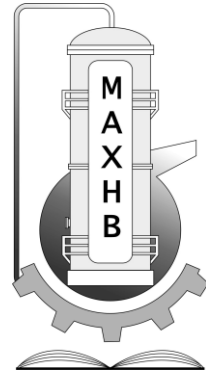


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»



МЕТОДИЧНІ ВКАЗІВКИ

ДО ПРОВЕДЕННЯ КОМП'ЮТЕРНОГО ПРАКТИКУМУ

З ДИСЦИПЛІНИ

“Методи комп'ютерного розрахунку”

Напрямок підготовки 133 – Галузеве машинобудування

**спеціалізація – Інжиніринг, комп'ютерне моделювання та проектування
обладнання хімічних і нафтопереробних виробництв**

Київ – 2017

Методичні вказівки до проведення комп'ютерного практикуму з дисципліни “Методи комп'ютерного розрахунку” для студентів напряму підготовки 133 – Галузеве машинобудування, спеціалізації – Інжиніринг, комп'ютерне моделювання та проектування обладнання хімічних і нафтопереробних виробництв: [Електронний ресурс]: / КПІ ім. Ігоря Сікорського; уклад Сачок Р.В. – Київ: КПІ ім. Ігоря Сікорського, 2017. – 112 с.

*Гриф надано Вченою радою
інженерно-хімічного факультету НТУУ „КПІ”
(Протокол № від січня 2017 р.)*

Для студентів інженерно-хімічного факультету.

Відповідальний редактор Степанюк Андрій Романович, доцент, к.т.н.

Рецензент:

Сокольський Олександр Леонідович, доцент, к.т.н.

Навчальне видання

Сачок Роман Володимирович, к.т.н., ст. викладач кафедри МАХНВ

МЕТОДИЧНІ ВКАЗІВКИ

ДО ПРОВЕДЕННЯ КОМП'ЮТЕРНОГО ПРАКТИКУМУ

з дисципліни

“Методи комп'ютерного розрахунку”

для студентів напряму підготовки 133 – Галузеве машинобудування, спеціалізації – Інжиніринг, комп'ютерне моделювання та проектування обладнання хімічних і нафтопереробних виробництв

Вступ

Шлях до впровадження у виробництво наукових розробок лежить через створення конструкторської документації. Розробка такої документації це творчий процес, який потребує від конструктора не тільки глибоких знань дисциплін, що викладаються у ВУЗі, але й уміння використовувати їх при проектуванні обладнання та проведенні розрахунків цього обладнання, що підтверджує його працездатність та надійність. Від якості конструкторської документації, як правило, залежить кінцевий результат наукової розробки та доля нових машин і апаратів. Проведення розрахунків для установок, апаратів та різних пристроїв є важливою частиною розробки технічної документації.

MathCAD є унікальною системою для проведення обчислень за визначеними виразами, з текстовою інформацією та графіками. Це гнучкий інструмент, призначений для вирішення широкого спектру задач, легкий в освоєнні і зручний у використанні. Виробник MathCAD – компанія MathSoftInc, впевнена, що її продукт є інструментом проведення наукових і технічних розрахунків для вчених і фахівців всього світу.

Mathcad має інтуїтивний і простий для використання інтерфейс користувача. Для введення формул і даних можна використовувати як клавіатуру, так і спеціальні панелі інструментів.

Mathcad відноситься до систем комп'ютерної алгебри, тобто засобів автоматизації математичних розрахунків. У цьому класі програмного забезпечення існує багато аналогів різної спрямованості і принципів побудови. Найбільш часто Mathcad порівнюють з такими програмними комплексами, як Maple, Mathematica, MATLAB, а також з їх аналогами MuPAD, Scilab, Maxima, тощо. Втім, об'єктивне порівняння ускладнюється у зв'язку з різним призначенням програм і ідеологією їх використання.

Методичні вказівки складено у відповідності до ДСТУ 3008-95.

Матеріал методичних вказівок викладено за авторами [1...8].

Частина I Використання середовища MathCad

1. Загальні характеристики та робоче вікно MathCAD.

Основні характеристики системи MathCAD:

- MathCAD дозволяє записувати на екрані комп'ютера формули в їх звичному вигляді.
- Виконувати символні перетворення і обчислення, визначати вид похідної будь-якої функції, а не тільки обчислювати її значення в заданих точках, знаходити визначені і невизначені інтеграли, вирішувати рівняння і системи рівнянь, виконувати операції над матрицями та векторами.
- Замінити комп'ютерні програми, що виконують різноманітні обчислення. Починаючи з версії 7.0 для професіоналів, є можливість запрограмувати певні дії, використовуючи такі керуючі конструкції, як умовні оператори, цикли, підпрограми, тощо.
- Замінити довідкові таблиці, наприклад таблиці інтегралів.
- Будувати двовимірні і тривимірні графіки, які можуть істотно полегшити подальші обчислення, гістограми і тривимірні стовпчасті діаграми для надання статистичної інформації.
- Дозволяє створювати документи з високою якістю оформлення.
- Допускає обмін даними з іншими додатками (наприклад, Excel, MATLAB та ін.).

Елементи вікна MathCAD

До елементів вікна MathCAD відносяться (Рис. 1.1):

- Рядок заголовка, яка містить назву програми та ім'я відкритого документа (за замовчуванням **Untitled:1**). При збереженні документа (див. «Збереження та друк документа») MathCAD за замовчуванням до заданого імені файлу додає розширення **MCD**.
- Рядок меню (головне меню).
- Кнопки на панелях інструментів.

– Вікно робочого листа, в якому формується MathCAD - документ. лінійки прокрутки забезпечують швидке переміщення по документу з метою виведення у вікні необхідних фрагментів документа.

– Рядок стану. На початку рядка виводиться призначення активного пункту меню. Натиснувши клавішу F1 викликається довідкова система. Тут же виводяться індикатор режиму обчислень і номер поточної сторінки документа.

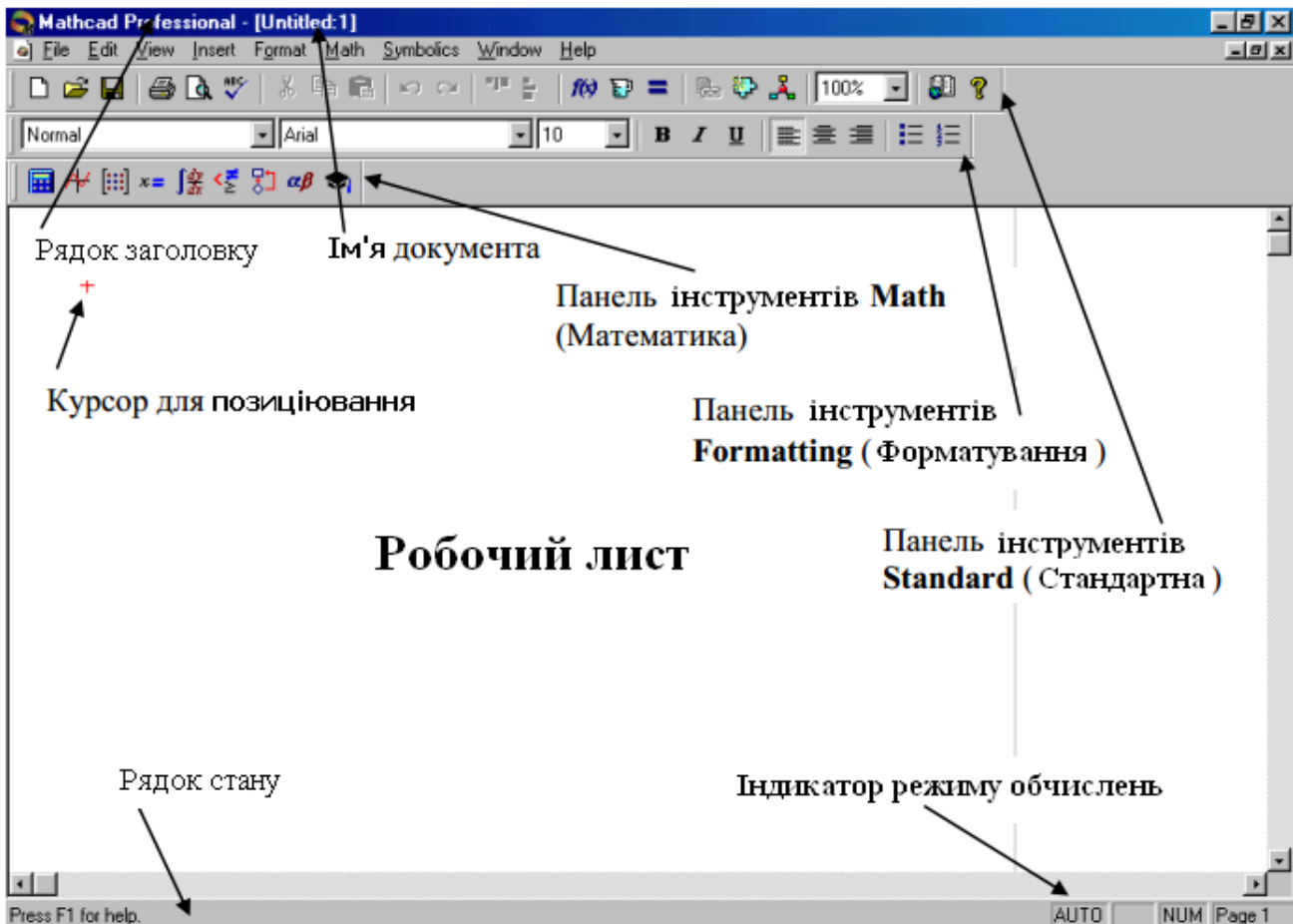


Рис. 1.1 – Елементи вікна MathCAD

Головне меню містить команди, за допомогою яких можна створювати та редагувати документ, забезпечуючи високу якість оформлення, необхідну при виведенні на друк, керувати режимом обчислень і параметрами середовища. Призначення пунктів меню дається в таблиці 1.1.

Команди, об'єднані в перерахованих групах, можуть бути активізовані за допомогою натискання відповідних кнопок, розміщених на панелі інструментів.

Таблиця 1.1 – Структура головного меню

Пункт меню	Призначення
File (Файл)	Команди для відкриття, збереження та виводу на друк файлів
Edit (Правка)	Команди редагування документу
Format (Формат)	Команди, які дозволяють задавати різноманітні параметри, що визначають зовнішній вид чисел, формул, тексту, колонтитулів, тощо.
Insert (Вставка)	Команди вставки в MathCAD– документ: графіків, матриць, функцій та інших об'єктів.
View (Вид)	Команди управління елементами екрану (панель інструментів, рядок стану, та інші).
Math (Математика)	Команди управління режимом обчислень та зміни параметрів обчислень.
Symbolics (Символи)	Команди символічних розрахунків.
Window (Вікно)	Команди для роботи з вікнами.
Help (Допомога)	Команди, що забезпечують доступ до довідкової системи.

1.1 Панелі інструментів

Основними панелями інструментів є:

- панель **Standard (Стандартні)**;
- панель **Formatting (Форматування)**;
- панель **Math (Математика)**.

Включення і виключення режиму відображення різних панелей інструментів здійснюється за допомогою команди **View (Вид)** з подальшим

вибором необхідного пункту меню. Рекомендується завжди мати включеними всі три перераховані панелі інструментів (див. рис.1.1).

Перші дві панелі містять кнопки, призначення яких однаково для будь-якого Windows-додатка (Рис. 1.2 вказано призначення деяких кнопок), тому докладно розглянемо тільки третю панель **Math**.

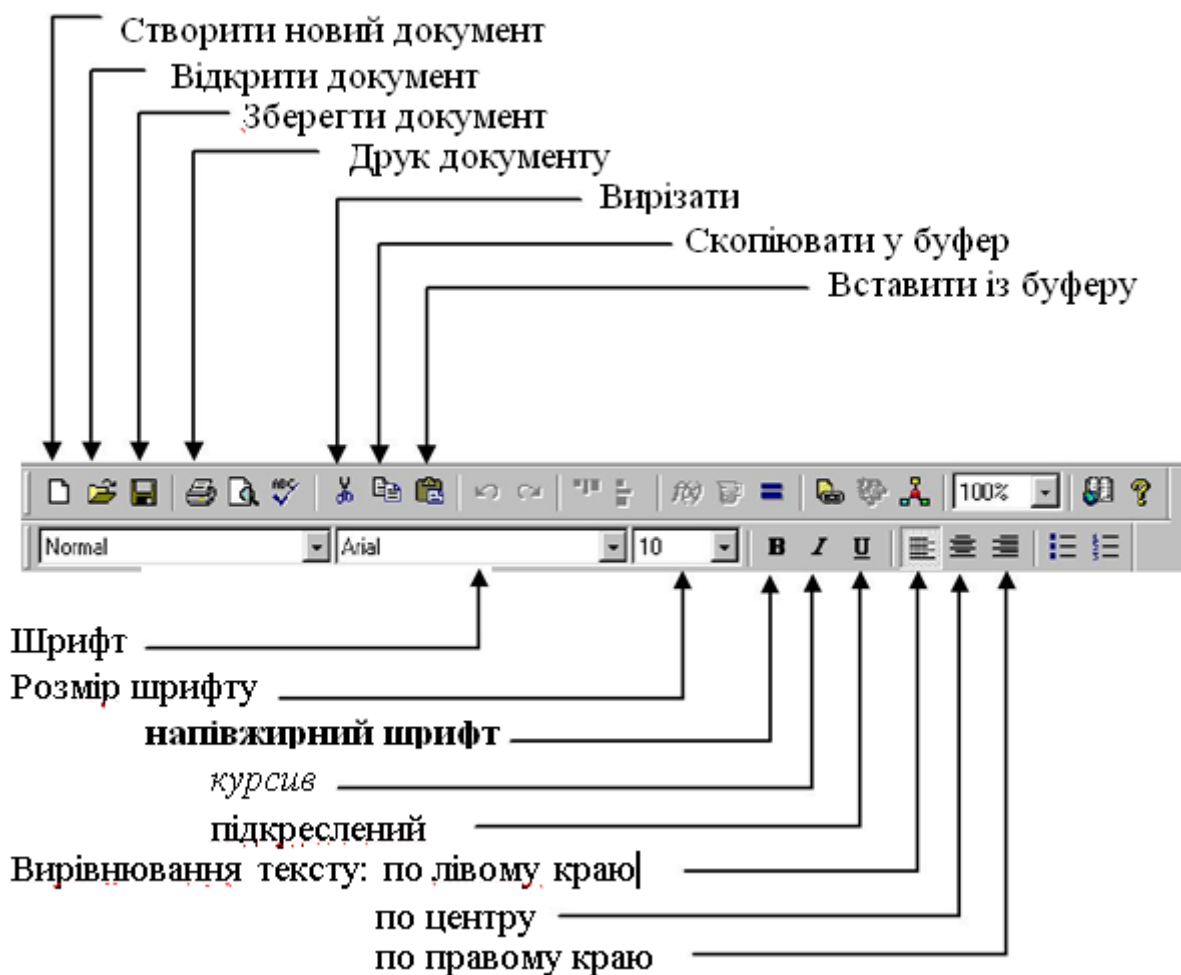


Рис. 1.2 – Склад панелей **Standard** і **Formatting**

Характеристика панелі інструментів **Math** (Математика)

Панель інструментів **Math** (Математика) містить кнопки для відображення ряду панелей інструментів, які використовуються для введення формул, побудови графіків, завдання символічних операцій. Загальна характеристика кнопок панелі дана в таблиці 1.2.

Таблиця 1.2 – Панель інструментів **Math (Математика)**

Піктограма	Назва панелі	Призначення панелі
	Calculator (Калькулятор)	Введення арифметичних операцій, деяких часто використовуваних функцій, операторів присвоювання і виведення результату.
	Boolean (Булева, Логічна)	Введення логічних операторів та операторів порівняння.
	Evaluation (Визначення, Обчислення,)	Введення операторів локального і глобального присвоювання і оператора результату.
	Graph (Графіка)	Побудова двох- і тривимірних графіків.
	Matrix (Матриці)	Введення векторів, матриць і операторів їх обробки.
	Calculus (Обчислення)	Введення операторів, обчислення похідних, інтегралів, сум, добутків і границь, тригонометричних функцій. Значення аргументу тригонометричних функцій задаються в радіанах.
	Greek (Грецький алфавіт)	Введення грецьких літер.
	Symbolic (Символи)	Задавання ключових слів для виконання символічних обчислень.
	Programming (Програмування)	Вставка операторів програмування.

На рисунку 1.3 показані склад і можливий варіант розміщення перерахованих у таблиці 1.2 панелей.

Деякі панелі можна розміщувати на тих же рядках вікна, де розміщені основні панелі інструментів, наприклад, як це показано на рисунку 1.4 (панель **Math** та панель **Calculus**, що входить до її складу, розміщені в одному рядку).

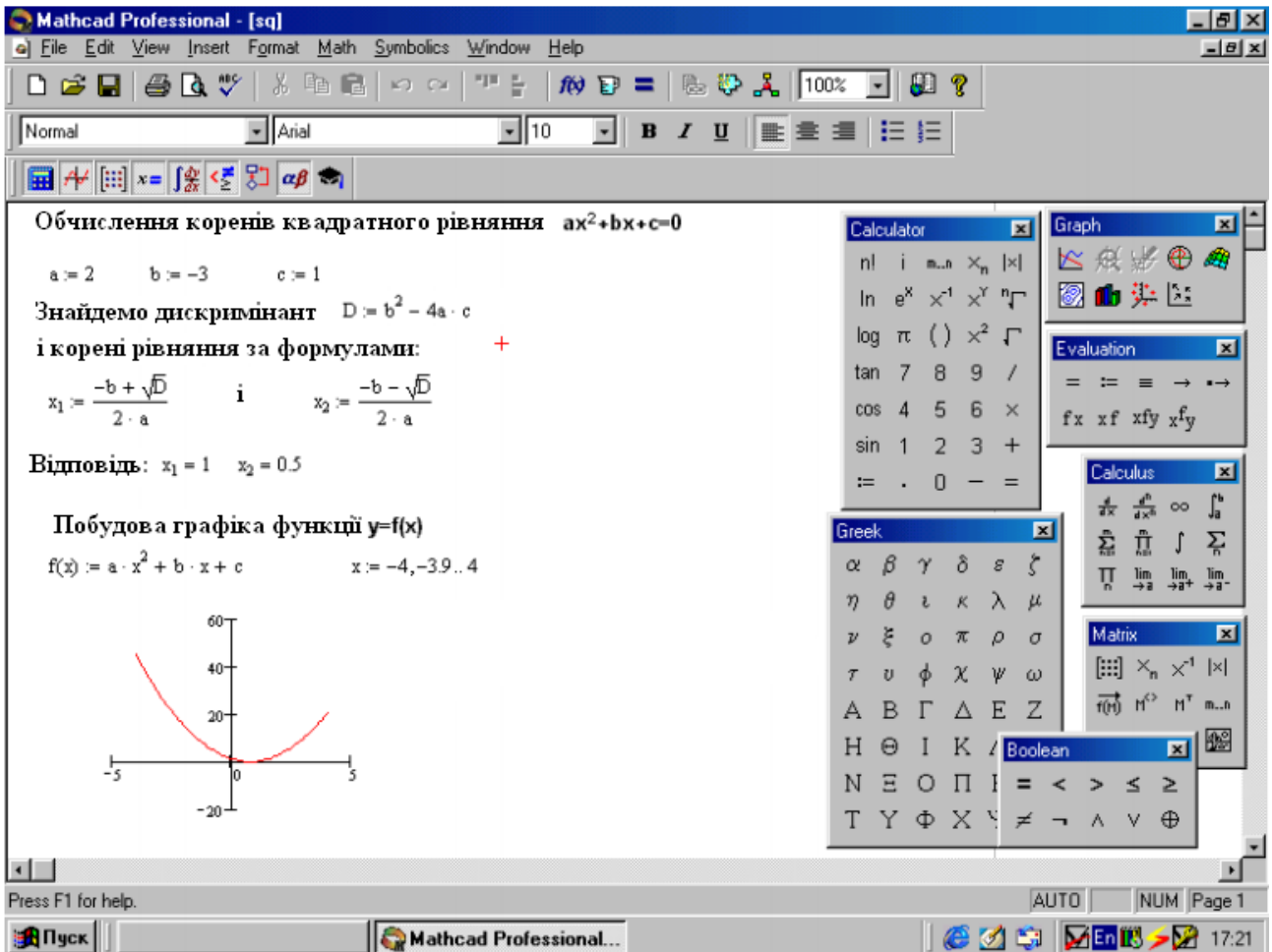


Рис. 1.3 – Структура основних панелей інструментів **Math**

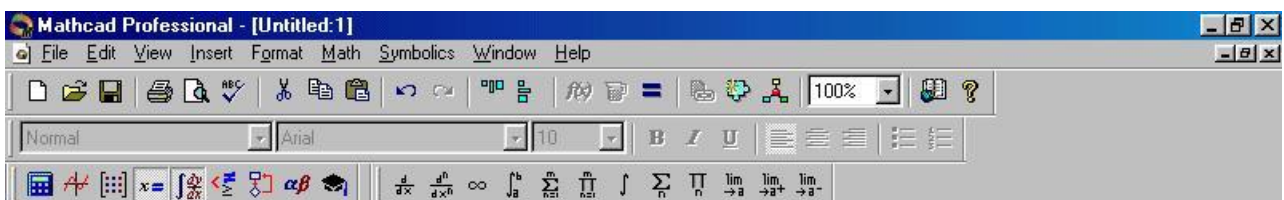


Рис. 1.4 – Можливий варіант розміщення панелей **Math** і **Calculus**

2 Елементи MathCAD

MathCAD-документ я сукупністю областей різних типів: формула, графік, текст. Приклад найпростішого документа наведено на рисунку 1.3. Кожна область має форму прямокутника і може розміщуватися в будь-якому місці робочого аркуша. Для позиціонування областей використовується курсор

(червоного кольору), що має хрестоподібну форму (Рис. 1.1). У текстовій області курсор має форму вертикальної лінії червоного кольору, а в області типу формула та графік – форму кутика синього кольору. Переміщення курсору виконується за допомогою клавіш управління курсором або переміщенням покажчика миші з подальшим одноразовим клацанням її лівої кнопки. Область, яка є активною або поточною (створюється, редагується), виділяється прямокутною рамкою. Рамка зникає, як тільки курсор виводиться з області.

До основних елементів математичних виразів MathCAD відносяться оператори, константи, змінні, масиви і функції.

2.1 Оператори

Оператори — елементи MathCAD, за допомогою яких створюють математичні вирази. До них, наприклад, відносять символи арифметичних операцій, знаки обчислення сумування, добутку, похідної, інтегралу, тощо.

Оператор визначає:

а) дію, яка повинна виконуватися при наявності тих, або інших значень операндів;

б) скільки, де і які операнди мають бути введені в оператор.

Операнд – число або вираз, на які діє оператор. Наприклад, у виразі $5!+3$ числа $5!$ і 3 – операнди оператора «+» (плюс), а число 5 – операнд факторіалу (!).

Будь-який оператор в MathCAD можливо ввести двома способами:

- натискуючи клавішу (комбінацію клавіш) на клавіатурі;
- використовуючи математичну панель.

Для присвоєння або виводу вмісту комірки пам'яті, що зв'язана із змінною, використовуються наступні оператори:

■ := ■ — знак присвоєння (вводиться натисканням клавіші: на клавіатурі (двокрапка на англійській розкладці клавіатури) або натисканням відповідної кнопки на панелі *Калькулятор*);

Таке присвоювання називається локальним. Присвоюється те значення, яке встановлюється в данному місці користувачем. До цього присвоювання змінна не визначена і її не можливо використовувати.

\approx — оператор наближеної рівності ($x1$). Використовується при розв'язанні систем рівнянь. Вводиться натисканням клавіші ; на клавіатурі (крапка з комою на англійській розкладці клавіатури) або натисканням відповідної кнопки на **Булевій панелі**.

Або оператор (дорівнює), відповідає за виведення значення константи або змінної. Присвоюється значення, яке було встановлено раніше або зарезервовано мовою MathCAD або встановлюється в данному місці користувачем.

\equiv – глобальний оператор присвоювання. Це присвоювання може проводитися в будь-якому місці документу. Наприклад, якщо змінній присвоєно таким чином значення в кінці документу, то вона буде мати це ж значення і на початку документу.

Найпростіші обчислення

Процес обчислення відбувається за допомогою наступних панелей:



панелі калькулятора **Calculator (Калькулятор)**;



панелі обчислень **Calculus (Обчислення, Визначення)**;



панелі оцінки. **Evaluation (Визначення)**

Увага. Якщо необхідно розділити весь вираз в чисельнику, то його потрібно спочатку виділити, натиснувши пробіл на клавіатурі або помістити в дужки.

2.2 Константи

Константи — іменовані об'єкти, що зберігають деякі значення, які не можуть бути змінені.

Наприклад, $\pi = 3.14$.

Розмірні константи — це загальноприйняті одиниці вимірювання. Наприклад, метри, секунди, та т. ін.

Щоб записати розмірну константу, необхідно після числа ввести знак * (помножити), вибрати пункт меню **Вставка** підпункт **Юніт**. У вимірюваннях найбільше відомі такі категорії: **Length** — довжина (м, км, см); **Mass** — маса (гр, кг, т); **Time** — час (хв, сек, год).

2.3 Змінні

Змінні є іменованими об'єктами, що мають певне значення, яке може змінюватися під час виконання програми. Змінні можуть бути числовими, строковими, символічними, та т. ін. Значення змінним задають за допомогою знаку присвоїти ($:=$).

Увага. В MathCAD заголовні та прописні літери сприймаються як різні ідентифікатори.

Системні змінні

В **MathCAD** міститься невелика група особливих об'єктів, які неможна віднести ні до класу констант, ні до класу змінних, значення яких визначенні зразу після запуску програми. Їх правильно рахувати *системними змінними*. Наприклад, **TOL [0.001]** – похибка чисельних розрахунків, **ORIGIN[0]** – нижня межа значення індексу індексації векторів, матриць. При необхідності значення цим змінним, можна задати інші величини.

Ранжируванні змінні

Ці змінні мають ряд фіксованих значень, або цілочисельних, або значень, які змінюються із певним кроком від початкового значення до кінцевого.

Для створення ранжируваної змінної використовується вираз:

$$\mathbf{Name} = \mathbf{N}_{\text{begin}}, (\mathbf{N}_{\text{begin}} + \mathbf{Step}) .. \mathbf{N}_{\text{end}}$$

де **Name** — ім'я змінної;

N_{begin} — початкове значення;

Step — крок зміни значення змінної;

N_{end} — кінцеве значення.

Ранжируванні змінні широко застосовуються при побудові графіків. Наприклад, для побудови графіку деякої функції $f(x)$ спочатку необхідно створити ряд значень змінної x — для цього вона повинна бути ранжируваною змінною.

Увага. Якщо в діапазоні змінної не вказувати крок, то програма автоматично встановить його рівним 1.

Приклад. Змінна x змінюється в діапазоні від -16 до $+16$ з кроком 0.1 .

Щоб записати ранжирувану змінну, необхідно ввести:

- назву змінної (x);
- знак присвоєння ($:=$);
- перше значення діапазону (-16);
- кому;
- друге значення діапазону, яке є сумою першого значення і кроку (тобто $-16+0.1$);
- знак ($..$) — зміна змінної в заданих межах (цей знак вводиться натисканням крапки з комою в англійській розкладці клавіатури);
- останнє значення діапазону (16).

В результаті отримаємо: $x := -16, -16+0.1..16$.

Увага. Як десятковий розподільник використовується крапка “.”.

Таблиці виводу

Будь-який вираз із ранжируемими змінними після знаку рівності ініціює таблицю виводу.

В таблиці виводу можна і вставляти числові значення і коректувати їх.

Змінна з індексом

Змінна з індексом — це змінна, якій присвоєно набір не пов'язаних одне з одним чисел, кожне із яких має свій номер (індекс).

Введення індексу здійснюється натисканням лівої квадратної дужки на клавіатурі або за допомогою кнопки x_n на панелі **Калькулятор**.

В якості індексу можна використовувати як константу, так і вираз. Для ініціалізації змінної з індексом необхідно ввести елементи масиву, розділяючи їх комами.

Приклад. Введення індексних змінних.

$i := 0..2$ — індекс змінюється від 0 до 2 (індексна змінна буде містити 3 елементи).

$s_i :=$

-2
2
5.75

— введення числових значень в таблицю відбувається через кому;

$s_0 = -2$ — вивід значення нульового елемента вектору S;

$s_1 = 2$ — вивід значення першого елемента вектору S.


2.4 Масиви

Масив — сукупність числових і символьних елементів, впорядкованих деяким чином, які мають визначені адреси та унікальне ім'я.

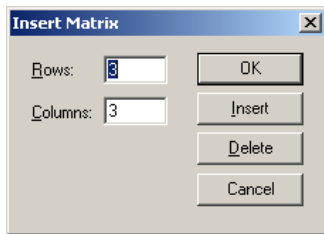
В пакеті **MathCAD** використовуються масиви двох найбільш поширених типів:

- одновимірні (вектори);
- двовимірні (матриці).

Вивести шаблон матриці або вектору можна одним із способів:

- вибрати пункт меню **Вставка - Матриця**;
- натиснути комбінацію клавіш **Ctrl + M**;
- натиснути кнопку  на **Панелі векторів і матриць**.

В результаті з'явиться діалогове вікно, в якому задається необхідне число рядків та стовпців:

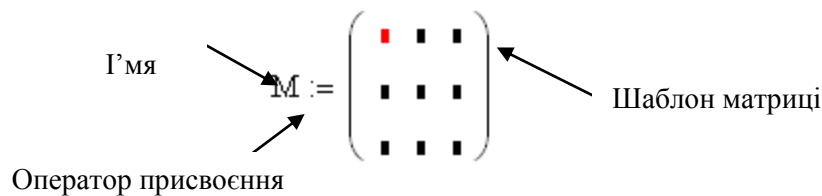


Rows — число рядків

Columns — число стовпців

Якщо матриці (вектору) потрібно присвоїти ім'я, то на початку вводиться ім'я матриці (вектору), а потім – оператор присвоєння, а в кінці – шаблон матриці.

Наприклад:




Матриця — двовимірний масив з іменем $\mathbf{M}_{n,m}$, що включає n рядків та m стовпців.

З матрицями можна виконувати різні математичні операції.

2.5 Функції

Функція — вираз, відповідно до якого проводяться певні обчислення із аргументами і визначається їх числове значення. Приклади функцій: **sin(x)**, **tan(x)**, тощо.

Функції в пакеті MathCAD можуть бути як вбудованими, так і визначеними користувачем. Способи вставки вбудованих функцій:

- Вибрати пункт меню **Вставка – Функція**.
- Натиснути комбінацію клавіш **Ctrl + E**.
- Натиснути  на панелі інструментів.
- Набрати ім'я функції на клавіатурі.

Функції користувача зазвичай використовуються при багатократних обчисленнях одного і того ж виразу. Для того, щоб задати функцію користувача необхідно:

– Ввести ім'я функції із обов'язковим записом аргумента в дужках, наприклад, $f(x)$.

– Ввести оператор присвоєння ($:=$).

– Ввести вираз для обчислення.

Приклад. $f(z) := \sin(2 \cdot z^2)$.

3. Форматування чисел

В MathCAD можна змінювати формат виводу чисел. Зазвичай обчислення проводяться з точністю 20 знаків, але виводяться на екран не всі значущі цифри. Щоб змінити формат числа, необхідно два рази натиснути на потрібному числовому результаті. З'явиться вікно форматування чисел, яке відкрите на вкладці **Number Format (Формат чисел)** із наступними форматами:

– **General (Основний)** – прийнятий за замовчуванням. Числа відображаються з порядком (наприклад, 1.22×10^5). Число знаків мантиси в полі **Exponential Threshold (Поріг експоненціального представлення)**. При перевищенні порогу число відображається з порядком. Число знаків після десяткового знаку (крапки) змінюється на поле **Number of decimal places**.

– **Decimal (Десятковий)** – десяткове представлення чисел з плаваючим десятковим знаком (наприклад, 12.2316).

– **Scientific (Науковий)** – числа відображаються тільки з порядком.

– **Engineering (Інженерний)** – числа відображаються тільки з порядком, кратним трьом (наприклад, 1.22×10^6).

Увага. Якщо після встановлення потрібного формату у вікні форматування чисел вибрати кнопку **Ок**, формат встановиться тільки для виділеного числа. А якщо вибрати кнопку **Set as Default**, формат буде застосований для всіх чисел даного документу.

Автоматично числа округляються до нуля, якщо вони менше встановленого порогу. Поріг встановлюється для всього документу, а не для конкретного результату. Для того щоб змінити поріг округлення до нуля, необхідно вибрати пункт меню **Форматування – Результат** і на вкладці **Tolerance**, в поле, що відкрилося **Zero threshold** ввести необхідне значення порогу.

4. Робота з текстом

Текстові фрагменти є частинами тексту, які користувач хотів би бачити в своєму документі. Це можуть бути пояснення, посилання, коментарі, тощо. Вони вставляються за допомогою пункту меню **Вставка – Текстовий регіон** (Рис. 4.1).

Дозволяється виконувати форматування тексту: поміняти шрифт, його розмір, вид, вирівнювання, тощо. Для цього потрібно його виділити і вибрати відповідні параметри на панелі шрифтів або в меню **Форматування – Текст**.

