**ЛАБОРАТОРНі РОБОТи № 3,4**

**ВСТУП ДО СИСТЕМИ ІНЖЕНЕРНИХ І НАУКОВИХ РОЗРАХУНКІВ MATLAB**

Мета роботи: Ознайомлення з основними операціями у середовищі MatLAB на прикладі використання стандартних функцій, побудови файлів-сценаріїв та створення функцій користувача.

1.1. ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Система MatLAB (скорочено від MATrix LABoratory – матрична лабораторія) є мовою технічних обчислень для виконання наукових та інженерних розрахунків. До основних переваг системи можна віднести таке:

– математичний апарат MatLAB наближений до звичного математичного апарату інженера, а правила виконання обчислень орієнтовані на роботу з матрицями та комплексними числами;

– мова програмування системи MatLAB є простою та зручною;

– MatLAB є відкритою системою, що передбачає можливість її доповнення створюваними користувачем власними функціями.

Робота у середовищі MatLAB може відбуватися у двох режимах:

1) в режимі калькулятора, коли формули вводяться в командному вікні, туди ж виводяться результати обчислення, користувач може присвоювати результати змінним, керувати пам’яттю, викликати та використовувати вбудовані функції, будувати графіки;

2) шляхом виконання програми, написаної користувачем на мові MatLAB.

MatLAB надає не тільки мову програмування високого рівня, а також має функції, властиві операційним системам. Він має багато можливостей, які забезпечують діалог з користувачем у режимі командного рядка або з графічним інтерфейсом, перегляд робочої області та шляхів доступу до файлів MatLAB, редактор m-файлів, можливість експорту та імпорту даних іншим апаратним та програмним системам. ,

До системи MatLAB разом з вбудованими функціями входять пакети прикладних програм (toolbox), в яких зібрані функції, орієнтовані на вирішення спеціалізованих інженерних задач, наприклад:

– Signal Processing Toolbox (Оброблення сигналів) – набір програм для детермінованого, статистичного та спектрального аналізу, апроксимації передавальних функцій, побудови фільтрів зі скінченною та нескінечною

імпульсними характеристиками, моделювання сигналів, оброблення звуку, модуляції, генерування стандартних сигналів тощо;

– Image Processing Toolbox (Оброблення зображень) – програми для візуалізації зображень різних форматів, обробки та аналізу зображень, фільтрації, роботи з кольорами;

– Statistics Toolbox (Статистика) – набір програм для оцінювання параметрів законів розподілу, функцій щільності ймовірностей, генерації випадкових чисел, побудови лінійних та нелінійних моделей випадкових сигналів, кластерного аналізу, перевірки гіпотез тощо;

– Mapping Toolbox (Робота з картами) – програми для оброблення картографічних даних;

– Symbolic Math Toolbox (Символьна математика) – програми для символьних обчислень та спрощень символьних виразів;

– Fuzzy Logic Toolbox (Нечітка логіка) – програми для реалізації апарату нечіткої логіки;

– Neural Network Toolbox (Нейронні мережі) – програми для створення нейронних мереж, їх навчання за різними алгоритмами і оцінювання їх роботи;

– Wavelet Toolbox (Вейвлет) – програми для проведення вейвлет-розкладання сигналів у базисах вейвлетів, створення вейвлет-фільтрів, стиснення і знешумлення сигналів за допомогою вейвлетів;

– Control System Toolbox (Системи управління) – набір програм для моделювання лінійних інваріантних в часі систем, аналізу моделей у часовій та частотній областях;

– Optimization Toolbox (Оптимізація) – одномірна та багатомірна мінімізація функцій, пошук нулів та екстремумів, вирішення задач лінійного та квадратичного програмування;

– Matrix functions - numerical linear algebra (Матричні функції – чисельна лінійна алгебра)

– набір програм для дослідження матриць, вирішення алгебраїчних рівнянь, обернення матриць, факторизації, обчислення власних векторів та власних значень тощо;

– Data Acquisition Toolbox (реєстрація даних) – програми для отримання даних з зовнішніх пристроїв.

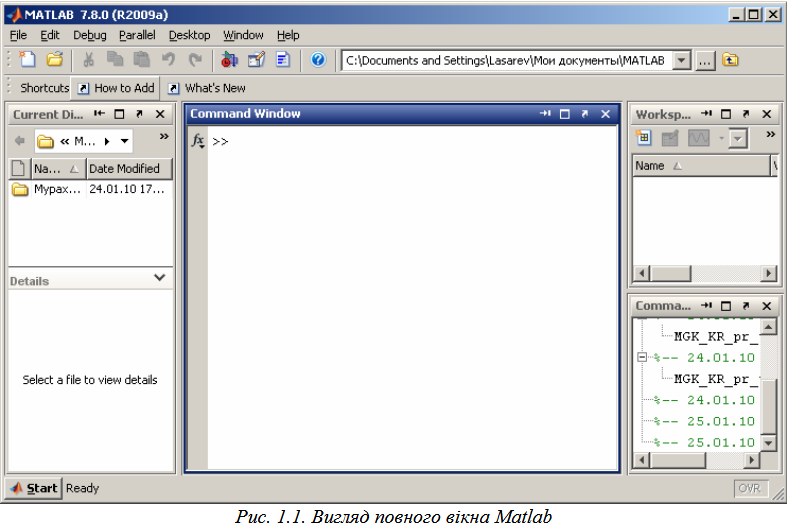
Робоча область MatLAB – це область пам‘яті, у якій розташовані змінні, з якими працює система MatLAB.

Система MatLAB використовує поняття поточного каталогу під час сеансу роботи, та список шляхів доступу до файлів.

1.2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

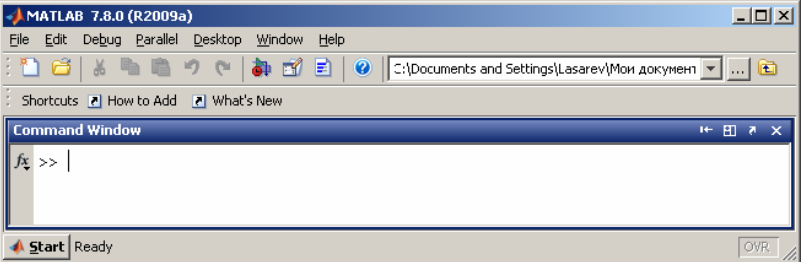
1. Ознайомитися з командним вікном MatLAB, поняттями робочої області та шляхів доступу.

Після виклику Matlab із середовища Windows на екрані виникає вікно, зображене на рис. 1.1.



Якщо закрити усі бокові допоміжні підвікна, залишиться одне вікно, яке

називають "командним" (Command Window) середовища Matlab (рис. 1.2).



Це вікно є головним у Matlab. У ньому виникають символи команд, що

набираються користувачем на клавіатурі дисплея, відображуються результати

виконання цих команд, текст програми, яка виконується, і інформація про по-

милки виконання програми, розпізнані системою .

Ознакою того, що Matlab готова до сприйняття і виконання чергової команди, є поява в останньому рядку текстового поля командного вікна знака запрошення (>>), праворуч якого миготить вертикальна риса.

У верхній частині вікна (під заголовком) розміщений рядок меню, в якому містяться меню File, Edit, Debug, Parallel, Desktop, Windows, Help. Щоб від-

чинити якесь меню, потрібно встановити на ньому курсор миші і клацнути її лівою кнопкою.

Тут відзначимо лише, що для виходу із середовища Matlab достатньо від-

чинити меню File і обрати у ньому команду Exit MATLAB, або просто зачинити

командне вікно, клацнувши лівою клавішою миші, коли курсор миші встановлений на зображенні верхньої крайньої правої кнопки цього вікна (з позначенням хрестика).

2. Ознайомитися з правилами введення змінних та констант, представлення даних у MatLAB.

