КОНСПЕКТ ЛЕКЦІЙ

“ІППП з використанням засобів програмування та обчислення за допомогою MatLab”

ЗМІСТ

ЕКСПЛУАТАЦІЙНІ ХАРАКТЕРИСТИКИ СИСТЕМИ MatLab ТА РОБОТА З НЕЮ

У розділі наведені технічні та експлуатаційні характеристики системи MatLab у порівнянні з іншими математичними САПР. Описані особливості інсталяції цієї системи, її інтерфейс та загальні основи роботи з нею.

Математичні САПР та їх експлуатаційні характеристики

До 80-х років XX століття та появи персональних комп'ютерів використання ЕОМ незначно спрощувало проведення складних наукових розрахунків, хоча і збільшувало швидкість розв'язування математичних задач. Для того, щоб поставити свої задачі на ЕОМ, інженери-професіонали: фізики, математики,хіміки, інженери?електроніки повинні були вивчити структуру комп'ютера, системи введення та виведення інформації, і, найголовніше, мову програмування. Крім того, для постановки задач на ЕОМ необхідно було знати чисельні методи вирішення математичних та фізичних задач, які, як відомо, значно відрізняються від аналітичних методів розрахунку. Це вимагало від кваліфікованих працівників не просто поглибленого вивчення основ інформатики та програмування, але й у багатьох випадках повної зміни спеціалізації.

Із середини 80-х років XX століття ситуація почала кардинально змінюватися, оскільки були створені перші інтегровані системи для постановки математичних задач, які називаються математичними САПР. Найпопулярніші такі системи: Eurica, PC MatLab, MathCAD, Maple, Mathematica.

Серед широкого кола користувачів дуже розповсюджена система MathCAD, яку завжди відрізняв зручний інтерфейс, відсутність орієнтації на внутрішню мову програмування під час вирішення простих математичних задач, і наявність засобів для кодування математичних формул за допомогою зрозумілих для фізика та математика символів вектора, матриці, похідної, інтеграла тощо. У зручній формі подаються і результати обчислень. У разі потреби можна легко отримати графічне подання розрахункових даних увигляді двовимірних або тривимірних графіків. Таким чином, у цілому структура документа, сформованого у системі MathCAD, має дуже простий вигляд, і виконується за технологією WYSIWYG (абревіатура англійської фрази What You See Is What You Get — Ви отримаєте те,що бачите). Оскільки така сама система подання документів притаманна найрозповсюдженішому у світі офісному пакету Microsoft Office та операційній системі Microsoft Windows, MathCAD на сьогодні має попит серед пересічних користувачів.

Наприкінці сімдесятих років минулого століття з'явився ще один напрям створення САПР математичних розрахунків — це автоматизація аналітичних обчислень. Тоді у Києві група вчених Київського Державного Університету та Інституту кібернетики Академії Наук УРСР під керівництвом В. М. Глушкова зробила значний внесок у розвиток теорії систем автоматизації аналітичних розрахунків. Слід зазначити, що ці роботи були втілені на практиці, на їх основі було створено та реалізовано принципово нову мову програмування

математичних формул«Аналітик». На жаль, тоді ці прогресивні ідеї не знайшли належної підтримки в СРСР, але відразу були підтримані за кордоном. Нині усі без винятку математичні САПР мають засоби для підтримки аналітичних обчислень, або, говорячи мовою програмістів, засоби реалізації комп'ютерної алгебри. Є досить потужний аналітичний процесор і в системі MathCAD.

Система MathCAD дуже розповсюджена, але недостатньо популярна у науковців. Це пов'язано з недоліками цієї системи,і серед них такі:

– орієнтація на універсальні алгоритми під час проведення розрахунків, і неможливість вибору користувачем метода, найпридатнішого для вирішення конкретної задачі;

– така орієнтація на універсальні алгоритми призводить до значного збільшення часу розв'язання математичних задач;

– далеко не завжди універсальні алгоритми забезпечують користувачу необхідну точність розв'язання та його збіг. Тому дуже часто для жорстких функцій в алгебраїчних та алгебро-диференційних рівняннях MathCAD видає стандартне повідомлення «Not convergence» — «Немає збігу»;

– у системі MathCAD немає поняття математичної нескінченності, тому неможливо аналізувати функцію у точці її розриву. Це потребує від користувачів оброблення функцій, які мають точки розриву, для конкретного аналізу. А саме такі функції часто застосовуються при розв'язанні математичних задач. Наприклад, на рис. 1.1 наведено графік простої функції y(x) = 1/x3, побудований з використанням засобів системи MathCAD 2001. Зрозуміло, що функцію у точці x = 0 система відображує неточно;

– система MathCAD реалізована закритим кодом. Її розробляє тільки фірма MathSoft, і змінити закладені до неї алгоритми неможливо. В останніх версіях системи розробники MathSoft додали можливість генерації коду програми, за якою проводять обчислення, мовою програмування C++. Однак аналіз таких програм досить складний і лише у деяких випадках він дає змогу користувачу створити оптимальний код;

– у системі MathCAD відсутня можливість використання принципу модульної архітектури, яка дає змогу писати функції та процеси, що викликаються з головної програми, у модулях користувача. Це унеможливлює виклик стандартних функцій з різних програм, і тим самим ускладнює роботу програмістів під час розв'язання математичних задач;

– складність роботи система MathCAD з великими масивами чисел при реалізації матричних операцій та чисельних методів;

– внутрішня мова програмування системи MathCAD досить складна та неефективна для програмування складних математичних алгоритмів. Наприклад, окремо визначені оператори локального та глобального присвоєння, а також присвоєння в рамках циклу. Така велика кількість математичних операцій та правил їх використання призводить до ускладнення реалізації навіть не дуже складних чисельних алгоритмів, та до серйозних помилок під час складання програм.

Науковці значно ширше використовують систему Maple, ніж систему MathCAD. Вона має попитом під час аналітичного розв'язання задач та

спрощення математичних виразів. Вона також є закритою системою і тому зміна та удосконалення закладених у ній алгоритмів неможлива. Система Maple містить найпотужніший аналітичний процесор, який існує сьогодні у математичних САПР. Аналітичні процесори систем MathCAD та MatLab використовують ядро аналітичного процесора системи Maple, але їх можливості дещо менші. Недолік — відсутність можливості для реалізації чисельних розрахунків та програмування.

Серед професіоналів?математиків популярна САПР Mathematica. Однак цей пакет дуже перевантажений символьною алгеброю та математичною логікою. Якщо для професіоналів у галузі математичної логіки мова системи Mathematica є зручною та зрозумілою, то для фізика, математика?аналітика, а тим більше для інженера, такий підхід викликає певні труднощі, як і за старих часів, вимагає зміни їхньої кваліфікації.

Серед математиків, фізиків та інженерів, які володіють основами програмування та використовують методи чисельних розрахунків на персональному комп'ютері для розв'язання складних фізичних та математичних задач, що потребують оброблення значних масивів інформації, найпопулярнішою є система автоматизації науково-технічних розрахунків MatLab.

. Загальна характеристика MATLAB як системи автоматизації науково-технічних розрахунків та середовища програмування

Система MatLab (скорочення від англійської фрази Matrix Laboratory— Матрична лабораторія) особливо виділяється серед усіх сучасних математичних САПР. Ця система розвивається вже понад 30 років. Спочатку її проектували як вузькоспеціалізований програмний модуль для роботи з матрицями на великих ЕОМ. Сучасна MatLab — це універсальна, САПР, орієнтована на професійних програмістів та виконана на комп'ютерах різного класу, включаючи IBM PC, Macintosh, Unix Work Station. Система MatLab створювалася під OS UNIX, і тому, відповідно до стандартів цієї системи, вона не має уніфікованого спрощеного інтерфейсу користувача для розв'язання математичних задач і для оформлення науково-технічної документації. У цьому MatLab дещо програє популярній серед користувачів системі MathCAD, головні експлуатаційні характеристики якої наводились у попередньому параграфі. Але система MatLab має інші переваги, які вкрай важливі для професіоналів?програмістів. Це розвинуті засоби діалогу і допомоги, наявність засобів програмування високого рівня для реалізації стандартних алгоритмів, велика кількість існуючих математичних бібліотек, які, на думку фахівців, перевищують можливості найстарішої мови FORTRAN, також призначеної для розв'язання саме математичних задач. Найголовніше —це відкритий код системи MatLab, всі модулі якої написані внутрішньо мовою системи, яка аналогічна розповсюдженій мові рограмування Бейсик. Крім того, існує можливість генерації

коду налагоджених програм, мовою програмування С із метою спрощення та оптимізації, що також є дуже суттєвим для програмістів-професіоналів. Таким чином, система вдало сполучає усобі потужний апарат для математичних

обчислень і розвинуті засоби програмування, що дає змогу говорити про неї як про універсальний інструмент для розв’язання складних науково-технічних обчислювальних задач.

Систему MatLab почала створювати у 70 роках XX століття корпорація MathWorks. Ця корпорація сьогодні є лідером на ринку інтелектуальних математичних САПР і виготовляє досконалі програмні продукти насамперед для підприємств військово?промислового комплексу, аерокосмічної галузі, електронної промисловості та автомобілебудування. Тому розробники внесли в систему MatLab не тільки найсучасніші чисельні методи, створені за останні десятиліття, але й розвинуті засоби моделювання механічних і електронних систем, а також найсучасніші засоби створення програмних комплексів, зокрема візуальне та об’єктно?орієнтоване програмування.

Фірма MathWork постійно розвиває систему MatLab і є лідером угалузі створення математичних САПР, оскільки вона весь час вносить у систему важливі нові елементи, які потім починають використовувати інші розробники. Так, усистемі MathCAD,починаючи з версії 6.0 PRO, розроблема внутрішня мова програмування, а починаючи з версії MathCAD 2000 у цій системі є можливість генерації коду злагоджених програм мовою C++.

Усі ці можливості у MatLab уже давно існували. Є система моделювання Simulink. Починаючи з версії 5.3 MatLab і Simulink постачаються в одному комплекті, що дає змогу користувачу розв’язувати системи рівнянь, а потім моделювати з використанням математичного апарату та графічних засобів аналогові,цифрові, лінійні, статичні та динамічні системи різної фізичної природи, зокрема електронні. Засоби Simulink є найбільш доступними та ефективними інструментами для розв’язання нетривіальних задач у таких напрямах науки: програмування нечіткої логіки (fussy logic), теорія нейронних мереж, аналіз сигналів і оброблення динамічних зображень, що включають вейвлет (wavlet) технологію. Велика кількість науково-технічної документації та літератури англійською, російською та українською мовами, а також інтернет-ресурси роблять MatLab найпопулярнішою системою в освітянських закладах.

Технічні характеристики системи MatLab та галузі її використання

Для узагальненої характеристики можливостей системи MatLab розглянемо спочатку головні математичні функції, призначені для подання та оброблення результатів розрахунків. У MatLab, крім можливостей обчислення елементарних алгебраїчних та тригонометричних функцій, великої кількості математичних операцій над числами, матрицями та векторами, що включають розв’язання вищої математики, лінійної алгебри та аналітичної геометрії, функціонального аналізу, використання різноманітних чисельних методів розв’язання нелінійних рівнянь та їх систем, диференціальних рівнянь, математичної фізики, а також методів інтерполяції та апроксимації функцій.

Перелік головних математичних можливостей системи MatLab.

1. В MatLab є можливість обчислення спеціальних математичних функцій, які часто?густо використовують у функціональному аналізі та в інженерній практиці при розв’язанні диференціальних рівнянь математичної фізики. Серед таких функцій необхідно зазначити функції Беселя, Ейрі, бета-функцію, гама-функцію, функцію помилок, еліптичні інтеграли, інтегральну показникову функцію, псі#функцію, поліноми Лежандра та Чебишева.

2. MatLab є велика кількість операцій з векторами та матрицями, зокрема такі, як: конкатенація матриць, створення матриць із заздалегідь заданою діагоналлю, поворот, виділення, створення матриць Адамара, Шенкеля, Гільберта, Паскаля, Уілкінсона та інших.

3. У MatLab викладені чисельні методи розв’язання нетривіальних задач геометрії та класичної механіки, зокрема методи тріангуляція ділянок, обчислення площ опуклої оболонки, площ правильних та неправильних багатокутників.

4. Важливою перевагою системи MatLab є опрацювання елементів упорядкованих структур, зокрема обчислення мінімально та максимально значення, сортування, обчислення середнього та середньоквадратичного значення, стандартного відхилення елементів масиву, коефіцієнтів кореляції та матриці коваріації. Відомо, що задані визначення мінімальних та максимальних елементів та їх сортування, які досить часто використовують програмісти при обробленні розрахункових даних, до недавнього часу не мали стандартних функцій навіть у найрозвиненіших мовах програмування, наприклад, FORTRAN, Pascal, C або C++. Тому наявність таких функцій у MatLab можна розцінювати як важливе досягнення розробників системи, яке значно полегшує роботу програмістів.

5. У MatLab впроваджена велика кількість чисельних методів для розв’язання систем лінійних рівнянь без обмежень або з обмеженнями значення змінних, методів вирішення нелінійних рівнянь, а також чисельних методів визначення екстремумів функцій однієї або кількох змінних.

6. Методи інтерполяції та апроксимації функцій: поліноміальна регресія, Фур’є -інтерполяція для періодичних функцій, одновимірна інтерполяція, двовимірна інтерполяція, багатовимірна інтерполяція, кубічні сплайни.

7. В Операції функціонального аналізу, що включають багатовимірне пряме перетворення Фур’є, швидке перетворення для простих одновимірних функцій, пряму та зворотну згортку, методи фільтрації.

8. Математичні операції для роботи з поліномами, що включають множення та ділення поліномів, обчислення коренів полінома, обчислення похідних, розкладення поліномів на прості дроби, розв’язання поліноміальних матричних рівнянь.

9. Методи інтерполяції та апроксимації похідних, зокрема апроксимація лапласіану, апроксимація похідних кінцевими різницями, обчислення градієнту функції.

10. Методи числового інтегрування функцій, зокрема метод трапецій, метод квадратур, метод Сімпсона. У версії MatLab 6.5 додані нові функції для обчислення потрійних інтегралів.

11. Методи для розв’язання звичайних диференціальних рівнянь (ЗДУ), зокрема: однокрокові методи Рунге-Кутта другого, четвертого та п’ятого порядків, багатокроковий метод Адамса-Башворта-Мултона, наявні методи Рунге -Кутта та метод Гіра.

12. Методи для розв’язання диференціальних рівнянь у часткових похідних, зокрема рівняння Лапласа, рівняння Пуасона, рівняння теплопровідності, хвильового рівняння.

Приклади розв’язання рівнянь математичної фізики можна знайти у демонстраційному пакеті MatLab у розділі Partial Differential Equations.

Як можливість подання результатів математичних розрахунків та програмування можна зазначити двовимірні та три вимірні графіки, анімаційна та дескрипторна графіка, інтерфейси користувача. Перелік графічних можливостей системи MatLab.

1. Побудова звичайних двовимірних графіків з задаванням масштабів осей, їх підпису, геометричних фігур для позначення точок графіка та інших елементів оформлення.

2. Побудова двовимірних графіків у логарифмічному масштабі.

3. Побудова різноманітних діаграм, зокрема стовпчикових та колових.

4. Побудова сходинкових графіків.

5. Побудова двовимірних графіків дискретних відліків функції та графіків із зонами похибки.

6. Побудова графіків у полярній системі координат та кутових гістограм.

7. Побудова контурних тривимірних графіків.

8. Побудова кольорових тривимірних графіків, на яких зміна кольору відповідає зміні функції.

9. Побудова тривимірних графіків з різним освітленням поверхні, на яких зміна освітленості відповідає зміні функції.

10. Засоби анімації, зокрема моделювання руху точки площині та у просторі.

11. Засоби дескрипторної графіки, які використовуються у тому випадку, коли жодна з наявних технологій візуалізації даних не влаштовує користувача і він хоче створити свою систему на основі елементарних геометричних об’єктів, таких як точка, лінія, коло, прямокутник та інші. У цьому разі у системі MatLab можуть бути ефективно використані засоби об’єктно-орієнтованого програмування.

12. Засоби побудови стандартного графічного інтерфейсу користувача GUI (Graphic User Interface) .

Інсталяція та завантаження системи MatLab

Нові версії системи MatLab є досить громіздкими для використання комп'ютерних ресурсів. Наприклад, версія 6.0, яка вийшла у 2001 році, займає на жорсткому диску від 1000 до 1500 Мбайт. Але приблизно 70% від цього обсягу становлять файли допомоги та довідникової системи, записані у форматах PDF (Postscript Data Format — формат даних для друку) та HTML (HyperText Metafile Language — гіпертекстова міжфайлова мова). Як відомо, ці формати використовують також для збереження інформації в Інтернеті. Ці формати удовідникових системах та системах допомоги дають змогу спростити подання інформації при гіпертекстових посиланнях між розділами документів. Але не завжди користувачу треба встановлювати всі довідники. Тому інсталяція MatLab постачається на двох ком пакт-дисках, на одному з них розташовуються архіви системних файлів та бібліотек, а інший містить довідники та документацію роботи із системою.

Головні вимоги до апаратного та системного забезпечення персонального комп'ютера, на якому користувач бажає встановити систему MatLab, є такими:

Вимоги до мікропроцесора та системної плати —комп'ютер з мікропроцесором Pentium. Рекомендується мікропроцесор Pentium PRO, Pentium II, Celeron, Pentium III, Pentium IV, AMD Athlon. Майте на увазі, що для встановлення на комп'ютер із процесором Pentium IV підходять тільки версії MatLab 6.1 та вищі.

Наявність маніпулятора «миші».

Вимоги до відеоадаптера — 8-розрядний графічний відео-адаптер та монітор, який підтримує як мінімум 256 кольорів.

Вимоги до оперативної пам'яті — 64 Мбайт оперативної пам'яті, рекомендований обсяг оперативної пам'яті — 128 МБайт.

Вимоги до вільного простору на жорсткому диску. При повному встановленні всіх компонентів системи, що включають довідники — 1,5 Гбайта вільного простору на жорсткому диску. Мінімальний обсяг при зменшенні обсягу довідникової документації становить 1 Гбайт.

Додаткове апаратне та програмне забезпечення персонального комп'ютера, необхідне для використання розширених функцій системи MatLab, зокрема можливостей перегляду технічної документації, оновлення системи через Інтернет та використання функцій програмування:

1. Графічний акселератор.
2. Звукова карта.
3. Принтер.
4. Текстовий редактор Notepad (Блокнот).
5. Текстовий процесор Microsoft Word.
6. Підключення комп'ютера до Інтернету через модем або через виокремлену лінію.
7. Програми перегляду Інтернет-сторінок, тобто Microsoft Internet Explorer
8. Acrobat Reader

9. Компілятори мов програмування C, C++ або FORTRAN.

Інсталяція системи у середовищі Windows 95/98/ME/2000/ NT4/XP проводиться звичайним чином і не має ніяких специфічних особливостей. Є два типи інсталяції: типова та вибіркова. Якщо не встановлювати якісь довідники або системні бібліотеки, можна вибрати вибіркову інсталяцію продукту, але треба мати на увазі, що при цьому будуть працювати не всі функції системи. Якщо Ви не впевнені в необхідності тих або

інших функцій, краще їх установити.

Послідовність дій при інсталяції системи MatLab:

1. При встановленні з локального диску запустити програму setup.exe.

2. Початок встановлення: відкриття вікна майстра, натискають кнопку NEXT (далі).

3. Відкривається вікно із запитом на введення ліцензійного номера продукту. Введіть ліцензійний номер та натисніть кнопку NEXT.

4. Ознайомтеся з ліцензією. Якщо Ви згодні з її текстом —натисніть кнопку NEXT.

5. Відкривається вікно з двома коробочками для введення тексту. У верхню коробочку введіть ім’я користувача, у нижню — організацію, після чого натисніть кнопку NEXT.

6. Відкривається вікно, у якому можна вибрати диск та директорію, до якої буде встановлена система, а також перелік компонентів. Тут вказується також обсяг, необхідний для кожного з компонентів, та розмір вибраного диску. Така повна інформація дасть змогу легко вибрати необхідні для встановлення компоненти. Наприклад, можна встановити файли допомоги японською мовою. Можна взагалі відмовитися від встановлення документації.

7. Після вибору всіх необхідних компонентів натисніть кнопку NEXT. З’явиться вікно копіювання файлів на диск. Зрозуміло, що це найдовший із процесів інсталяції, але він не потребує участі користувача. Усі файли розпаковуються з пакету та записуються на жорсткий диск у відповідні папки. Контролювати процес запису файлів можна лінійним індикатором.

8. При переході від встановлення системних засобів до встановлення документації слід змінити встановлювальний диск на диск з документацією і чекати далі. За статистикою встановлення повної версії системи на комп’ютері з процесором Pentium II 350 МГц та 32-швидкісним приводом становить близько двох годин.

9. Завантаження програми відразу після її встановлення без перезавантаження комп’ютера не рекомендується, оскільки окремі її функції можуть працювати неправильно. Але це зауваження стосується лише версій системи 6.0 та вищих, версії 5.x працюють без помилок і без перезавантаження комп’ютера. Завантаження програми MatLab можна здійснювати через файл MatLab.exe або звернення до відповідної іконки.

Якщо система не виконує якусь команду або програму, яка на іншому комп’ютері працювала правильно, на це може бути дві причини:

1. Несумісність версій системи. Якщо для багатьох математичних САПР вирішити цю проблему неможливо, для системи MatLab, як буде показано далі, ця

проблема є не дуже складною і вирішується переписуванням окремих бібліотечних файлів.

2. Система неправильно налаштована. Далі буде пояснено, як слід правильно налаштовувати систему MatLab, які головні директиви повинні бути на диску, та які файли у них повинні зберігатися.

Інтерфейси різних версій системи MatLab та їх особливості

Загальний вигляд системи меню оболонки MatLab

В оболонці MatLab 6.0 саме інтерфейс системи зазнав серйозних змін. Раніше він був досить скромним і лаконічним, зробленим згідно з ідеологією системи, призначеної для професіоналів. Оскільки початково система MatLab розроблялася під операційною системою UNIX, принцип її робот відповідає ідеології цієї операційної системи. Усю необхідну інформацію щодо призначення команд системи та їх синтаксису, параметрів та необхідних ключів, користувач може отримати за допомогою п’яти спеціальних команд:

help — виведення довідника в усіх розділах системи;

demo — виведення списку демонстраційних прикладів;

whatsnew — нові рішення даної версії (для досвідчених користувачів попередніх версій);

info — загальна довідкова інформація про систему;

subscribe — підписатися на відновлення системи на www.mathworks.com

Далі робота із системою йде з використанням тільки системи команд. Звертання до функцій меню здійснюється тільки для проведення системних операцій, наприклад, для завантаження файлу, завершення роботи, друку результатів, хоча навіть ці функції можна виконати через команди. Відповідно, інтерфейс системи версій 5.x має вигляд, стандартний та простий. Він містить рядок меню для виклику головних функцій переналаштування системи та системних операцій, спливаючі вікна, які реалізують ці функції, іконки, які у більшості випадків дублюють ці функції, а користувач послідовно вводить команди, які розташовуються у командних рядках.

Такий спрощений інтерфейс має свої переваги. Відсутність великої кількості вікон, іконок та стандартний рядок меню, дають змогу використовувати майже весь екран монітора і зберігати на екрані велику кількість введених користувачем команд. Тому саме такий інтерфейс є зручним для програмістів-професіоналів, які досконало знають усі можливості системи та можуть за допомогою відповідних команд її налаштувати та ефективно працювати з нею. Для того, щоб збільшити розмір командного вікна, можна також відключити іконки за допомогою функції Toolbar опції меню View. Якщо введена велика кількість команд і всі вони не вміщуються на екрані, для перегляду команд та повторного виклику будь?якої з них можна використовувати клавіші керування курсором ↑ та ↓. Крім того, для перегляду введених команд можна використовувати клавіші Page UP та Page Down. Така система роботи з командним рядком та спрощеним інтерфейсом притаманна операційній системі UNIX, у якій початково створювався MatLab.

Для користувачів ОС Windows, які тільки вчаться працювати з системою MatLab, на першому етапі зручніше використовувати рядок меню, оскільки для того, щоб запам’ятати всі команди, призначені для налагодження системи, треба матинемалий досвід роботи з нею.

Виклик опції меню File: на екрані з’являється нове меню,що включає наступні команди:

New — створити новий файл;

Open — відкрити проект MatLab;

Run Script — виконати M?файл, який є програмою, написаною мовою програмування MatLab;

Save Workspace As — зберегти дані у файлі для подальшого їх використання;

Load Workspace — завантажити попередньо сформований файл;

Show Workspace — показати імена змінних;

Show Graphics Property Editor — показати вікно редактору, у якому властивості графіків та засобів подання інформації системи MatLab;

Show GUI Layout Tool — показати вікно властивостей засобів проектування графічного інтерфейсу користувача;

Set Path — установити шлях до поточної робочої директиви системи MatLab;

Preferences — налаштування системних змінних та інтерфейсу системи MatLab;

Print — виведення на друк вмісту командного вікна. У цьому разі вважаються всі введені команди, а не лише ті, які відображені на екрані;

Print Selection — виведення на друк виділених рядків;

Print Setup — налаштування параметрів принтера та параметрів друку, яке здійснюють через використання стандартних засобів системи Windows;

Команда Show Workspace опції меню Filedає змогу користувачу переглядати та аналізувати імена, тип, надмірність та розмір у байтах для усіх змінних, які були сформовані та завантажені.

Усі завантажені користувачем змінні називаються робочою частиною програми. Для того, щоб подивитися значення змінної, достатньо викликати її, підвівши курсор до відповідного рядка та натиснувши ліву кнопку миші. Таким чином викликається внутрішній налагоджувач системи MatLab Editor/Debugger. Досконаліше поняття змінної та робочої частини. Особливо ефективно можна використовувати засоби налагодження при програмуванні у системі MatLab. Можливості програмування будуть розглянуті.

Команда Show Graphics Property Editor опції меню File дає змогу встановлювати та змінювати властивості системи відповідно до графічного вікна. Через виклик цієї команди меню завантажується програма Graphics Property Editor — редактор властивостей графіків. Інтерфейс цієї програми містить два перемикача за системою І»: Show Object Browser (показати вікно перегляду об’єктів) та Show Property List (показати вікно перегляду властивостей об’єктів). За замовчуванням обидва перемикачі ввімкнуті, що відповідає відображенню обох вікон. Зміна властивостей графіків у цьому разі можлива у вікні перегляду об’єктів через відповідні значення. Крім того, редактор властивостей графіків має власне командне меню. Тут головною є опція меню Tools (інструменти). Ця опція

дає змогу викликати інші засоби, призначені для зміни властивостей графіків. Основна програма — Control Panel (панель налаштовування), яка являє собою зручний інтерфейс для завантаження інших засобів, зокрема редактора для встановлення властивостей вирівнювання графіків.

Команда Open M?file відкриває діалогове вікно з переліком наявних на диску файлів з розширенням *.m, на яких повинен міститися текст програм, написаних мовою програмування системи MatLab. За умови вибору файла зі списку і натиснення кнопки OK запускається текстовий редактор із текстом відповідного M-файлу. Збереження файлу здійснюється через функції текстового редактора.

Команда Save WorkSpace As — «зберегти робочу частину як», у файлі із заданим ім’ям та з розширенням *.mat, значення всіх змінних, які були визначені у командних рядках. Слід мати на увазі, що у файлах з розширенням *.mat зберігаються тільки значення змінних, а не командні рядки.

При виконанні команди Run M-file («Виконати M-файл»), як і при виконанні команди Open M-file, із переліку файлів із розширенням *.m позначений файл викликається на виконання. Узагалі файли з розширенням *.m являють собою програми, написані внутрішнбою мовою системи MatLab.

Команда меню File Print виводить на друк увесь текст, який був набраний у вікні, при цьому параметри друку повинні бути попередньо задані за допомогою команди Printer Setup.

Команди опції меню Edit

У пункті меню Edit знаходяться стандартні операції обміну даних для оболонки Windows: Copy— копіювати, Cut—вирізати та Paste — вставити. Якщо немає виділених рядків,то команди Copy та Cut неактивні, тоді у спливаючому меню вони пишуться не яскраво?чорним, а блідо?сірим кольором. Виділення тексту програми у вікні командного рядка здійснюється стандартним способом — натисненням лівої кнопки миші. Опція Clear меню Edit видаляє текст без його копіювання у буфер обміну даних, а опція Clear Session очищує командне вікно, значення всіх уведених змінних зберігаються. Опція Select All призначена для виділення всього вмісту командного вікна.

Опція меню Clear Session призначена для очищення командного вікна. Спливаюче вікно команди Меню команди Edit

Опція Preference команди меню File та налаштування параметрів системи

Опція Preference призначена для зміни параметрів оформлення командного вікна оболонки MatLab та параметрів виведення результатів. При зверненні до неї відкривається вікно з трьома вкладниками: General, Command Windows Font та Copying Options.

Серед параметрів, які знаходяться на вкладниці General, є подання чисел при виведенні результатів розрахунків, під назвою Numeric Format— числовий формат. Усистемі MatLab передбачено такі формати виведення числових даних.

Short — стислий запис. Використовується системою по замовчуванню, десяткові знаки після коми відповідають найменшій кількості введених цифр у запису числа;

Long — довгий запис. При використанні цього формату всі числові результати виводяться з найвищою точністю, п’ятнадцятьма десятковими знаками після коми;

Bank — банківське, або економічне подання результатів, яке використовується в документах та кошторисах, з округленням до другого знака після коми;

Plus — знаковий формат. При виведенні результатів записується тільки знак числа;

Short e— формат стислого запису із плаваючою комою. Результати записуються такі: s e m, s — характеристика, m —мантиса числа. Кількість знаків після десяткової коми, відповідає меншій кількості десяткових знаків, а на запис мантиси відводиться 4 позиції — на першій завжди стоїть знак мантиси. Наприклад, число 1,9•10–19записують: 1.9e?019, а число 6•1023— 6.0e+023.

Long e — формат довгого запису із плаваючою комою. Числа записують через характеристику та мантису, але, на відміну від короткого запису, кількість знаків дорівнює 16;

Rational— подання числа правильним раціональним дробом.

Опції виведення даних Loose та Compact використовуються для роботи з великими командними рядками. Якщо довгі математичні вирази не вміщуються у формат екрану, то при виведенні даних Compact довгі рядки переносяться, а при форматі Loose — йдуть одним рядком, тоді для їх перегляду необхідно скористатися лінією прокрутки по горизонталі.

Параметри виведення результатів можна також змінити за допомогою команди format. Приклади використання команди формат:

> format short e

> format bank

> format rational

> format +

Розглянемо приклад, у якому форма подання результатів розрахунків змінюється через MatLab.

```
»
» format short
» 3/5
ans =
0.6000
» format long
» 3/5
ans =
0.60000000000000
» format short e
» 3/5
ans =
```

```
6.0000e?001
» format bank
» 3/5
ans =
0.60
» format rational
» 3/6
ans =
1/2
» format +
» 3/7
ans =
+
```

Легко побачити у наведеному прикладі, що оскільки виразу 3/5 не було надане ім’я будь?якої константи, система за замовчуванням присвоює йому найменування змінної ans (від англійського слова answer — відповідь).