The screenshot shows the Mathcad software interface. The menu bar includes 'Файл', 'Правка', 'Вид', 'Вставка', 'Формат', 'Инструменты', 'Символьные операции', 'Окно', and 'Справка'. The 'Вставка' (Insert) menu is open, showing options like 'График', 'Матрица...', 'Функция...', 'Единица измерения...', 'Рисунок', 'Область', 'Разрыв страницы', 'Регион формул', and 'Регион текста'. The 'Регион текста' option is highlighted. Below the menu, there is a text input field with the text 'Розв'язати рівняння' and mathematical expressions: $x := 0.. \frac{\pi}{3}$ and $f(x) := \cos(2 \cdot x)$. A graph of the function $f(x) = \cos(2x)$ is shown, with the x-axis ranging from -2 to 2 and the y-axis from -1 to 0.5. The graph is annotated with a red line and the text 'Текстова область'. At the bottom, there is explanatory text: 'Розв'язання: При чисельному розв'язанні рівнянь бажано завжди будувати графік функції. Це допоможе уникнути помилок і правильно визначити потрібний інтервал локалізації коренів. Побудуємо графік нашої функції:'. Annotations with arrows point from the text 'Вставка текстових фрагментів' to the 'Регион текста' menu item and from 'Текстова область' to the graph area.

Рис. 4.1– Приклад виконання текстових даних

5. Робота з графіками

При вирішенні багатьох задач, де відбувається дослідження функції, часто виникає необхідність в побудові її графіка, де наглядно буде відображено поведінку функції на певному проміжку.

В системі MathCAD існує можливість побудови різних видів графіків: в Декартові та полярній системах координат, трьохвимірних графіків, поверхонь тіл обертання, багатокутників, просторових кривих, графіків векторного поля. Розглянемо прийоми побудови деяких з них.

5.1 Побудова двохвимірних графіків

Для побудови двохвимірного графіка функції необхідно:

- задати діапазон значень аргументу;
- задати функцію;

– встановити курсор в те місце документу, де має бути побудований графік, на математичній панелі вибрати кнопку **Graph (Графік)**, далі на відкритій панелі кнопку **X-Y Plot (Двохвимірний графік)**;

– в отриманому шаблоні двовимірного графіку, що є пустий прямокутник із мітками даних, в центральну мітку даних по осі абсцис (вісь X) ввести ім'я змінної, а на місті центральної мітки даних по осі ординат (вісь Y) ввести ім'я функції (Рис. 5.1);

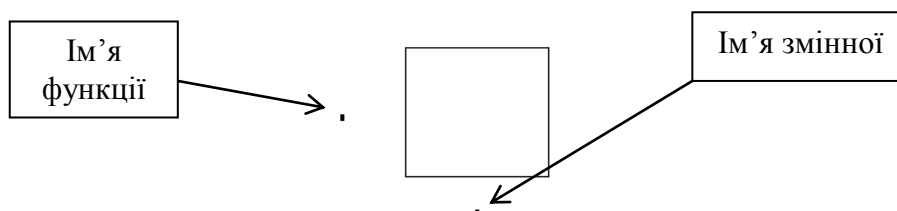


Рис. 5.1 – Шаблон двохвимірного графіку

– клацнути мишкою поза шаблоном графіку – графік функції буде побудовано.

Діапазон зміни аргументу складається із 3-х значень: початкове, друге і кінцеве.

Приклад. Нехай необхідно побудувати графік функції на інтервалі $[-2,2]$ з кроком 0.2. Значення змінної t задається у вигляді діапазону наступним чином:

$$t := -2, -1.8 .. 2,$$

де: 2 — початкове значення діапазону;

1.8 $(-2 + 0.2)$ — друге значення діапазону (початкове значення плюс крок);

2 — кінцеве значення діапазону.

Увага. Двокрапка вводиться натисканням крапки з комою на англійській розкладці клавіатури.

Приклад. Побудова графіка функції $y = x^2$ на інтервалі $[-5,5]$ з кроком 0.5. Вирішення наведено на рисунку 5.2.

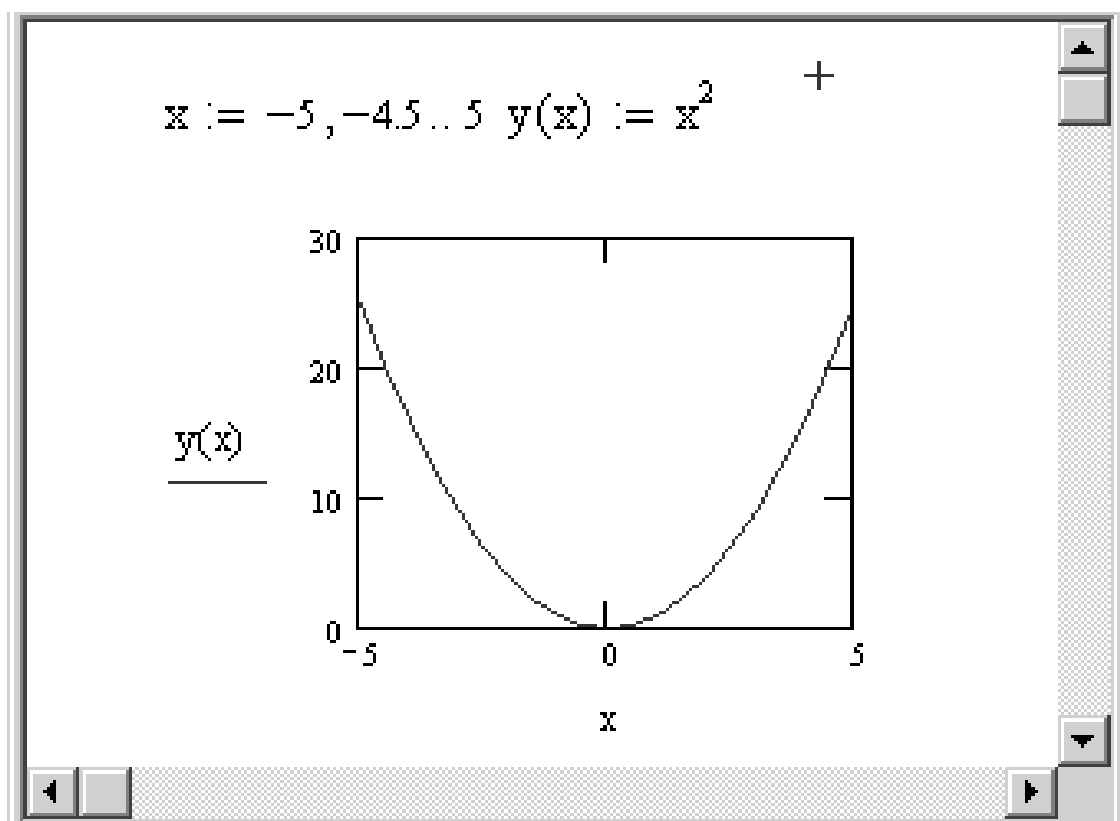


Рис.

5.2 – Побудова графіку функції $y = x^2$

При побудові графіків необхідно враховувати наступне:

– Якщо діапазон значень аргументу не заданий, то по замовчуванню графік будується в діапазоні $[-10,10]$.

– Якщо в одному шаблоні необхідно розмістити декілька графіків, то імена функцій вказуються через кому.

– Якщо дві функції мають різні аргументи, наприклад $f1(x)$ і $f2(y)$, то на осі ординат (Y) через кому вказуються імена функцій, а по осі абсцис (X) — імена обох змінних також через кому.

– Крайні мітки даних на шаблоні графіку служать для вказування граничних значень абсцис і ординат, тобто вони задають масштаб графіку. Якщо залишити ці мітки незаповненими, то масштаб буде встановлений автоматично. Автоматичний масштаб не завжди відображає графік в потрібному виді, тому граничні значення абсцис і ординат приходить редагувати, змінюючи вручну.

Примітка. Якщо після побудови графік не приймає потрібний вигляд, можна:

- Зменшити крок.
- Змінити інтервал побудови графіку.
- Зменшити на графіку граничні значення абсцис і ординат.

Приклад. Побудова кола з центром в точці $(2,3)$ і радіусом $R = 6$.

Рівняння кола з центром в точці з координатами (x_0, y_0) і радіусом R записується у вигляді:

$$(x - x_0)^2 + (y - y_0)^2 = R^2$$

Виразимо із цього рівняння y :

$$(y - y_0)^2 = R^2 - (x - x_0)^2$$
$$y - y_0 = \pm \sqrt{R^2 - (x - x_0)^2}$$
$$y = \pm \sqrt{R^2 - (x - x_0)^2} + y_0$$

Таким чином, для побудови кола необхідно задати дві функції: верхнього і нижнього півкола. Діапазон значень аргументу обчислюється наступним чином:

- початкове значення діапазону буде $x_0 - R$;
- кінцеве значення діапазону = буде $x_0 + R$;
- крок краще взяти рівним 0.1

Вирішення наведено на рисунку 5.3.

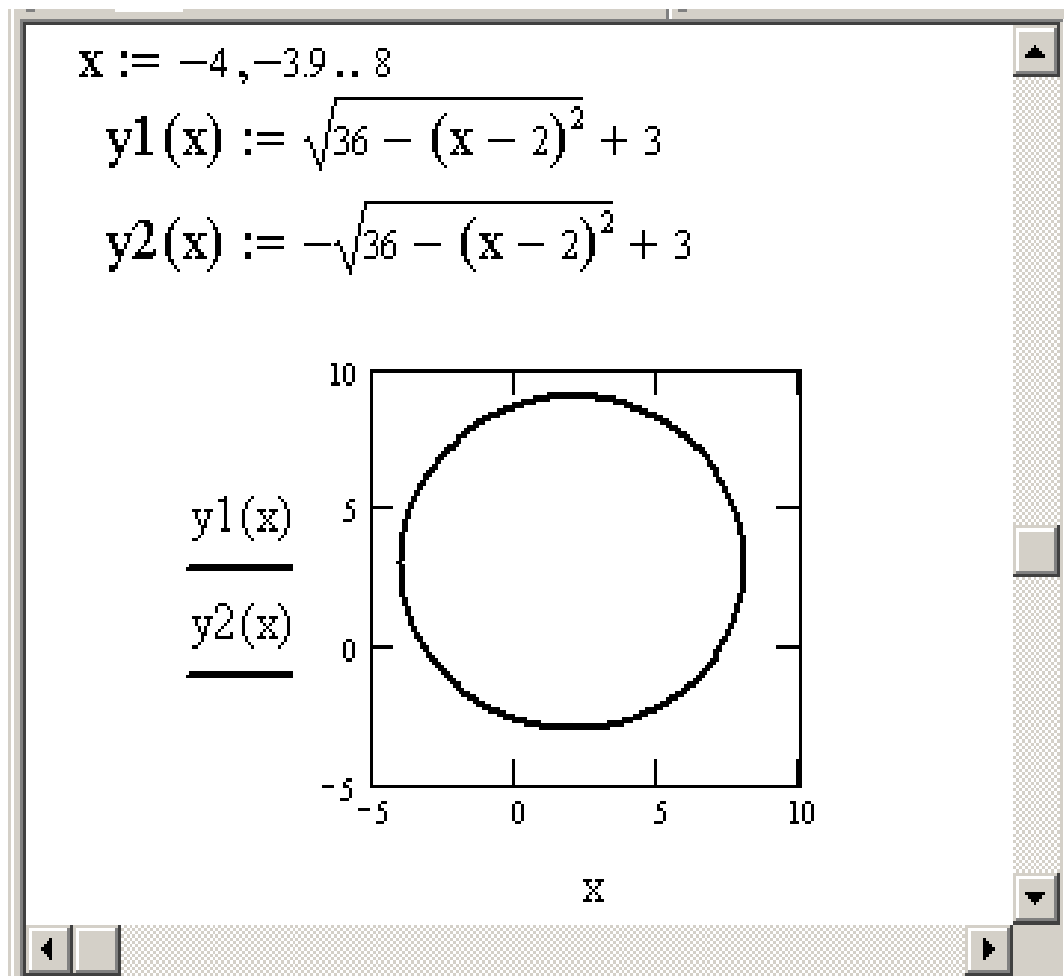


Рис. 5.3 – Побудова кола

Параметричний графік функції

Іноді краще замість рівняння лінії, що зв'язує координати x і y , розглядати так звані параметричні рівняння лінії, що дають вираз поточних координат x і y у вигляді функцій від деякої змінної величини t (параметра): $x(t)$ і $y(t)$. При побудові параметричного графіку на осях ординат і абсцис вказуються імена функцій одного аргументу.

Приклад. Побудова кола з центром в точці з координатами $(2,3)$ і радіусом $R = 6$. Для побудови використовується параметричне рівняння кола:

$$x = x_0 + R\cos(t) \quad y = y_0 + R\sin(t)$$

Вирішення наведено на рисунку 5.4.

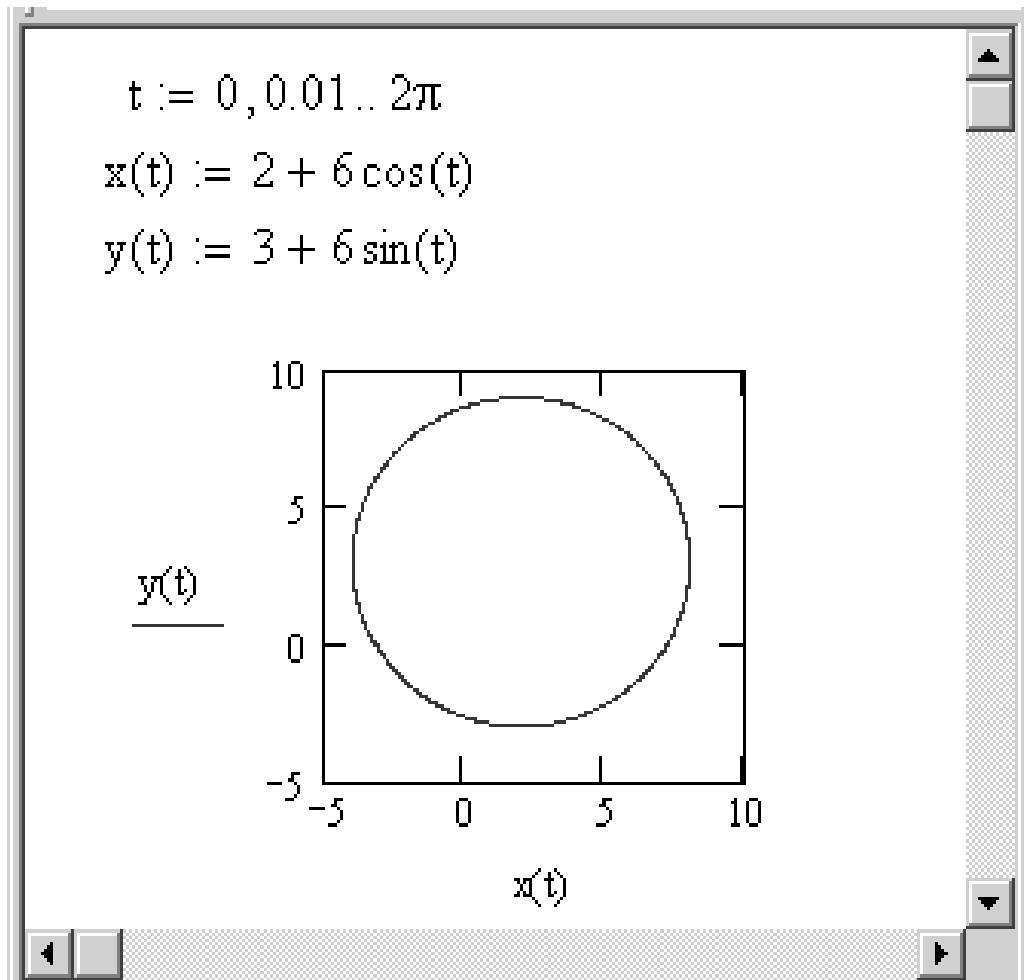


Рис. 5.4 – Побудова кола, використовуючи параметричне рівняння

Форматування графіків

Щоб відформатувати графік, необхідно два рази натиснути по області графіку. Відкриється діалогове вікно форматування графіку. Нижче перераховані вкладки вікна форматування графіку:

– **X-Y Axes** – форматування осей координат.

Встановивши необхідні прапорці можна:

– **Log Scale** – представлення числових значень на осях в логарифмічному масштабі (за замовчуванням числові значення наносяться в лінійному масштабі).

– **Grid Lines** – нанести сітку ліній.

– **Numbered** – розмістити числа по координатним осям.

– **Auto Scale** – автоматичний вибір граничних числових значень на осях (якщо цей прапорець знятий, граничними будуть максимальні обчислені значення).

– **Show Marker** – нанесення міток на графік у вигляді горизонтальних або вертикальних пунктирних ліній, що відповідають вказаному значенню на осі, причому самі значення виводяться вкінці ліній (на кожній осі з'являються два місця вводу, в які можна ввести чисельні значення, можна не вводити нічого, ввести лише одне число або буквені позначення констант).

– **Auto Grid** – автоматичний вибір числа ліній сітки (якщо цей прапорець знятий, необхідно задати число ліній в полі **Number of Grids**).

– **Crossed** – вісь абсцис проходить через нуль ординати.

– **Boxed** – вісь абсцис проходить по нижньому краю графіка.

– **Trace** – форматування ліній графіків функцій. Для кожного графіка окремо можна змінити:

– символ (**Symbol**) на графіку для вузлових точок (кружок, хрестик, прямокутник, ромб);

– вид ліній (**Solid** – суцільна, **Dot** – пунктир, **Dash** – штрихи, **Dadot** – штрих–пунктир);

– колір лінії (**Color**);

– тип (**Type**) графіку (**Lines** – лінія, **Points** – точки, **Bar** або **Solidbar** – стовпчики, **Step** – ступінчастий графік, та т.ін.);

– товщину ліній (**Weight**).

– **Label** – заголовок в області графіка. В поле **Title** (**Заголовок**) можна записати текст заголовка, вибрати його положення – зверху або знизу графіка (**Above** – зверху, **Below** – знизу). Можна вписати, якщо треба, назву аргументу і функції (**Axis Labels**).

– **Defaults** – за допомогою цієї вкладки можна вернутися до виду графіку, прийнятому за замовчуванням (**Change to default**), або зроблені вами зміни на графіку використати за замовчуванням для всіх графіків даного документу (**Use for Defaults**).

5.2 Побудова полярних графіків

Для побудови полярного графіка функції необхідно:

- задати діапазон значень аргументу;
- задати функцію;
- встановити курсор в те місце документу, де має бути побудований графік, на математичній панелі вибрати кнопку **Graph (Графік)** і, у відкритій панелі, кнопку **Polar Plot (Полярний графік)**;
- в місті введення шаблону необхідно ввести кутовий аргумент функції (внизу) і ім'я функції (зліва).

Приклад. Побудова лемніскати Бернуллі: $\rho = \sqrt{2 \cos(2\phi)}$. Вирішення наведено на рисунку 5.5.

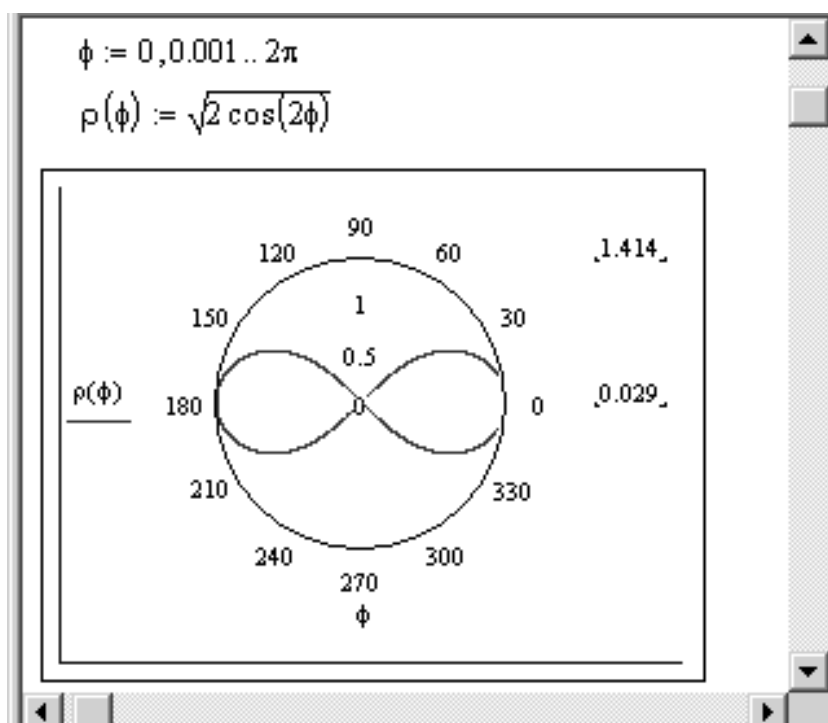


Рис. 5.5 – Приклад побудови полярного графіку

5.3 Побудова графіків поверхонь (трьохвимірних або 3D-графіки)

При побудові трьохвимірних графіків використовується панель **Graph (Графік)** математичної панелі. Трьохвимірний графік можна побудувати декількома способами:

- за допомогою майстра, що визивається із головного меню;
- створивши матрицю значень функції двох змінних;
- використати швидкий метод побудови;
- визвати спеціальні функції **Create Mech** і **Create Spase**, призначені

для створення масиву значень функції і побудови графіку.

Розглянемо швидкий метод побудови трьохвимірного графіку.

Швидка побудова графіку.

Для швидкої побудови трьохвимірного графіку функції необхідно:

- задати функцію;
- встановити курсор в те місце, де має бути побудований графік, на

математичній панелі вибрати кнопку **Graph (Графік)** і в відкритій панелі

кнопку  (**Поверхневий графік**);

- лише в одне місце шаблону введіть ім'я функції (не вказуючи змінні);
- клацніть лівою клавішею миші поза шаблоном графіку – графік функції

буде побудований.

Приклад. Побудова графіку функції $z(x,y) = x^2 + y^2 - 30$. Вирішення наведено на рисунку 5.6.

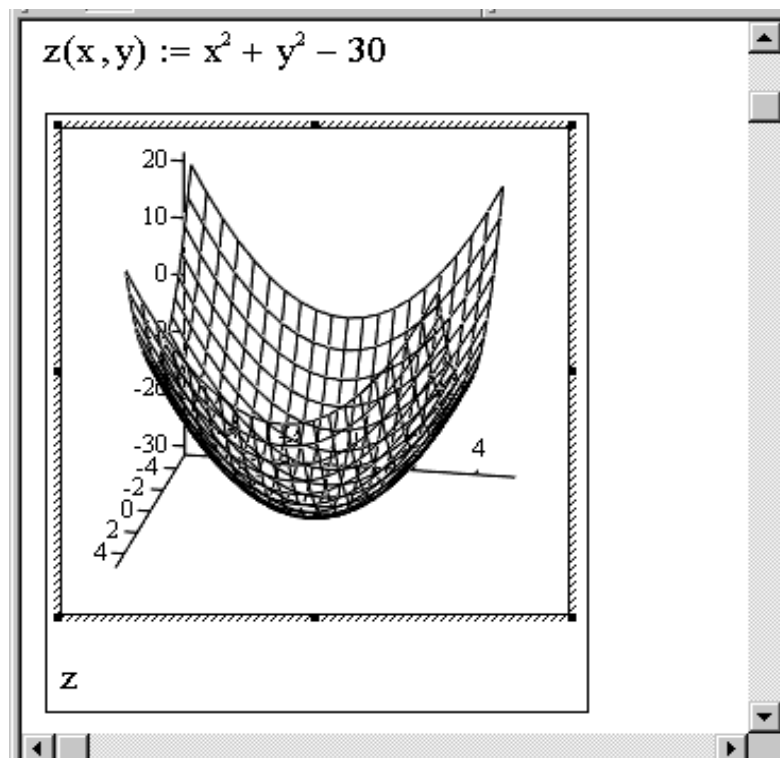


Рис. 5.6 – Приклад «швидкої» побудови поверхневого графіку

Побудованим графіком можна управляти:

- обертання графіку виконується після наведення на нього покажчика миші та наступному затисканні лівої клавіші миші;
- масштабування графіка виконується після наведення на нього покажчика миші при одночасному натисненні лівої кнопки миші і клавіші **Ctrl** (якщо рухати мишу, графік наближається або віддаляється);
- анімація графіка виконується аналогічно, але при натиснутій додатково клавіші **Shift**. Необхідно тільки почати обертання графіка мишею, далі анімація буде виконуватися автоматично. Для зупинки обертання слід клацнути лівою кнопкою миші всередині області графіка.

Існує можливість побудови відразу декількох поверхонь на одному малюнку. Для цього необхідно задати обидві функції і через кому вказати імена функцій на шаблоні графіка.

При швидкій побудові графіка за замовчуванням вибираються значення обох аргументів в межах від -5 до +5 і число контурних ліній, дорівнює 20. Для зміни цих значень необхідно:

- два рази натиснути по графіку лівою клавішею миші;
- у відкритому вікні вибрати вкладку **Quick Plot Data**;
- ввести нові значення в області вікна **Range1** — для першого аргументу і **Range2** — для другого аргументу (start — початкове значення, end — кінцеве значення);
- в полі **# of Grids** змінити число ліній сітки, що покривають поверхню;
- натиснути кнопку **Ok**.

Приклад. Побудова графіку функції $z(x,y) = -\sin(x^2 + y^2)$. Вирішення наведено на рисунку 5.7.

При побудові цього графіку границі зміни значень обох аргументів краще вибрати від -2 до +2.

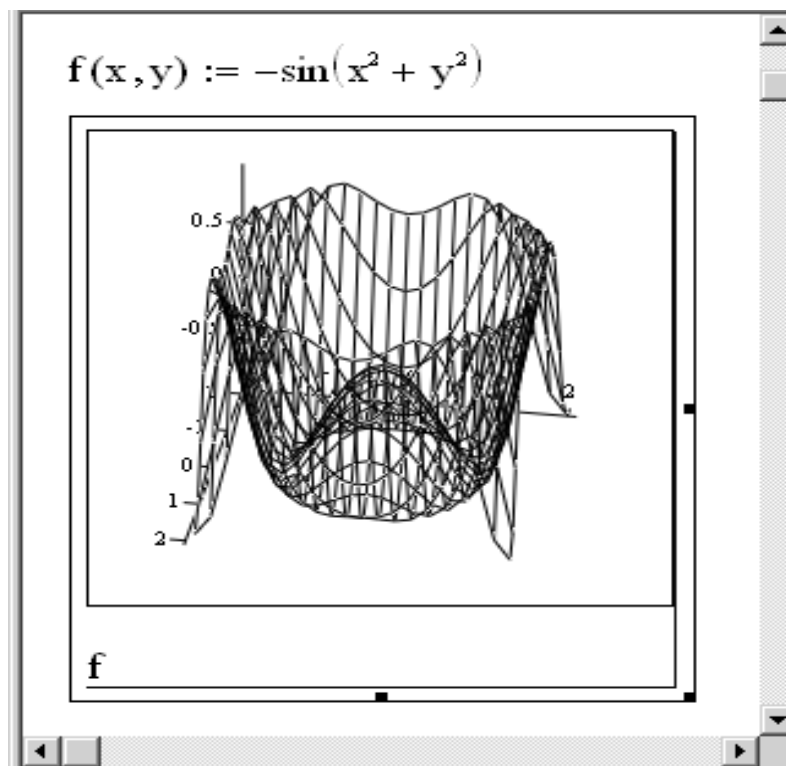


Рис. 5.7 – Приклад побудови графіку функції $z(x,y) = -\sin(x^2 + y^2)$

Форматування трьохвимірних графіків

Для форматування графіку необхідно два рази натиснути лівою кнопкою миші по області побудови – з’явиться вікно форматування із декількома вкладками: **Appearance**, **General**, **Axes**, **Lighting**, **Title**, **Backplanes**, **Special**, **Advanced**, **Quick Plot Data**.

Призначення вкладки **Quick Plot Data** було розглянуте вище.

Вкладка **Appearance** дозволяє змінювати зовнішній вид графіка. Поле **Fill Options** дозволяє змінити параметри заливки, поле **Line Option** — параметри ліній, **Point Options** — параметри точок.

На вкладці **General (Загальні)** в групі **View** можна вибрати кути повороту зображення поверхні навколо всіх трьох осей; в групі **Display as** можна змінити тип графіку.

На вкладці **Lighting (Освітлення)** можна керувати освітленням, встановивши прапорець **Enable Lighting (Увімкнути освітлення)** і перемикач **On (Увімкнути)**. Одна із 6-ти можливих схем освітлення вибирається в списку **Lighting scheme (Схема освітлення)**.

6. Способи розв'язання рівнянь в MathCAD

У даному розділі описано, яким чином в системі MathCAD вирішуються найпростіші рівняння виду $F(x) = 0$. Вирішити рівняння аналітично – значить знайти всі його корені, тобто такі числа, при підстановці яких у вихідне рівняння отримаємо рівність.

Вирішити рівняння графічно – значить знайти точки перетину графіка функції з віссю OX.

6.1 Розв'язання рівнянь за допомогою функції $\text{root}(f(x),x)$

Для рішень рівняння з одним невідомим виду $F(x)=0$ існує спеціальна функція:

$$\text{root}(f(x),x),$$

де $f(x)$ — вираз, що дорівнює нулю;

x — аргумент.

Ця функція повертає із заданою точністю значення змінної, при якому вираз $f(x)$ дорівнює 0.

Увага. Якщо права частина рівняння $\neq 0$, то необхідно привести його до нормального виду (перенести все в ліву частину).

Перед використанням функції **root** необхідно задати аргументу x початкове наближення. Якщо коренів декілька, то для пошуку кожного кореня необхідно задавати своє початкове наближення.

Увага. Перед вирішенням бажано побудувати графік функції, щоб перевірити, чи є корені (чи перетинає графік функції вісь OX), і якщо є, то скільки. Початкове наближення можна вибрати за графіком якнайближче до точки перетину.

Приклад. Розв'язати рівняння $x^3 = 15x$ за допомогою функції **root**.

Вирішення наведено на рисунку 6.1. Перед тим як приступати до розв'язання в системі MathCAD, в рівнянні все перемістимо в ліву частину. Рівняння набуде виду: $x^3 - 15x = 0$.

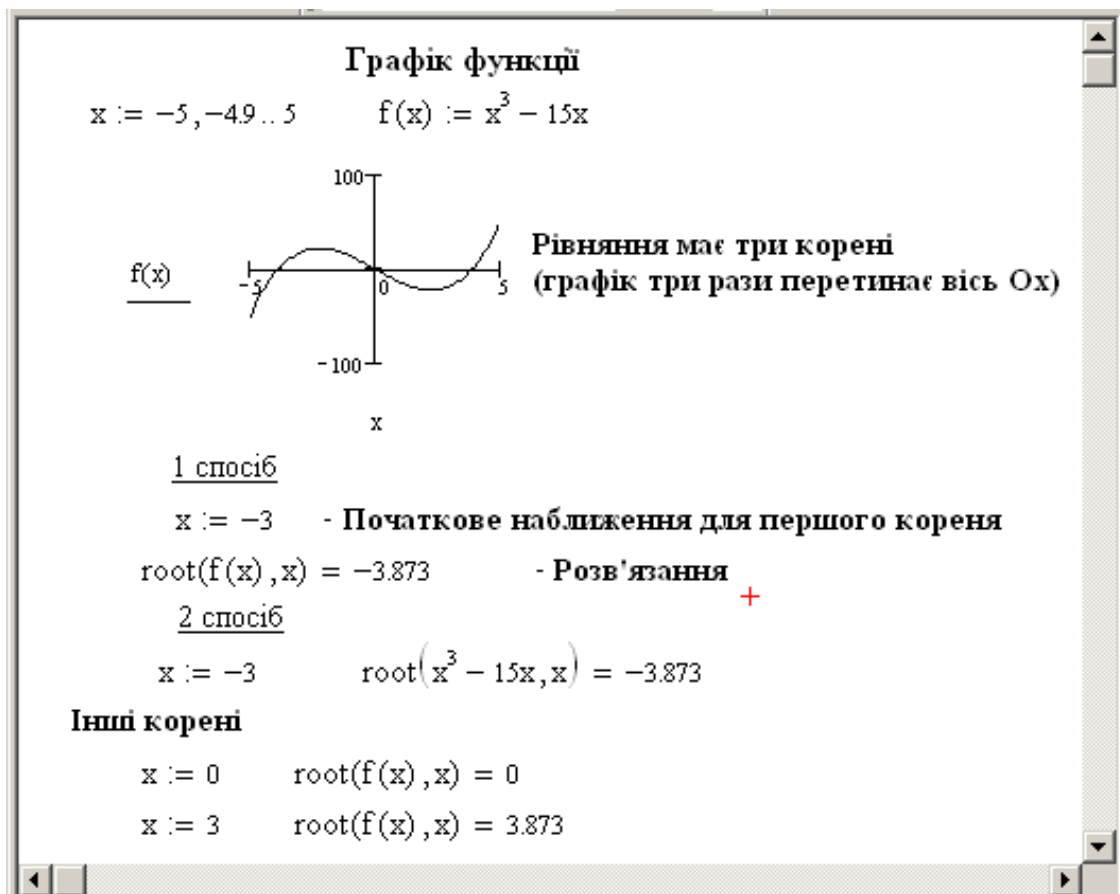


Рис. 6.1 – Вирішення рівняння за допомогою функції root

6.2 Розв'язання рівнянь за допомогою функції Polyroots(v)

Для одночасного знаходження всіх коренів поліному використовують функцію **Polyroots(v)**, де v — вектор коефіцієнтів поліному, починаючи із вільного члену. Нульові коефіцієнти пропускати необхідно. На відміну від функції **root** функція **Polyroots** не потребує початкового наближення.