**2.1. Введення дійсних чисел**

Введення чисел із клавіатури здійснюється по загальних правилах, прийнятих для мов програмування високого рівня:

для відділення дробової частини мантиси числа застосовується десяткова крапка (замість коми при звичайному записі);

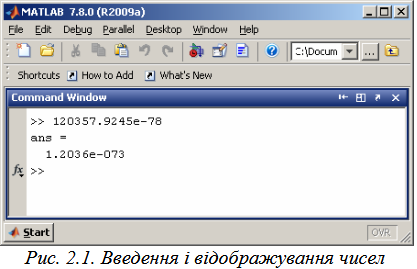
десятковий показник числа записується у вигляді цілого числа після попереднього запису символу "е";

між записом мантиси числа й символом "е" (який відокремлює мантису

від показника) не повинно бути ніяких символів, включаючи і символ пропуску.

Якщо, наприклад, ввести в командному вікні Matlab рядок

120357.9245e-78, то після натискання клавіші <Enter> у цьому вікні виникне запис (рис. 2.1):



Слід зазначити, що результат виводиться у виді (форматі), що визначається попередньо встановленим форматом подання чисел. Цей формат може бути встановлений за допомогою команди Preferences меню File (рис. 2.2).

Після її виклику на екрані з'явиться однойменне вікно (рис. 2.3). У поділі

Command Window одна з ділянок цього вікна має назву Numeric Format. Її призначено для встановлення і змінювання формату подання чисел, які виводяться в командне вікно в процесі розрахунків.

Передбачені такі формати:

Short (default) - стислий запис (застосовується за умовчанням);

Long - довгий запис;

Hex - запис у виді шістнадцяткового числа;

Bank - запис до сотих часток;

Plus - записується тільки знак числа;

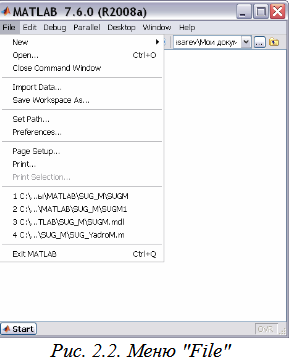
Short E - стислий запис у форматі із плаваючою комою;

Long E - довгий запис у форматі із плаваючою комою;

Short G - друга форма стислого запису у форматі з плаваючою комою;

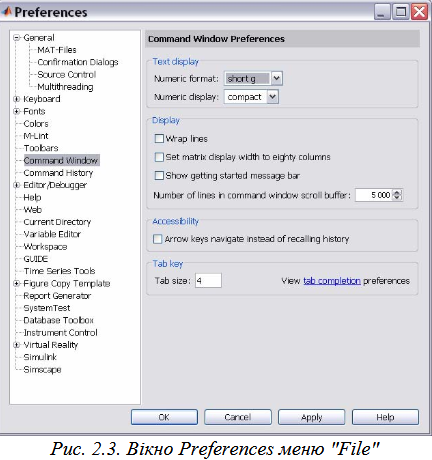
Long G - друга форма довгого запису у форматі з плаваючою комою;

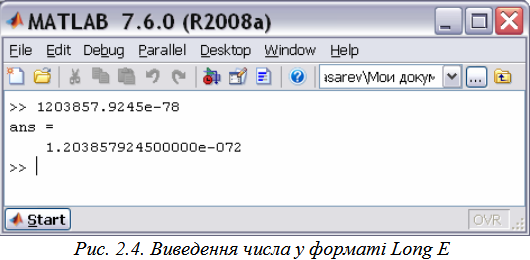
Rational - запис у вигляді раціонального дробу.



Обираючи за допомогою мишки потрібний вид подання чисел, можна забезпечити надалі виведення чисел у командне вікно саме в цій формі.

Як видно з рис. 2.1, число, яке виведено на екран, не збігається з введеним. Це обумовлено тим, що встановлений за замовчуванням формат подання чисел (Short G) не дозволяє вивести більше 5 значущих цифр числа. Насправді введене число усередині Matlab зберігається з усіма введеними його цифрами. Наприклад, якщо обрати мишкою селекторну кнопку Long E (тобто установити цей формат подання чисел), то, повторюючи ті ж дії, отримаємо результат, відображений на рис. 2.4, де вже всі цифри відображені вірно.





Слід пам’ятати:

- уведене число й результати всіх обчислень у системі Matlab зберігаються в пам'яті ПК із відносною похибкою біля (тобто з точними значеннями в 15 десяткових розрядах);

- діапазон подання модуля дійсних чисел лежить у проміжку між  і .

**2.2.** **Введення комплексних чисел**

Мова системи Matlab, на відміну від багатьох мов програмування високого рівня, містить у собі дуже просту в користуванні вбудовану арифметику

комплексних чисел. Більшість елементарних математичних функцій побудовано у такий спосіб, що аргументи припускаються комплексними числами, а результати також формуються як комплексні числа. Ця особливість мови робить

її дуже привабливою й корисною для інженерів і науковців.

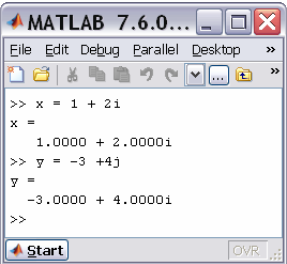
Для позначення уявної одиниці в мові Matlab зарезервовано два імені i

і j. Уведення із клавіатури значення комплексного числа здійснюється шляхом

запису в командне вікно рядка виду:

*<ім’я комплексної змінної> = <значення ДЧ> + i [j] \*<значення УЧ>,*

де ДЧ – дійсна частина комплексного числа, УЧ – уявна частина. Приклад наведено на рис. 2.5:



*Рис. 2.5*

*Введення комплексних чисел і виведення результату*

З нього видно, у якому виді система виводить комплексні числа на екран

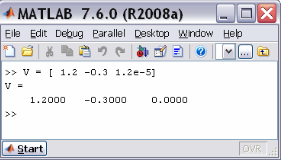
(і "до друку").

**2.3. Введення векторів і матриць**

Початкові значення векторів можна задавати із клавіатури шляхом поелементного введення. Для цього в рядку треба спочатку вказати ім’я вектора, потім поставити знак присвоювання '=', а далі, – відкривальну квадратну дужку, а за нею ввести задані значення елементів вектора, відділяючи їх пропусками або комами. Закінчується рядок записом квадратної дужки, що закриває.

Наприклад, запис рядка V = [ 1.2 -0.3 1.2e-5] задає вектор V, що містить

три елементи зі значеннями 1.2, -0. 3 і 1.2е-5 (рис. 2.6):



*Рис. 2.6. Введення вектора*

Після введення вектора система виводить його на екран. Те, що в наведеному прикладі останній елемент виведений як 0, обумовлено встановленим форматом short, відповідно до якого виводяться дані на екран.

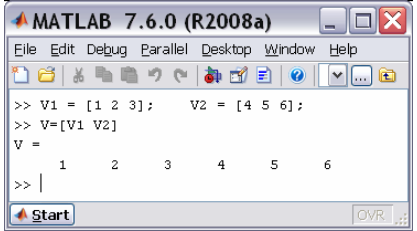
Довгий вектор можна вводити частинами, які потім об'єднувати за допомогою операції об’єднання векторів у рядок : v = [ v1 v2 ]. Приклад наведений на рис. 2.7.

Мова Matlab дає користувачеві можливість скороченого введення вектора, значення елементів якого є арифметичною прогресією. Якщо позначити: nz – початкове значення цієї прогресії (значення першого елемента вектора); kz – кінцеве значення прогресії (значення останнього елемента вектора); h – різницю прогресії (крок), то вектор можна ввести за допомогою короткого запису

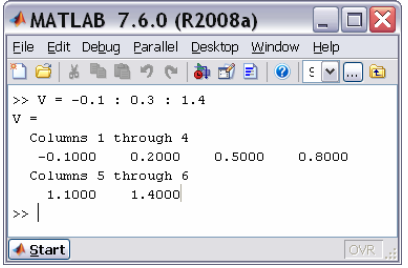
V = nz : h : kz .

Наприклад, введення рядка V = - 0.1 : 0.3 : 1.4 приведе до результату,

показаному на рис. 2.8.



