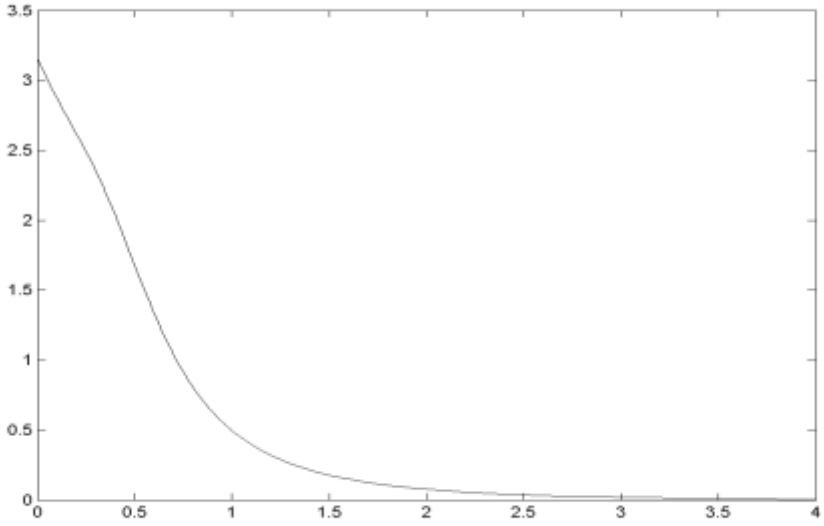


Рис. 2 Приклад використання fplot

Графік, побудований fplot, більш точно відбиває поведінку функції, тому що алгоритм автоматично підбирає крок аргументу, зменшуючи його на ділянках швидкої зміни досліджуваної функції.

Якщо функція має кілька вхідних аргументів, то вони розміщаються в списку через кому.

Якщо список вихідних аргументів порожній, то файл-функція не буде повертати ніяких значень.

Якщо функція має кілька вихідних аргументів (тобто це вектор-функція), то вони додаються через кому в список вихідних аргументів. Наприклад, файл-функція для переведення часу, заданого в секундах, у години, хвилини та секунди:

```
function [hour, minute, second] = hms(sec)
hour = floor(sec/3600);
minute = floor((sec - hour*3600)/60);
second = sec - hour*3600 - minute*60;
```

При виклику файлу-функції з кількома вихідними аргументами результат слід записувати у вектор відповідної довжини:

```
>> [H, M, S] = hms(10000)
H =
     2
M =
    46
S =
    40
```

Файл-функції можна супроводжувати коментарями. Всі коментарі після заголовка й до тіла функції або порожнього рядка виводяться в командне вікно по команді help. Наприклад:

```
function [hour, minute, second] = hms(sec)
 % hms - переведення секунд у години, хвилини та секунди
 % [hour, minute, second] = hms(sec)
 %       sec - число секунд
 %       hour - число повних годин
 %       minute - число повних хвилин
 %       second - залишок секунд
hour = floor(sec/3600);
```

```
minute = floor((sec - hour*3600)/60);
second = sec - hour*3600 - minute*60;
```

Перший рядок коментарів після заголовка функції називається H1-line і використається при пошуку командою lookfor. Ця команда шукає зазначене слово в рядках H1-line всіх файлів-функцій у каталогах, зазначених у шляхах пошуку, у тому числі, і в поточному каталозі.

1.3 РІЗНОВИДИ ФУНКЦІЙ

MATLAB надає різні способи організації зв'язків між функціями - приватні функції, підфункції, вкладені функції.

Використання **підфункцій** і **вкладених функцій** засноване на виділенні частини алгоритму в самостійну функцію, текст якої міститься в тому ж файлі, що й основна функція. Наприклад, є файл-функція simple, що зберігається у файлі simple.m, у якій деяке значення обчислюється часто:

```
function simple;
x = 1.1;
y = 2.1;
f1 = x^3 - 2*y^3 + 3*(x^2 + y^2) - x*y + 9
x = 3.1;
y = 4.2;
f2 = x^3 - 2*y^3 + 3*(x^2 + y^2) - x*y + 9
x = -2.8;
y = 0.7;
f3 = x^3 - 2*y^3 + 3*(x^2 + y^2) - x*y + 9
```

Бажано визначити вираз, що обчислюється, у підфункції f із двома вхідними та одним вихідним аргументом і помістити її в тому ж М-файлі simple.m:

```
function simple;
 % Основна функція
f1 = f(1.1, 2.1)
f2 = f(3.1, 4.2)
f3 = f(-2.8, 0.7)
function  z = f(x, y)
 % Підфункція
z = x^3 - 2*y^3 + 3*(x^2 + y^2) - x*y + 9;
```

Перша функція simple є основною функцією в simple.m, саме її оператори виконуються, якщо користувач викликає simple, наприклад, з командного рядка. Файл-функція може містити одну або кілька підфункцій зі своїми вхідними і вихідними параметрами, але основна функція може бути тільки одна. Заголовок нової підфункції одночасно є ознакою кінця попередньої. Основна функція обмінюється інформацією з підфункціями тільки за допомогою вхідних і вихідних параметрів.

Змінні, визначені в підфункціях і в основній функції, є локальними. Один з можливих варіантів використання змінних, які є загальними для всіх функцій М-файлу, полягає в оголошенні даних змінних на початку основної функції і підфункції як глобальних, за допомогою global зі списком імен змінних, відокремлених пробілом:

```
function simple2;
global  ALP  BET
ALP = 5.3;
BET = 9.1;
f1 = f(1.1, 2.1)
function   z = f(x, y)
global   ALP  BET
z = x^3 - 2*y^3 + 3*(x^2 + y^2) - x*y + 9 + ALP*BET;
```

Кращим способом обміну змінними між основною функцією та підфункціями є передача їх у параметрах підфункції. Значення глобальних змінних може бути випадково змінене в робочому середовищі або при виклику іншої файл-програми або файл-функції, що звертається до однойменних глобальних змінних.

Підфункція доступна тільки усередині основної функції, тобто спроба виклику підфункції f не з файл-функції simple, а з іншої файл-функції, файл-програми або з командного рядка призведе до помилки. Повідомлення про це може й не з'явитися, тому що ім'ям f може бути пойменована файл-функція, що зберігається у файлі f.m, або в робочому середовищі оголошено змінну f. Довідатися про те, чи є функція MATLAB убудованою, дозволяє спеціальна функція exist , що повертає, наступні аргументи:

1 – ім'я зайняте під змінну робочого середовища, 2 – функція розташована в шляхах пошуку файлів, 5 – убудована функція (built-in function). Приватні функції і підфункції не ідентифікуються.

На відміну від підфункції **вкладена функція** є внутрішньою. Тому змінні із середовища основної функції доступні також у вкладеній функції:

```
function main;
ALP = 5.3;
BET = 9.1;
f1 = fnested(1.1,  2.1)
        function  z = fnested(x, y)
                z = x^3 - 2*y^3 + 3*(x^2 + y^2) - x*y + 9 + ALP*BET;
        end
end
```

Необхідно використовувати оператор end для закриття тіла функції. Вкладена функція може розміщатися в будь-якому місці тіла функції, що її містить. Основна функція також завершується оператором end. В одному М-файлі допускається використання підфункцій і вкладених функцій одночасно, але останнім оператором підфункції повинен бути end.

Рівень вкладеності функцій не обмежений. Функція може звернутися до своєї вкладеної функції, але не може використовувати вкладену функцію нижнього рівня. Вкладена функція може звернутися до функції того ж рівня. Функція нижнього рівня може викликати функцію верхнього рівня, у яку вона вкладена, і всі функції, доступні з неї.

Змінні, визначені в зовнішній функції, доступні також у вкладеній функції, і навпаки. Виключення становить випадок колізії змінних для функцій одного рівня. У цьому випадку у вкладених функціях це різні локальні змінні з одним ім'ям. Зрозуміло, що в зовнішній функції доступ до двох змінних з одним ім'ям не можливий, тому жодна з них не доступна.

Використання вказівника на вкладену функцію дозволяє забезпечити доступ до вкладених функцій не тільки в М-файлі. У наступному прикладі наведене використання вкладеної функції для створення двох однотипних функцій з різними параметрами. Припустимо, існує функція fixed_parm:

```
function pointer = fixed_parm(a, b, c)
 % a, b, c - параметри процесу.
 % Створення вказівника на вкладену функцію
pointer = @process;
        function  y = process(t)
                y = a*sin(b.*t + c);
        end
end
```

Тоді при виконанні команд:

```
>> f = fixed_parm(1.5, 2, 0);
>> g = fixed_parm(1, 10, 25);
>> fplot(f, [-2, 2])
>> hold on;
>> fplot(g, [-2, 2])
```

будуть побудовані графіки двох різних функцій. Таким чином, створено дві функції f і g, що залежать від однієї змінної, для яких значення параметрів зафіксовані на момент звертання до функції fixed_parm (рис.3).

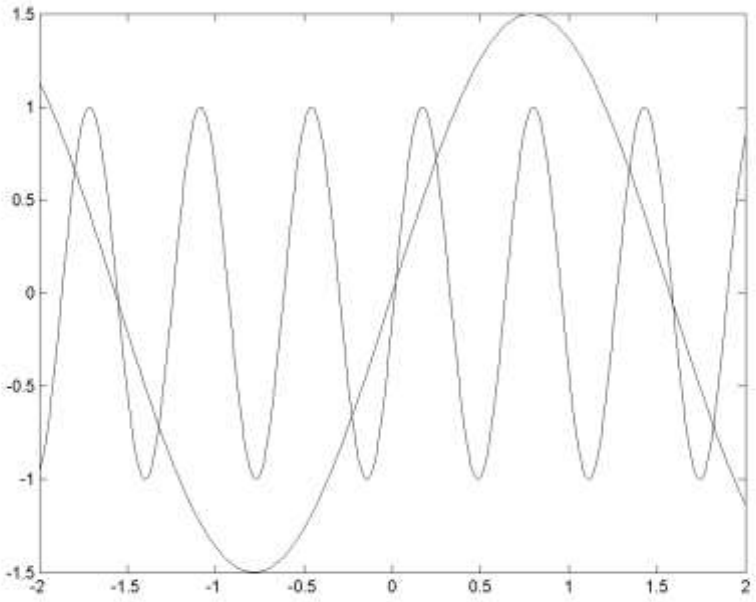


Рис. 3 Графік двох функцій однієї змінної

Керування доступністю файлів-функцій з боку інших файлів-функцій забезпечується використанням **приватних функцій**. Найпростіший прийом зробити функції доступними чи ні, полягає в розміщенні М-файлів у робочих каталогах. Визначена структура каталогу з файл-функціями користувача дозволяє задати деякі допоміжні функції, які використовуються тільки файл-функціями, що містяться в М-файлах даного ка-

талогу, а для файл-функцій з інших каталогів є недоступними. Для цього слід створити підкаталог з ім'ям private і розмістити в ньому допоміжні файли-функції. Наприклад, файл-функція c:\work2\test.m не має доступу до приватної функції c:\work1\private\fun1.m.

Анонімні функції і такі, що вбудовуються. Якщо досліджувана функція задається простій і короткою формулою, то не обов'язково використовувати файл-функцію. Замість неї зручно ввести функцію, що вбудовується (inline-функцію), скориставшись функцією inline, або визначити анонімну функцію.

Звертання до функції, що вбудовується, виглядає в такий спосіб:

Ім'я_функції=inline('формула', список_аргументів)

Список аргументів не обов'язковий, а формула є текстовим рядком і задає вираження для обчислення значення функції. Наприклад,

```
>> fun = inline('sin(x) - x.^2.*cos(x)')
```

Функція fun може бути використана як будь-яка інша функція, наприклад,

```
>> y=fun(0.5)
y=0.2600
```

Якщо функція залежить від декількох змінних, то всі вони є аргументами уведеної inline-функції і розташовуються за абеткою:

```
>> fun1=inline('sin(a*x) - x.^2.*cos(b*x)')
```

Для зміни порядку аргументів їх варто перелічити через коми в списку після вираження, яке визначає вид функції:

```
>> fun2=inline('sin(a*x) - x.^2.*cos(b*x)', 'x', 'a', 'b')
```

Всі аргументи функції повинні бути символьними рядками, укладеними в апострофи, або строковими змінними.

Якщо в списку пропущений хоча б один з аргументів, то скористатися inline-функцією не вдасться. Навіть при наявності пропущеної змінної в робочому середовищі виклик функції приведе до повідомлення про те, що аргумент не заданий, тобто змінні робочого середовища при обчислен-

ні значень inline-функції недоступні.

Альтернативний спосіб завдання функції складається в оголошенні анонімної функції за допомогою оператора покажчика @:

Ім'я_функції = @(список_аргументів) формула

На відміну від inline-функції, і аргументи, і формула записуються у звичайному виді, а не як текстові рядки в апострофах. Крім того, анонімної функції доступні змінні робочого середовища, які входять у формулу. Однак вони є константами, у якості яких беруться значення цих змінних у момент створення анонімної функції, і наступна зміна їхнього значень не буде враховуватися при обчисленні функції:

```
>> a=1;
>> gun3 = @(x,b) (sin(a*x) - x.^2.*cos(b*x))
>> gun3(5, 0)
ans = -25.9589
>> a = 1000;
>> gun3(5, 0)
ans = -25.9589
```

По способу використання анонімна функція нагадує inline-функцію, але відрізняється тим, що створюється покажчик на функцію, що пов'язаний з кодом, що виконується (це видно у вікні Workspace - інформація про виділення пам'яті показує, що для анонімної функції код, що виконується, і покажчик на неї відділені, а для inline-функції - це єдиний об'єкт).

1.4 РОЗБИВКА М-ФАЙЛУ НА ЕЛЕМЕНТИ

Редактор дозволяє розбити всю програму на елементи і виконувати їх незалежно. У цьому разі текст розбивається за допомогою рядків коментарів, що починаються із двох знаків відсотка (%%).

Для використання засобів редактора необхідно встановити режим Cell Mode, вибравши в меню Cell пункт Enable Cell Mode. При цьому стають доступними інші пункти цього меню, а обраний зміниться на Disable Cell Mode для скасування режиму. Крім цього, з'являється панель

інструментів для роботи з елементами. Для оформлення файлу можна використовувати наступні пункти меню Cell:

- Insert Cell Divider – вставка роздільника елементів із двох символів відсотка (%%). Якщо курсор знаходиться не в першій позиції рядка, то роздільник вставляється за поточним рядком, інакше - перед ним;
- Insert Cell Divider around Selection – вставка роздільника елементів (%%) до і після виділеного фрагмента;
- Insert Text Markup – пункт меню, що розкривається, для вставки рядків зразків коментарів у місце, де розташований курсор;
- Evaluate Current Cell – виконуються рядки поточного елементу, після чого він стає поточним;
- Evaluate Current Cell and Advance – виконуються рядки поточного елементу, після чого поточним стає наступний елемент;
- Evaluate Entire File – виконується весь файл.

При виконанні файлу потрібно, щоб змінні, що використовуються в командах і функціях, були присутніми у середовищі Workspace. Виконання елементів М-файлу в режимі Cell Mode відбувається без попереднього збереження його на диску, на відміну від запуску усього файлу за клавішею F5. Переміщатися по елементах вверх та вниз можна за опціями Previous Cell і Next Cell меню Cell.

Для файл-функцій застосування технології розбивки на елементи пов’язане із труднощами, оскільки їх змінні є локальними.

1.5 ДІАГНОСТИКА М-ФАЙЛІВ

У складі середовища MATLAB є засіб M-Lint для перевірки коректності написаного коду. Для цього після збереження М-файлу треба в меню редактора Tools вибрати пункт Check Code with M-Lint (якщо файл не був збережений, то цей пункт меню називається Save and Check Code with M-Lint). По закінченню перевірки з'явиться вікно M-Lint Code Check Report зі звітом.

1.6 ЗАВДАННЯ ДЛЯ САМОКОНТРОЛЮ

Указати вхідні змінні та обчислити значення математичного вираження. Результат вивести в різних форматах. З метою перевірки вирішити математичну задачу формалізованим способом. Вивчити задані функції MATLAB і пояснити їхнє застосування для обчислення математичного вираження.

Програму обчислення значень математичного вираження оформити у вигляді М-Файлу

1.7 ЗАПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

- Що таке "робоче середовище MATLAB" ?
- Для чого використовуються кнопки Dock та Undock ?
- Яке призначення вікна Command Windows ?
- Яке призначення й структура вікна Workspace ?
- Яке призначення вікна Array Editor ?
- Як використовуються команди who, whos, clear, exist ?
- Як встановлюється формат виводу результатів ?
- Як використовується команда format ?
- Яке призначення службових клавіш ↑, ↓ ?
- У яких випадках можлива поява виняткових ситуацій ?
- У чому полягає особливість запису комплексних чисел ?
- У чому полягає особливість комплексно-сполучених виразів ?
- Як можна одержати інформацію про убудовані елементарні функції ?
- Як можна одержати інформацію про спеціальні математичні функції?
- Які вимоги до запису імен змінних ?
- Як зберігаються та відновлюються змінні робочого середовища ?
- У чому полягає особливість зберігання змінних у файлах з розширенням mat ?
- Як використовується команда diary ?

- Які можливості MATLAB пропонує для роботи з текстовими файлами ?
- Для чого використовується файл history.m ?
- Яке призначення М-Файлів ?
- Як вказуються коментарі в М-Файлах ?
- Яка відмінність файл-програм від файлів-функцій ?
- Які особливості використання файл-програм ?
- Які особливості використання файл-функцій ?
- Яке призначення функції fplot ?
- Яке призначення підфункцій та вкладених функцій ?
- Яке призначення приватних функцій ?

2. РОБОТА З ВЕКТОРАМИ

Мета: вивчення методів обробки даних, що представлені у формі векторів, і виводу результатів

2.1 МЕТОДИ ОБРОБКИ ВЕКТОРІВ

2.1.1 Основні визначення

Усі дані в MATLAB, наприклад змінні, вектори, матриці, які є по своїй суті математичними об'єктами, представляються у вигляді масивів.

Масив – упорядкована, пронумерована сукупність однорідних даних. У масиву повинне бути ім'я. Масиви розрізняються за кількістю розмірностей або вимірів: однорідні, двовимірні, багатомірні. Розмірністю масиву називають число елементів уздовж кожного з вимірів. Елементи доступні через індекси. Нумерація елементів масивів в MATLAB починається з одиниці.

Вектор може бути записаний у стовпчик та у рядок. Вектори і матриці позначаються *курсивом*, а відповідні їм масиви прямим шрифтом.

2.1.2 Введення, додавання і вирахування векторів

Потрібно скласти два вектори:

$$a = \begin{pmatrix} 1.3 \\ 5.4 \\ 6.9 \end{pmatrix} \quad \text{та} \quad b = \begin{pmatrix} 7.1 \\ 3.5 \\ 8.2 \end{pmatrix}$$

Для зберігання векторів використовуються масиви a та b, які вводяться в командному рядку з використанням квадратних дужок і з відокремленням елементів векторів крапкою з комою:

```
>> a = [1.3; 5.4; 6.9]
a =
   1.3000
   5.4000
   6.9000
>> b = [7.1; 3.5; 8.2];
```

Для знаходження суми векторів використовується знак плюс:

```
>> c = a + b
c =
   8.4000
   8.9000
  15.1000
```

Інформація у вікні Workspace пояснює, що вектори *a, b, c* зберігаються у двовимірних масивах. У стовпці Size зазначено 3x1, тобто масиви мають три рядки та один стовпець, кожний з них займає 24 байта пам'яті. Аналогічні відомості можна одержати за допомогою убудованих функцій ndims і size:

```
>> na = ndims(a)
na =
        2
>> sa = size(a)
sa =
        3        1
```

Матричне подання даних дозволяє обчислювати значення тригонометричних функцій по відношенню до всіх елементів вектора:

```
>> d = sin(c)
d =
   0.8546
   0.5010
   0.5712
```

Таким чином, якщо аргумент функції є масивом, то результат функції буде масивом того ж розміру, але з елементами, рівними значенню функції від відповідних елементів вихідного масиву. Наприклад, добування квадратного кореня з елементів вектора *d* зі знаком мінус:

```
>> sqrt(-d)
ans =
      0 + 0.9244i
      0 + 0.7078i
      0 + 0.7558i
```

Вектор-рядок вводиться у квадратних дужках, однак на відміну від вектора-стовпця елементи слід відокремлювати пробілами або комами. Операції виконуються аналогічно, у результаті чого формується вектор-рядок. Наприклад:

```
>> s1 = [3 4 9 2]
s1 =
    3     4     9     2
>> s2 = [5 3 3 2]
s2 =
    5     3     3     2
>> s3 = s1 + s2
s3 =
    8     7    12     4
>> s4 = log(s3)
s4 =
   2.0794    1.9459    2.4849    1.3863
```

Інформація про масиви, у яких зберігаються вектори-рядки, з'являється у вікні Workspace. Для визначення довжини векторів-стовпців або векторів-рядків використовується убудована функція length:

```
>> L = length(s1)
L =
    4
```

За умовчанням всі елементи масивів зберігаються з подвійною точністю (double) і займають 8 байтів. Можливе зберігання елементів масивів в інших форматах: single – для дійсних чисел, що вимагають для розміщення 4 байти, int8, int16, int32 – для цілих чисел, що займають 1, 2 або 4 байти відповідно. Для зміни формату подання призначені однойменні з типом (Class) функції:

```
>> q4 = single(s4);
>> q3 = int32(s3);
>> q2 = int16(s2);
>> q1 = int8(s1);
```

При перетворенні цілого числа у випадку, коли для його точного подання не вистачає відведеної пам’яті, виходить максимальне число для да-

ного типу без попередження про помилку. Тому застосування цілих чисел обмежене. Спроба виконання арифметичних операцій над цілими числами різних типів також призводить до помилки.