Приклад. Розв'язання рівняння $0.75 \cdot x^3 - 8 \cdot x + 5 = 0$ за допомогою функції **polyroots** зображено на рисунку 6.2.

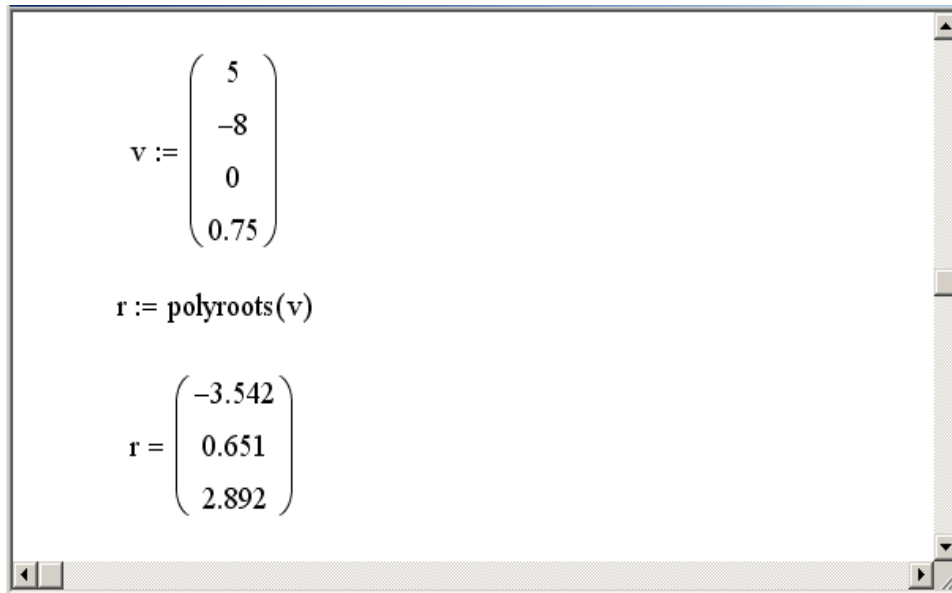


Рис. 6.2 – Розв’язання рівняння за допомогою функції polyroots

6.3 Розв’язання рівнянь за допомогою функції Find (x)

Функція **Find** (Знайти) працює в разом з ключовим словом **Given** (Дано). Конструкція **Given - Find** використовує розрахункову методику, засновану на пошуку кореня поблизу точки початкового наближення, заданої користувачем.

Якщо задано рівняння $f(x) = 0$, то його можна вирішити за допомогою блоку **Given - Find** наступним чином:

- задати початкове наближення:

$$x := x_0$$

- ввести службове слово:

Given

- записати рівняння, використовуючи знак жирне дорівнює:

$$f \mathbf{=} 0$$

- написати функцію **Find** з невідомою змінною як параметр:

$$\text{find}(x)=$$

В результаті після знаку дорівнює виводиться знайдений корінь.

Якщо існує декілька коренів, то їх можна знайти, міняючи початкове наближення x_0 на близьке до шуканого кореню.

Приклад. Розв'язання рівняння $x^2 + 8 = e^x$ за допомогою функції **find** зображено на рисунку 6.3.

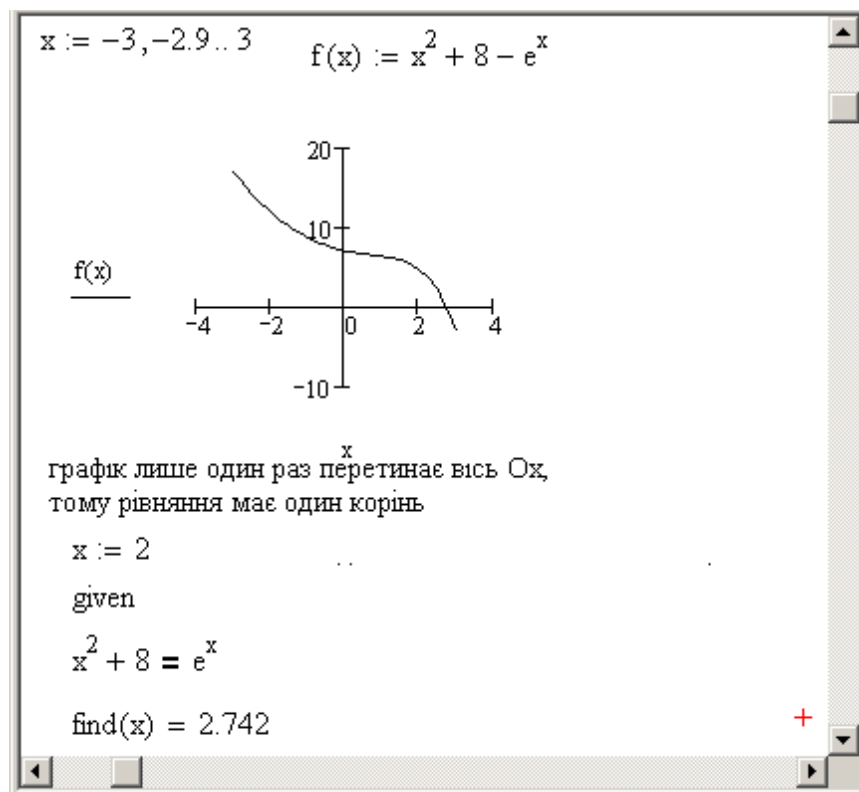


Рис. 6.3 – Розв'язання рівняння за допомогою функції **Find**

Іноді виникає необхідність відзначити на графіку якісь точки (наприклад, точки перетину функції з віссю Oх). Для цього необхідно:

- вказати значення x даної точки (по осі Oх) і значення функції в цій точці (по осі Oy);
- двічі клацнути по графіку і у вікні форматування у вкладці **Traces** для відповідної лінії вибрати тип графіка - points, товщину лінії - 2 або 3.

Приклад. На графіку відмічено точка перетину функції $f(x) = x^2 + 8 - e^x$ із віссю Oх. Координата x цієї точки була знайдена в попередньому прикладі: $x = 2.742$ (корінь рівняння). Вирішення наведено на рисунку 6.4.

У вікні форматування графіка у вкладці **Traces** для trace2 змінені: тип графіка - points, товщина лінії - 3, колір - чорний.

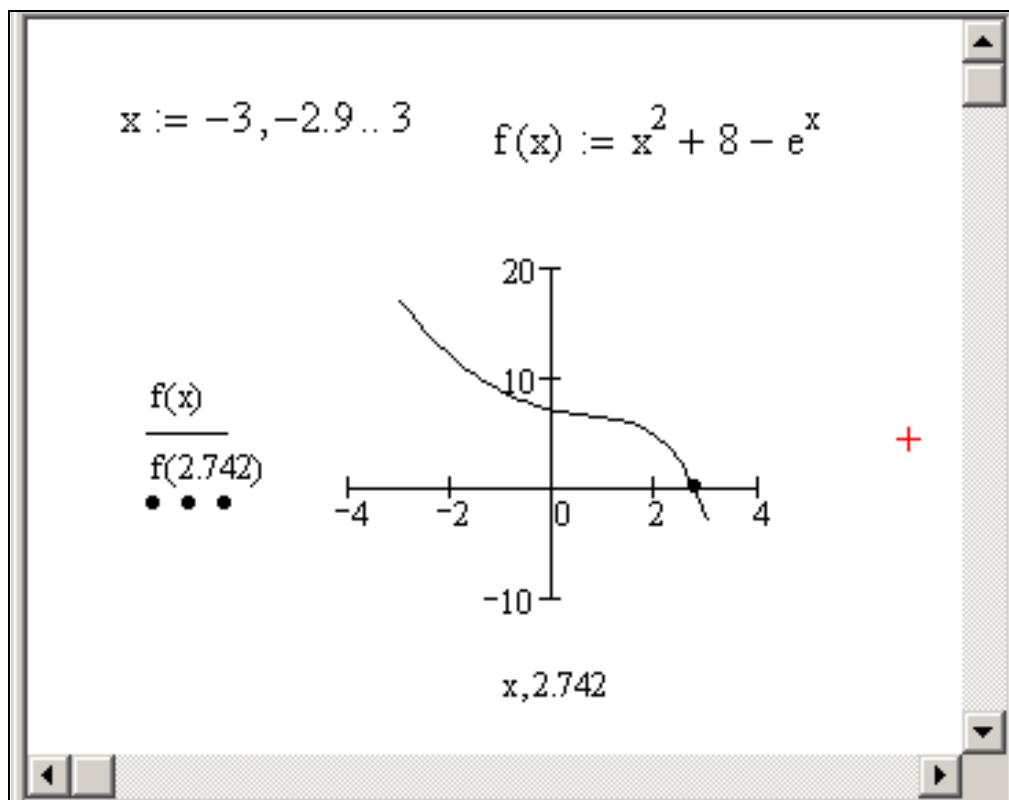


Рис. 6.4 – Графік функції $f(x) = x^2 + 8 - e^x$ із відміченою точкою перетину осі

7. Розв'язання систем рівнянь

7.1 Розв'язання систем лінійних рівнянь

Систему лінійних рівнянь можна вирішити *матричним методом* (або через зворотню матрицю, або використовуючи функцію **Isolve** (A, B)) з використанням двох функцій: функції **Find** і функції **Minerr**.

Матричний метод

Приклад. Дано систему рівнянь:

$$\begin{cases} 3x_1 + 4x_2 = 5 \\ 5x_1 + 4x_2 = 3 \end{cases}$$

Рішення даної системи рівнянь матричним методом зображено на рисунку 7.1.

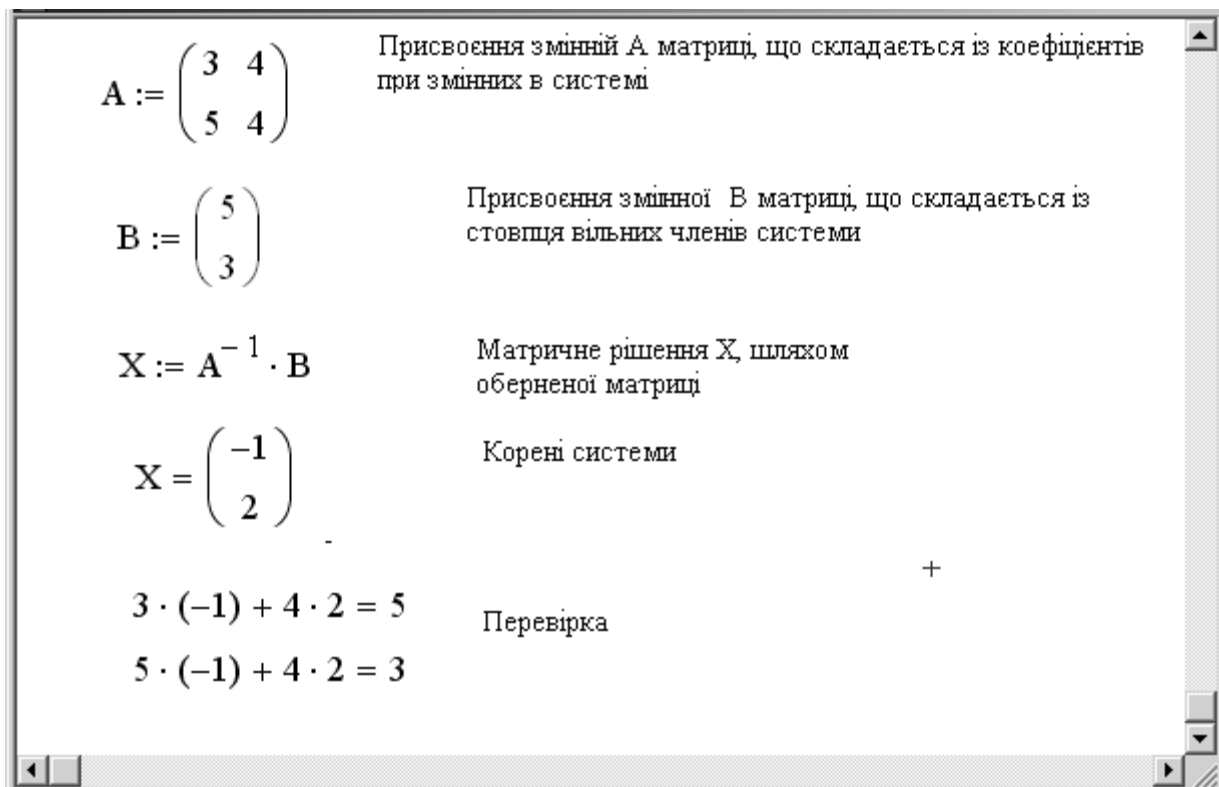


Рис. 7.1 – Рішення системи лінійних рівнянь матричним методом

Використання функції **lsolve(A,B)**

Lsolve (A, B) — це вбудована функція, яка повертає вектор X для системи лінійних рівнянь при заданій матриці коефіцієнтів A і векторі вільних членів B.

Приклад. Дано систему рівнянь:

$$\begin{cases} 1.2357 x_1 + 2.1742 x_2 - 5.4834 x_3 = 1 \\ 6.0696 x_1 - 6.2163 x_2 - 4.6921 x_3 = 1. \\ 3.4873 x_1 + 6.1365 x_2 - 4.7483 x_3 = 1 \end{cases}$$

Спосіб вирішення даної системи з використанням функції **lsolve** (A, B) наведено на рисунку 7.2.

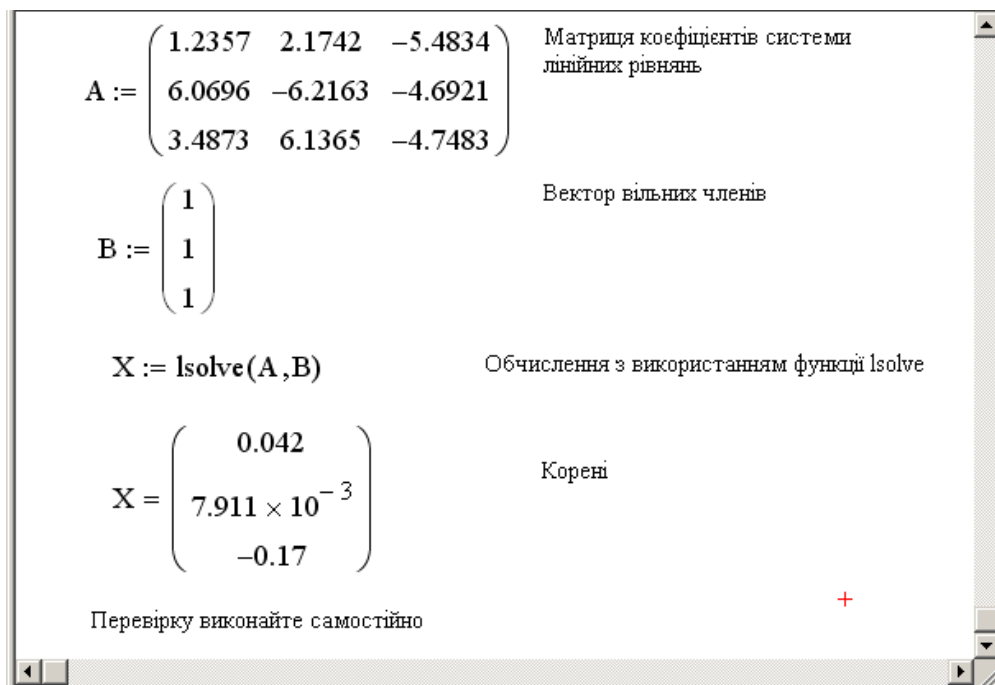


Рис. 7.2 – Рішення системи лінійних рівнянь з використанням функції **lsolve**

Рішення системи лінійних рівнянь за допомогою функції **Find**

При цьому методі рівняння вводяться без використання матриць, тобто в «натуральному вигляді». Попередньо необхідно вказати початкові наближення невідомих змінних. Це можуть бути будь-які числа, що входять в область визначення. Часто за них приймають стовпець вільних членів.

Для того щоб вирішити систему лінійних рівнянь за допомогою обчислювального блоку **Given - Find**, необхідно:

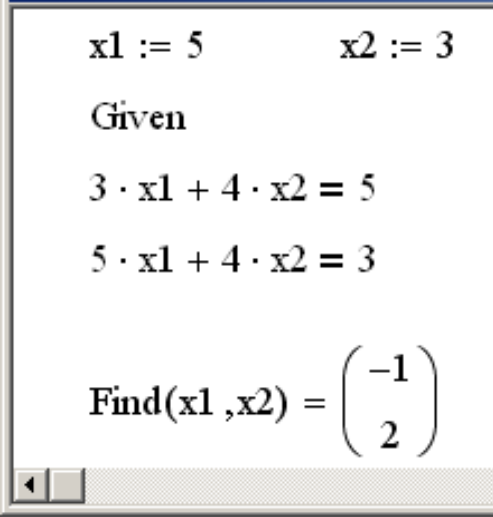
- 1) задати початкові наближення для всіх змінних;
- 2) ввести службове слово **Given**;
- 3) записати систему рівнянь, використовуючи знак **жирне дорівнює** (\equiv);
- 4) написати функцію **Find**, перерахувавши невідомі змінні в якості параметрів функції.

У результаті розрахунків виводиться вектор рішення системи.

Приклад. Дано систему рівнянь:

$$\begin{cases} 3x_1 + 4x_2 = 5 \\ 5x_1 + 4x_2 = 3 \end{cases}$$

Рішення даної системи за допомогою обчислювального блоку **Given - Find** наведено на рисунку 7.3.



```
x1 := 5      x2 := 3

Given

3 · x1 + 4 · x2 = 5
5 · x1 + 4 · x2 = 3

Find(x1 ,x2) =  $\begin{pmatrix} -1 \\ 2 \end{pmatrix}$ 
```

Рис. 7.3 – Рішення системи лінійних рівнянь за допомогою функції **Find**

Наближене рішення системи лінійних рівнянь

Рішення системи лінійних рівнянь за допомогою функцію **Minerr** аналогічно рішення за допомогою функції **Find** (використовується той же алгоритм), тільки функція **Find** дає точне рішення, а **Minerr** - наближене. Якщо в результаті пошуку не може бути отримано подальше уточнення поточного наближення до розв'язку, **Minerr** повертає це наближення. Функція **Find** в цьому випадку повертає повідомлення про помилку.

Загальні рекомендації за рішенням рівнянь і систем рівнянь

У випадку, коли MathCAD не може самостійно знайти рішення:

- Можна підібрати інше початкове наближення.
- Можна збільшити або зменшити точність розрахунків. Для цього в

меню вибрати **Math ► Options (Математика – Опції)**, вкладка **Built-In Variables (Вбудовані змінні)**. У вкладці, необхідно зменшити допустиму похибку обчислень (**Convergence Tolerance (TOL)**). За замовчуванням **TOL = 0.001**.

Увага. При матричному методі рішення необхідно розставити коефіцієнти згідно зростанню невідомих $x1, x2, x3, x4$.

7.2 Рішення систем нелінійних рівнянь

Системи нелінійних рівнянь в MathCAD вирішуються за допомогою обчислювального блоку **Given - Find**.

Конструкція **Given - Find** використовує розрахункову методику, засновану на пошуку кореня поблизу точки початкового наближення, заданої користувачем.

Для вирішення системи рівнянь за допомогою блоку **Given - Find** необхідно:

- задати початкові наближення для всіх змінних;
- ввести службове слово **Given**;
- записати систему рівнянь, використовуючи знак **жирне дорівнює** (\equiv);
- описати функцію **Find**, перерахувавши невідомі змінні в якості параметрів функції.

В результаті розрахунків виведеться розв'язання системи у векторному вигляді.

Якщо система має кілька рішень, алгоритм слід повторити з іншими початковими наближеннями.

Примітка. Якщо вирішується система з двох рівнянь з двома невідомими, перед вирішенням бажано побудувати графіки функцій, щоб перевірити, чи є корені у системи (чи перетинаються графіки заданих функцій), і, якщо є, то скільки. Початкове наближення можна вибрати за графіком найближче до точки перетину.

Приклад. Дано систему рівнянь:

$$\begin{cases} y = x^2 + 14 \\ y = 7x + 45 \end{cases}$$

Перед вирішенням системи побудуємо графіки функцій: параболічної (перше рівняння) і прямолінійної (друге рівняння). Побудова графіків в одній системі координат наведено на рисунку 7.4:

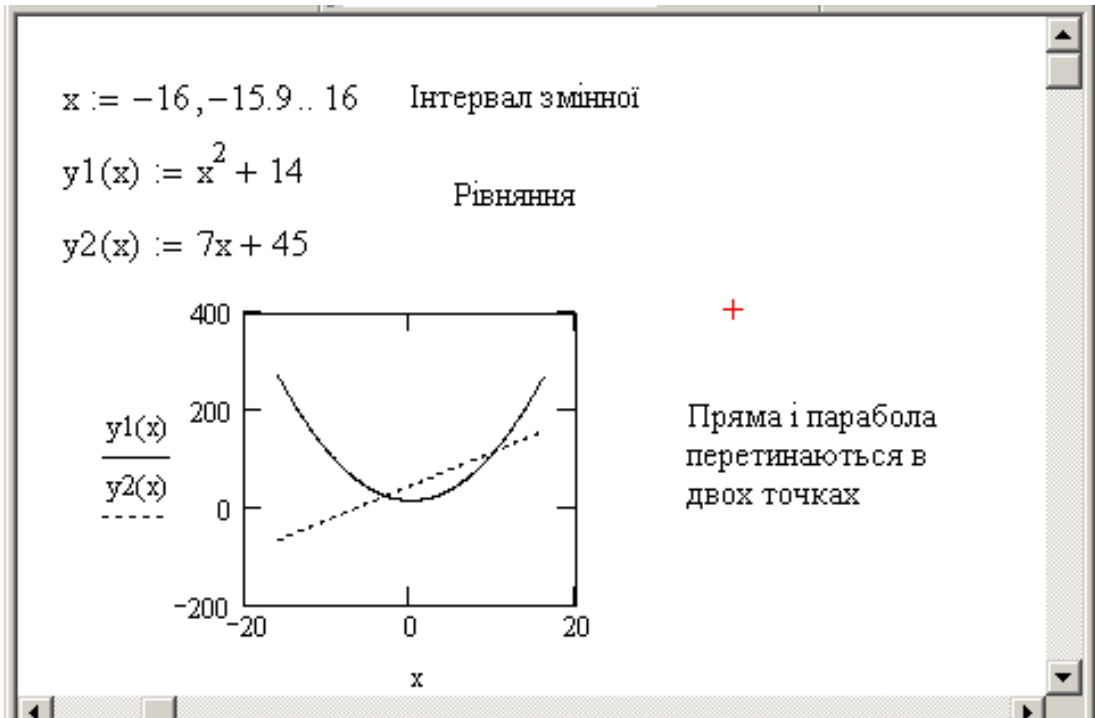


Рис. 7.4 – Побудова графіка двох функцій в одній системі координат

Пряма і парабола перетинаються в двох точках, значить, система має два кореня. За графіком вибираємо початкові наближення невідомих x і y для кожного рішення. Знаходження коренів системи рівнянь наведено на рисунку 7.5.

```

x := -4    y := 0
given
y = x^2 + 14    y = 7x + 45
  ( x1 )
  ( y1 ) := find(x, y)

  ( x1 ) = ( -3.076 )
  ( y1 ) = ( 23.465 )

x := 10    y := 50
Given
y = x^2 + 14    y = 7x + 45
  ( x2 )
  ( y2 ) := Find(x, y)

  ( x2 ) = ( 10.076 )
  ( y2 ) = ( 115.535 )
  
```

Рис. 7.5 – Знаходження коренів системи нелінійних рівнянь

Для того, щоб відзначити на графіку координати точок перетину параболи і прямої, знайдені при вирішенні системи, введемо через кому значення x (по осі Ox) та значення y (по осі Oy). У вікні форматування графіка у вкладці **Traces** для **trace3** і **trace4** змінимо: тип графіка – **points**, товщина лінії – 3, колір – чорний (Рис. 7.6).

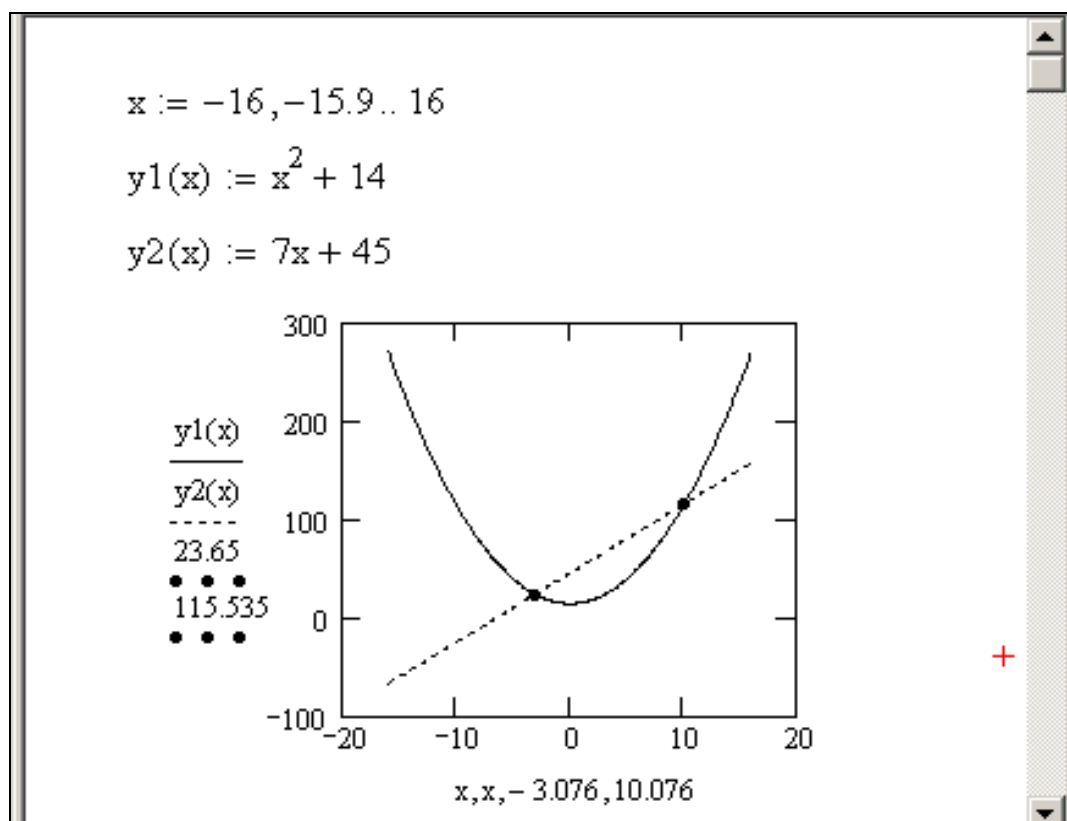


Рис. 7.6 – Графіки функцій з відміченими точками перетину

8 Програмування в пакеті Mathcad

Програмування – це створення блоку, в якому поєднано різноманітні операції. За допомогою програмних блоків створюють розгалужені програми з циклами. Функції із ранжованими змінними та операторами **if** дають змогу використовувати умовні вирази та організувати цикли. Проте їх можливості обмежені.

Блоки програмування створюють за допомогою спеціальних команд із списку інструкцій **Інструменти програмування (Programming Toolbar)** панелі інструментів **Математика (Math)**:

- **Add Line** – створити, або, при необхідності, продовжити жирну вертикальну лінію – спеціальний знак, що обмежує програмний блок;
- ← – оператор присвоювання в межах програмного блоку;
- **if** – оператор умовни;
- **while** – оператор циклу за визначених умов;
- **otherwise** – оператор іншого вибору;
- **break** – оператор припинення розрахунків;
- **continue** – оператор продовження розрахунків;
- **return** – оператор повернення на початок циклу;
- **on error** – оператор обробки помилок.

Оператор **Add Line**.

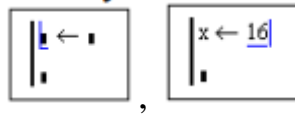
Оператор **Add Line** визначає межі програмного блоку (підпрограми), який є списком команд, що виконуються послідовно, згори вниз. Програмний блок може бути функцією без імені, яка розраховує результат.

Результат одержується у декілька етапів:

1. Натиснути на панелі програмування клавішу “] ” або **Add Line** з панелі інструментів **Математика (Math)**. На робочому полі, в місці розташування курсору з’явиться шаблон блоку з двома місцями введення інформації



2. У верхнє місце введення встановити курсор. Потім натиснути клавішу “ ← ” і чекати появи шаблону верхнього рядка. Шаблон необхідно заповнити іменем x та константою 16:



3. У нижнє місце введення ввести шаблон квадратного кореня, аргумент x і натиснути клавішу “Пробіл”. Створюється режим, коли курсор підкреслює увесь вираз.

4. Натиснути клавішу “=” і одержати результат.

Приклад. Знайти значення функції x , якщо $\sqrt{x} = 16$:

$$\left| \begin{array}{l} x \leftarrow 16 \\ \sqrt{x} \end{array} \right. = 4$$

Слід відзначити, що змінні у блоці та змінні поза блоком можуть мати однакові імена, але це різні змінні, що не залежать одна від одної. Іноді їх доцільно зв’язати процедурою передачі параметрів.

Приклад. Виконати обчислення функції $B = \frac{x + y + z}{x + y \cdot z}$ при двох значеннях змінних $x = 1, y = 2, z = 3$ та $x = 5, y = 7, z = 2$.

$$B(x, y, z) := \frac{x + y + z}{x + y \cdot z}$$

Для цього достатньо задати конкретні значення формальним параметрам x, y, z і одержати результат:

$$B(1, 2, 3) = 0.857$$

$$B(5, 7, 2) = 0.737$$

Розрахунки з формальними параметрами зручно виконувати за допомогою програмного блоку.

Приклад. Виконати обчислення функції, заданої у попередньому

прикладі $B = \frac{x+y+z}{x+y \cdot z}$ при двох значеннях змінних $x=1, y=2, z=3$ та $x=5, y=7, z=2$, за допомогою програмного блоку.

$$B(x,y,z) := \left| \begin{array}{l} a \leftarrow x + y \cdot z \\ \frac{x + y + z}{a} \end{array} \right.$$

$$B(1,2,3) = 0.857$$

$$B(5,7,2) = 0.737$$

Програмному блоку з ім'ям **B** присвоєно результат, а **x, y, z** – формальні параметри, що передають у блок фактичні значення.

Оператор **if** та **otherwise**.

Усі команди програмного блоку мають шаблони з місцями введення даних у вигляді констант, ідентифікаторів, арифметичних або логічних виразів. Наприклад, команди **if** та **otherwise** мають такі шаблони:

■ **if** ■ ■ **otherwise** .

Їх використання у програмному блоці демонструє програма визначення модуля за алгоритмом: якщо задане значення менше за 0, вивести це значення зі знаком мінус. В іншому випадку – зі знаком плюс.

$$\text{abs}(x) := \left| \begin{array}{l} -x \text{ if } x < 0 \\ x \text{ otherwise} \end{array} \right. \qquad \text{abs}(-15) = 15 \qquad \text{abs}(15) = 15.$$

Приклад. Знайти значення критерію Нусельдта. задано значення критерію Рейнольдса $Re = 2800$. При $Re \geq 10000$ значення критерію Нусельдта знаходиться за залежністю $Nu = 0,021 \cdot Re^{0,6}$, а при $Re < 10000$ – залежністю $Nu = 0,08 \cdot Re^{0,8}$.

Розв'язання: Приклад наведено на рисунку 8.1:

- записуємо задачу і задаємо функцію $Nu :=$;
- за допомогою оператор **Add Line** задаємо програмний блок вказуємо у верхньому рядку $0.021 \cdot Re^{0.6}$ if $Re \geq 1000$,

вказуємо у нижньому рядку $0.08 \cdot Re^{0.9}$ та вибираємо умову (команду програмного блоку) otherwise – інакше;

– записуємо функцію $Nu =$ та отримуємо результат.