*Рис. 2.7. Конкатенація (об’єднання) векторів*



*Рис. 2.8. Введення вектора арифметичної прогресії*

Якщо середній параметр (різниця прогресії) не зазначений, то він за замовчуванням приймається рівним одиниці. Наприклад, команда >> -2. 1 : 5

приводить до формування такого вектора

ans =

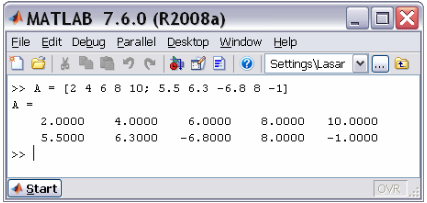
-2.1000 -1.1000 -0.1000 0.9000 1.9000 2.9000 3.9000 4. 9000

У такий спосіб уводяться вектори-рядки. Вектор-стовпець вводиться

аналогічно, але значення елементів відокремлюються знаком ";".

Введення значень елементів матриці здійснюється в Matlab у квадратних

дужках, по рядках. При цьому елементи рядка матриці один від одного відокремлюються пропуском або комою, а рядки один від одного відокремлюються знаком ";" (рис. 2.9).



*Рис. 2.9. Приклад введення матриці*

3. Ознайомитися з операціями над числами, матрицями.

**3.1. Найпростіші арифметичні дії**

В арифметичних виразах мови Matlab використовуються такі знаки арифметичних операцій:

+ – додавання;

– – віднімання;

\* – множення;

/ – ділення зліва праворуч;

\ – ділення справа ліворуч;

^ - –піднесення до степеня.

Використання Matlab у режимі калькулятора може відбуватися шляхом

простого запису в командний рядок послідовності арифметичних дій з числами,

тобто звичайного арифметичного виразу, наприклад:

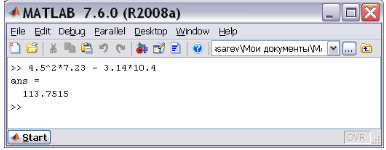
(4.5 )^2\*7.23 – 3.14\*10.4.

Якщо після введення із клавіатури цієї послідовності натиснути клавішу

<Enter>, у командному вікні виникне результат виконання у виді, поданому на

рис. 3.1, тобто на екран під ім’ям системної змінної **ans** виводиться результат дії

останнього виконаного оператора.



*Рис. 3.1. Введення арифметичного виразу і відображення результату*

Взагалі виведення проміжної інформації у командне вікно підпорядковується таким правилам:

- якщо запис оператора не закінчується символом ';', результат дії цього оператора одразу ж виводиться в командне вікно;

- якщо оператор закінчується символом ';', результат його дії не відображується в командному вікні;

- якщо оператор не містить знака присвоювання (=), тобто є просто записом деякої послідовності дій над числами і змінними, значення результату присвоюється спеціальній системній змінній за ім'ям **ans**;

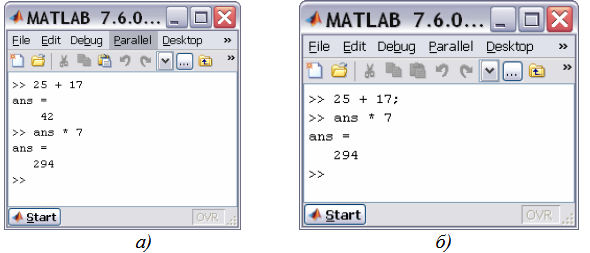
- отримане значення змінної **ans** можна використовувати в наступних операторах обчислень, використовуючи це ім’я **ans**; при цьому варто пам’ятати, що значення системної змінної **ans** змінюється після дії чергового оператора без знака присвоювання;

- у загальному випадку форма подання результату в командне вікно має вид:

<Ім'я змінної> = <результат>.

***Приклад.***

Нехай потрібно обчислити вираз (25+17)\*7. Це можна зробити таким чином. Спочатку набираємо послідовність 25+17 і натискаємо <Enter>. Одержуємо на екрані результат у вигляді **ans** = 42. Тепер записуємо послідовність **ans**\*7 і натискаємо <Enter>. Одержуємо **ans** = 294 (рис. 3.2 а). Щоб запобігти виведення проміжного результату дії 25+17, достатньо після запису цієї послідовності додати символ ' ; '. Тоді будемо мати результати у вигляді, поданому на рис.3.2 б.



*Рис. 3.2. Використання системної змінної* ***ans***

Особливістю Matlab як калькулятора є можливість використання імен змінних для запису проміжних результатів у пам’ять ПК. Для цього застосовується операція присвоювання, що вводиться знаком рівності ' = ' у відповідності зі схемою :

<Ім’я змінної> = <вираз>[;]

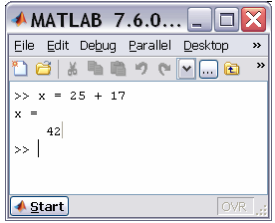
Ім’я змінної може містити до 30 символів і повинно не збігатися з іменами функцій, процедур системи і системних змінних. При цьому система розрізнює великі й малі букви в змінних. Так, імена 'amenu' , 'Аmenu', 'aMenu' у Matlab позначають різні змінні.

Вираз праворуч від знака присвоювання може бути просто числом, арифметичним виразом, рядком символів (тоді ці символи потрібно укласти в апострофи) або символьним виразом. Якщо вираз не закінчується символом ' ; ', після натискання клавіші <Enter> у командному вікні виникне результат виконання у вигляді :

<Ім'я змінної> = <результат>.

Наприклад, якщо ввести в командне вікно рядок 'x = 25 + 17', то на екрані

виникне запис (рис. 3.3) :



*Рис. 3.3. Присвоювання значення змінній*

Система Matlab має кілька імен змінних, що використовуються самою системою і входять до складу зарезервованих:

***i, j*** – уявна одиниця (корінь квадратний з -1);

***pi*** – число π (зберігається у виді 3.141592653589793);

***inf*** – позначення машинної нескінченності;

***NaN*** – позначення невизначеного результату (наприклад, типу 0/0 або inf/inf);

***eps*** – похибка операцій над числами із плаваючою комою

***ans*** – результат останньої операції без знака присвоювання;

***realmax*** – максимальна величина числа, що може бути використана;

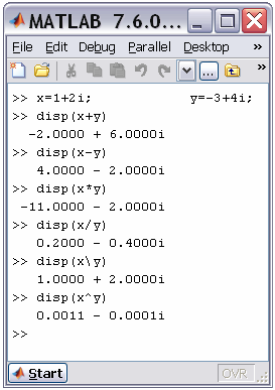
***realmin*** – мінімальна величина числа, що може бути використана.

Ці змінні можна використовувати в математичних виразах.

**3.2. Елементарні дії з комплексними числами**

Найпростіші дії з комплексними числами – додавання, віднімання, множення, ділення й піднесення до степеня – здійснюються за допомогою звичайних арифметичних знаків +, –, \*, /, \ і ^ відповідно.

Приклади використання наведені на рис. 3.4



*Рис. 3.3. Приклади використання функції* ***disp***

Примітка. У наведеному фрагменті використана функція **disp** (від слова 'дисплей'), яка також дозволяє виводити в командне вікно результати обчислень або деякий текст. При цьому чисельний результат, як очевидно, виводиться вже без указування імені змінної або **ans**.

**3.3. Дії над векторами**

Розрізнюватимемо дві групи дій над векторами:

а) векторні дії – тобто такі, що передбачені векторним зчисленням у математиці;

б) дії по перетворенню елементів – це дії, що перетворюють елементи вектора, але не є операціями, дозволеними математикою з векторами.

**3.3.1. Векторні дії над векторами**

Додавання векторів. Як відомо, складатися (підсумовуватися) можуть тільки вектори однакового типу (тобто такі, які обидва є або векторами-рядками, або векторами-стовпцями), що мають однакову довжину (тобто однакову кількість елементів). Якщо X і Y є саме такими векторами, то їхню суму Z можна одержати, увівши команду Z = X + Y, наприклад:

» x = [1 2 3] ; y = [ 4 5 6];

» v = x + y

v = 5 7 9

Аналогічно за допомогою арифметичного знака "– " здійснюється віднімання векторів, що мають однакову структуру ( Z = X – Y ).

Наприклад:

» v = x – y

v = -3 -3 -3

Транспонування вектора здійснюється застосуванням знака апострофу, що записується одразу за записом імені вектора, який транспонується.