2.1.3 Зчеплення і редагування векторів

З кількох векторів-стовпців можна скласти один, використовуючи квадратні дужки і відокремлюючи вихідні вектори-стовпці крапкою з комою:

```
>> v1 = [1; 2];
>> v2 = [3; 4; 5];
>> v = [v1; v2]
v =
    1
    2
    3
    4
    5
```

Для зчеплення векторів-рядків також застосовуються квадратні дужки, але вектор-рядки, що зчіплюються, відокремлюються пробілами або комами:

```
>> v1 = [pi  pi/2];
>> v2 = [pi/3  pi/4  pi/5];
>> v = [v1  v2]
v =
   3.1416    1.5708    1.0472    0.7854    0.6283
```

Для перегляду та зміни значень елементів масивів зручно використати редактор масивів Array Editor. Вікно редактора відкривається після подвійного кліка "мишею" по імені масиву v у вікні Workspace або натисканню кнопки Open Selection на панелі інструментів вікна Workspace при положенні курсору на імені масиву. Перехід у редагування елементу масиву - при натисканні на кнопку F2. Редагувати масив можна за допомогою опцій контекстного меню і панелі інструментів. Слід зазначити, що

операція Cut призводить тільки до обнуління елемента масиву, а для його видалення необхідно викликати Delete.

Для роботи з даними редактор масивів MATLAB надає можливості, аналогічні MS Excel.

Редактор масивів дозволяє переглядати значення елементів декількох масивів. Для цього досить подвійним кліком зробити виклик чергового масиву та установити за допомогою панелі інструментів зручний варіант перегляду.

2.1.4 Звертання до елементів вектора

Для доступу до елементів вектора використовується індекс, що міститься у круглих дужках після імені масиву, у якому зберігається вектор. Наприклад, для доступу до четвертого елемента масиву v, визначеного вектором-рядком

```
>> v = [1.3  3.6  7.4  8.2  0.9];
```

використається індексація:

```
>> h = v(4)
h =
        8.2000
```

Зазначення елемента масиву в лівій частині оператора присвоювання призводить до зміни в масиві:

```
>> v(2) = 555
v =
        1.3000  555.0000    7.4000    8.2000    0.9000
```

З елементів масиву можна формувати нові масиви, наприклад:

```
>> u = [v(3);  v(2);  v(1)]
u =
   7.4000
   555.0000
   1.3000
```

Для переносу визначених елементів вектора в інший вектор у заданому порядку служить індексація за допомогою вектора. Наприклад, за-

пис у вектор-рядок w четвертого, другого і п'ятого елементів v провадиться в такий спосіб:

```
>> ind = [4  2  5];
>> w = v(ind)
w =
   8.2000  555.0000    0.9000
```

Для звертання до блоків послідовно розташованих елементів вектора служить індексація за допомогою знака двокрапки. Наприклад, для заміни нулями елементів з другого по шостий у векторі-рядку w з семи елементів потрібно виконати:

```
>> w = [0.1  2.9  3.3  5.1  2.6  7.1  9.8];
>> w(2:6) = 0;
>> w
w =
   0.1000         0         0         0         0         0    9.8000
```

Індексація за допомогою двокрапки використається й при виділенні частини з великого об'єму в новий масив:

```
>> w = [0.1  2.9  3.3  5.1  2.6  7.1  9.8];
>> w1 = w(3:5)
w1 =
   3.3000    5.1000    2.6000
```

Наступний приклад показує формування вектора-рядка w2, що містить елементи w, крім четвертого:

```
>> w2 = [w(1:3)  w(5:7)]
w2 =
        0.1000    2.9000    3.3000    2.6000    7.1000    9.8000
```

Такий же результат дає запис w(5:end) замість w(5:7).

Елементи масиву можуть входити у вирази. Обчислення, наприклад, середнього геометричного з елементів вектора u можна зробити в такий спосіб:

```
>> gm = (u(1) * u(2) * u(3)) ^ (1/3)
gm =
        17.4779
```

2.1.5 Застосування функцій обробки даних

Елементи вектора перемножуються за допомогою функції prod:

```
>> z = [3;  2;  1;  4;  6;  5];
>> p = prod(z)
p =
    720
```

Таким чином, знайти середнє геометричне можна в такий спосіб:

```
>> gm = prod(z) ^ (1/length(z))
gm =
   2.9938
```

Функція sum призначена для підсумовування елементів вектора.

Для обчислення середнього арифметичного – функція mean:

```
>> mean(z)
ans =
        3.5000
```

Для знаходження мінімуму й максимуму з елементів вектора – функції min й max:

```
>> M = max(z)
M =
        6
>> m = min(z)
m =
        1
```

При виклику функції min із двома векторами в якості вхідних аргументів вийде вектор, кожний елемент якого складається мінімум із двох елементів вихідних векторів з однаковими розмірами:

```
>> p = [3  12  8];
>> s = [4  10  7];
>> min (p, s)
ans =
    3    10     7
```

Наступна команда дозволяє визначити індекс мінімального елемента масиву, що буде занесений у змінну k:

```
>> [m, k] = min(z)
```

```
m =
        1
k =
        3
```

Для пошуку індексів елементів, що задовольняють визначеній умові, використовується функція find. Наприклад, потрібно знайти номери всіх елементів вектору x , рівних максимальному значенню (знак == позначає логічну рівність):

```
>> x = [1  2  5  3  4  5  1  5];
km = find(x == max(x))
km =     3     6     8
```

До числа основних функцій для роботи з векторами належить функція sort упорядкування вектора за зростанням його елементів:

```
>> r = [9.4  -2.3  -5.2  7.1  0.8  1.3];
>> R = sort (r)
R =
        -5.2000   -2.3000    0.8000    1.3000    7.1000    9.4000
```

Цю функцію можна використовувати для впорядкування вектора за убуванням:

```
>> R1 = -sort (-r)
R1 =
        9.4000    7.1000    1.3000    0.8000   -2.3000   -5.2000
```

Упорядкувати елементи вектора в порядку зростання їхніх модулів можна за допомогою функції abs:

```
>> R2 = sort (abs(r))
R2 =
        0.8000    1.3000    2.3000    5.2000    7.1000    9.4000
```

Виклик sort із двома вихідними аргументами призводить до утворення масиву індексів відповідності елементів упорядкованого й вихідного масивів:

```
>> [rs,  ind] = sort(r)
rs =
  -5.2000   -2.3000    0.8000    1.3000    7.1000    9.4000
ind =
```

```
    3     2     5     6     4     1
```

Рівність r(ind(k)) = rs(k) для k від 1 до length(k) зв'язує вихідний масив r, упорядкований масив rs і масив індексів ind.

Якщо аргументом функцій max і min є вектор, що складається з комплексних чисел, то результатом є максимальний і мінімальний за модулем елемент. Функція sort також упорядковує комплексний вектор за модулем, а компоненти з рівними модулями розташовуються в порядку зростання фаз.

Інформацію про функції обробки даних дає команда help datafun.

2.1.6 Поелементні операції з векторами

Для поелементного множення векторів рівної довжини використовується операція .* (крапка й зірочка, записані без пробілів). Наприклад:

```
>> v1 = [2  -3  4  1];
>> v2 = [7  5  -6  9];
u = v1 .* v2
u =
   14   -15   -24     9
```

Операція .^ - поелементне зведення в ступінь:

```
>> p = v1 .^ 2
p =
    4     9    16     1
```

Показником ступеня може бути вектор тієї ж довжини, що й призначений для зведення в ступінь. При цьому кожний елемент першого вектора зводиться в ступінь, рівний відповідному елементу другого вектора:

```
>> P = v1 .^ v2
P =
        128.0000 -243.0000    0.0002    1.0000
```

Ділення – операція ./

```
>> d = v1 ./ v2
d =
        0.2857   -0.6000   -0.6667    0.1111
```

Зворотне поелементне ділення (ділення елементів другого вектора на відповідні елементи першого) - операція .\

```
>> dinv = v1 .\ v2
dinv =
        3.5000   -1.6667   -1.5000    9.0000
```

Таким чином, крапка в MATLAB використовується не тільки для вводу десяткових дробів, але й для зазначення поелементних операцій для масивів однакового розміру.

До поелементних належать операції вектора з числом. Додавання вектора й числа не призводить до повідомлення про помилку. MATLAB додає число до кожного елемента вектора (це справедливо і для вирахування):

```
>> v = [4  6  8  10];
>> s = v  + 1.2
s =
        5.2000    7.2000    9.2000   11.2000
>> s1 = 1.2 + v
s1 =
        5.2000    7.2000    9.2000   11.2000
>> r = 1.2 - v
r =
       -2.8000   -4.8000   -6.8000   -8.8000
>> r1 = v - 1.2
r1 =
        2.8000    4.8000    6.8000    8.8000
```

Множити вектор на число можна як справа, так і зліва:

```
>> v = [4  6  8  10];
>> p = v*2
p =
        8    12    16    20
>> p1 = 2*v
p1 =
        8    12    16    20
```

За допомогою операції / можна ділити вектор на число:

```
>> p = v/2
```

```
p =
     2     3     4     5
```

Спроба ділення числа на вектор-рядок (>> p = 2/v) призводить до помилки. Але при діленні числа на вектор-стовпець повідомлення про помилку не видається, тому що відбувається розв'язування системи лінійних рівнянь із прямокутною матрицею, у якій число невідомих перевершує число рівнянь.

Для ділення числа на кожний елемент вектора із записом результату в новий вектор слід використовувати операцію ./

```
>> w = [4  2  6];
>> d = 12 ./w
d =
    3     6     2
```

Всі зазначені операції застосовні як до векторів-рядків, так і до векторів-стовпців.

Для вектора-стовпця u, наприклад із трьома **комплексними** змінними (зокрема з дійсними), сполучений до нього вектор u^* визначається як вектор-рядок з його комплексно-сполучених елементів, а транспонований u^T – просто як вектор-рядок з його елементів. Приклад:

$$u = \begin{bmatrix} 2+3i \\ 1-2i \\ 3+2i \end{bmatrix}, \quad u^* = [2\text{-}3i \quad 1\text{+}2i \quad 3\text{-}2i], \quad u^T = [2\text{+}3i \quad 1\text{-}2i \quad 3\text{+}2i].$$

Аналогічно визначається сполучення й транспонування для вектора-рядка, що приводить до вектора-стовпця. Ясно, що для векторів, що складаються тільки з дійсних чисел, операції сполучення і транспонування збігаються.

Для знаходження сполученого вектора використається апостроф, а для транспонування слід застосовувати крапку з апострофом:

```
>> u = [2 + 3i;   1 - 2i;  3 + 2i];
>> v = u'
v =
        2.0000 - 3.0000i   1.0000 + 2.0000i   3.0000 - 2.0000i
```

```
>> v = u.'
v =
        2.0000 + 3.0000i   1.0000 - 2.0000i   3.0000 + 2.0000i
```

Операції .' і ' над векторами, що складаються з дійсних чисел, призведуть до однакових результатів.

2.1.7 Скалярний добуток векторів

Скалярний добуток (його іноді називають внутрішнім) векторів a й b довжини N, що складаються з дійсних чисел, визначається формулою $a \cdot b = \sum_{k=1}^{N} a_k b_k$. Отже, для обчислення скалярного добутку необхідно просумувати компоненти вектора, отриманого в результаті поелементного множення a на b, тобто треба використати функцію sum і поелементне множення. Наприклад для скалярного множення векторів $a = \begin{bmatrix} 1.2 \\ -3.2 \\ 0.7 \end{bmatrix}$ і $b = \begin{bmatrix} 4.1 \\ 6.5 \\ -2.9 \end{bmatrix}$ потрібна наступна послідовність команд:

```
>> a = [1.2;  -3.2;  0.7];
>> b = [4.1;  6.5;  -2.9];
>> s = sum(a .* b)
s =
        -17.9100
```

Скалярний добуток векторів можна обчислити функцією dot:

```
>> s = dot(a, b);
```

Функція dot дозволяє знайти довжину (тобто модуль) вектора a $|a| = \sqrt{a \cdot a}$:

```
>> d = sqrt(dot(a, a))
d =
        3.4886
```

2.1.8 Векторний добуток векторів

Векторний добуток *a x b* визначено тільки для тривимірного простору, тобто для векторів, що складаються з трьох елементів. Результатом також є вектор із тривимірного простору. Для обчислення векторного добутку в MATLAB служить функція cross:

```
>> a = [1.2;  -3.2;  0.7];
>> b = [4.1;  6.5;  -2.9];
>> c = cross(a, b)
c =
   4.7300
   6.3500
  20.9200
```

Змішаний добуток векторів *a, b, c* визначається за формулою *abc=a·(bxc).* Модуль змішаного добутку дорівнює об'єму паралелепіпеда, побудованого на цих векторах. При $a = \begin{bmatrix} 3.5 \\ 0 \\ 0 \end{bmatrix}$ $b = \begin{bmatrix} 0.5 \\ 2.1 \\ 0 \end{bmatrix}$ $c = \begin{bmatrix} -0.2 \\ -1.9 \\ 2.8 \end{bmatrix}$

```
>> a = [3.5;  0;  0];
>> b = [0.5;  2.1;  0];
>> c = [-0.2;  -1.9;  2.8];
>> V = abs(dot(a, cross(b, c)))
V =
        20.5800
```

2.1.9 Зовнішній добуток векторів

Зовнішнім добутком векторів $a = (a_j)_{j=1,\dots,N}$, $b = (b_k)_{k=1,\dots,M}$ називається матриця $C = (c_{jk})_{j=1,\dots,N,k=1,\dots,M}$ розміру *NxM*, елементи якої обчислюються за формулою $c_{jk} = a_j b_k$. Вектор-стовпець *a* в MATLAB представляється у вигляді двовимірного масиву розміру *N на один*. Вектор-стовпець *b* при транспонуванні переходить у вектор-рядок розміру *один на M*. Вектор-рядок і вектор-стовпець – це матриці, у яких один з ро-

змірів дорівнює одиниці. Фактично $C = ab^T$, де множення відбувається за правилом матричного добутку (оператор "зірочка"):

```
>> a = [1;  2;  3];
>> b = [5;  6;  7];
>> C = a*b'
C =
    5    6    7
   10   12   14
   15   18   21
```

Таким чином, у командне вікно виведена матриця у звичному виді – по рядках. Для перегляду змінних робочого середовища можна використати вікно Workspace або команду whos. Числа, вектори, матриці зберігаються у двовимірних масивах. Числа - у масивах, розмірністю один на один, вектори-стовпці і вектори-рядки - у масивах, в яких один з вимірів дорівнює одиниці, а для матриць виділяються двовимірні масиви відповідних розмірів.

2.1.10 Відображення функції у вигляді таблиці

При порівняно невеликій кількості значень зручно відображати функцію у вигляді таблиці. Наприклад, потрібно вивести в командне вікно таблицю значень функції: $y(x) = \frac{sin^2 x}{1 + cos\, x} + e^{-x} \cdot ln\, x$ у точках 0.2, 0.3, 0.5, 0.8, 1.3, 1.7, 2.5. Для цього створюється вектор-рядок x, що містить координати заданих точок. Потім обчислюється функція *y(x)* від кожного елемента вектора x і отримані значення записуються у вектор-рядок y.

```
>> x = [0.2  0.3  0.5  0.8  1.3  1.7  2.5]
>> x =
      0.2000    0.3000    0.5000    0.8000    1.3000    1.7000    2.5000
>> y = sin(x) .^2 ./ (1 + cos(x)) +exp(-x) .* log(x)
y =
     -1.2978   -0.8473   -0.2980    0.2030    0.8040    1.2258    1.8764
```

Таблиці можна надати більш зручний для читання вигляд, розташувавши значення функції безпосередньо під значеннями аргументу:

```
>> x
x =
0.2000    0.3000    0.5000    0.8000    1.3000    1.7000    2.5000
>> y
y =
-1.2978   -0.8473   -0.2980    0.2030    0.8040    1.2258    1.8764
```

Часто потрібно вивести значення функції в точках відрізка, що відстоять одна від одної на рівну відстань, тобто крок. Для вводу таких векторів використається двокрапка. Наприклад, якщо крок дорівнює 0.2 :

```
>> x = 1:0.2:2
x =
        1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
```

Причому необов'язково треба піклуватися про те, щоб сума передостаннього значення і кроку дорівнювала кінцевому значенню, тому що вектор-рядок заповниться до елемента, що не перевершує зазначене кінцеве значення.

Крок може бути й від'ємним:

```
>> x = 1.9 : -0.2 : 1
x =
        1.9000    1.7000    1.5000    1.3000    1.1000
```

У випадку від'ємного кроку початкове значення повинне бути більшим або дорівнювати кінцевому для одержання непустого вектора-рядка.

Для заповнення вектора-стовпця, наприклад, елементами, що починаються з нуля і закінчуються 0.5 із кроком 0.1, слід заповнити вектор-рядок, а потім використати операцію транспонування:

```
>> x =(0:0.1:0.5)'
x =
    0
   0.1000
   0.2000
   0.3000
   0.4000
```

0.5000

Елементи вектора, що заповнюється за допомогою двокрапки, можуть бути тільки дійсними, тому для транспонування можна набрати знак апострофа замість крапки з апострофом.

Крок, що дорівнює одиниці, допускається не вказувати при автоматичному заповненні:

```
>> x = 1:5
x =
     1     2     3     4     5
```

Якщо результуючі вектори не вміщуються на екрані цілком, то вони виводяться блоками. Наприклад, потрібно обчислити значення функції $y(x) = e^{-x}\, sin10x$ на відрізку [0, 1] із кроком 0.05:

```
>> x = 0:0.05:1;
>> y = exp(-x) .* sin(10*x);
```

Результат, виведений на екран, не схожий на таблицю:

```
>> x
x =
Columns 1 through 8
0    0.0500    0.1000    0.1500    0.2000    0.2500    0.3000    0.3500
Columns 9 through 16
0.4000    0.4500    0.5000    0.5500    0.6000    0.6500    0.7000    0.7500
Columns 17 through 21
0.8000    0.8500    0.9000    0.9500    1.0000
>>y
y =
Columns 1 through 8
0    0.4560    0.7614    0.8586    0.7445    0.4661    0.1045   -0.2472
Columns 9 through 16
-0.5073   -0.6233   -0.5816   -0.4071   -0.1533    0.1123    0.3262    0.4431
Columns 17 through 21
 0.4445    0.3413    0.1676   -0.0291   -0.2001
```

Тому що x та y зберігаються у двовимірних масивах один на двадцять один, то вони виводяться стовпцями, кожний з яких складається з

одного елемента. Кількість елементів, виведених в один рядок, визначається поточними розмірами вікна і форматом виводу даних.

Одним зі способів одержання таблиці є формування матриці із двох стовпців, перший - значення абсцис, а другий - ординат:

```
>> [x'  y']
ans =
         0         0
    0.0500    0.4560
    0.1000    0.7614
    0.1500    0.8586
    0.2000    0.7445
      . . .     . . .
    0.9000    0.1676
    0.9500   -0.0291
    1.0000   -0.2001
```

Часто більш зручним виявляється графічне подання функцій.