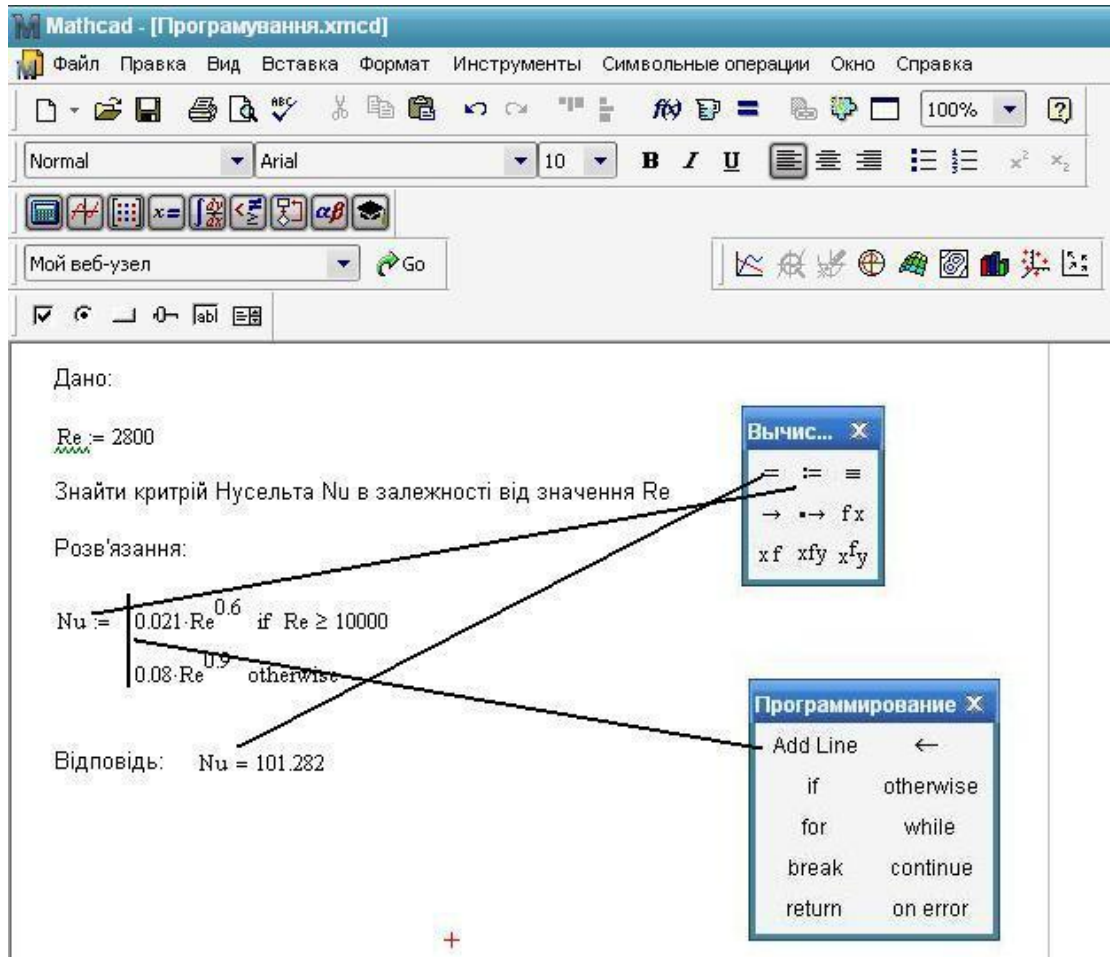


Рис. 8.1 – Приклад виконання програми з командою **if**

Оператор **for**, **while** та **break**.

Програмні блоки з командою **for** доцільно застосувати у циклах з підсумовуванням. Команда має такий шаблон:

```
for i ∈ 1..n
  |
```

Варіант програми:

$$s(n) := \begin{cases} s \leftarrow 0 & s(5) = 15 \\ \text{for } i \in 1..n & \\ s \leftarrow s + i & s(10) = 55 \end{cases}$$

Програма складається з трьох операторів. Перший присвоює змінній s початкове значення нуля. Другий програмує цикл розрахунків, в якому параметр лічильника i набуває значень від 1 до n з кроком 1 , а третій підсумовує попереднє значення s з черговим i . Формальний параметр n передає до програми конкретне значення.

Приклад. Знайти значення функції $y = \sin(x)$ при значеннях $x = 2,37; 2,38, \dots, 5,52$.

Розв'язання: Приклади наведено на рисунках 8.2 та 8.3:

- записуємо задачу і задаємо функцію $y(a,b):=$ (Рис. 8.2);
- за допомогою оператор **Add Line** задаємо зовнішній та внутрішній програмні блоки (Рис. 8.2);
- записуємо межі зміни функції $y(2.37,2.52)$, натискаємо на клавіатурі знак дорівнює або вибираємо з панелі **Evaluation (Обчислення)** та отримуємо результат (Рис. 8.3).

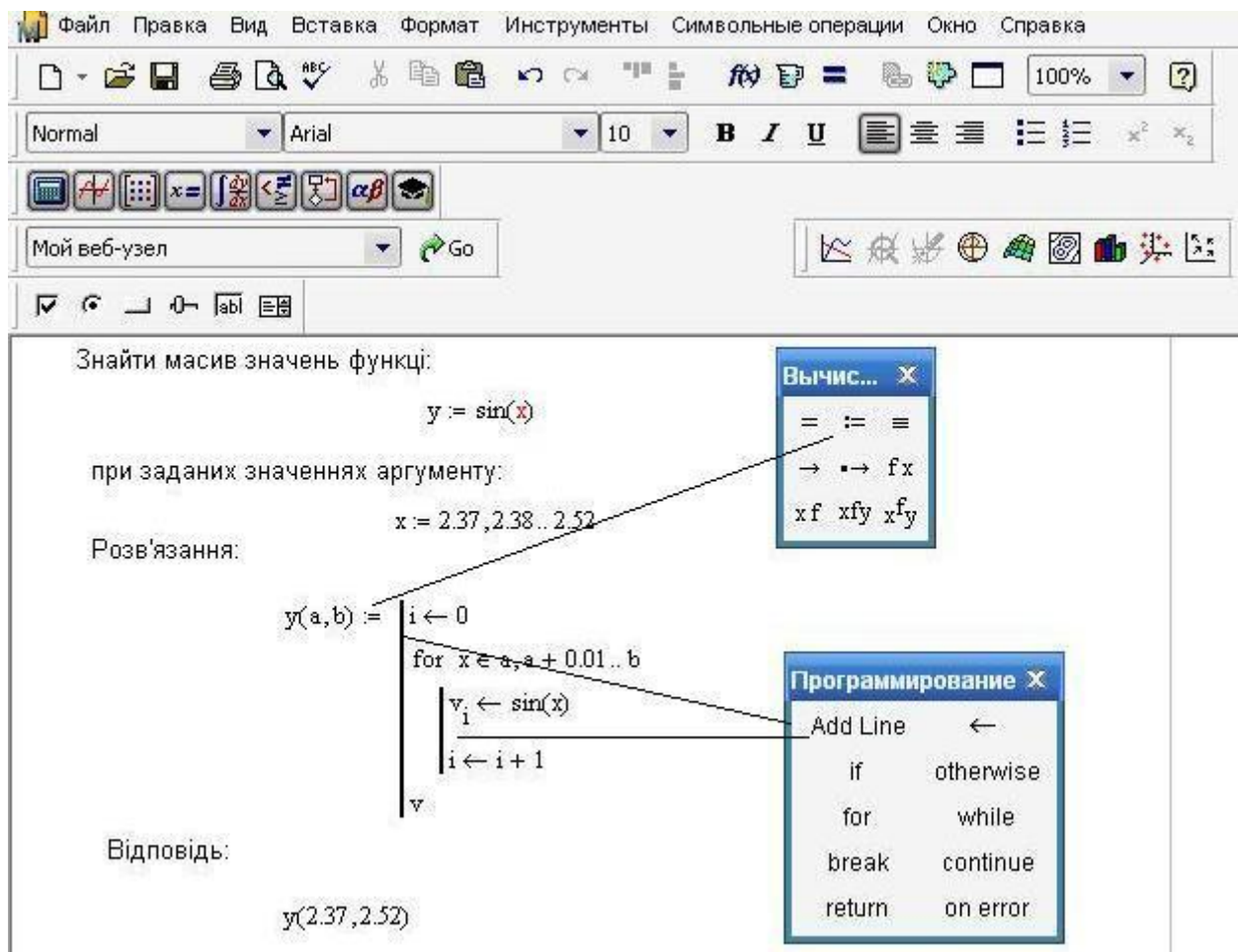


Рис. 8.2 – Приклад задавання циклу командою **for**

Mathcad - [Програмування.xmcd]

Файл Правка Вид Вставка Формат Інструменти Символьні операції Окно Справка

Normal Arial 10 **B** *I* U x^2 x_2

Мой веб-вузел Go

Знайти масив значень функції:

$$y := \sin(x)$$

при заданих значеннях аргументу:

$$x := 2.37, 2.38 \dots 2.52$$

Розв'язання:

$$y(a, b) := \begin{cases} i \leftarrow 0 \\ \text{for } x \in a, a + 0.01 \dots b \\ \quad \begin{cases} v_i \leftarrow \sin(x) \\ i \leftarrow i + 1 \end{cases} \\ v \end{cases}$$

Відповідь:

	0
0	0.697
1	0.69
2	0.683
3	0.675
4	0.668
5	0.661
6	0.653
7	0.645
8	0.638
9	0.63
10	0.622
11	0.614
12	0.606
13	0.598
14	0.59
15	0.582

$y(2.37, 2.52) =$

Вычис...

- = := ≡
- •→ f x
- x f x f y x^f y

Програмування

- Add Line ←
- if otherwise
- for while
- break continue
- return on error

+

Рис. 8.3 – Приклад отримання результатів роботи циклу команди **for**

Наступні три приклади є варіантами розрахунків факторіала цілого числа за допомогою програмного блока з використанням:

а) оператора **for**:

$$\text{pr}(n) := \left| \begin{array}{l} p \leftarrow 1 \\ \text{for } i \in 1..n \\ p \leftarrow p \cdot i \end{array} \right. \quad \begin{array}{l} \text{pr}(3) = 6 \\ \text{pr}(5) = 120 ; \end{array}$$

б) оператора **while**:

$$\text{fc}(n) := \left| \begin{array}{l} f \leftarrow 1 \\ \text{while } n \leftarrow n - 1 \\ f \leftarrow f \cdot (n + 1) \\ f \end{array} \right. \quad \text{fc}(5) = 120 ;$$

в) операторів **while** та **break**:

$$\text{fct}(n) := \left| \begin{array}{l} f \leftarrow n \\ \text{while } 1 \\ \left| \begin{array}{l} f \leftarrow f \cdot (n - 1) \\ n \leftarrow n - 1 \\ \text{break if } n = 1 \end{array} \right. \\ f \end{array} \right. \quad \text{fct}(5) = 120$$

Оператор **error**.

Рядок з оператором **error** вступає в дію, коли результати в інших рядках не визначені. Поруч з таким оператором у дужках позначається зміст спеціального повідомлення, яке необхідно вивести.

Приклад. Програмна функція $R(i)$ виводить на екран текстові повідомлення, якщо формальному параметру i задаються значення 1 або 2 . В іншому випадку функція не визначається, курсор лишається у межах оператора виведення і на екран робочого поля виводиться заплановане повідомлення (тут виводиться текст **no!no!**) (Рис. 8.4).

$$R(i) := \begin{cases} \text{"One"} & \text{if } i = 1 \\ \text{"Two"} & \text{if } i = 2 \\ \text{error("no!no!")} & \text{otherwise} \end{cases}$$

$$R(1) = \text{"One"}$$

$$R(2) = \text{"Two"}$$

$$R(3) = \text{"no!no!"}$$

Рис. 8.4 – Приклад програми оператором **error**

Оператор **on error**.

Оператор **on error** має шаблон з двома місцями введення:

■ **on error** ■

Ліворуч вводиться функція або текстовий рядок повідомлення, що активізується при неможливості розрахунків за функцією праворуч.

Приклад. Програма виводить на екран повідомлення про результати обчислення функції, якщо це значення неможливо обрахувати, як у прикладі на рисунку 8.5, оператор **on error** виводить результати обчислення вказаної ліворуч оператора функції (або текстового повідомлення) – $t(0)=0$.

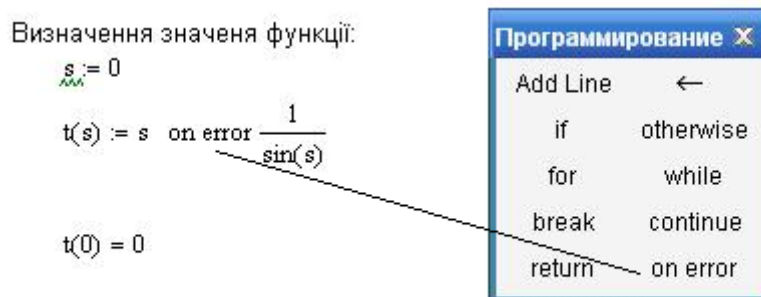


Рис. 8.5 – Приклад програми оператором **error**

Індивідуальні завдання

Варіант 1

1. Знайти границю числової послідовності

$$\lim_{n \rightarrow \infty} n \left(\sqrt{n^2 + 2} - \sqrt{n^2 - 2} \right)$$
$$\lim_{n \rightarrow \infty} \left(\sqrt{(n^2 + 2)(n^2 - 3)} - \sqrt{n^4 - 8} \right)$$
$$\lim_{n \rightarrow \infty} \left(\frac{3n^2 + 2}{3n^2 + 1} \right)^{n^2}$$

2. Знайти границю функції

$$\lim_{x \rightarrow 0} \frac{1 - \cos 5x}{e^{x^2} - 1}$$
$$\lim_{x \rightarrow 0} \frac{e^{\sin 2x} - e^{\sin 3x}}{\operatorname{tg} 2x}$$
$$\lim_{x \rightarrow 1} \left(\frac{2 - x}{x} \right)^{\frac{1}{\ln(3-2x)}}$$

3. Знайти похідну функції

$$y = \ln \left(x + \sqrt{4x^2 + 1} \right) + x \sqrt{x^2 + 3} \sqrt{4x^2 + 1}$$
$$y = \frac{x \arccos 3x}{\sqrt{1 - 9x^2}} + \ln \sqrt{1 - 9x^2}$$
$$y = 2 \ln \frac{x}{1 + \sqrt{1 - x^2}} - \frac{\sqrt{1 - x^2}}{2}$$

4. Знайти невизначений інтеграл

$$\int \frac{3x^3 + 11x^2 + 16x + 10}{(x+1)^2(x^2+x+2)} dx$$
$$\int (x^2 - px + 1) e^{2x} dx$$
$$\int x^2 \cos 2x dx$$

5. Обчислити визначений інтеграл

$$\int_1^2 x^2 \sqrt[5]{2-x} dx$$
$$\int_{-2}^1 \frac{xdx}{x^3 + 1}$$
$$\int_{\pi/2}^{\pi} x \sin^3 x dx$$

Варіант 2

1. Знайти границю числової послідовності

$$\lim_{n \rightarrow \infty} n \left(\sqrt{n(n-1)} - \sqrt{n^2 + 2} \right)$$

$$\lim_{n \rightarrow \infty} \frac{\sqrt{n^5 - 6} - n\sqrt{n(n^2 + 4)}}{\sqrt{n}}$$

$$\lim_{n \rightarrow \infty} \left(\frac{5n+3}{5n+1} \right)^{n+2}$$

2. Знайти границю функції

$$\lim_{x \rightarrow 0} \frac{\ln(1 - 2 \sin x)}{\sin 3x}$$

$$\lim_{x \rightarrow -1} \frac{x^2 - 1}{\sin 2\pi x}$$

$$\lim_{x \rightarrow 4} \left(\frac{2x-3}{x+1} \right)^{\frac{1}{\sqrt{x}-2}}$$

3. Знайти похідну функції

$$y = \frac{x^3}{\arcsin x} + \frac{x^2 + 1}{x} \sqrt{1 - x^2}$$

$$y = \ln \left(x + \sqrt{1 + 4x^2} \right) - \sqrt{1 + 4x^2} \operatorname{arccotg} 2x$$

$$y = 3 \arccos \frac{1}{2x+3} - 2\sqrt{x^2 + 3x + 2}, \quad 2x + 3 > 0$$

4. Знайти невизначений інтеграл

$$\int \frac{3x^3 - 2x^2 + x + 1}{(x-1)^2(x^2 + x + 1)} dx$$

$$\int (x+2)^{-2} \cdot e^{-3x} dx$$

$$\int x \sin^2 2x dx$$

5. Обчислити визначений інтеграл

$$\int_{\pi/4}^{\pi/2} x^2 \cos 3x dx$$

$$\int_{e/2}^1 x^2 \ln 2x dx$$

$$\int_0^{1/2} \arccos \sqrt{1 - 2x^2} dx$$

Варіант 3

1. Знайти границю числової послідовності

$$\lim_{n \rightarrow \infty} \left(-\sqrt[3]{n^3 - 4} \right) \cdot n\sqrt{n}$$

$$\lim_{n \rightarrow \infty} \left(+\sqrt[3]{3 - n^3} \right)$$

$$\lim_{n \rightarrow \infty} \left(\frac{2n-1}{2n+3} \right)^{n-2}$$

2. Знайти границю функції

$$\lim_{x \rightarrow 0} \frac{\cos 3x - \cos 5x}{\cos x - 1}$$

$$\lim_{x \rightarrow 0} \frac{2^x + 2^{-x} - 2}{1 - \cos 3x}$$

$$\lim_{x \rightarrow 1} \left(\frac{2x-1}{x} \right)^{\frac{1}{\sqrt{x}-1}}$$

3. Знайти похідну функції

$$y = \ln \left(x + \sqrt{1+9x^2} \right) - x \left(x^2 + 3 \right) \sqrt{1+9x^2}$$

$$y = \ln \left(x - \sqrt{1+4x^2} \right) + \frac{\sqrt{1+4x^2}}{x}$$

$$y = \sqrt{4-10x-3x^2} + \frac{1}{2\sqrt{3}} \arcsin \frac{3x+5}{3\sqrt{3}}$$

4. Знайти невизначений інтеграл

$$\int \frac{2x^3 + 2x^2 + 6x + 1}{(x^2 + 4)(x^2 + x + 1)} dx$$

$$\int (x^2 + 1) \ln x dx$$

$$\int (x^2 + 4) \operatorname{arctg} x dx$$

5. Обчислити визначений інтеграл

$$\int_1^3 (-|2-x^2|)^3 dx$$

$$\int_1^2 \frac{\ln^2 x}{x^3 \sqrt{1+\ln x}} dx$$

$$\int_{2e}^{3e} \frac{dx}{x \ln x \ln \ln x}$$

Варіант 4

1. Знайти границю числової послідовності

$$\lim_{n \rightarrow \infty} (\sqrt{n^2 - 5n + 6} - n)$$
$$\lim_{n \rightarrow \infty} (\sqrt{n(n+3)} - \sqrt{n^2 - 3n + 2})$$
$$\lim_{n \rightarrow \infty} \left(\frac{2n^2 - n + 3}{2n^2 + 2n + 1} \right)^{-n+1}$$

2. Знайти границю функції

$$\lim_{x \rightarrow 0} \frac{\arcsin 2x}{x^2 + \pi x}$$
$$\lim_{x \rightarrow 0} \frac{\sqrt{1 + \sin 2x} - \sqrt{1 + \operatorname{tg} 2x}}{x^3}$$
$$\lim_{x \rightarrow \pi} (\cos 2x)^{\frac{1}{\sin^2 x}}$$

3. Знайти похідну функції

$$y = \sqrt{5+x} \sqrt{2-x} + 2 \ln (\sqrt{5+x} + \sqrt{2-x})$$
$$y = \operatorname{arctg} \frac{3x+1}{\sqrt{5}} + \ln \frac{\sqrt{3x^2+2x+2}}{x}$$
$$y = \ln \left(\sqrt[3]{\frac{2x-1}{2x+1}} \right) - \arcsin \sqrt{\frac{2x-1}{2x+1}}$$

4. Знайти невизначений інтеграл

$$\int \frac{3x^3 + 10x^2 + 14x + 5}{(x+1)^2 (x^2 + 3x + 4)} dx$$
$$\int \ln^3 x dx$$
$$\int (x-1)^2 \arcsin 3x dx$$

5. Обчислити визначений інтеграл

$$\int_0^1 \frac{dx}{e^{-x} + e^{-3x}}$$
$$\int_0^1 x^2 \operatorname{arctg} \pi x dx$$
$$\int_{\frac{1}{e}}^e (-|\ln x|)^2 dx$$

Частина II Використання середовища MatLab

1 Команди і функції пакету MATLAB

Після запуску програми MATLAB на екрані комп'ютера з'являється головне вікно, що містить меню, інструментальну лінійку з кнопками і клієнтську частину вікна із запрошенням `>>`. Цю частину прийнято називати командним вікном.

У командне вікно з клавіатури вводяться команди, що складаються з букв, цифр і знаків операцій. Натиснення клавіші "Enter" служить сигналом для виконання набраної команди.

1.1 Корисні команди пакету

Знайомство з пакетом MATLAB корисно почати з команд виклику операторів загального призначення, покликаних полегшити роботу користувача. До цих операторів відносяться наступні:

- *help* - виводить список підключених TOOLBOX'ів.
- *help help* - виводить інформацію про роботу довідника *help*.
- *help «ім'я функції»* - виводить інформацію про функцію. Інформацією про функцію вважається текст поміщений в початкові рядки коментарів М-файлу з текстом функції, помічених символом '%'. Для вбудованих функцій пакету в піддиректоріях директорії TOOLBOX заводиться М-файл, що містить тільки рядки коментарів. Ім'я цього файлу співпадає з ім'ям функції.
- *help «ім'я файлу»* - виводить інформацію про вміст М-файлу аналогічно інформації про функцію.
- *lookfor «набір символів»* - виводить список М-файлів, доступних для звернення у нинішній момент часу, у яких в рядках коментарів зустрічається приведений набір символів. Цю команду зручно використовувати для пошуку по ключовому слову або словосполученню необхідних користувачеві функцій.
- *demo* - виводить список демонстраційних прикладів.
- *tour* - пропонує меню огляд демонстраційних прикладів.
- *!« команда»* - виконує команду DOS.
- *casesen « on/off»* - перемикає режим розрізнення регістрів за відсутності опцій, опція *on* встановлює режим розрізнення регістрів, а опція *off* - відмінняє його.
- *diary «ім'я файлу/on/off»* - управляє режимом паралельного запису у файл. Якщо вказано ім'я файлу, то встановлюється режим, при якому все, що виводиться на екран одночасно записується у файл з цим ім'ям. Опція *off* перериває цей режим, а опція *on* повертає знову. Існуючий файл відкривається на дозапис.

- *format «flag»* - управляє форматом виведення результатів в командне вікно. Всі обчислення в MATLAB'е проводяться з точністю 32-розрядної арифметики. Проте виведення значень в командне вікно проводиться з меншою точністю. При наступних значеннях параметра «*flag*» встановлюються наступні види формату виведення:

— *short* - формат з фіксованою крапкою з 5 значущими цифрами (використовується за умовчанням). Цей формат встановлюється при зверненні *format* без вказівки опції *flag*.

— *long* - формат з фіксованою крапкою з 15 значущими цифрами.

— *short e* - формат з плаваючою крапкою з 5 значущими цифрами.

— *long e* - формат з плаваючою крапкою з 15 значущими цифрами.

— *short g* - вибирається кращий з двох попередніх форматів з 5 значущими цифрами.

— *long g* - вибирається кращий з двох попередніх форматів з 15 значущими цифрами.

— *hex* - шістнадцятиричний формат.

— *+* - відображається тільки знак.

— *rat* - наближення результату у вигляді відношення цілих чисел.

Ці ж формати висновку можна встановити у вікні *General* в результаті виклику пункту підміню *Preferences* в пункті *File* головного меню.

- *home* - встановлює курсор з поточним командним рядком у верхній лівий кут вікна.
- *clc* - очищення командного вікна.
- *computer* - дає довідку про тип комп'ютера (Для операційних систем після Windows 95 і Windows NT і їх спадкоємців команда практично даремна, оскільки повідомляє тільки тип операційної системи).
- *exit, quit*- ці команди припиняють роботу MATLAB.
- *;* - дуже корисний символ. Виведення на екран результату виконання команди в робоче вікно відміняється, якщо команда закінчується цим символом.

1.2 Змінні. Вектора і матриці

Імена змінних в MATLAB'і можуть позначатися довільним набором букв, цифр і знаків підкреслення ('_'). Вони повинні починатися з букви і містити не більше 31 символу. Подальші символи будуть проігноровані. При цьому не рекомендується використовувати імена операторів і функцій MATLAB'а і імена стандартних змінних використовуваних MATLAB'ом:

- *i, j*- уявна одиниця ($\sqrt{-1}$);
- *inf* - невизначеність типу 1/0 (∞);
- *NAN* - невизначеність типу 0/0 (Not a number);
- *ans* - результат останньої виконуваної операції або функції (якщо в командному рядку відсутній оператор присвоювання, результат операції

- присвоюється змінній *ans* автоматично);
- $pi - \pi = 4 * atan(1) = 3.1415926 \dots$;
 - *rand* - псевдовипадкове число рівномірне розподілене на інтервалі $[0,1]$;
 - *eps* - відносна точність обчислень. За цю величину береться відстань від 1 до наступного дійсного числа доступного комп'ютеру;
 - *realmin*, *realmax* - мінімальне і максимальне дійсні числа.

Основним об'єктом MATLAB'a є матриця. Вектором називається матриця, один з розмірів якої дорівнює одиниці, скаляром - матриця 1×1 .

Всі основні функції і оператори в MATLAB'і - матричні, окрім випадків, що обумовлюються особливо.

Ввести скаляр можна простим оператором присвоювання $a = 2,34$.

Для формування матриць і векторів використовуються символи «[...]». Наприклад матрицю 2×3 можна записати у вигляді $a = [1 \ 2.3 \ 4; 3.4 \ 2 \ 3]$. Різні елементи в такому записі розділяються пропуском (або комою), рядки відділяються один від одного символом ';'. Різне число чисел в рядках або стовпцях породжує помилку, про яку MATLAB повідомляє користувача.

Відповідно до вищеописаного оператор $b = [1 \ 2 \ 3]$ вводить вектор рядок, а $a = [1; 2; 3]$ – вектор-стовпець.

Подібними правилами можна також користуватися при формуванні блокових матриць. Наприклад, якщо виконані присвоювання попереднього абзацу, оператор $d = [a; b]$ породжує матрицю

$$d = \begin{bmatrix} 1 & 2.3 & 4 \\ 3.4 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix}$$

Звернення до елементу матриці a , розташованому на перетині рядка з номером i і стовпця з номером j має вигляд $a(i,j)$. Вектор стовпець співпадаючий з j -м стовпцем матриці a отримаємо в результаті звернення $a(:,j)$, а i -й вектор-рядок - $a(i, :)$.

Підматрицю матриці a можна отримати, вказавши замість індексів вектора u і v , що містять номери рядків і стовпців матриці a , на перетині яких розташовані елементи підматриці. Пояснимо це на прикладі: для матриці d , наведеної раніше, і векторів $u = [1 \ 3]$ і $v = [1 \ 2]$

$$a = d(u,v) = \begin{bmatrix} 1 & 2.3 \\ 1 & 2 \end{bmatrix} \quad (1)$$

У загальному випадку значення компонент векторів u і v в приведеному операторові при обчисленні індексів будуть закруглені до найближчого цілого числа.

Блок матриці можна отримати також використавши символ перерахування ':'. Річ у тому, що в MATLAB'і оператор

$$u = start : step : fin \quad (2)$$

задає вектор, що складається з кінцевого числа членів арифметичної прогресії, перший член якої рівний $start$, крок прогресії – $step$, а останнім є

найближчий до значення fin член прогресії, що належить сегменту, граничними значеннями якого є $start$ і fin . При $step > 0$ останнім компонентом сформованого вектора буде максимальний член, що належить сегменту $[start, fin]$, а при $step < 0$ – мінімальний член, що належить сегменту $[fin, start]$. В записі $v = start : fin$ величина $step$ приймається рівною 1. Використавши такий спосіб запису матрицю a з (1) можна також отримати, наприклад, в результаті запису

$$a = d(1 : 1.6 : 3,1 : 2).$$

У MATLAB'і можна також задати символічні змінні, які є вектора символів. Найпростіше задати таку змінну командою типу

$c = 'Hello!'$

Вона еквівалентна команді $c = ['H', 'e', 'l', ';', 'o', '!']$.

Необхідні операції над рядками в MATLAB'і можна виконувати за правилами векторних операцій, описаних в цьому параграфі.

1.3 Прості арифметичні операції

Символьні позначення основних операцій, що проводяться над змінними:

- $=$ – присвоювання;
- $+$ – складання;
- $*$ – множення;
- \backslash – ділення зліва (для матричних величин результат виконання операції $X = A \backslash B$ приблизно те ж, що і $A^{-1} * B$, еквівалентний розв'язку матричного рівняння $A * X = B$, обчисленого методом виключення Гауса. Якщо матриця A погано обумовлена або вироджена – виводиться попередження. Якщо матриця розміру $M \times N$, а B - стовпець довжиною M , розв'язок матричного рівняння $A * X = B$ проводиться методом найменших квадратів (див. також функцію *pinv*).);
- $/$ – ділення справа (для матричних величин результат виконання операції $X = A / B$ приблизно те ж, що і $A * B^{-1}$, і еквівалентний розв'язку матричного рівняння $X * B = A$ подібно до того, як це реалізовано для оператора \backslash);
- $^$ - піднесення до степеня;
- $'$ – транспонування матриці;
- $./$ - поелементне ділення справа;
- $.^$ - поелементне піднесення до степеня;

Ці операції виконуються з урахуванням вимог традиційної матричної алгебри. Істотною гідністю пакету MATLAB є організована перевірка розмірності, здійснювана при проведенні обчислень. При невідповідності розмірності операндів видається повідомлення про помилку.

Послідовність операцій визначається відповідно до звичайних правил алгебри, причому для виділення першочергових операцій традиційно використовуються круглі дужки (...).

1.4 Робоча область і операції над нею

Для змінних, заданих при роботі в MATLAB'і, виділяється область пам'яті комп'ютера. Область пам'яті, в якій зберігаються доступні для обчислень змінні, називається робочою областю (*workspace*). Проглянути вміст робочої області дозволяють команди *who* і *whos*.

- *who* - виводить на екран список змінних, що зберігаються в робочій області
- *whos* - виводить на екран список цих змінних, доповнений інформацією про їх розмір і тип. Як приклад, наведемо результат роботи цієї команди для випадку, коли при роботі в MATLAB визначені наступні дійсні змінні - скаляр *a*, вектор-рядок *b* з 3-х елементів, матриця *c* розміру 2×3 і комплексний вектор-стовпець *d* з 2-х елементів.

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x3	24	double array
c	2x3	48	double array
d	2x1	32	double array (complex)

Grand total is 12 elements using 112 bytes

Перевірити наявність змінної *a* в робочій області пам'яті дозволяє функція *exist* ('*a*'). Вона повертає 1, якщо змінна *a* існує в робочому просторі; 2 - якщо *a* – це ім'я файлу на диску, 0 - якщо *a* не існує.

Для збереження і відновлення змінних, розміщених в робочій області використовуються команди *save* і *load*.

- *save* - зберігає всі змінні робочої області в бінарному файлі *matlab.mat*.
- *save* « ім'я файлу » - зберігає всі змінні робочої області в бінарному файлі з розширенням *.mat*. Цю ж дію можна зробити також в результаті виклику пункту підміню *Save Workspace as* в пункті *File* головного меню.
- *save* « ім'я файлу » « змінні » - зберігає змінні в бінарному mat-файлі із заданим ім'ям. Як змінні задається список їх імен розділених пропуском.
- *save* « ім'я файлу » « змінні » – *ascii* - зберігає змінні у файлі, в стандартних кодах ASCII. Ім'я файлу може мати довільне розширення.
- *load* - відновлює змінні з файлу *matlab.mat*.
- *load* « ім'я файлу » - відновлює змінні в робочу область з *mat-файлу* із заданим ім'ям. Цю ж дію можна зробити також в результаті виклику пункту підміню *Load Workspace* в пункті *File* головного меню.

- *load* « ім'я файлу» -*ascii* - читає в робочу область з файлу, матрицю записану в кодах ASCII у вигляді таблиці, стовпці якої розділені пропуском. Розумні версії MATLAB'у (4 і вище) за відсутності ключа -*ascii* самі благополучно розбираються в чому справа і працюють нормально. Результат виконання цієї операції поміщається в масив, ім'я якого співпадає з ім'ям файлу (природно з відкинутим розширенням). Зрозуміло, що ім'я такого файлу повинне починатися з букви і не повинно містити знаків типу '-', '+' і т.д.

Нарешті видалити змінні з робочої області допомагає команда *clear*.

- *clear* - видаляє всі змінні з робочої області.
- *clear* « змінні» - видаляє перераховані списком через пропуск змінні.

1.5 Робоча папка

За відсутності вказівки повного шляху до читаних або записуваних файлів пакет MATLAB здійснює пошук файлу для читання і проводить запис в папку, звану робочою. Всі версії MATLAB'a дозволяють встановити шлях в робочу папку, вибрану користувачем. Універсальним є спосіб, що використовує команду *cd*

- *cd* « шлях» - встановлює папку, вказану змінною «шлях» як робоча.

При бажанні останню вимогу можна порушити, але тоді ім'я файлу повинне бути скрізь присутнім тільки у вигляді символічної константи. Робота з таким ім'ям сильно ускладнена і її опис відсутній.

Проглянути вміст цієї теки дозволяють команди:

- *dir* - виводить список файлів, що містяться в робочій папці;
- *what*- виводить список файлів MATLAB'у (*L.m*, *L.mat* *M .tex*), що містяться в робочій папці.

У сучасних версіях MATLAB'a існують способи встановити робочу папку, використовуючи інструментальну панель. Починаючи з версії 5.0 в випадаючому меню пункту "*File*" головного меню командного вікна введений пункт установки шляхів "*Set Path...*", при зверненні до якого надається можливість змінити параметр "*Current directory* ", вказуючий шлях в робочу папку. Знак цієї дії виведений також і на панель інструментів. У всіх варіантах 4-ої версії спеціальний пункт меню відсутній, проте для зміни робочої папки досить спробувати відкрити файл, використавши підпункт "*open*" у спадаючому меню пункту "*File*" головного меню командного вікна.

При такій спробі MATLAB пропонує вибрати шлях в папку, в якій знаходиться файл призначений користувачем для читання і редагування. Після відкриття такого файлу редактором, папка в якій він знаходився встановлюється як робоча. Якщо у вибраній папці користувач не бажає відкривати ніяких файлів, то можна відмовитися в останню мить від операції відкриття файлу, вибравши у вікні відкриття файлу кнопку "*cancel*".

Остання з папок, переглянутих користувачем, буде встановлена як робоча.