Наприклад:

» x'

ans =

1

2

3

Множення вектора на число здійснюється в Matlab за допомогою знака арифметичного множення ( \* ) у такий спосіб: Z = X\*r або Z = r\*X, де r –деяке дійсне число.

Приклад:

» v = 2\*x

v = 2 4 6

Множення двох векторів визначено у математиці тільки для векторів однакового розміру (довжини) і лише тоді, коли один із векторів-множників є рядком, а другий – стовпчиком. Тобто, якщо вектори X і Y є рядками, то математичний зміст мають лише дві форми множення цих векторів: U = X' \* Y і

V = X \* Y'. Причому в першому випадку результатом буде квадратна матриця, а

в другому – число.

У МatLAB множення векторів здійснюється застосуванням звичайного знака множення (\*), який записується між множниками-векторами.

Приклад:

» x = [1 2 3] ; y = [ 4 5 6];

» v = x' \* y

v =

4 5 6

8 10 12

12 15 18

» v = x \* y'

v = 32

Для трикомпонентних векторів у Matlab існує функція **cross**, яка дозволяє знайти векторний добуток двох векторів. Для цього, якщо задані два трикомпонентних вектори v1 i v2, достатньо ввести оператор **cross**(v1, v2).

Приклад:

» v1 = [1 2 3]; v2 = [4 5 6];

» cross(v1,v2)

ans = -3 6 -3

На цьому перелік припустимих математичних операцій з векторами вичерпується.

**3.3.2. Поелементне перетворення векторів**

У мові Matlab є ряд операцій, які перетворюють заданий вектор в інший того ж розміру й типу, але в той же час не є математичними операціями з вектором як математичним об'єктом. Усі ці операції перетворюють елементи вектора як елементи звичайного одновимірного масиву чисел. До таких операцій належать, наприклад, усі елементарні математичні функції, наведені раніше і які залежать від одного аргументу. У мові Matlab запис, наприклад, виду Y = sin(X), де X – деякий відомий вектор, приводить до формування нового вектора Y, що має той самий тип і розмір, але елементи якого дорівнюють синусам відповідних елементів вектора-аргументу X.

Наприклад:

» x = [ -2,-1,0,1,2];

» y = sin(x)

y = -0. 9093 -0. 8415 0 0. 8415 0. 9093

» z = tan(x)

z = 2. 1850 -1. 5574 0 1. 5574 -2. 1850

» v = exp(x)

v = 0. 3679 1. 0000 2. 7183 7. 389

Крім цих операцій у МаtLAB передбачено декілька операцій поелементного перетворення, що здійснюються за допомогою знаків звичайних арифметичних дій. Ці операції застосовуються до векторів однакового типу й розміру. Результатом їх є вектор того ж типу й розміру.

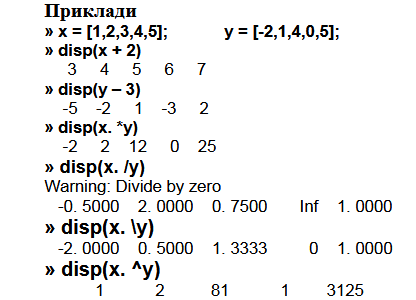
Додавання (віднімання) числа до (від) кожного елемента вектора. Здійснюється за допомогою знаку "+ " ("– ").

Поелементне множення векторів. Проводиться за допомогою сукупності знаків " .\* ", що записується між іменами векторів, які перемножуються. У результаті утворюється вектор, кожний елемент якого є добутком відповідних елементів векторів –"співмножників".

Поелементне ділення векторів. Здійснюється за допомогою сукупності знаків "./". Результат – вектор, кожний елемент якого є часткою від ділення відповідного елемента першого вектора на відповідний елемент другого вектора.

Поелементне ділення векторів в оберненому напрямку. Здійснюється за допомогою сукупності знаків ".\". В результаті отримують вектор, кожний елемент якого є часткою від ділення відповідного елемента другого вектора на відповідний елемент першого вектора.

Поелементне піднесення до степеня. Здійснюється за допомогою сукупності знаків ".^". Результат – вектор, кожний елемент якого є відповідним елементом першого вектора, піднесеним до степеня, розмір якого дорівнює значенню відповідного елемента другого вектора.



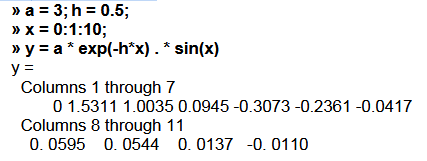
Вищевказані операції дозволяють дуже просто обчисляти (а потім – будувати графіки) складних математичних функцій, не використовуючи при цьому оператори циклу, тобто робити побудову графіків у режимі калькулятора.

Для цього достатньо задати значення аргументу як арифметичну прогресію так, як це було показано раніше, а потім записати потрібну функцію, використовуючи знаки поелементного перетворення векторів.

Наприклад, нехай потрібно обчислити значення функції:



при значеннях аргументу х від 0 до 10 із кроком 1. Обчислення масиву значень цієї функції у зазначених умовах можна здійснити за допомогою лише двох простих операторів :



**3.3.3. Операції з поліномами**

В системі Matlab передбачені деякі додаткові можливості математичного

оперування з поліномами.

Поліном (багаточлен) як функція визначається виразом :



В системі Matlab поліном задається й зберігається у виді вектора, елементами якого є коефіцієнти полінома від до :

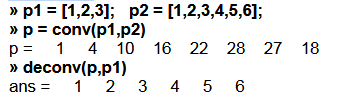


Введення полінома у Matlab здійснюється у такий самий спосіб, як і введення вектора довжиною n+1, де n – порядок полінома.

Множення поліномів. Добутком двох поліномів степенів n і m відповідно, як відомо, називають поліном степеня n+m, коефіцієнти якого визначають простим перемножуванням цих двох поліномів. Фактично операція множення двох поліномів зводиться до побудови розширеного вектора коефіцієнтів по заданих векторах коефіцієнтів поліномів-співмножників. Цю операцію в математиці називають згорткою векторів (а самий вектор, одержуваний у результаті такої процедури – вектором-згорткою двох векторів). У Matlab її здійснює функція conv(P1, P2).

Аналогічно, функція **deconv**(P1, P2) здійснює ділення полінома P1 на поліном P2, тобто обернену згортку векторів P1 і P2. Вона визначає коефіцієнти полінома, що є часткою від ділення P1 на P2.

Приклад :



В загальному випадку ділення двох поліномів призводить до одержання двох поліномів – полінома-результату (частки) і полінома-остачі. Щоб одержати обидва ці поліноми, треба оформити звернення до функції у такий спосіб:

[Q,R] = **deconv**(B,A) .

Тоді результат буде виданий у виді вектора Q з остачею у виді вектора R таким чином, що буде виконане співвідношення

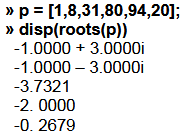
B = **conv**(A,Q) + R.

Система Matlab має функцію **roots**(P), що обчислює вектор, елементи якого є коренями заданого полінома Р.

Нехай потрібно знайти корені полінома:



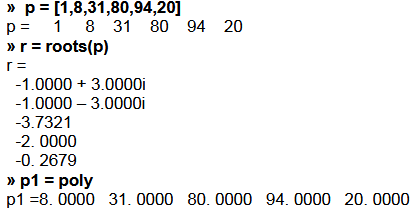
Нижче показано, як просто це зробити :



Обернена операція – побудова вектора р коефіцієнтів полінома за заданим вектором його коренів – здійснюється функцією **poly**:

p = **poly**

Тут r – заданий вектор значень коренів, p – обчислений вектор коефіцієнтів полінома. Наведемо приклад:



Зауважимо, що одержуваний вектор не показує старшого коефіцієнта, який за замовчуванням покладається рівним одиниці.

Ця ж функція у випадку, коли аргументом її є деяка квадратна матриця А розміром (n\*n), будує вектор характеристичного полінома цієї матриці. Звернення

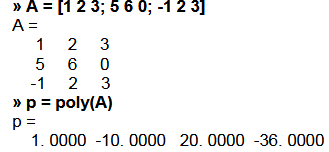
**p =** **poly**(A)

формує вектор p коефіцієнтів полінома



де Е – позначення одиничної матриці розміром (n\*n).