Порівняння двох способів налаштування параметрів виведення розрахункових даних — через меню системи та командний рядок, дає змогу зробити важливі висновки щодо загальної концепції системи. Як ми бачимо у наведеному прикладі, меню виконує ті самі функції. Який же спосіб роботи зручніший? Зрозуміло, що користувачі, які добре знають загальні основи роботи ОС Windows, але тільки починають працювати з MatLab, частіше користуються системою меню і командними вікнами, оскільки для них цей шлях зручніший. Але використання командного рядка має свої переваги для професійних користувачів системи MatLab. Це пов’язано з широкими можливостями редактора.

Дані раніше команди можна копіювати або повторювати, використовуючи клавіші переміщення курсору та функції пункту меню Edit з рядка меню. Загалом перевагу має саме робота з командними рядками. Якщо Ви користуєтесь опціями налаштування меню при відлагодженні, а потім встановлюєте програму на іншому комп’ютері, то перед тим, як запускати програму на тестування, Вам необхідно налагодити параметри системи. Цей недолік є дуже суттєвим у написанні розрахункових програм мовою системи MathCAD. Тому програмісти завжди підлагоджують програми таким чином, щоб всі параметри налаштовувались через командний рядок і не залежали від опцій меню. Надалі, при вивченні можливостей цієї системи, ще не раз траплятимуться випадки, коли командні рядки дублюють функції системного меню. Завантаження та закриття файлів також можна робити через командні рядки, і в деяких випадках такий спосіб зручніший.

Розглянемо призначення інших параметрів конфігурації системи.

Editor preference — функція налаштування текстового редактора, який буде використовуватися при наборі текстів програм. Тут надається дві можливості: можна брати внутрішній редактор системи або працювати з ОС Windows. Якщо віддаєте перевагу іншому редактору, необхідно вказати шлях до відповідної

програми у командному вікні. У цьому разі для пошуку відповідного файла на жорсткому диску можна натиснути кнопку Browse

Echo on — функція «Включення відлуння» при обробленні програм, написаних мовою MatLab. У режимі Echo On програма одночасно виводить рядок команд, які виконуються. Це дуже зручно при налагодженні програми, але незручно при її використанні для модельних експериментів. Тому після налагодження програми включати функцію Echo On не має сенсу. Включення та відключення функції відлуння можна здійснювати через командний рядок:

```
> echo on
та
> echo off
```

За допомогою функції Show Toolbars відображується рядок панелі інструментів. Відключення дає змогу використати поле екрана для відображення решти команд.

Функція Enable Graphical Debugging (можливість відлагодження графіки)підключає або відключає можливість аналізу графічних команд. У деяких випадках можна її відключати — це спрощує аналіз математичних алгоритмів.

На вкладниці Command Windows Font можна змінити тип (Font) та розмір (Size) шрифту, яким пишеться командний рядок, а також колір шрифту (Color) та фон (Background Color) командного вікна. Наприклад, одним користувачам більше подобаються чорні літери на фоні як білому, у текстовому редакторі ОС Windows, а іншим — білі на чорному фоні, як у системному командному вікні.

На вкладці Copying Options можна знайти три поля: Clipboard Format, Match Figure Screen Size, White Background. За їх допомогою задається форма та властивості графічних об’єктів при їх копіюванні з оболонки MatLab. У полі Clipboard Format замовляються тип графічних об’єктів при копіюванні убуфер обміну Windows. Це можуть бути мета файли Windows(WindowsMetafileFormat,WMF)або точкове поєднання (Windows Bitmap). Формат точкового подання забезпечує високу якість при передачі графічних зображень, але обсяг інформації, який передається у буфер обміну даних, при копіюванні значно більший, ніж у форматі WMF.

Опція MatchFigureScreen Size забезпечує копіювання графічних об’єктів з оболонки MatLab, графіки мають вигляд, у якому вони з’являються на екрані. При відключенні цієї опції розміри графіків установлюють відповідно до формату сторінки, до якої вставляється графічний об’єкт. Опція WhiteBackground встановлює вид об’єкту з білим фоном.

Команда головного меню Window

Команда Window аналогічна подібній команді інших засобів текстового редактора. При зверненні до неї з’являється спливаюче меню списку відкритих вікон. До них можна звернутися, підвівши курсор до відповідного рядку списка та натиснувши ліву кнопку миші. Наприклад, після виведення графіків в окремому вікні, можна звернутися до нього через команду Window.

Особливості інтерфейсу оболонки MatLab 6.0 та її довідникової системи

Користувачів, які мають досвід роботи із системою MatLab, дивує новий інтерфейс системи MatLab 6.0. Раніше описаний інтерфейс MatLab 5.3, зручний лише для програмістів-професіоналів.

Відмінністю наведеного інтерфейсу є наявність вікна допомоги, зробленого у зручному стилі гіпертекстового документа, знайомого всім користувачам Internet. Тому пошукову систему MatLab версії 6.0 називають Help navigator — навігатор допомоги. На відміну від багатьох сучасних виробників програмного забезпечення, фірма MathWork відмовилася від використання стандартних Internet-браузерів для виведення інформації та розробила для цього зручний інтерфейс. У ньому є Internet дерево каталогів із перемикачами «+» та «–», за допомогою яких відкриваються та закриваються розділи довідника. Як зазначається у розділі 1.4, MatLab інтегрована система моделювання динамічних об’єктів Simulink і вона може бути використана для електронних схем та систем з великою кількістю компонентів. Але особливості її застосування предмет окремого розгляду. Наведемо розділи довідкової системи MatLab, розбивши їх на окремі групи, відповіднодо змісту.

Загальні основи роботи із системою

• особливості інсталяції системи MatLab(Installation);

• можливості отримання версій та системи через Internet (Real-Time Workshop);

• опис компілятора MatLab та бібліотек для зв’язку проектів з мовами програмування C та C++ (MatLab Compiler, Math Library, MatLab C/C++ Graph Library).

Математичні бібліотеки

• розв’язання задач оптимізації (Optimization Toolbox);

• чисельне розв’язання диференціальних рівнянь у часткових похідних та їх систем (Partial Differential Equation (PDE) Toolbox);

• розв’язання статистичних задач (Statistics Toolbox);

• розв’язання задач сплайнової інтерполяції (Spline Toolbox);

• символічні математичні перетворення.

Моделювання електронних схем, систем та сигналів

• система для моделювання складних об’єктів з обмеженою кількістю станів Stateflow;

• мережі мобільного зв’язку з доступом до ресурсів стандарту CDMA (Code Division Multiple Access — множинний доступ з кодовим розділенням каналів);

• засоби моделювання об’єктів зв’язку Communication Toolbox;

• засоби лінійних, нелінійних систем автоматизованого керування з підвищеною стійкістю (Control System Toolbox, Nonlinear Control Design Blockset та Robust Control

Toolbox);

• засоби математичного та графічного моделювання вимірювальних систем (Instrument Control Toolbox та Requirements Management Interface);

• блоки цифрового оброблення сигналів DSP (Digital Signal Processing та Motorola DSP Developer’s Kit);

• блок цифрових фільтрів Filter Designing Toolbox;

• моделювання систем енергозабезпечення (Power System Blockset);

• засоби систем з нечіткою логікою Fuzzy Logic Tool-box;

• математичне оброблення сигналів Signal Processing Toolbox;

• ідентифікація складних систем різної фізичної природи (System Identification Toolbox).

Інші функції системи

• робота з базами даних Database Toolbox;

• система математичного оброблення зображень (Image Processing Toolbox та Wavelet Toolbox);

• математичні бібліотеки розв’язання задач економіки та бізнесу, зокрема основні та допоміжні функції, а також часові ряди для проведення фінансових обчислень (Financial Toolbox, Financial Derivatives Toolbox, Financial Times Series).

Одна з переваг системи MatLab, і завдяки їй вона є найпопулярнішою САПР серед професійних програмістів. Розвинута довідникова система, за вміння користуватися нею, допомагає фахівцям під час розв’язання складних задач. Але скористатися наведеною структурою не завжди зручно. Так, розділи пакету Simulink, у яких описується можливість моделювання схем, систем та сигналів, мають досить прозору структуру, а кількість розділів, у яких описуються математичні функції пакету MatLab, — досить обмежену.

В електронному довіднику немає окремих розділів реалізованих елементарних та спеціальних математичних функцій, а також чисельних методів розв’язання нелінійних рівнянь та їх систем. Відсутній окремий розділ для функцій оброблення структур даних та матриць є невід’ємною складовою частиною системи MatLab. Довідник не має також окремого розділу для великої кількості графічних систем, зокрема побудови двовимірних та тривимірних графіків та діаграм, які набули популярності при оформленні наукової та ділової документації. Крім того,довідник має тільки посилання на формат запису відповідних функцій, параметри та можливості їх застосування у більшості випадків не розтлумачуються.

Початок роботи з довідником відбувається у вікні, після звернення до навігатора допомоги MatLab → Help. При цьому з’являється вікно навігатора допомоги. У правій частині внизу відображуються сторінки, а зверху знаходяться кнопки навігації:

повернутися на крок назад (Back),

перейти на крок уперед (Forward),

та перезавантажити поточну сторінку (Reload).

Поряд з кнопками, знаходиться система пошуку вказаного користувачем контексту на завантаженій сторінці довідника, текстове вікно для введення інформації та кнопка Go. Крім того, тут знаходиться відома іконка для виведення вмісту документа на друк

Біля текстового вікна знаходиться кнопка «Add to Favorites» — «Запам’ятати сторінку». Натиснувши на цю кнопку користувач може додати до списку сторінок, необхідні для подальшої роботи.

Текст сторінок довідника читати незручно через малу ширину вікна. Урахувавши це, розробники системи передбачили можливість відключення вікна навігатора допомоги для збільшення розміру, розташування сторінки довідника. Таку операцію можна виконати двома шляхами: натисненням кнопки з хрестом або через опцію меню. Поновити вигляд вікна можна через опцію системного меню, натисненням кнопки .

Можна побачити, що «вікно навігатора» має чотири вкладинки, які відповідають різним режимам робити довідкової системи. На першій вкладниці Contents знаходиться зміст довідника з посиланням на розділи. Зазначена структуризація не завжди може задовольнити користувача.

Друга вкладинка Index відповідає пошуку інформації у довіднику в алфавітному порядку. Слід зазначити, що відразу після інсталяції системи ця функція може не працювати, при цьому система видає повідомлення «Error: Help directory is not set» («Помилка: Не вказана директорія з файлами допомоги»). Уникнути цієї помилки можна через функцію системного меню File → Set Path. При зверненні до цієї функції з’являється командне вікно Set Path (Установлення шляху).

Для встановлення директорій необхідно натиснути кнопку «Add with Subfolders» та знайти її у вікні на дереві каталогів. У цьому випадку треба додати директорію, розташовану у каталозі, який був визначений для інсталяції MatLab. Після цього слід натиснути кнопку «Ok» та зачекати, поки виконається операція. Слід зазначити, що зміни до списку робочих каталогів заносяться відразу, і система в алфавітному порядку починає працювати.

На третій вкладинці «навігатора допомоги» «Search» («Пошук») можна виконати пошук сторінок довідника та функцій за ключовим словом. При переході на цю вкладинку з’являються два текстових вікна. В одному із них, потрібно вказати один з чотирьох можливих типів пошуку («Весь текст», «Заголовки документів», «Імена функцій»,«Безпосереднійвихіднабазиданих Інтернет»).

Крім того, поряд із вікном «Search type» знаходиться кнопка «Tips» («Підказки») . При натисненні відкривається документ, у якому описані функції навігатора допомоги.

У текстовому полі «Search For», яке знаходиться внизу, вказується фрагмент, за яким проводиться пошук. Посилання на сторінки довідника з’являються після введення тексту та натиснення кнопки «Go». Тут слід зазначити великий обсяг

довідника MatLab та його орієнтованість на професіоналів математиків та програмістів.

Як бачимо, завантажена сторінка містить не тільки досконалий опис алгоритмів пошукового методу, але й посилання на англомовну літературу, список якої завжди наводиться у кінці тексту.

Нарешті — це пошук серед обраних сторінок, які виводяться на вкладниці «Favorites». Спосіб запам’ятовування за допомогою кнопки «Add to Favorites». Переглянути обрану сторінку можна, звернувшись до списку обраних сторінок.

Крім розвинутого «навігатора допомоги» в інтерфейсі користувача MatLab 6.0 з’явилася додаткова опція Web, через яку можна отримати доступ до сайту фірми MathWork в Інтернеті (www.mathwork.com) для оновлення системи та для отримання додаткової інформації англійською мовою. Серед численних російськомовних Інтернет?сайтів, присвячених MatLab, найпопулярнішим є www.exponenta.ru

Зазначимо, що досвідчені користувачі MatLab задоволені новим інтерфейсом та «навігатором допомоги». По?перше, як наголошувалось, нова система допомоги має опис випроваджених алгоритмів та посилання на наукові джерела, де ці алгоритми описані досконаліше. Це дуже спрощує використання досвідченими програмістами складних розрахунків, особливо у разі внесення змін до базових функцій системи. По-друге, для тих користувачів, які звикли до старого стандарту інтерфейсу, є можливість змінити його за допомогою команди View→Desk topLayout→CommandWindowsOnly. Після цього інтерфейс системи спрощується та схожий до наведеного.

Для оновлення багатовіконного інтерфейсу MatLab 6.0 достатньо виконати функцію View → Desktop Layout → Default.

Дуже зручними у багатовіконному інтерфейсі MatLab 6.0 є вікна історії команд (Command History) та робочого середовища (Work Space). За допомогою вікна історії команд можна звернутися до будь-якої з команд, переглянувши їх за допомогою лінії прокручування та позначення курсором команди, яку потрібно повторити. Принцип роботи вікна команд історії.

У попередніх версіях MatLab також існувала можливість повторення команд за допомогою клавіш переміщення курсору ↑ (вгору) та ↓ (вниз), в спосіб, що притаманний UNIX, але переглядати велику кількість команд значно зручніше через багато-віконний інтерфейс. Тому поява нової версії полегшила роботу як малодосвідчених користувачів, так і фахівців-програмістів.

Командне вікно вкладинки робочої частини WorkSpace відповідає команді меню Файл та вікну налагодження. У ньому можна переглянути значення змінних, формати подання числових даних. Розташування вікна робочої частини у багато-віконному інтерфейсі MatLab 6.0.

Слід зазначити, що рядок Меню File MatLab версії 6.0 має інший інтерфейс, схожий знавігатором, на якому, замість вкладок, використовуються значки «+», «–». Вікно Preference MatLab 6.0

Меню MatLab має вкладинку Demos, яка корисна тим, хто починає працювати з цією системою. За допомогою цієї команди ви можете викликати демонстраційну галерею, у якій знайдете приклади застосування системи MatLab

для виконання наукових розрахунків. У разі звернення до вкладки Demos відкривається «вікно галереї» прикладів, дуже схоже на «вікно допомоги». Ліворуч побачите список розділів, праворуч — список прикладів. Щоб побачити список підрозділів, треба натиснути на кнопку «+». Натиснувши лівою кнопкою миші на відповідний приклад, ви відкриєте вікно його демонстрації.

Поряд з «вікном навігації», показаний з приклад, який називається «Curve Fitting» — «Апроксимація кривих».

Користувач має можливість натисненням кнопок Start (Почати), Next (Наступна) та Pervious (Попередня) у правій частині вікна послідовно розглядати код функцій MatLab у лівій частині вікна внизу та результат виконання цих функцій.

Слід зазначити, що користуватися меню у MatLab не обов'язково. Ті самі функції виконують команди help, demo та what snew, наприклад:

help functions — виведення функцій;

help sin — опис функцій sin;

whatsnew — нові функції;

demo — інформаційне вікно з прикладами.

Тому інтерфейсом меню MatLab версії 6.0 і вищих версій користуються недосвідчені користувачі, професіонали частіше працюють з командним рядком.

Особливості файлової структури оболонки MatLab

Повна версія MatLab складається з багатьох тисяч файлів,які знаходяться у великій кількості папок, і займають 1Гб. Для вивчення особливостей роботи складної системи — необхідно зрозуміти, що це за папки, яка інформація у них зберігається. Крім того, через наявність папки на комп'ютері, можна з'ясувати, чи встановлені відповідні функції на даному комп'ютері, і, як буде працювати на ньому ваша програма.

Для користувачів MatLab особливе значення мають файли двох типів — це розширення *.mat і *.m. У файлах *.mat можна зберегти всі числові змінні програми, але у стислому форматі.

Історія команд не запам'ятовується. У файлах *.m зберігаються тексти програм, виклики функцій системи. Ці два типи файлів створюються користувачами та основними. У створюваних бібліотечних модулях системи найчастіше зустрічаються файли з розширенням *.c та *.m — коди програм, а також файли баз даних *.mex.

Структура системи MatLab містить велику кількість папок та підпапок, кожна з яких має своє призначення. Особливе значення для роботи системи має папка ./MatLab/ToolBox/MatLab. У ній зберігається вся бібліотека стандартних М-файлів системи. Їх перегляд дає змогу оцінити можливості версії, установленої на комп'ютері. Нижче перелічені основні підпапки, які містяться у каталозі MatLab/ToolBox/MatLab, та вказане призначення файлів, які у цих папках зберігаються.

- Підпапка General, де зберігаються файли, що відповідають командам MatLab. За допомогою цих команд виконуються головні системні функції:

збереження файлів, установлення та зміна робочого каталогу, функції допомоги та інші.

- Підпапки операторів, конструкцій мови програмування та системних функцій:

– ops — оператори мови програмування та спеціальні символи;
– lang — конструкції мови програмування;
– strfun — рядкові функції;
– iofun — функції введення?виведення даних;
– timefun — функції обчислення часу і дати;
– datatypes — команди опису типів та структур даних.

- Підпапки основних математичних і матричних функцій:

– elmat — команди для створення простих матриць і для роботи над ними;
– elfun — елементарні математичні функції;
– specfun — спеціальні математичні функції;
– matfun — матричні функції лінійної алгебри;
– datafun — аналіз даних у матрицях та перетворення Фур'є;
– polyfun — оброблення поліномів і функції інтерполяції;
– funfun — функції від функції, розв'язування звичайних диференціальних рівнянь;
– soarfun — оброблення вироджених матриць.

- Підпапки графічних команд:

– graph2d — команди двовимірної графіки;
– graph3d — команди тривимірної графіки;
– specgraph — команди спеціальної графіки;
– graphics — команди дескриторної графіки (графічні діаграми);
– vitools — команди для створення графічного інтерфейсу користувача.

Вміст папок можна подивитися двома способами: по-перше, він знаходиться у файлі contents.m, а по-друге, його папки можна вивести у вікно командного рядка MatLab, набравши команду help ім'я_папки, наприклад:

help datafun

Як зберегти свій проект у системі MatLab та відкрити його

Працювати з файлами у MatLab можна через опцію меню File. Якщо Ви провели відповідні розрахунки в системі MatLab, то простіше за все зберегти значення обчислених змінних у файлі з розширенням *.mat. Як зазначалось - це файли бази даних, де зберігаються значення обчислених змінних, але оскільки записані вони не в текстовому форматі, без оболонки їх прочитати не можна. Для виконання цієї операції достатньо скористатися опцією меню File → Save Work Space As. За замовчуванням MatLab усі файли записує в директорію Work до папки, де встановлена система. Можна зберігати проекти і в інших директоріях, указуючи відповідний шлях до них. Результати розрахунків можна також зберігати, використовуючи команду save. Формат команди:

save ім'я файлу, наприклад:

save fff

Розширення *.mat вказувати не треба, воно завжди встановлюється за замовчуванням. При вказаному форматі файли записуються в робочу папку Work, у робочому каталозі. За бажанням зберегти файл у іншій папці необхідно вказати шлях до неї, наприклад:

save d:/123/fff

Після імені файлу можна через пробіл вказати змінні, які необхідно запам’ятати, ідентифікатори змінних відділяють комою, наприклад:

save fff a,b,c

У разі записування команди в такому форматі будуть запам’ятовуватися тільки визначені змінні. Тобто, застосування командного рядка дає користувачу більше можливостей, ніж збереження всіх змінних через систему меню.

Після параметрів команди save через знак – можуть бути указані ключі, які уточнюють формат запису файлів:

– mat — формат баз даних, який використовується за замовчуванням;

– ascii — запис у ASCII файл з точністю до 8 цифр;

– ascii-double — запис у ASCII-файл із подвійною точністю, до 16 цифр;

– v4 — запис у форматі MatLab-4;

– append — додати визначені дані в існуючий файл.

Зберегти значення змінних ви можете, виконавши команду меню Load WorkSpace; через системну команду load, формат якої аналогічний команді save:

load ім’я файлу

Однак недосвідчені користувачі MatLab, які звикли до Windows-технологій подання інформації, коли на екрані відображається абсолютно все, що вводилося з клавіатури, будуть дуже здивовані, і можуть подумати, що вони зробили щось некоректно і дані з їхнього проекту кудись зникли. Річ в тому, що після виконання команди меню Load WorkSpace робоче вікно командного рядка залишиться незаповненим, хоча, проводячи обчислення, Ви вводили певну кількість команд. Що ж трапилося, невже відбувся якийсь збій у роботі програми? Ні, система все виконала чітко, запам’ятала всі змінні, які Ви визначили,і тепер можна до них звернутися. Пишучи a=, побачите на екрані значення змінної a, якщо змінна з таким ім’ям була визначена і запам’ятована. По суті, це є відмінністю професійних систем від систем масового використання. На екран виводиться мінімум інформації, і тільки запитувана, а інша зберігається у пам’яті комп’ютера і залишається без змін, поки користувач не надасть відповідної команди для її відображення. Таке подання інформації пов’язано також із серйозною історичною причиною. Річ у тому, що коли починав створюватися MatLab, існували тільки текстові монітори, і виведення графіки було ускладнене. Оскільки на алфавітно-цифровий монітор не можна було вивести велику кількість інформації, програмісти частіше працювали з телетайпами, виводячи на друк кожну команду та реакцію комп’ютера на неї. Уявіть собі, скільки паперу знадобилося б, щоб роздрукувати програму на 1000 рядків! Тому інтерфейс професійного програмного забезпечення був досить лаконічний, і не на кожну введену команду надходила відповідна реакція комп’ютера, наприклад, «Command completed successfully» — «Команда успішно виконана». За замовчуванням завжди передбачалося, що команда надана без помилок і комп’ютер її успішно виконав.

Але більшість користувачів швидко звикають до аскетичного інтерфейсу системи MatLab, і через певний час він починає їм подобатися, оскільки зайва інформація на екрані зазвичай не допомагає, а заважає роботі. А спочатку, щоб було простіше, намагайтеся дублювати отримані результати на папері, якщо впевнені у їхній правильності. Пізніше, коли вивчите систему команд, потребав цьому відпаде, оскільки MatLab має велику кількість команд, які дають змогу отримати будь-яку інформацію про проект, враховуючи і послідовність введених команд. Ще більше можливостей щодо роботи з даними дає мова програмування системи MatLab.

Зберігати змінні у середовищі MatLab зручніше не через систему меню, а через використання команд save та load. У цьому випадку можна запам’ятати не всі описані змінні, а тільки ті, що знадобляться для подальшого обчислення. Тоді формат команди save має вигляд: save ім’я файлу список змінних. Результатом виконання наведеної нижче послідовності команд буде збереження змінних a,b та c, але не буде збережена змінна d.

```
>>a=2;
>>b=a?3;
>>c=a*b;
>>d=b/a;
>>save fff a b c
```

ВХІДНА МОВА СИСТЕМИ MatLab ТА ПРОВЕДЕННЯ ПРОСТИХ ОБЧИСЛЕНЬ

У розділі описані основи роботи з командним рядком у системі MatLab та наведені базові поняття вхідної мови системи, такі як дійсне та комплексне число, вектор, матриця, змінна, оператор, функція та математичний вираз. Розглянуті основні засоби системи для створення арифметичних та логічних виразів із змінними різних типів.

Матриці, вектори та числа як основні поняття вхідної мови системи MatLab

Систему MatLab початково було розроблено для роботи з великими структурами даних, матрицями та векторами. Хоча на сьогодні це потужний інструмент для проведення різноманітних числових та аналітичних розрахунків, система залишається орієнтованою на роботу з матрицями та векторами, а її вхідна мова направлена саме на оброблення упорядкованих числових структур. Тому з приводу базових визначень та понять вхідної мови системи у розробників та фахівців не існує єдиної думки. Є спеціалісти, які вважають, що простіше зрозуміти вхідну мову MatLab, якщо взяти за базове поняття матриці і на його основі сформувати поняття вектора як матриці розміром (1, n) та поняття числа як матриці розміром (1, 1). Також суттєво відрізняють поняття розміру та розмірності матриці. Зрозуміло, що така теоретична структура дещо суперечить

сучасним математичним правилам, де базовим є поняття числа, але вона дійсно відповідає парадигмі мови системи MatLab. Інші фахівці роблять традиційно: вводять базове поняття числа, на його основі формують поняття вектора та матриці як упорядкованих сукупностей чисел, і вже надалі формулюють головні правила мови системи MatLab, які більше орієнтовані на роботу з матрицями та векторами, ніж зокремими числами. Такий підхід має свої переваги, оскільки постановка на комп'ютері практичних задач математичного моделювання потребує саме проведення операцій з великими структурами даних. У галузі електроніки до таких задач належать побудова математичних моделей фізичних явищ у реальних приладах (навіть у межах простих одновимірних моделей) та аналіз електронних схем та систем. Конкретні приклади таких задач та їх розв'язання з використанням засобів MatLab. Проте у більшості математичних САПР, і навіть у мовах програмування (FORTRAN, Паскаль, C, C++) не існує стандартних функцій оброблення числових масивів, зокрема пошуку мінімальних та максимальних елементів, їх сортування у порядку збільшення або зменшення, не кажучи про елементарні математичні операції з елементами двох матриць одного розміру: наприклад, множення або ділення відповідних елементів однієї матриці на елементи другої. Зазвичай у мовах програмування для виконання таких нескладних операцій з масивами даних застосовують невеликі фрагменти програм, основою яких є циклічні структури. Організувати стандартні процедури для виконання таких операцій зі структурами даних невизначеного розміру досить непросто. Розвинений у сучасних мовах програмування високого рівня апарат динамічних змінних тільки оптимізує розташування надмірних структур даних у пам'яті, але не спрощує, а у деяких випадках навіть ускладнює оброблення даних. Як результат — структурні програми, написані сучасними мовами програмування високого рівня, зазвичай перевантажені великою кількістю циклів, призначених для оброблення числових структур, що значно ускладнює написання та розуміння таких програм. Складність роботи із структурами даних у мовах програмування високого рівня призводить до алгоритмічних помилок, виявлення яких, на відміну від синтаксичних та логічних, є досить кропіткою працею та значно ускладнює роботу програмістів. Зазвичай виправлення таких помилок займає більше, ніж 90% часу створення програмного забезпечення. Через відсутність стандартних функцій та процедур оброблення числових даних у відомих мовах програмування майже вся навчальна література орієнтована на вивчення оброблення упорядкованих та ієрархічних структур даних. Навіть кількість посібників та підручників з алгоритмів чисельних методів, реалізованих мовами програмування високого рівня, вкрай обмежена. Орієнтація вхідної мови MatLab на виконання матричних операцій математичних функцій для роботи з ними дає змогу уникати використання циклічних структур, створювати досить короткі та надійні програми, призначені для розв'язання конкретних задач при наявності відповідного математичного апарату. Приклади використання MatLab для моделювання електронних приладів.

Числа, змінні та математичні вирази в системі MatLab

Таким чином, будемо дотримуватися традиційних теоретичних побудов, згідно з якими поняття числа та математичного виразу є базовими для вхідної мови MatLab.

Число — найпростіший об’єкт вхідної мови MatLab, який представляє числові дані. Можливі формати подання дійсних чисел: ціле, дробове, з фіксованою комою, із плаваючою комою. Ці формати досить прості та відповідають правилам записування чисел у більшості мов програмування. Нижче наведені приклади різних числових форматів:

```
2
-3
213
2.301
0.0001
1e-4
1e10
```

При записуванні цифр одного числа пробіли між ними не допускаються.

Просто в системі MatLab записуються і комплексні числа. Вони подаються у стандартному вигляді Re(z) + i*Im(z) або Re(z) + j*Im(z).

Приклади записування комплексних чисел:

```
3+5*i
i
-8*i
-1
j
i
```

Для роботи зкомплексними числами у MatLab існує ряд спеціальних функцій:

real (z) — повертає дійсну частину числа;

imag (z) — повертає уявну частину числа;

abs (z)— повертає модуль дійсного або комплексного числа;

angle (z) — повертає кут нахилу між віссю абсцис і вектором комплексного числа у радіанах;

conj (z) — функція повертає число, комплексно-спряжене до числа z, тобто якщо z = a + ib,

Із чисел легко складаються математичні вирази, у яких можна використати знаки арифметичних операцій +, –, *, / і весь набір математичних функцій системи.

Розглянемо роботу з виразами у системі MatLab у режимі прямих обчислень, тобто, без написання програми та запам’ятовування командних рядків. Робота із системою у такому режимі нагадує використання дуже потужного наукового калькулятора, який має велику кількість елементарних та спеціальних математичних функцій. Зрозуміло, можливості такого калькулятора значно більші, ніж у звичайних, як за обчислювальними потужностями, так і за графічними, ураховуючи двовимірну та тривимірну наукову та ділову графіку. Робота в режимі прямих обчислень носить діалоговий характер за принципом

«питання-відповідь». Користувач повинен набрати у командному рядку математичний вираз, за потреби відредагувати його, переміщуючи курсор, і завершити введення виразу натисненням клавіші «Enter».

```
z
a ib
Im
Im(z)
Re(z)
|z|
-z
0
Re
-(z)
```

Сформулюємо основні правила формування та введення математичних виразів у системі MatLab.

1. Для індикації можливості введення вихідних даних новий командний рядок починається із символа >> (два знаки «більше»).

2. Для того, щоб не виводити результати обчислення виразу, введеного в даному рядку на екран після останнього символу необхідно поставити символ ; (крапка з комою).

3. Будь-якому виразу може присвоюватися значення константи з відповідним ім’ям, і надалі це ім’я може використовуватися в інших математичних виразах. Якщо значенню виразу не присвоєна ніяка змінна, за замовчуванням використовується змінна ans. Наведемо приклади роботи з числовими виразами у системі MatLab:

```
>> 2+3
ans=
5
>> sin(1)
ans=
0.8415
```

Усі командні рядки, які вводить користувач, починаються із символів >>, а в рядках, які формуються у результаті відповіді системи на надані команди, на відміну від командних рядків таких символів немає.

4. Як і в мовах програмування FORTRAN та C, оператору присвоєння в математичних виразах мови MatLab відповідає знак =.

5. Під час написання імен змінних та функцій система MatLab, як і компілятори мов програмування C та C++, розрізняє малі та великі літери. Бібліотечні функції системи (у нашому прикладі це функція сінусу sin) записуються малими літерами. Аргументи функцій указуються в круглих дужках.

6. Результати обчислень виводяться в окремих рядках, що відрізняються від рядків з іменами змінних тим, що у них нестоїть що знак =.

7. У випадку використання математичних виразів ізвеликою кількістю символів, коли для запису не вистачає одного рядка, перехід на новий рядок здійснюється за допомогою трьох крапок (…). Наприклад:

```
>> S=1?1/2+1/3?1/4+1/5?1/6+1/7?...
```

```
1/8+1/9?1/10+1/11?1/12
S=
0.6532
>>
```

Розривання рядків у MatLab необхідне тільки для виведення документів на друк, оскільки максимальна кількість символів у командному рядку може досягати 4096, а для програм, написаних у форматі m-файлів, вона взагалі не обмежена.