1.6 Програмування в MATLAB'і. Сценарії і функції

До цих пір читачеві пропонувався простий спосіб роботи в системі MATLAB - послідовне введення команд в командному вікні, негайна їх обробка пакетом і видача обчисленого результату. Ефективнішим представляється спосіб попередньої підготовки послідовності команд, запис її у вигляді файлу і подальше неодноразове виконання цих команд. У MATLAB'і існує два типи файлів, що дозволяють реалізувати таку можливість: *M-сценарій* і *M-функція*.

- *M-сценарієм (Script-файлом)* називається файл, що містить послідовність команд, операцій і функцій MATLAB'а, що виконують дії над змінними робочої області. M-сценарій не має вхідних і вихідних аргументів.
- *M-функцією* (надалі просто функцією) називається послідовність команд і операцій, поміщена у файл, забезпечений спеціальним заголовком, що здійснює операції тільки над вхідними змінними.

1.6.1 Сценарії

Script -сценарій поміщається в текстовий файл з розширенням **.m* і містить послідовність операторів MATLAB'а. У сучасних версіях MATLAB'у сценарій не вимагає додаткових заголовків або коментарів. Як приклад приведемо наступний простий m-сценарій.

```
c=3.5;  
d=a/c*cos(b);  
e=a*(c*sin(b)+cos(pi-b));  
f=e/d
```

Створіть в поточній робочій папці будь-якими зручними Вам засобами файл з ім'ям *my1prog.m* (у версіях MATLAB 5.0 і вище для цього зручно скористатися програмою *MATLAB Editor/Debugger*), наберіть в нім вищезгаданий текст і запишіть його на диск. Тепер введемо в командному вікні значення $a = 7$; $b = \pi/4$; і наберемо команду *my1prog*, яка викликає виконання вказаної послідовності операцій над змінними, що зберігаються в робочій області пам'яті. Всі використані змінні c , d , e і f також зберігаються в робочій області пам'яті і доступні для подальших операцій.

У програмах оброблювальних великі масиви даних збереження допоміжних змінних і масивів незручно, оскільки веде до зайвого засмічення оперативної пам'яті. Через це при програмуванні на мові MATLAB'у введено поняття функції.

1.6.2 Функції

Текст функції також поміщається в текстовий файл з розширенням **.m*. Проте цьому файлу пред'являються деякі додаткові вимоги. Перший виконуваний оператор файлу містить опис функції вигляду:

$$\text{function [outpars]=progrname(inpars)}$$

Цей опис повідомляє про те, що у файлі записана функція з ім'ям *progname*, яка має вхідні параметри, перераховані в списку *inpars*, і вихідні параметри в списку *outpars*. Для прикладу переробимо попередній сценарій у функцію, яка отримує вхідні параметри *a* і *b*, і обчислює величини *e* й *f*.

Хорошим тоном вважається декілька рядків функції, наступних за початковим описом, присвятити коментарям до функції. Ці рядки починаються символом % і як правило містять опис звернення до функції, інформацію про вхідні і вихідні параметри і коментують нетрадиційні методи обчислень. Вказані коментарі видаватимуться в командне вікно MATLAB'a при виклику оператора виду *help «progname»*. Коментарі у файлах можуть бути написані і російською мовою, проте при їх виведенні програмою по команді *help* в різних версіях можуть виникати труднощі.

Виконання функції припиняється в наступних випадках:

- всі оператори виконуються послідовно до кінця файлу;
- чергового виконаного оператора *return* припиняє виконання поточної функції і передає управління функції, що його викликала;
- ознакою завершення тіла функції може служити також команда оголошення наступної функції. Приклад такого типу буде приведений пізніше в параграфі 5.4.

```
function [e,f]=mylprog(a,b)
% function [e,f]=mylprog(a,b)
% It is my first function in MA TLAB
% Inputs: scalar constants a and b
% Outputs: scalar constants e and f
% Calculation formulas is in text of file.
c=3.5;
d=a/c*cos(b);
e=a*(c*sin(b)+cos(pi-b));
f=e/d;
```

Тепер звернення до функції з командного вікна може мати вигляд

$$[e1,f1] = mylprog(a,b);$$

Як формальні параметри можна було також безпосередньо вказати їх значення 7 і $\pi/4$.

При зверненні *help mylprog* в командному вікні з'являється напис:

```
function [e,f]=mylprog(a,b)
It is my first function in MA TLAB
Inputs: scalar constants a and b
Outputs: scalar constants e and f
Calculation formulas is in text of file.
```

Зробимо ряд корисних зауважень, що пояснюють особливості роботи функцій в пакеті MATLAB.

1. При роботі в MATLAB 'і початківцям рекомендується

дотримуватися правила: один файл – одна функція.

2. Головним ім'ям функції служить ім'я файлу, в якому функція записана. У випадку, якщо ім'я записане в описі функції, не співпадає з ім'ям файлу, при зверненні до функції MATLAB шукатиме файл з вказаним ім'ям. Тому дотримуйтеся правила – ім'я функції і ім'я файлу повинні співпадати.
3. Функція в MATLAB'і може викликати іншу функцію і число вкладень, строго кажучи, не обмежено. Це дає великі можливості для створення рекурсії (автовиклика функції) і зациклення програм. Побоюйтеся цієї можливості. Основний недолік MATLAB'у полягає в тому, що перервати виконання такої функції можливо тільки в результаті аварійного переривання роботи при одночасному натисненні клавіш *CTRL — BREAK*.
4. Вхідні параметри a і b передаються функції по значенню, і довільні їх зміни при роботі функції не відбивається на значеннях в командному вікні або зухвалій функції.
5. У пакеті MATLAB кожна функція обов'язково має дві змінні *nargin* і *nargout*, які дозволяють отримати інформацію про кількість вхідних і вихідних параметрів функції.

1.7 Найбільш вживані стандартні функції пакету

У пакеті MATLAB реалізований великий набір стандартних функцій. Ці функції об'єднані в пакети – так звані TOOLBOX'и. Всі TOOLBOX 'и як правило розміщуються в піддиректорії TOOLBOX. Основним TOOLBOX'ом слід вважати TOOLBOX з ім'ям MATLAB. Файли цього TOOLBOX'у розміщені в піддиректоріях цієї директорії, проте значна їх частина містить тільки опис відповідних вбудованих функцій, чиє виконання здійснюється безпосередньо ядром програми. Набір таких функцій достатньо великий і достатньо повний опис їх міститься в документації пакету MATLAB [13] і в численній літературі [1]- [3] і [5]. Тому тут обмежимося тільки коротким описом найбільш вживаних функцій.

Першою відзначимо функцію $disp(x)$. Ця функція не має вихідних параметрів і виводить в робоче вікно MATLAB'а значення змінної x .

1.7.1 Елементарні математичні функції

Аргументами всіх вказаних в цьому пункті функцій - матриці, як і результати обчислень. Функції обчислюються для заданих матриць поелементно.

- $y = exp(x)$ - поелементна експонента ($y_{ij} = e^{x_{ij}}$)
- $y = sin(x)$ - синус
- $y = cos(x)$ - косинус
- $y = tan(x)$ - тангенс
- $y = asin(x)$ - арксинус
- $y = acos(x)$ - арккосинус
- $y = atan(x)$ - арктангенс

- $y = \text{sqrt}(x)$ - квадратний корінь
- $y = \text{abs}(x)$ - абсолютна величина
- $y = \text{log}(x)$ - натуральний логарифм
- $y = \text{log10}(x)$ - десятковий логарифм
- $y = \text{sign}(x)$ - сінгнум-функція

$$y = \begin{cases} 1, & \text{якщо } x_{ij} > 0 \\ 0, & \text{якщо } x_{ij} == 0 \\ -1, & \text{якщо } x_{ij} < 0 \end{cases}$$

1.7.2 Функції округлення і їм супутні.

- $y = \text{round}(x)$ - округлення елементів x до найближчого цілого (округлення в традиційному сенсі)
- $y = \text{ceil}(x)$ - округлення елементів x до найближчого цілого у бік збільшення
- $y = \text{fix}(x)$ - округлення елементів x до найближчого цілого у бік нуля (служить для виділення цілої частини чисел)
- $y = \text{floor}(x)$ - округлення елементів x до найближчого цілого у бік зменшення
- $y = \text{rem}(x, y)$ - залишок. $\text{rem}(x, y) = x - y * N$, де $N = \text{fix}(x/y)$.
- $y = \text{gcd}(x, y)$ - найбільший спільний дільник для елементів матриць x і y .

1.7.3 Функції над комплексними числами

Комплексні числа в MATLAB'е задаються простими операціями вигляду $a = 6.7 + 7.897i$, або $d = a + i*b$, якщо a і b задані раніше дійсні змінні. Комплексні числа можуть бути елементами матриці і відповідно, якщо аргументами нижченаведених функцій є комплекснозначні матриці, то функції обчислюються поелементно.

- $y = \text{angle}(x)$ - аргумент комплексного числа.
- $y = \text{conj}(x)$ - комплексне сполучення.
- $y = \text{imag}(x)$ - уявна частина.
- $y = \text{real}(x)$ - дійсна частина.
- $y = \text{sign}(x)$ - сінгнум-функція, для комплексного x , $\text{sign}(x) = x./\text{abs}(x)$.

Тут же згадаємо функції перетворення типів змінних:

- num2str - перетворить числову змінну в рядок символів;
- str2num - перетворить рядок символів в числову змінну. Рядок може містити цифри, десяткові крапки, знаки '+' і '-', позначення уявної одиниці i і букву 'e' при представленні числа з плаваючою комою.

1.7.4 Формування векторів і матриць

- $y = \text{linspace}(x_{\min}, x_{\max})$ - формує вектор y з 100 елементів

рівномірно розташованих між точками $xmin$ і $xmax$. Якщо вектор повинен містити інше число компонент N , звернення до цієї функції повинне мати вигляд $= linspace(xmin, xmax, N)$.

- $y = logspace(xmin, xmax)$ - формує вектор y з 100 елементів логарифмічно розташованих між точками $xmin$ і $xmax$. Якщо вектор повинен містити інше число компонент N , звернення до цієї функції повинне мати вигляд $= linspace(xmin, xmax, N)$.
- $y = zeros(N, M)$ - формує нульову матрицю розмірності $N \times M$. За наявності єдиного операнда N формується квадратна матриця розмірності $N \times N$.
- $y = ones(N, M)$ - формує матрицю розмірності $N \times M$, всі елементи якої рівні 1. За наявності єдиного операнда N формується квадратна матриця розмірності $N \times N$.
- $y = eye(N)$ - формує одиничну матрицю I розмірності $N \times N$.
- $y = rand(N, M)$ - формує матрицю розмірності $N \times M$, елементами якої є випадкові числа, рівномірно розподілені на інтервалі (0.0,1.0).
- $y = diag(v, n)$ - формує матрицю, на n -й наддіагоналі якою розташований вектор v . При $n = 0$ результатом виконання цієї функції є діагональна матриця, на діагоналі якої розташовані елементи вектора v . При значенні $n < 0$ компоненти вектора v розташовані на піддіагоналі з номером $|n|$. Наприклад $diag([1,2, 3], 2)$ породжує матрицю:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

а $diag([5, 2], -1)$ – матрицю:

$$\begin{bmatrix} 0 & 0 & 0 \\ 5 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$$

- $y = diag(A, n)$ - вирізує n -ную наддіагональ з матриці A . Матриця може не бути квадратною. При $n < 0$ виводиться піддіагональ з відповідним номером.

Для повороту векторів і матриць використовуються наступні функції:

- $fliplr(A)$ - переставляє стовпці матриці A симетрично щодо вертикальної осі
- $flipud(A)$ - переставляє рядки матриці A симетрично щодо горизонтальної осі;
- $rot90(A, n)$ - n разів повертає матрицю A на 90° проти годинникової стрілки. За відсутності параметра n проводиться поворот матриці на 90° . При нецілому значенні n функція не проводить ніяких дій, не видаючи ні попереджень ні повідомлень про помилку.

Крім того є функції, що генерують "іменні" матриці Ганкеля, Теплиця, Гильберта, Адамара, Уїлкінсона і т.д. Ці функції носять відповідні імена і про їх виклик найпростіше дізнатися використавши оператора *help*. При цьому слід враховувати, що ці імена записуються в загальноприйнятій транскрипції мови оригіналу: *hankel*, *toeplitz*, *hilb*, *hadamard*, *wilkinson* і т.д.

1.7.5 Операції над векторами і матрицями

- $y = cross(u, v)$ - векторний добуток двох векторів u і v довжини 3;

Якщо для наступних функцій аргументами є вектора, то результатом їх застосування є скаляр. Якщо ж аргументами є матриці, то результат обчислень вектор, j -й компонент якого - результат застосування відповідної функції до j -го стовпця матриці.

- $y = sum(x)$ - сума компонент вектора ;
- $y = prod(x)$ - добуток компонент вектора ;
- $y = min(x)$ - мінімальний компонент вектора;
- $y = max(x)$ - максимальний компонент вектора;
- $y = mean(x)$ - середнє арифметичне компонент вектора;
- $y = std(x)$ - середнє квадратичне відхилення компонент вектора;
- $y = median(x)$ - медіанна компоненту вектора. Під медіаною в цьому випадку розуміється величина, отримана в результаті наступних операцій: все n компонент вектора розташовуються в порядку зростання і в новому векторі вибирається компоненту з номером $(n + 1)/2$ для непарних n і середнє арифметичне компонент з номерами $n/2$ і $n/2 + 1$ для парних n ;
- $y = finite(x)$ - повертає 1, якщо всі компоненти вектора x кінцеві і 0 інакше;
- $y = isNaN(x)$ - повертає 1, якщо хоч би один з компонентів вектора x приймає значення *NAN* і 0 інакше;
- $y = all(x)$ - повертає 1, якщо всі компоненти вектора x не рівні і 0 інакше;
- $y = any(x)$ - повертає 1, якщо хоч би одна компоненту вектора x не рівні 0 і 0 інакше;;
- $y = find(x)$ - повертає номери ненульових компонент вектора. У

випадку, якщо аргумент x є матрицею, використовується наскрізна відрядкова нумерація;

Якщо для наступних функцій аргументами є вектора, то результатом їх застосування є вектор. Якщо ж аргументами є матриці, то результат обчислень матриця, j -й стовпець якої - результат застосування відповідної функції до j -ома стовпцю початкової матриці.

- $y = \text{cumsum}(x)$ - кумулятивна сума компонент (i -й компонент вектора y рівний сумі i перших компонент вектора x);
- $y = \text{cumprod}(x)$ - кумулятивний добуток компонент вектора (i -й компонент вектора y рівний добутку i перших компонент вектора x);
- $y = \text{sort}(x)$ - сортує компоненти вектора за збільшенням (стовпці в матричному x сортуються незалежно);
- $y = \text{diff}(x)$ - кінцево-різничне диференціювання вектора. Для вектора x довжиною N вектор y має довжину $N - 1$ і $y_k = x_{k+1} - x_k$. Для матриці x проводиться диференціювання стовпців $y = x(2:N,:) - x(1:N-1, :)$. При зверненні $y = \text{diff}(x, n)$ - аналогічним чином обчислюється n -та кінцево-різнична похідна;
- $[Px, Py] = \text{gradient}(x)$ - обчислення кінцево-різничних матричних градієнтів. При зверненні $[Px, Py] = \text{gradient}(x, Dx, Dy)$ аргументи Dx і Dy задають величини кроку по осях x і y .
- $z = \text{trapz}(x, y)$ - обчислює методом трапецій інтеграл по значеннях ординат y , які інтерпретуються як значення заданої функції в точках з абсцисами, записаними в x .

1.7.6 Норми векторів і матриць

$y = \text{norm}(v, N)$ - норма вектора або матриці. Функція обчислює різні норми залежно від того v - вектор або матриця, і від значення, яке приймає аргумент N : v - вектор (матриця один з розмірів якої рівний 1)

- $N = 2$ або цей аргумент відсутній - евклидова норма вектора v

$$\|v\|_2 = \left(\sum_{i=1}^n v_i^2 \right)^{1/2}$$

- $N = 1$ - перша норма вектора v

$$\|v\|_1 = \sum_{i=1}^n |v_i|$$

- $N = \infty$ - нескінченна (чебишевська) норма вектора v

$$\|v\|_\infty = \max_i |v_i|$$

• $N = \text{inf}$ - норма вектора v вигляду

$$\|v\|_{-\infty} = \min_i |v_i|$$

• $N = p$ - для будь-якого p величина вигляду

$$\|v\|_p = \left(\sum_{i=1}^n v_i^p \right)^{1/p}$$

$y = \text{norm}(A, N)$, де A - матриця

• $N = 2$ або цей аргумент відсутній - спектральна норма матриці A рівна найбільшому сингулярному числу цієї матриці

$$\|A\|_2 = \max_i \lambda_i^{1/2}(A^T A) = \max_i \sigma_i(A)$$

• $N = 1$ - перша (стовпчикова) норма матриці A

$$\|A\|_1 = \max_j \sum_{i=1}^n |A_{ij}|$$

• $N = \text{inf}$ - нескінченна (рядкова) норма матриці A

$$\|A\|_{\infty} = \max_i \sum_{j=1}^m |A_{ij}|$$

• $N = \text{'fro'}$ - фробеніусова норма матриці A

$$\|A\|_F = \text{trace}^{1/2}(A^T A) = \left(\sum_{i=1}^n \sum_{j=1}^m A_{ij}^2 \right)^{1/2}$$

1.7.7 Елементарні операції над матрицями

На початку параграфа нагадаємо, що транспонування матриці проводиться оператором $'$ ($A^T = A'$).

- $[N, M] = \text{size}(A)$ - розміри матриці (A - матриця розміру $N \times M$;
- $L = \text{length}(A)$ - максимальний розмір матриці ($L = \max(\text{size}(A))$);
- $y = \text{det}(A)$ - визначник матриці;
- $y = \text{trace}(A)$ - слід матриці;
- $y = \text{inv}(A)$ - обернення матриці;
- $y = \text{pinv}(A)$ - псевдообернення матриці ($y = (A^T A)^{-1} A^T$);
- $y = \text{sqrtn}(A)$ - корінь квадратний з матриці;
- $y = \text{expm}(A)$ - експонента від матриці;
- $y = \text{logm}(A)$ - логарифм матриці.
- $v = \text{poly}(A)$ - обчислює вектор коефіцієнтів характеристичного полінома (коефіцієнти розташовані у векторі в порядку убавання

ступенів).

- $[V, D] = eig(A)$ - обчислення власних чисел і власних векторів матриці A . Діагональні елементи матриці D (жорданової канонічної форми) є власними числами матриці A , а стовпці матриці V є власними векторами. За наявності єдиного вихідного параметра $L = eig(A)$ - вихід L - вектор-стовпець власних чисел матриці;
- $[U, S, V] = svd(A)$ - обчислення сингулярного розкладання матриці A . Матриця A представляється у вигляді добутку трьох матриць $A = USV^T$. Тут матриці U і V - ортогональні матриці, S - діагональна матриця сингулярних чисел

$$\sigma_i(A) = \lambda_i^{1/2}(A^T A)$$

При оберненні $S = svd(A)$ видається тільки матриця сингулярних чисел S .

- $y = rank(A)$ - ранг матриці;

Тут слід уточнити деякі подробиці про те, що в MATLAB'і розуміється під рангом. MATLAB оперує з чисельними оцінками і обчислювальними операціями. Відповідно до цього при достатньо малих, хоч і ненульових значеннях детермінанта чисельне обернення матриці виявляється неможливим. Таким чином при обчисленнях вводиться узагальнене поняття рангу таким чином: рангом вважається кількість сингулярних чисел матриці, що перевищують заданий поріг $tol = max(size(A)) * \|A\|_2 * eps$.

З особливостями обчислювальних процедур при зверненні матриць, обчисленні власних чисел і інших операціях зв'язаний також ряд важливих функцій MATLAB'a. З теоретичними основами відповідних обчислювальних алгоритмів можна познайомитися, наприклад, в книгах [14] і [15]. Тут же ми приведемо тільки функції пакету MATLAB, що виконують відповідні операції.

- $c = cond(A)$ - Обчислює число обумовленості матриці $\mu = \|A^{-1}\|_2 * \|A\|_2 = \sigma_{max}(A) / \sigma_{min}(A)$
- $[T, B] = balance(A)$ - балансування матриці використовується для поліпшення обумовленості матриці. Процедура балансування полягає в тому, що шукається діагональна матриця T така, що матриця $B = T^{-1} * A * T$ має приблизно рівні стовпчикову і рядкову норми $\|B\|_1 \approx \|B\|_\infty$. Така процедура еквівалентна масштабуванню змінних вирішуваної задачі.

Стандартна програма eig обчислення власних чисел автоматично проводить балансування матриць. Для відмови від цієї процедури необхідно додати в звернення другий аргумент - текстову константу 'nobalance' ($[V, D] = eig(A, 'nobalance')$).

- $Z = null(A)$ - обчислення ортонормального базису нульового

підпростору матриці. Базис нульового підпростору утворюють стовпці матриці Z . Обернення $Z = null(A', r')$ приводить до обчислення "раціонального" базису, коли компонентами векторів є раціональні числа.

- $Q = orth(A)$ - обчислення ортонормального базису матриці. Стовпці матриці Q утворюють той же підпростір, що і стовпці матриці A , причому $Q^T Q = I$.

Для ознайомлення приведемо приклади обчислення погано обумовлених матриць. Погано обумовленою з погляду обернення є квадратна матриця Гільберта високого порядку. Її елементи обчислюються за формулою $R = 1/(i+j - 1)$, $i, j = 1 \dots n$ - номери рядків і стовпців, a_n - розмірність матриці. У MATLAB'і для формування цієї матриці можна використовувати функцію $a = hilb(n)$, де n - розмірність квадратної матриці a . Розглянемо простий приклад матриці a Гільберта при $n = 6$ і матриці b , всі компоненти якої рівні компонентам матриці a , окрім однієї. Ця компонента b_{16} відрізняється від компоненти a_{16} на 1%. Розглянемо наступну послідовність обчислень:

$a = hilb(6)$

$a =$

1.0000	0.5000	0.3333	0.2500	0.2000	0.1667
0.5000	0.3333	0.2500	0.2000	0.1667	0.1429
0.3333	0.2500	0.2000	0.1667	0.1429	0.1250
0.2500	0.2000	0.1667	0.1429	0.1250	0.1111
0.2000	0.1667	0.1429	0.1250	0.1111	0.1000
0.1667	0.1429	0.1250	0.1111	0.1000	0.0909

$b = a$

$b(1,6) = b(1,6) * 1.01$

$(b^{(-1)} - a^{(-1)}) ./ a^{(-1)}$

$ans =$

-1.2762	-2.1878	-2.8715	-3.4033	-3.8287	-4.1768
-1.2762	-1.6409	-1.9144	-2.1271	-2.2972	-2.4365
-1.2762	-1.4586	-1.5953	-1.7017	-1.7867	-1.8564
-1.2762	-1.3674	-1.4358	-1.4890	-1.5315	-1.5663
-1.2762	-1.3127	-1.3401	-1.3613	-1.3783	-1.3923
-1.2762	-1.2762	-1.2762	-1.2762	-1.2762	-1.2762

$eig(a)$

ans =

0.0000
0.0000
0.0006
0.0163
0.2424
1.6189

eig(b)

ans =

1.6191
0.2420
0.0165
0.0006
0.0000
-0.0000

d=cond(a)

d =

1.4951e+007
det(a)

ans =

5.3673e-018

У приведенному прикладі відносна помилка обчислення компонент зворотної матриці складає більше 100% при помилці завдання однієї з компонент матриці в 1%. При цьому власні числа обох матриць на перший погляд обчислюються цілком прийнятно. Основною ознакою можливого неблагополуччя в даному прикладі слід вважати велике значення числа обумовленості і мале значення визначника матриці.

Приклад поганої обумовленості при обчисленні власних чисел дає функція *gallery*. (Ця функція дозволяє обчислювати значення різних "стандартних" матриць. Список цих матриць і параметри можна подивитися за допомогою команди *help gallery*)

Розглянемо приклад

```
> a=gallery(5)
```

```
a =
```

```
   -9    11   -21    63   -252  
   70   -69   141  -421   1684  
 -575   575 -1149   3451 -13801  
 3891 -3891   7782 -23345  93365  
 1024 -1024   2048  -6144  24572
```

```
> b=a;
```

```
> b(5,1)=b(5,1)*1.01;
```

```
> [eig(a),eig(b)]
```

```
> ans =
```

```
ans =
```

```
-0.0408    -0.9201 +50.8135i  
-0.0119 + 0.0386i -0.9201 -50.8135i  
-0.0119 - 0.0386i -0.0000  
 0.0323 + 0.0230i  0.9201 + 0.4053i  
 0.0323 - 0.0230i  0.9201 - 0.4053i
```

```
> cond(a)
```

```
ans =
```

```
2.0253e+018
```

Власні числа матриць a і b в даному прикладі відрізняються на два порядки. Ознакою такої поведінки власних чисел служить число обумовленості 10^{18} .

На закінчення цього параграфу відмітимо, що в пакеті *MATLAB* реалізований багатий набір функцій матриць, що здійснюють приведення, до різних канонічних форм, таким як форми Жордана, Фробеніуса, Шура, Хессенберга, а також часто використовувані функції *lu* і *qr* розкладань.

2 Програмування в пакеті MATLAB. Умовні переходи, цикли, перемикачі

У цьому параграфі обговоримо організацію основних елементів програмування в пакеті MATLAB - умовні переходи і організацію циклів.

2.1 "Логічні"змінні

У пакеті MATLAB існує 6 логічних операцій відношення: $<$, $>$, $<=$, $>=$, $==$, \sim . Ці операції виконують поелементне порівняння двох матриць. Результатом є матриця того ж розміру, елементи якої рівні 1, якщо відповідна умова виконується, і рівні 0 інакше.

Аналогічно визначені логічні операції:

- $\&$ - логічне I (AND). Результатом операції $A\&B$ є матриця того ж розміру, кожен компонент якої рівний 1, якщо обидва

відповідні компоненти матриць A і B не рівні 0, і рівний 0 інакше (тобто, якщо хоча б один з компонент дорівнює 0).

- / - логічне АБО (OR). Результатом операції A/B є матриця того ж розміру, кожен компонент якої рівний 1, якщо хоч би один з відповідних компонент матриць A і B не рівні 0, і рівний 0 інакше (тобто, якщо обидва компоненти рівні 0).

\sim - логічне НІ (NOT). Результатом операції $\sim A$ є матриця того ж розміру, кожен компонент якої рівний 1, якщо відповідний компонент матриці A рівний 0, і рівний 0 інакше (тобто якщо компонент не рівний 0).

Наприклад

```
>> a=[1 2; 3 4]
```

```
a =
```

```
1 2
```

```
3 4
```

```
>> b=[4 2; 3 1]
```

```
b =
```

```
4 2
```

```
3 1
```

```
>> c = a >= b
```

```
c =
```

```
0 1
```

```
1 1
```

```
>> d = ~ (b <= c) & a
```

```
d =
```

```
1 1
```

```
1 0
```

2.2 Умовний оператор

У пакеті MATLAB для програмування умовних переходів використовується умовний оператор, який в загальному вигляді записується таким чином:

```
if Логічна змінна 1,  
    Інструкції 1  
elseif Логічна змінна 2,  
    Інструкції 2  
else  
    Інструкції 3  
end
```

В якості логічної змінної може бути використано також логічний вираз, тобто права частина привласнення описаного в попередньому пункті.

Під інструкціями тут розуміються набір команд, які виконуватимуться, якщо всі елементи матричної "логічної" змінної – ненульові.

У приведеному умовному операторі конструкції, що починаються з *elseif* і *else*, можуть бути опущені.

Для ясності приведемо два наступні приклади

```
>> a=[1 0; 0 1]
```

```
a =
```

```
1 0  
0 1
```

```
>> b=[1 1; 1 1]
```

```
b =
```

```
1 1  
1 1
```

```
>> d=[1 1; 1 0]
```

```
d =
```

```
1 1  
1 0
```

```
if d, c = a; else c = b; end; c
```

```
c =
```

```
1 1  
1 1
```

```
>> if a <= b , c = a; else c = b; end; c
```

c =

```
1 0
0 1
```

Також ці операції використовуються звичайним способом на випадок скалярних величин, і саме таке традиційне звернення до оператора порівняння можна зустріти в програмах *if a < b*

```
    c=a;    elseif
a==b,
    c=1;
else
    c=b;
end
```

2.3 Організація циклів

При програмуванні в пакеті MATLAB можливе використання двох видів циклів

- цикл типу *for-end*
- цикл типу *while*

Перша з цих конструкцій - цикл типу *for-end* - використовується для організації обчислень із заданим числом циклів, що повторюються. Конструкція такого циклу має вигляд:

```
for var = Вираз,
    Інструкції
end
```

Як "Вираз" в запропонованій конструкції можна використовувати довільну матрицю розміру $m \times n$. В цьому випадку "Інструкції" тіла циклу повторюватимуться n разів, причому параметр *var* приймає послідовно векторні значення, співпадаючі із стовпцями матриці. Це добре видно з наступного прикладу:

```
>> a = [1 2 3; 4 5 6]
```

```
a =
```

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

```
>> for i = a; i, end
```

```
i =
```


1
4

i =

2
5

i =

3
6

Проте зазвичай "вираз" записується в більш традиційному вигляді і звернення до циклу *for-end* в програмах має вигляд:

```
for i = 1 : 3, for j = 1 : 3  
a(i,j)=i+j;  
end  
end
```

В результаті виконання *script-файлу*, з вказаним набором операторів буде отримана матриця

a =

2	3	4
3	4	5
4	5	6

(3)

Другий тип циклів дозволяє виконувати інструкції в тілі циклу до тих пір поки виконується задана умова. Конструкція такого циклу має вигляд:

```
while Логічна змінна, Інструкції  
end
```

У цьому виразі "Логічна змінна" має той же сенс, що і для оператора *if* в розділі 2.2 і обчислення продовжуються до тих пір, поки в матриці використаної як "Логічна змінна" немає нульових компонентів. Так само як і в 2.2 в конструкції *while* загально прийнято використання умовного оператора з параграфу 2.1. У зв'язку з цим приведемо традиційний і нетрадиційний приклади використання *while*.

Традиційний приклад:

```
p = 1; k = 0; while p < 10, p = p * 2; k = k + 1; end; k
```

Обчислює мінімальне значення показника ступеня k таке, що $2^k \geq 10$. Очевидно, що після закінчення роботи циклу $k = 4$ і $p = 16$.

Менш традиційним слід рахувати приклад вигляду

```
a = [2.5, 3; 4, 5]; while a, a = a - 1; end
```

В даному прикладі задана матриця a перетвориться наступним способом: всі елементи її зменшуються на кожному кроці на 1, до тих пір, поки одне з них не стане рівним 0. Таким чином процедура завершується якщо хоч би одна з компонент матриці a - додатне ціле число – буде дорівнювати 0. Якщо це не так, то відбудеться зациклення програми.

Для захисту від такої ситуації можна використовувати оператор *break*, який припиняє дію останнього по вкладенню циклу *for-end* або *while*, всередині якого розташований оператор *break*. Попередній приклад перерваний таким оператором, переписаний у файл прийме вигляд:

```
a = [2.5, 3; 4, 5]
while a,
a = a - 1;
if a < 0, break
end
end
```

2.4 Перемикач *switch-case-otherwise-end*

Конструкція перемикача *switch* використовується для множинного вибору (або розгалуження). Вона визначена у версіях MATLAB'a старше 5-ї і в загальному випадку має вигляд:

```
switch switch-вираз, case Case-вираз
    Інструкції 1 case { case-вираз 1, Case-вираз 2, Case-вираз 3 ... }
    Інструкції 2
otherwise
    Інструкції 3
end
```

Як *switch*-вираз використовується будь-який з описаних раніше об'єктів MATLAB'у. Якщо поточне значення цього виразу співпадає з одним з *Case*-виразів, виконуються оператори з відповідних "Інструкцій". Якщо жодне з *Case*-виразів не підійшло, виконуються "Інструкції", наступні за оператором *otherwise*.

Як приклад розглянемо сценарій *asw.m*:

```
switch var
case {1,2,12,'January','February','December'}
disp('Winter')
case {3,4,5,'March','April','May'}
disp('Spring')
case {6,7,8,'June','July','August'}
```

```

disp('Summer')
case {9,10,11,'September','October','November'}
disp('Autumn')
otherwise
disp('It is not a month.')
end

```

Звернення до цього сценарію приводить до наступних результатів:

```

> var = 3;
> asw
Spring
> var = 'May';
> asw
Spring
> var = 10;
> asw
Autumn
> var = 'Autumn';
> asw
It is not a month.

```

3 Спеціальні можливості при зверненні до функцій

Докладний розгляд стандартних функцій пакету MATLAB дозволяє зробити висновок про реалізацію додаткових нетривіальних можливостей при створенні функцій пакету, таких як виконання різних обчислень залежно від числа і типу аргументів і вихідних параметрів функції, передача функції імені функції, що викликається, передача довільного числа аргументів цієї допоміжної функції і т.д.

Зрозуміло, що виконання різних обчислень залежно від числа аргументів і вихідних параметрів функції нескладно організувати, використовуючи параметри *nargin* та *nargout*, вже згадувані раніше, і оператори *if* або *switch*. Також можна перевірити розмірність і тип аргументів.

Деякі особливості MATLAB'у, які будуть використані у функціях, описані нижче, розглянемо в цьому розділі. Для того, щоб користуватися функціями, описаними в подальших розділах, читати цей розділ необов'язково, але його зміст допомагає зрозуміти, які влаштовані ті або інші функції і дозволить акуратніше, писати власні функції з аналогічними властивостями.