Розглянемо приклад:



Для обчислення значення полінома за заданим значенням його аргументу в Matlab передбачена функція polyval. Звернення до неї здійснюється за схемою:

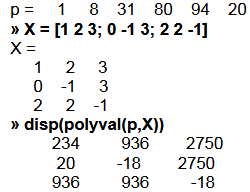
**y = polyval(p,x)**,

де p – заданий вектор коефіцієнтів полінома, а x – задане значення аргументу.

Приклад:



Якщо як аргумент поліному зазначена матриця Х, то функція **polyval(p,X)** обчислює матрицю Y, кожний елемент якої є значенням зазначеного полінома при значенні аргументу, рівному відповідному елементу матриці Х, наприклад:

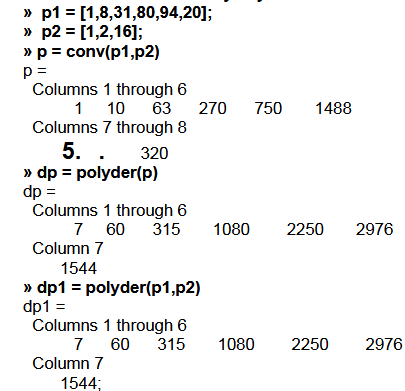


У цьому випадку функція обчислює значення полінома для кожного елемента матриці Х, і тому розміри вхідної й вихідної матриць однакові **size(Y) = size(X).**

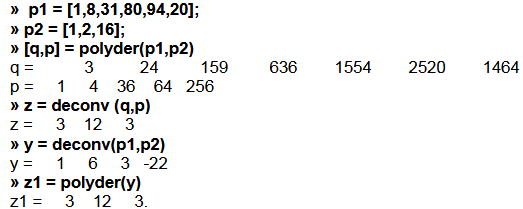
Обчислення похідної від полінома здійснюється функцією **polyder.** Ця функція створює вектор коефіцієнтів полінома, який є похідною від заданого полінома. Вона має три види звернень: **dp = polyder(p)** за заданим поліномом р обчислює вектор **dp**, елементи якого є коефіцієнтами полінома-похідної від заданого:



**dp = polyder(p1,p2)** обчислює вектор **dp**, елементи якого є коефіцієнтами полінома-похідної від добутку двох поліномів **р1** і **р2**:



**[q,p] = polyder(p1,p2)** обчислює похідну від відношення **(p1/p2)** двох поліномів **р1** і **р2** і видає результат у виді відношення **(q/p)** поліномів **q** і **p**:



4. Ознайомитися з правилами виклику функцій MatLAB.

**4.1. Елементарні математичні функції**

Загальна форма використання функції у Matlab така:

<ім’я результату> = <ім’я функції>(<перелік аргументів або їх значень>).

У мові Matlab передбачені наступні елементарні арифметичні функції.

*Тригонометричні й гіперболічні функції*

**sin**(Z) – синус числа Z;

**sinh**(Z) – гіперболічний синус;

**asin**(Z) – арксинус (у радіанах, у діапазоні від - π /2 до + π /2);

**asinh**(Z) – обернений гіперболічний синус;

**cos**(Z) – косинус;

**cosh**(Z) – гіперболічний косинус;

**acos**(Z) – арккосинус (у діапазоні від 0 до π );

**acosh**(Z) – обернений гіперболічний косинус;

**tan**(Z) – тангенс;

**tanh**(Z) – гіперболічний тангенс;

atan(Z) – арктангенс (у діапазоні від - π /2 до + π /2);

**atan2**(X,Y) – чотириквадрантний арктангенс (кут у діапазоні від – π до + π між горизонтальним правим променем і променем, що проходить через точку з координатами X і Y);

**atanh**(Z) – обернений гіперболічний тангенс;

**sec**(Z) – секанс;

**sech**(Z) – гіперболічний секанс;

**asec**(Z) – арксеканс;

**asech**(Z) – обернений гіперболічний секанс;

**csc**(Z) – косеканс;

**csch**(Z) – гіперболічний косеканс;

**acsc**(Z) – арккосеканс;

**acsch**(Z) – обернений гіперболічний косеканс;

**co**t(Z) – котангенс;

**coth**(Z) – гіперболічний котангенс;

**acot**(Z) – арккотангенс;

**acoth**(Z) – обернений гіперболічний котангенс.

*Експоненціальні функції*

**exp**(Z) – експонента числа Z;

**log**(Z) – натуральний логарифм;

**log10**(Z) – десятковий логарифм;

**sqrt**(Z) – квадратний корінь із числа Z;

**abs**(Z) – модуль числа Z.

*Цілочислові функції*

**fix**(Z) – округлення до найближчого цілого убік нуля;

**floor**(Z) – округлення до найближчого цілого убік від’ємної нескінченності;

**ceil**(Z) – округлення до найближчого цілого убік додатної нескінченності;

**round**(Z) – звичайне округлення числа Z до найближчого цілого;

**mod**(X,Y) – цілочислове ділення ділення X на Y;

**rem**(X,Y) – обчислення остачі від ділення X на Y;

**sign**(Z) – обчислення сигнум-функції числа Z (0 при Z=0, -1 при Z<0, 1 при Z>0).

**4.2. Спеціальні математичні функції**

Крім елементарних у мові Matlab передбачено цілу низку спеціальних математичних функцій. Нижче наведено перелік і стислий зміст цих функцій. Правила звернення до них і використання користувач може відшукати в описах цих функцій, що виводяться на екран, якщо набрати команду help і вказати в тому ж рядку ім’я функції.

*Функції перетворення координат*

**cart2sph** – перетворення декартових координат у сферичні;

**cart2pol** – перетворення декартових координат у полярні;

**pol2cart** – перетворення полярних координат у декартові;

**sph2cart** – перетворення сферичних координат у декартові.

*Функції Бесселя*

**besselj** – функція Бесселя першого роду;

**bessely** – функція Бесселя другого роду;

**besseli** – модифікована функція Бесселя першого роду;

**besselk** – модифікована функція Бесселя другого роду.

*Бета-функції*

**beta** – бета-функція;

**betainc** – неповна бета-функція;

**betaln** – логарифм бета-функції.

*Гамма-функції*

**gamma** – гамма-функція;

**gammainc** – неповна гамма-функція;

**gammaln** – логарифм гамма-функції.

*Еліптичні функції й інтеграли*

**ellipj** – еліптичні функції Якобі;

**ellipke** – повний еліптичний інтеграл;

**expint** – функція експоненціального інтегралу.

*Функції похибок*

**erf** – функція похибок;

**erfc** – додаткова функція похибок;

**erfcx** – масштабована додаткова функція похибок;

**erfinv** – обернена функція похибок.

*Інші функції*

**gcd** – найбільший загальний дільник;

**lcm** – найменше загальне кратне;

**legendre** – узагальнена функція Лежандра;

**log2** – логарифм за основою 2;

**pow2** – піднесення 2 до зазначеного степеня;

**rat** – подання числа у вигляді раціонального дробу;

**rats** – подання чисел у вигляді раціонального дробу.

**4.3. Функції комплексного аргументу**

Практично всі елементарні математичні функції, наведені раніше, обчислюються за комплексних значень аргументу й одержують у результаті цього

комплексні значення результату.

Завдяки цьому, наприклад, функція sqrt обчислює, на відміну від інших мов програмування, квадратний корінь із від’ємного аргументу, а функція **abs** при комплексному значенні аргументу обчислює модуль комплексного числа.

Приклади наведені на рис. 4.1.