2.1.11 Графіки функцій однієї змінної у лінійному масштабі

MATLAB дозволяє будувати графіки функцій у лінійному, логарифмічному та напівлогарифмічному масштабах. Причому в одному вікні можна побудувати графіки кількох функцій.

Основним засобом двовимірної графіки в лінійному масштабі є функція plot. Типовий варіант її використання виглядає так:

plot(x, y)

Тут x та y - вектори однакової довжини, що задають координати точок, виведених на графік. Таким чином, для побудови графіка функції в робочому середовищі MATLAB повинні бути визначені два вектори однакової розмірності. Вектор x містить значення аргументів, а y - значення функції від цих аргументів. За умовчанням точки з координатами (x(i), y(i)) з'єднуються прямими лініями синіх кольорів. При цьому осі автоматично масштабуються для оптимального розташування графіка у вікні.

На рис.1 показано приклад графіка функції $y(x) = e^{-x} sin10x$, визначеної на відрізку [0, 1]. Набір команд виглядає так:

```
>> x = 0:0.05:1;
>> y = exp(-x).*sin(10*x);
>> plot(x, y)
```

Після виконання команд на екрані з'являється вікно із графіком функції (рис.3).

Для побудови графіків зручно розташувати на екрані командне вікно MATLAB і вікно із графіком так, щоб вони не перекривалися. Наприклад, використовуючи кнопку Dock Figure (праворуч у рядку меню вікна), можна вмонтувати графічне вікно в робоче середовище. Для одержання окремого графічного вікна треба натиснути на кнопку Undock Figure.

Побудований графік функції не повинен мати зломів, тому що сама функція гладка. Для точної побудови графіка необхідно обчислити функцію у великій кількості точок на відрізку [0, 1], тобто задати менший крок при вводі вектора x. Наступні команди:

```
>> x = 0:0.01:1;
>> y = exp(-x).*sin(10*x);
>> plot(x, y)
```

призводять до побудови графіка функції у вигляді плавної кривої (рис.4).

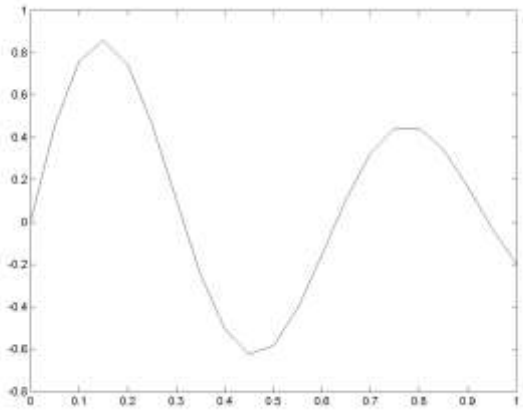


Рис.3 Приклад графіка функції

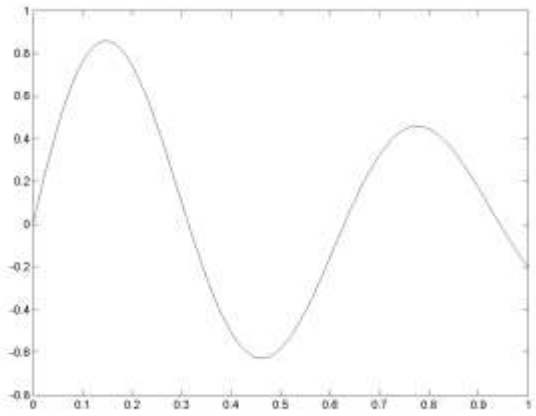


Рис.4 Приклад згладженого графіка функції

Для порівняння декількох функцій їхні графіки відображаються на одних осях. Наприклад, на рис.5 показано графіки функцій $f(x) = sin\frac{1}{x^2}$, $g(x) = sin\frac{1.2}{x^2}$ на відрізку [-1, -0.3]. Графіки отримані за допомогою наступної послідовності команд:

```
>> x = -1:0.005 : -0.3;
>> f = sin(x.^-2);
>> g = sin(1.2*x.^-2);
>> plot(x, f, x, g)
```

Якщо функції визначені на різних відрізках, то при побудові графіків MATLAB вибирає максимальний відрізок, що містить інші. Важливо тільки в кожній парі векторів абсцис й ординат, кількість яких може бути довільною, вказати у plot відповідні один одному вектори.

Якщо значення функцій сильно відрізняються одне від одного, то графік функції з невеликими значеннями може зливатися з віссю абсцис. Для усунення цього ефекту використовується функція plotyy, що виводить графіки у вікно із двома вертикальними осями, які мають придатний масштаб. Наприклад, на рис.6 показано графіки двох функцій $f(x) = x^{-3}$ й $F(x) = 1000 \cdot (x + 0.5)^{-4}$. Графіки отримані за допомогою наступної послідовності команд:

```
>> x = 0.5 : 0.01 : 3;
>> f = x.^-3;
>> F = 1000*(x+0.5).^-4;
>> plotyy(x, f, x, F)
```

Колір графіка збігається з кольором відповідної йому вісі ординат.

Функцію plot припустимо використовувати з одним аргументом, що призводить до побудови "графіка вектора", тобто залежності значень елементів вектора від їхніх номерів.

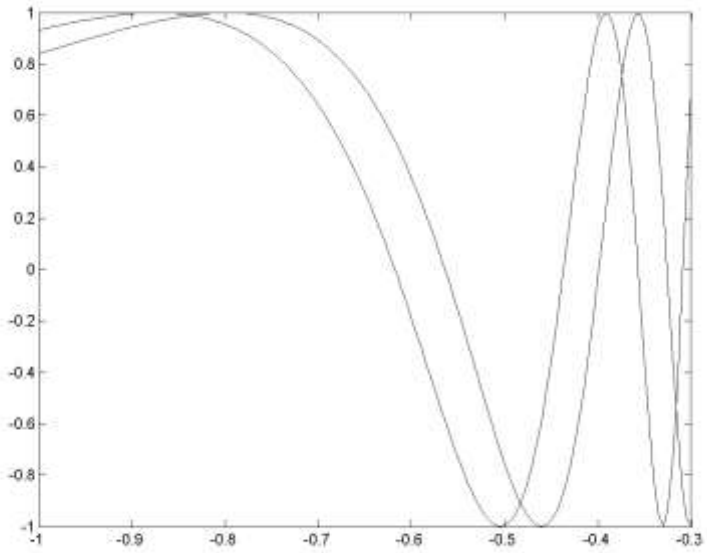


Рис. 5 Приклад двох графіків на одних осях

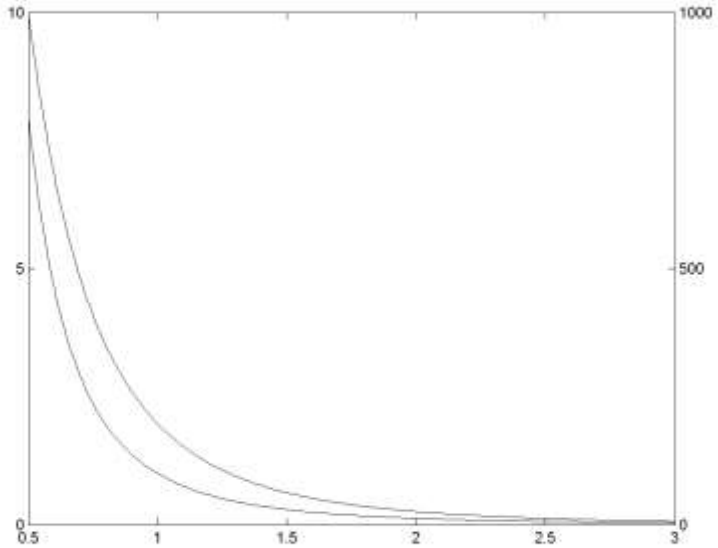


Рис. 6 Приклад порівняння функцій за допомогою plotyy

2.1.12 Графіки функцій однієї змінної в логарифмічному масштабі

Для побудови графіків у логарифмічному й напівлогарифмічному масштабах служать функції:

- loglog – логарифмічний масштаб по обох осях,
- semilogx – логарифмічний масштаб тільки по осі абсцис,
- semilogy – логарифмічний масштаб тільки по осі ординат.

Аргументи цих функцій задаються у вигляді пари векторів значень абсцис й ординат аналогічно функції plot. На рис.7 показано графіки функ-

ції $f(x) = ln\,0.5x$ і $g(x) = sin\,ln\,x$ на відрізку [0.1, 5] у логарифмічному масштабі по осі x. Графіки отримані за допомогою послідовності команд:

```
>> x = 0.1 : 0.01 : 10;
>> f = log(0.5*x);
>> g = sin(log(x));
>> semilogx (x, f, x, g)
```

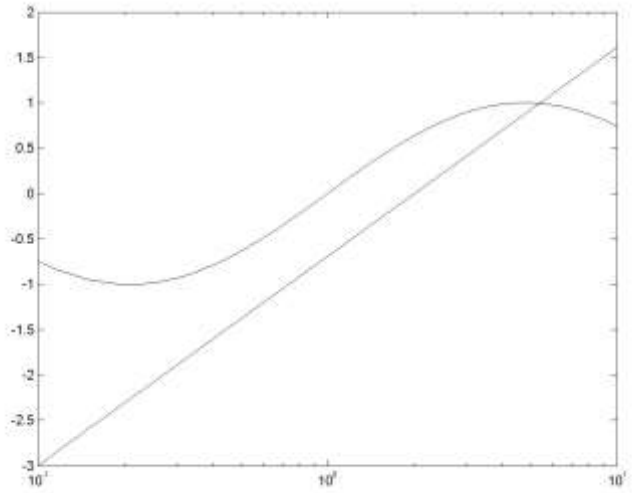


Рис. 7 Графіки в напівлогарифмічній шкалі

Функції loglog та semilogy викликаються аналогічним чином.

2.2 ЗНАХОДЖЕННЯ НУЛІВ І ЕКСТРЕМУМІВ ФУНКЦІЙ

Для пошуку значень параметрів, при яких функція дорівнює нулю, можна використовувати функцію fzero, Алгоритм роботи fzero заснований на пошуку інтервалу, у якому функція міняє знак. При виклику fzero другим параметром варто вказати інтервал, у якому варто шукати корінь. Наприклад, для файл-функції

```
function y=myf(x)
y=sin(x)-x.^2.*cos(x);
```

на інтервалі [-3 -1] буде знайдене нульове значення функції

```
>> x2=fzero('myf', [-3 -1])
Zero found in the interval: [-3 -1].
```

x2 = -1.8539

В межах інтервалу, що вказується, функція повинна приймати значення різних знаків, інакше видається повідомлення про помилку.

Убудована математична функція може бути як досліджувана. Наприклад:

```
>> fzero('sin', [2  4])
Zero found in the interval: [2  4].
ans = 3.1416
```

Крім того, функцію можна задати за допомогою покажчика на неї:

```
>> x2=fzero(@myf, [-3  -1])
x2 = -1.8539
```

Припустимо використання inline-функції:

```
>> fun=inline('sin(x)-x.^2.*cos(x)')
fun =  Inline function:
    fun(x) = sin(x)-x.^2.*cos(x)
>>x1=fzero(fun, -5)
x1 =  -4.7566
>> fun(x1)
ans =  2.6645 e-015
```

Припустимо використання анонімної функції:

```
>> fun=@(x) sin(x)-x.^2.*cos(x)
fun =  @(x) sin(x)-x.^2.*cos(x)
>> x1=fzero(fun,  -5)
x1 =  -4.7566
```

Важливою особливістю функції fzero є те, що вона обчислює ті коріння, у яких функція міняє знак, а не тільки торкається осі абсцис. Таким чином, знайти корінь рівнянь $x^2 = 0$ за допомогою fzero не вдасться:

```
>> fun=inline('x.^2');
>> x=fzero(fun, -0.1)
Exiting fzero: aborting search for an interval containing a sign change
```

because Na or Inf function value encountered during search.

(Function value at 1.37296e+154 is Inf.)

Check function or try again with a different starting value.

x = Na

У даному прикладі fzero намагалася знайти інтервал, в межах якого значення функції myf мають різні знаки, що гарантувало б існування кореня. Оскільки такий інтервал знайдений не був, то було видане повідомлення про помилку.

Алгоритм функції fzero за замовчуванням знаходить корінь рівняння з точністю eps, тобто ± 2 в шістнадцятій цифрі після десяткової крапки. Звертання до fzero із двома вихідними параметрами дозволяє не тільки приблизно знайти корінь, але й одержати значення функції в заданій точці.

>> [x2, f] = fzero(@myf, -2)

x2 = -1.8539

f = -2.2204 e-016

Звертання до функції fzero із трьома вихідними аргументами дозволяє вибрати подальші дії залежно від значення третього аргументу (у прикладі – це flag):

>> [x1, f1, flag] = fzero(fun, [-3 -1]);

Позитивне значення flag свідчить про успішне завершення процесу. Негативне значення свідчить про те, що не вдалося визначити інтервал зі зміною знака функції, або в процесі обчислень вийшло комплексне значення, нескінченність, або виконана операція з невизначеним результатом, наприклад, ділення нуля на нуль.

Пошук локального мінімуму функції однієї змінної на деякому інтервалі здійснюється за допомогою fminbnd. Спочатку необхідно створити файл-функцію, використовувати inline або анонімну функцію. Наприклад, при пошуку локальних мінімумів функції $e^{x^2} + \sin 3\pi x$ на відрізку [-1.5 1.5]:

```
function y=ftest(x)
y=exp(x.^2)+sin(3*pi.*x);
>> x2=fminbnd(@ftest, -0.5, 0)
x2 = -0.1629
```

Припустимо задати інтервал пошуку, що містить всі чотири крапки локальних мінімумів:

```
>> xx=fminbnd(@ftest, -1.5, 1.5)
xx =0.4861
```

Для одночасного обчислення значення функції в точці мінімуму варто викликати fminbnd із двома аргументами:

```
>> [x2, f]=fminbnd(@ftest, -0.5, 0)
x2 = -0.1629
f = 0.0275
```

Функція fminbnd може бути викликана із трьома вихідними аргументами:

```
>> [x2, f, flag]=fminbnd(@ftest, -0.5, 0);
```

Аргумент flag може приймати три значення: позитивне – при знаходженні локального мінімуму, нульове – при досягненні максимальної кількості викликів досліджуваної функції й негативне – у випадку расходимости обчислювального процесу.

Знайти мінімум можна і за допомогою fminsearch, що вимагає зазначення початкового наближення для шуканої точки. Наприклад:

```
>> x2=fminsearch(@ftest, -0.5)
x2 = -0.1629
```

Для визначення точок локального максимуму немає спеціальної функції, оскільки досить шукати мінімум функції зі зворотним знаком.

2.3 ЗАВДАННЯ ДЛЯ САМОКОНТРОЛЮ

Вихідні дані - дві функції однієї змінної (задати самостійно). Необхідно:

1) Використовуючи правила роботи з векторами побудувати графіки заданих функцій однієї змінної на відрізках, найбільш характерних для відображення сутності функцій. Графіки вивести різними способами:
   - в окремі графічні вікна,
   - в одне вікно на одні осі,
   - в одне вікно на окремі осі.
2) Вивести результати в табличному вигляді.
3) Зберегти результати у файлі.
4) Виконати контрольні прорахунки.
5) Виконати одне з наступних завдань до роботи з векторами. Спосіб розв'язання завдань повинен носити універсальний характер і бути придатним для довільних векторів (як дані для контролю розв'язання завдання треба використовувати результати роботи із заданими функціями):
   - Виділити в нові вектори елементи вектора з більшими й меншими значеннями.
   - Виділити в нові вектори елементи вектора з парними й непарними номерами.
   - Знайти суми позитивних та негативних елементів вектора.
   - Замінити елементи вектора, що відрізняються від середнього геометричного його елементів більш ніж на 10%, на середнє геометричне.
   - Замінити всі мінімальні елементи вектора максимальним значенням його елементів.
   - Визначити кількість позитивних, негативних і нульових елементів вектора.
   - Знайти число елементів вектора, що відрізняються від середнього арифметичного менше, ніж на 20%.
   - Замінити елементи вектора, що відрізняються від нульового рівня не більше ніж на 5%, на нуль.

- Замінити елементи вектора, що відрізняються від максимального позитивного значення не більше ніж на 5%, на максимальне.
- Замінити елементи вектора, що відрізняються від мінімального негативного значення не більше ніж на 5%, на мінімальне.
- Визначити номер елемента вектора з найбільшим відхиленням від середнього арифметичного всіх елементів вектора.
- Обчислити суми всіх елементів вектора з парними й непарними індексами.
- Обчислити добуток елементів вектора, не переважаючих середнє арифметичне значення його елементів.
- Визначити кількість позитивних елементів вектора, розташованих між його максимальним і мінімальним елементами.
- Замінити позитивні елементи вектора сумою всіх його негативних елементів.

2.4 ЗАПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

- Яке призначення функцій ndims, size та length ?
- Як використовуються тригонометричні функції при обробці векторів ?
- У яких форматах можливе зберігання елементів масивів ?
- Як виконуються операції зчеплення векторів ?
- Як виконуються операції редагування векторів?
- Як і коли використовується індексація за допомогою вектора ?
- Яке призначення функцій prod, sum, min, max та sort ?
- Як виконуються операції .* при обробці векторів ?
- Як виконуються операції .^ при обробці векторів ?
- Як виконуються операції ./ и.\ при обробці векторів ?
- Як виконуються поелементні операції вектора із числом ?
- Як формується сполучений і транспонований вектор для вектора-стовпця з комплексними змінними ?

- Як формується сполучений і транспонований вектор для вектора-рядка з комплексними змінними ?
- Як виконуються операції .' і ' над векторами, що складаються з дійсних чисел ?
- Як виконується скалярний добуток векторів ?
- Як виконується векторний добуток векторів ?
- Як виконується зовнішній добуток векторів ?
- Як відобразити функцію у вигляді таблиці ?
- Як використовується операція транспонування при відображенні функції у вигляді таблиці ?
- Які функції використовуються для побудови графіків функцій у лінійному масштабі?
- Які функції використовуються для побудови графіків функцій у логарифмічному та напівлогарифмічному масштабах ?
- Як можна відобразити графіки декількох функцій на одних осях ?

3. РОБОТА З МАТРИЦЯМИ

Мета: вивчення методів обробки даних, представлених у формі матриць.

3.1 МЕТОДИ ОБРОБКИ МАТРИЦЬ

3.1.1 Введення матриць

Невеликі за розміром матриці зручно вводити безпосередньо з командного рядка. Наприклад, для зберігання матриці A розміром два на три

$$A = \begin{pmatrix} 3 & 1 & -1 \\ 2 & 4 & 3 \end{pmatrix}$$

використовується двовимірний масив A , який можна розглядати як вектор-стовпець із двох елементів, кожен з яких є вектором-рядком довжиною три. Тому рядки при наборі відокремлюються крапкою з комою:

```
>> A = [3   1   -1;  2   4   3]
A =
    3    1   -1
    2    4    3
```

Інший спосіб введення з командного рядка починається з набору

```
>> B = [4  3  -1
```

і натискання клавіші Enter. Але при цьому MATLAB нічого не обчислює, а переводить курсор на наступний рядок без символу >>, де необхідно продовжити введення матриці, натискаючи наприкінці кожного введення рядка клавішу Enter. Останній рядок слід завершити квадратною закриваючою дужкою:

```
 2  7  0
-5  1  2]
```

У результаті виходить:

```
B =
    4    3   -1
    2    7    0
   -5    1    2
```

Ще один спосіб введення матриць полягає в тому, що матрицю можна трактувати як вектор-рядок, кожен елемент якого є вектором-стовпцем. Наприклад, матрицю два на три

$$C = \begin{pmatrix} 3 & -1 & 7 \\ 4 & 2 & 0 \end{pmatrix}$$

можна ввести за допомогою команди:

```
>> C = [[3; 4]  [-1; 2]  [7; 0]]
C =
    3   -1    7
    4    2    0
```

Інформація у вікні Workspace підтверджує наявність трьох матриць - двох прямокутних і однієї квадратної.