3.1 Звернення до функції по імені

У ряді випадків (наприклад при рішенні нелінійних рівнянь або чисельній інтеграції звичайних диференціальних рівнянь) необхідно передати функції як аргумент ім'я деякої іншої функції (наприклад, ім'я функції обчислення правих або лівих частин рівнянь), яка буде викликана по цьому імені. Ім'я функції або

змінної у такому разі зберігається у вигляді символьних одновимірних масивах (векторах). Такий масив можна передати функції як формальний параметр.

Виконання відповідної команди або виклик функції дозволяють зробити функції *eval* і *feval*.

Функція *eval* дозволяє виконати команду системи MATLAB. Звернення до неї має вигляд

```
eval(command)
```

Текстова змінна *command* тут містить текст команди. Як приклад приведемо уривок програми, в якому створюються 5 матриць Уїлкінсона розмірності від 5 до 10, імена яких встановлюються *M5*, *M6*, *M7*, *M8*, *M9*, *M10* відповідно.

```
for k=5:10
```

```
eval(['M',num2str(k), '=wilkinson(',num2str(k);'])
```

```
end
```

Функція *feval* дозволяє по імені викликати функцію MATLAB'a. Звернення до неї має вигляд

```
[outpars]=feval(FunName,inpar1,inpar2...)
```

Текстова змінна *FunName* тут містить ім'я функції, що викликається, а аргументи *gn-par1,inpar2...* - містять аргументи функції, що викликається. Через *[outpars]* позначений список результатів виконання функції, що викликається.

Наприклад, при зверненні

```
v=feval(fname, 0,10,50);
```

при *fname = 'linespace'* формується вектор, 50 компонент якого рівномірно розподілені на сегменті [0,10], а при *fname = 'logspace'* ці компоненти матимуть логарифмічний розподіл.

3.2 Багатовимірні масиви

Разом з традиційними двовимірними масивами - матрицями, в пакеті MATLAB, починаючи з 5-ої версії введені багатовимірні таблиці. З їх допомогою в MATLAB'і відображаються тензорні величини. Тут ми приділяємо увагу таким масивам у зв'язку з тим, що вони використовуються при представленні результатів деякими функціями моделювання і дослідження систем управління.

Звернення до елементів багатовимірного масиву має традиційний вигляд *a(i1, i2, i3..., in)*. Ввести багатовимірний масив можна збільшуючи число розмірностей матриці. Розглянемо, наприклад, послідовність команд:

```
>> a=[1 2; 3 4]
```

```
a =
```

```
1 2
```

```
3 4
```

```
>> a(:, :, 2)=3
```

```
a(:, :, 1) =
```

```
1 2
```

```
3 4
```

```
a(:, :, 2) =
```

```
3 3
```

Створюють багатовимірний масив також і функції *ones*, *zeros*, *rand*, *randn*.
При зверненні вигляду

$$a = \text{ones}(n1, n2 \dots, nn)$$

ці функції повертають масив, розмірність якого дорівнює числу аргументів функції, причому *n1*, *n2...*, *nn* – "довжини" відповідних розмірностей. Вказані функції створюють багатовимірні масиви всі компоненти яких приймають відповідно значення: для *ones* - одиниця; для *zeros* - нуль; для *rand* - випадкове число рівномірне розподілене на інтервалі (0,1); для *randn* - нормально розподілене випадкове число з нульовим середнім і одиничною дисперсією. Перерахуємо тут основні функції, що дозволяють працювати з багатовимірними масивами

•функція

$$n = \text{ndims}(a)$$

дозволяє визначити число розмірностей багатовимірного масиву *a*.

•функція

$$n = \text{size}(a)$$

дозволяє визначити розмір багатовимірного масиву *a* по кожній розмірності.

•Функція $B = \text{cat}(dim, A1, A2, A3, A4 \dots)$ дозволяє об'єднати масиви *A1, A2, A3, A4...* у один уздовж розмірності *dim*. Розглянемо простий приклад

```
>> A1=ones(2);
>> A2=eye(2);
>> B1=cat(1,A1,A2)
B1 =
1 1
1 1
1 0
0 1
>> B2=cat(2,A1,A2)
B2 =
1 1 1 0
1 1 0 1
>> B3=cat(3,A1,A2)
B3(:,:,1) =
1 1
1 1
B3(:,:,2) =
1 0
0 1
```

Тут видно, що в першому випадку дія опинилася еквівалентна команді $B1 = [A1;A2]$, у другому випадку $B2 = [A1, A2]$, а в третьому функція створила тривимірний масив $B3$.

- Функція $B = \text{permute}(A, \text{order})$ дозволяє переставляти місцями розмірності масиву в порядку, що встановлюється цілочисельним вектором перестановок order . Наприклад звернення $C3 = \text{permute}(B3 [213])$ для масиву $B3$ з попереднього прикладу приведе до транспонування матриць $B3(:, :, 1)$ і $B3(:, :, 2)$.
- Функція $C = \text{ipermute}(B, \text{order})$ зворотна по відношенню до функції permute .
- Функція $B = \text{shiftdim}(B, n)$ проводить зрушення розмірностей вліво, на величину n . При $n < 1$ така дія еквівалентна циклічній перестановці розмірностей і збільшення їх числа не відбувається. При $n < 0$ відбувається зрушення розмірностей вправо. При цьому число розмірностей збільшується на $|n|$ і молодші розмірності заповнюються одиницями.
- Функція $C = \text{squeeze}(B)$ дозволяє видалити розмірності завдовжки 1. Річ у тому, що для MATLAB'у звернення $B3(:, :, 1)$ у використаному вище прикладі описує тривимірний масив, в той час, як насправді це матриця і розмірність її на 1 менші. Такий об'єкт не вдається використовувати при зверненні до стандартних матричних операторів і функцій. Перетворення цього об'єкту, що знижує число розмірностей і проводить функція squeeze .

3.3 Комірки і операції над ними

Масиви комірок використовуються в MATLAB'і для зберігання і передачі даних різних типів. Створити масив комірок можна двома способами: з використанням функції cell або виведенням об'єктів у фігурні дужки. Наприклад, якщо в робочу область пам'яті введена матриця a розміру 2×2 , наступна команда вводить масив комірок розміру 2×3 :

```
>>b={min(a) max(a) 3; 5 1 'text' }
```

```
b =
```

```
[1 . 2double] [1 . 2double] [ 3]
```

```
[ 5] [ 1] 'text'
```

Звернення $b = \text{cell}(N, M)$ створює порожній масив комірок розміру $N \times M$, а за відсутності аргументу M масив розміру $N \times N$. Для звернення до елементів масиву використовуються фігурні дужки, причому на такі звернення розповсюджуються всі основні синтаксичні правила для роботи з традиційними матрицями:

```
>>b{2,3}
```

```
ans =
```

```
text
```

```
>>b{:,1}
```

```
ans =
```

```
1 2
```

ans =

5

Багатовимірний масив комірок заповнюється за правилами, аналогічними правилам заповнення багатовимірних масивів. При цьому для масивів комірок визначені функції *squeeze* і *shiftdim*.

Відзначимо, що в масиві комірок зберігаються копії об'єктів і зміна самих об'єктів не змінює вмісту масиву комірок і навпаки. Іншими словами масив комірок не слід плутати з масивом адрес.

На закінчення приведемо декілька функцій і команд для роботи з масивом комірок.

- *celldisp(C)* - виводить на екран вміст масиву комірок *C*.
- *cellplot(C)* - виводить на екран вміст масиву комірок *C* у вигляді графічного вікна.
- *C = cellstr(s)* - перетворює масив рядків *s* в масив символічних комірок *C*.
- *C = iscell(C)* - повертає логічне *TRUE(1)*, якщо *C* - масив комірок, і *FALSE(0)* інакше.
- *C = num2cell(A)* - перетворить масив *A* в масив осередків, розміщуючи кожен елемент масиву в окремому осередку.

З поняттям масивів комірок пов'язана можливість опису списків вхідних і вихідних аргументів змінної довжини. Для цього використовуються змінні *varargin* і *varargout*.

Змінна *varargin* дозволяє об'єднати довільну кількість вхідних змінних. Вона є одновимірним масивом комірок, який містить аргументи функції, що викликається. Ця змінна повинна завершувати список вхідних аргументів функції.

Аналогічно змінна *varargout* дозволяє об'єднати довільну кількість вихідних змінних. Вона є одновимірним масивом комірок, що містить аргументи виходу функції, що викликається. Ця змінна повинна завершувати список вихідних аргументів функції.

Приведемо приклад опису такої функції:

```
function [z,varargout]=my1fun(a,b,varargin)
```

```
...
```

```
varargout=my2fun(b,varargin);
```

```
...
```

Прикладом використання цих змінних служить звернення до функцій типу *ode*, які обговорюватимуться в розділі 5.

3.4 Глобальні змінні

У попередньому параграфі описаний спосіб передачі довільного числа змінних допоміжної функції. Цей спосіб у багатьох випадках не є найбільш ефективним [12]. Ефективнішим і швидшим є спосіб передачі змінних з використанням конструкції `global`.

Команда

```
global ім'я1,ім'я2 ...
```

визначає змінні з іменами *ім'я1*, *ім'я2* ... як "глобальні". Якщо декілька функцій і сценарій оголошують деяку змінну глобальною, то всі вони використовують одну і ту ж копію цієї змінної. Зміна її значення в одній з функцій приведе до того, що при подальшому зверненні до всіх інших функцій вказана змінна приймає змінене значення. Це ж змінене значення приймає змінна з цим ім'ям в робочій області (та яку можна подивитися в командному вікні).

Приведемо приклад використання глобальних змінних

Звернення до цієї функції з командного вікна приводить до результату »

```
my1fun(3,4)  
5 7
```

При цьому в командному вікні і у функції *my2fun* змінна *z* або не визначена, або не міняє свого значення (якщо воно було задане).

```
function z=my1fun(a,b)
```

```
global c
```

```
c=b;
```

```
d=my2fun(a);
```

```
disp ([c,d])
```

```
...
```

```
function d=my2fun(b);
```

```
d=my3fun(b);
```

```
...
```

```
function d=my3fun(b);
```

```
global c
```

```
d=b+c;
```

```
c=5;
```


Звернення до цієї функції з командного вікна приводить до результату

```
» my1fun(3,4)
    5     7
```

При цьому в командному вікні і у функції *my2fun* змінна *C* або не визначена, або не змінює свого значення (якщо воно було задане).

4 Графічні функції пакету MATLAB

4.1 Двовимірна графіка

4.1.1 Креслення графіків в декартових координатах

Основною функцією малювання графіків в пакеті MATLAB є функція *plot*.

Звернення до цієї функції мають вигляд

- *plot(x, y)* будує графік, відкладаючи по осі абсцис значення компонент вектора *x* і по осі ординат значення компонент вектора *y*, що має ту ж довжину, що і вектор *x*. Надалі у всіх подібних випадках говоритимемо про побудову графіка *y* від *x*.

Побудова графіка полягає в нанесенні на координатну площину послідовних точок і з'єднанні їх прямими. Відповідно такий спосіб побудови не зв'язаний обмеженнями при відображенні неоднозначних функцій.

- *plot(y)* будує аналогічний графік, відкладаючи значення компонент вектора *y* по осі ординат, а по осі абсцис відкладає номер відповідною компоненти у векторі.

Для обох варіантів звернення до функції *plot* для заданої матриці *y* розміру $n \times m$ на одній координатній площині буде побудовано *m* графіків (поодиноці для кожного стовпця матриці *y*). З перших семи графіків кожен новий матиме свій колір. Далі кольори циклічно повторюються.

Наприклад, послідовне виконання команд

```
t=0:0.05:pi;
```

```
plot(t[sin(t);cos(t)]);
```

Ці дії викличуть відкриття графічного вікна MATLAB'а із зображенням, приведеним на рис. 1

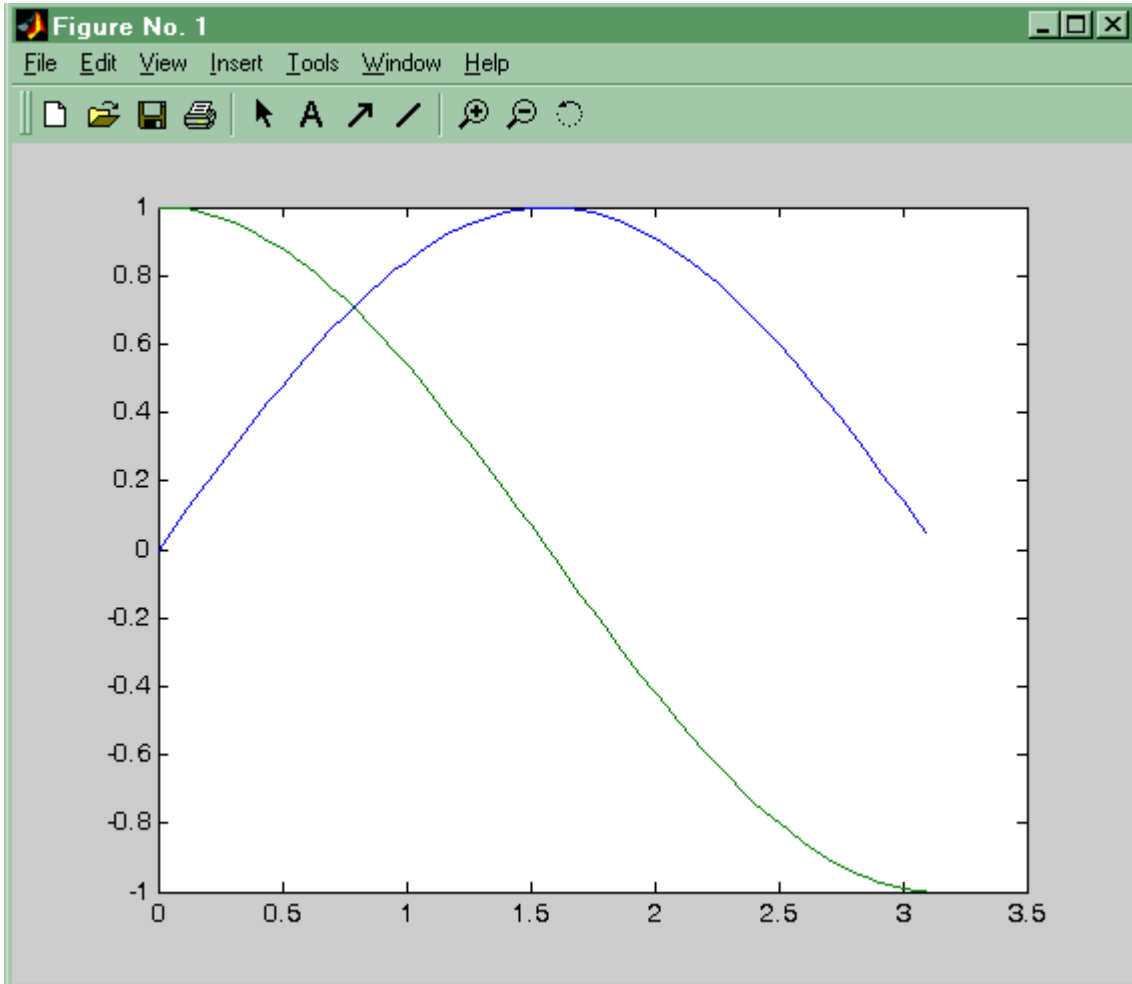


Рис. 1: Результат дії функції *plot*

Двовимірним може бути також і масив x . Якщо при цьому масив y є вектором, то будуються графіки, на яких по осі абсцис відкладаються значення компонент векторів, - стовпців матриці x , а по осі ординат значення компонент вектора y . Якщо x і y – двовимірні масиви однієї розмірності, будуються залежності стовпців матриці y від стовпців матриці x .

- $plot(x, y, s)$.

Це звернення дозволяє побудувати графік функції, вказавши в рядковій константі s колір і спосіб відображення лінії, а також вид вузлових точок. Як і раніше вектора (або матриці) x і y задають значення абсцис і ординат точок на графіці, а рядкова константа s може містити по одному символу з трьох наборів, наведених в таблиці 1:

Таблиця 1 – значення змінної s

	Колір		Вузлова точка		Вигляд лінії
y	жовтий (yellow)	.	точка (•)	-	суцільна
m	фіолетовий (magenta)	o	коло (O)	:	пунктир
c	голубий (cyan)	x	хрест (x)	-.	штрих- пунктир
r	червоний (red)	+	плюс (+)	—	штрихова
g	зелений (green)	*	зірочка (*)		
b	синій (blue)	s	квадрат		
w	білий (white)	d	ромб		
k	чорний (black)	v	набла (∇)		
		^	трикутник (Δ)		
		<	трикутник (лівий)		
		>	трикутник (правий)		
		p	пятикінцева зірка		
		h	шестикінцева зірка		

Приведені вище способи звернення до функції *plot* мають істотний недолік, оскільки не дозволяють будувати в одних координатних осях декілька графіків у яких вектора абсцис і ординат мають різну довжину. Ця трудність долається наступними способами.

Перший з них - звернення
`plot(x1, y1, s1, x2, y2, s2, x3, y3, s3. ..).`

В цьому випадку в одних і тих же координатних осях будуть побудовані графіки $y1$ від $x1$ відповідно до рядкової константи $s1$, графіки $y2$ від $x2$ відповідно до рядкової константи $s2$ і т.д. При цьому звичайно ж повинна дотримуватися відповідність розмірів серед пар масивів $x1$ і $y1$, $x2$ і $y2$ і т.д., але число точок в різних парах може не співпадати. Як приклад, наведемо наступну послідовність команд

```
t=0:0.05:pi;
t1=0:0.05:2*pi;
plot(t,sin(t), 'b-',t1,cos(t1), 'r:');
```

В результаті у відкритому графічному вікні MATLAB'a буде побудовано зображення, приведене на рис. 2

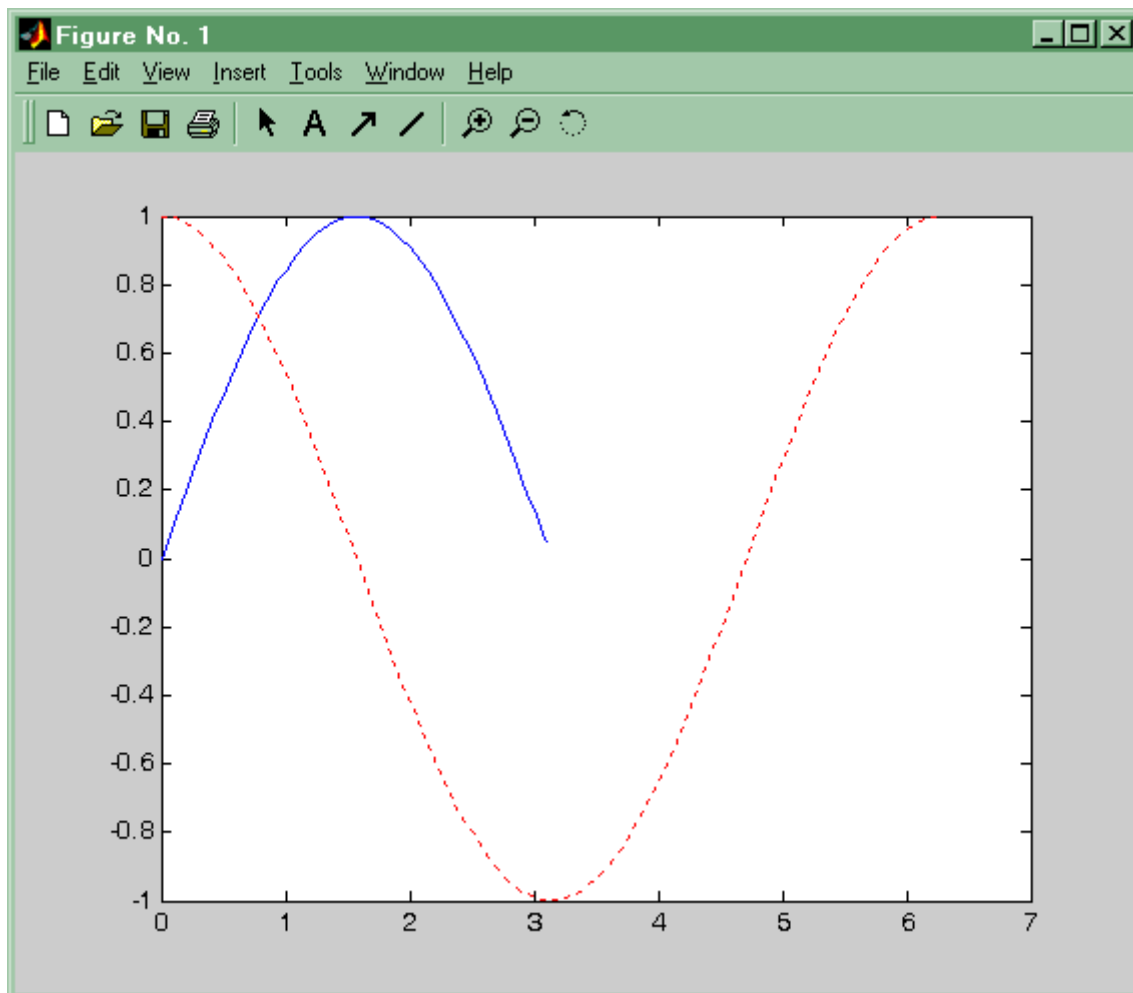


Рис. 2 Ще один приклад дії функції *plot*

Інший спосіб побудови декількох різних графіків пов'язаний з поетапною побудовою графіків в режимі збереження вмісту графічного вікна. Як читач вже відмітив, при побудові чергового графіка MATLAB за замовчуванням повністю перемальовує зображення в графічному вікні. Це означає, що діє режим збереження координатних осей. Для його вклучення служить оператор *hold*.

- Звернення *hold {on/off}* вклучає або відключає режим збереження координатних осей.
- Звернення *hold* змінює поточний стан режиму збереження координатних осей на протилежний.
- Нарешті, функція $k = ishold$ видає інформацію про поточний стан режиму: $k = 1$ при вклученому режимі, і $k = 0$ інакше.

Зображення на рис. 2 можна, наприклад, отримати з використанням наступної послідовності команд

```
t=0:0.05:pi;
hold on
plot(t,sin(t), 'b-')
t=0:0.05:2*pi;
plot(t,cos(t), 'v:');
hold off
```

Завершального оператора *hold* відключає режим збереження вмісту графічного вікна, і використаний для відновлення первинного стану прийнятого за замовчуванням.

На закінчення відзначимо, що при зверненні
comet(x,y,p)

функція будує графік залежності y від x поточною, показуючи в часі послідовність малювання точок графіка. Пояснити, як працює ця функція достатньо важко, оскільки для цього треба описувати процес, що розгортається в часі. Простіше запропонувати читачеві виконати команди

t=0:0.0001:pi;

*comet(sin(2*t),cos(t))*

і побачити самому, як виглядає результат. Необов'язковий параметр p управляє довжиною "хвоста".

4.1.2 Графіки в інших системах координат

У теорії управління і ряду інших областей часто використовуються логарифмічні осі, коли уздовж осі відкладаються значення не самої величини, а її логарифма. Графіки в таких осях дозволяють будувати наступні функції:

- *semilogx(x,y)* - будує графік з логарифмічним масштабом осі абсцис;
- *semilogy(x,y)* - будує графік з логарифмічним масштабом осі ординат;
- *loglog(x,y)* - будує графік з логарифмічним масштабом по обох осях.

Всі ці функції допускають такі ж форми звернення, як і описана раніше функція *plot*.

Аналогічна побудова графіків в полярних координатах, що задаються кутом ϕ і радіусом ρ , здійснює функція

polar(phi,rho,s)

Як приклад приведемо команди, що дозволяють побудувати криву, звану равликом Паскаля. Її рівняння в полярних координатах має вид

$\rho = a \cos \phi + l$. При $a = 0.5$ і $l = 0.15$

*phi=0:0.05:2*pi; polar(phi,0.5*cos(phi) + 0.15)*

Результат виконання цих команд приведений на рис. 3.

4.1.3 Сітка, написи і пояснення на графіках

Всі попередні графіки, побудовані тут, не мали сітки і не містили ніяких написів, тим часом вони необхідні при оформленні результатів обчислень.

Показати сітку на графіці дозволяє команда *grid*, яка послідовно проводить включення і відключення сітки. Команда *grid on* додає сітку до поточного графіка, а команда *grid off* відповідно відключає сітку.

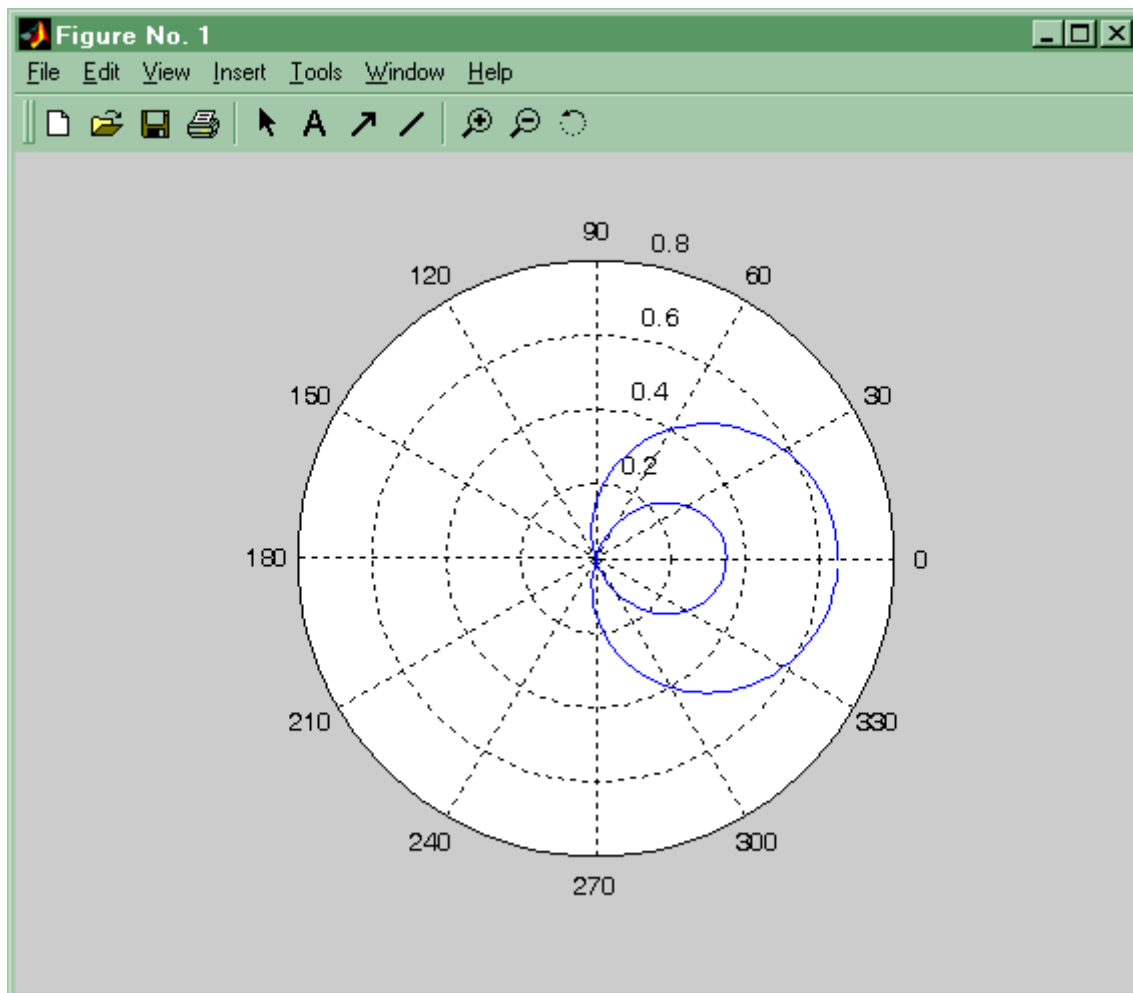


Рис. 3 Результат дії функції *polar*

Зробити написи на малюнках допомагають наступні функції

- *title(stext)* - виводить в графічне вікно символічну константу *stext* як надрисунковий напис;
- *xlabel(stext)* - виводить в графічне вікно символічну константу *stext* як напис до осі *x*;
- *ylabel(stext)* - виводить в графічне вікно символічну константу *stext* як напис до осі *y*;
- *text(x, y, stext)* - виводить в графічне вікно символічну константу *stext* як напис, лівий верхній кут якого поміщається в точці з координатами *x, y*. Якщо *x* та *y* є вектора, текст поміщається у всі позиції, що задаються цими векторами. Координати текстів задаються у фізичних координатах;
- *gtext(stext)* - виводить в графічне вікно символічну константу *stext* як напис, лівий верхній кут якого поміщається в точці з координатами, вказуваними мишею;
- *legend(stext1, stext2...)* - створює в графічному вікні пояснення до графіків. Відповідність текстових констант *stext1, stext2 ...* лініям на графіці відповідає порядку виведення графіків. Можливі також

наступні форми звернення до функції:

- $legend(M)$, де M - масив рядків однакової довжини.
- $legend off$ видаляє пояснення.
- $legend(stext1, stext2..., n)$ - цілий аргумент n встановлює граничне значення позицій для розміщення пояснення, $n = -1$ - пояснення розміщується поза область графіка, $n = 0$ - пояснення розміщується в області графіка, якщо місця для цього достатньо.

Продемонструємо використання цих функцій на наступному прикладі

```
t=0:0.05:2*pi;  
plot(t,[sin(t);cos(t);sin(2*t)])  
title('Тригонометричні функції')  
xlabel('Абсциса')  
ylabel('Ордината')  
text(5.5,0.2,'2*pi')  
legend('sin(t)','cos(t)','sin(2t)',0)
```

Результати виконання цих команд наведено на рис. 4.

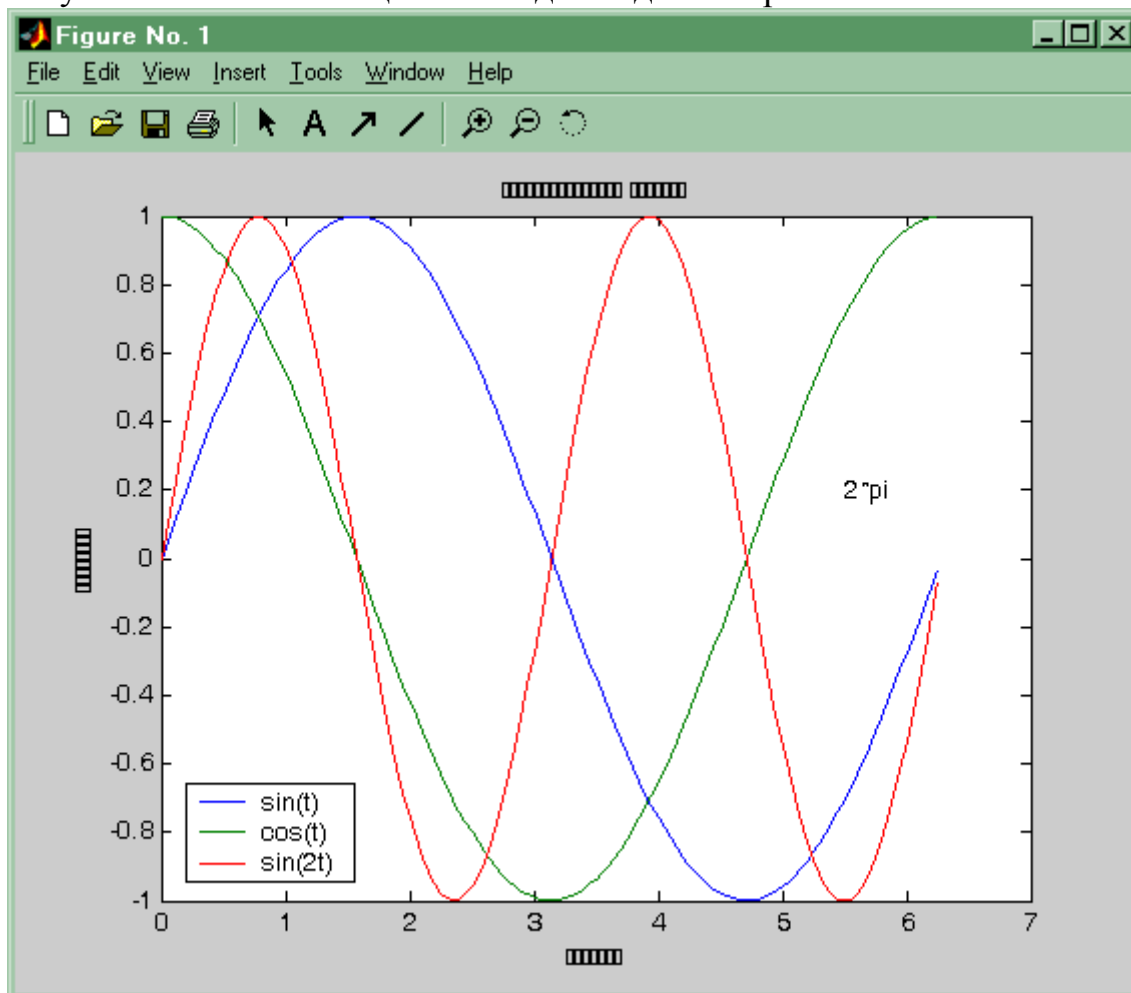


Рис. 4 Зразок написів до графіків

4.1.4 Декілька графіків в одному вікні

Проглянувши набір різних демонстраційних програм MATLAB'у, ви напевно відмітили, що для зручності аналізу результатів обчислень часто будуються декілька графіків в одному графічному вікні. Проводити такі побудови дозволяє функція *subplot*. Можливі наступні форми звернення до цієї функції

- $hp = subplot(n,m,p)$ і $hp = subplot(nmp)$ - створює розбиття поточного графічного вікна на $n \times m$ "підвікон". При цьому n число "підвікон" по горизонталі, а m - число "підвікон" по вертикалі. У "підвікні" з номером p ця команда будує осі і робить їх активними для подальшого звернення до функції *plot*. Номер p указує на j -е "підвікно" в i -м ряду, де $i = floor(p/n)$, а j залишок від ділення ($j = rem(p,n)$). Інакше кажучи, для "підвікон" прийнята наскрізна нумерація по рядах зліва направо і зверху вниз. Повертає функція *subplot* покажчик hp - адреса структури для осей, розташованих в p -ом "підвікні".