У Matlab є кілька додаткових функцій, розрахованих тільки на комплексний аргумент:

**real**(Z) – виділяє дійсну частину комплексного аргументу Z;

**imag**(Z) – виділяє уявну частину комплексного аргументу;

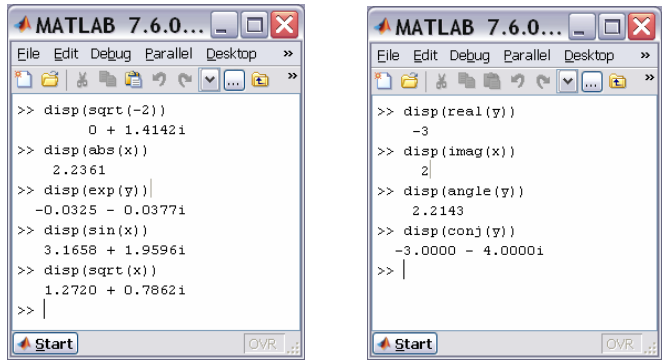
**angle**(Z) – обчислює значення аргументу комплексного числа Z (у радіанах від - π до + π );

**conj**(Z) – видає число, комплексно спряжене щодо Z.

Приклади наведені на рис. 4.2.

Крім того, у Matlab є спеціальна функція **cplxpair**(V), що здійснює сортування заданого вектора V із комплексними елементами у такий спосіб, що комплексно-спряжені пари цих елементів розташовуються у вихідному векторі в

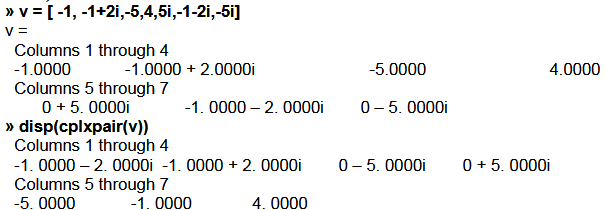
порядку зростання їхніх дійсних частин, при цьому елемент із від'ємною уявною частиною завжди розташовується першим. Дійсні елементи завершують комплексно-спряжені пари.



*Рис. 4.1. Застосування функцій Рис. 4.2. Застосування функцій*

*з комплексним аргументом комплексного аргументу*

Наприклад (надалі в прикладах команди, що набираються із клавіатури, будуть написані масним шрифтом, а результат їхнього виконання – звичайним шрифтом):



Пристосованість більшості функцій Matlab до оперування з комплексними числами дозволяє значно простіше будувати обчислення з дійсними числами, результат яких є комплексним, наприклад, знаходити комплексні корені квадратних рівнянь.

5. Ознайомитися з роботою команди HELP.

6. Ознайомитися з роботою функції plot для побудови графіків у MatLAB. Побудувати графік заданої функції, позначити вісі та заголовок графіку (функції xlabel, ylabel, title), нанести координатну сітку (функція grid).

**6.1. Виведення графіків. Процедура plot**

Виведення графіків у системі Matlab є настільки простою і зручною процедурою, що нею можна користуватися навіть при обчисленнях у режимі калькулятора.

Основною функцією, яка забезпечує побудову графіків на екрані дисплея,

є функція ***plot***. Загальна форма звернення до цієї процедури така: ***plot***(x1,y1,s1,x2,y2,s2,...).

Тут x1, y1 – задані вектори, елементами яких є масиви значень аргументу (х1) і функції (у1), що відповідають першої кривої графіка; x2, y2 – масиви значень аргументу й функції другої кривої і т.д. При цьому передбачається, що значення аргументу відкладаються уздовж горизонтальної осі графіка, а значення функції – уздовж вертикальної осі. Змінні s1, s2,... є символьними (їхня вказівка не є бов'язковою). Кожна з них може містити до трьох спеціальних символів, що визначають відповідно: а) тип лінії, що з'єднує окремі точки графіка; б) тип точки графіка; в) колір лінії. Якщо змінні s не зазначені, то тип лінії за умовчанням - відрізок прямої, тип точки – пиксел, а колір установлюється (у версії 5) за такою черговістю: - *синій, зелений, червоний, блакитний, фіолетовий, жовтий, чорний і білий* - у залежності від того, яка по черзі лінія виводиться на графік. Наприклад, звернення виду ***plot***(x1,y1,x2,y2,...) призведе до побудови графіка, у якому перша крива буде лінією з відрізків прямих синього кольору, друга крива – такого ж типу зеленою лінією і т.д.

*Графіки* в Matlab завжди *виводяться в окреме (графічне) вікно*, яку називають *фігурою*.

Наведемо приклад. Нехай потрібно вивести графік функції

y = 3sin(x + π /3) на проміжку від -3 π до +3 π із кроком π/100.

Спочатку треба сформувати масив значень аргументу х:

x = -3\*pi : pi/100 : 3\*pi, потім обчислити масив відповідних значень функції:

y = 3\*sin(x+pi/3) і, нарешті, побудувати графік залежності у(х).

У цілому в командному вікні ця послідовність операцій буде виглядати

так:

**» x = -3\*pi:pi/100:3\*pi;**

**» y = 3\*sin(x+pi/3);**

**» plot(x,y)**

В результаті на екрані з'явиться додаткове вікно із графіком (див. рис.

6.1а).

*Якщо вектор аргументу при зверненні до функції* ***plot*** *не зазначено явно,*

*то система обирає за умовчанням як аргумент номер елемента вектора функ-*

*ції.* Наприклад, якщо ввести команду

**» plot(y),**

то результатом буде поява графіка у виді, наведеному на рис. 3.5б.

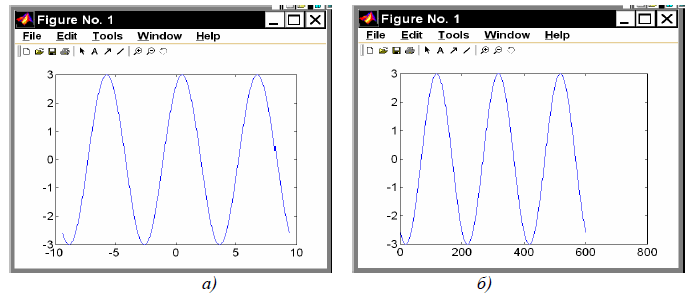


Рис. 6.1. *Виведення графіка синусоїди*

Графіки, наведені на рис. 3.5, мають декілька недоліків:

- на них не нанесено сітку з координатних ліній, що утруднює "зчитування" графіків;

- немає загальної інформації про криві графіка (заголовка);

- невідомо, які величини відкладені по осях графіка.

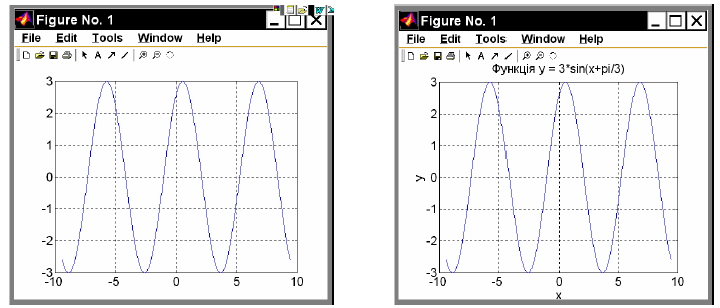
Перший недолік усувається за допомогою функції ***grid***. Якщо цю функцію записати одразу після звернення до функції ***plot***:

**» x = -3\*pi:pi/100:3\*pi;**

**» y = 3\*sin(x+pi/3);**

**» plot(x,y), grid,**

то графік буде споряджений координатною сіткою (рис. 6.2).



*Рис.6.2. Застосування* ***grid*** *Рис. 6.3. Застосування* ***title, xlabel,***

***Ylabel***

Цінною особливістю графіків, побудованих у системі Matlab, є те, що *сітка координат завжди відповідає "цілим" крокам змінювання*, що робить графіки "читабельними", тобто за графіком можна робити "відлік" значення функції при будь-якому заданому значенні аргументу і навпаки. Такої властивості не має жодний із графічних пакетів-додатків до мов програмування високого рівня.

Заголовок графіка виводиться за допомогою процедури ***title.*** Якщо після

звернення до процедури ***plot*** викликати ***title*** у такий спосіб:

***title***('<текст>'), то понад графіком з'явиться текст, записаний між апострофами у дужках. При цьому варто пам'ятати, що *текст завжди має міститися в апострофах.*

Аналогічно можна вивести пояснення до графіка, що розміщаються уздовж горизонтальної осі (функція ***xlabel***) і уздовж вертикальної осі (функція ***ylabel***).