8. Під час введення чисел необхідно дотримуватися форматів їх записування.

Системні константи та змінні у системі MatLab

Константа MatLab — це унікальне ім’я, якому надають значення введеного виразу. У цьому разі принцип унікальності повинен зберігатися для всіх командних рядків сесії роботи із системою, для відповідного проекту, який може бути збереженим та завантаженим за допомогою методів. Звичайні числа, наприклад, 1, –2, 5.78 можуть розглядатися як безіменні числові константи, за замовчуванням надається значення константи ans. Можна описувати тип константи,використовуючи команду format, але у більшості випадків система сама коректно визначає його.

Крім того, у MatLab існує інший тип констант, які прийнято називати системними змінними. Їм за замовчуванням система надає відповідне значення, яке може змінюватися за допомогою оператора присвоювання. Змінні і та j, які описують уявну одиницю. Наведемо інші системні змінні мови MatLab:

pi — відповідає числу π = 3,1415926…;

eps— похибка обчислювальних операцій над числами із плаваючою крапкою, за замовчуванням eps = 2^{-52};

realmin — мінімальне можливе число із плаваючою крапкою, яке може бути отримане у результаті обчислень, за замовчуванням;

realmin= 2.225073858507201e-308;

realmax— максимально можливе число з плаваючою крапкою, яке може бути отримане у результаті обчислень, за замовчуванням;

realmax= 1.797693134862316e+308;

inf — значення машинної нескінченності.

Остання системна змінна виводиться у тому випадку, коли результат обчислень перевищує значення realmax, або при некоректних математичних операціях, наприклад, при діленні константи на 0. Ця операція також супроводжується повідомленням системи: «Warning: Divide by zero» («Попередження: Ділення на нуль»). Наприклад:

```
>> 5/0
Warning: Divide by zero.
ans =
Inf
>>
```

Тут треба звернути увагу на те, що система виконує цю некоректну з математичної точки зору операцію, і видає не помилку введення команди, а попередження. Загалом для мови MatLab математична нескінченність — такий самий коректний результат, як і числовий. Введення розробниками системи MatLab поняття математичної нескінченності спрощує розв’язання математичних задач, оскільки під час проведення обчислень немає потреби у попередньому аналізі використаних функцій на наявність розривів другого роду, система здійснює цю операцію сама. Поняття математичної нескінченності суттєво спрощує візуалізацію функцій із розривами. А як відомо, саме пошук точок розриву у функціональних залежностях та оброблення цих особливих точок є однією з головних задач математичного оброблення упорядкованих структур даних. Зазвичай при розв’язанні на комп’ютері складних математичних задач засобами мов програмування програмісти витрачають близько 90% часу на пошук особливих точок використаних функцій для уникнення обчислювальних помилок під час роботи програми. У разі розв’язання задач математичного моделювання у системі MatLab здебільшого необхідність попереднього аналізу використовуваних функцій на наявність точок розриву повністю відпадає.

NaN — показник невизначеності даних при аналітичних і арифметичних обчисленнях, наприклад, при виникненні математичної невизначеності типу 0/0. Слід зазначити, що результатом обчислень у MatLab може бути не тільки дійсне, але й комплексне число. Тому при обчисленні таких математичних виразів, як arcsin (2) або ln (–1) система видає відповідне комплексне число, а не виводить системну змінну NaN.

Нижче наведено приклад кількох команд, де використовуються системні змінні:

```
>>2*pi
ans=
6.2832
>>eps
ans=
2.0e-052
>>realmin
ans=
2.225e?308
>>realmax
ans=
1.7977e+308
>> 1/0
Warning: Divide by zero
ans=
Inf
>>0/0
ans=
NaN
>>log(0)
Warning: Log of zero.
```

```
ans =
?Inf
>>ans =
-Inf
>>log(?1)
ans =
0 + 3.14159265358979i
>>asin(-5)
ans =
?1.57079632679490 + 2.29243166956118i
>>
```

Під час введення команд системні змінні можуть перевизначатися, наприклад:

```
>> i=1;
>> eps=0.001;
```

Однак, оскільки значення цих змінних встановлюються при завантаженні системи, вони ніколи не бувають невизначеними.

У MatLab є також символьні константи, які являють собою ланцюжок будь-яких символів, які беруться в лапки. Наприклад:

```
“Hello, friends!“
“Привіт, друзі!“
“2+3“
```

Оскільки вираз “2+3“ може бути розглянутий як математичний, MatLab має спеціальні функції для перетворення символоподібних виразів у числові.

Текстові коментарі в програмах, написаних мовою MatLab

Зазвичай систему MatLab використовують для проведення складних науково?технічних розрахунків, застосовуються бібліотечні функції системи, реалізовані в інших файлах. Розбиратися у таких програмах, які мають складну модульну структуру, досить непросто, тому значну роль під час їх написання та використання відіграють текстові коментарі. Текстовими коментарями у програмуванні називають будь-яку послідовність символів, які ставляться після виконуваних команд або окремим рядком та завжди починаються з відповідного ключового символу. У програмах, написаних мовою MatLab, текстові коментарі займають окремий рядок, який завжди починається із символа %.

Наприклад:

```
% Обчислення функції Бесселя.
```

Текстові коментарі ніяк не впливають на виконання програми, але в них завжди описується призначення тих або інших рядків, тому вони значно полегшують роботу програмістів та користувачів під час написання та застосування програм. Бібліотечні функції MatLab реалізовані відкритим програмним кодом, що полегшує роботу фахівців зцією досить складною системою, та у значній мірі сприяє її постійному розвитку та удосконаленню. Програмістам дуже зручно і просто на базі наявних розробляти нові бібліотеки, і

тим самим вони удосконалюють та розвивають набір функцій цієї системи. Крім того, у перших рядках m-файлів, написаних мовою системи MatLab, описується призначення цих файлів у вигляді коментарів, і саме ця інформація виводиться на екран при введенні команди

```
>> help ім'я_файлу
```

Тому розвинені коментарі в текстах програм, написаних мовою системи MatLab, вважаються просто «правилом доброго стилю програмування». У цьому MatLab дуже нагадує операційну систему UNIX, під якою його початково створювали та відлагоджували.

На жаль уверсії MatLab 6.0 є один серйозний недолік. Код символу української букви «с» система сприймає як перехід на інший рядок і використовувати у коментарях його неможливо. Ця помилка досі не виправлена, і для її уникнення рекомендується використовувати латинський символ «c», хоча це не дуже зручно.

Змінні у системі MatLab та робота з ними

У мові MatLab, як і в мовах програмування, змінні— це об’єкти, яким присвоюються відповідні імена та які призначені для збереження відповідних даних, значення змінних, на відміну від констант, змінюються під час виконання програми. Залежно від типів даних змінні бувають числовими, символьними, векторними або матричними. Досконаліше типи даних та їх класифікація будуть розглянуті при описанні засобів програмування системи MatLab.

Змінним можна задавати відповідні значення за допомогою виразів, використовуючи знак = як оператор присвоєння. Формат виразу з оператором присвоєння такий:

ім’я_змінної = вираз

Для простоти синтаксису тип змінних заздалегідь не декларується. Вони визначаються типом виразу, значення якого присвоюється відповідній змінній. Наприклад, якщо результатом обчислення введеного виразу є матриця, відповідна змінна також буде типу матриця.

Правила написання імен змінних у системі MatLab досить прості. Ім’я змінної, або ідентифікатор, може містити будь-яку кількість символів, але запам’ятовуються та ідентифікуються з них тільки перший 31 символ. У цьому разі повинен виконуватися принцип унікальності, тобто, ім’я змінної не може збігатися з іменами інших змінних або функцій системи. Ім’я повинне починатися з літери і може містити літери, цифри або символ підкреслення (_). Неприпустимо ставити в імена змінних символи +, –, *, /, &, :, @, =, оскільки у цьому випадку правильна інтерпретація математичного виразу стає неможливою.

Є можливість вилучення імен змінних для очищення пам’яті та для підвищення ефективності роботи програми. У пам’яті комп’ютера дані користувача займають відповідне місце, яке називається робочою областю. За великої кількості змінних необхідно очищувати робочу область, вилучивши ті змінні, які для подальшої роботи не потрібні. Особливо важливо це робити, якщо такі змінні описують надмірний вектор або матрицю, і, відповідно, займають

великий обсяг оперативної пам’яті. Як зазначалося у розділі 1.5.1, переглянути робочу область програми можна за допомогою команди Меню File → Show Workspace. Одна числова змінна займає у пам’яті комп’ютера 8 байт незалежно від використовуваного формату виведення чисел на екран. З одного боку, це зручно для користувачів з точки зору простоти роботи зданими, але впливати на ефективність їх розташування у пам’яті комп’ютера через визначення типів, як це зроблено у мовах програмування, програміст не може. Тому очищення робочої області через вилучення змінних є єдиною можливістю ефективного використання оперативної пам’яті комп’ютера.

Очистити робочу область можна через систему меню, якщо виконати команду File → Show Workspace, виділити за допомогою миші у вікні Workspace Browser відповідні змінні та натиснути кнопку Delete, Після виконання цієї операції значення раніше визначеної змінної a стає невизначеним, тому за спроб користувача вивести це значення на екран система видає повідомлення «Undefined function or variable «a».» («Невизначена функція або змінна a»):

```
>> a
??? Undefined function or variable «a».
>>
```

Іншим ефективним способом очищення робочої області є команда MatLab clear у різних її модифікаціях. Можливі формати команди clear:

clear — вилучення всіх описаних раніше змінних;

clear x — вилучення змінної x;

clear a,b,c — вилучення змінних a, b та c;

clear functions — вилучення функцій;

clear all — вилучення всіх змінних та функцій.

Наведемо приклад програми, де змінні спочатку визначаються, а потім вилучаються. Цей фрагмент містить також вирази зі змінними типу матриця та вектор, які будуть розглянуті у наступному параграфі.

```
>> x=2*pi
x=
6,2832
>>v=[1,2,3,4]
v=
1 2 3 4
>> MAT
???Underfined function or variable “MAT”
>>MAT=[1,2;3,4]
>>MAT=
1 2
3 4
>>clear v
>>v
??? Underfined function or variable “v”
>>clear
>>x
??? Underfined function or variable “x”
```

```
>>MAT
???Underfined function or variable “MAT”
```

При очищенні робочої області та завантаженні нових змінних у оперативній пам’яті комп’ютера створюється велика кількість неупорядкованих вільних областей, які межують із завантаженими. Така організація пам’яті не заважає роботі із системою, якщо вона працює в одно-задачному режимі, але у цьому разі втрачається ефективність Windows як багато-задачної операційної системи.

Щоб уникнути зайвих витрат пам’яті при роботі з великими масивами інформації, крім команди clear слід використовувати також команду pack. Ця команда здійснює перерозподіл даних у робочій області оперативної пам’яті, послідовно заповнюючи порожні місця. У цьому разі зменшується загальний обсяг пам’яті, який система відводить під змінні користувача. Крім того, команда pack сприяє ефективному використанню віртуальної пам’яті, записуючи змінні, до яких користувач рідко звертаєтеся, на жорсткий диск. Таким чином, при сумісному використанні команд clear і packможна значною мірою оптимізувати роботу системи, зокрема у багатозадачному режимі. Команда pack завжди виконує описану стандартну системну функцію і використовується без параметрів.

Використання апарату змінних у мові системи MatLab є дещо іншим та у деяких випадках набагато ефективнішим, ніж у мовах програмування. Оскільки система MatLab має розвинені бібліотеки для виконання операцій з матрицями та векторами, у багатьох випадках значно простіше визначити векторні або матричні константи та використовувати їх у математичних виразах, ніж виконувати ті самі операції через числові змінні. Наприклад, такі нескладні, але важливі з практичної точки зору задачі, як табуляція значень функції або множення елементів однієї матриці на відповідні елементи іншої, засобами MatLab вирішуються дуже просто через матричні та векторні операції, і відповідні програми займають кілька рядків. Для розв’язання аналогічних задач засобами стандартного програмування необхідно складати циклічні структури, і їх велика кількість значно ускладнює написання програми.

Функція вилучення змінних у системі MatLab деякою мірою відповідає апарату динамічних змінних усучасних мовах програмування, але використовувати її значно простіше.

Визначення матриць та векторів

Описання матриць та векторів здійснюється через визначення змінних. У цьому разі використовують три прості правила.

1. Усі елементи матриці або вектора необхідно брати у квадратні дужки.

2. Між числами, які відповідають елементам вектора або рядка матриці, можна ставити пробіл або кому.

3. Між рядками матриці ставлять крапку з комою.

Приклади визначення матриць та векторів:

```
>> v=[1, 2, 3, 4]
v=
```

```
1 2 3 4
>> MAT =[1,2;3,4]
M=
1 2
3 4
>>
```

Нескладно також звернутися до елементів вектора або матриці для виведення на екран або використання у виразах. Для цього достатньо вказати в круглих дужках порядковий номер елемента вектора або номер рядка та стовпця матриці через кому. Ці правила запису цілком відповідають відповідним правилам у математиці. Продовжуючи попередній приклад, розглянемо таку послідовність командних рядків:

```
>> v(1)=
1
>> M(1,1)=
1
>>
```

Найважливіше, що є у системі MatLab і чого немає в інших системах програмування та математичних САПР — це можливість виконання арифметичних дій із матрицями та векторами та їх використання як параметрів математичних функцій. У цьому разі обчислення проводяться з кожним елементом матриці бо вектора. З основ програмування відомо, що в кожній з мов програмування запис типу sin(v) або exp(M) , де M — матриця, v — вектор, був би некоректним. Щоб виконати будь-яку математичну дію з елементами вектора або матриці, потрібно організувати цикл. Обчислення функції від всіх елементів вектора або матриці є можливим у системі MathCAD, але там це робиться через спеціальну функцію векторизації. З точки зору точного математичного опису в цьому є певна рація, оскільки у математиці такі операції заборонені. Але під час розробки реальних програм найчастіше необхідно обробляти масиви великих чисел саме таким чином, наприклад, поелементно додавати, множити поелементно або на константу, обчислювати функції від елементів масиву. А такі дії, як множення матриць або векторний добуток двох векторів, потрібні програмісту значно рідше, тому їх можна реалізувати через внутрішні функції системи. Розробники MatLab виходили, ймовірно, саме із цих міркувань, адже система створювалася для професіоналів?програмістів. Тому у MatLab використані записи типу exp(M) або sin(v) як поелементні дії з матрицями та векторами. Адже програмісту зручно працювати саме так, хоча такий запис неправильний з математичної точки зору. До того ж, такий підхід відповідає всій ідеології системи, тому що вона розроблена для роботи з матрицями, і вони розглядаються як звичайні числа. А спеціальні матричні та векторні операції тут також введені, але через окремі функції.

Тому множення, ділення, піднесення у степінь умові MatLab реалізовані як за правилами лінійної алгебри, так і для поелементних операцій, але запис цих функцій різний. Якщо визначені дії слід виконувати поелементно, використовуються оператори .* для поелементного множення, ./ для по

елементного ділення та .^ для поелементного піднесення у ступінь, у цьому разі розмірність масивів для поелементних операцій має бути однаковою. При використанні операторів */ та ^математичні дії виконуються за правилами лінійної алгебри. У випадку множення вектора або матриці на числову константу по елементні операції повністю збігаються з операціями лінійної алгебри.

Дії додавання та віднімання для матриць або векторів та числових констант мають єдине значення. Під час додавання або віднімання двох матриць або векторів ці операції також виконуються поелементно, у цьому разі для уникнення помилок програмування розмірності векторів або матриць повинні бути однаковими. Наведемо приклад використання по елементних операцій та операцій лінійної алгебри мовою MatLab.

Приклад

```
>> format short
>> v=[1,2,3,4];
>> sin(v)
ans=
0,8415 0,9093 0.1411 ?0,7568
82
83
>> 3*v
ans=
3  6  9  12
>> v^2
???Error using= =>^Matrix must be square.
>> v.^2
ans=
1  4  9  16
>> v+2
ans=
3  4  5  6
MAT=[1,2;3,4]
MAT =
1     2
3     4
>> sin (MAT)
ans =
0.8415    0.9093
0.1411   ?0.7568
>> MAT^2
ans =
7    10
15    22
>> MAT.^2
ans =
1     4
9    16
```

```
>> 3*MAT
ans =
3     6
9    12
>> 2+MAT
ans =
3     4
5     6
>>
```

У наведеному прикладі повідомлення ???Error using =>^ Matrix must be square (Помилкове використання оператора ^. Повинна бути квадратна матриця) означає, що використання операції множення векторів за правилами лінійної алгебри неможливе, можна множити одну на одну тільки квадратні матриці. Як бачимо, для квадратної матриці система дійсно не видає помилку і виконує операцію множення матриці саму на себе за правилами лінійної алгебри, але результат суттєво відрізняється від поелементного множення. Усі результати поелементних операцій для вектора v та матриці MAT однакові оскільки містять ті самі елементи.

Головні команди текстового редактора системи MatLab

Ctrl+B — переміщення курсору вправо на один символ;
Ctrl+F — переміщення курсору вліво на один символ;
Ctrl+R — переміщення курсору вправо на одне слово;
Ctrl+L — переміщення курсору вліво на одне слово;
Ctrl+A — переміщення курсору на початок рядка;
Ctrl+E — переміщення курсору на кінець рядка.
↑ та ↓ або Ctrl+P та Ctrl+n — перегляд попередніх команд для підстановки в рядок. Ці операції стандартні для системи UNIX і є аналогом вікна History у багатовіконній версії MatLab 6.0.
Del або Ctrl+D — вилучення символа, що знаходиться праворуч від курсору;
BackSpace або Ctrl+H — вилучення символа, що знаходиться ліворуч від курсору;
Ctrl+K — вилучення символів від поточного до кінця рядка.
Esc — очищення введеного рядка.

Оператори системи MatLab

Узагальнена класифікація операторів системи MatLab

Оператор — це спеціальне позначення для конкретної математичної дії зі змінними та числовими константами, які називаються операндами. Найпростішими математичними операторами є знаки суми (+), різниці (–), множення (*), ділення (/) та піднесення у ступінь (^). У попередньому розділі були розділені оператори матричної алгебри та поелементні оператори, які в англійській літературі згідно з правилами їх написання називають точковими операторами (англійський термін dot-ope-rator). Оператори використовують у математичних виразах з операндами, наприклад 2 + 3 або a + b.

Для отримання відомостей по всім операторам функцією допомоги, набравши команду help ops. Результат роботи цієї команди займає кілька сторінок і має такий вигляд:

```
help ops
Operators and special characters.
Arithmetic operators.
plus
– Plus
+
uplus
– Unary plus
+
minus
– Minus
```

?
uminus
— Unary minus
?
mtimes
— Matrix multiply
*
times
— Array multiply
.*
mpower
— Matrix power
^ power
— Array power
.^ mldivide — Backslash or left matrix divide \
mrdivide — Slash or right matrix divide
/
ldivide
— Left array divide
.\
rdivide
— Right array divide
./
kron
— Kronecker tensor product
kron
Relational operators.
eq
— Equal
= =
ne
— Not equal
~ =
lt
— Less than
<
gt
— Greater than
>
le
— Less than or equal
< =
ge
— Greater than or equal
> =
Logical operators.

and
— Logical AND
&
or
— Logical OR
|
not
— Logical NOT
~
xor
— Logical EXCLUSIVE OR
any
— True if any element of vector is
nonzero
all
— True if all elements of vector
are nonzero
Special characters.
colon
— Colon
:
paren
— Parentheses and subscripting ( )
paren
— Brackets
[ ]
paren
— Braces and subscripting
{ }
punct
— Decimal point
.
punct
— Structure field access
.
punct
— Parent directory
..
punct
— Continuation
...
punct
— Separator
,
punct
— Semicolon

;
punct
— Comment %
punct
— Invoke operating system command !
punct
— Assignment
=
punct
— Quote
‘
transpose — Transpose
.’
ctranspose— Complex conjugate transpose
’
horzcat
— Horizontal concatenation
[,]
vertcat
— Vertical concatenation
[;]
subsasgn — Subscripted assignment ( ),{ },.
subsref
— Subscripted reference ( ),{ },.
subsindex — Subscript index
Bitwise operators.
bitand
— Bit?wise AND.
bitcmp
— Complement bits.
bitor
— Bit?wise OR.
bitmax
— Maximum floating point integer.
bitxor
— Bit?wise XOR.
bitset
— Set bit.
bitget
— Get bit.
bitshift — Bit?wise shift.
Set operators.
union
— Set union.
unique
— Set unique.

```
intersect — Set intersection.
setdiff
— Set difference.
setxor
— Set exclusive?or.
ismember — True for set member.
See also ARITH, RELOP, SLASH.
>>
```

Кількість операторів досить велика, але вони систематизовані за розділами та за своїм функціональним призначенням.

Окремо виділяють

арифметичні оператори (Arithmetic operators),

оператори математичних співвідношень (Relational operators),

логічні оператори (Logical operators),

спеціальні символи (Special characters),

побітові оператори (Bitwise operators)

оператори для роботи із множинами (Set operators).

Важливим є те, що хоча команда help ops видає тільки список операторів та короткі узагальнені фрази щодо їх призначення, але надано й посилання на розділи ARITH, RELOP та SLASH, де можна знайти докладний опис операторів, які у діях системи MatLab мають відповідне значення. Наведений приклад використання «системи допомоги» показує, що вона досить розвинена, і через використання команди help можна дійсно вивчити основи роботи із системою. Враховуючи безліч функцій різного призначення, у багатьох випадках такий шлях вивчення можливостей системи є найпростішим,і надалі ми будемо ефективно його використовувати. Але для того, щоб робота із системою допомоги MatLab була ефективною, треба знати англійську мову.

Слід зазначити, що, як і у звичайній арифметиці, у мові MatLab оператори мають відповідний пріоритет виконання. Так, пріоритет логічних операторів вищий, ніж арифметичних. у математиці, пріоритет піднесення у ступінь вищий, ніж операторів множення і ділення, а пріоритет множення та ділення вищий за пріоритет операторів додавання та віднімання. Для зміни пріоритетів виконання у математичних виразах, використовуються круглі дужки.

Арифметичні оператори

Переглянемо призначення основних математичних операторів та правила роботи з ними за допомогою команди help ARITH.

```
>> help ARITH
Arithmetic operators.
+
Plus.
X + Y adds matrices X and Y.  X and Y
must have the same dimensions unless one
is a scalar (a 1?by?1 matrix).
```

```
A scalar can be added to anything.
? Minus.
X ? Y subtracts matrix X from Y.  X and Y
must have the same dimensions unless one
is a scalar.  A scalar can be subtracted
from anything.
*  Matrix multiplication.
X*Y is the matrix product of X and Y.
Any scalar (a 1?by?1 matrix) may multiply
anything.  Otherwise, the number of
columns of X must equal the number of
rows of Y.
.* Array multiplication
X.*Y denotes element?by?element
multiplication.  X and Y must have the
same dimensions unless one is a scalar.
A scalar can be multiplied into anything.
^  Matrix power.
Z = X^y is X to the y power if y is
a scalar and X is square. If y is an
integer greater than one, the power is
computed by repeated multiplication. For
other values of y the calculation
involves eigenvalues and eigenvectors.
Z = x^Y is x to the Y power, if Y is a
square matrix and x is a scalar, computed
using eigenvalues and eigenvectors.
Z = X^Y, where both X and Y are matrices,
is an error.
.Ârray power.
Z = X.^Y denotes element?by?element
powers.  X and Y must have the same
dimensions unless one is a scalar.
A scalar can operate into anything.
>>
```

Тут викладено зміст математичних операцій з матрицями, тобто, матричні та поелементні операції для додавання, віднімання, множення,піднесення у ступінь. Поелементні операції в цьому описі розглядаються як дії з масивами, на відміну від матричних операцій, які визначаються як дії з матрицями. Програмісти часто називають матриці масивами, але терміни матричні операції та поелементні операції більш точні з математичної точки зору.

Серед математичних дій до окремого розділу SLASH належать оператори прямого та зворотного ділення матриць. Пряме ділення, або ділення зліва направо, — це звичайне матричне ділення за відсутності крапки, або поелементне ділення за її наявності. Матричне ділення реалізується за правилами лінійної алгебри таким чином: якщо C=A*B, то A=C/B або B=C/A, де A, B, C — матриці

відповідного розміру. Поелементне ділення можливе тільки для матриць або векторів однакового розміру. Як і всі поелементні дії, воно реалізується через оператор . / .

Матричне зворотнє ділення, або ділення справа наліво — це ділення елементів другої матриці виразу на елементи першої, тобто X/Y=Y\X. Пряме та зворотне ділення ефективно використовується для розв’язання систем лінійних рівнянь. Таким самим чином, але як поелементна операція, визначається поелементне зворотне ділення.

Оператори + та — при роботі з матрицями та векторами можуть також бути використані як унарний плюс та унарний мінус. У цьому випадку знаки + та – ставлять безпосередньо перед ідентифікатором матриці, наприклад: +M або –M, де M —матриця. Ці оператори працюють таким чином: опреатор –M змінює знаки всіх елементів матриці на зворотні, після чого можна через оператор +M повернути всім елементам початкове значення. Таким чином, згідно з командою help ops, покажемо, що кожному оператору відповідає функція, що виконує ту саму дію. При наведенні синтаксису операторів та функцій матриці позначено літерою M, а числові константи — літерою n.

Наведемо приклад арифметичних дій з векторами. У прикладі системна функція disp призначена для виведення значення виразу на екран. Більш досконало системні функції MatLab. Функція rot90, використана у наведеному прикладі, виконує операцію транспортування матриці. Ця функція використовується для перстворення вектора-рядка на вектор-стовпчик.

Приклад

```
>> x=[1,2,3,4,5]; y=[?2,1,4,0,5];
>> x+2
ans=
3  4  5  6  7
>> disp(x+2)
3  4  5  6  7
>> x+y
ans=
?2  2  12  0  25
90
```

Оператори математичних співвідношень

Як відомо, у математиці та програмуванні при записуванні математичних виразів, крім операторів математичних дій, значну роль відіграють оператори співвідношень. Саме через ці оператори створюються нетривіальні алгоритми реальних математичних задач з розгалуженнями та циклами. Усі оператори математичних співвідношень мають два операнди, числові значення яких порівнюються між собою з використанням шести

операцій порівняння: менше, більше, дорівнює, не дорівнює, менше або дорівнює, більше або дорівнює. У сучасних мовах програмування результати виконання дій порівняння можуть мати два значення булевих змінних: True —

істина, та False —хибність. Система MatLab більше орієнтована на роботу з числами, ніж із математичною логікою, тому тут прийнято числове подання результатів порівнянь, яке також широко використовується у програмуванні та у математиці: True –1, та False— 0. Оператори порівняння MatLab схожі з їх реалізацією мовами програмування С та BASIC. У MatLab ті самі дії порівняння можна виконувати і через функції, які аналогічні реалізованим мовами програмування BASIC та FORTRAN. Оператори та функції математичних співвідношень наведені у табл. 2.2, де через x та y позначені математичні вирази.

Якщо для виразів з дійсними числами оператори порівняння не мають ніяких особливостей, то при використанні виразів з комплексними числами застосування операторів порівняння в системі MatLab має свої особливості. Зрозуміло, що операції дорівнює або не дорівнює цілком коректно визначені для них, а операції більше та менше для комплексних чисел не визначаються. Тому розробники системи при введенні операторів порівняння для виразів із комплексними числами визначили всі вищезгаданої дії, але працюють вони по?різному. Оператори < , > , < = , > = можуть бути застосовані з комплексними операндами, але тільки для порівняння їх дійсної частини, уявна частина операндів відкидається. А в разі використання операторів порівняння = = та ~ = порівнюються як дійсні, так і їх частини. Для порівняння модуля двох комплексних чисел можна використовувати функцію модуля abs. Нижче наведено приклад використання операцій порівняння, який показує,що при записуванні уявної частини комплексних чисел між числом та константою і не ставити знак множення.

```
Приклад
>>  eq(5,10)
ans =
0
>>  5==5
ans =
1
>>  ne (5,10)
ans =
1
>>  5~=5
ans =
0
» 5>10
ans =
0
>>  ge(10,5)
ans =
1
>> (2+3i)==(2+i)
ans =
0
>> (2+3i)>=(2+i)
```

```
ans =
1
>> abs(2+3i)>=abs(2+i)
ans =
1
>> (2+3i)~=(2+i)
ans =
1
>>
```

Важливо, що оператори порівняння можуть бути використанні для масивів, векторів та текстових констант, тобто, спектр використання цих операторів значно ширший, ніж у класичних мовах програмування. Можна порівняти масив із числовою константою. Результатом порівняння буде масив, розмірність якого відповідає розмірності початкового масиву, а всі елементи дорівнюють нулю або одиниці, залежно від результатів їх порівняння з визначеною константою. Можливим є також присвоєння значення результату виконання операції порівняння іншому масиву через математичні вирази, у яких оператори порівняння використовуються разом з операторами присвоєння.

Формат такого виразу:

```
>> M2=(M1>n)
```

Через M2 та M1 тут позначені масиви, а літерою n — числова константа. Такі вирази дуже нагадують програмістам синтаксис мови програмування С. Наведемо приклад використання операторів співвідношень:

```
>> M1=[1,2;3,4]
M1=
1     2
3     4
>>M2=(M1>1)
M2 =
0     1
1     1
>> M1>=1
ans =
1     1
1     1
>> M1~=1
ans =
0     1
1     1
>>
```

У MatLab є можливість використання операторів співвідношень з текстовими константами, єдиною умовою коректності цієї операції є однакова кількість символів у константах, які порівнюються. Результат порівняння двох символів відповідає значенням їх ASCII?кодів, а для двох рядкових констант буде вектор, розмірність якого відповідає довжині рядка, а елементи дорівнюють нулю або одиниці залежно від символів, які порівнюють. При порівнянні векторів значення

результату можна присвоїти векторній константі або змінній. Розглянемо застосування операторів порівняння у рядкових констант на прикладі:

```
>> ‘b’>‘a’
ans =
1
94
95
>> ‘b’~=‘a’
ans =
1
>>  ‘ba’~=‘aa’
ans =
1     0
>>  ‘bac’>‘aaa’
ans =
1     0     1
>>  c=(‘bac’~=‘aaa’)
c =
1     0     1
>> c=(‘bac’=‘aaa’)
??? c=(‘bac’=
A closing right parenthesis is missing.
Check for a missing «)» or a missing operator .
>> c=(‘bac’==‘aaa’)
c =
0     1     0
>>
```

Як можна побачити з наведеного прикладу, у MatLab, на відміну від мов програмування C та C++, оператори присвоєння і порівняння чітко розрізняються, і в разі їх неправильного використання система видає синтаксичну помилку.

Логічні оператори

Значну роль у системі MatLab відіграють логічні оператори та функції. Під час розв’язання задач з електроніки їх можна ефективно використовувати для аналізу цифрових схем. Але особливо велике значення мають логічні дії при написанні програми мовою MatLab, які будуть розглянуті у розділі 4. Особливість виконання логічних операцій із числовими даними полягає у тому, що логічній одиниці відповідає будь?яке число, крім нуля. При виконанні логічних операцій над рядками, як і для операцій порівняння, усі символи рядка подаються через їх ASCII коди.