При виклику цієї функції для існуючої системи "підвікон" у разі, коли величини n і m не змінилися, функція встановлює активним "підвікно" з номером p . Якщо хоч би одне з чисел m і n змінилося, функція стирає попередній вміст графічного вікна, і проводить розбиття наново.

- *subplot(hp)* - встановлює активним "підвікно"с покажчиком (адресою) hp .
- *subplot('position'[left bottom width height])* - дозволяє створити "підвікно" в області графічного вікна, що указується програмно. У цьому обігу 'position'- ключове слово, що повідомляє функції про те, що далі буде задана прямокутна область для розміщення осей "підвікна", а *left bottom width height* аргументи задають положення лівого нижнього кута "підвікна", а також його ширину і висоту в долях від розмірів графічного вікна.

Продемонструємо відповідні можливості на наступному прикладі

```
t=0:0.05:4*pi;
h1=subplot(221);
plot(cos(t).*exp(-0.1*t),sin(t).*exp(-0.1*t))
h2=subplot(222);
polar(t,exp(-0.1*t))
h3=subplot('position',[0.1,0.1,0.8,0.35])
plot(t,sin(t).*exp(-0.1*t),'-',t,cos(t).*exp(-0.1*t),'_')
xlabel('Время,с, Угол, рад.')
ylabel('Координати, м')
subplot(h2)
title('Траектория в полярних координатах ')
subplot(2,2,1)
xlabel('Дальність, м.')
ylabel('Висота, м')
```


title('Траекторія в декартових координатах')

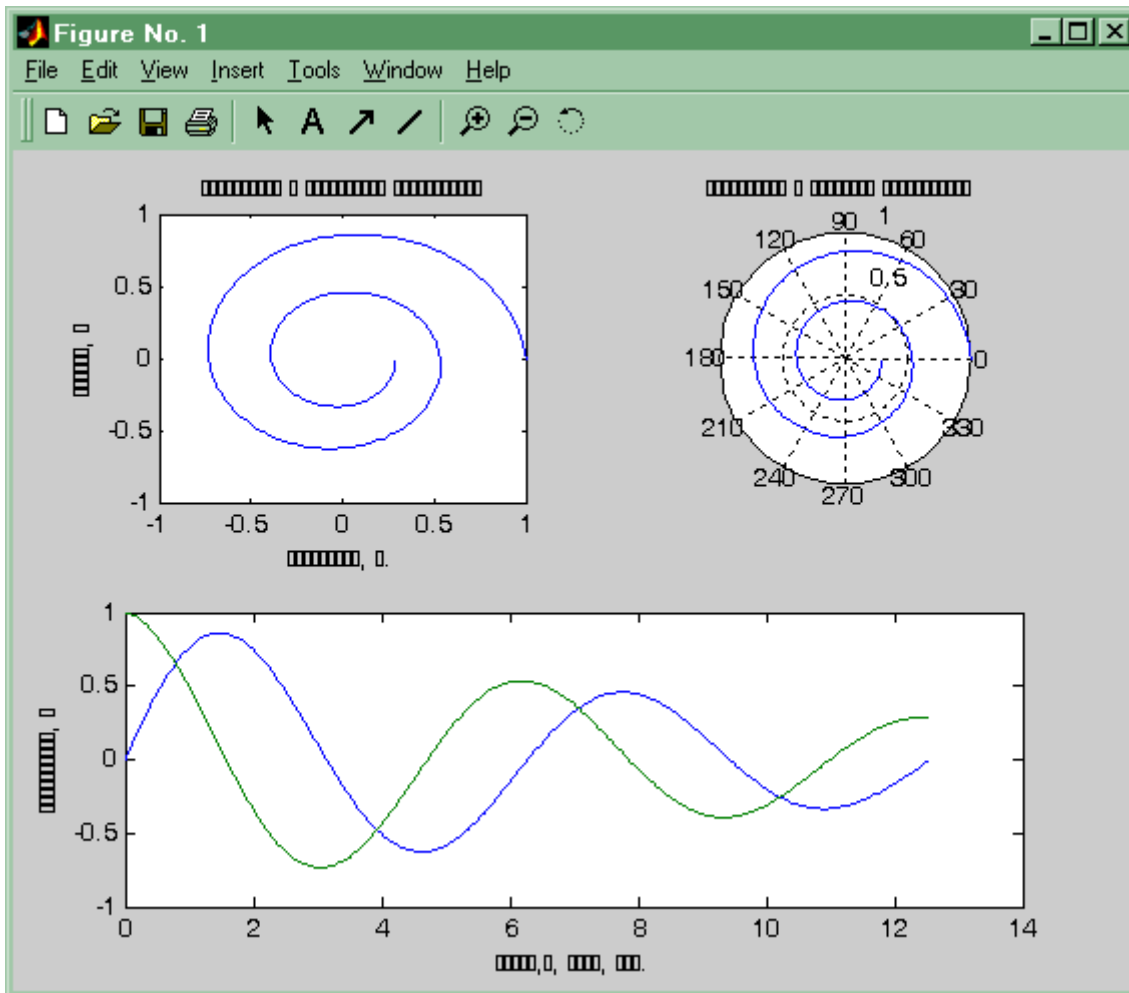


Рис. 5 Розбиття графічного вікна на "підвікна"

Звичайно ж в даному прикладі надруковані заголовки і підписи під координатними осями можна було б зробити відразу після побудови графіків функцією *plot*. Повернення до відповідних осей проводиться спеціально для демонстрації повернення за допомогою функції *subplot*.

У "підвікні", взагалі кажучи, можна будувати будь-яке зображення, в тому числі і тривимірне, робота з тривимірними зображеннями буде розглянута нижче.

4.1.5 Графічні вікна і управління ними

Звичайно ж MATLAB допускає одночасне створення декількох графічних вікон і креслення графіків в них.

- Для створення нового графічного вікна використовуємо функцію *figure*. За відсутності формальних параметрів звернення $h = \text{figure}$ створює нове графічне вікно і робить його активним. Тепер всі викликані функції побудови графіків будуватимуть зображення в цьому графічному вікні. Звернення $h = \text{figure}(N)$ для будь-якого цілого числа N створює нове графічне вікно з номером N , якщо такого вікна не існувало, і робить його

активним. У обох випадках функція повертає покажчик на вказане графічне вікно. Якщо h покажчик на графічне вікно звернення $figure(h)$ робить це графічне вікно активним.

- Дізнатися покажчик на активне графічне вікно дозволяє звернення $h = gcf$.
- Очистити активне графічне вікно дозволяє оператор clf .
- Закрити активне графічне вікно дозволяє оператор $close$. Звернення $close(h)$ закриває графічне вікно з покажчиком h , а $close(N)$ графічне вікно з номером N . Звернення $closeall$ закриває всі відкриті графічні вікна.

Важливу можливість розглянути деталі на графіках в графічному вікні дає оператора $zoom$. Звернення до нього може приймати наступні форми

- $zoom$ - перемикає стан режиму зміни масштабу графіка.
- $zoom on$ - включає режим зміни масштабу графіка.
- $zoom off$ - вимикає режим зміни масштабу графіка.
- $zoom xon$ і $zoom yon$ - включає режим зміни масштабу графіка тільки по одній з осей x або y відповідно.
- $zoom out$ - відновлює початковий масштаб графіків в активному вікні.
- $zoom reset$ - встановлює поточний масштаб графіків в активному вікні як результатне (незміншуваного початку відліку).
- $zoom(F)$ - встановлює масштаб графіків в активному вікні відповідно до параметра F , іншими словами збільшує зображення в F разів ($F > 1$).

Масштабування графіка при основних формах звернення до оператора $zoom$ проводиться за допомогою миші. Для цього курсор миші треба підвести до центру області малюнка, що цікавить користувача. Якщо режим $zoom$ включений, то натиснення лівої клавіші збільшує масштаб удвічі, а натиснення правої клавіші - зменшує удвічі. При натиснутій лівій клавіші миші можна виділити пунктирним прямокутником потрібна ділянка графіка - при відпуску клавіші ця ділянка з'явиться в збільшеному вигляді і в тому масштабі, який відповідає розмірам виділяючого прямокутника.

4.1.6 Діаграми, гістограми, вектора

Спеціальні можливості надає MATLAB для побудови різного виду діаграм і гістограм і інших стандартних видів представлення інформації.

- **Стовпчикова діаграма.** Така діаграма будується з використанням функції *bar*.

- При зверненні *bar(Y)* функція будує n груп стовпчикових діаграм по m стовпців в групі (тут $n \times m$ розмір матриці Y). По осі абсцис відкладається номер стовпця, в якому міститься відповідне значення.

- При зверненні *bar(x, Y)* функція по матриці Y будує також будує групи діаграм, але позиції стовпців визначаються вектором x .

- *bar(x, Y, 'stack')* - будує єдину групу діаграм, в якій для кожного рядка з Y будується один стовпець діаграми.

- *bar(..., s)* - при такому зверненні в рядковій константі s задається колір діаграми відповідно до таблиці приведеної при описі функції *plot*.

- **Горизонтальна стовпчикова діаграма.** Ця діаграма будується з використанням функції *barh* складається з горизонтальних стовпців, основні форми звернення до цієї функції такі ж, як і у функції *bar*.

- **Кругові діаграми.** Звернення *pie(x)* дозволяє побудувати кругову діаграму. При зверненні *pie(x, explode)* на круговій діаграмі відділяються окремі сектори. Нумери цих секторів задаються вектором *explode*, довжина якого співпадає з довжиною вектора x . Ненульові компоненти вектора *explode* відзначають сектори відокремлювані від діаграми.

- **Гістограми.** Функція побудови гістограм *hist* має декілька форм звернення.

- $N = hist(Y, M)$ - повертає вектор числа попадань значень з вектора Y в M інтервалів, що рівномірно розділяють інтервал $[min(Y), max(Y)]$. За відсутності аргументу M число інтервалів рівне 10. Якщо Y матриця розміру $n \times m$, то N також є матрицею розміру $M \times m$, i -й стовпець якої є гістограмою i -ої рядка матриці Y .

- $N = hist(Y, x)$ - повертає вектор числа попадань значень з вектора Y в інтервали, середини яких задані компонентами вектора x .

- $[N, X] = hist(Y, M)$ - повертає окрім вектора числа попадань N , вектор X містить середини інтервалів.

- **Кругові гістограми.** Функція побудови кругової гістограми *rose* має форми схожі з функцією *hist*.

- **Ступінчастий графік**, $stairs(x, Y, s)$ - представляє залежність Y від x у вигляді ступінчастої функції. Форми звернення до неї ті ж, що і для функції $plot$.
- **Графік дискретних відліків**. $stem(x, Y, s)$ - представляє дискретні значення залежності Y від x , сполучені вертикальною лінією з віссю абсцис. Форми звернення схожі з функцією $plot$.
- **Графік з інформацією про погрішності**. $errorbar(x, Y, L, U)$ - представляє дискретні значення залежності Y від x з вказівкою довірчих інтервалів значень функції. Верхні і нижні межі цих інтервалів задаються аргументами L і U . При зверненні $errorbar(x, Y, V)$ межі довірчих інтервалів встановлюються $L = Y - V$ і $U = Y + V$.
- **Графік проєкцій векторів на площину**. Для побудови векторів на площині використовується функція $feather$. При зверненні $feather(U, V)$ на координатній площині будуть побудовані вектори, почала яких знаходяться в крапках з нульовими ординатами і абсцисами $X0 = 1 : length(U)$, а закінчення в крапках з координатами $(X0 + U, V)$. Звернення $feather(Z)$ для комплексного вектора Z еквівалентно зверненню $feather(real(Z), imag(Z))$.
- **Графік векторів в полярних координатах**. Для зображення векторів використовується також функція $compass$. $compass(U, V)$ будує на площині вектори, почала яких знаходяться в крапці з нульовими координатами, а закінчення в крапках з координатами (U, V) . Звернення $compass(Z)$ для комплексного вектора Z еквівалентно зверненню $compass(real(Z), imag(Z))$.

Кінцевий опис, даний тут описаним функціям є неповним, і приведений список служить тільки "маршрутним листом" для допитливого читача при знайомстві з відповідними функціями.

4.2 Тривимірна графіка

4.2.1 Побудова кривих в просторі. Перше знайомство з функцією $plot3$.

Просте просторове зображення, яке ми тут розглядатимемо, можна отримати в результаті креслення графіка параметрично заданої тривимірної кривої. Для побудови такої кривої використовується функція $plot3$, звернення до якої аналогічно функції $plot$, розглянутою раніше в розділі 4.1.1

$plot3(x1, y1, z1, s)$.

Це звернення в основному співпадає із зверненням до функції побудови графіка на площині і відрізняється тільки додаванням вектора $z1$. Аргумент - рядок s може приймати значення, вказані в таблиці розділу 4.1.1. Результат дії цієї функції розглянемо на прикладі, результат дії якого приведений на рис. 6.

$x = (1:1000)/1000 * 3 * pi;$
 $plot3(x, sin(10 * x), x.^2);$

Відмітимо тут, що справедливе звернення і до решти всіх форм функції $plot3$, що повторює форми функції $plot$.

Менш вдалий приклад побудови тривимірної кривої приведений на рис. 7. Цю криву будує сценарій:

```
x=(1:1000)/1000*3*pi;  
plot3(sin(25*x), cos(25*x), cos(2*x));
```

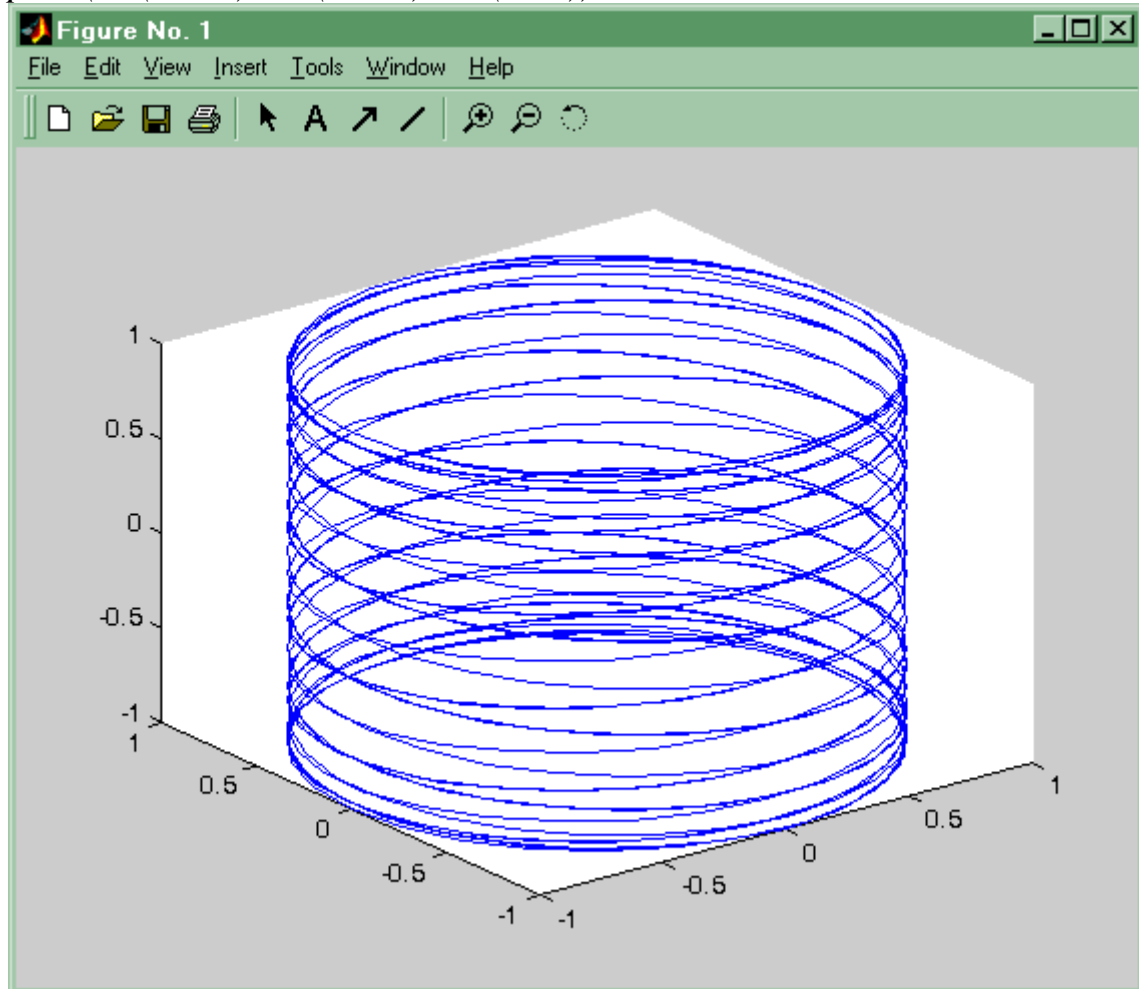


Рис. 6 Результати роботи оператора *plot3*

Для приведеного прикладу зрозуміти, де почало і кінець побудованої кривої не представляється можливим. Незрозуміло також, як будуються намальовані криві, оскільки функція *plot* "будує" зображення в пам'яті комп'ютера, і видає на екран отриманий результат. Як побудова графіка розгортається у часі, можна подивитися, звернувшись до функції *comet3*:

```
comet3(sin(25 * x), cos(25 * x), cos(2 * x));
```

При цьому слід поклопотатися про те, щоб на екрані одночасно були видні і командне і поточне графічне вікна MATLAB'a. Підсумковий результат побудови буде таким же, але уважно відстеживши те, що відбувається на екрані вдається зрозуміти, що побудова приведеної кривої аналогічно намотуванню нитки на котушку і далі по виду аргументів прикинути число витків, число "рухів руки"с ниткою вниз-вгору тощо.

Написи до зображення і конкретних осей для тривимірних зображень виконуються вже відомим способом, описаним в розділі 4.1.3. Для тривимірної графіки додатково введені функції *zgrid* і *zlabel* аналогічні описаним раніше *xgrid*, *ygrid* і *xlabel*, *ylabel*.

4.2.2 Поверхні в просторі

Найпростіший метод побудови поверхні в просторі представляє вже знайома функція *plot3*. Якщо як аргументи цієї функції використовувати матриці однакового розміру, то функція *plot3* буде набір кривих, сукупність яких утворює поверхню. Розглянемо приклад вигляду

```
X=0.1:0.1:3;  
for i=2:30, X(i,:)=X(1,:); end  
Y=X';  
Z=sin(0.3*X.*Y); Z=1./(sin(0.3*X.*Y)+1);  
plot3(X,Y,Z)  
title('Plot3 surface illustration')  
xlabel('Absciss axis')  
ylabel('Ordinate axis')  
zlabel('Z-axis')
```

Для приведеного прикладу нескладно побудувати сітчасту поверхню. Для цього можна скористатися симетрією масивів X і Y і використовувати звернення $plot3(X, Y, Z, 'k', Y, X, Z, 'k')$.

У результаті в графічному вікні з'явиться зображення представлене на рис. 8. У приведеному прикладі автором свідомо приведений приклад не найнаочнішого графіка такого вигляду, хоча в популярних виданнях наводяться вдаліші приклади.

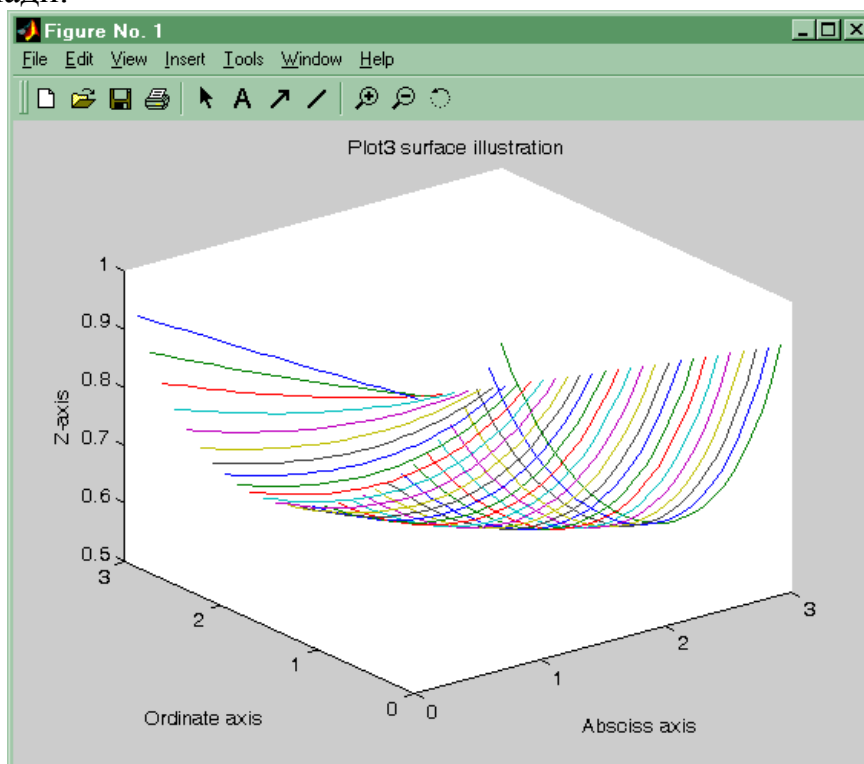


Рис. 7 Незрозумілий результат дії оператора *plot3*

Для зручного графічного представлення тривимірних поверхонь, в MATLAB'і передбачений такий набір спеціальних функцій:

- *mesh(X,Y, Z)* - будує кольорову сітчасту поверхню $Z(X,Y)$, причому

колір вузлів задається висотою поверхні.

- $mesh(X, Y, Z)$ - будує кольорову сітчасту поверхню $Z(X, Y)$ того ж типу, що і $mesh$, а на площині X, Y проводяться лінії рівня поверхні.
- $meshz(X, Y, Z)$ - будує кольорову сітчасту поверхню $Z(X, Y)$, того ж типу, що і $mesh$, а краї поверхні відмічені стовпчасто. Для ілюстрації роботи цієї функції краще використовувати поверхню виду

$$Z1 = \sin(0.3 * X * Y) + 1;$$

- $surf(X, Y, Z)$ - будує кольорову комірчасту поверхню $Z(X, Y)$, причому колір комірок задається висотою поверхні.
- $surf(X, Y, Z)$ - будує кольорову комірчасту поверхню $Z(X, Y)$, того ж типу, що і $surf$, а на площині X, Y проводяться лінії рівня поверхні.
- $surfl(X, Y, Z)$ - будує кольорову комірчасту поверхню $Z(X, Y)$, того ж типу, що і $surf$, але колір поверхні імітує підсвічування від додаткового джерела світла. При зверненні $surfl(X, Y, Z, S)$ - тривимірний вектор $S = [Sx, Sy, Sz]$ задає положення джерела світла в декартовій системі координат, а $S = [Az, El]$ - в системі сферичних координат. Уточнимо, що для декартових координат насправді тут йдеться про паралельний світловий потік, напрям якого задається вектором, направленим з точки S в початок координат.
- $water\ fall(X, Y, Z)$ будує кольорову листкову поверхню $Z(X, Y)$ схожу на $meshz$, проте стиль зображення більш нагадує порізаний на вертикальні шматки пиріг.
- $contour3(X, Y, Z)$ будує кольорову листкову поверхню $Z(X, Y)$ в якій шари відмічені лініями рівного рівня, проведеними в оксанометричній проекції. Таке зображення нагадує класичний листковий пиріг. У обігу $contour3(X, Y, Z, n)$ аргумент n задає кількість рівнів.

Для всіх вищеперелічених функцій допустимі види звернень типу

- $surf(x, y, Z)$, де x і y вектора однакової довжини n , а Z - квадратна матриця розміру $n \times n$. В цьому випадку функція сама будує матриці X і Y .

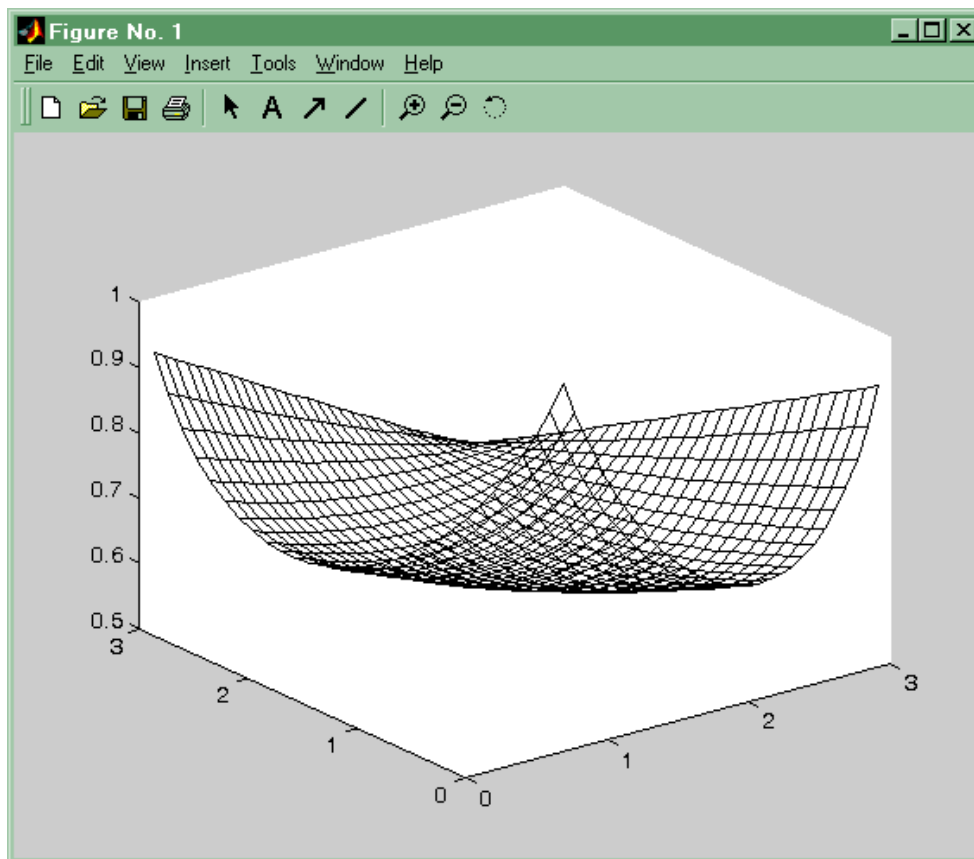


Рис. 8 Результат будування поверхні функцією `plot3`

`surf(Z)` будує поверхню відкладаючи по осях абсцис і ординат номера стовпців і рядків матриці Z .

На закінчення цього підрозділу відмітимо, що побудувати лінії рівня на площині xy для даної поверхні дозволяє функція `contour`, звернення до якої цілком співпадає із зверненням до функції `contour3`. Обидві ці функції повертають матрицю Cs опису контурних ліній. Маркірувати ці лінії для результуючих графіків обох функцій дозволяє функція `clabel` в результаті звернення `clabel(Cs)`, де матрицю Cs обчислюють і повертають функції `contour` і `contour3`. В результаті звернення `clabel(Cs, v)` маркіруються тільки висоти, вказані у векторі v . Функції побудови ліній рівня можуть також при зверненні `[Cs, H] = contour(X, Y, Z)` повертати вектор, що містить додаткову інформацію про висоти. При зверненнях `clabel(Cs, H)` і `clabel(Cs, H, v)` написи ліній рівня без додаткових маркерів будуть вставлені в розриви контурних ліній.

Представити характер поверхні допомагає також функція `quiver`. В результаті звернення `quiver(X, Y, U, V)` в графічному вікні будується зображення градієнтного поля. В точках заданих координатами з матриць X і Y зображаються вектора, величини і напрями яких задаються матрицями U і V . Часто буває зручно використовувати цю функцію для побудови градієнтного поля поверхні, заданою матрицею Z таким чином:

```
[px,py] = gradient(Z, hx, hy);
quiver(X, Y, px, py)
```

У приведеному раніше прикладі значення кроків hx і hy по осях абсцис і ординат слід вказати рівними 0.1.

Нарешті відмітимо, що для побудови матриць X і Y по векторах x і y можна використовувати спеціальну функцію, звернення до якої має вигляд
 $[X, Y] = meshgrid(x, y)$.

4.2.3 Вигляд зображення з різних боків

Після побудови тривимірної поверхні у користувача виникає природне бажання розглянути її з іншого боку. Вибрати точку огляду, відмінну від тієї, яку надає стандарт оксанометричної проєкції, дозволяє функція *view*. Звернення до неї можуть мати вигляд

• *view(Cv)*, де Cv - вектор, задаючий напрям перегляду.

Для тривимірного вектора $Cv = [Cx, Cy, Cz]$ - Cv - декартові координати "точки перегляду". Лінія візування є променем, проведеним з початку координат і Cv , що проходить через точку.

Для двовимірного вектора $Cv = [Az, El]$ - величини Az і El задають лінію візування, використовуючи кути азимута і піднесення. Так само буде сприйнято і звернення *view(Az, El)*.

Зображення, виведене на екран є проєкцією на площину ортогональну лінії візування, проведenu поза межами об'єкту.

- *view(2)* встановлює штатне положення "точки перегляду" для двовимірної графіки ($Az = 0^\circ$, $El = 90^\circ$).
- *view(3)* встановлює штатне положення "точки перегляду" для тривимірної графіки (оксанометрична проєкція – $Az = -37.5^\circ$, $El = 30^\circ$).
- $[Az, El] = view$ - повертає поточне значення кутів азимута і піднесення.
- $T = view$ повертає поточне значення узагальненої матриці T перетворень, використовуюваною для отримання зображення. Цю матрицю можна отримати також звернувшись до функції.

$T = viewmtx(Az, El)$. Звернення *view(T)* встановлює положення "точки перегляду", що задається матрицею T .

Як приклад розглянемо поверхню

```
x=(1:100)/100*3*pi-pi;  
y=(1:100)/100-0.5;  
[X, Y]=meshgrid(x,y);  
Z=sin(X.*Y);  
mesh(X, Y, Z)
```

Звернення *view(2)* дає уявлення про вигляд зверху на зображену поверхню, *view(30, 30)* дозволяє поглянути на неї "збоку", а *view(45 -60)* - знизу. Забавно, що в останньому випадку MATLAB буде координатні осі для $Az = 60$.

Розібратися в ситуації з картинкою, окрім вашої просторової уяви дозволяє функція *colorbar*, яка додає до поточного графіка шкалу палітри. При зверненні

до цієї функції без аргументів або зверненні *colorbar* ('vert') ця шкала вертикальна. При зверненні *colorbar*('horiz') - вона горизонтальна.

Шкала палітри зображена в графічному вікні є окремим графіком, намальованим в "подвікні", що описане в підрозділі 5. Це "підвікно" залишається активним після закінчення виконання функції *colorbar*, що необхідно враховувати при подальшому кресленні зображень в графічному вікні.

4.2.4 Друк, зберігання і експорт зображень

Надрукувати зображення з графічного вікна на принтері, приєднаному до комп'ютера, на якому працює користувач, нескладно. Для цього достатньо в меню графічного вікна вибрати позицію "File", а в тому, що відкрився підменю вибрати позицію "Print". Подальші дії також відповідають стандартним діям в системі Windows.

Перенести зображення на комп'ютер з тією ж версією MATLAB'a вдається, зберігши його. Для цього необхідно в меню графічного вікна вибрати позицію "File", а в тому, що відкрився підменю позиції "Save"или "Save as". Різні версії MATLAB'a пропонують різні способи збереження зображень: версії, раніші ніж MATLAB 5.2 використовують для збереження *.m* і *.mat* файли, версія MATLAB 5.3 - *.m* і *.fig* файли. При відкритті цих *.fig* файлів з використанням позиції підменю "Open" зображення з'являється в поточному графічному вікні. Для відновлення зображення з *.m* і *.mat* слід виконати відповідний сценарій з *.m* файлу. Проте не всі версії MATLAB'у нормально читають чужі файли.

Проблеми виникають у випадку, якщо Вам необхідно включити зображення в звіт. При наборі звіту в редакторі MS Word можна скористатися стандартним шаблоном M-book, що поставляється спільно з системою MATLAB. Для використання інших редакторів слід експортувати зображення використовуючи Clipboard системи Windows. Для цього необхідно в меню графічного вікна вибрати позицію "Edit", а в тому, що відкрився підменю позицію "Copy Figure". Залежно від установок проводиться копіювання зображення в одному з форматів "Windows Metafile" або "Windows Bitmap". Ці зображення можна відновити в будь-якому графічному редакторі з використанням пункту підменю "Paste"і/або комбінації клавіш Ctrl-v. Для формату "Windows Metafile" при відновленні зображення в редакторах Paint, Paintbrush, Adobe Photoshop і подібних до них спостерігається спотворення шрифтів в написах. При використанні формату "Windows Bitmap" шрифти передаються без спотворень. Для установки відповідного формату необхідно в меню графічного вікна вибрати позицію "File", а в тому, що відкрився підменю позицію "Preferences...". У вікні, що знов відкрилося, слідує вибрати сторінку із закладкою "Copying Options" в розділі "Clipboard Format" встановити вибраний Вами формат.