Наприклад, сукупність операторів

**» x = -3\*pi : pi/100 : 3\*pi;**

**» y = 3\*sin(x+pi/3);**

**» plot(x,y), grid**

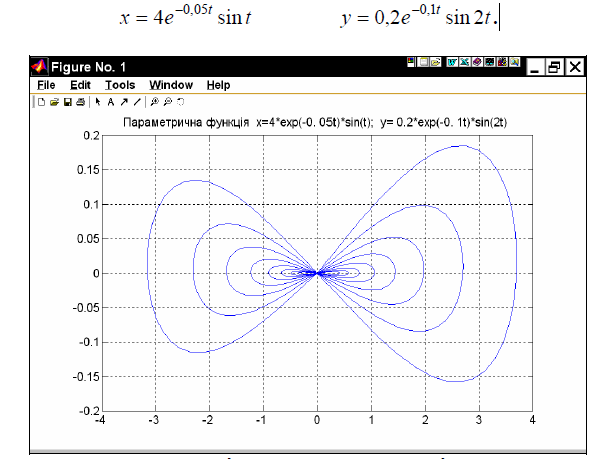
**» title('Функція y = 3\*sin(x+pi/3)');**

**» xlabel('x'); ylabel('y');**

призведе до оформлення поля фігури у виді, поданому на рис. 6.3. Очевидно, така форма вже цілком задовольняє вимоги, що висуваються до інженерних графіків.

Не більш складним є виведення у середовищі Matlab графіків функцій, заданих *параметрично*.

Нехай, наприклад, необхідно побудувати графік функції у(х), що задана параметричними формулами:



*Рис. 6.4. Графік параметрично заданої функції*

Виберемо діапазон змінювання параметра *t* від 0 до 50 із кроком 0.1. Тоді, набираючи сукупність операторів

**» t = 0:0.1:50;**

**» x = 4\*exp(-0.05\*t).\*sin(t);**

**» y = 0.2\*exp(-0. 1\*t). \*sin(2\*t);**

**» plot(x,y)**

**» title**('Параметрична функція x=4\*exp(-0. 05t)\*sin(t); y= 0.2\*exp(-0. 1t)\*sin(2t) ')

**» grid,** одержимо графік рис. 6.4.

**6.2. Спеціальні графіки**

Великою зручністю, наданою системою Matlab, є зазначена раніше можливість не вказувати аргумент функції при побудові її графіка. У цьому випадку як аргумент система приймає номер елемента вектора, графік якого будується.

Користуючись цим, наприклад, можна побудувати "графік вектора":

**» x = [ 1 3 2 9 6 8 4 6];**

**» plot (x)**

**» grid**

**» title('Графік вектора Х')**

**» ylabel('Значення елементів')**

**» xlabel(' Номер елемента').**

Результат поданий на рис. 6.5.

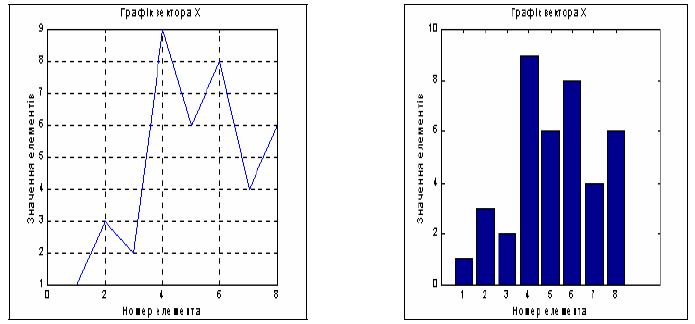
Ще більш наочним є подання вектора у виді *стовпцевої діаграми* за допомогою функції ***bar*** (див. рис. 6.6):

**» bar(x)**

**» title(**'Графік вектора Х'**)**

**» xlabel(**' Номер елемента'**)**

**» ylabel**('Значення елементів'**)**



*Рис. 6.5. Застосування* ***plot*** *Рис 6.6. Застосування* ***bar***

Якщо функція задана своїми значеннями при дискретних значеннях аргументу, і невідомо, як вона може змінюватися в проміжках між значеннями аргументу, зручніше подавати графік такої функції у виді окремих вертикальних

ліній для кожного із заданих значень аргументу. Це можна зробити, застосовуючи процедуру ***stem***, звернення до якої цілком аналогічно зверненню до процедури ***plot***:

**x = [ 1 3 2 9 6 8 4 6];**

**stem(x,'k')**

**grid**

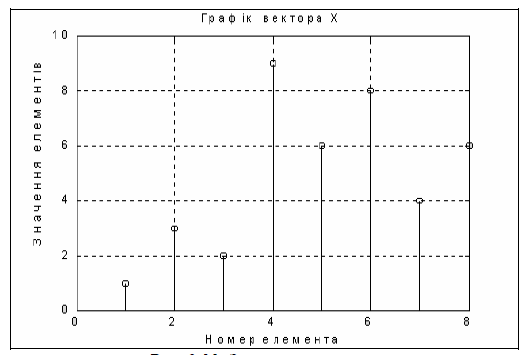
**set(gca,'FontName','Arial','FontSize',14),**

**title('Графік вектори Х')**

**ylabel('Значення елементів')**

**xlabel(' Номер елемента')**

На рис. 6.7 зображено одержаний при цьому графік.



*Рис. 6.7. Застосування* ***stem***

Інший приклад – побудова графіка функції у виді стовпцевої діаграми (рис. 6.8):

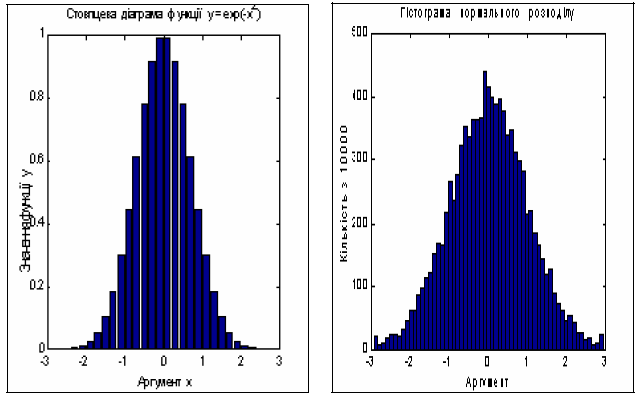
**» x = - 2.9 : 0.2 : 2.9;**

**» bar(x, exp(-x . \* x))**

**» title(**'Стовпцева діаграма функції y = exp(-x^2)'**)**

**» xlabel (**' Аргумент х'**)**

**» ylabel** (' Значення функції у'**)**



*Рис. 6.8. Процедура* ***bar*** *Рис. 6.9. Процедура* ***hist***

Ще одна корисна інженеру функція – ***hist*** (побудова графіка *гістограми*

заданого вектора). Стандартне звернення до неї має вид:

***hist***(y,x), де у – вектор, гістограму якого потрібно побудувати; х – вектор, що визначає інтервали змінювання першого вектора, усередині яких підраховується кількість елементів вектора “у”. Ця функція робить дві операції

- підраховує кількість елементів вектора “у”, значення яких потрапляють усередину відповідного діапазону, зазначеного вектором “х”;

- будує стовпцеву діаграму підрахованих чисел елементів вектора “у” як функцію діапазонів, зазначених вектором “х” .

Як приклад роздивимося побудову гістограми випадкових величин, що формуються вмонтованою функцією ***randn***. Візьмемо загальну кількість елементів вектора цих випадкових величин 10 000. Побудуємо гістограму для діапазону змінювання цих величин від -2,9 до +2,9. Інтервали змінювання нехай будуть рівні 0,1. Тоді графік гістограми можна побудувати за допомогою сукупності таких операторів:

**» x = -2.9:0.1:2.9;**

**» y = randn(10000,1);**

**» hist(y,x)**

**» ylabel(**'Кількість з 10000'**)**

**» xlabel(** 'Аргумент'**)**

**» title**('Гістограма нормального розподілу'**)**

Результат поданий на рис. 6.9. З нього, зокрема, випливає, що вмонтована функція ***randn*** достатньо вірно відображає нормальний гауссовий закон розподілу випадкової величини.