Функції any та all призначені для оброблення векторів. Якщо аргументом функції є матриця, операція за замовчуванням проводиться з кожним стовпчиком матриці, результат — вектор. Використовуючи формат цих функцій з двома аргументами, одним з яких є число n, можна обрати напрям обробки матриці.

Значенню n=1 відповідає обробка рядків матриці (перший напрям), а значенню n=2 — оброблення стовпчиків (другий напрям). Команди можуть використовуватися як для окремих операндів, так і для математичних виразів, що дуже важливо при реалізації функцій програмування. Формат логічних команд під час роботи з виразами такий:

(вираз) оператор (вираз)

або

функція ((вираз), (вираз))

Наведемо приклад використання логічних операторів.

Приклад

```
>> M1=[1,2,3]
M1 =
1     2     3
>> M2=[0,2,0]
M2 =
0     2     0
>> and(M1,M2)
ans =
0     1     0
>> or(M1,M2)
ans =
1     1     1
>> M1&M2
ans =
```

Спеціальні символи та їх використання

Мова MatLab, у якій значною мірою запозичений синтаксис мови С, загалом побудована на використанні символів та символьної логіки. Тому перейдемо до вивчення спеціальних символів, яких у цій мові не бракує, і розробники систематизували їх за категоріями. Основні категорії операторів — це punct (Punctuation, пунктуація), та paren (Parentheses, дужки), але деякі оператори мають окрему категорію. Серед них дуже важливим є двокрапка :, який розробники віднесли до категорії colomn (двокрапка). У наведеній таблиці 2.4 перелік усіх спеціальних символів MatLab. Дії, призначені для виконання спеціальних функцій над матрицями та масивами числових даних. У наведеній таблиці літерою n позначені числові константи, M — матриці, O — оператори, W — вирази, FL — поля структур даних.

Більшість спеціальних операторів ми були розглянули при описанні відповідних операцій оброблення числових структур даних.

Особливе значення у вхідній мові системи MatLab при формуванні векторів має оператор двокрапка (:). За допомогою цього оператора формуються вектори з табульованими значеннями. Тобто, елементи вектора складають арифметичну прогресію з відповідним кроком, і значення кожного наступного елемента відрізняється від попереднього на постійне число.

Вектори з табульованими значеннями необхідні програмістам для формування таблиць математичних функцій та побудови їх графіків. Як указано в табл. 2.4, формат запису оператора визначення векторів з табульованими значеннями має вигляд:

>> n1:n2:n3

де n1 — значення першого елемента вектора, n2 — крок зміни елементів вектора, n3— границя, яку не повинен перевищувати останній елемент вектора при додатному кроці, або менше якого він не повинен бути при від'ємному кроці. У наведеному прикладі параметр n2 може бути пропущеним, тоді вважається що «за замовчуванням» n2=1. Користувачам, які знайомі із вхідною мовою MathCAD, треба звернути увагу на те, що нумерація елементів векторів у MatLab починається з одиниці, а не з нуля.

Це відповідає правилам формування векторів у математиці.

Якщо при написанні наведеного виразу n3<n1 при додатному кроці або n3>n1при від'ємному кроці, то система формує порожній вектор, який не містить жодного елемента, та видає відповідне системне повідомлення. Наведемо приклади формування векторів із табульованими значеннями за допомогою символа (:).

```
>> format short
>> a=1
a =
1
>> b=10
b =
10
>> s=0.1
s =
0.1000
>> v=a:b:s
v =
Empty matrix: 1?by?0
>> v=a:s:b
v =
Columns 1 through 7
1.0000 1.1000 1.2000 1.3000 1.4000 1.5000 1.6000
Columns 8 through 14
1.7000 1.8000 1.9000 2.0000 2.1000 2.2000 2.3000
Columns 15 through 21
2.4000 2.5000 2.6000 2.7000 2.8000 2.9000 3.0000
Columns 22 through 28
3.1000 3.2000 3.3000 3.4000 3.5000 3.6000 3.7000
Columns 29 through 35
3.8000 3.9000 4.0000 4.1000 4.2000 4.3000 4.4000
Columns 36 through 42
4.5000  4.6000 4.7000 4.8000 4.9000 5.0000 5.1000
Columns 43 through 49
```

```
5.2000 5.3000 5.4000 5.5000 5.6000 5.7000 5.8000
Columns 50 through 56
5.9000 6.0000 6.1000 6.2000 6.3000 6.4000 6.5000
Columns 57 through 63
6.6000 6.7000 6.8000 6.9000 7.0000 7.1000 7.2000
Columns 64 through 70
7.3000 7.4000 7.5000 7.6000 7.7000 7.8000 7.9000
Columns 71 through 77
100
101
8.0000 8.1000 8.2000 8.3000 8.4000 8.5000 8.6000
Columns 78 through 84
8.7000 8.8000 8.9000 9.0000 9.1000 9.2000 9.3000
Columns 85 through 91
9.4000 9.5000 9.6000 9.7000 9.8000 9.9000 10.0000
>>
```

Як бачимо, за великої кількості елементів вектора вони виводяться на екран не одним рядком, а розташовуються як таблиця чисел, при цьому для кожного рядка цієї таблиці вказуються номери елементів, розташованих у ньому. Така систематизація інформації значно спрощує роботу зі структурами числових даних. Сам по собі вектор з табульованими значеннями нібито не надає ніяких переваг для проведення подальших обчислень.Але в MatLab є можливість використовувати векторні змінні як параметри математичних функцій, і при цьому значення функції обчислюється для кожного з елементів вектора. Тоді досить просто розв’язати задачу обчислення значень функції для відповідного вектора аргумента, і результатом таких обчислень буде вектор значень функції. У більшості мов програмування для розв’язання такої задачі необхідно формувати циклічну структуру, а у MatLab вона розв’язується за допомогою кількох простих командних рядків.

У попередньому прикладі було табульовано вектор значень у числовому діапазоні [1;10] з кроком 0,1. Тепер, використовуючи сформований вектор як параметр функції, легко знайти значення функції cos (x) у визначеному числовому діапазоні:

```
>> b=cos(v)
b =
Columns 1 through 7
0.5403 0.4536 0.3624 0.2675 0.1700 0.0707 ?0.0292
Columns 8 through 14
?0.1288 ?0.2272 ?0.3233 ?0.4161 ?0.5048 ?0.5885 ?0.6663
Columns 15 through 21
?0.7374 ?0.8011 ?0.8569 ?0.9041 ?0.9422 ?0.9710 ?0.9900
Columns 22 through 28
?0.9991 ?0.9983 ?0.9875 ?0.9668 ?0.9365 ?0.8968 ?0.8481
Columns 29 through 35
?0.7910 ?0.7259 ?0.6536 ?0.5748 ?0.4903 ?0.4008 ?0.3073
Columns 36 through 42
```

```
?0.2108 ?0.1122 ?0.0124 0.0875 0.1865  0.2837  0.3780
Columns 43 through 49
0.4685  0.5544  0.6347  0.7087  0.7756  0.8347  0.8855
Columns 50 through 56
0.9275  0.9602  0.9833  0.9965  0.9999  0.9932  0.9766
Columns 57 through 63
0.9502  0.9144  0.8694  0.8157  0.7539  0.6845  0.6084
Columns 64 through 70
0.5261  0.4385  0.3466  0.2513  0.1534  0.0540  -0.0460
Columns 71 through 77
-0.1455 ?0.2435 ?0.3392 -0.4314 -0.5193 ?0.6020 -0.6787
Columns 78 through 84
-0.7486 ?0.8111 ?0.8654 -0.9111 -0.9477 ?0.9748 -0.9922
Columns 85 through 91
-0.9997 ?0.9972 ?0.9847 -0.9624 -0.9304 ?0.8892 -0.8391
>>
```

Маючи такі числові результати, тобто вектор аргументу функції v та вектор значень функції b, можна використати їх у подальших обчисленнях. За отриманими результатами обчислень нескладно побудувати графік функції y(x)=cos(x), використовуючи графічні засоби системи MatLab. Зрозуміло, що під час роботи з векторами та матрицями великої надмірності часто немає сенсу виводити всі результати на екран. Слід мати на увазі, що при обчисленні складних математичних функцій з великою кількістю аргументів виведення результатів на екран не тільки ускладнює їх аналіз, але й значно уповільнює роботу програми. Щоб не виводити на екран результати обчислень, треба у кінці рядка ставити символ ; (крапку з комою).

Розглянемо інший цікавий приклад використання векторів з табульованими значеннями параметрів функції.

```
>> x=0:5
x=
0 1 2 3 4 5
>> sin(x)/x
ans=
-0.0862
>>
```

Видно, що обчислення функції sin(x)/x у даному випадку виконано неправильно. Річ у тому, що ми використали не поелементне ділення, а розділили один вектор на інший за правилами лінійної алгебри. Тепер ви бачите, до яких грубих помилок призводить пропущення крапки у поелементних операторах! Реультатом ділення двох векторів є скаляр. Система провела обчислення саме таким чином, і цілком правильно. У даному випадку зрозуміло, що замість матричного оператора / треба використовувати поелементний оператор ./ .

```
>> sin(x)./x
Warning: Devide by zero
ans=
NaN
```

```
0.8415   0.4546   0.0470   ?0.1892   ?0.1918
>>
```

Тут теж не обійшлося без помилки, оскільки для x = 0 вираз невизначений. Зверніть увагу на те, що такому діленню відповідає не машинна нескінченність (Inf), а невизначеність (NaN).

Визначення матриць та векторів

Описання матриць та векторів здійснюється через визначення змінних. У цьому разі використовують три прості правила.

1. Усі елементи матриці або вектора необхідно брати у квадратні дужки.

2. Між числами, які відповідають елементам вектора або рядка матриці, можна ставити пробіл або кому.

3. Між рядками матриці ставлять крапку з комою.

Приклади визначення матриць та векторів:

```
>> v=[1, 2, 3, 4]
v=
1 2 3 4
>> MAT =[1,2;3,4]
M=
1 2
3 4
>>
```

Нескладно також звернутися до елементів вектора або матриці для виведення на екран або використання у виразах. Для цього достатньо вказати в круглих дужках порядковий номер елемента вектора або номер рядка та стовпця матриці через кому. Ці правила запису цілком відповідають відповідним правилам у математиці. Продовжуючи попередній приклад, розглянемо таку послідовність командних рядків:

```
>> v(1)=
1
>> M(1,1)=
1
>>
```

Найважливіше, що є у системі MatLab і чого немає в інших системах програмування та математичних САПР — це можливість виконання арифметичних дій ізматрицями та векторами та їх використання як параметрів математичних функцій. У цьому разі обчислення проводяться з кожним елементом матриці або вектора. З основ програмування відомо, що в кожній з мов програмування запис типу sin(v) або exp(M) , де M — матриця,v — вектор, був би некоректним. Щоб виконати будь-яку математичну дію з елементами вектора або матриці, потрібно організувати цикл. Обчислення функції від всіх елементів вектора або матриці є можливим у системі MathCAD, але там це робиться через спеціальну функцію векторизації. З точки зору точного математичного опису

вцьому є певна рація, оскільки у математиці такі операції заборонені. Але під час розробки реальних програм найчастіше необхідно обробляти масиви великих чисел саме таким чином, наприклад, поелементно додавати, множити поелементно або на константу, обчислювати функції від елементів масиву. А такі дії, як множення матриць або векторний добуток двох векторів, потрібні програмісту значно рідше, тому їх можна реалізувати через внутрішні функції системи. Розробники MatLab виходили, ймовірно, саме із цих міркувань, адже система створювалася для професіоналів?програмістів. Томуу MatLab використані записи типу exp(M) або sin(v) як поелементні дії з матрицями та векторами. Адже програмісту зручно працювати саме так, хоча такий запис неправильний з математичної точки зору. До того ж, такий підхід відповідає всій ідеології системи, тому що вона розроблена для роботи з матрицями, і вони розглядаються як звичайні числа. А спеціальні матричні та векторні операції тут також введені, але через окремі функції.

Тому множення, ділення, піднесення у степінь умові MatLab реалізовані як за правилами лінійної алгебри, так і для поелементних операцій, але запис цих функцій різний. Якщо визначені дії слід виконувати поелементно, використовуються оператори .* для поелементного множення, ./ для по елементного ділення та .^ для поелементного піднесення у ступінь, у цьому разі розмірність масивів для поелементних операцій має бути однаковою. При використанні операторів */ та ^математичні дії виконуються за правилами лінійної алгебри. У випадку множення вектора або матриці на числову константу по елементні операції повністю збігаються з операціями лінійної алгебри.

Дії додавання та віднімання для матриць або векторів та числових констант мають єдине значення. Під час додавання або віднімання двох матриць або векторів ці операції також виконуються поелементно, у цьому разі для уникнення помилок програмування розмірності векторів або матриць повинні бути однаковими. Наведемо приклад використання по елементних операцій та операцій лінійної алгебри мовою MatLab.

Приклад

```
>> format short
>> v=[1,2,3,4];
>> sin(v)
ans=
0,8415 0,9093 0.1411 ?0,7568
82
83
>> 3*v
ans=
3  6  9  12
>> v^2
???Error using= =>^Matrix must be square.
>> v.^2
ans=
1  4  9  16
```

```
>> v+2
ans=
3  4  5  6
MAT=[1,2;3,4]
MAT =
1     2
3     4
>> sin (MAT)
ans =
0.8415    0.9093
0.1411   ?0.7568
>> MAT^2
ans =
7    10
15    22
>> MAT.^2
ans =
1     4
9    16
>> 3*MAT
ans =
3     6
9    12
>> 2+MAT
ans =
3     4
5     6
>>
```

У наведеному прикладі повідомлення

```
???Error using =
=>^ Matrix must be square
```

(Помилкове використання оператора ^. Повинна бути квадратна матриця)

означає,що використання операції множення векторів за правилами лінійної алгебри неможливе, можна множити одну на одну тільки квадратні матриці. Як бачимо, для квадратної матриці система дійсно не видає помилку і виконує операцію множення матриці саму на себе за правилами лінійної алгебри, але результат суттєво відрізняється від поелементного множення. Усі результати поелементних операцій для вектора v та матриці MAT однакові оскільки містять ті самі елементи.

Функції перетворення матриць

У системі MatLab є багато функцій для створення матриць на основі заданої та перетворення матриць. Такі матричні операції дуже часто використовують під час числових обчислень для погодження отриманих результатів між собою,

Розглянемо головні матричні функції системи, наведемо формати їх запису та приклади використання.

Першою з таких важливих функцій є контанктенція, тобто об'єднання матриць. Матриці можна об'єднувати горизонтально або вертикально. Має місце, крім того, третій спосіб об'єднання, зі створенням тривимірного масиву, але багатовимірні структури числових даних і потужний апарат роботи з ними, розвинутий у системі MatLab.

Формат функції об'єднання матриць cat(dim, A, B), де

dim — специфікація розмірності масиву, а A та B — матриці, що об'єднуються. Параметри специфікації розмірності dim визначені таким чином:

dim = 1 — об'єднання матриць за рядками, кількості рядків у матрицях A та B повинні збігатися;

dim = 2 — об'єднання матриць за стовпчиками, кількості стовпчиків у матрицях A та B повинні збігатися;

dim = 3 — об'єднання матриць зі створенням тривимірного масиву, кількості матриць A та B повинні бути однакової розмірності.

Слід зазначити, що функцію об'єднання матриць легко здійснити через множинну індексацію та методи формування матриць, наявність функції cat у системі MatLab є дещо надлишковою і користуються нею рідко.

Іншими важливими функціями опрацювання матриць є функції переставлення їх елементів. Наприклад, функція fliplr(A) здійснює дзеркальне відображення елементів матриці зліва'направо відносно вертикальної осі. Функція flipud(A) здійснює дзеркальне відображення елементів матриці зверху вниз відносно горизонтальної осі. Функція reshape(A, l,k) створює з матриці A розмірності m*n іншу матрицю розмірності l*k через послідовний вибір елементів матриці A по стовпцях. Обов'язковою умовою для проведення такої операції є виконання співвідношення m•n = l•k. Але безперечною перевагою функції reshape є те, що за її допомогою можна здійснювати будь'які перетворення розмірності матриці.

Розглянемо тепер важливі функції створення трикутних матриць і матриць з відповідними діагональними елементами. Функція tril(A)створює нижню трикутну матрицю на основі матриці A через присвоєння значення «0» усім її елементам, розташованим нижче головної діагоналі. Функція triu(A) створює верхню трикутну матрицю на основі матриці A, анулюючи її елементи, розташовані вище головної діагоналі. Для створення діагональної матриці використовують функцію MatLab diag.

Можливі такі формати функції:

diag(v) — створення квадратної матриці, на головній діагоналі якої розташовані елементи вектора v;

diag(v, n) — створення прямокутної матриці, на одній із діагоналей якої розташовані елементи вектора v, у цьому разі параметр n визнає зсув діагоналі розташування елементів відносно головної діагоналі. Значення n > 0 відповідають зсуву відносно головної діагоналі вгору, а значення n < 0 — зсуву вниз.

Інша функція формування матриці на основі заданої — це функція repmat(A,m,n), яка повертає матрицю B, що є результатом розмноження матриці A

m разів по стовпчиках та n разів по рядках. Розглянули два способи виконання тієї самої операції через використання множинної індексації, тому наявність функції repmat у системі MatLab теж є дещо надлишковою.

Є крім того, функція повороту елементів матриці на 90°

rot90(A, k), де A — матриця, що перетворюється, k — кількість поворотів, за замовчуванням k = 1.

Приклад

```
» format short
» M1=rand(2)
M1 =
0.4321    0.7660
0.2840    0.0976
» M2=rand(2)
M2 =
0.5558    0.2109
0.4483    0.1773
» cat(1,M1,M2)
ans =
0.4321    0.7660
0.2840    0.0976
0.5558    0.2109
0.4483    0.1773
» cat(2,M1,M2)
ans =
0.4321    0.7660    0.5558    0.2109
0.2840    0.0976    0.4483    0.1773
» format short
» M1=rand(2)
M1 =
0.1559    0.6852
0.2797    0.8352
» M2=rand(2)
M2 =
0.2201    0.4075
0.5674    0.7049
» C1=cat(1,M1,M2)
C1 =
0.1559    0.6852
0.2797    0.8352
0.2201    0.4075
0.5674    0.7049
» C2=cat(2,M1,M2)
C2 =
0.1559    0.6852    0.2201    0.4075
0.2797    0.8352    0.5674    0.7049
» fliplr(C1)
```

```
ans =
0.6852    0.1559
0.8352    0.2797
0.4075    0.2201
0.7049    0.5674
» flipud(C2)
ans =
0.2797    0.8352    0.5674    0.7049
0.1559    0.6852    0.2201    0.4075
» tril(C1)
ans =
0.1559         0
0.2797    0.8352
0.2201    0.4075
0.5674    0.7049
» triu(C1)
ans =
0.1559    0.6852
0    0.8352
0         0
0         0
» v=rand(3,1)
v =
0.2580
0.4546
0.3964
» diag(v)
ans =
0.2580         0         0
0    0.4546         0
0         0    0.3964
» diag(v,2)
ans =
0         0    0.2580         0         0
0         0         0    0.4546         0
0        0       0       0   0.3964
0         0         0         0         0
0         0         0         0         0
» diag(v,?2)
ans =
0         0         0         0         0
0         0         0         0         0
0.2580         0         0         0         0
0    0.4546         0         0         0
0         0    0.3964         0         0
» C1
```

```
C1 =
0.1559    0.6852
0.2797    0.8352
0.2201    0.4075
0.5674    0.7049
326
327
» rot90(C1)
ans =
0.6852    0.8352    0.4075    0.7049
0.1559    0.2797    0.2201    0.5674
» rot90(C1,2)
ans =
0.7049    0.5674
0.4075    0.2201
0.8352    0.2797
0.6852    0.1559
»
```

У MatLab функція норми вектора norm має такі формати:

norm(X, p) — функція повертає норму порядку p для вектора X;

norm(X, p) — функція повертає норму другого порядку для вектора X;

norm(X,’inf’) — функція повертає максимальне значення з абсолютних значень елементів вектора;

norm(X,’– inf’) — функція повертає мінімальне значення з абсолютних значень елементів вектора.

Наведемо приклади обчислення норми вектора.

Приклад

```
» format short
» S=[16,0,25,9,4,16,25,1,25]
S =
16     0    25     9     4    16    25
1    25
» norm(S)
ans =
49.8498
» norm(S,10)
ans =
27.9245
» norm(S,’inf’)
ans =
25
»
```

Наведемо стандартний приклад розв’язання системи лінійних алгебраїчних рівнянь. Запишемо систему з трьох рівнянь із трьома невідомими. Зрозуміло, можна провести аналіз сингулярних чисел матриці, яка формує нашу систему

рівнянь, використати функцію обчислення оберненої матриці, або LU-розкладення. Але виявляється, що поставлена задача має простий і гарний розв’язок у межах лише елементарних матричних операцій, і звертатися до функцій лінійної алгебри у нашому випадку немає потреби.

Справді, якщо AX = B, то X = BA–1= B/A = A\B. Здається, що немає різниці, як написати: B/A або A\B, ці операції є еквівалентними. Але розмірності структур A і B у нашому випадку не збігаються, і тому порядок здійснення операції ділення має значення з точки зору часу проведення розрахунків. І зворотне ділення матриці на вектор A\B з цієї точки зору ефективніше, оскільки здійснюється воно не через складну функцію обчислення оберненої матриці, а через дуже ефективний алгоритм методу Гауса.

Тому ефективне розв’язання поставленої задачі буде таким

Приклад

```
» A=[10,?7,2;3,?15,32;13,?9,15]
A =
10    ?7     2
3   ?15    32
13    ?9    15
» B=[6;10;26]
B =
6
10
26
» X=A\B
X =
2.3274
2.8804
1.4445
» A*X
ans =
6.0000
10.0000
26.0000
»
```

Функції в системі MatLab.
Загальні правила роботи з функціями в системі MatLab

Функції — це об’єкти MatLab, які мають унікальні імена та список параметрів, та при зверненні до них виконують відповідні дії з даними за впровадженим алгоритмом. Відрізняють вхідні, які передаються при зверненні з головної програми, та вихідні параметри функцій, які після закінчення оброблення даних функція передає до точки виклику. Параметрами функцій є змінні. Відомо, що у класичних мовах програмування функції, на відміну від

підпрограм, повертають результат через єдиний параметр за місцем виклику. Ця особливість дає змогу успішно використовувати функції для формування математичних виразів. Визначення функцій MatLab інше та ширше. Як було зазначено, ця система припускає використання у математичних виразах матриць та векторів, а не тільки числових змінних. Наявність по елементних дій зі структурами числових даних дає можливість ефективного використання масивів даних як параметрів математичних функцій. Приклади використання векторів, які були розглянуті у попередньому підрозділі, доводять простоту та ефективність такого підходу, який не дуже суворий з погляду правил математики, але спрощує розв'язання задач програмування. Наприклад, у задачі обчислення табличних значень функції у заданому діапазоні аргумента були використані два командних рядки. У цьому разі не використовувались циклічні структури, без яких розв'язання задачі засобами класичного програмування є неможливим. Можливість та ефективність використання матриць і векторів як вхідних та вихідних параметрів функцій під час формування математичних виразів робить вхідну мову MatLab простою, не перенасиченою зайвими поняттями та правилами. У разі такого розширення поняття функції відпадає потреба у визначенні поняття процедури, оскільки функція може повертати будь?яку кількість числових значень за місцем виклику. Правда, виникають певні труднощі, кожна функція потребує знання всіх вхідних, вихідних параметрів та порядку їх запису. Але тут у роботі з MatLab допомагає ефективне використання системи допомоги.

Загальний формат оператора виклику функції такий:

[Y1, Y2…] = ім'я_функції [X1, X2…]

Розширення поняття функції MatLab не обмежується можливістю використання векторів та матриць як вхідних та вихідних параметрів. У мовах програмування є також поняття перевантажених функцій, які мають одне ім'я, але відрізняються набором параметрів. Використання таких функцій у MatLab дозволяється, є вони і серед стандартних функцій системи. Стандартні функції діляться на внутрішні та зовнішні. Внутрішніми є найрозповсюдженіші математичні функції, наприклад exp (x),sin (x), atan (x) та інші. Зовнішні функції виконані окремими m-файлами, ім'я файлу відповідає назві функції. Прикладом зовнішньої математичної функції є гіперболічний синус sinh (x).

Відрізняють системні функції, які можуть мати вхідні параметри, але ніколи не повертають ніяких числових значень. Результатом роботи таких функцій є інформація, що відображується на екрані через текст або графіку. Функції системних дій нагадують принципом роботи системні команди, а функції графіки схожі на команди графічних редакторів. Імена функцій завжди збігаються з іменами файлів. Тому вони систематизовані за каталогами згідно з принципами, при цьому довідки щодо функцій можна отримати за допомогою команд help ім'я_функції або helpім'я_директорії:

help elfun— список елементарних математичних функцій;

help specfun— список спеціальних математичних функцій.

Функції користувача та їх описання

Зрозуміло, що, незважаючи на велику кількість бібліотек функцій, у MatLab визначені тільки основні математичні функції. Аті функції, які є результатом їхньої композиції, користувач повинен визначати сам. Це можна зробити за допомогою функції inline, або окремим файлом із розширенням *.m, написавши програму мовою системи MatLab. Другий спосіб використовується, якщо функція містить велику кількість рядків або користувачу потрібно звернутися до неї з різних файлів. Перший метод відповідає створенню внутрішніх, а другий — зовнішніх функцій. Робити описання функцій через командний рядок inline для програмістів ефективніше і простіше, ніж описувати функції не систематизовано, як це зроблено, наприклад, у системі MathCAD. Функції можуть знаходитися у будь-якій частині, вони ніде не описуються і, ніяким чином не упорядковані. Знайти таку функцію у великій програмі непросто. Виклик функцій з інших файлів не передбачений. Програмісту легше, коли здійснюється опис всіх функцій на початку або в кінці програми. Хоча оператор inline може стояти у будь-якому місці, з точки зору стилю програмування краще розташовувати функції послідовно на початку програми. Формат оператора inline:

змінна=inline(‘вираз’)

Параметри функції у цьому операторі не вказуються, вони формуються через послідовне зчитування параметрів усіх функцій у виразі, але при зверненні до функції їх необхідно вказувати.

Розглянемо простий приклад використання функції inline:

```
>> format short
>> sc2=inline(‘sin(x).^2+cos(y).^2’)
sc2 =
Inline function:
sc2(x,y) = sin(x).^2+cos(y).^2
>> sc2(3,4)
ans =
0.4472
>>
```

Спосіб створення зовнішньої функції — збереження її на диску у *.m файлі. Докладніше засоби створення таких функцій ми розглянемо у розділі 4, а зараз звернемо увагу на загальні принципи побудови з використанням уже відомих операторів.

За допомогою команди New опції меню File необхідно створити новий файл та запам’ятати його вдиректорії ./MatLab/work під ім’ям sc2. Текст повинен займати всього 2 командних рядки, обов’язково необхідно вводити коментарі для спрощення роботи з програмою. Тоді рядки для обчислення функції f(x, y) = sin2(x) + cos2(y) можуть мати такий вигляд:

```
% Обчислення функції f(x,y)=sin(x)^2+cos(y)^2
function I=sc2(x,y)
I=sin(x).^2+cos(y).^2
```

Створення зовнішньої функції закінчено, будь-який користувач системи може викикати її, а у разі потреби змінити рядки цієї програми. У цьому полягають переваги використання систем з відкритими кодами (Open Sources). Звернення до функції відбувається ще простіше: через список імен змінних,sc(x, y). Приклад використання функції sc2:

```
>> sc2(1,2)
ans=
0.8813
>> sc2(2,1)
ans=
1.1187
>>
```

Припускається посилання на функцію через ім’я змінної.Воно відповідає правилам мови програмування С:

```
>> fh=@sc2
```

Тепер, використовуючи функцію feval(fh,x,y), отримується значення функції sc2 для заданих x і y:

```
>> feval(fh,1,2)
ans=
0.8813
>> feval(fh,2,1)
ans=1.1187
>>
```

Механізм оброблення зовнішніх функцій у системі MatLab

У MatLab, на відміну від систем програмування, виклик зовнішніх функцій нічим не відрізняється від виклику внутрішніх і вбудованих, тобто, користувачу не треба описувати назву бібліотек та імена файлів, які необхідно підключати для правильного виконання програми.Аналіз імен функцій здійснюється автоматично таким чином.Спочатку система за внутрішньо індексним каталогом швидко визначає, чи є відповідне ключове слово серед службових слів системи. Якщо це так — відразу проводиться аналіз для вбудованих та внутрішніх inline функцій, якщо серед їх імен ключового слова немає — система шукає m?файл із відповідним ім’ям. Якщо файл не знайдений, система видає повідомлення про помилку:

```
???Undefined function or variable.
```

Такий алгоритм пошуку розвинений у системах з нечіткою логікою (fussy logic systems), і в мовах програмування задач нечіткої логіки: ЛОГО, ФОРТ, ПРОЛОГ.

Спеціальні дії з числами та структурами в системі MatLab, реалізовані через функції. Порозрядне оброблення даних

MatLab має спеціальний набір функцій, які є елементарними математичними діями з числами та множинами, і тому вони розглядаються як операнди. До цього класу функцій належать побітові операції з числами (Bitwise operators) та функції оброблення множин (Set operators). Ці функції MatLab розробники віднесли до класу операторів, система допомоги видає їх у списку операторів при виконанні команди help ops.

Окреме місце займають побітові дії з числами. Відомо, що комп'ютер здійснює побайтове оброблення даних, і навіть у мові програмування Асемблер побітові дії виконуються над байтами даних за допомогою відповідних команд, а у більшості мов програмування високого рівня, таких як Pascal, FORTRAN, побітові дії неможливі. Але такі дії часто необхідні програмістам для реалізації ефективних алгоритмів оброблення даних, особливо при програмуванні зовнішніх апаратних пристроїв та при розв'язанні інших задач системного програмування. Тому можливість виконання побітових дій у MatLab зайвий раз доводить,що ця система створена для фахівців?програмістів і призначена для розв'язання нетривіальних задач програмування. У MatLab функції побітових дій здійснюються тільки з додатними натуральними числами. Наведемо перелік функцій системи, призначений для виконання побітових операцій. Зверніть увагу на те, що аргументами функцій можуть бути матриці.

bitand (M1,M2) — функція поелементно виконує побітове «І» для всіх елементів матриць M1 та M2, розмір яких повинензбігатися, якщо одна з матриць не є числом.

bitcmp(M1,M2)— функція виконує побітове доповнення всіх елементів матриць M1 та M2. Розміри цих матриць повинні бути однаковими, якщо одна з них не є числом. Функція d(m, n) побітового доповнення числа m до числа n у двійковій арифметиці визначається так:

d(m, n)=2n– 1 – m.

bitor (M1,M2) — функція поелементно виконує побітове «АБО» для всіх елементів матриць M1 та M2, розміри яких повинні бути однаковими, якщо одна з матриць не є числом.

bitmax— системна змінна, яка повертає значення максимального цілого беззнакового числа, що може бути подано у форматі з плаваючою комою на даному комп'ютері. Для обчислювальних машин з арифметикою стандарту IEEE, якому відповідають і комп'ютери IBM PC, це значення дорівнює 253– 1. Наприклад:

```
>> 2^53-1
ans =
9.007199254740991e+015
>> bitmax
ans =
9.007199254740991e+015
>>
```

bitset (M1,M2) — функція встановлює одиничне значення у відповідному біті числа, при цьому змінюваному числу відповідає елемент матриці M1,ана порядковий номер змінюваного біта вказує відповідний елемент матриці

M2.Розміри матриць M1та M2 повинні бути однаковими, якщо одна з цих матриць не є числом.

bitset (M1,M2,M3) — функція встановлює одиничне або нульове значення у відповідному біті числа, при цьому змінюваному числу відповідає елемент матриці M1, на порядковий номер змінюваного біта вказує елемент матриці M2, а матриця M3 повинна складатися тільки з нулів або одиниць, та її відповідний елемент визначає значення змінюваного біта. Розмірність матриць M1, M2 та M3 повинна бути однаковою, якщо будь-яка з них не є числом.

bitget(M1,M2)— функція зчитує значення біта числа, у цьому разі числу, яке читається, відповідає елемент матриці M1, а порядковий номер біта, указує елемент матриці M2. Розміри матриць M1 та M2 повинна бути однаковою, якщо одна з цих

матриць не є числом.

bitshift (M1,M2) — функція поелементно виконує логічний зсув праворуч або ліворуч всіх елементів матриці M1, при цьому кількість позицій зсуву визначається відповідними елементами матриці M2. На відміну від попередніх функцій побітового оброблення даних, елементами матриці M2 можуть бути як додатні, так і від’ємні числа. При додатних значеннях елементу матриці M2 здійснюється логічний зсув ліворуч, а при від’ємних — праворуч. Розміри матриць повинні бути однаковими, якщо одна з матриць не є числом. Як відомо з основ двійкової арифметики, зсув на n позицій ліворуч відповідає множенню числа на 2n, а зсув праворуч — його діленню на 2n,або множенню на 2–n[7, 10].

bitxor (M1,M2) — функція поелементно виконує побітове «виключного АБО» для всіх елементів матриць M1та M2, розміри яких повинні бути однаковими, якщо одна зматриць не є числом.