Елементи матриці доступні за допомогою двох індексів - номерів рядка і стовпця, котрі знаходяться у круглих дужках, наприклад

```
>> C(2, 3)
ans =
  0
```

Елементи матриць можуть входити до складу виразів:

```
>> C(1, 1) + C(2, 2) + C(2, 3)
ans =
  5
```

В якості індексів можуть виступати вектори, що містять номери потрібних рядків і стовпців. Наприклад, для виділення елементів першого і другого рядків другого та третього стовпців матриці B досить ввести команди:

```
>> i = [1  2];
>> j = [2  3];
>> B1 = B(i, j)
B1 =
    3   -1
    7    0
```

Розташування елементів матриці в пам'яті комп'ютера визначає наступний спосіб звертання до них. Оскільки елементи q_{ij} матриці Q розміру m на n містяться в пам'яті у послідовності

$q_{11}, q_{21}, \ldots, q_{m1}, q_{12}, q_{22}, \ldots, q_{m2}, \ldots, q_{1n}, q_{2n}, \ldots, q_{mn}$

то для доступу до них можна використовувати один індекс, що задає порядковий номер елемента матриці у векторі. Наприклад, елементи матриці C записані в такому порядку:

C(1, 1), C(2, 1), C(1, 2), C(2, 2), C(1, 3), C(2, 3).

Тому >> C(5) дає ans = 7

3.1.2 Логічне індексування

Звертатися до всіх елементів матриці, що задовольняє деякій умові, заданій логічним виразом, можна за допомогою логічного індексування. Наприклад, потрібно вибрати з матриці B всі негативні елементи і записати їх у вектор f. Спочатку в змінну ind записується результат порівняння матриці й числа нуль:

```
>> ind = B < 0
ind =
    0    0    1
    0    0    0
    1    0    0
```

Таким чином, утворився логічний масив ind того ж розміру, що й B, що складається з нулів та одиниць, причому одиниці відповідають від'ємним елементам масиву B. Вказівка логічного масиву ind як єдиний індекс вихідного масиву B дозволяє вирішити задачу:

```
>> f = B(ind)
f =
   -5
   -1
```

Можна обійтися без допоміжного масиву ind, записавши >> f=B(B<0).

Якщо потрібно присвоїти нове значення елементам масиву, що задовольняє певній умові, то вираз B (B < 0) повинен увійти в ліву частину оператора присвоювання.

Особливістю цього способу є те, що масив, який застосовується для індексування, повинен бути логічним. Це означає, що недостатньо створити числовий масив і вказати його як індекс масиву. Така дія призведе до помилки:

```
>> ind1 = [0 0 1; 0 0 0; 1 0 0];
>> B(ind1)
```

Числовий масив перед використанням слід перетворити в логічний за допомогою спеціальної функції:

```
>> ind2 = logical(ind1);
B(ind2)
ans =
   -5
   -1
```

Довідатися інформацію про масиви можна у вікні Workspace.

Для пошуку індексів елементів, що задовольняють певній умові, служить функція find. Наприклад, потрібно знайти номери всіх елементів вектора x , рівних максимальному значенню (знак == позначає логічну рівність):

```
>> x = [1  2  5  3  4  5  1  5];
km = find(x == max(x))
km =
    3     6     8
```

У матриці функція find працює аналогічно. Наприклад, потрібно знайти індекси всіх невід'ємних елементів матриці *B*. Функція find викликається із двома вихідними аргументами – векторами, у які потрібно записати значення рядкових і стовпцевих індексів шуканих елементів матриці

```
>> [i, j] = find(B <= 0)
i =
    3
    1
    2
j =
    1
    3
    3
```

Дійсно, елементи b_{31}, b_{13}, b_{23} менше або дорівнюють нулю. У цьому прикладі до функції find можна звернутися з одним вихідним аргументом:

```
>> k = find(B <= 0)
k =
    3
    7
```

8

У цьому випадку вектор k містить номера необхідних елементів матриці у відповідності зі схемою їхнього зберігання в пам'яті.

3.1.3 Використання матричних операцій

При додаванні або відніманні матриці повинні бути одного розміру, а при перемножуванні число стовпців першої матриці повинно дорівнювати числу рядків другої матриці. Приклади додавання та віднімання матриць:

```
>> S = A + C
S =
    6    0    6
    6    6    3
```

```
>> R = C - A
R =
    0   -2    8
    2   -2   -3
```

Для запису операції множення матриць використовується символ * :

```
>> P = C * B
P =
  -25    9   11
   20   26   -4
```

Множення матриці на число (або числа на матрицю):

```
>> P = A * 3
P =
    9    3   -3
    6   12    9
```

```
>> P = 3 * A
P =
    9    3   -3
    6   12    9
```

Символ .' використовується при транспонуванні матриці, а символ ' означає комплексне сполучення. Для матриць, що складаються з дійсних чисел, ці операції призводять до однакових результатів:

```
>> B'
ans =
  4   2  -5
  3   7   1
 -1   0   2
```

```
>> B .'
ans =
   4   2  -5
   3   7   1
  -1   0   2
```

Сполучення і транспонування матриць, що містять комплексні числа, приведуть до різних результатів:

```
>> K = [1 - 1i,  2 + 3i;  3 - 5i,  1 - 9i]
K =
   1.0000 - 1.0000i   2.0000 + 3.0000i
   3.0000 - 5.0000i   1.0000 - 9.0000i
>> K'
```

```
ans =
  1.0000 + 1.0000i   3.0000 + 5.0000i
  2.0000 - 3.0000i   1.0000 + 9.0000i
>> K .'
ans =
  1.0000 - 1.0000i   3.0000 - 5.0000i
  2.0000 + 3.0000i   1.0000 - 9.0000i
```

Оператор ^ використовується при зведенні квадратної матриці в ступінь:

```
>> B2 = B^2
B2 =
   27    32    -6
   22    55    -2
  -28    -6     9
```

Значення виразу $(A+C)B^3(A-C)^T$ обчислюється такою командою:

```
>> (A + C) * B^3 * (A- C)'
ans =   1848        1914
        10290       3612
```

Відповідно до пріоритету операцій спочатку виконується транспонування, потім зведення в ступінь, потім множення, а в останню чергу - додавання і віднімання.

3.1.4 Перемножування матриці й вектора

Оскільки вектор-стовпець або вектор-рядок в MATLAB є матрицями, у яких один з розмірів дорівнює одиниці, то для множення матриці на вектор застосовні всі операції множення векторів. Наприклад, вираз

$$\begin{bmatrix}1 & 3 & -2\end{bmatrix}\begin{pmatrix}2 & 0 & 1\\ -4 & 8 & -1\\ 0 & 9 & 2\end{pmatrix}\begin{bmatrix}-8\\ 3\\ 4\end{bmatrix}$$

обчислюється так:

```
>> a = [1  3  -2];
>> B = [2  0  1;  -4  8  -1;  0  9  2];
>> c = [-8; 3; 4];
>> a*B*c
ans =
   74
```

У математиці не визначена операція ділення матриць і векторів. Символ / використовується для розв'язання систем лінійних рівнянь.

3.1.5 Конструювання блокових матриць

Блокові матриці складаються з непересічних підматриць (блоків). Відповідні розміри блоків повинні збігатися. Наприклад, послідовність команд

```
>> A = [-1  4;  -1  4]
>> B = [2   0;   0  5]
>> C = [3  -3;  -3  3]
>> D = [8   9;  1  10]
>> K = [A  B;  C  D]
K =
   -1    4    2    0
   -1    4    0    5
    3   -3    8    9
   -3    3    1   10
```

дозволяє одержати блокову матрицю $K = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$

із чотирьох матриць:

$$A = \begin{pmatrix} -1 & 4 \\ -1 & 4 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 0 \\ 0 & 5 \end{pmatrix} \qquad C = \begin{pmatrix} 3 & -3 \\ -3 & 3 \end{pmatrix} \qquad D = \begin{pmatrix} 8 & 9 \\ 1 & 10 \end{pmatrix}$$

Якщо розглядати блокову матрицю, яка складається із двох стовпців, у першому – матриці A і C, а в другому - B і D, то команда для одержання блокової матриці виглядатиме так:

```
>> K = [[A; C]  [B; D]]
```

Функція blkdiag дозволяє конструювати блочно-діагональні матриці. Наприклад,

```
>> R = [1  2;  3  4];
>> Q = [5  6  7;  8  9  10;  11  12  13];
>> T = [-3  5;  6  7];
Z = blkdiag(R, Q, T)
Z =
    1    2    0    0    0    0    0
    3    4    0    0    0    0    0
    0    0    5    6    7    0    0
    0    0    8    9   10    0    0
```

```
    0     0    11   12    13     0      0
    0     0     0    0     0    -3      5
    0     0     0    0     0     6      7
```

Блоки, використовувані функцією blkdiag, не обов'язково повинні бути квадратними й одного розміру.

Зворотною задачею до конструювання блокових матриць є виділення блоків. Блоки матриць виділяються індексацією за допомогою двокрапки. Наприклад, наступна команда

```
>> P1 = Z(2:3,  2:3)
P1 =
    4     0
    0     5
```

виділяє блок *P1* з матриці *Z*.

Для виділення з матриці стовпця або рядка треба в якості одного з індексів використати номер стовпця або рядка матриці, а інший індекс замінити на двокрапку без вказівки границь. Наприклад, виділити другий рядок матриці *Z* у вектор *p* можна командою:

```
>> p = Z(2, :)
p =
    3     4     0     0     0     0     0
```

При виділенні блоку до кінця матриці можна не вказувати її розміри, а використовувати end:

```
>> p = Z(4, 3:end)
p =
    8     9    10     0     0
```

3.1.6 Видалення рядків і стовпців

Для видалення рядка слід присвоїти йому порожній масив (порожній масив задається парними квадратними дужками). Наприклад, команда видалення першого рядка виглядатиме так:

```
>> M = [2  0  3
        1   1   4
        6   1   3];
>> M(1, :) = [ ];
>> M
M =
    1     1     4
```

```
    6     1     3
```

Відповідну зміну розмірів масивів можна подивитися у вікні Workspace або перевірити за допомогою size.

Аналогічним способом видаляються й стовпці. Для видалення декількох стовпців або рядків, що йдуть підряд, їм потрібно привласнити порожній масив. Наприклад, видалення другого й третього стовпців у матриці M :

```
>> M(: ,  2:3) = [  ]
M =
    1
    6
```

3.1.7 Заповнення матриць за допомогою індексації

Якщо матриця має просту структуру, то замість вводу її можна згенерувати. Наприклад, необхідно згенерувати матрицю:

$$T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \end{pmatrix}$$

Матрицю T можна згенерувати в три етапи:

1. Створення нульового масиву T розміром п'ять на п'ять.
2. Заповнення першого рядка одиницями.
3. Заповнення частини останнього рядка мінус одиницями до останнього елемента.

Відповідні команди:

```
>> A (1:5,  1:5) = 0;
>> A(1, : ) = 1;
>> A(end, 3 : end) = -1;
```

3.1.8 Створення матриць спеціального виду

Убудованою функцією zeros, аргументами якої є число рядків і стовпців, прямокутна матриця заповнюється нулями. Наприклад:

```
>> A = zeros(3, 6);
```

Один аргумент функції zeros приводить до утворення квадратної матриці заданого розміру:

```
>> A = zeros(3);
```

Одинична матриця ініціалізується за допомогою функції eye:

```
>> I = eye(4)
I =
    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1
```

Функція eye із двома аргументами створює прямокутну матрицю, у якої на головній діагоналі стоять одиниці, а інші елементи дорівнюють нулю:

```
>> I = eye(4, 8)
I =
    1    0    0    0    0    0    0    0
    0    1    0    0    0    0    0    0
    0    0    1    0    0    0    0    0
    0    0    0    1    0    0    0    0
```

Матриця, що складається з одиниць, утвориться функцією ones:

```
>> E = ones(3, 8);
```

Функція ones з одним аргументом приводить до створення квадратної матриці, що складається з одиниць.

Результатом функції rand є матриця чисел, розподілених випадково між нулем та одиницею, а функції randn - матриця чисел, розподілених за нормальним законом:

```
>> R = rand(3, 5)
R =
    0.9501    0.4860    0.4565    0.4447    0.9218
    0.2311    0.8913    0.0185    0.6154    0.7382
    0.6068    0.7621    0.8214    0.7919    0.1763
>> RN = randn(3, 5)
RN =
   -0.4326    0.2877    1.1892    0.1746   -0.5883
   -1.6656   -1.1465   -0.0376   -0.1867    2.1832
    0.1253    1.1909    0.3273    0.7258   -0.1364
```

Звертання до функцій rand і randn з одним вхідним аргументом призводить до формування квадратних матриць. Ці функції можна використовувати і для формування векторів. Наприклад:

```
>> r = rand(1, 6)
r =
   0.4057    0.9355    0.9169    0.4103    0.8936    0.0579
```

Для формування діагональної матриці, у якої всі позадиагональні елементи дорівнюють нулю, використовується функція diag. Наприклад:

```
>> d = [1; 2; 3; 4];
>> D = diag(d)
D =
    1    0    0    0
    0    2    0    0
    0    0    3    0
    0    0    0    4
```

Для заповнення не головної, а побічної діагоналі функція diag викликається із двома аргументами. У цьому випадку другий аргумент означає, наскільки побічна діагональ відстоїть від головної, а його знак указує на напрямок, плюс - угору, мінус - униз від головної діагоналі:

```
>> d = [1;  2];
>> D = diag(d,  2)
D =
    0    0    1    0
    0    0    0    2
    0    0    0    0
    0    0    0    0
>> D = diag(d,  -2)
D =
    0    0    0    0
    0    0    0    0
    1    0    0    0
    0    2    0    0
```

Функція diag служить і для виділення діагоналі матриці у вектор:

```
>> A = [10   1   2;   1   20   3;   2    3    30]
A =
   10     1     2
    1    20     3
    2     3    30
>> d = diag(A)
d =
```

```
    10
    20
    30
```

Функція true(M,N) служить для побудови матриці розміром Mx з логічних одиниць (істина).

Функція false(M,N) служить для побудови матриці розміром Mx з логічних нулів.

Функція magic(M) створює “магічний квадрат” розміром Mx.

3.1.9 Поелементні операції та убудовані функції

Оскільки вектори та матриці зберігаються у двовимірних масивах, то математичні функції і поелементні операції до матриць застосовуються так само, як і для векторів, але з деякими особливостями.

Елемент однієї матриці множиться на відповідний елемент іншої за допомогою оператора .* . Наприклад, задані дві матриці, що підлягають перемножуванню:

$$A = \begin{pmatrix} 2 & 5 & -1 \\ 3 & 4 & 9 \end{pmatrix} \quad B = \begin{pmatrix} -1 & 2 & 8 \\ 7 & -3 & -5 \end{pmatrix}$$

Послідовність команд наступна

```
>> A = [2  5  -1;  3   4   9];
>> B = [-1  2   8;  7  -3  -5];
>> C = A .* B
C =
    -2    10     -8
    21   -12   -45
```

Для ділення елементів першої матриці на відповідні елементи другої використовується ./ , а для ділення елементів другої матриці на відповідні елементи першої служить .\ , наприклад

```
>> R1 = A  ./  B
R1 =
   -2.0000    2.5000    -0.1250
   0.4286     -1.3333   -1.8000
>> R2 = A  .\  B
R2 =
   -0.5000    0.4000    -8.0000
    2.3333    -0.7500    -0.5556
```

Операція .^ використовується для поелементного зведення в ступінь

```
>> P = A  .^ 2
P =
4    25     1
9    16    81
```

Показник ступеня може бути матрицею того ж розміру, що й матриця, яка зводиться в ступінь. При цьому елементи першої матриці зводяться в ступені, рівні відповідним елементам другої матриці:

```
>> PB = A  .^ B
PB =
   1.0e+003 *
   0.0005    0.0250    0.0010
   2.1870    0.0000    0.0000
```

Спочатку виводиться загальний множник 1.0e+003 для всіх елементів результуючої матриці, тобто відповідь виглядатиме так:

$$PB = 10^3 \cdot \begin{pmatrix} 0.0005 & 0.0250 & 0.0010 \\ 2.1870 & 0.0000 & 0.0000 \end{pmatrix} \equiv \begin{pmatrix} 0.5 & 25 & 1 \\ 2187 & 0 & 0 \end{pmatrix}$$

При зведенні $4^{-3}=0.0156$ і $9^{-5}=0.000016935$ вийшли нулі, тобто відбулася втрата точності через те, що використовувався формат short. Якщо все-таки необхідно отримати більш точний результат зведення в ступінь, то необхідно задати формат long і потім повторно вивести PB:

```
>> format  long
>> PB
PB =
1.0e+003 *
  0.00050000000000   0.02500000000000   0.00100000000000
  2.18700000000000   0.00001562500000   0.00000001693509
```

Таким чином, повторного обчислення елементів матриці PB не треба було, тому що незалежно від установленого формату виводу всі обчислення проводяться з подвійною точністю.

3.1.10 Обчислення математичних функцій від елементів матриць

Запис *C=cos(A)* призводить до обчислення косинусів від елементів масиву *A* і запису їх у масив *C* того ж розміру, що й *A*. Наприклад, косинуси від елементів матриці

$$A = \begin{pmatrix} \pi/2 & -\pi/2 & 0 \\ \pi & -\pi & 2\pi \\ 0 & 2\pi & \pi/3 \end{pmatrix}$$

обчислюються за допомогою наступних команд:

```
>> A = [pi/2   -pi/2    0;   pi    -pi    2*pi;   0    2*pi    pi/3];
>> C = cos(A)
C =
    0.0000    0.0000    1.0000
   -1.0000   -1.0000    1.0000
    1.0000    1.0000    0.5000
```

Аналогічно обчислюються й інші математичні функції. Але використання функцій обробки даних, таких як знаходження максимуму, мінімуму, суми й ін., для матриць трохи відрізняється від їхнього застосування при роботі з векторами.

Функція sum обчислить вектор-рядок, довжина якого дорівнює числу стовпців матриці, а кожен елемент є сумою відповідного стовпця матриці, наприклад:

```
>> M = [1  -2   -4
        3   -6    4
        2   -2    0];
>> s = sum(M)
s =
    6   -10     0
```

Функція sum за умовчанням обчислює суму по стовпцях, змінюючи перший індекс масиву при фіксованому другому. Для того щоб підсумувати по рядках, необхідно викликати sum із двома аргументами, указавши місце індексу, за яким необхідно підсумовувати. Наприклад

```
>> s2 = sum(M, 2)
s2 =
   -5
    1
```

```
     0
```

Отже, sum(M) і sum(M, 1) призводять до однакових результатів.

Аналогічно sum працює й функція prod:

```
>> p = prod(M)
p =
    6   -24     0
```

```
>> p2 = prod(M,  2)
p2 =
     8
   -72
     0
```

Функція sort упорядковує елементи кожного зі стовпців матриці в порядку зростання. Виклик sort із другим аргументом, рівним 2, призводить до впорядкування елементів рядків:

```
>> MC = sort(M)
MC =
    1    -6    -4
    2    -2     0
    3    -2     4
```

```
>> MR = sort(M,  2)
MC =
   -4    -2     1
   -6     3     4
   -2     0     2
```

Так само як і для векторів, функція sort дозволяє отримати матрицю індексів відповідності елементів вихідної та упорядкованої матриць. Для цього необхідно викликати sort із двома вихідними аргументами:

```
>> [MC, Ind] = sort(M)
MC =
    1    -6    -4
    2    -2     0
    3    -2     4
Ind =
    1     2     1
    3     1     3
    2     3     2
```

Матриці M, MC, Ind зв'язані між собою в такий спосіб: MC(i,j)=M(Ind(i,j), j), де i та j змінюються від одного до трьох.

Функції min і max обчислюють вектор-рядок, що містить мінімальні або максимальні елементи у відповідних стовпцях матриці:

```
>> mx = max(M)
mx =
     3    -2     4
```

```
>> mn = min(M)
mn =
     1    -6    -4
```

Можна встановити не тільки значення максимальних і мінімальних елементів, але і їхні номери в стовпцях (при наявності декількох рівних

максимальних або мінімальних елементів повертається номер першого елемента):

```
>> [mx, k] = max(M)
mx =
  3   -2    4
k =
  2    1    2
```

```
>> [mn, n] = min(M)
mn =
   1   -6   -4
n =
   1    2    1
```

Функції min і max дозволяють виділити мінімальні й максимальні елементи із двох матриць однакових розмірів і записати результат у нову матрицю того ж розміру, що й вихідні:

```
>> P = [4   3   -1
        2    0   7];
>> Q = [10   0   11
       -5   3   22];
>> R = max(P, Q)
R =
   10    3   11
    2    3   22
```

Одним з аргументів може бути число. У результуючу матрицю записується максимум із цього числа і відповідного елемента вихідної матриці:

```
>> S = max(P, 1)
S =
    4    3    1
    2    1    7
```

Якщо обидва аргументи функцій min або max є числами, то повертається мінімальне або максимальне із цих чисел.