5 Числовий розв'язок задачі Коші для звичайних диференціальних рівнянь

У пакеті MATLAB існує декілька можливостей розв'язку задачі Коші для звичайних диференціальних рівнянь: стандартні ODE-вирішувачі, спеціалізовані програми дослідження моделей керованих систем, засоби імітації

аналогового моделювання середовища SIMULINK. Виклад засобів середовища SIMULINK може бути предметом окремого обговорення (див. наприклад [10], [5]). Дві перші можливості розглянемо тут докладніше.

5.1 Стандартні ODE-вирішувачі.

Стандартні ODE-вирішувачі є пакетом програм, що складається з М-функцій: *ode45*, *ode23*, *ode113*, *ode15s*, *ode23s*, *ode23t*, *ode23tb*, які надалі позначатимемо через *solver*

Вказані функції реалізують наступні методи рішення систем звичайних диференціальних рівнянь

- *ode45* - однокрокові явні методи Рунге-Кутта 4 і 5 порядку. Цей класичний метод, рекомендується для початкової проби розв'язку, зазвичай дає задовільні результати.
- *ode23* - однокрокові явні методи Рунге-Кутта 2 і 3 порядку. При помірних вимогах до точності рішення для нежорстких систем метод може дати вигравш в швидкості розв'язку.
- *ode113*- багатокроковий метод Адамса-Башворта-Мултона змінного порядку. Цей адаптивний метод покликаний забезпечити високу точність розв'язку.
- *ode15s* - багатокроковий метод змінного порядку (від 1 до 5). Цей адаптивний метод покликаний забезпечити високу точність розв'язку для жорстких систем.
- *ode23s* - однокроковий модифікований метод Розенброка 2-го порядку. Покликаний забезпечити високу швидкість обчислень для жорстких систем при низькій точності.
- *ode23t*- метод трапецій з інтерполяцією. Метод дає хорошу точність при рішенні жорстких задач, що описують осцилятори з майже періодичним вихідним сигналом.
- *ode23tb* - неявний метод Рунге-Кутта в початковій стадії розв'язку і метод, що використовує формули зворотного диференціювання 2-го порядку в подальшому. При низькій точності для жорстких систем цей метод може опинитися ефективніше, ніж *ode23s*.

Всі вказані функції призначені для числового розв'язку задачі Коші для систем звичайних диференціальних рівнянь вигляду.

$$\begin{cases} \frac{dy}{dt} = f(y,t) \\ y(0) = y_0 \end{cases} \quad (4)$$

де y - вектор довжини n .

Вирішувачі *ode15s*, *ode23s*, *ode23t*, *ode23tb* призначені також для вирішення завдання Коші для систем звичайних диференціальних рівнянь неявного вигляду

$$\begin{cases} M(t) \frac{dy}{dt} = f(y,t) \\ y(0) = y_0 \end{cases} \quad (5)$$

де $M(t)$ - матриця масових коефіцієнтів. Для функції *ode23s* матриця M не повинна залежати від аргументу t .

Звернення до функцій числового інтегрування може приймати наступні форми:

$$\begin{aligned} [T, Y] &= \text{solver}('Fun', \text{tspan}, y0) \\ [T, Y] &= \text{solver}('Fun', \text{tspan}, y0, \text{options}) \\ [T, Y] &= \text{solver}(Fun, \text{tspan}, y0, \text{options}, p1, p2 \dots) \\ [T, Y, TE, YE, IE] &= \text{solver}('Fun', \text{tspan}, y0, \text{options}, p1, p2 \dots) \end{aligned}$$

• *Fun* - ім'я файлу, в якому поміщена M -функція обчислення правих частин системи диференціальних рівнянь. Опис цієї функції приймає одну з наступних форм

$$\text{function } f = \text{Fun}(t, y) \quad (6)$$

$$\text{function } f = \text{Fun}(t, y, \text{flag}) \quad (7)$$

$$\text{function } f = \text{Fun}(t, y, \text{flag}, p1, p2 \dots) \quad (8)$$

- *tspan* - вектор значень моментів часу. Якщо вектор *tspan* складається з двох компонент $[t0, tfin]$, то ці значення сприймаються як початкове і фінальне значення аргументу. Для вектора більшої довжини $[t0, t1 \dots, tfin]$, компоненти якого розташовані в порядку зростання або спадання, результати числового інтегрування видаються для вказаних значень аргументу.
- $y0$ - вектор початкових значень.
- *options* - аргумент задаючий параметри для числового інтегрування. Цей аргумент задається функцією *odeset*, використання якої обговорюватиметься нижче.
- $p1, p2 \dots$ - довільні параметри, що передаються у функцію *Fun*.
- T - вектор значень аргументу, для якого видаються значення функції y .
- Y - матриця рішень, кожен рядок якої відповідає значенню аргументу, поверненому в Y .
- TE, YE, IE - вихідні матриці, в яких виводяться значення аргументу t , функції y і номери події ('event') для роботи опції фіксації подій.

Простий приклад розв'язку задачі Коши

Перш ніж розбирати далі різні можливості рішення задач числового інтегрування систем звичайних диференціальних рівнянь, приведемо простий приклад, який може послужити зразком для вирішення 90% завдань такого типу. Шукатимемо розв'язок рівняння Ван дер Поля

$$x'' = -x + 0.3 * x' * (1 - x^2)$$

за початкових умов

$$x(0) = 1; x'(0) = 0$$

Зробимо заміну змінних $y_1 = x$ і $y_2 = x'$. Перетворимо завдання до вигляду

$$\begin{cases} y_1' = y_2 \\ y_2' = -y_1 + 0.3 * y_2 * (1 - y_1^2) \\ y_1(0) = 1 \\ y_2(0) = 0 \end{cases} \quad (9)$$

Для цієї системи створимо функцію обчислення правих частин з описом типу 9, помістивши цю функцію у файл *vanderp.m* у робочій папці

```
function f=vanderp(t,x)
%function f=vanderp(t,x)
% right part for VanDerPol equation calculation
% dx1/dt=x2
% dx2/dt=-x1+e*x2*(1-x1^2)
f(1,1)=x(2);
f(2,1)=-x(1)+0.3*x(2)*(1-x(1)^2);
```

Результат отримаємо набравши в командному вікні MATLAB'a команду:

```
ode45('vanderp',[0,30],[1,0])
```

В результаті такого звернення MATLAB відкриє графічне вікно і виводитиме в нього залежність $x(t)$ у міру числового інтегрування. Результат приведений на рис. 9. Графіки різних компонент вектора у виводяться різними кольорами. Маркерами відмічені точки, в яких обчислені значення.

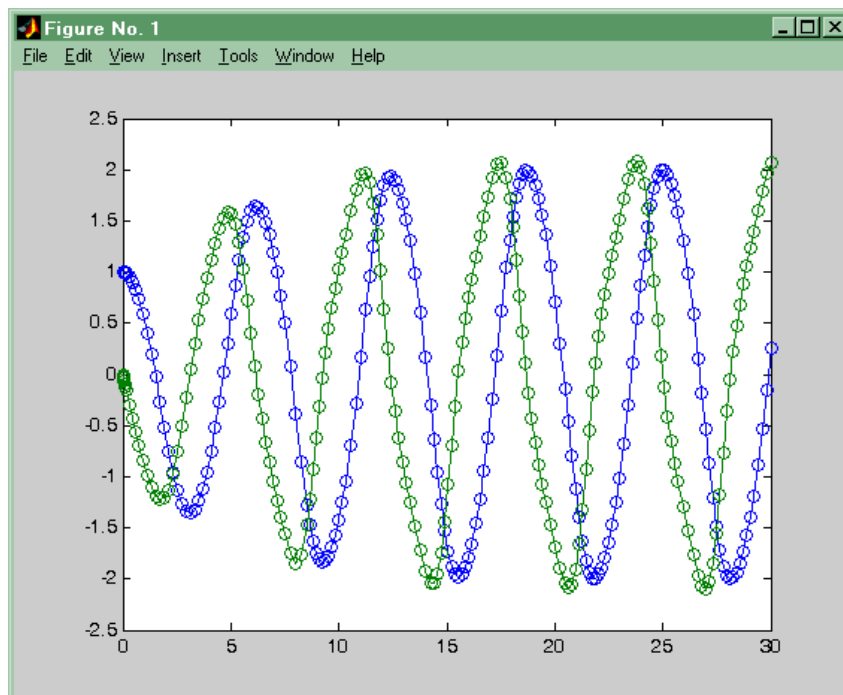


Рис. 9 Результат дії функції ode45

Щоб отримати числові значення розв'язку, використовуємо звернення:

```
[T, Y]=ode45('vanderp',[0,30],[1,0]);
```

В цьому випадку зображення на екрані не відображається, але вихідні масиви T , Y заповнені. З їх використанням можна отримати, наприклад, фазовий портрет для знайденого вирішення. Для цього побудуємо залежність y_2 від y_1 за допомогою команди

```
plot(Y(:, 1),Y(:,2))
```

Результат побудови приведений на рис. 10.

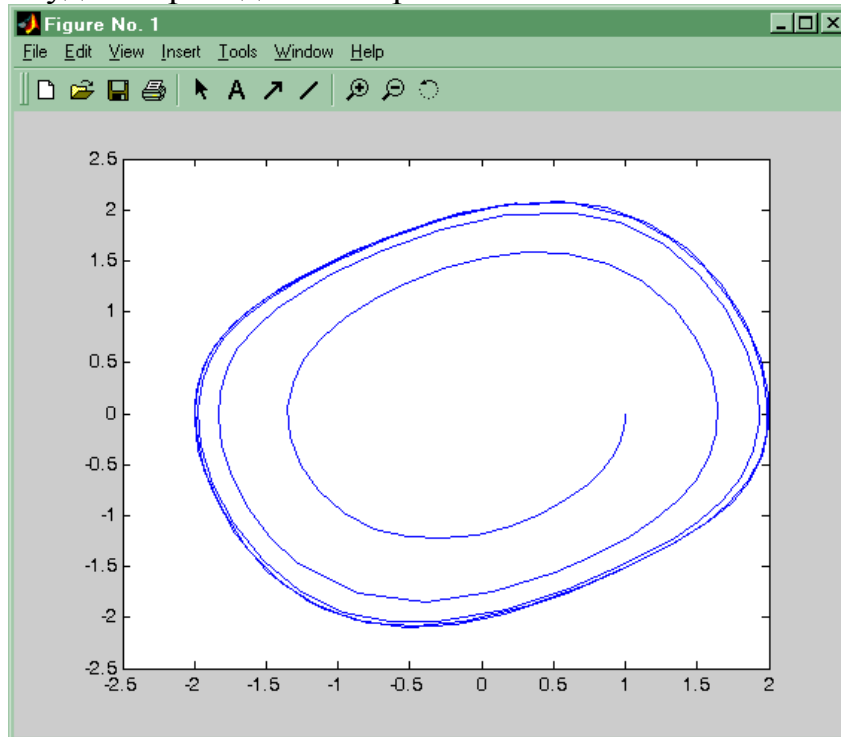


Рис. 10 Фазовий портрет розв'язків рівняння Ван дер Поля.

Читачеві, що цікавиться, раджу також подивитися залежність довжини кроку інтеграції від номера кроку. Це нескладно зробити, виконавши команду $plot(diff(T))$. Отриманий графік наочно продемонструє змінність кроку в ході числового інтегрування. Подібний автоматизм у виборі кроку може надати погану послугу при інтеграції систем з розривною правою частиною. Автоматичний вибір кроку при частому "переключенні" приводить "точні" методи до непомірного збільшення часу обчислень.

5.2 Опції вирішувача

У цьому параграфі ми обговоримо деякі можливості використання аргументу *options*. За допомогою цього масиву встановлюються різні спеціальні параметри *ode-вирішувачів*. Перерахуємо ті з них, які представляються найбільш важливими:

1. *RelTol*- відносна погрішність. Скалярна величина за умовчанням приймається рівною 10^{-3} . Помилка на кожному кроці інтеграції оцінюється величиною $ei < \max(RelTol * \text{abs}(y_i), AbsTol_i)$.

2. *AbsTol* - вектор абсолютних погрішностей (за умовчанням всі його компоненти рівні 10^{-6}).

3. *Refine* - позитивне ціле число - чинник якості, що підвищує кількість значень аргументу, що виводяться. Використовується для згладжування висновку. За замовчуванням вирішувачі окрім *ode45* використовують значення 1, а *ode45* - 4. Опція *Refine* не використовується, якщо довжина вектора *tspan* більше 2.
4. *OutputFcn* - текстова константа '*AFun*', що містить ім'я додаткової функції виведення. Вирішувач викликатиме вказану функцію після закінчення кожного кроку інтегрування. Зазвичай за умовчанням додаткова функція виведення відсутня, але за відсутності вихідних параметрів вона приймає значення '*odeplot*'. Її роботу ми і бачили в першому з простих прикладів, що наводилися. Для того, щоб на екран комп'ютера виводився графік за наявності вихідних параметрів необхідно встановити значення опції *OutputFcn* рівне '*odeplot*'. При необхідності можна використовувати також функції:
 - *odephas2* - зображає двовимірний фазовий портрет;
 - *odephas3* - зображає тривимірний фазовий портрет;
 - *odeprint* - виводить в командне вікно проміжні результати числового інтегрування.

Користувач може самостійно написати функцію такого типу. Єдиною вимогою до такої функції є те, що її опис повинен мати вигляд:

```
function AFun(t, z)
```

де z - вектор у змінних стану вирішуваної системи рівнянь або вектор меншої довжини, який складається з компонент вектора u у порядок яких задається опцією *OutputFcn*.

5. *OutputSel* - вектор номерів компонент вектора стану u , використовуваних вирішувачем при зверненні до функції додаткового виведення *OutputFcn*. За замовчуванням використовуються всі компоненти вектора стану.
6. *Stats* - відображає обчислювальну статистику після закінчення роботи вирішувача (приймає значення *on/off*).
7. *Jacobian* - ознака можливості обчислення Якобіана (*on/off*). Для використання цієї опції необхідно, щоб звернення до функції *Fun* обчислення правих частин системи рівнянь мало вигляд (7). При цьому при значенні змінної *flag* рівному '*Jacobian*' вказана програма повинна обчислювати матрицю Якобі, компоненти якої мають вигляд $\partial f_i / \partial x_j$. Матриця Якобі є такою, що повертається параметром функції *Fun*. Використання цієї опції може значно понизити число кроків і час числового інтегрування.
8. *JConstant* - ця опція повинна приймати значення '*on*' у випадку, якщо матриця Якобі для системи диференціальних рівнянь є постійною.
9. *Events* - ознака використання обробника подій *on/off*. Для використання цієї опції необхідно, щоб звернення до функції *Fun* обчислення правих частин системи рівнянь мало вигляд (7). При значенні змінної *flag* рівному '*Events*'

вказана програма повинна повертати наступні величини

- *value* - вектор величин, для яких повинно бути зафіксований момент зміни знаку
- *isterminal* - логічний вектор, компоненти якого приймають значення 0 або 1 залежно від того, чи слід зупинити числове інтегрування при зміні знаку відповідною компоненти вектора *value*.
- *direction* - вектор тієї ж довжини, вказуючий напрям зміни знаку, при якому відповідна подія буде зафіксована. Компоненти цього вектора приймають значення 1 і -1 залежно від напрямку зміни знаку. Значення відповідною компоненти рівне 0 означає, що напрям зміни знаку неістотний.

10. *Mass* - ознака використання обчислюваної матриці $M(t)$ (інерційних коефіцієнтів при похідних в лівій частині рівнянь) [*on*/*off*]. Для використання цієї опції необхідно, щоб звернення до функції *Fun* обчислення правих частин системи рівнянь мало вигляд (7). При значенні змінної *flag* рівному 'Mass' вказана програма повинна повертати матрицю інерційних коефіцієнтів - "мас". Цей параметр обробляють функції *ode15s*, *ode23s*, *ode23t*, *ode23tb*. (Відмітимо, що змінну матрицю мас обробляє тільки функція *ode15s*).

11. *MassConstant* -признак використання постійної матриці "мас"(інерційних коефіцієнтів при похідних в лівій частині рівнянь) [*on*/*off*]. Цей параметр повинен приймати значення 'on', якщо при зверненні до функції *Fun* із змінній *flag* рівною 'Mass'. Це значення інформує про те, що функція *Fun* повертає постійну матрицю інерційних коефіцієнтів ("мас").

12. *MaxStep* - позитивний скаляр - максимальне значення кроку інтеграції (за умовчанням приймає значення десятої частини інтервалу часу *tspan*).

13. *Initial Step* - позитивний скаляр - початкове значення кроку інтеграції. За умовчанням програми визначають початковий розмір кроку автоматично.

14. *MaxOrder* - максимальний порядок для інтегруючої функції *ode15s* і приймає одне із значень [1 | 2 | 3 | 4 | 5].

5.3 Програми числового інтегрування лінійних моделей керованих систем

Одним з основних достоїнств пакету MATLAB є наявність широкого набору бібліотек - TOOLBOX'ів - призначених для розв'язку спеціальних типів математичних і інженерних завдань. У поточному параграфі розглянемо використання декількох функцій бібліотеки "Control system toolbox" для розв'язку систем лінійних диференціальних рівнянь. Детальніше з функціями з "Control system toolbox" ознайомитися в [11].

Модель лінійної динамічної керованої системи запишемо у вигляді системи диференціальних рівнянь 1-го порядку

$$\begin{aligned}\frac{dx}{dt} &= Ax + Bu \\ y &= Cx + Du\end{aligned}\tag{10}$$

де x - вектор стану системи, u - вектор управління, y - вектор виходів, A, B, C, D - матриці з постійними коефіцієнтами.

У бібліотеці "Control" представлені наступні функції, що дозволяють вирішувати задачі інтегрування:

- *initial* - функція рішення задачі Коші для заданих початкових умов. Звернення до цієї функції може мати вигляд:

$$[Y, T, X] = \text{initial}(A, B, C, D, x_0, Tf)$$

Тут A, B, C, D - матриці, що описують систему вигляду (10); x_0 - вектор початкових умов, Tf - значення моменту часу, до якого здійснюється інтегрування. Вихідними параметрами є матриці Y і X , в яких поміщаються порядково значення векторів y і x для моментів часу з відповідного вектора T .

- *impulse* - обчислення вагової функції системи (10). Функція розв'язку задачі про поведінку системи при імпульсному збуренні входу з номером i сигналом типу дельта-функції.

$$u_i(t) = \delta(t) = \begin{cases} 1 & \text{при } t=0 \\ 0 & \text{при } t \neq 0 \end{cases}$$

при нульових початкових умовах.

Звернення до цієї функції може мати вигляд:

$$[Y, T, X] = \text{impulse}(A, B, C, D, i, Tf)$$

Тут A, B, C, D - матриці, що описують систему вигляду (10); i - номер збурюваного входу, Tf - значення моменту часу, до якого здійснюється інтегрування. Вихідними параметрами є матриці Y і X , в яких поміщаються відрядкові значення векторів y і x для моментів часу з відповідного вектора T .

- *step* - обчислення перехідної функції системи (10). Функція розв'язку задачі про поведінку системи при збуренні входу з номером i сигналом у вигляді одиничної сходинки:

$$u_i(t) = \delta(t) = \begin{cases} 1 & \text{при } t < 0 \\ 0 & \text{при } t \geq 0 \end{cases}$$

Звернення до цієї функції може мати вигляд:

$$[Y, T, X] = \text{step}(A, B, C, D, i, Tf)$$

Аргументи і вихідні параметри функції *step* мають той же сенс, що і для функції *impulse*.

- *lsim* - обчислює рішення системи для випадку довільного вхідного сигналу, заданого масивом своїх відліків - матрицею *U*. Звернення до цієї функції може мати вигляд:

$$[Y, T, X] = \text{lsim}(A, B, C, D, U, T, x_0)$$

Тут *A, B, C, D* - матриці, що описують систему вигляду (10); *U* і *T* - матриця значень векторної функції *u* і відповідних ним моментів часу. Вихідними параметрами є матриці *Y* і *X*, в яких поміщаються відрядкові значення векторів *y* і *x* для моментів часу з відповідного вектора *T*. Матриця *X* і вектор *T* з числа вихідних параметрів може бути опущений. Якщо аргументом *x0* відсутній, програма проводить обчислення з нульовими початковими умовами.

У всіх випадках матриця *X* з числа вихідних параметрів може бути опущена. Якщо аргументом *Tf* є вектор, програма сприймає його як вектор моментів часу, в які виводиться рішення.

Ефективність використання вказаних процедур пов'язана з тим, що вони здійснюються не за допомогою чисельної інтеграції, а в результаті рішення задачі про власні числа і власні вектори матриці *A* і подальшого використання аналітичних залежностей для розв'язку вказаної системи.

Індивідуальні завдання

Завдання 1. Розрахувати заданий інтеграл з точністю

Варіант	Інтеграл	Точність
1.	$\int_1^3 \frac{x^2}{2x+3} dx$	0,001
2.	$\int_1^4 x^2 \sqrt{x+2} dx$	0,1
3.	$\int_{0.2}^1 \frac{x}{\sin^2 3x} dx$	0,01
4.	$\int_2^3 x \cdot e^{0.8x} dx$	0,1
5.	$\int_1^2 \frac{1}{x\sqrt{x^2+0.25}} dx$	0,01
6.	$\int_1^2 \frac{x^2}{(2x+0.3)^2} dx$	0,001
7.	$\int_3^5 \frac{x}{0.5x+0.1} dx$	0,01
8.	$\int_0^1 x^2 \cdot \sin(x) dx$	0,001
9.	$\int_1^4 x \cdot 2^{3x} dx$	0,1

10.	$\int_0^2 \frac{x}{(x+3)^2} dx$	0,001
11.	$\int_{0.2}^1 \frac{\sqrt{4-x^2}}{x} dx$	0,01
12.	$\int_0^{\pi/4} x \cdot \sin(x) dx$	0,001
13.	$\int_0^1 2^{3x} dx$	0,01
14.	$\int_1^5 \frac{\ln^2 x}{x} dx$	0,01
15.	$\int_1^3 \frac{x^2}{2x+3} dx$	0,001
16.	$\int_1^4 x^2 \sqrt{x+2} dx$	0,1
17.	$\int_{0.2}^1 \frac{x}{\sin^2 3x} dx$	0,01
18.	$\int_2^3 x \cdot e^{0.8x} dx$	0,1
19.	$\int_1^2 \frac{1}{x\sqrt{x^2+0.25}} dx$	0,01
20.	$\int_1^2 \frac{x^2}{(2x+0.3)^2} dx$	0,01
21.	$\int_1^5 \frac{\ln^2 x}{x} dx$	0,01

Завдання 2

Розв'язок систем алгебраїчних рівнянь

Завдання

Вар 1 - 15

Вар	Система рівнянь	Початкові значення
1	$\left. \begin{aligned} 3x^2 + xy - 1 &= 0 \\ x - 3y + 5 &= 0 \end{aligned} \right\}$	$x_0=1$ $y_0=-2$
2	$\left. \begin{aligned} xy - 2y + y^2 &= 0 \\ x + xy &= 0 \end{aligned} \right\}$	$x_0=0$ $y_0=0.2$
3	$\left. \begin{aligned} xy^2 + y - 7 &= 0 \\ x^2 + y^2 - 12 &= 0 \end{aligned} \right\}$	$x_0=4.2$ $y_0=-1$
4	$\left. \begin{aligned} 7x + 9y - 15,6 &= 0 \\ 3,5x^2 + 20,8y - 30 &= 0 \end{aligned} \right\}$	$x_0=2.1$ $y_0=-3$
5	$\left. \begin{aligned} 12x - xy - 8 &= 0 \\ xy - 3x - 7 &= 0 \end{aligned} \right\}$	$x_0=6$ $y_0=2.5$
6	$\left. \begin{aligned} 3x^2 + xy - 1 &= 0 \\ x - 3y + 5 &= 0 \end{aligned} \right\}$	$x_0=1$ $y_0=-2$
7	$\left. \begin{aligned} xy - 2y + y^2 &= 0 \\ x + xy &= 0 \end{aligned} \right\}$	$x_0=0$ $y_0=0.2$
8	$\left. \begin{aligned} xy^2 + y - 7 &= 0 \\ x^2 + y^2 - 12 &= 0 \end{aligned} \right\}$	$x_0=4.2$ $y_0=-1$
9	$\left. \begin{aligned} 7x + 9y - 15,6 &= 0 \\ 3,5x^2 + 20,8y - 30 &= 0 \end{aligned} \right\}$	$x_0=2.1$ $y_0=-3$
10	$\left. \begin{aligned} 12x - xy - 8 &= 0 \\ xy - 3x - 7 &= 0 \end{aligned} \right\}$	$x_0=6$ $y_0=2.5$
11	$\left. \begin{aligned} 3x^2 + xy - 1 &= 0 \\ x - 3y + 5 &= 0 \end{aligned} \right\}$	$x_0=1$ $y_0=-2$
12	$\left. \begin{aligned} xy - 2y + y^2 &= 0 \\ x + xy &= 0 \end{aligned} \right\}$	$x_0=0$ $y_0=0.2$
13	$\left. \begin{aligned} xy^2 + y - 7 &= 0 \\ x^2 + y^2 - 12 &= 0 \end{aligned} \right\}$	$x_0=4.2$ $y_0=-1$
14	$\left. \begin{aligned} 7x + 9y - 15,6 &= 0 \\ 3,5x^2 + 20,8y - 30 &= 0 \end{aligned} \right\}$	$x_0=2.1$ $y_0=-3$

15	$\left. \begin{aligned} 12x - xy - 8 &= 0 \\ xy - 3x - 7 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 6 \\ y_0 &= 2.5 \end{aligned}$
16	$\left. \begin{aligned} 12x - xy - 8 &= 0 \\ xy - 3x - 7 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 6 \\ y_0 &= 2.5 \end{aligned}$
17	$\left. \begin{aligned} 3x^2 + xy - 1 &= 0 \\ x - 3y + 5 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 1 \\ y_0 &= -2 \end{aligned}$
18	$\left. \begin{aligned} xy - 2y + y^2 &= 0 \\ x + xy &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 0 \\ y_0 &= 0.2 \end{aligned}$
19	$\left. \begin{aligned} xy^2 + y - 7 &= 0 \\ x^2 + y^2 - 12 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 4.2 \\ y_0 &= -1 \end{aligned}$
20	$\left. \begin{aligned} 7x + 9y - 15,6 &= 0 \\ 3,5x^2 + 20,8y - 30 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 2.1 \\ y_0 &= -3 \end{aligned}$
21	$\left. \begin{aligned} 12x - xy - 8 &= 0 \\ xy - 3x - 7 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 6 \\ y_0 &= 2.5 \end{aligned}$
22	$\left. \begin{aligned} 3x^2 + xy - 1 &= 0 \\ x - 3y + 5 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 1 \\ y_0 &= -2 \end{aligned}$
23	$\left. \begin{aligned} xy - 2y + y^2 &= 0 \\ x + xy &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 0 \\ y_0 &= 0.2 \end{aligned}$
24	$\left. \begin{aligned} xy^2 + y - 7 &= 0 \\ x^2 + y^2 - 12 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 4.2 \\ y_0 &= -1 \end{aligned}$
25	$\left. \begin{aligned} 7x + 9y - 15,6 &= 0 \\ 3,5x^2 + 20,8y - 30 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 2.1 \\ y_0 &= -3 \end{aligned}$
26	$\left. \begin{aligned} 12x - xy - 8 &= 0 \\ xy - 3x - 7 &= 0 \end{aligned} \right\}$	$\begin{aligned} x_0 &= 6 \\ y_0 &= 2.5 \end{aligned}$

Завдання 3 – точкова апроксимація функції за методом найменших квадратів

Вар	Значення x						Значення y						
	№	1	2	3	4	5	6	1	2	3	4	5	6
1	1,45	1,36	1,14	-	-	-	-	3,14	4,15	5,65	-	-	-
2	0	1	2	5	-	-	2	3	12	147	-	-	-
3	0	1,5	3,4	6,8	-	-	1,45	3,14	4,65	4,11	-	-	-
4	11	13	14	18	19	21	1342	2210	2758	5850	6878	9282	-
5	1,3	1,75	1,83	1,9	-	-	3,2	4,32	4,89	5,6	-	-	-
6	2,5	3,7	4,8	4,9	5,1	-	2,9	3,8	4,5	5,8	6,3	-	-
7	-2	1	2	4	-	-	25	-8	-15	-23	-	-	-
8	-2	-1	0	1	2	-	6	0	2	0	6	-	-
9	15	1,54	1,56	1,60	1,63	1,70	3,873	3,924	3,950	4,000	4,037	4,123	4,123
10	0	1,5	2	2,5	3	-	1,54	3,41	4,56	4,21	4,11	-	-
11	10	11	12	15	17	19	1324	2310	2658	3120	5970	7320	-
12	1,52	1,56	1,58	1,62	1,65	1,72	3,973	3,993	4,000	4,037	4,123	4,224	-
13	2	2,3	2,5	3	3,5	3,8	5,848	6,127	6,3	6,694	7,047	4,243	-
14	2,2	2,4	2,7	3,2	3,7	4	5,948	6,259	4,400	6,749	7	7,368	-
15	2,22	2,41	2,78	3,34	3,75	4,21	6,123	7,223	7,373	7,989	8,123	9,324	-
16	14	17	31	35	-	-	68,7	64	44	39,1	-	-	-
17	0	1	2	4	10	12	2	3	12	147	156	193	-
18	-1	0	3	10	12	19	33	29	10	-9	-10	-21	-
19	0,9	1,1	1,9	2,2	-	-	3,1	3,3	3,4	4,2	-	-	-
20	13	15	23	29	-	-	73,7	68,7	52,3	44	-	-	-

Перелік посилань

1. <http://ablbook.com/index.php?newsid=5507> Макаров Е.Г., Самоучитель MathCad 14 від 18.12.2015 р.
2. Е. Любимов. / Mathcad. Теория и практика проведения электротехнических расчетов в среде Mathcad и Multisim. / Москва, 2014. - 400 с.: ил.
3. Кирьянов Д.В. / Mathcad 15/Mathcad Prime 1.0. - СПб.: БХВ-Петербург, 2012. - 432 с.: ил.
4. 3. Крушель Е.Г., Алейникова Е.А. Методические указания к лабораторной работе по дисциплине «Методы оптимизации». Использование градиентных методов оптимизации для решения задач оценки параметров математических моделей. Часть 1. / ВолгГТУ, Волгоград, 2010. – 42 с.: ил.
5. Очков В. Ф. О-94 Mathcad 14 для студентов, инженеров и конструкторов. — СПб.: БХВ-Петербург, 2007. — 368 с.: ил.
6. Дьяконов В. Mathcad 2001: учеб. курс. - СПб.: Питер, 2001 - 624с.
7. Кудрявцев Е.М. MathCAD 2000 Pro. – М.: ДМК „Пресс“, 2001. – 576с.
8. Кирьянов Д.В. Самоучитель MathCAD 2001 – СПб: БХВ-Петербург, 2001.
9. Потемкин В.Г. Система MATLAB.// М.: Изд. "Диалог-МИФИ", 1997.
10. Потемкин В.Г. MATLAB 5 для студентов.// М.: Изд. "Диалог-МИФИ", 1998.
11. Потемкин В.Г. Система инженерных и научных расчетов MATLAB 5.x.// в 2-х томах. М.: Изд. "Диалог-МИФИ", 1999.
12. Дьяконов В.П. Справочник по применению системы PC MATLAB.// М.: Физматлит, 1993.
13. Дьяконов В.П., Абраменкова И.В. MATLAB 5.0/5.3. Система символьной математики. // М.: Нолидж, 1999.
14. Мартынов Н.Н., Иванов А.П. MATLAB 5.X. Вычисления, визуализация, программирование.-М.: КУДИЦ-ОБРАЗ, 2000.
15. Конев В.Ю., Мироновский Л.А. Основные функции пакета MATLAB.// СПб, Изд. СПГААП, 1992.
16. Коробова Н.Л., Загашвили Ю.В. Комплекс автоматизированного проектирования MATLAB-CTRL.// СПб, Изд. СПГААП, 1993.

17. Андрушевский Б.Р., Фрадков А.Л. Элементы математического моделирования в программных средах MATLAB и Scilab.// СПб, изд. Наука, 2001.
18. Гультяев А. MATLAB 5.2. Имитационное моделирование в среде Windows. // СПб.: КОРОНАпринт, 1999.
19. Медведев В.С. Потемкин В.Г. Control System Toolbox. MATLAB 5 для студентов.// М.: Изд. "Диалог-МИФИ", 1999.
20. Селезнев А.В. Эффективное программирование в среде MATLAB.// в кн. Тезисы докладов всероссийской научной конференции "Проектирование научных и инженерных приложений в среде MATLAB"// М.: ИПУ РАН. 2002. с.199-200.
21. PC MATLAB: User's Guide. MathWorks Inc., 1998.
22. Голуб Дж., Ван Лоун Ч. Матричные вычисления.// М.: Мир. 1999.
23. Воеводин В.В., Кузнецов Ю.А. Матрицы и вычисления.// М.: Наука. 1984.