Якщо програміст не впевнений, що операнди функцій побітового оброблення є цілими числами, рекомендується використовувати їх разом з функціями округлення.

Наведемо приклад використання побітових дій оброблення \даних.

Приклад
```
>> A=[1,2;3,4]
A=
1     2
3     4
>> B=[1,7;3,8]
B=
1     7
3     8
>> C=bitand(A,B)
C=
1     2
3     0
>> D=bitor(A,B)
D=
1     7
3    12
```

```
>> E=bitcmp(A,5)
E=
30    29
28    27
>> U=bitcmp(C,5)
U=
116
117
30    29
28    31
>> G=bitset(C,4)
G=
9    10
11     8
>> F=[1,0;0,1]
F=
1     0
0     1
>> H=[2,3;4,5]
H=
2     3
4     5
>> I=bitset(C,H,F)
I=
3     2
3    16
>> J=bitget(D,H)
J=
0     1
0     0
>> K=bitshift(E,H)
K=
120   232
448   992
>> L=bitshift(E,?H)
L=
7     3
1     0
>> M=bitxor(H,L)
M=
5     0
5     5
>>
```

Дії з множинами

Множина— це одне з первинних понять математики, яке визначає неупорядковану сукупність об’єктів будь- якої фізичної природи. Дії з множинами у сучасній математиці стали класичними та ефективно використовуються для оброблення

неупорядкованих структур. У системі MatLab дії з множинами виконуються через наведені нижче функції оброблення векторів та матриць.

intersect (V1,V2) — функція обчислює вектор, який має спільні елементи векторів V1 та V2, упорядковані за зростанням. Ця функція відповідає математичній дії перетину двох множин. Недолік роботи цієї функції у наведеному форматі полягає в тому, що вона не надає інформації про номери спільних елементів, тому існує ще один спосіб використання цієї функції:

[R1,R2,R3]=intersect (V1,V2)

вектор R1 містить спільні елементи векторів V1 та V2, вектор R2 визначає їх порядкові номери у векторі V1, а R3 — порядкові номери у V2.

intersect (M1,M2, ‘rows’) — функція обчислює матрицю, яка має спільні рядки порівнюваних матриць M1, M2, у цьому разі вони повинні мати однакову кількість стовпчиків.

ismember (V1,V2) — результатом роботи функції є вектор, розмір якого збігається з розміром вектора V1. Елемент вектора результата дорівнює 1, якщо відповідний елемент вектора V1 входить до множини, яка визначається вектором V2, в ін. шому разі цей елемент дорівнює 0.

ismember (M1,M2, ‘rows’) — результатом роботи функції є вектор, кількість елементів якого відповідає кількості стовпчиків у матрицях M1 та M2, при цьому матриці повинні мати однакову кількість стовпчиків. Елемент вектора дорівнює 1, як?

що відповідний рядок матриці M1 є рядком M2, в іншому разі елемент вектора результата дорівнює 0.

setdif (V1,V2) — результатом роботи функції є вектор, який містить усі елементи вектора V1, які не є елементами вектора V2. У математиці таку дію називають різницею двох множин. Елементи вектора результату сортуються у бік збільшення.

Функція setdif (V1,V2) використовується також у форматі:

[R1,R2]= setdif (V1,V2)

У наведеному виразі вектор R1має елементи вектора V1, які не є елементами V2, авектор R2визначає їх порядковий номер.

setdif (M1,M2, ‘rows’) — результатом роботи функції є матриця, що містить усі рядки M1, які не є рядками M2, при цьому матриці M1та M2 повинні мати однакову кількість стовпчиків.

setxor (V1,V2) — результатом роботи функції є вектор, який містить усі елементи векторів V1 та V2, які не є для них спільними. Тобто, вектор результату має ті елементи, які не є перетином двох множин. Результат сортується у бік збільшення.

Функція setdif (V1,V2) використовується також у форматі:

[R1,R2,R3]= setxor (V1,V2)

У наведеному виразі вектор R1 містить усі неспільні елементи векторів V1 та V2, вектор R2 визначає їх порядковий номер у векторі V1, а вектор R3 — порядковий номер у векторі V2.

setxor (M1,M2, ‘rows’) — результатом роботи функції є матриця, що містить усі рядки матриць M1 та M2, які не є для них спільними, при цьому матриці M1 та M2 повинні мати одна кову кількість стовпчиків.

union (V1,V2) — результатом роботи функції є вектор, який містить усі елементи векторів V1 та V2, але без повторень. Елементи вектора результату сортуються у бік збільшення. У математиці таку дію називають об’єднанням двох множин.

Функція union (V1,V2) використовується також у форматі:

[R1,R2,R3]= union (V1,V2)

У наведеному виразі вектор R1містить усі елементи векторів V1 та V2без повторень, R2визначає їх порядковий номер у векторі V1, а R3 — порядковий номер у векторі V2.

union (M1, M2, ‘rows’) — результатом роботи функції є матриця, що містить усі рядки матриць M1 та M2 без повторень, у цьому разі матриці M1 та M2 повинні мати однакову кількість стовпчиків;

unique (V1) — результатом роботи функції є вектор, який має елементи вектора V1 без повторень;

unique (M1, ‘rows’) — результатом роботи функції є матриця, що має рядки матриці M1 без повторень.

Наведемо приклад використання дій з множинами для оброблення структур числових даних.

Приклад

```
>> M1=[2,3,4;4,3,5;5,6,7];
>> M2=[3,5,4;4,3,5;5,6,7];
```

```
>> V1=[2,3,4,5,6,7];
>> V2=[3,4,6,7,8,1];
>> intersect(V1,V2)
ans =
3    4     6     7
>> [A,B,C]=intersect(V1,V2)
A=
3    4     6     7
B=
2    3     5     6
C=
1    2     3     4
>> intersect(M1,V2,'rows')
??? Error using ==> intersect
A and B must have the same number of columns.
>> intersect(M1,M2,'rows')
ans=
4    3     5
5    6     7
>> ismember (A,V1)
ans=
1    1     1     1
>> ismember (M1,M2,'rows')
ans=
0
1
1
>> ismember (A,C)
ans=
1    1     0     0
>> setdiff(A,C)
ans=
6    7
>> [D,E]=setdiff(A,C)
D=
6    7
E=
3    4
>> setdiff(M1,M2,'rows')
ans=
2    3     4
>> setdiff(M2,M1,'rows')
ans=
3    5     4
>> setxor (A,B)
ans=
```

```
2    4     5     7
>> [F,G]=setxor (A,B)
F=
2    4     5     7
G=
2     4
>> setxor (M1,M2,'rows')
ans=
2    3     4
3    5     4
>> union (V1,V2)
ans=
1    2     3     4     5     6     7    8
>> [H,K]=union (A,B)
H=
2    3     4     5     6     7
K=
2    4
>> union (M1,M2,'rows')
ans=
2    3     4
3    5     4
4    3     5
5    6     7
>> unique (V1)
ans=
2    3     4     5     6     7
>> V3=[2,5,6,2,6,4,5,2]
V3=
2    5     6     2     6     4     5    2
>> unique (V3)
ans=
2    4     5     6
>> M3=[V1;V2;V1]
M3=
2    3     4     5     6     7
3    4     6     7     8     1
2    3     4     5     6     7
>> unique (M3)
ans=
1
2
3
4
5
6
```

```
7
8
>> unique (M3, ‘rows’)
ans=
2     3     4     5     6     7
3     4     6     7     8     1
>>
```

Елементарні математичні функції для роботи з числовими даними

Тригонометричні функції

У розділі описані елементарні математичні функції системи MatLab, зокрема тригонометричні, зворотні тригонометричні, гіперболічні, зворотні гіперболічні, показникові, експоненціальні, логарифмічні та цілочисельні функції, наведені прикладиїх використання.

У системі MatLab є набір стандартних тригонометричних функцій, які можуть бути використані у математичних виразах. Переважна більшість із них вбудовані у ядро системи і не можуть змінюватись. Список тригонометричних функцій системи можна переглянути через команду help elfun:

sin(x) — функція синуса;
cos(x) — функція косинуса;
tan(x) — функція тангенса;
sec(x) — функція секанса,
cot(x) — функція котангенса,
Приклади обчислення тригонометричних функцій:

```
» format short
» x=pi
x=
3.1416
1
csc(x) – функція косеканса,
1
csc( );
sin( )
x
x
=
» y=sin(x)
y=
1.2246e?016
» y=cos(x)
y=
-1
```

```
» y=tan(x)
y=
-1.2246e?016
» y=cot(x)
y=
-8.1656e+015
» y=sec(x)
y=
-1
» y=csc(x)
y=
8.1656e+015
»
```

Зворотні тригонометричні функції

У системі MatLab існує набір зворотних тригонометричних функцій, що використовуються у математиці та у математичних виразах при написанні програм. Треба пам’ятати про те,що при обчисленні зворотних тригонометричних функцій у системі MatLab результатом можуть бути не тільки дійсні, але й комплексні числа.

asin(x)— функція арксинуса.Обчислюється кут у радіанах, який лежить у діапазоні

acos(x) — функція арккосинуса. Обчислюється кут у радіанах, який лежить у діапазоні [0; π];

atan2(x,y) — чотириквадратний тангенс. Функція обчислює кут, який лежить удіапазоні [–π; π] між віссю абсцис і крапкою (x, y);

atan(x) — функція арктангенса;

asec(x) — функція арксеканса;

acsc(x) — функція арккосеканса;

acot(x) — функція арккотангенса.

Приклади обчислення зворотних тригонометричних функцій.

Приклад

```
» x=1
x=
1
» y=asin(x)
y=
1.5708
» y=acos(x)
y=
0
» y=atan(x)
y=
0.7854
» y=asec(x)
y=
0
» y=acsc(x)
y=
1.5708
» y=acot(x)
y=
0.7854
» y=atan2(2,3)
y=
0.5880
» x=10
x=
```

```
10
» y=asin(x)
y=
1.5708 + 2.9932i
»y=acos(x)
y =
0 ? 2.9932i
»
```

Гіперболічні та зворотні гіперболічні функції

У системі MatLab є набір гіперболічних та зворотних гіперболічних функцій. Результатом обчислення цих функцій можуть бути не тількі дійсні, але й комплексні числа.

sinh(x) — функція гіперболічного синуса,

asinh(x) — зворотній гіперболічний синус, або ареа-синус.

Назва ареа пов’язана з тим, що ця функція визначає площу сектора, обмеженого двома променями, що виходять із початку координат, і відрізком гіперболи x2– y2= 1 (

Якщо y = Arsh(x), то x = sh(y);

acosh(x) — зворотній гіперболічний косинус, або ареа-косинус.

Якщо y = Arch(x), то x = ch(y), то,x ≥ 1.

tanh(x) — функція гіперболічного тангенса tanh (x) =

atanh(x) — зворотний гіперболічний тангенс, або ареатангенс.|x| < 1;

sech(x) — функція гіперболічного секанса

cosh(x)— функція гіперболічного косинуса, ch( )

asech(x) — ареасеканс, або гіперболічний арксеканс;

csh(x)— функція гіперболічного косеканса

acsch(x) — функція зворотного гіперболічного косеканса;

coth(x) — функція гіперболічного котангенса ctg(x) =

acoth(x) — зворотний гіперболічний котангенс, ареакотангенс областю визначення аргументу функції y=acoth(x)є |x| > 1.

Приклади обчислення гіперболічних та зворотних гіперболічних функцій.

Приклад

```
» x=0.5
x=
0.5000
» y=sinh(x)
y=
0.5211
» y=cosh(x)
y=
1.1276
» y=asinh(x)
y=
0.4812
» y=acosh(x)
y=
0 + 1.0472i
» y=tanh(x)
y=
0.4621
» y=atanh(x)
y=
```

```
0.5493
»
1
```

. Експоненціальна, степенева та логарифмічна функції

У системі MatLab існують розповсюджені експоненціальна,ступенева та логарифмічна функції. Результатом обчислення логарифмічної функції можуть бути не тількі дійсні, але й комплексні числа, що суттєво впливає на область визначення логарифмічної функції та дає змогу використовувати від’ємні числа як її аргументи. Якщо користувач бажає обчислювати логарифми тільки від додатних чисел і отримувати дійсні числа як результат, можна перевіряти обчислені дані за допомогою функції оброблення комплексних чисел isreal. Окремої функції для піднесення у ступінь немає, оскільки ця операція віднесена до елементарних, і для неї є відповідний оператор ^. Замість оператора піднесення у ступінь можна використовувати функцію power, але здебільшого під час формування математичних виразів зручніше застосовувати оператори. Але функція квадратного кореня від числа, яка найчастіше використовується під час проведення обчислень, існує окремо.

exp(x) — функція експоненти;

log(x) — функція натурального логарифма;

log10(x) — десятковий логарифм;

sqrt(x) — функція обчислення квадратного кореня;

nextpow2(x) — показник ступеня числа 2, який відповідає найближчому цілому числу, більшому за x. Цю функцію, разом з функцією цілочисельного ділення, дуже часто використовують в електроніці при моделюванні аналогово-цифрових перетворювачів.

Приклади обчислення експоненціальних, ступеневих та логарифмічних функцій.

Приклад

```
» format short
» x=?3:1:3;
» log(x)
Warning: Log of zero.
ans=
Columns 1 through 4
1.0986 + 3.1416i  0.6931 + 3.1416i  0 + 3.1416i ?Inf
Columns 5 through 7
0           0.6931        1.0986
» sqrt(log(x))
168
169
Warning: Log of zero.
ans=
Columns 1 through 4
1.4877+1.0558i 1.3983+1.1234i 1.2533+1.2533i 0+ Inf i
Columns 5 through 7
0             0.8326             1.0481
» sqrt(log(x)+exp(x))
Warning: Log of zero.
```

```
ans=
Columns 1 through 4
1.4989+1.0480i 1.4278+1.1001i 1.3287+1.1822i 0+Inf i
Columns 5 through 7
1.6487             2.8429             4.6026
» nextpow2 (100)
ans=
7
»
```

Функції округлення та обчислення цілої частини числа

У системі MatLab виконані функції оброблення чисел, округлення та обчислення цілого числа.

abs(x) — модуль числа x;

fix(x) — округлення до найбільшого цілого у бік нуля;

floor(x) — округлення до найменшого цілого у бік негативної нескінченності;

ceil(x) — округлення до найменшого цілого у бік позитивної нескінченності;

round(x) — округлення числа x у бік найближчого цілого.

Графічні функції цілих чисел MatLab для різних значень аргументу x

mod(x,y)— обчислення залишку ділення числа хна число y;

rem (x,y) — обчислення залишку ділення числа x на число y за формулою x – y.*fix(x./y). Функція rem (x,y) відрізняється від mod (x,y) тим, що замість floor(x./y) використовується функція fix(x./y);

sign(x) — обчислення знакової функції sign (x):

Файли, у яких реалізовані математичні функції, зберігаються в директорії: /MatLab/Toolbox/MatLab/elfun. Допомогу щодо цих функцій можна отримати через команди help elfuna bohelp ім’я функції.

Приклад

```
» format short
» x=3.4; y=-3.8;
» abs (x*y)
ans=
12.9200
» fix (x)
ans=
3
» fix (y)
ans=
-3
» floor (x)
ans=
3
» floor (y)
ans=
?4
» ceil (x)
ans=
4
» ceil (y)
ans=
?3
» round (x)
ans=
3
» round (y)
ans=
```

```
170
kx2k + 1
nx1n + 1
–1
0
1
floor(x2)
round(x2)
fix(x1)
floor(x1)
round(x1)
fix(x2)
ceil(x2)
ceil(x1)
171
?4
» mod (x,y)
ans=
?0.4000
» rem (x,y)
ans=
3.4000
» mod (3,4)
ans=
3
» rem (3,4)
ans=
3
» mod (10,2)
ans=
0
» rem (10,2)
ans=
0
```

Графічні функції системи MatLab та особливості їх використання

У розділі описані графічні функції системи MatLab, ураховуючи двовимірну та тривимірну побудову поверхонь та контурних ліній. Наведені приклади використання графічних функцій MatLab для подання наукової та ділової графіки.

Основні функції двовимірної графіки

Побудова графіків через інтерполяцію дискретнихвідліків відрізками прямих ліній

Побудова графіка функції y=f(x) у декартовій системі координат у MatLab здійснюється дуже легко. Для виконання цієї дії використовується команда plot,

яка має кілька різновидів. Найпростіша з них — це команда із двома параметрами plot(x,y), де x і y — вектора або матриці однакового розмі ру. Якщо x і y — вектора, то будується один графік, у якому значення першої змінної x відкладаються по осі абсцис, а другої змінної y — по осі ординат. Якщо x та y — матриці, то будується сукупність графіків за даними, що міститься у стовпчиках матриць. Очевидно, що явний опис функції y=f(x) не є обов'язковим. Достатньо задати два вектори — значеня аргументу та значення функції. Наприклад, графіки функцій y=sin(x) і y=cos(x) можна побудувати на одній осі координат, написавши такий фрагмент програми:

```
»x=0:5;
»y=[sin(x);cos(x)];
»plot(x,y)
```

Графік складається зі з'єднаних лінійних відрізків, як утворюють ламану лінію. Наведений приклад наочно показує побудову графіків у системі MatLab. Точки відліку відповідають цілочисловим значенням, тому за лінійної інтерполяції незначна кількість точок не дає змогу відобразити функції між відліками. Якщо ви бажаєте отримати більш гладку криву, необхідно визначити більше значень аргумента x. Результат роботи програми, яка містить команди виведення графіків, відображується в окремому графічному вікні. Скопіювати отриманий графік у буфер обміну Windows можна через команду менюEdit → Copy Figure.

Під час копіювання графіка не беруть до уваги сірий фон,на якому він розташований. Таку опцію копіювання можна задати із системного меню File → Preferences → Copying Options.

Інший формат команди plot (y), де y — вектор дійсних чисел. Буде побудований графік функції y(i), де i — номер елемента вектора. Приклад такого використання функції plot

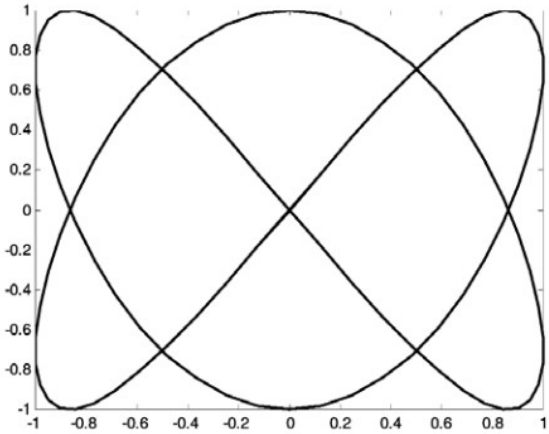
```
» x=1:10
x =
1     2     3     4     5     6     7     8     9    10
» y=sin(x)
y =
Columns 1 through 7
0.8415    0.9093    0.1411    ?0.7568    ?0.9589  ?0.2794
0.6570
Columns 8 through 10
0.9894    0.4121   ?0.5440
» plot(y)
»
```

Якщо y— вектор комплексних чисел, команда plot(y)виконується так: plot(real(y), imag(y)). У всіх інших випадках, коли під час використання команди plot будуються графіки у дійсній площині, уявна частина даних ігнорується. Розглянемо приклад використання команди plot(y) для побудови фігури Лісажу, які є результатом накладання гармонічних коливань, що відбуваються у повздовжньому та поперечному напрямках на кратних частотах. Такі коливання

описують фізичні явища утеоретичній електродинаміці та електротехніці, тому фігури Лісажу часто використовуються велектроніці під час моделювання коливальних процесів для подання остаточних результатів.

Приклад

```
»x=-2*pi:0.02*pi:2*pi;
»y=sin(2*x)+i*cos(3*x);
»plot(y)
```



Інший спосіб побудови геометричних фігур у системі MatLab —це параметричне визначення функцій для координат x та y. Таким чином можна моделювати траєкторії часток у фізиці і механіці. Коли розподіл сил, що діють на тіло, яке рухається, має складний характер, і функція залежності координат y(x) не може бути визначена аналітично, можна побудувати траєкторію об’єкту як параметричну криву, визначену функціями y(t) та x(t), де t — час. У цьому разі використовується стандартна графічна функція plot(x, y), де x та y — вектори однакового розміру. У наступному прикладі через параметричне визначення фігура Лісажу побудована як параметрична крива.

Приклад

```
» t=-2*pi:0.001*pi:2*pi;
» y=sin(3*t);
» x=sin(10*t);
» plot(x,y)
```

Зміна властивостей графіків через систему меню

Для встановлення та зміни властивостей графіків можна використовувати меню опрацювання об’єктів із відповідними іконками. Рядок іконок розташований над графіком. Найчастіше доводиться використовувати іконки збільшення та зменшення масштабу подання графіка.

Розглянемо приклад, як за допомогою масштабування можна аналізувати графіки, що описують складні коливання. Наведений графік функції y(t) = (sin (w2t) + a) x x sin(w1t) для a > 1, w2<< w1. Подібні гармонійні функції широко використовуються в електроніці, зокрема, у теорії сигналів оскільки за їх допомогою описується процес амплітудної модуляції. Проблема перегляду та аналізу графіків таких функцій полягає в тому, що при w2<< w1неможливо одночасно спостерігати на моніторі високочастотні та низькочастотні коливання.

Зміна масштабу подання графіка дає таку можливість. Наведемо програму мовою MatLab для аналізу цієї функції:

Приклад

```
» w1=1000; w2=10;
» t=0:0.0001:2;
» y=(sin(w2*t)+1.5).*sin(w1*t);
» plot (t,y)
```

Результат роботи програми для заданих значень частот w1, w2, у визначеному діапазоні часу, відліки якого описує вектор t. На обраному відрізку часу $t \in [0; 2]$ можна спостерігати низькочастотні коливання з частотою w2= 10 Гц, але коливання високої частоти зливаються у єдину темну смугу і спостерігати їх неможливо за умови збільшення масштабу через вибір іконки та виділення частини графіка для перегляду у збільшеному вигляді, можна спостерігати високочастотні коливання з частотою w1= 1000 Гц

Слід зазначити, що для спостерігання коливань високої частоти необхідно обирати крок часу $\Delta t \leq 0{,}1 / w1$. За меншого кроку через малу кількість точок відліку часу високочастотні сінусоїдальні коливання відображуються на графіку ламаною лінією.

За замовчуванням команда MatLab plot виводить графіки синім кольором на білому фоні, із проставленими значеннями відліків по осях координат, але без координатної сітки, підписів під осями, та без назви графіка. У системі є можливість змінити такий спосіб подання графічних даних. Один зі способів редагування графіків полягає у використанні іконок, які розташовані над графіком, та системи меню для зміни властивостей об’єктів. Насамперед необхідно вибрати об’єкт, властивості якого редагуються, за допомогою стрілки виділення об’єктів «Enableplotediting. Якщо команда редагування властивостей подається без виділення об’єкта, на екрані з’являється вікно з повідомленням про помилку.

Під час вибору об’єкта для редагування можна помітити весь графік, якщо необхідно змінити будь?які елементи графіка або осей, або підписати його. Для зміни властивостей лінії, якою нарисований графік, наприклад, її кольору або товщини, треба виділяти тільки цю лінію. У разі виділення всього графіка вікно має вигляд. Характерними тут є невеликі квадратики у кутах та посередині графіка.

Редагування властивостей здійснюється за допомогою випливаючого меню Tools. Опція «Enable plot editing» позначена, що свідчить про можливість виправлення графіка.

Підпишемо графік та його осі, а також накладемо на поле графіка координатну сітку. Після виділення графіка ці дії виконують через опцію меню «Axis properties». У результаті з’являється вікно для зміни властивостей осей графіка «Edit Axis Properties.

У верхньому текстовому вікні «Title» вказується назва графіка, нижче, у вікнах «Label» — підписи під віссю абсцис X та віссю ординат Y. У рядку «Limits» розташовані чотири вікна, призначені для визначення верхньої та нижньої границь. Ці границі визначаються за замовчуванням, і відповідні величини пишуться у текстових вікнах сірим кольором. Якщо користувач бажає

змінити визначені системою величини для покращення вигляду графіка, необхідно встановити для рядка «Limits» опцію «Manual». У рядку «Tick step» встановлюється крок для розташування відліків за осями X та Y, який також визначається системою автоматично і може змінюватися через установлення опції «Manual». У рядку «Scale» установлюється лінійна («Liner») або логарифмічна («Log») шкала відліків для осей X та Y, а також звичайний («Normal»), зворотний («Reverse») напрямок осей. Через рядок «Grid» установлюється координатна сітка для осей X та Y.

Усі опції встановлені правильно, але необхідно окремо виставити кириличні шрифти на підписах. Для виконання цієї дії слід виділити текстовий блок, та, натиснувши праву кнопку миші, викликати випливаюче меню редагування блоку, зокрема типу шрифта. При зверненні до нього виникає вікно.

У вікні виставляння типу шрифта обираємо Arial Unicode MS, кириличний набір символів Cyrillic. Можна обрати інші шрифти стандарту Unicode, якщо такі є в системі. Стандарт було розроблено для використання одного типу символів для різних мов і тепер його застосовують в ОС Windows. Після зміни властивостей текстових блоків графік буде мати такий вигляд.

Для редагування властивостей лінії необхідно виділити її та виконати команду меню Tools → Line Properties. У результаті зя’вляється вікно.

У цьому вікні можна змінити товщину лінії, задану у пунктах, її тип та колір. Для побудови графіків використовують наступні типи ліній:

• суцільна лінія (dot);
• пунктирна лінія (dash);
• точкова лінія (dot);
• штрих?пунктирна лінія (dash?dot);
• графік без лінії (none).

Для визначення кольору необхідно натиснути кнопку Select та відкрити стандартну палітру Windows.

Тип маркера визначається у тому випадку, коли кількість точок відліку невелика, та значення функції апроксимуються або інтерполюються, наприклад, під час опрацювання даних експерименту. У результаті отримуємо графік, який може бути використаний під час оформлення наукової документації.

Якщо необхідно позначити лінії на графіку та зробити біля них відповідні підписи, можна використовувати іконки для додавання лінії , лінії зі стрілкою , або текстового фрагмента

Ці іконки стандартні для більшості графічних редакторів. Графік із доданими позначеннями та підписами на його полі

Зміна властивостей та групування графіків через командні рядки

Визначення властивостей графіка через систему меню зручне у випадку одноразового розв’язання нескладних задач. Але у разі написання програм із виведенням графічних результатів на екран, і, особливо, у разі подальшого їх використання для модельних експериментів, постійно змінювати графіки через систему меню незручно. Крім того, у розглянутому форматі команда plot не дає змоги поєднувати кілька графіків на одному полі або декілька полей графіків на

одному малюнку, що часто потрібно для оформлення графічної документації. Розглянемо простий приклад. Якщо виконати послідовність таких команд:

```
» t=0:0.0001*pi:2*pi;
» y=sin(t); z=cos(t);
» plot (t,y); plot (t,z);
```

будуть створені два графічні вікна, в одному з яких виведений графік синуса, а у другому — графік косинуса. А при модельних експериментах часто треба поєднувати графіки обчислених функцій на одній координатній площині з метою порівняння їх властивостей та пошуку оптимального розв’язання. Для побудови двох графіків в одній системі координат використовують команду MatLab hold. Вона має два параметра і формат її досить простий: holdon— увійти урежим виведення графіків на одному полі;

hold off — вийти з режиму виведення графіків на одномуполі.

Інший спосіб поєднання графіків — це використання спеціального формату команди plot:

plot (вектор змінної, вектор функція 1, вектор змінної, вектор функція 2, …)

Зверніть увагу, що при використанні такого формату команди всі функції необхідно обчислювати від одного вектора змінної. Таким чином, є два способи, за допомогою яких можна змінити написану вище програму з метою виведення графіків двох функцій в одному графічному вікні.

Приклад:

```
» t=0:0.0001*pi:2*pi;
» y=sin(t); z=cos(t);
» hold on;
» plot (t,y);
» plot (t,z);
```

Перший спосіб зручно використовувати у разі поєднання великої кількості графіків, оскільки всі команди plot можна писати окремими рядками. Якщо кількість графіків невелика, то можна використовувати і другий спосіб.Після побудови графіків для зміни властивостей можна використати такі команди:

grid on — увійти у режим відображення сітки на графіку;

grid off— вийти зрежиму відображення сітки на графіку;

title(‘Текст заголовка графіка’)— команда виводить рядок, наведений у лапках, як заголовок графіка;

xlabel (‘Вісь X’) — команда виводить рядок, наведений у лапках, як заголовок осі x;

ylabel (‘Вісь Y’) — команда виводить рядок, наведений у лапках, як заголовок осі y;

figure — команда формує нове графічне вікно для виведення графіків.

Програма для виведення графіків синуса і косинуса в одній системі координат має такий вигляд:

Приклад

```
» t=0:0.0001*pi:2*pi;
» y=sin(t); z=cos(t);
```

```
» hold on;
» plot (t,y); plot (t,z);
» grid on;
186
» t=0:0.0001*pi:2*pi;
» y=sin(t);
» z=cos(t);
» plot (t,y,t,z);
» title (‘Графіки синуса і косинуса’);
» xlabel (‘Вісь часу’);
» ylabel (‘Графіки функцій sin(t) та cos(t)’);
```

Слід зазначити, що за замовчуванням функції виведення тексту title, xlabel та ylabel не використовують кириличні шрифти, тому поки будемо користуватися графічними засобами зміни типу шрифта. Оскільки робота із системою меню має низку незручностей під час використання налагоджених програм для модельних експериментів, передбачена можливість зміни типу шрифта через командний рядок. Цю дію виконують за допомогою дескрипторної графіки, і вона буде описана у наступному підрозділі. У цьому разі на екрані перша крива відображується синім, адруга — зеленим кольором.