Для знаходження мінімуму або максимуму не по стовпцях матриці, а по рядках, передбачена така форма виклику із другим аргументом - порожнім масивом:

```
>> mx = max(S, [ ], 2)
mx =
    4
    7
```

Для того щоб додатково отримувати номери максимальних елементів у рядках, використовується виклик max із двома вихідними аргументами:

```
>> [mx, j] = max(S, [ ], 2)
```

```
mx =
     4
     7
j =
     1
     3
```

Повернути масив на 90 градусів проти годинникової стрілки дозволяє функція rot90:

```
>> p = [-1  3  4;  7  0  2]
>> q1 = rot90(p)
p =
    -1     3     4
     7     0     2
q1 =
     4     2
     3     0
    -1     7
```

і далі

```
>> q2 = rot90(q1)
q2 =
     2     0     7
     4     3    -1
```

У свою чергу, функція fliplr забезпечує дзеркальне відображення від умовної вертикалі, що проходить через середину масиву:

```
>> fliplr(q2)
ans =
     7     0     2
    -1     3     4
```

Але вектор-стовпець не змінюється під впливом функції fliplr.

Команда help datafun дозволяє вивести список всіх убудованих функцій.

3.1.11 Точність представлення елементів масиву

За замовчуванням всі числа, у тому числі й елементи масивів, зберігаються з подвійною точністю (double) і займають 8 байтів. Більші масиви вимагають для зберігання значних об'ємів пам'яті. Для зменшення об'єму можна застосовувати інші способи зберігання: single - для речовинних чисел (4 байти), int8, int16, int32 - для цілих чисел (1, 2, 4 байти відповідно).

Для призначення точності представлення чисел призначені функції: single, int8, int16, int32. Наприклад:

```
>> single (s4);
>> int32 (s3);
>> int16 (s2);
>> int8 (8);
```

По виконанні цих команд під масиви приділяється наступний розподіл пам'яті:

```
>> whos
```

Name	Size	Bytes	Class
q1	1x4	4	int8 array
q2	1x4	8	int16 array
q3	1x4	16	int32 array
q4	1x4	16	single array
s1	1x4	32	double array
s2	1x4	32	double array
s3	1x4	32	double array
s4	1x4	32	double array

Виконання операцій з речовинними числами різного типу припустимо й дає результат з найменшою точністю, тобто single:

```
>> qs44 = q4 + s4;
>> whos qs44
```

Name	Size	Bytes	Class
qs44	1x4	16	single array

Спроба виконати арифметичні операції над цілими числами різних типів приводить до помилки:

```
>> qs31 = q3 + q1
??? Error using => plus
Integers can only be combined with integers of the same class, or scalar doubles.
```

Більше того, при перетворенні цілого числа, коли для точного подання не вистачає відводиться памяти, що, утворюється максимальне число для даного типу, тобто помилковий результат, без попередження про це:

```
>> w1=34567;
>> r1=int8(w1)
```

```
r1 = 127
>> t1=int16(w1)
t1 = 32767
```

3.1.12 Побудова графіків двох змінних за допомогою матриць

Матричне подання даних може бути використане при побудові графіків функцій двох змінних. Для цього треба:

1. Згенерувати матриці з координатами вузлів сітки на прямокутній області визначення функції.
2. Обчислити значення функції у вузлах сітки і записати їх у матрицю.
3. Використати одну із графічних функцій MATLAB.
4. Нанести на графік додаткову інформацію, зокрема, відповідність кольорів значенням функції.

Наприклад, потрібно побудувати графік функції $z(x,y)=x^2+y^2$ на області визначення у вигляді квадрата $x\in[0,1]$, $y\in[0,1]$. Необхідно розбити квадрат рівномірною сіткою, наприклад, із кроком **0.2** і обчислити значення функції у вузлах (рис.8). Використовуються два масиви x та y розмірністю шість на шість для зберігання інформації про координати вузлів. Масив x складається з однакових рядків, в яких записані координати x1, x2, …, x6, а масив y містить однакові стовпці з y1, y2, …, y6... Значення функції у вузлах сітки заносяться в матрицю z такої ж розмірності шість на шість, причому для обчислення матриці z використовується вираз для функції, але із поелементними матричними операціями. Тоді, наприклад, z(3,4) буде дорівнювати значенню функції z(x,y) у точці (x3,y4).

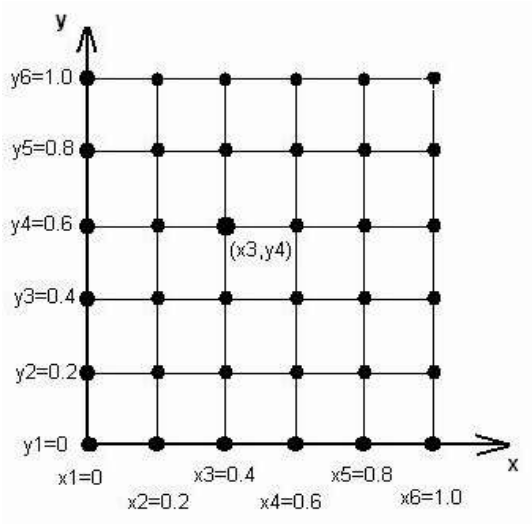


Рис. 8 Область побудови графіка

Для генерації масивів сітки x та y по координатах вузлів в MATLAB передбачена функція meshgrid, що викликається із двома вхідними й двома вихідними аргументами. Для побудови графіка у вигляді каркасної поверхні використовується функція mesh, що викликається з трьома аргументами.

Наступні оператори формують графік функції (рис.9а) і виводять на екран уміст масивів (тому що крапка з комою наприкінці операторів відсутня):

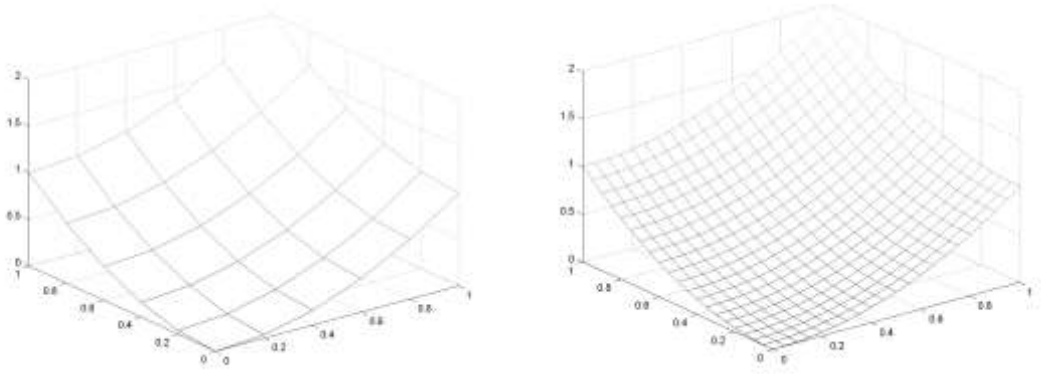
```
>> [X, Y] = meshgrid(0:0.2:1,  0:0.2:1)
>> Z = X.^2 + Y.^2
>> mesh(X, Y, Z)
```

Для більш точної побудови графіка треба вибрати менший крок сітки (рис.9б):

```
>> [X, Y] = meshgrid(0:0.05:1,  0:0.05:1)
>> Z = X.^2 + Y.^2
>> mesh(X, Y, Z)
```

Якщо область побудови функції - квадрат і крок сітки по обох напрямках однаковий, то припустимо вказати один аргумент. Наприклад, в останньому прикладі припустимий запис

```
>> [X, Y] = meshgrid(0:0.05:1)
```



а) б)

Рис. 9 Графік функції $z(x,y) = x^2 + y^2$

У свою чергу, крім наведеного способу виклику mesh із трьома вхідними аргументами (матрицями), допускається також ряд інших. Зокрема, якщо зазначено тільки один вхідний аргумент - матрицю, то на осях абсцис і ординат відкладаються значення, відповідно, стовпцевих і рядкових індексів її елементів. Замість номерів рядків і стовпців можна вказати вектори, що складаються з необхідних чисел. Ці вектори задаються в перших двох вхідних аргументах, а матриця - у третьому.

Наочну інформацію про співвідношення величин елементів матриці дає функція imagesc, що інтерпретує матрицю як прямокутне зображення. Кожен елемент матриці представляється у вигляді квадратика, колір якого відповідає значенню елемента.

Щоб дізнатися про відповідність між кольорами і величинами елементів, треба використовувати команду colorbar, що виводить поруч із зображенням матриці шкалу кольорів. Для друку на монохромному принтері зручно використовувати команду colormap(gray).

2.1.13 Перетворення зображень за допомогою матриць

Зображення в системі MATLAB представляються матрицями. На рис.10а показане напівтонове вихідне зображення. Його дзеркальне відображення (рис.10б) отримане за командою

```
>> ih=h(end:-1:1,:);
```

де h - масив даних вихідного зображення.

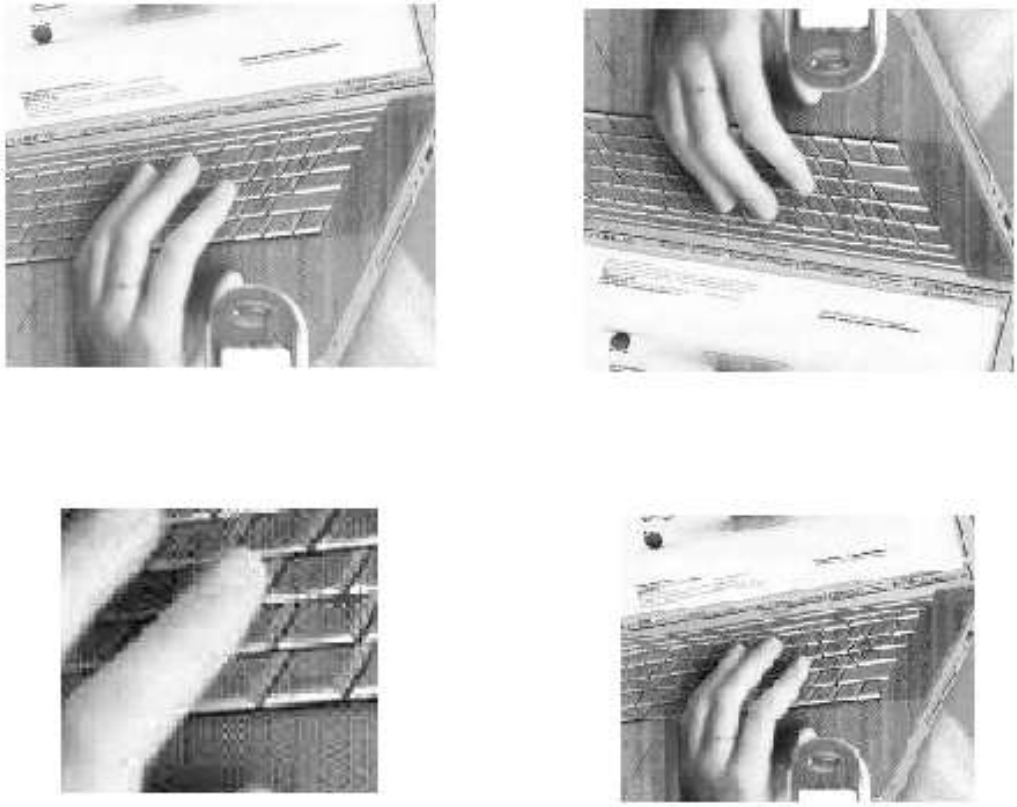


Рис.10 Приклад перетворення зображення

Зображення, що показані на рис.10в,г отримані за допомогою команд

```
>> ch=h(150:250, 150:250);
>> dh=h(1:2:end, 1:2:end);
```

Сформувати графік значень середнього рядка зображення (рис.11) можна за допомогою команди

```
>> plot(h(round(size(h,1)/2), :))
```

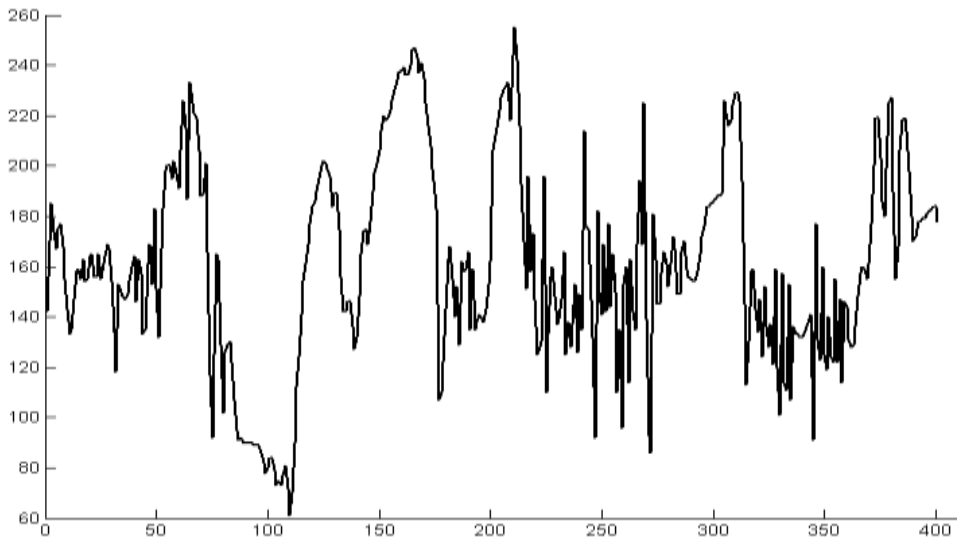


Рис. 11 Графік значень рядка зображення

3.2 ПОШУК ЕКСТРЕМУМІВ ФУНКЦІЇ ДЕКІЛЬКОХ ЗМІННИХ

Відшукати екстремуми функції декількох змінних дозволяє алгоритм функції fminsearch. Початкове наближення вказується вектором.

Перед застосуванням необхідно створити файл-функцію, що обчислює значення шуканої функції, причому аргументом файл-функції повинен бути вектор, перший елемент якого відповідає змінної x, а другий - y. Наприклад, для функції $z = \sin \pi x \cdot \sin \pi y$:

```
function z=ftest2(v)
x=v(1);
y=v(2);
z=sin(pi*x).*sin(pi*y);
```

Для знаходження локального мінімуму необхідно викликати fminsearch із двома вхідними аргументами – ім'ям файл-функції та початковим наближенням і вихідним аргументом – вектором з координатами шуканої точки мінімуму:

```
>> M= fminsearch('ftest2', [1.4,  0.6])
M =  1.5000    0.5000
```

Рішення знайдене з точністю 10-4, як за значеннями аргументів, так і за значенням функції. Для одержання не тільки вектора з координатами точки мінімуму, але й значення функції варто викликати fminsearch із двома вихідними аргументами:

```
>> [M, z]= fminsearch('ftest2', [1.4,  0.6])
M =   1.5000    0.5000
z =   -1.0000
```

Звертання до fminsearch із третім додатковим вихідним аргументом:

```
>> [M, z, flag]= fminsearch('ftest2', [1.4,  0.6])
```

дозволяє записати в нього інформацію про причину зупинки обчислень. Зміст його значень аналогічний змісту для функції fminbnd.

Пошук мінімуму можливий і при використанні анонімної функції:

```
>> fun = @(v) sin(pi*v(1)).*sin(pi*v(2));
```

```
>> [M, z]= fminsearch(fun, [1.4,  0.6])
M =   1.5000    0.5000
z =   -1.0000
```

3.3 ВИКОРИСТАННЯ ЗМІШАНИХ МАСИВІВ І СТРУКТУР У ПЕРЕТВОРЕННЯХ

3.3.1. Змішані масиви

При звертанні до змінних різних типів можна використовувати *змішані масиви*. У системі MATLAB змішаним масивом називається багатомірний масив, елементами якого є копії інших масивів, які можуть належати різним класам даних. Наприклад, змішаний масив

```
c={'gauss', [1  0;  0  1],  3,  A}
```

складається із чотирьох елементів: символьного рядка ‘gauss’, числової матриці розміром 2x2, скаляра та вектора. Для адресації елементів змішаного масиву використовуються цілі індекси, укладені у фігурні дужки. Наприклад,

```
>> c{1}          >> c{2}          >> c{3}          >> c{4}
ans =            ans =            ans =            ans =
    gauss            0    1           3                3 5 4 2
                     1    0
```

При виклику змішаного масиву для багатьох елементів видається повідомлення про їхні властивості, а не значеннях:

```
>> c =
   'gauss'    [2x2 double]    [3]    [1x4 double]
```

Аналогічна інформація буде видана при виконанні команд:

```
>> c(1)          >> c(2)          >> c(3)          >> c(4)
ans =            ans =            ans =            ans =
    'gauss'          [2x2 double]     [3]              [1x4 double]
```

Функція size видає розмір змішаного масиву:

```
>> size(c)
ans =
    1     4
```

Функція cellfun, що має синтаксис

```
D=cellfun('fname', c)
```

застосовує функцію fname до елементів змішаного масиву c і повертає результат у вигляді масиву D класу double. Наприклад,

```
>> D=cellfun('length', c)
D =
    5     2     1     4
```

Інакше, length(‘gauss’)=5, length([0 1; 1 0])=2, length(3)=1, length(A)=4.

Важливою особливістю змішаних масивів є факт зберігання в них копій відповідних аргументів, а не покажчиків на ці аргументи. Наприклад, при зміні вектора A зміст масиву c залишиться незмінним доти, поки масив не буде перевизначений.

Наприклад, необхідно створити функцію, виходом якої є середня яскравість зображення, його розмір, середня яскравість по рядках і середня яскравість по стовпцях. Звичайний варіант функції має вигляд:

```
function [AI, dim, AIrows, AIcols] = image_stats(f)
dim = size(f);
AI = mean2(f);
AIrows = mean(f, 2);
AIcols = mean(f, 1);
```

де f – вихідне зображення.

Функція з використанням змішаного масиву має вигляд:

```
function G = image_stats(f)
G{1} = size(f);
G{2} = mean2(f);
G{3} = mean(f, 2);
```

```
G{4} = mean(f, 1);
```

Змішані масиви можуть бути багаторозмірними. Припустима також запис G(1)={size(f)}. Наприклад, попередню функцію можна визначити в такий спосіб:

```
function H = image_stats2(f)
H(1, 1) = {size(f)};
H(1, 2) = {mean2(f)};
H(2, 1) = {mean(f, 2)};
H(2, 2) = {mean(f, 1)};
```

Для зображення розміром 512x512:

```
G = image_stats(f)
>> H =  image_stats2(f);
>> G
G =
    [1x2 double]    [1]    [512x1 double]    [1x512 double]
>> H
H =
    [1x2 double]       [1]
    [512x1 double]    [1x512 double]
```

Якщо необхідно працювати зі змінной, що втримується в масиві, наприклад, з розміром зображення, то можна її визначити звертанням до елемента змішаного масиву:

```
>> v = G{1}          або          >> v = H{1, 1}
```

У свою чергу, застосування команди [M, N] = G{1} викличе помилку, оскільки тільки функції можуть формувати багатомірні результати. Для одержання M і N варто записати M = v(1) і N = v(2).

3.3.2. Структури

Структури також дозволяють групувати дані різних типів, призначаючи їм єдину змінну. Але, на відміну від змішаних масивів, у яких індексами служать цілі числа, елементам структур присвоюються імена, нази-

вані *полями*, що більш наочно. Наприклад, S – структурна змінна, у якій є поля з іменами char_string, matrix, scalar, vector. Тоді призначити структуру можна наступними присвоєннями:

```
S.char_string = 'gauss';
S.matrix = [1  0;  0  1];
S.scalar = 3;
S.vector = A;
```

Вирішити задачу визначення середньої яскравості зображення, його розміру, середніх яскравостей по рядках і по стовпцях можна в такий спосіб:

```
function s = image_stats(f)
s.dim = size(f);
s.AI = mean2(f);
s.AIrows = mean(f, 2);
s.AIcols = mean(f, 1);
```

де s – структура, полями якої є AI (скаляр), dim (вектор 1x2), AIrows (вектор Mx2) і AIcols (вектор 1x), де M і N позначають число рядків і стовпців зображення.