Інший ефективний спосіб групування графіків — це функція MatLab subplot, за допомогою якої можна розташовувати графіки на одному рисунку для різних площин із однаковими або різними осями координат. Таке подання графічних даних також важливе і чйого застосовують під час оформлення науково-технічної документації. Формати команди subplot:

subplot — створити нове вікно для виведення графічних даних. У цьому разі надані раніше команди subplot ігноруються і відкривається одне поле для виведення наступного графіка. Команда дублює figure за винятком того, що отримані раніше дані знищуються. Використовується рідко;

subplot (m, n, p)— команда розбиває площину графічного вікна на m x n полів для виведення графіків, де m— кількість рядків, n — кількість графіків в одному рядку, p — номер графіка.

Нумерація графіків здійснюється наскрізно рядками розглянемо роботу послідовностей команд plot та subplot на прикладі. У цьому разі будемо наводити вигляд графічного вікна після виконання фрагментів програми, які його змінюють.

```
» x=0:0.001*pi:10;
» subplot(2,2,1);
```

Рисунок розбито на чотири частини, є два вікна для виведення графіків зверху та два знизу. Активне перше вікно, розташоване у лівому верхньому куті. Інші вікна поки

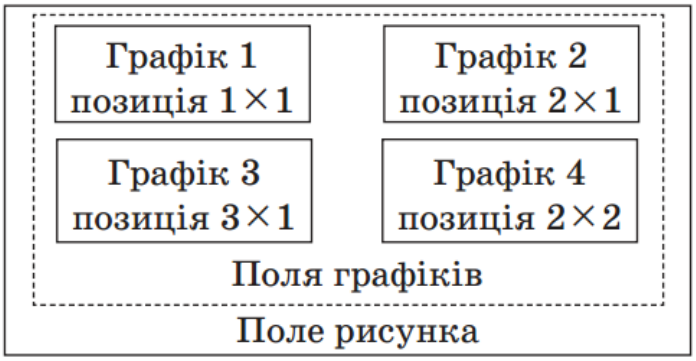
не визначені, простір, відведений для них — порожній. Побудуємо графік в активному вікні, виконуючи наступні команди:

Приклад

```
» plot(x,sin(x));
» grid on;
```

```
» xlabel(‘x’);
» ylabel(‘sin(x)’);
» title (‘Графік функції y=sin(x)’);
```

На другому графіку розташований поряд із першим за горизонталлю, побудуємо дві криві, графік синуса, та графік косинуса. Це можна зробити за допомогою таких команд:



Приклад

```
» subplot(2,2,2);
» hold on;
» plot(x,sin(x));
» plot(x,cos(x));
» grid on;
» xlabel(‘x’);
» ylabel(‘sin(x)?? cos(x)’);
» title (‘Графіки функцій y=sin(x) та y=cos(x)’);
```

Після виконання команд графічне вікно має вигляд, наведений на рис.

На третьому графіку побудуємо коло через параметричне визначення функції. Послідовність команд може бути такою.

Приклад

```
» subplot(2,2,3);
» plot(cos(x),sin(x));
» grid on;
» xlabel(‘x=sin(t)’);
» ylabel(‘y=cos(t)’);
» title (‘Коло – параметрична функція’);
```

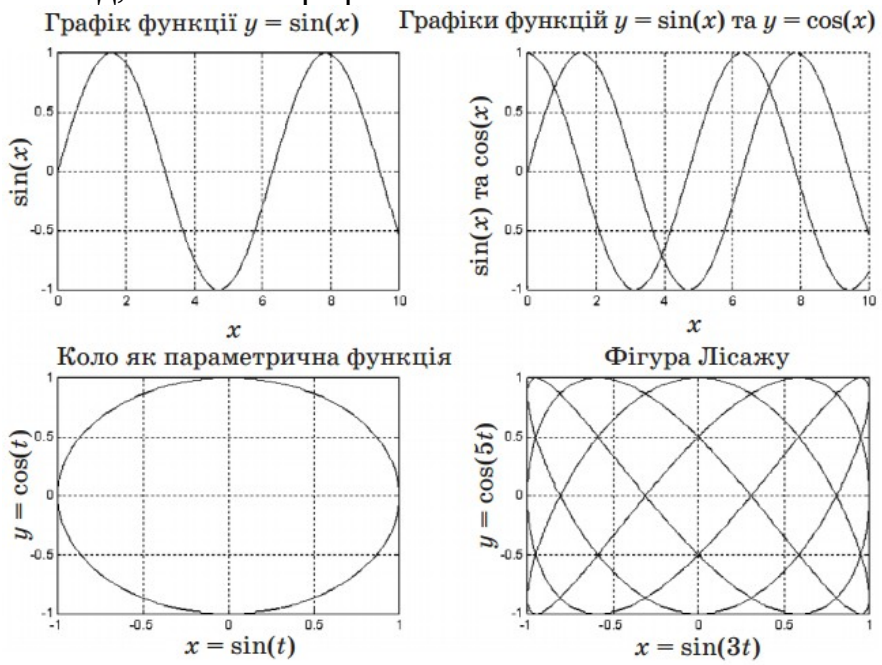
Побудуємо на четвертому графіку фігуру Лісажу, через таку послідовність команд:

Приклад

```
» subplot(2,2,4);
» plot(cos(3*x),sin(5*x));
» grid on;
» xlabel(‘x=sin(3t)’);
» ylabel(‘y=cos(5t)’);
» title (‘Фігура Лісажу’);
sin(x)
```

```
sin(x) та cos(x)
```

Графік функції y = sin(x) Графіки функцій y = sin(x) та y = cos(x)
Вигляд, який має графічне вікно після виконання команд



Зміна властивостей графіків через параметри функції plot

Наведемо ще один формат запису функції plot з трьома параметрами: plot(x,y,s) Тут x та y — матриці або вектори однакового розміру. s — рядкова константа. Значення константи s повинні відповідати стандартам, уведеним розробниками системи, і визначають властивості маркерів для точок відліку та ліній на графіку. Тобто, через значення s користувач може змінити вигляд графіка, дотримуючись стандартів щодо оформлення науково-технічної документації. Наведені можливі значення рядкової константи s для різних параметрів ліній та маркерів.

Рядкова константа може мати три компоненти: «k+–»: чорні суцільні лінії, тип маркера «+». Такий самий результат буде, якщо написати «–k+» або «+–k». Якщо будь?який компонент не визначений, він використовується за замовчуванням.

Для побудови графіків кількох функцій на одному полі з установленням властивостей ліній та маркерів можна використовувати інший формат команди plot. Він схожий на формат команди, призначеної для побудови декількох графіків, без встановлення властивостей, але для кожного набору даних додається рядкова константа, яка визначає параметри графіка. Формат команди:

plot(x1, y1, s1, x2, y2, s2…)

Розглянемо приклад побудови графіка трьох функцій з різним стилем подання.

Приклад

```
»x=-2*pi:0.1*pi:2*pi;
»y1=sin(x);
»y2=sin(x).^2;
```

```
»y3=sin(x).^3;
»plot(x,y1,’?k’,x,y2,’?.+k’,x,y3,’–ok’)
»grid on;
»xlabel(‘x’);
»ylabel(‘y’);
»title (‘Графіки функцій sin^n(x)’);
```

У цьому випадку рядкова константа s1 містить тільки два символи, тому тип маркера визначається за замовчуванням. Графік креслиться без маркера. Крім того, визначений чорний колір лінії на всіх трьох графіках. Зазвичай таку процедуру виконують для спрощення виведення на чорно-білий друк та читання роздрукованих даних. Щоб розрізняти такі графіки, використовують різні типи ліній та маркерів. Тому графік 1 побудований суцільною лінією без маркерів, графік 2 — штрих пунктирною лінією з маркером у вигляді перехрестя, а графік 3 — штриховою лінією з маркером у вигляді кола.

Слід зазначити, що якщо використання різних стилів ліній зручне для розрізнення графіків, то маркери та їх стилі — не кращий спосіб подання розрахункових даних. Зазвичай для поліпшення оцінок властивостей функції при проведенні розрахунків беруть велику кількість значень аргумента. За малої кількості розрахункових точок оцінки функцій можуть бути неправильними, наприклад, можна пропустити екстремуми або інші характерні точки. У тому випадку, коли поведінка функції точках відліку визначена правильно, точність лінійної інтерполяції між відліками за малої їх кількості невисока і отримані ламані лінії уможливлюють оцінювання значення між точками відліку з грубою помилкою. Приклад такого подання функції модульованих синусоїдальних коливань. Неточність початкового обчислення значень функції може вплинути на подальші розрахунки.

У наведеному вище прикладі використана 41 розрахункова точка на інтервалі [–2π; 2π], і в цьому разі графіки з маркерами, наведені на мають досить гарний вигляд. З іншого боку, для всіх трьох функцій на один півперіод випадає по 10 точок відліку, що достатньо для уникнення ламаних ліній. Збільшення точок відліку призводить до погіршення вигляду графіків через велику кількість маркерів, а зменшення — до неточного подання проміжних даних ламаними лініями. Графіки для кроків по осі 0,01π та 0,3π.

Під час складних розрахунків десяти точок на півперіод функції недостатньо, оскільки точність визначення початкових даних суттєво впливає на остаточні розрахунки. Тому розрахункові дані у науково?технічній документації рідко позначають маркерами. Частіше таке подання використовують для експериментальних даних, коли за їх невеликої кількості для оцінки значень функції між визначеними точками застосовують методи апроксимації.

Недоліком подання даних є відсутність підписів, які б розшифровували належність кривих до тієї чи іншої функції. У науково-технічній документації такі підписи називають легендами і розташовують їх на полі графіка. Іншим способом розшифрування кривих є написання цифр над ними, але для цього необхідно виводити текст точно над кривою, а таку операцію дуже важко автоматизувати. У наступному параграфі розглядатиметься можливість розташування текстового блоку за конкретними координатами, але в разі автоматизованої побудови

графіків функцій частіше користуються легендами. Сформувати і розташувати легенду на полі графіка просто — для цього використовують команду legend. Формат команди:

legend (s1, s2, s3 …)

де s1, s2, s3 — рядкові константи. Записані символи будуть виведені як підписи для кожної кривої, тому кількість кривих і констант повинна бути однаковою. Зверніть увагу, що символу ^у підписах на графіках відповідає верхній індекс, а символу під креслення — нижній, що значно спрощує читання математичних виразів. Крім того, у легенді для ідентифікації кривої виводиться тип маркера та тип лінії, які їй відповідають. Для нашого випадку команду формування легенди можна записати у вигляді:

legend (‘y_1=sin(x)’, ‘y_2=sin^2(x)’, ‘y_3=sin^3(x)’)

Є інші формати команди legend:

legend on — виведення легенди на поточний графік;

legend off — вилучення легенди з поточного графіка.

legend (s1, s2, s3 …, p) — розташування легенди на заданій позиції, визначеній параметром p, який може мати такізначення:

0 — легенда автоматично розташовується на полі графіка там, де вона не перетинається з жодною кривою;

1 — легенда розташовується у верхньому правому куті, цей параметр використовується за замовчуванням;

2 — легенда розташовується у верхньому лівому куті;

3 — легенда розташовується у нижньому лівому куті;

4 — легенда розташовується у нижньому правому куті;

–1 — легенда розташовується праворуч від поля графіка.

Зміна властивостей побудованих графічних об’єктів за допомогою дескрипторної графіки

У попередньому параграфі розглянуті можливості встановлення властивостей графічних об’єктів через параметри функції plot, але відразу зрозумілі недоліки такого інструментарію. Насамперед, не всі властивості об’єктів розробники визначили через текстові константи. І по друге, здебільшого користувачі бажають послідовно змінювати вигляд отриманого графіка, як ми зробили це у попередньому параграфі. А такі дії зручніше виконувати не повторною побудовою графіка, а зміною його властивостей та введенням додаткових об’єктів.

Зрозуміло, що деякі з цих дій можна виконати за допомогою графічного інтерфейсу, але далеко не всі. У програмуванні є інша можливість, яка полягає у використанні методів та засобів об’єктно-орієнтованого програмування (ООП). Базовим

поняттям теорії ООП є об’єкт, який відрізняється від звичайних структур даних тим, що поєднує всобі властивості, які визначаються через значення змінними, та методи, реалізовані через процедури і функції. Методологія ООП почала розвиватися із середини вісімдесятих років минулого століття, і з її впровадженням робота програмістів стала набагато легшою. Особливо це

позначилося на написанні складних програмних комплексів, швидкості їх налагодження та доведення до готових комерційних продуктів.

Значно зменшилася кількість помилок у складних програмах. Методи та засоби ООП впроваджені в системі MatLab. Але ООП спрощує і роботу користувачів, надаючи можливість ефективно змінювати та аналізувати отримані результати не через пошук ланцюгів відповідних змінних і аналіз їх значень, а через зміну властивостей об’єктів. Тобто, аналізуючи властивості, користувач їх змінює, і всі необхідні дії при цьому виконуються автоматично.

Ці засоби ефективно працюють з об’єктами, які мають складну структуру і легко розчеплюються на окремі елементи. Дуже складно структуризувати та підвести під ідеологію ООП складні фізичні та біологічні об’єкти, які утворюють одне ціле і мають складні неупорядковані зв’язки між окремими елементами. Легко піддаються структуризації геометричні об’єкти, електронні системи, інформаційні структури, комп’ютерні системи. Оскільки геометричні об’єкти серед перелічених є найбільш простими і легко піддаються структуризації, розглянемо простоту та ефективність використання ООП у MatLab саме на них. Через ООП та зміну властивостей об’єктів у MatLab реалізована дескрипторна графіка, засоби якої фахівці відносять до низькорівневих операцій. Основи дескрипторної графіки полягають у тому, що користувач працює не зіскладними структурами, як то графік або графічне вікно, азбазовими об’єктами, кожен зяких налаштовується потрібним чином зміною його властивостей. Глибоке вивчення дескрипторної графіки MatLab є вкрай важким завданням, оскільки налічується понад что окремих графічних об’єктів. Але при роботі з невеликою кількістю простих об’єктів дескрипторний метод є простим і зручним, навіть простішим, ніж високорівневі методи опрацювання графічних об’єктів. Наявність системи допомоги дає змогу вдосконалювати свої знання про властивості відомих графічних об’єктів та вивчити нові.Наведемо простий приклад дескрипторної графіки, в основі якого лежить функція plot. Важлива відмінність полягає у тому, що цій функції присвоюється значення змінної.

```
x=0:0.01*pi:3*pi;
y=sin(x);
c=plot(x,y);
grid on;
```

Поки подання графіка функції y=sin(x),отриманого з використанням наведеної системи команд, нічим не відрізняється від стандартного, графік будуть у декартовій системі координат тонкою блакитною лінією. Але важливим є те, що можна переглянути властивості отриманого графічного об’єкта та змінити їх. Перегляд властивостей здійснюють за допомогою команди get (c), де c — ім’я змінної, яка відповідає функції plot. Для розглянутого прикладку результат виконання команди буде таким:

Приклад

```
» get (c)
Color = [0 0 1]
EraseMode = normal
LineStyle = ?
```

```
LineWidth = [0.5]
Marker = none
MarkerSize = [6]
MarkerEdgeColor = auto
MarkerFaceColor = none
XData = [ (1 by 301) double array]
YData = [ (1 by 301) double array]
ZData = []
ButtonDownFcn =
Children = []
Clipping = on
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = on
HitTest = on
Interruptible = on
Parent = [73.0009]
Selected = off
SelectionHighlight = on
Tag =
Type = line
UIContextMenu = []
UserData = []
Visible = on
»
```

Кількість властивостей графіка досить велика, але на початку перелічені головні з них, які кожен користувач зрозуміє інтуїтивно. Перелічимо параметри графіка та розглянемо способи їх подання у числовому варіанті.

Color — цей параметр визначає колір лінії через вектор, який містить три компоненти: [r, g, b]. Компонент r відпові дає червоному, g — зеленому, b — синьому кольору. За замовчуванням параметр має значення [0, 0, 1], тому початково лінії графіка кресляться синім кольором;

EraseMode — параметр визначає порядок зміни графіків за допомогою функції get. За замовчуванням EraseMode=‘normal’, цей параметр відповідає виведенню нового графіка відразу після зміни його параметрів. У разі використання значення EraseMode=‘none’ для зміни зображення необхідно використовувати функцію drawnow;

LineStyle — стиль лінії, який визначається текстовими константами.

LineWidth — товщина лінії у пунктах, за замовчуванням має значення 0,5;

Marker— визначення типу маркера відповідно до текстових констант, наведених у таблиці 3.6. За замовчуванням Marker =‘none’, тобто відсутній;

MarkerSize — розмір маркера, за замовчуванням Marker Size = 6.

Оскільки, як було зазначено у попередньому параграфі, маркери на теоретичних графіках використовуються рідко, найчастіше змінюються параметри лінії. Наприклад, установлюється чорний колір лінії, збільшується її

товщина для більшого контрасту та виразності графіка. Простіше це робити тепер через дескрипторну графіку, адже визначена константа c, що формує массив даних, який описує властивості графіка, і необхідно тількі її змінити. Для цього не треба глибоких знань об’єктно орієнтованого програмування, як під час написання програм мовами C++, Object Pascal або Java. У MatLab зміна властивостей здійснюється дуже просто через команду set. Формат команди:

set(посилання на об’єкт через змінну, ‘параметр1’,значення1…)

Наприклад, для зміни кольору та товщини лінії необхідновикористовувати таку команду:

set (c, ‘Color’, [0,0,0], ‘LineWidth’,3)

Тут константа c — дескрипторна змінна, яка попередньо визначається через графічну функцію та дає посилання на відповідний графік. Дескрипторні змінні також називають описами.

Оскільки англійська назва дескриптора — handle, досвідчені програмісти починають імена таких змінних з літери h, але цього правила дотримуватися не обов’язково. Параметру ‘Color’ надається значення [0, 0, 0], яке відповідає чорному кольору,а параметру ‘LineWidth’ надано значення 3. Після виконання цієї команди графік стає контрастним.

Розглянемо іншу можливість ефективного використання дескрипторної графіки — зміну типу та розміру шрифта під час виведення текстів. Такий спосіб не є ефективним, кращим є використання засобів програмування. Як було показано раніше, для нормального читання підписів на графіках треба застосовувати шрифти стандарту Unicode, наприклад, @Arial Unicode MS. Як змінити тип тарозмір шрифта засобами дескрипторної графіки, якщо ми не знаємо змінних, які відповідають за ці властивості? Стане у нагоді команда get, яку ми використовували у попередньому прикладі.

Приклад

```
»ht=title (‘Графік функції y(x)=sin(x)’);
»get (ht)
Color = [0 0 0]
EraseMode = normal
Editing = off
Extent = [?1.74194 1.0117 3.4424 0.0643275]
FontAngle = normal
FontName = Helvetica
FontSize = [10]
FontUnits = points
FontWeight = normal
HorizontalAlignment = center
200
201
Position = [?0.00692841 1.02053 17.3205]
Rotation = [0]
String = Графік функції y(x)=sin(x)
Units = data
Interpreter = tex
```

```
VerticalAlignment = bottom
ButtonDownFcn = scriberestoresavefcns
Children = []
Clipping = off
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = off
HitTest = on
Interruptible = on
Parent = [72.0009]
Selected = off
SelectionHighlight = on
Tag =
Type = text
UIContextMenu = []
UserData = []
Visible = on
```

Уважно прочитавши імена та значення всіх змінних, знайти серед них необхідні досить легко — це змінні

```
FontName =
Helvetica та FontSize = [10].
»set(ht,’FontName’,’@Arial
Unicode
MS’,
‘FontSize’,12);
```

Зробимо схожі підписи на осях графіка, відповідні команди повинні бути такими: Приклад

```
»ht2=xlabel (‘?’);
»set(ht2,’FontName’,’@Arial Unicode MS’, ‘Font?
Size’,12);
»ht3=ylabel (‘y(?)=sin(x)’);
»set(ht3,’FontName’,’@Arial   Unicode   MS’,   ‘Font?-
Size’,12);
```

Досконаліше дескрипторні змінні можна вивчити, використовуючи команду get та систему допомоги. Ми ще неодноразово звернемося до засобів дескрипторного опису під час вивчення особливостей інших функцій двовимірної та тривимірної графіки, а також під час вивчення засобів проектування графічних інтерфейсів користувача.

Спеціальні функції двовимірної графіки у системі MatLab

Побудова графіків у логарифмічному та напівлогарифмічному масштабах

Для побудови графіка у логарифмічному масштабі у MatLab використовуються 2 основні функції:

logspace(x1, x2) — завдання логарифмічного масштабу на інтервалі [x1, x2] для зміни значення аргументу;

loglog(x,y(x))— завдання логарифмічного масштабу для діапазону зміни функції y(x) та виведення графіка. Команда аналогічна команді plot.

Приклад побудови графіка функції, близької до експоненціальної:

Приклад

```
»x=logspace(?1,3);
»loglog(x,exp(x)./x);
»grid on
```

У деяких випадках більш зручним є напівлогарифмічний масштаб графіків, коли на одній осі використовується логарифмічна, а на іншій — лінійна шкала. Під час роботи з графіками у напівлогарифмічному масштабі використовують дві команди, кожна з яких установлює логарифмічний масштаб заоднією з осей координат:

semilogx(x, y) — будує графік функції y(x) у логарифмічному масштабі з основою 10 у координаті x та лінійному в координаті y;

semilogy(x, y) — будує графік функції y(x) в логарифмічному масштабі з основою 10 у координаті y та y лінійному у координаті x.

Розглянемо приклад, у якому графік функції y(x) = exp(x) / xпобудований тричі: у логарифмічному масштабі, у напівлогарифмічному масштабі у осі x та y напівлогарифмічному масштабі в осі y. Сукупність графіків, яка є результатом виконання цього прикладу

Приклад

```
» subplot (3,1,1);
» hc1=loglog(x,exp(x)./x);
» grid on;
204
205
» set (hc1,’Color’,[0,0,0],’LineWidth’,2);
» subplot (3,1,2);
» hc2=semilogx(x,exp(x)./x);
» set (hc2,’Color’,[0,0,0],’LineWidth’,2);
» grid on;
» subplot (3,1,3);
» hc3=semilogy(x,exp(x)./x);
» set (hc3,’Color’,[0,0,0],’LineWidth’,2)
» grid on;
```

Стовпчикові діаграми та гістограми

Стовпчикові діаграми широко використовують в економіці та математиці. Для побудови стовпчикових діаграм використовують функцію bar різних модифікацій. Наприклад, функція bar(x, y) будує стовпчиковий графік значень вектора y із описанням положення та ширини стовпчиків, заданих значеннями вектора x, при цьому значення елементів вектора x повинні зростати. Розглянемо приклад побудови графіка функції e–x/2y вигляді стовпчикової діаграми в діапазоні значень x є [–2,9; 2,9] з кроком 0,2.

Приклад

```
»x=-2.9:0.2:2.9;
»hc=bar(x,exp(?x.*x));
»get (hc)
CData = [ (4 by 30) double array]
CDataMapping = scaled
FaceVertexCData = [ (151 by 1) double array]
EdgeColor = [0 0 0]
EraseMode = normal
FaceColor = flat
Faces = [ (30 by 4) double array]
LineStyle = ?
LineWidth = [0.5]
Marker = none
MarkerEdgeColor = auto
MarkerFaceColor = none
MarkerSize = [6]
Vertices = [ (151 by 2) double array]
XData = [ (4 by 30) double array]
YData = [ (4 by 30) double array]
ZData = []
FaceLighting = flat
EdgeLighting = none
BackFaceLighting = reverselit
AmbientStrength = [0.3]
DiffuseStrength = [0.6]
SpecularStrength = [0.9]
SpecularExponent = [10]
SpecularColorReflectance = [1]
VertexNormals = [ (151 by 3) double array]
NormalMode = auto
ButtonDownFcn =
Children = []
Clipping = on
CreateFcn =
DeleteFcn =
BusyAction = queue
```

```
HandleVisibility = on
HitTest = on
Interruptible = on
Parent = [72.0009]
Selected = off
SelectionHighlight = on
Tag =
Type = patch
UIContextMenu = []
UserData = []
Visible = on
»set (hc,’FaceColor’,’black’);
»title(‘Стовпчиковадіаграмафункціїy(x) = exp(–x^2)’);
»xlabel (‘Аргумент функції’);
»ylabel (‘Значення функції’);
```

Розглянемо тепер приклад побудови гістограм розподілу випадкових величин. Формат функції побудови гістограми hist такий:

hist(y, x)

де y— вектор, для якого необхідно побудувати гістограму, x— інший вектор, що визначає інтервали зміни значень вектора y, для яких підраховується кількість значень y до заданого інтервалу. Важливо, що функція hist виконує одразу дві дії: будує стовпчикову діаграму для підрахованого числа елементів як Стовпчикова діаграма функції y(x) = exp(–x2) Значення функції Аргумент функції функцію зазначених вектором x діапазонів, і повертає вектор кількості елементів для кожного діапазону. Для виконання цієї дії функція hist записується в іншому форматі:

v = hist(y, x)

де v — вектор результату.

Побудуємо гістограму розподілу випадкових чисел, які генеруються математичною функцією randn.

Приклад

```
»x=-3:0.2:3;
»y=randn(1000,1);
»hist(y,x);
»grid on
»h=hist(y,x)
Columns 1 through 12
0  0  3  7  8  9  11  23  33  43  57  55
Columns 13 through 24
70  62  83  93  68  70  65  41  35  27  21
Columns 25 through 31
12  5  6  3  2  1  0
```

Функція randn генерує 1000 випадкових чисел на інтервалі [–3; 3]. У результаті роботи цієї програми з’являється гістограма.

З невеликими відхиленнями закон розподілу згенерованої випадкової величини близький до нормального. Якщо збільшити кількість чисел, які генеруються, збіг із нормальним законом буде більшим,яка є результатом виконання таких командних рядків:

Приклад

```
»x=?3:0.2:3;
»y=randn(1e6,1);
»hist(y,x);
»grid on
```

Сходинкові графіки

У фізиці та електроніці часто необхідно зображувати графіки функцій у вигляді сходинок, які проходять через дискретні відліки. Такі графіки відображають процес квантування функцій по одній зі змінних, у цьому разі між точками відліків значення функції вважаються постійними. Для побудови таких графіків у MatLab є функція stairs. Можливі формати функції:

stairs(y) — побудова сходинкового графіка за даними вектора y. Значення аргументу відповідає порядковому номеру даних;

stairs(y, x) — будує сходинкову функцію y(x), зміна значень аргументу задається вектором x.

Приклад

```
»x=0:0.1*pi:4*pi;
»y=sin(x);
»hc=stairs(y,x);
»set (ch,’Color’,[0,0,0],’LineWidth’,2);
```

У наведеному графіку дискретизація значень осі абсцис є постійною величиною, висота сходинок змінюється залежно відфункції.

Приклад

```
»subplot(2,1,1);
»x=0:0.01*pi:4*pi;
»y=sin(x);
»ch=stairs(x,y);
»grid on
»set (ch,’Color’,[0,0,0],’LineWidth’,2);
»v=title(‘???? 0.01*pi’);
»set   (v,’FontName’,’@Arial   Unicode   MS’,   ‘Font-
Size’,12);
»subplot(2,1,2);
»x=0:0.3*pi:4*pi;
»y=sin(x);
»ch=stairs(x,y);
»grid on
```

```
»set (ch,’Color’,[0,0,0],’LineWidth’,2);
»v=title(‘ ???? 0.01*pi’);
»set (v,’FontName’,’@Arial Unicode MS’, ‘Font?
Size’,12);
```

Сходинкові функції відповідають інтерполяції точок відліку функціями нульового порядку, тобто постійними значеннями, а команда MatLab plot будує графіки з інтерполяцією першого порядку, тобто ламаними лініями. Сходинкові функції, широко використовуються в електроніці для аналізу аналого-цифрових та цифроаналогових перетворень. Недоліком подання сходинкових функцій через команду MatLab stairs є неможливість дискретизації рівнів відліку за оссю ординат, тому реальні сходинкові функції здискретизацією рівнів по обох осях, необхідно будувати засобами програмування, використовуючи при обчисленнях функції округлення

Побудова графіків із зонами похибки даних

З урахуванням того, що MatLab — це система для професіоналів-програмістів, і її широко використовують фізики та математики в наукових установах, зокрема для опрацювання експериментальних даних, у системі введена можливість побудови графіків із відображенням похибок даних. Для такого подання даних необхідно спочатку для кожної точки відліку сформувати вектор похибки. Формати функції errorbar, призначеноїдля побудови графіка з похибками:

errorbar(x, y, l, u) — будує графік значень функції для елементів векторів y(x), l відповідає нижній границі похибки,u — верхній;

errorbar(x, y, e) — будує графік залежності y(x) із указівкою границь у симетричному діапазоні [y – e; y + e], де e— похибка. Наведемо приклад програми для побудови графіка з похибкою:

Приклад

```
»x=?2:0.1:2;
»y=erf(x);
»e=rand(size(x))/10;
»errorbar(x,y,e)
```

Графіки дискретних відліків функції

Під час описування квантування сигналів і в деяких інших випадках зручно подавати отримані дані графіками дискретних відліків. У цьому разі кожен відлік функції відображується вертикальною рискою, а положення рисок відповідає значенням функції. У MatLab такі графіки будуються з використанням команди stem. Є 2 формати команди stem:

stem(y) — будує функцію значень вектора y, аргументи функції — номери значень;

stem(x, y) — будує функцію y(x), діапазон зміни аргументу задається значеннями x.

Приклад

```
»x=0:0.1:4;
»y=sin(x.^2).*exp(?x);
»stem(x,y)
```

При побудові графіків за допомогою функції stem необхідно пам’ятати про те, що надмірно висока щільність ліній не даєзмоги зрозуміти ці графіки.

Побудова графіків у полярній системі координат

Для побудови графіків у полярній системі координат використовують функцію polar. Є два формати цієї функції:

polar(theta, ro) — побудова функції ρ(θ);

polar(theta, ro, s) — побудова функції ρ(θ), при цьому стиль лінії та маркера задається константою s, як було описано для функції plot. Наведемо приклад використання функції polar:

Приклад

```
»t=0:pi\100:2*pi;
»polar(t,sin(3*t),’?k’);
» hold on
»polar(t,sin(5*t), ‘:k’);
»polar(t,sin(7*t), ‘?.k’);
```

Під час розв’язання задач електротехніки, теорії поля, теорії хвильових процесів, часто зручніше подавати послідовність відліків функції у полярній системі координат у вигляді стрілок, які виходять із початку координат. Такий спосіб побудови відповідає графікам дискретних відліків функцій у декартовій системі координат. У MatLab відповідні графічні побудови здійснюються з використанням функції compass. Оскільки вектор у полярній системі координат можна подавати через комплексне число, у MatLab побудова векторних діаграм орієнтована саме на використання комплексних чисел. Команда compass маєтакі формати:

compass(u,v)— функція будує радіус?вектор комплексногочисла u + i•v, де u та v — вектори або матриці одного розміру;

compass(Z) — будує радіус?вектори комплексних чисел Z.

Приклад

```
»Z=[?1+2*i,?2?3*i,2+3*i,5+2*i];
»compass(Z,’?k’)
```

Побудова графіків від аналітично заданої функції

На прикладі функції y(x) = 1 / x3було показано, що для елементарних функцій система MatLab, на відміну від системи MathCAD, завжди чітко відображує особливі точки та розриви. Але для спеціальних функцій точки розриву під час побудови графіків можуть бути відображені неправильно, Це стосується жорстких функцій, коли значення похідних біля точок розриву різко змінюються від нуля до нескінченності. Після побудови графіка таких функцій за значеннями не відображується поведінка їх біля точок розриву. Крім того, іноді в програмах під час проведення складних розрахунків необхідно переглянути графік аналітично заданої функції, без обчислення її значень.

У цьому випадку використовують команду ezplot, яка на відміну від функції plot, будує графіки аналітично заданих функцій у вказаному діапазоні аргументу без попереднього формування вектора значень функції. Функція задається через рядкову константу f. Формати команди ezplot:

ezplot(f)— побудова графіка заданої функції f(x)на відрізку [–2π; 2π];

ezplot (f, [xmin, xmax, ymin, ymax]) — побудова графіка заданої функції f(x) на відрізку [–xmin; xmax];

ezplot (f, [xmin, xmax, ymin, ymax]) — побудова графіка заданої функції f(x) на відрізку [–xmin; xmax] у діапазоні значень [–ymin; ymax];

ezplot (x, y) — побудова графіка параметричної функції по значеннях аналітично заданих функцій x(t) та y(t).

При використанні команди ezplot відразу виводяться підписи на осях координат та заголовок графіка, а координатна сітка за замовченням не виводиться.

Приклади використання команди ezplot:

```
ezplot('cos(x)')
ezplot('1/y?log(y)+log(?1+y)+x ? 1',[?1.5,1.5])
ezplot('sin(3*t)*cos(t)','sin(3*t)*sin(t)',[0,pi])
```

Систематизація графічних команд двовимірної графіки

Команди побудови двовимірних графіків
bar
Побудова гістограм за вказаними значеннями
compass
Побудова векторних діаграм
errorbar
Побудова графіків із похибками даних
ezplot
Побудова графіків від аналітично заданої функції
hist
Побудова гістограм випадкових величин
plot
Будує двовимірні графіки у декартових та параметричних координатах
polar
Побудова графіків у полярних координатах

stairs
Побудова сходинкових графіків
steam
Побудова дискретних відліків функції
Команди побудови графіків у логарифмічному масштабі
logspace
Створення логарифмічного масштабу для зміни значення аргументу
loglog
Створення логарифмічного масштабу для зміни значення функції
semilogx
Будує графік функції y(x) у логарифмічному масштабі по координаті x
semilogy
Будує графік функції y(x) у логарифмічному масштабі по координаті y
Команди оформлення графіків
figure
Відкриття нового рисунка, старий рисунок зберігається
grid on
Включення режиму відображення сітки на графіку
grid off
Виключення режиму відображення сітки на графіку
hold on
Включення режиму поєднання двох кривих на координатній площині
hold off
Виключення режиму поєднання двох кривих на координатній площині
legend
Виведення легенди на поле графіка
subplot
Оформлення кількох графіків на одному рисунку
text
Формування підпису на полі графіка, положення підпису задається координатами
title
Формування заголовка графіка
xlabel
Формування підпису на осі абсцис
ylabel
Формування підпису на осі ординат Команди дескрипторної графіки
get
Відображення властивостей графічного об’єкта
set
Зміна властивостей графічного об’єкта

Основні функції тривимірної графіки

Створення масивів даних для визначення функцій від двох аргументів

Специфіка побудови тривимірних графіків полягає у тому, що завдання значень змінних x і y недостатньо. Для функції двох аргументів z(x, y) необхідно визначити двовимірний масив.

z = f(x, y) за двовимірним масивом значень x та y У системі MatLab для визначення двовимірних масивів із векторів використовується функція meshgrid. Формат її запису:

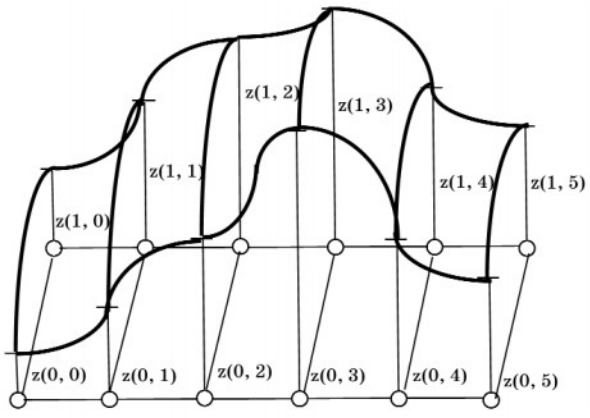
[X, Y]=meshgrid(x,y)— перетворити область, задану векторами x та y на двовимірні масиви X та Y, які можуть бути використані для обчислення функції двох змінних та для побудови її графіка. Рядки вихідного масиву Xє копіями вектора x, а стовпчики вихідного масиву Y — копіями вектора y.

[X,Y]=meshgrid(x) — команда схожа з попередньою, для формування стовпчиків масиву Y використовується вектор x.

Наведемо приклади використання команди meshgrid:

Приклад

```
»[X,Y]=meshgrid(1:4,13:17)
x=
1 2 3 4
1 2 3 4
1 2 3 4
218
219
1 2 3 4
Y=
13 13 13 13
14 14 14 14
15 15 15 15
16 16 16 16
```



»

Інший приклад визначення сітки для тривимірної функції:

```
»[X,Y]=meshgrid(-2:0.01:2);
```

Цей приклад більш досконалий і придатний для використання, оскільки значення аргументів функції табулюються малим кроком. Як і увипадку здвовимірною графікою, наведеному в попередньому підрозділі, це дає змогу уникнути похибок при інтерполяції поверхонь ламаними лініями. Наочність

попереднього прикладу полягає лише в тому, що двовимірну матрицю з малою кількістю елементів можна легко переглянути на екрані монітора.

Побудова аксанометрії тривимірних поверхонь

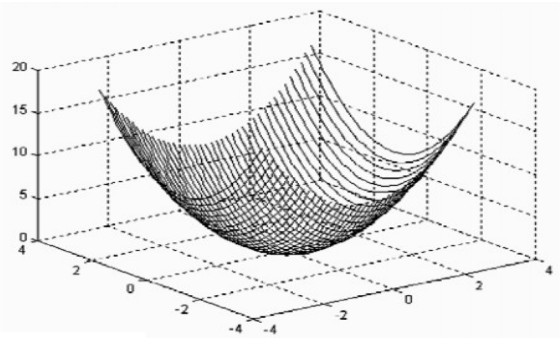
Для побудови тривимірних поверхонь у MatLab використовують декілька графічних команд. Аналогом команди plot для тривимірної графіки є команда plot3. За її допомогою можна побудувати аксонометрію тривимірної поверхні. Команда має такий формат:

plot3(X, Y, Z), де X, Y, Z — матриці однакового розміру.

При використанні цієї функції будуються точки з координатами X, Y, та відповідні значення функції Z, які поєднуються відрізками прямих ліній. Розглянемо приклад виконання команди plot3 разом ізкомандою meshgrid, у якому побудований графік функції $z(x, y) = x^2 + y^2$.

Приклад

```
»[X,Y]=meshgrid([?3:0.15:3]);
»z=X.^2+Y.^2;
»plot3(X,Y,Z);
»grid on;
```



Графік функції $z(x, y) = x^2 + y^2$ V

Варто звернути увагу на те, як при побудові тривимірних поверхонь виконується відома Вам команда grid on. Аналогічно команді plot, у команді plot3 можна вказу?

вати стиль ліній і точок відліку, наприклад:

Приклад

```
»[X,Y]=meshgrid([-3:0.15:3]);
»z=X.^2+Y.^2;
»plot3(X,Y,Z,’?k’,Y,X,Z,’?k’);
»grid on;
```

Недоліком такого подання є велика кількість ліній, які перетинаються.

Безмежні можливості для побудови та відображення складних тривимірних поверхонь відкриваються при використанні дескрипторної графіки. Виведемо список властивостей три вимір ного графіка . Для цього скористаємося системними функціями set та get. Головна частина програми, у разі ви користання дескрипторної графіки майже не змінюється. Але, як і в попередньому випадку, через функцію get виводяться властивості побудованого тривимірного об’єкта, які, після аналізу виведених даних, можна змінити командою set. Наведемо результат роботи функції get.

Приклад

```
»[X,Y]=meshgrid([?3:0.15:3]);
```

```
»z=X.^2+Y.^2;
»h?=plot3(X,Y,Z,’?k’,Y,X,Z,’?k’);
»grid on;
»d=get(hc)
d =
41x1 struct array with fields:
BusyAction
ButtonDownFcn
Children
Clipping
Color
CreateFcn
DeleteFcn
EraseMode
HandleVisibility
HitTest
Interruptible
LineStyle
LineWidth
Marker
MarkerEdgeColor
MarkerFaceColor
MarkerSize
Parent
Selected
SelectionHighlight
Tag
Type
UIContextMenu
UserData
Visible
XData
YData
ZData
»
```

Дії, що виконуються за допомогою дескрипторної графіки, можна виконати і через командний рядок. Так, визначення чорного кольору лінії, зроблене через опцію ‘k’команди plot3,можна зробити через команду set:

```
» set (hc, ‘Color’, [0, 0, 0]);
```

Як і у двовимірній графіці, три компоненти вектора [0, 0, 0] відповідають за яскравість червоного, зеленого та синього кольорів. Дескрипторна графіка надає більше можливостей для різноманітного подання результатів та дає змогу встановлювати властивості об’єктів не під час виведення на екран, а після цього. Крім того, аналіз та встановлення властивостей об’єктів при використанні функцій set та get проводиться значно простіше, ніж через графічні команди.

Оформлення тривимірних графіків

Оформлення тривимірного графіка у системі MatLab майже не відрізняється від відповідних дій вже розглянутих.Ці дії можна виконати трьома способами: через систему меню, через системні команди або через дескрипторну графіку. Раніше було показано, як для тривимірної графіки працює команда встановлення сітки grid. Аналогічно для поєднання тривимірних графіків можна використовувати команди hold on та hold off. Команди виведення підписів також подібні до відповідних команд двовимірної графіки.

xlabel(текстова константа або змінна) — установлення підпису для осі X, зміст підпису визначається текстовою константою або змінною;

ylabel(текстова константа або змінна) — установлення підпису для осі Y, зміст підпису визначається текстовою константою або змінною;

zlabel(текстова константа або змінна) — установлення підпису для осі Z, зміст підпису визначається текстовоюконстантою або змінною;

title (текстова константа або змінна)— установлення заголовка графіка;

text(x,y, текстова константа або змінна) — виведення тексту у рамку графіка, при цьому зміст тексту визначає текстова константа, а його положення — координати x та y;

gtext(текстова константа або змінна) — виведення тексту на площину графіка та визначення його положення за допомогою миші.

Особливо ефективне використання команди title(C) та gtext(C), де C — масив рядкових констант. Ці команди дають змогу розбивати підписи на графіках на декілька рядків.Поставити підписи на графік можна за допомогою наступного фрагмента програми:

Приклад

```
» xlabel(‘X’)
» ylabel(‘Y’)
» zlabel(‘Z’)
» title (‘Z(X,Y)=X^2+Y^2’)
```

Інші засоби редагування властивостей графіків, такі як видалення ліній, зміна освітленості та створення тіні, не можуть бути використані для аксонометричних поверхонь, які будують за

Y

X

Z

Z(X, Y) = X2+ Y2

допомогою функції plot3. Наприклад, на графіку неможливо вилучити невидимі лінії, які ускладнюють розуміння та аналіз поданої графічної інформації. Проте такі засоби доступні для інших, більш ефективних графічних команд системи MatLab. Ці функції роботи з тривимірною графікою будуть розглянуті у наступних параграфах.

Сітчасті тривимірні графіки та графіки із зафарбуванням

Сітчасті тривимірні графіки при поданні науково?технічної документації найбільш наочні. У системі MatLab існує кілька команд для їх побудови, але у їх назві завжди присутнє слово mesh (сітка). Два основних формати команди mesh:

mesh(X, Y, Z, C) — виводить у графічне вікно сітчаступоверхню Z(X, Y) із кольорами, що задаються масивом C;

mesh(X, Y, Z) — аналог попередньої команди. За замовчуванням вважається C = Z.

На тривимірних графіках, отриманих через використання команди mesh, застосовується функціональне фарбування залежно від висоти поверхні, у цьому разі за замовчуванням блакитні тони відповідають мінімальним значенням функції, а червоні —максимальним. Для встановлення палітри зафарбовування графіків можна використовувати команду MatLab colormap. Наприклад, для встановлення червоного кольору зафарбування поверхні може бути використана команда:

colormap ([1, 0, 0])

Для градієнтного зафарбовування графіків сірим кольором для виведення їх на чорно-білий друк використовують такий набір команд:

```
»colormap (gray)
»shading interp
```

На відміну від аксонометричних, сітчасті графіки можуть відображуватися як із невидимими лініями, так і без них. Оскільки сітчасті графіки мають багато властивостей, за рахунок яких можна поліпшити якість подання графічної інформації, зручно використовувати функцію mesh разом з засобами дескрипторної графіки.

Розглянемо приклад, у якому будується графік тієї самої функції z(x, y) = x2+ y2, але з використанням команди mesh та дескрипторної графіки. Програма для побудови сітчастого графіка буде виглядати так.

Приклад

```
»[X,Y]=meshgrid([?3:0.15:3]);
»Z=X.^2+Y.^2;
»ch=mesh (X,Y,Z)
»d=get(ch)
d =
AmbientStrength: 0.30000000000000
BackFaceLighting: ‘reverselit’
BusyAction: ‘queue’
ButtonDownFcn: ‘’
CData: [41x41 double]
CDataMapping: ‘scaled’
Children: [0x1 double]
Clipping: ‘on’
CreateFcn: ‘’
DeleteFcn: ‘’
DiffuseStrength: 0.60000000000000
EdgeColor: [0 0 0]
EdgeLighting: ‘flat’
```

```
EraseMode: ‘normal’
FaceColor: [1 1 1]
FaceLighting: ‘none’
HandleVisibility: ‘on’
HitTest: ‘on’
Interruptible: ‘on’
LineStyle: ‘?’
LineWidth: 0.50000000000000
Marker: ‘none’
MarkerEdgeColor: ‘auto’
MarkerFaceColor: ‘none’
MarkerSize: 6
MeshStyle: ‘both’
NormalMode: ‘auto’
Parent: 72.00036621093750
Selected: ‘off’
SelectionHighlight: ‘on’
SpecularColorReflectance: 1
SpecularExponent: 10
SpecularStrength: 0.90000000000000
Tag: ‘’
Type: ‘surface’
UIContextMenu: []
UserData: []
VertexNormals: [41x41x3 double]
Visible: ‘on’
XData: [41x41 double]
YData: [41x41 double]
ZData: [41x41 double]
»set (ch,’EdgeColor’,[0,0,0]);
»set (ch,’FaceColor’, ‘interp’);
» xlabel(‘X’)
» ylabel(‘Y’)
» zlabel(‘Z’)
» title (‘Z(X,Y)=X^2+Y^2’)
```

У наведеній програмі цікавими є два рядки, що пов’язані зі зміною властивостей виведеного на екран графічного об’єкта:

```
»set (ch,’EdgeColor’,[0,0,0]);
»set (ch,’FaceColor’, ‘interp’);
```

Перша команда змінює колір ліній, які формують сітку. За замовчуванням він змінюється залежно від значення функцій, що деякою мірою поліпшує перегляд графіка на екрані, але є дуже незручним під час оформлення документів. Переглянувши за допомогою команди d=get(ch)змінні, які відповідають графічному об’єкту, знаходимо необхідну змінну ‘EdgeColor’ (колір границі), яка відповідає за колір лінії сітки. Річ у тому, що використання змінної неможливе у даному

випадку. Але деякі змінні, наприклад,‘LineStyle’ та ‘LineWidth’, можна використовувати як для двовимірних, так і для сітчастих графіків. Така концепція цілком відповідає принципам наслідування та поліморфізму, які є базовими для об’єктно-орієнтованого програмування. Сітчастий графік, отриманий як результат виконання команди set (ch,’EdgeColor’,[0,0,0]).

Для покращення подання графіка можна використовувати градієнтне фарбування поверхні, яке здійснюється за допомогою команди дескрипторної графіки:

```
»set (ch,’FaceColor’, ‘interp’)
```

або розглянутого вище набору команд:

```
»colormap (gray)
»shading interp
```

Графік функції z(x, y) = x2+ y2, виконаний за допомогою команди mesh, графіки з проекціями ліній рівня на площину (x, y). Розглянемо програму, яка будує графік функції z(x, y) = x2+ y2з контурними лініями на площині (x, y).

Приклад

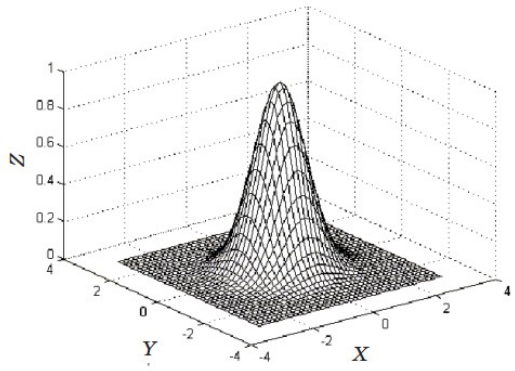
```
»[X,Y]=meshgrid([?3:0.15:3]);
»Z=X.^2+Y.^2;
»hc=meshc (X,Y,Z);
Y
X
Z
Z(x, y) = x2+ z2
»set (hc,’EdgeColor’,[0,0,0]);
»xlabel(‘X’);
»ylabel(‘Y’);
»zlabel(‘Z’);
»title (‘Z(X,Y)=X^2+Y^2’);
»colormap (gray);
»shading interp;
```

Для побудови графіків поверхонь з вертикальними лініями для визначення координати z використовують команду meshz. Нижче наведена програма для побудови тривимірного графіка із використанням команди meshz,

Приклад

```
»[X,Y]=meshgrid([?3:0.15:3]);
»Z=X.^2+Y.^2;
»meshz (X,Y,Z);
```

Розглянемо приклади побудови сітчастих графіків інших функцій, які мають екстремуми та точки розриву Подання графіків є наочним, і всі лінії розриву



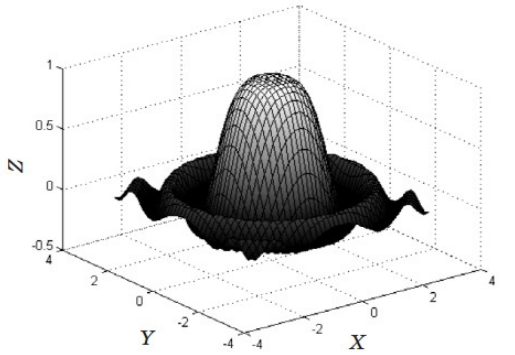
Сітчастий графік функції z(x, y) = = e $^{-(x2+\ y2)}$

У математиків та фізиків існує окремий інтерес щодо можливостей побудови тривимірних поверхонь обертання, які використовують під час геометричного та фізико-топологічного моделювання. У цьому разі при використанні команди mesh

```
Y, X, Z,
```

```
Z = exp(–(x2+ z2)
```

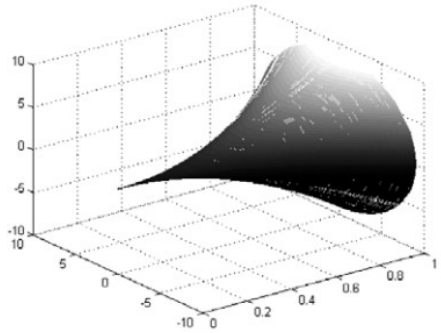
виникають певні проблеми, пов’язані з тим, що область значень таких функцій за координатою у обчислюється через двовимірну функцію f(x, y), тому їх неможливо визначити на сітці за координатами X та Y, як це було зроблено у попередніх прикладах. Геометричне пояснення неможливості побудови поверхонь обертання через команду meshgrid.



Z= sin(x2+ y2)/(x2+ y2) Сітчастий графік функції

Побудова поверхонь обертання може бути виконана через використання функції MatLab cylinder, яка призначена для математичного опису циліндричних поверхонь. Формат команди cylinder:

```
[X, Y, Z] = cylinder(R, n),
```



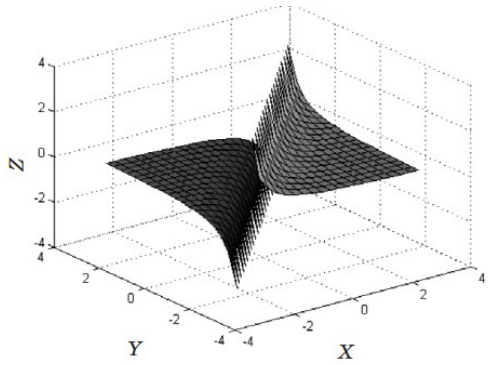
де R— радіус циліндра, а n— кількість відліків, яка визначає кут дискретизації поверхні циліндра. Оскільки величина R може бути змінною, функцію cylinder

можна застосувати для геометричного опису поверхонь обертання. Нижче наведено програму, що будує поверхню обертання функції y(x) = x2на відрізку [0; 3].

```
»[X,Y]=meshgrid(0:0.01:3);
»[X,Y,Z]=cylinder(X.^2,50);
»hc=mesh(Z,X,Y)
»set (hc,’EdgeColor’,[0,0,0]);
»colormap (gray)
»shading interp;
```

Для побудови поверхонь із фарбуванням також можна використовувати команду surf. Ця команда потребує формування прямокутної сітки на осях x та y, з використанням команди meshgrid. Формати команди surf:

surf(X, Y, Z, C) — команда будує кольорову поверхню заданими матриць X, Y, Z, що задається масивом C;



Z(x, y) = 1/(x – y)

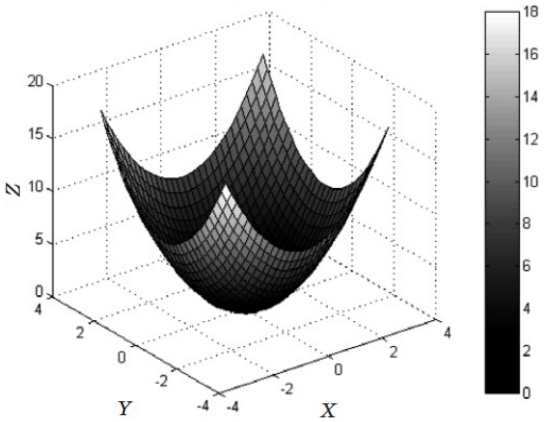
surf(X, Y, Z) — аналогічна команді surf(X, Y, Z, C) з параметром C = Z. Як і при використанні команди mesh, колір «За замовчуванням» задається висотою того або іншого фрагмента поверхні;

surf(x, y, Z) або surf(x, y, Z, C) — x, y — вектори, що визначають значення змінних x та y. У цьому разі, якщо вектор x має розмір n, а вектор y – m, матриця Z повинна мати розмір mxn. Вершини відображуваних областей визначаються відповідними координатами: (x(j), y(i), Z(i, j)). Вектор x відповідає стовпчикам матриці Z, а вектор y — її рядкам.

Розглянемо приклад програми для побудови параболоїда.

Приклад

```
»[x,y]=meshgrid([?3:0.2:3]);
»z=x.^2+y.^2;
»surf(x,y,z)
»colormap (gray)
»colorbar
»xlabel(‘x’)
»ylabel(‘y’)
»zlabel(‘z’)
»title (‘z(x,y)=x^2+y^2’)
```



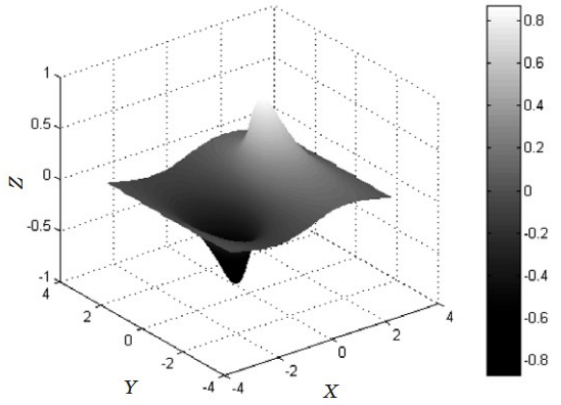
У результаті роботи програми будується поверхня (яку описує функція z(x, y) = x2+ y2) з накладеною на неї сіткою та функціональним фарбуванням, що значно сприяє візуальному сприйняттю графіка. Результат роботи функції surf майже не відрізняється від відповідного графіка, отриманого за допомогою функції mesh, який наведений на рис. 3.55. Різниця полягає у використанні функції colorbar, призначеної для виведення біля графіка градієнтної лінійки напівтонів сірого кольору. Таке подання графіка поліпшує його сприйняття та аналіз.

Подаються тривимірні графіки по-різному. Якщо під час оформлення технічної документації необхідно дати інформацію про координати кожної точки поверхні, найбільш зручними є сітчасті графіки з каркасним відображенням точок. Можна вилучити сітку, інтерполюючи кольорову гаму відтінками сірого кольору. Тому команди, призначені для побудови тривимірних поверхонь, мають засоби побудови як сітчастих, так і зафарбованих поверхонь. У цьому разі для виведення сітки та зафарбування використовують одні й ті самі процедури.

У наступному прикладі використовується функціональне зафарбування поверхні відтінками сірого. За допомогою команди surf з вилученням контурних ліній

Приклад

```
»[x,y]=meshgrid([?3:0.1:3]);
»[x,y]=meshgrid([?3:0.1:3]);
»z=sin(x)./(x.^2+y.^2+0.3);
»surf(x,y,z)
»colormap(gray)
»shading interp
»colorbar
»xlabel(‘x’)
»ylabel(‘y’)
»zlabel(‘z’)
»title (‘z=sin(x)/(x^2+y^2+0.3)’)
```



Команда colormap(gray) задає фарбування тонами сірого, команда shading interp забезпечує усунення зображення сітки та інтерполяцію кольору на зображенні графіка, а команда colorbar задає відображення лінійки інтерполяції кольору.

Команда surfc визначає побудову поверхні з відображенням проекцій контурних ліній, а команда surfl — будує графік поверхні з підсвічуванням від точкового джерела світла. Команда surfc працює подібно до команди meshc. Сітчастий графік функції з зафарбуванням, отриманий через використання команди surfl.

У разі використання команди surfl можуть застосовують такі функції керування підсвічуванням:

diffuse — дифузійне розсіювання світла;

lighting — керування яскравістю джерела світла;

material — імітація розсіювання світла від різних матеріалів;

specular — ефект дзеркального відображення.

Є такі системні функції для керування кутом огляду тривимірної поверхні:

view — визначення положення точки огляду;

viewmtx — визначення та обчислення матриці повороту графіка;

rotate3D — поворот тривимірної фігури.

Програмування в середовищі MatLab. Створення M-файлів. Створення Script-файлів

Мета: познайомитися з особливостями роботи системи MatLab в режимі програмування. Навчитися створювати M-файли, Script-файли та працювати з ними.

Теоретичні відомості

Система MATLAB дозволяє користувачеві надавати ім'я певним ланцюжкам команд таким чином, що це ім'я починає працювати як команда MATLAB, виконання якої в командному рядку викликає виконання того самого ланцюжка команд. Команди, створені користувачем, називаються ***програмами***, процес їхнього створення – ***програмуванням***. Якщо користувач створив певну команду (програму), то та в свою чергу може бути використана в якійсь більш складній

програмі. Програмування, таким чином, дозволяє складні, розвинені й довготривалі обчислювальні процеси складати з більш простих.

Система MATLAB розуміє програми двох типів – програми-функції (*functions*) і програми-сценарії (*scripts*), які ми для скорочення будемо називати *функціями* і *скриптами*. Для їхнього створення MATLAB має спеціальний текстовий редактор, який зберігає написані програми у файлі із вказаним користувачем іменем і з розширенням "***.****m*". (Можна використовувати і якийсь інший текстовий редактор, та це навряд чи доцільно).

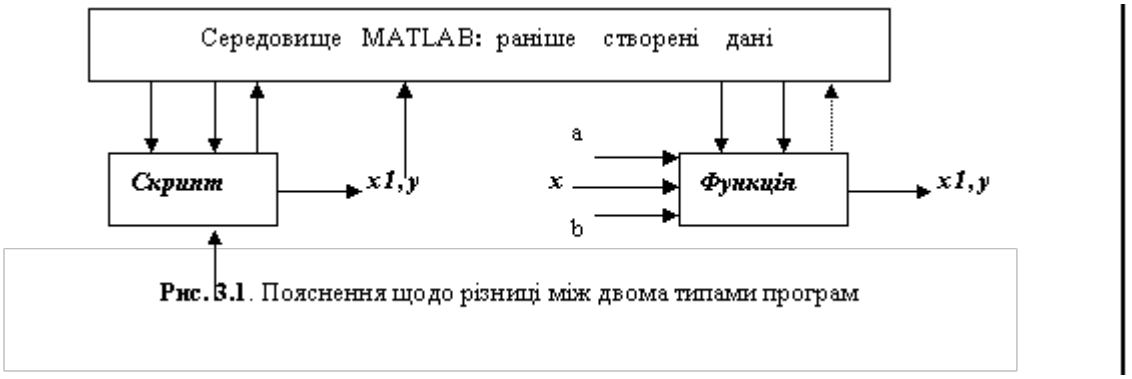


Рис. 3.1. Пояснення щодо різниці між двома типами програм

Поняття м-файлу. У систему MATLAB включена мова програмування високого рівня, за допомогою якої можна розв’язувати задачі векторної алгебри, лінійної алгебри і аналітичної геометрії за допомогою матричного запису, а також як автоматизувати, так і унаочнити процес дослідження впливу параметрів, що входять в рівняння, на положення і форму лінії і поверхні в координатному просторі.

Система MATLAB орієнтована на роботу в напівавтоматичному режимі, тобто необхідно в командному рядку ввести команду і передати її до ядра системи. Після перевірки синтаксису команда автоматично опрацьовується в ядрі і результат виводиться в командне і/або графічне вікно, після чого система переходить в режим очікування введення нової команди. Якщо необхідно переглянути результат виконання команди з іншими початковими даними, слід заново ввести в командний рядок всі необхідні команди, оскільки в системі MATLAB не передбачено можливості динамічного оновлення результату при зміні значення початкової змінної. Існує дві можливості автоматизації повторного введення серії команд. Перший спосіб полягає у використанні вікна Command History, в якому зберігаються введені раніше команди. Цей спосіб зручно використовувати для повторного виконання невеликої кількості команд. Проте в деяких випадках, залежно від поточних значень змінних, потрібно використовувати різні набори команд невідоме число разів. У таких випадках слід скористатися другим способом, заснованим на застосуванні m-файлів, де можуть міститися команди і управляючі структури мови MATLAB. Звертання до створеного m-файлу здійснюється за допомогою вказування (введення) його імені, а якщо необхідно заданням вхідних і вихідних параметрів в командному рядку.