Ім'ям поля може бути будь-яка послідовність букв і цифр, що починається з букви. Сама змінна s є скаляром, з яким, у цьому випадку, асоційовані чотири поля:

```
>> s =
s =
        dim: [512  512]
         AI: 1
    AIrows: [512x1  double]
    AIcols: [1x512  double]
>> size(s)
ans =
          1    1
```

Для декількох зображень, кількість яких дорівнює Q, організованих у єдиний масив, розміром MxNx, функція має вигляд:

```
function s = image_stats(f)
K = size(f);
for k = 1:K(3)
    s(k).dim = size(f(:, :, k));
    s(k).AI = mean2(f(:, :, k));
    s(k).AIrows = mean(f(:, :, k), 2);
    s(k).AIcols = mean(f(:, :, k), 1);
end
```

Таким чином, структури можна індексувати.

При добуванні даних з полів необхідно запам'ятовувати розмірність як змінної s, так і її полів. Наприклад, що випливає команда витягає всі змінні з AIrows і зберігає їх в v:

```
for k = 1:length(s)
   v(:, k) = s(k).AIrows;
end
```

Для цього приклада в v двокрапка стоїть на першому місці, а індекс k на другому, оскільки s має розмірність 1x, а розмірність AIrows дорівнює Mx. Таким чином, розмірність v дорівнює Mx. При добуванні даних з AIcols варто записати v(k, :).

Квадратні дужки можна використовувати при записі даних у вектор або матрицю, якщо поля структури містять скаляри. Наприклад, якщо D.Area містить площу кожної з 20 областей зображення, то запис

```
>> w = [D.Area];
```

формує вектор w розміром 1x20, кожний елемент якого є площею однієї області.

Аналогічно змішаному масиву, у структурі втримуються копії аргументів, зміна значень яких не відображається в поле структури доти масив не буде перевизначений.

3.4 ЗАВДАННЯ ДЛЯ САМОКОНТРОЛЮ

Вихідні дані - функція двох змінних (задаються самостійно). Необхідно:

1. Використовуючи правила роботи з матрицями побудувати графік заданої функції двох змінних на області визначення, найбільш характерної для відображення сутності функції.
2. Вивести результати в табличному вигляді.
3. Зберегти результати у файлі.
4. Виконати контрольні прорахунки.
5. Виконати одне з наступних завдань до роботи з матрицями (у якості даних для контролю рішення завдання треба використовувати результати роботи із заданою функцією):

- Переставити стовпці матриці у порядку зростання суми їх елементів.
- Замінити елементи матриці, більші середнього арифметичного її елементів, на середнє арифметичне.
- Замінити всі мінімальні елементи матриці напівсумою максимального та мінімального її елементів.
- Знайти число позитивних, негативних і нульових елементів матриці.
- Знайти число елементів матриці, що відрізняються від середнього арифметичного менше, ніж на 20%.
- Замінити елементи матриці, що відрізняються від нульового рівня не більше ніж на 5%, на нуль.
- Замінити елементи матриці, що відрізняються від максимального позитивного значення не більше ніж на 5%, на максимальне.
- Замінити елементи матриці, що відрізняються від мінімального негативного значення не більше ніж на 5%, на мінімальне.
- Визначити максимальний стовпцевий і рядковий індекси негативних елементів матриці.
- Обчислити суми позитивних і негативних елементів матриці.

- Знайти максимальне значення серед діагональних елементів матриці.
- Переставити перший стовпець квадратної матриці з її діагоналлю.
- Обчислити суму усіх позадіагональних елементів матриці.
- Замінити мінімальний елемент матриці сумою елементів останнього стовпця матриці.
- Підрахувати число нулів та одиниць у матриці.
- Обчислити суму негативних елементів матриці, що лежать нижче головної діагоналі.

3.5 ЗАПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

- Як з командного рядка задати матриці ?
- За допомогою яких операцій можна звернутися до елементів матриці?
- У чому полягає сутність логічного індексування ?
- Які особливості використання матричних операцій ?
- Які особливості перемножування матриці на вектор ?
- Як формуються блокові матриці ?
- Яким чином можна видалити рядки і стовпці ?
- Яким чином можна згенерувати матрицю ?
- Як працюють функції zeros, eye ?
- Як працюють функції ones, rand ?
- Як працює функція diag ?
- Як виконуються операції .* при обробці матриць ?
- Як виконуються операції .^ при обробці матриць ?
- Як виконуються операції ./ и.\ при обробці матриць ?
- Як обчислюється функція cos від елементів матриць ?
- Як обчислюється функція sum від елементів матриць ?
- Як обчислюється функція sort від елементів матриць ?
- Як обчислюються функції min і max від елементів матриць ?

- Як використовуються функції rot90 і fliplr ?
- Як використовуються функції meshgrid і mesh ?
- Як використовується функція imagesc ?
- Як використовується функція colorbar ?

4. ГРАФІЧНІ ЗАСОБИ В MATLAB

Мета: вивчення можливостей високорівневої графіки MATLAB для відображення функцій двох і трьох змінних.

4.1 ЗМІНА ВЛАСТИВОСТЕЙ ПОДАННЯ ГРАФІКІВ

Оскільки MATLAB - матрична програма, то її графічні команди можуть різноманітними способами візуалізувати результат обробки векторів і матриць.

MATLAB надає можливість керувати виглядом графіків, побудованих за допомогою plot, loglog, semilogx, semilogy, для чого служить аргумент, що міститься за кожною парою векторів. Цей аргумент записується в апострофи й містить від одного до трьох символів, які визначають колір, тип маркера і тип лінії. У табл. 1 наведено можливі значення даного аргументу із вказівкою результату.

Таблиця 1 Властивості лінії

Колір		Тип маркера		Тип лінії	
y	жовтий	.	крапка	-	суцільна
		o	кружок		
		x	хрестик		
c	блакитний	+	знак "плюс"		
		*	зірочка	:	пунктирна
		s	квадрат		
g	зелений	d	ромб		
		v	трикутник вершиною вниз		
b	синій	^	трикутник вершиною нагору	-.	штрих-пунктирна
		<	трикутник вершиною вліво		
r	червоний				
		>	трикутник вершиною вправо		
w	білий	p	п'ятикінечна зірка	--	штрихова
k	чорний	h	шестикінечна зірка		

Наприклад, у результаті виконання команд

```
>> x = -1:0.005 : -0.3;
```

```
>> f = sin(x.^-2);
>> g = sin(1.2*x.^-2);
>> plot(x, f, 'k-', x, g, 'k:')
```

перший графік зображується суцільною чорною лінією, а другий – чорною пунктирною (рис.12). Аргументи 'k-' та 'k:' задають стиль і колір першої та другої ліній. Тут k означає чорний колір, а дефіс або двокрапка – суцільну або пунктирну лінію.

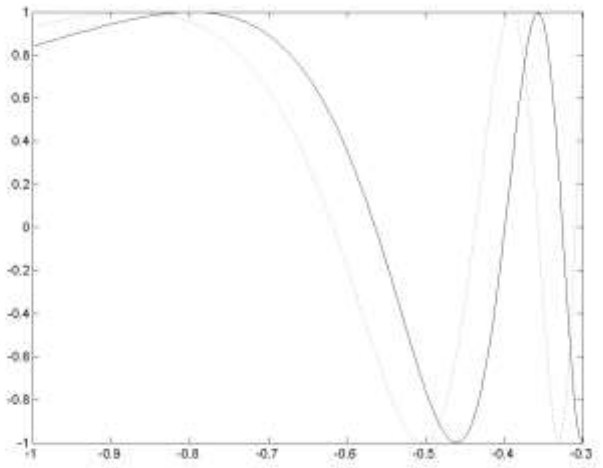


Рис.12 Приклад зміни стилю графіків

За допомогою додаткових параметрів або функцій можна задати елементи оформлення графіків - координатної сітки, підписів до осей, заголовка і легенди.

Сітка наноситься командою grid on. Функції xlabel, ylabel служать для розміщення підписів до осей. Функція title – для розміщення заголовка. При необхідності супроводити графік легендою слід використовувати функцію legend. Всі перераховані команди застосовні до графіків як у лінійному, так і в логарифмічному масштабах.

Наприклад, наступні команди виводять графіки зміни добової температури, що зображені на рис.13, які супроводжуються відповідною інформацією:

```
>> time = [0 4 7 9 10 11 12 13 13.5 14 14.5 15 16 17 18 20 22];
>> temp1 = [14 15 14 16 18 17 20 22 24 28 25 20 16 13 13 14 13];
>> temp2 = [12 13 13 14 16 18 20 20 23 25 25 20 16 12 12 11 10];
>> plot(time, temp1, 'ro-', time, temp2, 'go-')
>> grid on
```

```
>> title('Добові температури')
>> xlabel('Час (год.)')
>> ylabel('Температура (С)')
>> legend('10 травня', '11 травня')
```

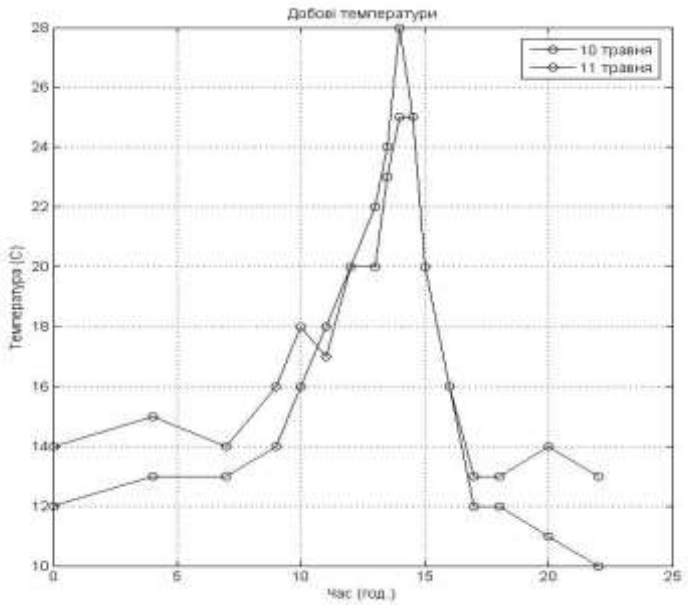


Рис.13 Графік зміни добової температури

При розміщенні легенди слід враховувати, що порядок і кількість аргументів команди legend повинні відповідати лініям на графіку. Останнім додатковим аргументом legend може бути місцезнаходження легенди в графічному вікні:

- -1 - поза графіком у правому верхньому куті графічного вікна,
- 0 - вибирається найкраще розташування в межах графіка так, щоб якнайменше перекривати графіки,
- 1 - у верхньому правому куті графіка (це положення використається за умовчанням),
- 2 - у верхньому лівому куті графіка,
- 3 - у нижньому лівому куті графіка,
- 4 - у нижньому правому куті графіка.

4.2 ГРАФІКИ ПАРАМЕТРИЧНИХ І КУСКОВЕ-ЗАДАНИХ ФУНКЦІЙ

Для побудови функцій, заданих параметрично, необхідно спочатку згенерувати вектор значень аргументу. Потім необхідно обчислити зна-

чення функцій і записати їх у вектори, які й треба використовувати як аргументи plot. Наприклад, графік функції $x(t) = 0.5 \cdot sin\, t$, $y(t) = 0.7 \cdot cos\, t$ для $t \in [0,2\pi]$ (еліпс), наведений на рис.14, утворюється за допомогою наступних команд:

```
>> t = 0:0.01:2*pi;
>> x = 0.5*sin(t);
>> y = 0.7*cos(t);
>> plot(x, y)
```

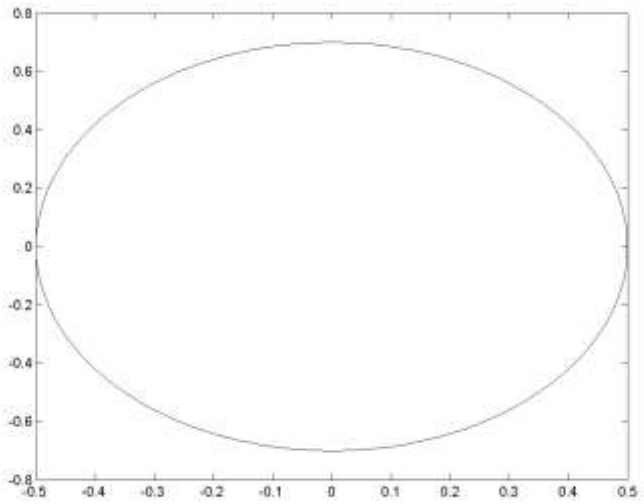


Рис.14 Графік функції, заданої параметрично

Наступний приклад показує формування графіка кусково-заданої функції:

$$y(x) = \begin{cases} \pi \cdot sin\, x, -2\pi \le x \le -\pi; \\ \pi - |x|, -\pi < x < \pi; \\ \pi \cdot sin^3\, x, \pi \le x \le 2\pi. \end{cases}$$

Спочатку обчислюється кожна із трьох гілок, тобто формуються три пари масивів x1 та y1, x2 та y2, x3 та y3, потім значення абсцис об'єднуються у вектор x, а значення ординат – у вектор y і виводиться графік функції, що задається парою масивів x та y:

```
>> x1 = -2*pi : pi/30 : -pi;
>> y1 = pi*sin(x1);
>> x2 = -pi : pi/30 : pi;
>> y2 = pi-abs(x2);
>> x3 = pi : pi/30:2*pi;
```

```
>> y3 = pi*sin(x1).^3;
>> x = [x1 x2 x3];
>> y = [y1 y2 y3];
>> plot(x, y)
```

При такому підході виходить графік, зображений на рис.15.

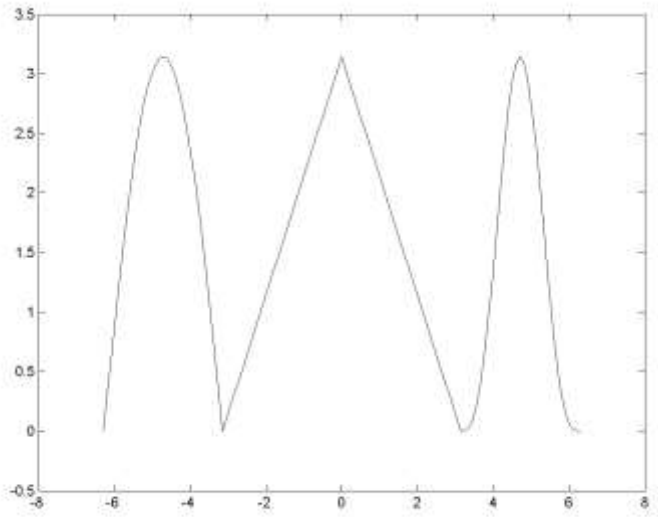


Рис.15 Графік функції, заданої кусковим способом

Можна також побудувати графіки трьох гілок як три різні функції, указавши кожну своїми кольорами і маркером:

```
>> plot(x1, y1, 'r+', x2, y2, 'kx', x3, y3, 'bs')
```

У цьому випадку графік має більш наочний вигляд (рис.16).

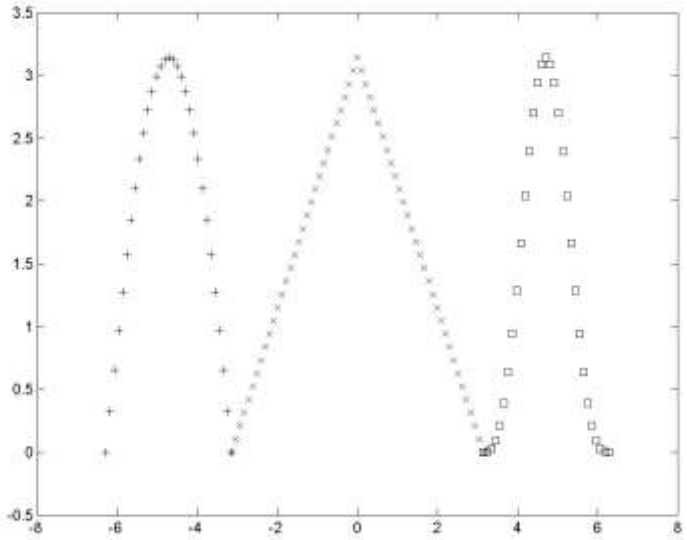


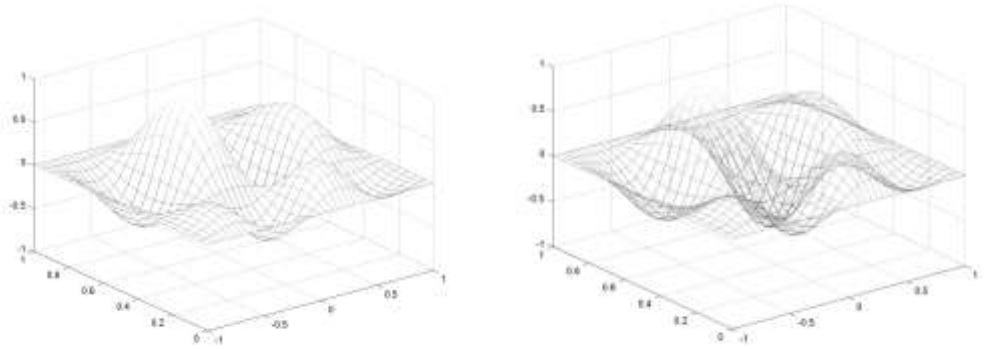
Рис.16 Графік функції, заданої кусковим способом з різними кольорами і маркерами гілок

4.3 КЕРУВАННЯ ВІЗУАЛІЗАЦІЄЮ ТРИВИМІРНИХ ГРАФІКІВ

Наприклад, на прямокутній області визначення $x \in [-1,1]$, $y \in [0,1]$ сформовано графік функції $z(x,y) = 4 \cdot sin\,2\pi x \cdot cos\,1.5\pi y \cdot (1-x^2) \cdot y \cdot (1-y)$ за допомогою наступного набору команд (рис.17а):

```
>> [X, Y] = meshgrid(-1:0.05:1,   0:0.05 : 1);
>> Z = 4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*Y.*(1-Y);
>> mesh(X, Y, Z)
```

За умовчуванням рисується тільки видима частина поверхні. За допомогою команди hidden off можна зробити каркасну поверхню "прозорою" (рис.17б). Команда hidden on прибирає невидиму частину поверхні, повертаючи графіку колишній вигляд.



а) б)

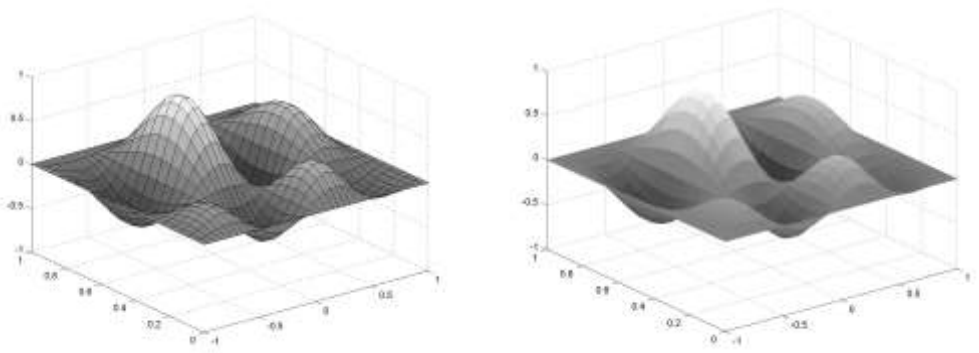
Рис.17 Варіанти каркасної поверхні графіка

Функція surf будує каркасну поверхню графіка функції і заливає кожну клітку поверхні певними кольорами, що залежать від значення функції в точках, що відповідають кутам клітки. Команда >> surf(X, Y, Z) будує графік, зображений на рис.18а.

У межах кожної клітини колір постійний. Команда shading flat дозволяє прибрати каркасні лінії. Для одержання поверхні, плавно залитої кольором, що залежить від значення функції (рис.18б), призначена команда shading interp. Повернутися до початкового вигляду поверхні можна за допомогою команди shading faceted.