М-файли являють собою текстові файли, що збережені з розширенням m. Оскільки m-файл є текстовим файлом, то його можна створювати за допомогою будь-якого текстового редактора. До складу системи MATLAB входить редактор Editor/Debugger, за допомогою якого можна як створювати m-файли, так і

виконувати їх налагодження, що приводить до скорочення часу створення робочого m-файлу. Відкрити редактор Editor/Debugger для створення нового m-файлу можна за допомогою команди M-file, розташованої в підменю New головного меню File, або за допомогою "натиснення" на кнопку New на панелі інструментів робочого столу системи MATLAB. М-файли підрозділяються на два типи: сценарії (script) і функції (function).

Файли-сценарії. Сценарії, що є найпростішим типом m-файлів, містять послідовності команд, які задаються в командному рядку, і коментарі, що починаються із знаку %.

Особливостями сценаріїв є:

1. виконання команд в режимі інтерпретації, тобто команди перетворюються у код програми і
2. виконуються порядково;
3. використовуючи тільки змінні, розташовані в робочому просторі MATLAB.

Головною особливістю описування програм мовою MATLAB є те, що типи змінних на початку програми не декларуються, досить надати змінній значення певного типу.

Приклад 1.Обчислити кути нахилу вектора a = (-1;2;5) до осей координат. Перевірити результат. Розв'язування оформити у вигляді сценарію з ім'ям example_l. File\\ New\\M-file

```
A=[-1 2 5]
A=acos(A./sqrt(sum(A.*A)))*180/pi
%Перевірка результату
A=sum(cos(A./180*pi).^2)
```

Після завершення введення файл-сценарію необхідно зберегти його в поточному робочому каталозі MATLAB. Для цього у вікні редактора m-файлів потрібно звернутися до послуги (команди) File\\Save as. В допоміжному вікні Save file as розкриється підкаталог work основного каталога MATLAB, який за замовчуванням визначений як поточний робочий каталог (Current Directory). У полі Ім'я файлу слід ввести ім'я файл-програми example_l замість імені Untitled. У полі Тип файлу автоматично відображуватиметься необхідний тип файлу M-files (* . m) який не слід змінювати. Виконати створену файл-програму можна одним з наступних способів:

1. у вікні редактора m-файлів обрати команду Debug Run, або натискувати клавішу F5;
2. ввести ім'я m-файла (без розширення), що містить файл-сценарій, в командний рядок і натиснути клавішу Enter.

Таким чином, при виклику файл-програми з командного рядка досить лише ввести її ім'я, і не потрібно задавати жодних вхідних даних і змінних всі змінні створюються при виконанні програми або створені раніше і знаходяться в робочому просторі.

Результати виконання сценарію:

```
>> example_1
```

```
A =
   -1     2     5
A =
  100.5197   68.5833   24.0948
A =
     1
```

Файл-функції.Функції разом з сценаріями також містять команди, але є складнішим типом m-файлів в порівнянні з сценаріями. Відмінними особливостями функцій від сценаріїв є:

1. можливість компіляції всієї функції у код програми з подальшим розміщенням його в пам’яті;
2. наявність власного робочого простору, де зберігаються локальні змінні;
3. наявність вхідних і вихідних параметрів.

Після виконання функції її робочий простір вилучається з пам’яті. Отже, значення всіх змінних, які були створені під час виконання функції і розташовувалися в її робочому просторі, вилучаються з пам’яті.

Приклад 2. Обчислити кути нахилу вектора a= (-1;2;5) до осей координат. Обчислення оформити у вигляді функції з ім'ям angles:

```
function a=angles(A)
%Обчислення  кута нахилу вектора
A=acos(A./sqrt(sum(A.*A)))*180/pi
```

Тут перший рядок є заголовком функції. Він включає ім’я функції (в даному випадку angles), а також один вхідний (A) і один вихідний (a) параметри. Наступний рядок з коментарем, який при обчисленні функції ігнорується. Це необов’язкова частина файл-функції. Проте якщо виникне потреба отримати довідку про функцію, коментар допоможе пригадати, яке призначення даної функції:

```
>> help angles
%Обчислення  кута нахилу вектора
```

Наступний рядок тіло функції вираз, за яким обчислюється значення функції. Після введення файл-функцію необхідно зберегти в поточному робочому каталозі, аналогічно до збереження файл-сценарію, але ім’я m-файлу, в якому зберігається файл-функція, обов’язково повинне співпадати з іменем функції.

Створену файл-функцію можна використовувати як в командному режимі (аналогічно до будь-якої вбудованої функції системи MATLAB), так і викликати з інших файл-програм або файл-функцій.

При виклику файл-функції потрібно вказати всі її вхідні і вихідні параметри.

Результати виконання функції angles () :

```
>> A=[-1 2 5];
>> angles(A)
A =
  100.5197   68.5833   24.0948
>> angles ([-1 2 5])
A =
```

100.5197 68.5833 24.0948

Файл-функції з кількома вхідними та вихідними аргументами

Приклад 3. Обчислити вираз y(x)=x3(asin2(x)+bcos2(x)). Обчислення оформити у вигляді функції з ім’ям example3.

```
function f =example3(x,a,b)
%Обчислення значення виразу
f=x.^3.*(a*sin(x).^2+b*cos(x).^2);
```

Результати виконання функції example3 () :

```
>> example3(5,5,7)
ans =
  645.1161
```

Якщо функція має кілька вихідних параметрів (тобто повертається кілька значень), їх вказують в квадратних дужках через кому після слова function.

Приклад 4. Обчислити середнє значення елементів вектора та стандартне відхилення його елементів від середнього. Обчислення оформити у вигляді функції з іменем example4:

```
function [mean,stdev]=example4(x)
n=length(x);%Визначення довжини вектора
mean=sum(x)/n;%Обчислення середнього значення
stdev=sqrt(sum((x-mean).^2/n));
%Обчислення стандартного відхилення
```

Результати виконання функції example4 () для вектора A:

```
>> A=[2 4 6 8 9 7 5 3 1];
>> [mean,stdev]=example4(A)
m =
     5
s =
    2.5820
```

Приклад 5. Створити в редакторі m-файлів файл-програму, результатом виконання якої є

побудова в одному графічному вікні графіків трьох функцій: y1=exp(x\2), y2=cos4x, y3=sin4x

що задані на проміжку [0, 4].

```
x=[0:0.05:4*pi];
y1=exp(x/2);
subplot(3,1,1);
plot(x,y1);
y2=cos(4*x);
subplot(3,1,2);
```

```
plot(x,y2);
y3=sin(4*x);
subplot(3,1,3);
plot(x,y3);
```

Перейти до режиму фрагментації та поділити файл-програму на три фрагменти, кожний з яких буде включати команди для побудови графіків окремих функцій (y1, y2, y3) можна за допомогою звернення до команд Cell

Типи даних Більшість обчислювальних процесів в MATLAB виконуються над дійсними або комплексними числами, поданими у форматі double. Тип даних double є числовим типом, встановленим за замовчуванням. Всі класи даних є матричними, підтримується робота з розрідженими матрицями, де перевагу складають нульові елементи. Для кожного класу даних встановлюються свої операції. В MATLAB існує 15 фундаментальних типів даних (або класів об’єктів). Схема, де наведені основні типи даних і взаємозв’язки між ними.

Арифметичні та логічні класи даних Арифметичні класи даних (NUMERIC) і логічний клас (logical) є найбільш поширеними класами даних в системі MATLAB в порівнянні з іншими класами. Арифметичні класи даних поділяються на цілі та дійсні.

Дані цілого класу можуть бути числами із знаком (класи int8, int16, int32, int64) або числами без знаку (uint8, uint16, uint32, uint64). Дані дійсного класу можуть бути числами одинарної (single) та подвійної (double) точності. Дані логічного класу можуть набувати одного з двох значень: true або false. Значення логічного класу утворюються як результати операцій порівняння.

Приклад 6. Створити в редакторі m-файлів файл-функцію, за допомогою якої виконується побудова заштрихованої області у графічному вікні. Визначити, чи знаходиться точка A1(0,5;0,5) та A2(0,8;0,8) всередині цієї області.

```
function Aria(X,Y)
x=2-4*rand(1,1000000);% вісь x
y=2-4*rand(1,1000000);% вісь y
% Визначення вектора
L=((x.^2+y.^2)>=1)&(abs(x)<=1)&(abs(y)<=1);
plot(x(L),y(L),'.r')% Побудова області
grid on % Включення координатної сітки
axis equal, axis([-1.1 1.1 -1.1 1.1])% Встановлення однакових
масштабів
line([-2 2],[0 0],'color','black')% виведення осі Ox
line([0 0],[-6 2],'color','black')% виведення осі Oy
xlabel('x'), ylabel('y')
z=(X.^2+Y.^2)>=1;
z=and(z,and(abs(X)<=1,abs(Y)<=1))
```

В прикладі спочатку виконується формування двох масивів, які відповідають осям координат та містять 1000000 випадкових значень в діапазоні від -2 до 2,

отриманих за допомогою функції rand(). Параметром кожного координатного вектора є вектор L , розмірність якого дорівнює розмірності масивів значень координат. Це приводить до того, що використовуючи функцію plot () виконується відображення точки.

Результати виконання функції Aria(X,Y):

```
>> Aria([0.5 0.8],[0.5 0.8])
z =
    0     1
```

Особливості роботи з символьним процесором системи MatLab

Мета: познайомитися з особливостями роботи символьного процесора системи MatLab.

Символьний клас даних

Разом з числовими і логічними даними в системі MATLAB можна працювати з даними, які є набором символів. Змінні, що набувають символьних значень, є змінними символьного класу (char). Для створення змінної символьного класу досить надати їй значення, що записано в апострофи.

Приклад 7. Створити символьну змінну student, в першому рядку якої розміщується прізвище студента, а в другому – його ім’я:

```
>> student=['Петренко ';'Олександр']
student =
Петренко
Олександр
>> first_name='Олександр'; last_name='Петренко ';
>> student =[last_name;first_name]
student =
Петренко
Олександр
>> name='Олександр';student =char('Петренко ',name)
student =
Петренко
Олександр
```

Приклад був розв’язаний трьома способами. В результаті змінна studentє двовимірним масивом символів, що складається з двох рядків і дев'яти стовпців. При заданні значення змінної student в першому рядку для вирівнювання числа символів в рядках був добавлений один пропуск.

В системі MATLAB передбачено використання при об’єднанні рядків в масиви символів імен інших змінних, які самі містять масиви символів. В системі MATLAB можна об’єднувати рядки, що містять різну кількість символів. Ця операція реалізується за допомогою функції char (), за якою автоматично вирівнюється довжини рядків додаванням, де необхідно, пропусків в кінці рядка.

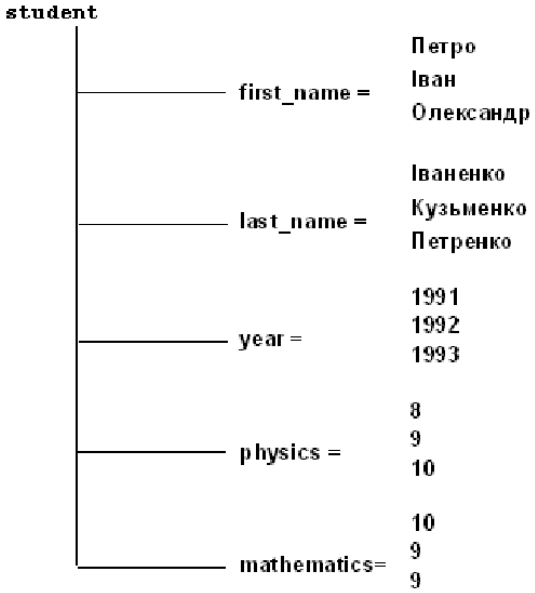
Організація структури

В мові MATLAB підтримується робот із змінними, за допомогою яких репрезентують дані різних класів. Подання різних даних за допомогою однієї змінної в системі MATLAB реалізується за допомогою класу даних структура (structure).

У системі MATLAB можна використовувати два підходи до організації структури: матричний і по елементний. Підходи розрізняються в розміщенні елементів структури. У першому випадку дані про імена, прізвища, дати народжень і оцінки всіх студентів розміщуються в одному відповідному полі в матричному вигляді. При цьому перші елементи в кожному з полів – дані про першого студента, другі елементи – про другого студента і так далі. Цей підхід переважно застосовується у випадку, коли необхідно виконувати операції над всіма елементами структури. У другому випадку дані про студентів знаходяться окремо у відповідних полях стосовно кожного студента. Даний підхід є переважним, коли необхідно опрацьовувати дані окремих елементів структури. Створити змінну класу структури можна двома способами: використовуючи роздільник [.]; за допомогою функції-конструктора stract().

Приклад 8. Створити змінну student, яка складається з трьох полів: first_name, де знаходиться ім'я студента, last_name з його прізвищем і year, що містить дату його народження.

```
>> student =struct('last_name','Петренко',...
'first_name', 'Олександр','year',1993)
student =
    last_name: 'Петренко'
   first_name: 'Олександр'
         year: 1993
```



Масив комірок

Масивом комірок називається впорядкований набір даних різнорідних класів. Масиви комірок можуть мати довільний розмір. На рис. 9 показано двовимірний

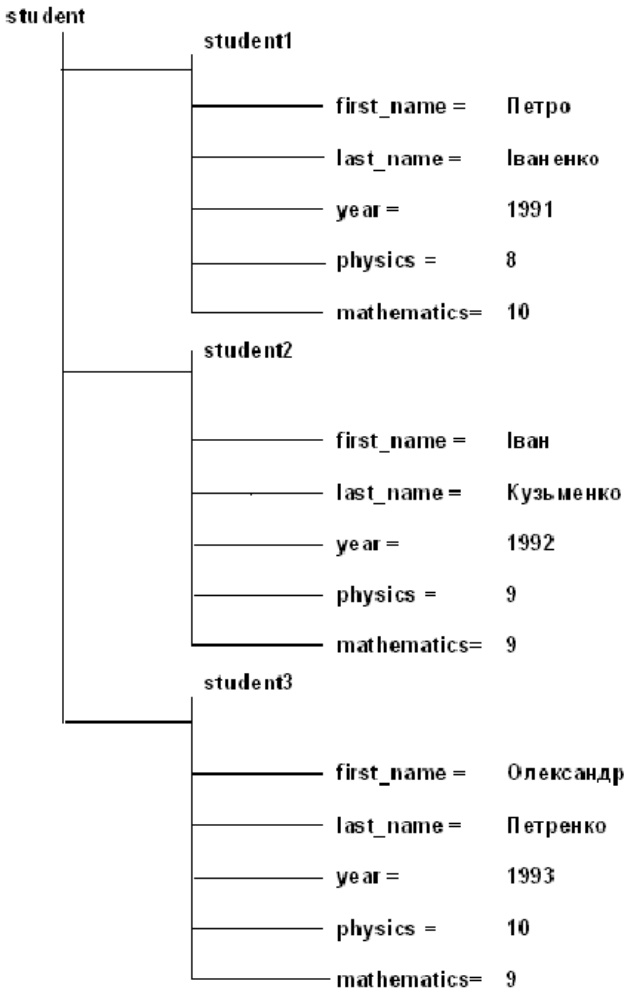
масив комірок 2x3, який містить масив символьних даних 3x1, масив арифметичних даних 2x3, структура, масив символьних даних 5x7, масив логічних даних 5x1 та масив комірок 2x3. Масив комірок є відмінною рисою системи MATLAB. Їх використання дозволяє реалізувати всі переваги роботи з матричними даними. Масиви комірок слід використовувати при утворенні масиву рядків різної довжини, оскільки немає необхідності стежити за рівністю їх довжин. Масиви комірок широко використовуються при створенні функцій, оскільки це дозволяє працювати з довільним числом вхідних і вихідних параметрів функції.

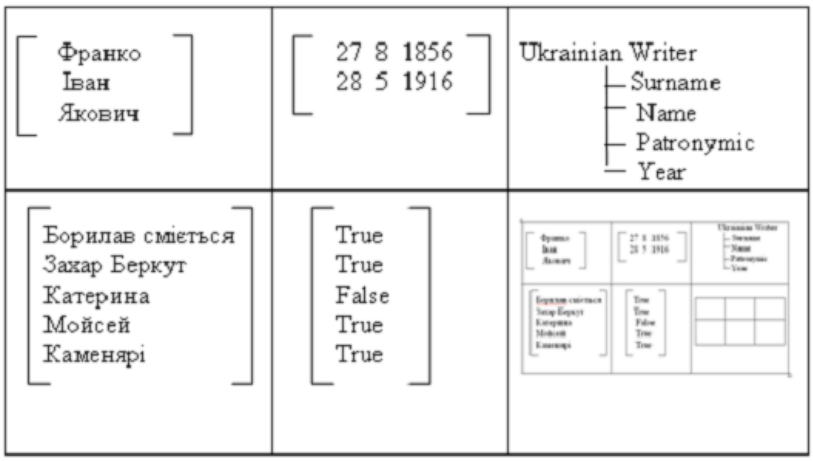
Масив комірок можна створити двома способами: послідовно надавати значення коміркам; використовуючи функцію-конструктор cell ().

Приклад 9. Використовуючи перший спосіб, створити масив комірок A
Шоста комірка містить порожній миссив.

```
>> A{1,1}=char('Франко','Іван','Якович');
>> A{1,2}=[27 8 1856;28 5 1916];
>> A{1,3}=struct('Ukranian_Writer',struct('Surname', ...
'Франко','Name','Іван','Patronymic','Франко',...
'Date',[27 8 1856;28 5 1916]));
>> A(2,1)={char('Борислав сміється', ...
'Захар Беркут','Катерина','Мойсей','Каменярі')};
>> A(2,2)={[true;true;false;true;true]};
>> A(2,3)={[]}
A =
   [3x6  char]    [2x3 double ]    [1x1 struct]
   [5x17 char]    [5x1 logical]           []
```

Фігурні дужки використовуються при встановлюванні відповідності комірок і даних. У трьох перших командах фігурні дужки знаходяться зліва від знака надання значення, а в трьох останніх справа. У першому випадку використовується індексація змісту (content indexing), а в другому випадку індексація комірок (cell indexing). Перше число в парі індексів позначає номер рядка, а друге номер стовпця. Значення логічного масиву пов’язані з назвами творів, які знаходяться в другій комірці. Оскільки Іван Франко не є автором твору «Катерина», то відповідний елемент логічного масиву має значення false.





Функція конструктор cell () використовується для попереднього виділення пам’яті під порожній масив комірок вказаного розміру.Доступ до даних масиву комірок здійснюється за допомогою індексації змісту, або за допомогою індексації комірок. Результатом виконання індексації змісту будуть дані, які містяться у відповідних комірках. Отже, клас результату відповідає класу даних, що містяться у відповідних комірках. При виконанні індексації комірок результатом буде масив комірок, розмірність яких відповідає розмірності набору індексів.

Звернутися до комірки можна і за допомогою одного індекса. Комірки нумеруються вздовж стовпців, які перебираються зліва направо.

Оператор циклу з рекурентними ітераційними обчисленнями

Приклад 1.1 Реалізація рекурентного ряду Фібоначчі через оператор циклу

```
» x(1)=1;
» x(2)=1;
» for i=3:10 x(i)=x(i-1)+x(i-2); end;
» x
  x =1 1 2 3 5 8 13 21 34 55
»
```

Приклад 1.2 Реалізація рекурентного ряду Фібоначчі, через множинну індексацію

```
» i=1:10;
» j=3:10;
» x(1)=1;
» x(2)=1;
» x(j)=x(j-1)+x(j-2);
??? Index exceeds matrix dimensions.
»
```

Реалізація рекурентного ряду Фібоначчі, через множинну індексацію. Слід перед оператором множинної індексації визначити всі 10 елементів вектора x, а потім перерахувати їх за рекурентною формулою. Третій елемент послідовності обчислений правильно, але при обчисленні четвертого та всіх наступних елементів використовуються попередньо задані, а не обчислені у циклі значення елементів вектора.

Приклад 1.3

```
» x=i;
» x(2)=1;
» x(j)=x(j-1)+x(j-2);
» x
x =1 1 2 4 7 9 11 13 15 17
»
```

Приклад 1.4

```
» x=i;
» x(2)=1;
» x=[1,1,x(j1)+x(j2)]
x =1 1 2 4 7 9 11 13 15 17
»
```

Функцію рекурсії визначають найпростішим способом - через рядок символів. Існують дві системні функції, які дають змогу обчислювати значення математичних функцій, попередньо визначених через рядок символів. Функція eval передбачає визначення імені функції та задання в дужках її параметрів при формуванні рядку

символів, тоді як для функції feval у рядку символів записується лише ім’я функції, а її параметри задаються окремо. Наприклад:

```
» eval(‘exp(1)’)
ans =
2.71828182845905
» feval(‘exp’,1)
ans =
2.71828182845905
»
```

Визначати функцію з параметрами виклику в дужках простіше для коротких математичних виразів з відомими параметрами, проте завдання тільки імені функції є більш гнучким способом оскільки таке подання дає змогу визначити параметри виклику залежно від результатів обчислень безпосередньо у командному рядку.

Вважатимемо також, що обчислення необхідно перервати при виконанні однієї з таких трьох умов:

1. Якщо досягнуто максимальну наперед задану кількість елементів вектора n.

2. Якщо досягнуто максимальну наперед задану різницю між значеннями сусідніх елементів вектора ε h. Цей параметр можна ефективно використовувати при формуванні розбіжних послідовностей.

3. Якщо досягнуто мінімальну наперед задану різницю міжзначеннями поточного та попереднього елементів вектора єl.

Цей параметр можна ефективно використовувати при формуванні збіжних послідовностей.

Відповідно до визначеної загальної концепції функція, призначена для виконання рекурентних обчислень, повинна мати такі вхідні параметри:

nr— кількість попередніх елементів послідовності, за якими проводяться обчислення. Наприклад, якщо xi= f(i, xi–1), то nr = 1, а якщо xi= f(i, xi–1, xi–2), то nr = 2;

n— максимальна кількість елементів послідовності, які необхідно обчислити;

v — вектор значень початкових елементів послідовності c1, c2, …, cn;

ff— рядкова змінна, яка визначає функцію рекурсії f(i,xi–1, xi–2, …, xi–n);

el— мінімальне значення різниці сусідніх елементів вектора, при якому необхідно припинити розрахунки;

eh— максимальне значення різниці сусідніх елементів вектора, при якому необхідно припинити розрахунки.

Сформулюємо обмеження, які слід накласти на вхідні параметри для коректної роботи функції.

1. Параметри nr, n, el, eh повинні бути додатними числами.

2. Кількість елементів nr, за якими проводяться обчислення, має бути меншою, ніж максимальна кількість елементів послідовності n, тобто nr < n.

3. Кількість елементів вектора початкових значень v повинна бути більшою або рівною кількості елементів nr, за якими проводяться обчислення, але меншою або рівною максимальній кількості елементів послідовності n, тобто nr ≤lenth(v) ≤n.

Приклад обчислення послідовності Фібоначчі таким способом: створимо функцію fibon,
в якій будемо послідовно обчислювати члени ряду Фібоначчі через виклик тієї ж самої функції. Програмний код функції матиме вигляд:

Приклад 2.

```
    function r=fibon (n)
    if ((n==1)|(n==2)) r=1; else r=fibon(n-1)+fibon(n-2);
end;
    return
```

Приклад 3

```
    » for ii=1:10 c(ii)=fibon(ii), end;
    c =1
    c =1 1
    c =1 1 2
    c =1 1 2 3
    c =1 1 2 3 5
    c =1 1 2 3 5 8
    c =1 1 2 3 5 8 13
    c =1 1 2 3 5 8 13 21
    c =1 1 2 3 5 8 13 21 34
    c =1 1 2 3 5 8 13 21 34 55
    »
```

Оператор циклу з післяумовою while

Перший спосіб	Другий спосіб
while — умова, яка містить змінну n вираз 1; вираз 2; …; вираз, який містить змінну n; … % блок виразів % end;	while — умова, яка містить змінну n вираз 1, вираз 2, …, вираз, який містить змінну n, … % блок виразів % end;

Приклад 4 Розрахунок функції exp(x) через степеневий ряд

```
    function s=expwhile(x,el)
    s=1; p=1; nn=1;
    while (abs(p)>el) p=p*x/nn; nn=nn+1; s=s+p; end;
    return;
    » expwhile (5,1e-6)
    ans =1.484131589827695e+002
    » expwhile (5,1e-8)
    ans =1.484131591017449e+002
    » expwhile (5,1e-10)
    ans =1.484131591025724e+002
```

```
» exp(5)
ans =1.484131591025766e+002
```

Оператор вибору switch ... case ... otherwise … end

Оператор множинного вибору, або оператор перемикач, використовується для пошуку значень змінної серед елементів дискретної множини. Обмеження - у структурі case можна вказувати лише дискретні значення змінної, визначеної декларацією switch.

Формат оператора:

```
switch змінна
case {перший набір значень змінної}, блок виразів;
case {другий набір значень змінної}, блок виразів;
case {третій набір значень змінної}, блок виразів;
……
otherwise блок виразів;
end;
```

Приклад 5 використання оператора перемикача у зовнішній функції sw.m.

```
function sw(x)
switch x
case {1}, disp (‘x=1’),
case {2}, disp (‘x=2’),
case {3}, disp (‘x=3’),
otherwise disp
(‘x~=1,2,3’),
end
```

Результат роботи наведеної функції
для різних числових параметрів її
виклику має такий вигляд:

```
» sw(1)
x=1
» sw(2)
x=2
» sw(3)
x=3
» sw(5)
x~=1,2,3
» sw(–5)
x~=1,2,3
» sw(1.5)
x~=1,2,3
»
```

Оператори закінчення керуючих обчислювальних структур
end, break, continue та return

При реалізації циклів у структурному програмуванні буває недостатньо наявності оператора закінчення структури. Іноді необхідно при виконанні тієї або іншої умови перейти або на кінець, або на початок циклу. Для цього використовується оператори break та continue.

Оператор break ставиться у тілі циклу для дострокового переривання його виконання з виходом на кінець циклу, а оператор continue - для переривання з виходом на початок циклу.

Оператор return використовується для виходу з тіла зовнішньої функції або вбудованої функції, але при цьому він може бути розташований усередині тіла функції.

Приклад 6 використання операторів в циклі з фіксованою кількістю повторень for

```
    function p=testbreak (x)
    p=1;
    for ii=1:100
    if (x<0) error (‘x can’nt be negative value’), break;
end;
    if (x>100) x=10; continue, end;
    p=p*x*ii;
    end
    x
    return;
```

Результат роботи наведеної функції буде такий:

```
    >> testbreak(5)
    x =5
    ans =8.452725758442828e-089
    >> testbreak(–5)
    ??? Error using ==> testbreak
    x can’nt be negative value
    >> testbreak(10)
    x =10
    ans =1.071510288125467e-058
    >> testbreak(100)
    x =100
    ans =1.071510288125466e+042
    >> testbreak(150)
    x =10
    ans =1.071510288125467e-059
    >>
```

Особливості реалізації класичних алгоритмів обробки даних через матричні макрооперації.

Пошук максимального або мінімального значення та заміна елементів

Пошук максимальних і мінімальних елементів пов’язаний із перебором всіх елементів вектора або матриці та з порівнянням їх значень, математичний запис алгоритму пошуку через матричні макрооперації повинен враховувати ці особливості. Необхідно реалізувати операцію порівняння, еквівалентом якої є арифметико логічні вирази. З іншого боку, пошук — це не операція над усією структурою, а поелементна операція, і тому реалізувати її можна тільки через рекурентне співвідношення. Для реалізації алгоритму пошуку максимального або мінімального значення через матричні макрооперації необхідно записати його у вигляді рекурентного арифметико-логічного співвідношення.

Розглянемо спосіб формування такого співвідношення на прикладі алгоритму пошуку максимального елемента вектора. Спочатку слід рекурентний алгоритм у звичайній формі - через функцію порівняння:

$$x_{\max}(1) = x(1);\ x_{\max}(i)\big|_{i>1} = \begin{cases} x(i),\ \text{для } x(i) \ge x_{\max}(i-1), \\ x_{\max}(i-1),\ \text{для } x(i) < x_{\max}(i-1), \end{cases}$$

де x(i) — значення початкового вектора,
xmax(i) — значення вихідного вектора, останній елемент якоговідповідає максимальному елементу вектора x.

$$x_{\max}(1) = x(1);\ x_{\max}(i) =$$
$$= (x(i) \ge x_{\max}(i-1)) \cdot x(i) + (x(i) < x_{\max}(i-1)) \cdot x_{\max}(i-1).$$

Приклад 1.

```
» x=[1, 100, –25, 20, 150, 40, –50];
» ffh='v(ii)*(v(ii)>=w(ii-1))+(v(ii)<w(ii-1))*w(ii-1)';
» c=recvect(1, length(x), x, ffh, 1e-10,1e10)
c =1 100 100 100 150 150 150
»
```

Приклад операції заміни від’ємних елементів вектора x, визначеного у попередньому прикладі, на нулі. Відповідний рекурентний алгоритм можна записати таким чином:

$$x_m(1) = x(1);\ x_m(i)\big|_{i>1} = \begin{cases} x(i),\ \text{якщо } x(i) \ge 0, \\ 0,\ \text{якщо } x(i) < 0 \end{cases}$$

або у вигляді арифметико-логічного виразу

$$x_m(1) = x(1);\ x(i) = (x(i) \ge 0)\ x(i) + (x(i) < 0)\ 0.$$

Приклад 2

```
» x=[1, 100, –25, 20, 150, 40, –50];
```

```
» ffh='v(ii)*(v(ii)>=0)+0*(v(ii)<0)';
» d=recvect(1,length(x),x,ffh,0,1e10)
d =
1 100 0 20 150 40 0
»
```

Список рекомендованої літератури

Базова

1. Николайчук Я. М. Проектування спеціалізованих комп'ютерних систем : навч. посіб. / Николайчук Я. М., Возна Н. Я., Пітух І. Р. - Т. : Терно-граф, 2010. - 392 с.
2. Йордан, Э. Объектно-ориентированный анализ и проектирование систем / Э. Йордан, С. Аргила. - М.: Издательство «ЛОРИ», 2007. - 264 с.
3. Синтес Антони. Освой самостоятельно объектно-ориентированное программирование за 21 день / А. Синтес ; пер. с англ. А. И. Захаров [и др.]. - М. ; СПб. ; К. : Издательский дом "Вильямс", 2002. - 671 с.
4. Герман-Галкин С.Г. Компьютерное моделирование полупроводниковых систем в MATLAB 6.0 -СПб.: КОРОНА принт, 2001. - 320 с.
5. Кравець П.О. Об'єктно-орієнтоване програмування: навч.посібн. / П.О. Кравець.— Львів: Вид-во Львівської політехніки, 2012.— 624 с.

Додаткова

1. І.В.Мельник – Система науково-технічних розрахунків Matlab та її використання для розв`язання задач з електроніки. Т 1. Основи роботи та функції системи. Навчальний посібник. Київ Університет «Україна » 2009- 507 с.
2. І.В.Мельник – Система науково-технічних розрахунків Matlab та її використання для розв`язання задач з електроніки. Т 2. Основи програмування та розв`язівання прикладних задач. Навчальний посібник. Київ Університет «Україна » 2009- 327 с.
3. Ю.Лазарев. Начала программирования в среде Matlab . Учебное пособие. Киев НТУУ «КПИ», 2003, -424с.

Інформаційні ресурси

http://matlab.exponenta.ru/index.php – центр компетенцій Math Work
http://www.mathworks.com/matlabcentral/ - Matlab Central
http://matlaber.ru/ * - Matlab Club
http://www.matrixlab-examples.com – Matrixlab