Команда colorbar виводить поруч із графіком колірну шкалу, що встановлює відповідність між кольорами та значенням функції (рис.19а). Цю команду можна застосовувати в сполученні з усіма функціями, що будують тривимірні об'єкти:

```
>> surf(X, Y, Z)
>> colorbar
```

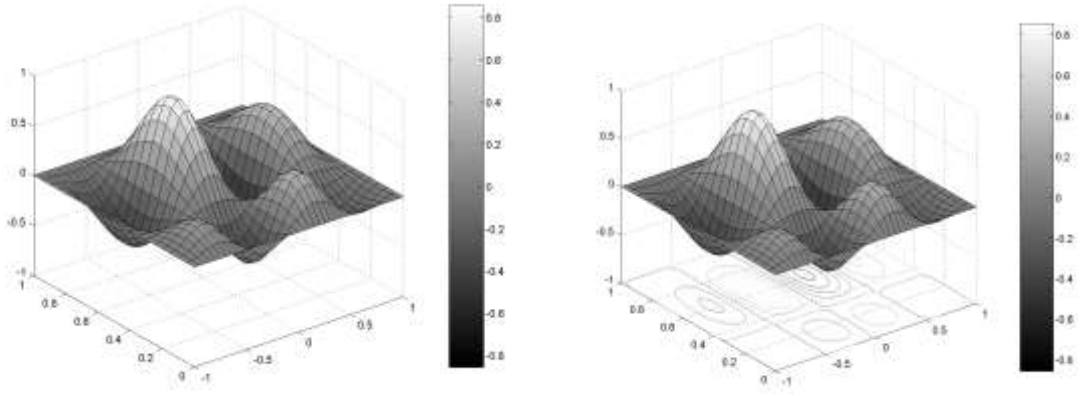


а) б)

Рис.18 Варіанти поверхні, залитої кольорами

Можливо також формування графіка, що містить лінії рівня функції, тобто лінії сталості значень функції на площині xy (рис.19б). Для цього замість команд mesh або surf треба використовувати команди meshc або surfc :

```
>> surfc(X, Y, Z)
>> colorbar
```



а) б)

Рис.19 Приклади використання функцій colorbar і surfc

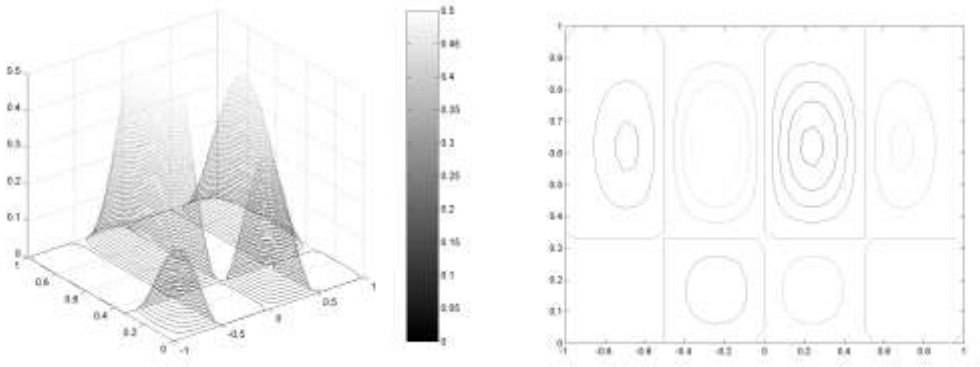
4.4 КОНТУРНІ ГРАФІКИ

MATLAB дозволяє побудувати поверхню, що складається з ліній рівня, за допомогою функції contour3. Цю функцію можна використовувати аналогічно функціям mesh, surf, meshc, surfc із трьома аргументами. При цьому число ліній рівня вибирається автоматично. Є можливість задати четвертим аргументом в contour3 або число ліній рівня, або вектор, елементи якого дорівнюють значенням функції, що відображуються у вигляді ліній рівня. Використовувати вектор зручно, коли потрібно досліджувати поведінку функції в деякій області її значень. Наприклад, на рис.20 показаний результат формування поверхні, яка складається з ліній рівня, котрі відповідають значенням функції від 0 до 0.5 із кроком 0.01:

```
>> levels = 0:0.01:0.5;
>> contour3(X, Y, Z, levels)
>> colorbar
```

Але більш змістовну інформацію про числові значення дають плоскі контурні графіки з лініями рівня досліджуваної функції. Різні типи контурних графіків можна одержати за допомогою функцій contour і contourf. Наприклад, використання contour із трьома аргументами призводить до графіка з лініями рівнів на площині xy (рис.20б):

```
>> contour(X, Y, Z).
```



а) б)

Рис.20 Графіки зрізу (а) і ліній рівнів функції (б)

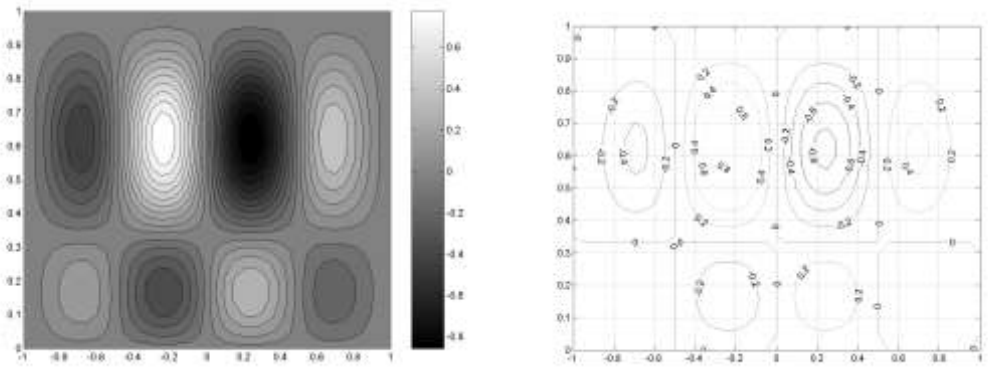
Кожну лінію рівня можна позначити ярликом з відповідним значенням за допомогою функції clabel, що викликається із двома аргументами: матрицею, що містить інформацію про лінії рівня та вказівником на графік, на якому треба нанести розмітку. Користувачеві не потрібно самому створювати аргументи clabel. Функція contour, викликана із двома вихідними параметрами, не тільки будує лінії рівня, але й знаходить для clabel параметри. У наступному прикладі матриця CMatr містить інформацію про лінії рівня, а вектор h – вказівники:

```
>> [CMatr, h]  = contour(X, Y, Z);
>> clabel(CMatr, h)
>> grid on
```

Отриманий графік показаний на рис.21а. Додатковим аргументом функції contour може бути або число ліній рівня, або вектор, що містить значення функції, для яких потрібно побудувати лінії рівня.

Наочну інформацію про зміну функції дає заливання області визначення кольором, котрий залежить від значення функції в точках площини. Для побудови таких графіків призначена функція contourf, результат використання якої для графіка, що складається із двадцяти ліній рівня, показаний на рис.21б:

```
>> contourf(X, Y, Z, 20)
>> colorbar
```



а) б)

Рис.21 Графіки з маркірованими лініями (а) і заливанням рівнів (б)

4.5 ОФОРМЛЕННЯ ТА ПОВОРОТ ГРАФІКІВ

За допомогою функції colormap можна змінювати колірне оформлення графіків. Кольорові палітри, доступні в MATLAB, наведені в табл.2.

Таблиця 2 Палітри кольорів

Палітра	Зміна кольорів		Палітра	Зміна кольорів
autumn	Плавна зміна червоний-жовтогарячий-жовтий		jet	Плавна зміна синій-блакитний-зелений-жовтий-червоний
bone	Аналогічна палітрі gray, але з легким відтінком синього кольору		pink	Аналогічна палітрі gray, але з легким відтінком коричневого кольору
colorcube	Кожний колір змінюється від темного до яскравого		prism	Циклічна зміна червоний-жовтогарячий-жовтий-зелений-синій-фіолетовий
cool	Відтінки блакитного та пурпурного кольорів		spring	Відтінки пурпурного та жовтого
copper	Відтінки мідного кольору		summer	Відтінки зеленого та жовтого
flag	Циклічна зміна червоний-білий-синій-чорний		vga	Палітра Windows із шістнадцяти кольорів
gray	Відтінки сірого		white	Лише білий колір
hot	Плавна зміна чорний-червоний-жовтогарячий-жовтий-білий		winter	Відтінки синього та зеленого
hsv	Плавна зміна за аналогією із кольорами веселки			

Заголовки графіків і підписи до осей установлюються командами title, xlabel, ylabel. Для вертикальної осі використовується команда zlabel.

Використання в аргументах команд математичних позначень у форматі TeX дозволяє додавати формули на графік. Наприклад, для занесення в заголовок графіка формули відображуваної функції $z(x,y) = 4 \cdot sin\,2\pi x \cdot cos\,1.5\pi y \cdot (1-x^2) \cdot y \cdot (1-y)$ використається команда (три крапки служать для розміщення команди у двох рядках):

```
>> title('4 sin(2\pi{\itx}) cos(1.5\pi{\ity}) (1 - {\ity}^{2})...
{\ity} (1 - {\ity})')
```

Правила набору формул і зміни властивостей шрифтів у форматі TeX наведені в табл.3. Прямий шрифт тексту встановлюється командою \rm. Для одержання звичайного прямого шрифту можна не вказувати ніяких команд. За умовчанням в MATLAB підтримується основний формат TeX.

Таблиця 3 Правила набору формул і зміни властивостей шрифтів

Дія	Команда TeX	Результат
Виділення курсивом одного символу або тексту	{\itx}	x
	1.2{\it}	1.2*P*
	{\itГіперболічний} синус	*Гіперболічний* синус
Виділення жирним шрифтом одного символу або тексту	Шаблон матриці {\bf}	Шаблон матриці **M**
	{\bfАЧХ} фільтра	**АЧХ** фільтра
Набір символу або тексту жирним курсивом	Вектори {\bf\itx} та {\bf\ity}	Вектори **x** та **y**
	{\bf\itОптимальна} крива	***Оптимальна*** крива
Зміна шрифту і його розміру	{\fontname{arial}\fontsize{14}z-функція}	Z-функція
Ступінь, верхній індекс	x^{2}	x^2
	{\itx}^{2.5}	$x^{2.5}$
	{\ite}^{\it-x}	e^{-x}
Нижній індекс	f_{5}	f_5
	f_{\itxx}	f_{xx}

Можливе використання грецьких букв і спеціальних символів. Наприклад, title('Залежність при a=\pi') призводить до заголовка "Залежність при a=π". У таблицях 4 і 5 наведені команди TeX для вводу деяких прописних і рядкових грецьких букв і спеціальних символів.

Таблиця 4 Грецькі букви

Команда	Символ	Команда	Символ	Команда	Символ
\alpha	α	\lambda	λ	\chi	χ
\beta	β	\mu	μ	\psi	ψ
\gamma	γ	\nu	ν	\omega	ω
\delta	δ	\xi	ξ	\Gamma	Γ
\epsilon	ε	\rho	ρ	\Delta	Δ
\eta	η	\sigma	σ	\Theta	Θ
\theta	θ	\tau	τ	\Lambda	Λ
\kappa	κ	\phi	φ	\Phi	Φ

Таблиця 5 Спеціальні символи

Команда	Символ	Команда	Символ
\leq	$\leq$	\leftrightarrow	$\leftrightarrow$
\geq	$\geq$	\leftarrow	$\leftarrow$
\pm	$\pm$	\rightarrow	$\rightarrow$
\propto	$\propto$	\downarrow	$\downarrow$
\partial	∂	\uparrow	$\uparrow$

Текст у форматі TeX можна використовувати як аргумент функцій title, xlabel, ylabel при побудові двовимірних графіків й у тих же командах разом з zlabel для тривимірних графіків. Наступний приклад пояснює використання різних функцій:

```
>> [X, Y] = meshgrid(0:0.05:1,  -2:0.05:0);
>> Z = -exp(-Y.^2).*cos(3*pi*X).*X.*(1-X).*Y;
>> surfc(X, Y, Z)
>> colormap(gray)
>> colorbar
>> title('Графік {\itz} = - {\ite}^{-{\ity}^{2}} cos(3 \pi {\itx})...
{\itx} (1-{\itx}) {\ity}')
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
```

Результат виконання команд показаний на рис.22.

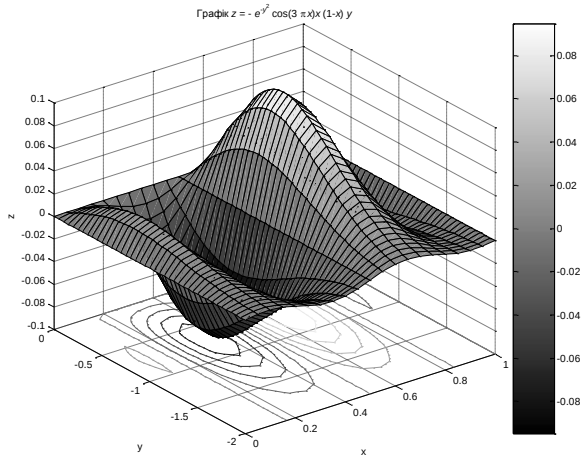


Рис.22 Приклад графіка у відтінках сірого

Графік можна повернути, тобто змінити положення спостерігача, функцією view. Аргументами view є азимут *Az* і кут підвищення *El*, відлічувані в градусах. Азимут відраховує від осі, протилежної y, а кут підвищення – від площини xy. За умовчанням *Az=-37.50*, *El=30*^{0}. Щоб довідатися про поточне положення спостерігача, слід викликати view із двома вихідними аргументами:

```
>> [Az, El] = view
Az =
        -37.5000
El =
        30
```

Потрібне положення спостерігача задається вхідними аргументами view, наприклад, view(135, 45).

Можна повертати графік "мишею". Для цього досить активізувати останню праворуч кнопку панелі інструментів із зображенням пунктирної окружності зі стрілкою. Тепер, увівши курсор "миші" в область графіка і натиснувши ліву кнопку, можна круговими рухами змусити графік обертатися.

MATLAB дозволяє одержати наочне зображення поверхні в тривимірному просторі, освітленої з однієї або декількох сторін. Для цього використовується функція surfl. При використанні surfl зручно задавати колірні палітри copper, bone, gray, pink, у яких інтенсивність кольорів змінюється лінійно. Для одержання відтінків, що змінюються плавно, необхідно використовувати shading interp. За умовчанням джерело світла має азимут, більший на 45^0, ніж спостерігач, і той же кут підвищення. Додатковим четвертим аргументом surfl може бути вектор-рядок із двох елементів – азимута і кута підвищення джерела світла. Змінити азимут джерела на -90^0 стосовно спостерігача та установити кут підвищення в нуль можна командами:

```
>> [Az, El] = view;
>> surfl(X, Y, Z, [Az-90, 0])
>> shading interp
```

MATLAB надає можливість одержати анімований графік, на якому кружок, що позначає точку, переміщається на площині або в просторі, залишаючи за собою слід у вигляді лінії – траєкторії руху. Для побудови анімованих графіків застосовуються функції comet і comet3:

```
>> t = [0:0.001:10];
>> x = sin(t)./(t+1);
>> y = cos(t)./(t+1);
>> comet(x, y)
```

Швидкістю руху кружка можна керувати, задаючи різні кроки при автоматичному заповненні вектора, відповідного часу. Використання comet з одним аргументом (вектором) приводить до побудови графіка, що

динамічно рисується, зі значеннями елементів номера залежно від їхніх номерів. Функцію comet можна викликати і з третім додатковим числовим параметром, що задає довжину хвоста комети. За умовчуванням він дорівнює 0.1. При зміні розмірів вікна траєкторія руху зникає.

Для побудови траєкторії точки, що переміщується в просторі, використовується функція comet3. Наприклад:

```
>> t = [0:0.1 : 100];
>> x = exp(abs(t-50)/50).*sin(t);
>> y = exp(abs(t-50)/50).*cos(t);
>> z = t;
>> comet3(x, y, z)
```

Функцію comet3 можна викликати із четвертим числовим аргументом, що задає довжину хвоста комети.

4.6 РОБОТА З ДЕКІЛЬКОМА ГРАФІКАМИ

MATLAB надає наступні можливості роботи з декількома графіками:

- Виведення кожного графіка в окреме вікно.
- Виведення декількох графіків на одні координатні осі в одне вікно.
- Відображення декількох графіків на своїх осях в межах одного вікна.

4.6.1 Виведення кожного графіка в окреме вікно

Для створення окремого вікна для кожного графіка використається функція figure. Вікно стає поточним. Для одержання нового графічного вікна слід знову використати функцію figure. Наприклад, послідовність команд:

```
>> [X, Y] = meshgrid(-1:0.05:1,  0:0.05:1);
>> Z = 4*sin(2*pi*X).*cos(1.5*pi*Y).*(1 - X.^2).*Y.*(1 - Y);
>> figure
>> mesh(X, Y, Z)
>> figure
>> surfl(X, Y, Z)
```

призводить до появи на екрані двох графічних вікон **Figure1** і **Figure2,** що містять різні види поверхонь. Вікно **Figure2** є поточним, тому що було створено останнім. Тому наступні команди призведуть до зміни саме в цьому вікні, наприклад:

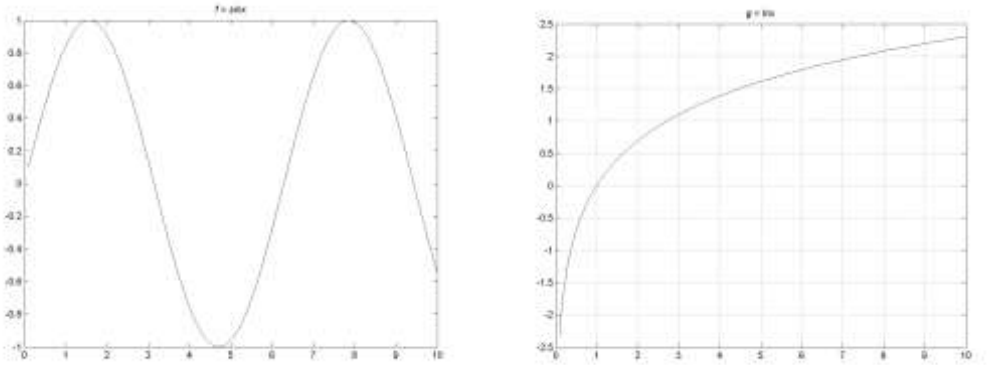
```
>> colormap('copper')
>> shading interp
```

Для очищення всього поточного вікна використається команда clf (скорочення від clear figure), а щоб прибрати тільки графік, але залишити осі, заголовок і назви осей, застосовується команда cla (скорочення від clear axes).

Щоб зробити поточне вікно **Figure1** слід клацнути на ньому мишкою, повернутися в робоче середовище MATLAB і продовжувати уведення команд. Більш зручним є використання вказівника графічного вікна. Для цього при створенні кожного нового вікна команду figure слід викликати з вихідним аргументом. Значенням вихідного аргументу є число, що збігається з номером вікна. Для того, щоб зробити вікно поточним, слід викликати figure, указавши як вхідний аргумент вказівник на потрібне вікно.

Наприклад, потрібно створити два графічних вікна, побудувати в них графіки функцій $f = \sin x$ та $g = \ln x$, а потім оформити їх, тобто вказати заголовки й нанести сітку на другий графік (рис.23). Ці задачі вирішуються наступною послідовністю команд:

```
>> sinGr = figure;
>> lnGr = figure;
>> x = [0.1 : 0.05 : 10];
>> f = sin(x);
>> g = log(x);
>> figure(sinGr)
>> plot(x, f)
>> figure(lnGr)
>> plot(x, g)
>> figure(sinGr)
>> title('\itf = sin\itx')
>> figure(lnGr)
>> title('\itg = ln\itx')
>> grid on
```



а) б)

Рис.23 Приклад висновку графіків у різні вікна

Для очищення вікна з вказівником lnGr, слід використати clf(lnGr). Видалити графік з першого вікна, на яке вказує sinGr, можна за допомогою cla(sinGr).

4.6.2 Виведення декількох графіків на одні осі

Виводити графіки декількох функцій на одні осі дозволяють функції plot, plotyy, semilogx, semilogy, loglog задаючи відповідні векторні аргументи парами. Для об'єднання графіків призначена команда hold on, яку необхідно задати перед побудовою чергового графіка. У наступному прикладі виводиться перетинання площини та конуса, заданого параметрично (рис.24):

```
>> u = (-2*pi : 0.1*pi : 2*pi)';
>> v = -2*pi : 0.1*pi : 2*pi;
>> X = 0.3*u*cos(v);
>> Y = 0.3*u*sin(v);
>> Z = 0.6*u*ones(size(v));
>> surf(X, Y, Z)
>> [X, Y] = meshgrid(-2:0.1:2);
>> Z = 0.5*X + 0.4*Y;
>> hold on
>> mesh(X, Y, Z)
>> hidden off
```

Команда hidden off дозволяє показати частину конуса, що знаходиться під площиною.

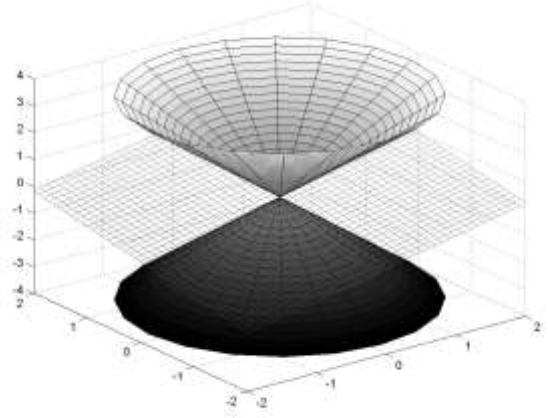


Рис.24 Перетинання площини та конуса

Команда hold on поширюється на всі наступні виводи графіків. Для розміщення графіків на нових осях слід виконати команду hold off. Команда hold on може застосовуватися і для розташування декількох графіків функції однієї змінної, наприклад,

```
>> plot(x. f, x, g)
```

еквівалентно послідовності

```
>> plot(x, f)
>> hold on
>> plot(x, g)
```

4.6.3 Виведення декількох графіків в одне вікно

MATLAB дозволяє розмістити в графічному вікні кілька осей і вивести на них різні графіки. Найпростіший спосіб полягає в розбивці вікна на кілька частин по горизонталі і вертикалі з використанням функції subplot, що розташовує осі у вигляді матриці й використовується із трьома параметрами: subplot(i, j, n). Тут i та j – число підграфіків по вертикалі та горизонталі, а n – номер підграфіка, який треба зробити поточним. Номер відраховується від лівого верхнього кута порядково. Послідовність викликів

```
>> subplot(3, 2, 1)
>> subplot(3, 2, 2)
…
>> subplot(3, 2, 6)
```

призводить до розміщення шести осей координат у графічному вікні (рис. 14). Поточними є останні створені осі, тобто всі графічні функції будуть

виводитися на праві нижні осі. Для виводу графіків на інші осі їх треба зробити поточними. Це досягається або кліком миші на них, або повторним викликом функції subplot. Наприклад, команда subplot(3, 2, 4) припускає наявність шістьох підграфіків і робить четвертий поточним.

У наступному прикладі показано, як можна вивести графіки функції $z(x,y) = 4 \cdot sin\,2\pi x \cdot cos\,1.5\pi y \cdot (1-x^2) \cdot y \cdot (1-y)$ різними способами. У заголовки включаються назви команд, застосовуваних для побудови графіків.

```
>> [X, Y] = meshgrid(-1:0.05:1,   0:0.05:1);
>> Z = 4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*Y.*(1-Y);
>> subplot(3, 2, 1)
>> mesh(X, Y, Z)
>> title('mesh')
>> subplot(3, 2, 2)
>> surf(X, Y, Z)
>> title('surf')
>> subplot(3, 2, 3)
>> meshc(X, Y, Z)
>> title('meshc')
>> subplot(3, 2, 4)
>> surfc(X, Y, Z)
>> title('surfc')
>> subplot(3, 2, 5)
>> contour3(X, Y, Z)
>> title('contour3')
>> subplot(3, 2, 6)
>> surfl(X, Y, Z)
>> shading interp
>> title('surfl')
>> colormap(gray)
```

У результаті формується графічне вікно, зображене на рис.25. Остання команда colormap(gray) змінює палітру всього графічного вікна, а не підграфіків окремо.

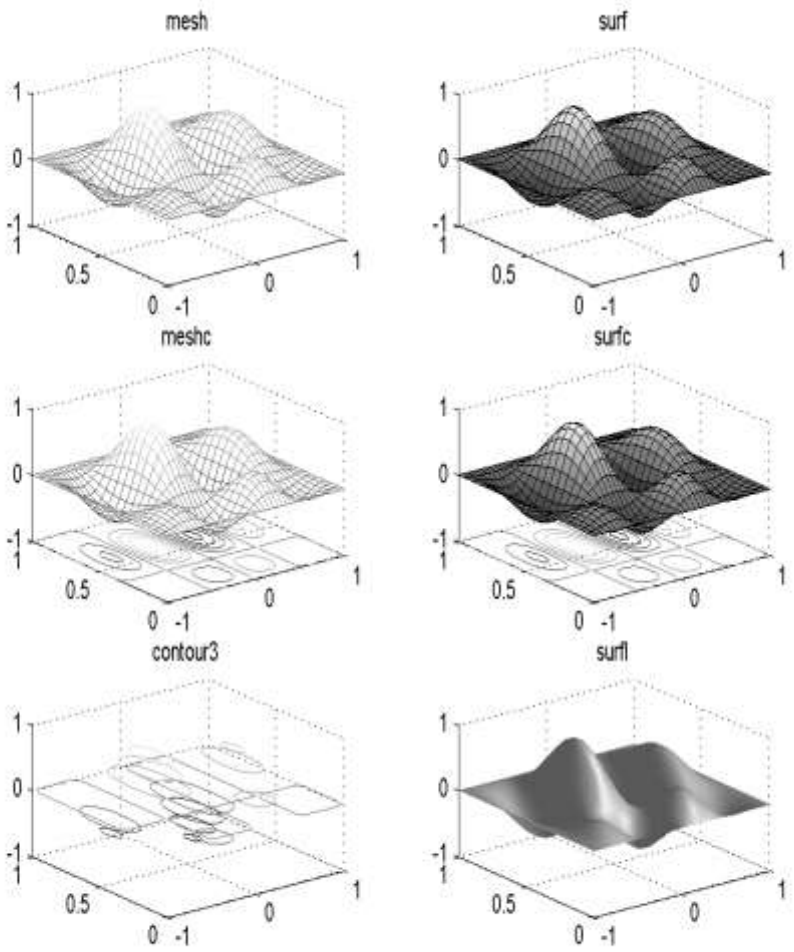


Рис.25 Приклад використання підграфіків **(**subplot**)**

4.7 ПОБУДОВА ГРАФІКІВ З ВІКНА Workspace

Візуалізувати графіки можна за допомогою засобів графічного інтерфейсу вікна Workspace. Повідомлення про масиви з'являються в цьому вікні по закінченні їхнього формування. Для побудови графіка необхідно виділити ці масиви кліком "миші" по імені масиву з утриманням <Ctrl>, і вибрати тип графіка. Можна також вибрати графік зі списку типів графіків, що розкривається, на панелі інструментів вікна Workspace.

При використанні такого способу візуалізації даних необхідно стежити за тим, щоб обрані дані відповідали одне одному. Наприклад, такою умовою є збіг довжин векторів x та y.

4.8 РЕДАГУВАННЯ ГРАФІКІВ

Для виклику редактора графіків використовується команда plottools. Редактор графіків має набір інструментів для побудови і редагування графіків.

Перейти в режим редагування графіка можна по кнопці EditPlot із зображенням курсору-стрілки. У цьому режимі графіком можна управляти за допомогою контекстного меню, яке викликається кліком правої кнопки "миші".

Проілюструвати можливості графіки можна на наступному прикладі, у якому задано вісім об'єктів – прямокутник fh (об'єкт класу figure), осі з мітками ah (об'єкт класу axes), тривимірна поверхня sh (об'єкт класу surface), три сфери різних розмірів, поверхня peaks і циліндр (рис.26).

```
>> [x, y] = meshgrid([-2:.4:2]);
>> Z=sin(x.^2+y.^2);
>> fh=figure('Position', [350 275 400 300], 'Color', 'w');
>> ah=axes('Color', [.8.8.8], 'XTick', [-2 -1 0 1 2], 'YTick', [-2 -1 0 1 2]);
>> sh = surface('XData',x,'YData',y,'ZData',Z,'FaceColor',get(ah,'Color')+.1, ...
'EdgeColor', 'k', 'Marker', 'o', 'MarkerFaceColor', [.5 1.85])
>> h(1) = axes('Position', [0 0 1 1]); sphere
>> h(2) = axes('Position', [0 0.4.6]); peaks;
>> h(3) = axes('Position', [0.5.5.5]); sphere
>> h(4) = axes('Position', [.5 0.4.4]); sphere
>> h(5) = axes('Position', [.5.5.5.3]); cylinder([0 0 0.5])
>> set(h, 'Visible', 'off'); set(gcf, 'Renderer', 'painters')
```

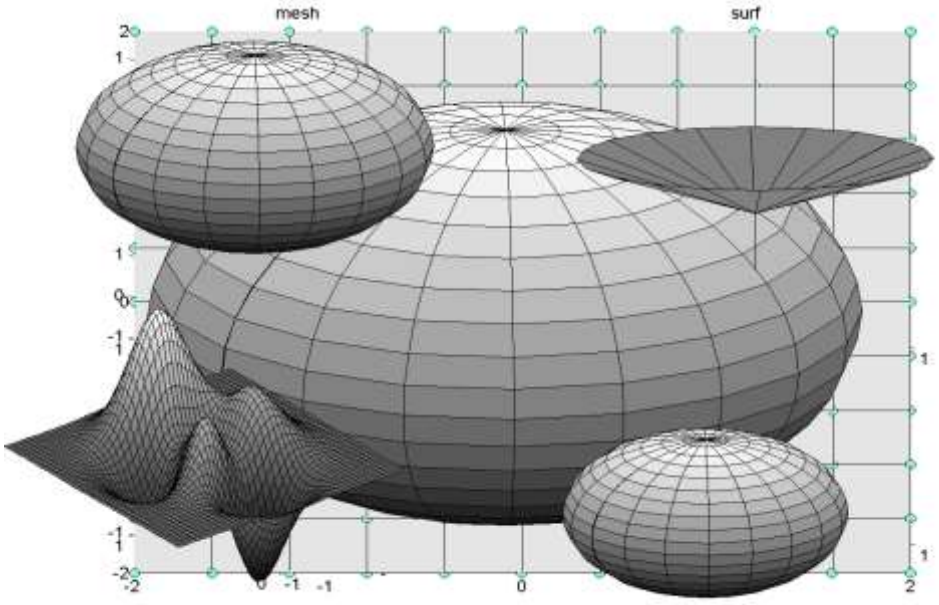


Рис.26 Приклад комбінованого графіка

4.9 ПОБУДОВА ГІСТОГРАМ

Гістограмой цифрового зображення називається дискретна функція $h(r_k) = n_k$, де r_k - це k-*й* рівень яскравості з можливого діапазону яскравості, наприклад від 0 до 255, а n_k - кількість пикселів зображення, рівень яскравості яких дорівнює значенню r_k. Оскільки індекси в системі MATLAB починаються з 1, а не з 0, тому r_1 відповідає рівню яскравості 0, r_2 відповідає рівню яскравості 1 і так далі до $r_{\max}$.

Гістограми нормуються діленням значення функції на загальне число пикселів зображення: $p(r_k) = \dfrac{h(r_k)}{n} = \dfrac{n_k}{n}$, при $k = 1,2,...,L.$

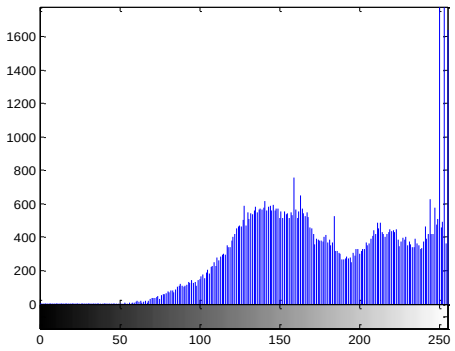
Для побудови гістограм використовується функція imhist(f, b), де f - це вихідне зображення, b - кількість груп рівнів яскравості, використаних при формуванні гістограми (за замовчуванням b=256).

Для одержання нормованої гістограми необхідно виконати команду: $p = \dfrac{imhist(f,b)}{numel(f)}$, тому що функція numel(f) повідомляє про кількість елементів масиву f.

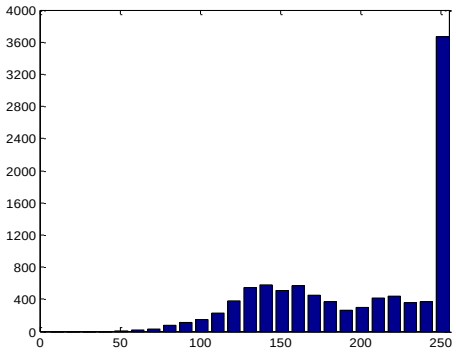
На рис.27б показаний результат роботи функції imhist(f) стосовно зображення на рис.27а.

Гістограма може бути побудована за допомогою стовпчастих діаграм. Для цього використана функція bar(horz, v, width), де v – вектор-рядок, графік якої потрібно побудувати, horz – вектор того ж розміру, що й вектор v, що складається із приростів по горизонтальній шкалі, а width – число між 0 і 1. При опущеному параметрі horz горизонтальна вісь ділиться із кроком 1 від 0 до length(v). При width=1 сусідні стовпчики стикаються, а при width=0 стовпчики є вертикальними лініями. За замовчуванням width=0.8.

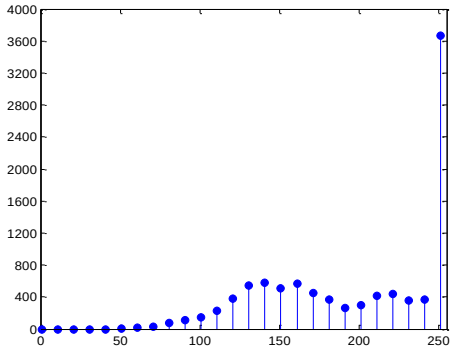
а)



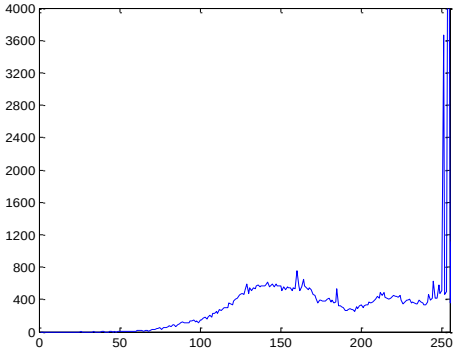
б)



в)



г)



д)

Рис.27 Побудова гістограм за різними методами

Програма побудови графіка на рис.27в містить команди:

```
f = imread('handkey.jpg');
h = imhist(f);
h1 = h(1:10 : 256);
horz = 1:10 : 256;
bar(horz, h1);
axis([0  255  0  4000]);
set(gca, 'xtick', 0:50 : 255)
set(gca, 'ytick', 0 : 400 : 4000)
```

У свою чергу, стебелева діаграма будується аналогічно стовпчастій діаграмі. Її синтаксис має вигляд stem(horz, v, ‘color_linestyle_marker’, ‘fill’), де v – вектор-стовпець, графік якого необхідно побудувати, а змінна horz має сенс, аналогічний у функції bar. Аргумент ‘color_linestyle_marker’ складається з комбінації символів, можливі комбінації яких наведені в табл.1 для функції plot. Наприклад, команда stem(v, ‘r - - s’) побудує стеблеву діаграму, на якій вертикальні стебла будуть проведені червоними пунктирними лініями, а голівки стебел будуть квадратними.

Стеблева діаграма на рис.27г отримана за допомогою команд:

```
f = imread('handkey.jpg');
h = imhist(f);
h1 = h(1:10 : 256);
horz = 1:10 : 256;
stem(horz, h1, 'fill');
axis([0  255  0  4000]);
set(gca, 'xtick', 0:50 : 255)
set(gca, 'ytick', 0 : 400 : 4000)
```

Результат застосування функції plot для побудови гістограми показаний на рис.27д, а програма містить команди:

```
f = imread('handkey.jpg');
h=imhist(f);
plot(h);
axis([0 255 0 4000]);
set(gca, 'xtick', 0:50:255)
set(gca, 'ytick', 0:400:4000)
```

4.10 ЗАВДАННЯ ДЛЯ САМОКОНТРОЛЮ

По заданій функції двох змінних необхідно:

1. Сформувати графіки функції різними способами:
   - каркасною поверхнею,
   - каркасною поверхнею, залитою кольорами,
   - маркіруваними лініями рівня (значення функції, відображувані лініями рівня, вибрати самостійно),
   - освітленою поверхнею.
2. Розташувати графіки в окремих графічних вікнах і в одному вікні з відповідним числом осей. Вказати вид каркасної або освітленої поверхні з декількох точок огляду.

4.11 ЗАПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

- Яким чином можна управляти видом графіків ?
- Яке призначення команди grid on ?
- Яке призначення функцій xlabel, ylabel, zlabel, title ?
- Які правила використання функції legend ?
- Як побудувати графіки функцій, заданих параметрично ?
- Як побудувати графік кусково-заданої функції ?
- Яке призначення команд hidden off і hidden on ?
- Яке призначення функцій surf і surfc ?
- Яке призначення команд shading interp і shading faceted ?
- Яке призначення команди colorbar ?
- Яке призначення функцій mesh і meshc?
- Яке призначення функції contour ?
- Яке призначення функції contourf ?
- Яке призначення функції contour3 ?

- Яким чином можна змінювати кольорове оформлення графіків ?
- Які правила використання в аргументах команд математичних позначень?
- Які правила використання грецьких букв і спеціальних символів ?
- Які правила повороту графіків ?
- Яке призначення функції surfl ?
- Як одержати зображення освітленої поверхні ?
- Як одержати анімований графік ?
- Як забезпечити виведення кожного графіка в окреме вікно ?
- Як забезпечити виведення декількох графіків на одні координатні осі в одне вікно ?
- Як забезпечити відображення в межах одного вікна декількох графіків, кожного на своїх осях ?

5. ДОДАТКОВІ ЗАВДАННЯ

- Згідно заданим векторам x і y побудувати багатокутник з координатами вершин (x_i, y_i).
- Відобразити елементи заданого вектора синіми маркерами, а максимальний елемент - червоним і повернути значення та номер максимального елемента.
- Підрахувати число входжень підрядка в рядок.
- Видалити підряд однакові символи, що йдуть, у рядку, утворивши новий масив з вилучених символів.
- Вивести номери однакових рядків у масиві рядків.
- Визначити кількість символів у кожному рядку масиву рядків без урахування пробілів.
- По заданому масиву рядків утворити новий масив, виключивши повторювані рядки.
- Виділити числа в числовий масив з рядка, що містить текст і числа.
- Утворити рядок, що складається з перших і останніх літер рядків, що входять у масив рядків.
- Замінити підряд однакові символи, що йдуть, у рядку на символ і число його повторень.
- По заданому масиву створити вектор, що містить значення масиву в порядку убування.
- По заданому масиву створити вектор, що містить значення масиву відповідно до зиґзаґоподібного обходу масиву.

СПІСОК ЛІТЕРАТУРИ

1. Сергиенко А.Б. Цифровая обработка сигналов: Учебник для вузов. – СПб.: Питер, 2006. – 751с.
2. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB. – М.: Техносфера, 2006. – 616с.
3. Дьяконов В.П. MATLAB 6.5 SP1/7 + Simulink 5/6. Обработка сигналов и проектирование фильтров. – М.: СОЛОН-Пресс, 2005. – 576с.
4. Дьяконов В.П. MATLAB 6.5 SP1/7/7 SP1 + Simulink 5/6. Работа с изображениями и видеопотоками. – М.: СОЛОН-Пресс, 2005. – 400с.
5. Ануфриев И.Е., Смирнов А.Б., Смирнова Е.Н. MATLAB 7. – СПб.: БХВ-Петербург, 2005. – 1104с.
6. Айфичер Э., Джервис Б. Цифровая обработка сигналов: практический подход.: Пер. с англ. – М.: Издательский дом "Вильямс", 2004. – 992с.