**6.3. Вставлення графіків до документу**

Щоб вставити графік із графічного вікна (фігури) до документу Word, слід скористатися командами меню, розташованого у верхній частині вікна фігури. У меню ***Edit*** оберіть команду ***Copy Figure***. Перейдіть у вікно документу і натисніть клавіші <Ctrl>+<C>. Вміст графічного вікна виникне у тому місці документу, де містився курсор.

7. Ознайомитися із роботою функцій генерації випадкових чисел (rand, randn, randi) із заданими густинами розподілу імовірності.

Функцsя randn генерує масив з випадковими елементами, розподіленими за нормальним законом з нульовим математичним очікуванням і середньоквадратичним відхиленням, що дорівнює 1:

* randn(n) — повертає матрицю розміром n х n. Якщо n — не скаляр, то з’явиться повідомлення про помилку;
* randn(m.n) або randn([m n]) — повертає матрицю розміром m x n;
* randn(m,n,p,...) або randn([m n р...]) — повертає масив з елементами, значення яких розподілені за нормальним законом;
* randn(size(A)) — повертає масив того ж розміру, що і А, з елементами, розподіленими за нормальним законом;
* randn (без аргументів) — повертає одне випадкове число, яке змінюється при кажному наступному виклику і має нормальний розподіл;
* randn( 'state') — повертає двоелементний вектор, що вміщує поточний стан нормального генератора. Для зміни стану генератора можна застосувати наступні форми цієї функції:
  + randn('state',s) — встановлює стан в *s;*
  + randn('state' ,0) — скидає генератор в початковий стан;
  + randn('state', j) — для цілих j встановлює генератор в *J-ий* стан;
  + randn('state', sum( 100\*clock)) — кожного разу скидає генератор в стан, який залежить від часу.

Функція rand генерує масиви випадкових чисел, значення елементів яких *рівномірно* розподілені на проміжку (0, 1):

* rand(n) — повертає матрицю розміром n х n. Якщо n — не скаляр, то з’явиться повідомлення про помилку;
* rand(m.n) або rand([m п]) — повертає матрицю розміром m x n;
* rand(m.n,p....) або rand([m n р...]) — повертає багатовимірний масив;
* rand(size(A)) — повертає масив того ж розміру, що і А, з елементами, розподіленими за рівномірним законом;
* rand (без аргументів) — повертає одне випадкове число, яке змінюється при кожному наступному виклику і має рівномірний закон розподілу;
* rand(' state') — повертає вектор з 35 елементами, що вміщує поточний стан генератора випадкових чисел з рівномерним розподілом. Для зміни стану генератора можна застосувати наступні форми цієї функції:
* rand('state' .s) — встановлює стан в *s;*
* rand( 'state' ,0) — скидає генератор в початковий стан;
* rand( 'state'. j) — для цілих j встановлює генератор в *J-ий* стан;
* rand( 'state' ,sum(100\*clock)) — кожного разу скидає генератор в стан, який залежить від часу.

8. Ознайомитися з роботою функції hold, побудувати два графіки в одному вікні.

Для того, щоб декілька послідовно обчислюваних графіків були зображені в одному графічному вікні в одному стилі і одному графічному полі, можна використати команду ***hold on***, тоді кожний такий графік буде будуватися в тому ж попередньо відкритому графічному вікні, тобто кожна нова лінія буде додаватися до раніше побудованих. Команда ***hold off*** виключає режим зберігання графічного вікна, встановленого попередньою командою.

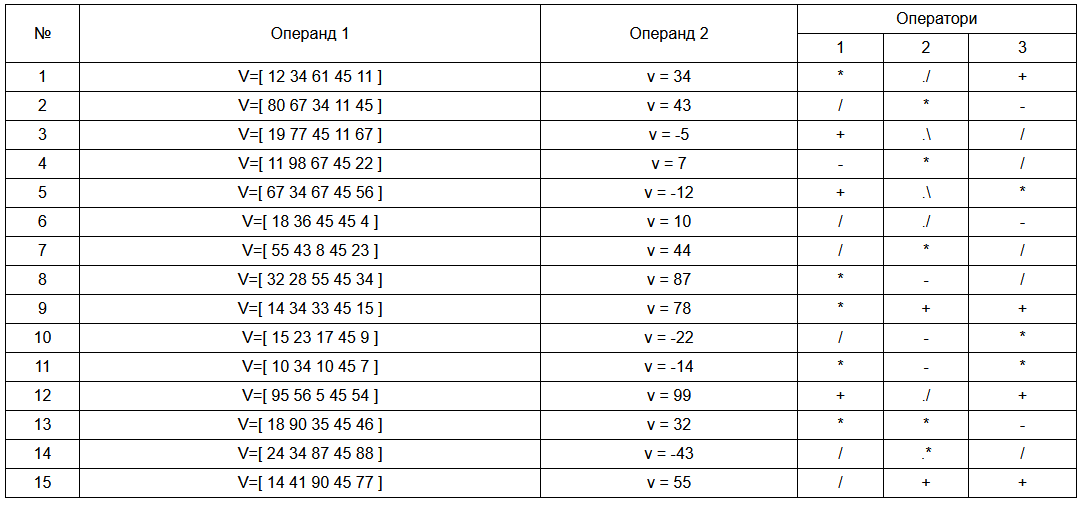
9. Ознайомитися з роботою функцій save та load, зберегти дані розрахунку в файл. Прочитати їх із файлу в іншому сценарії, побудувати графік.

**Варіанти завдань для лабораторної роботи 3:**

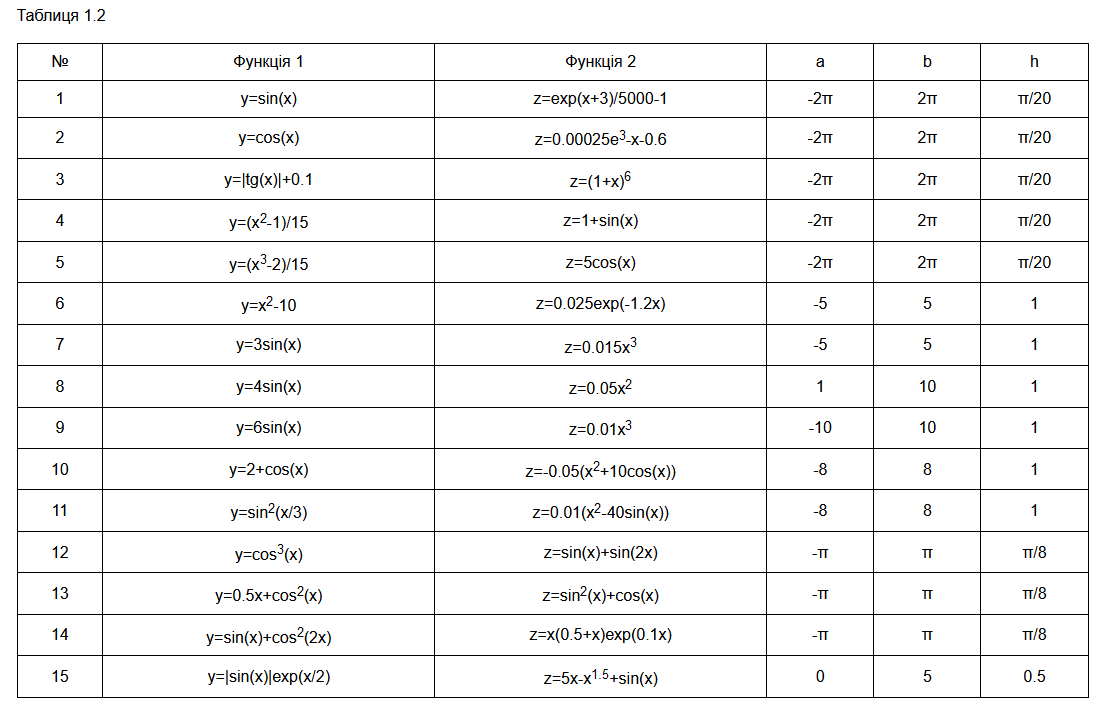
Таблиця 1.1

Виконати в режимі калькулятору наступні дії:

1. ввод вихідних операндів;
2. виконати над операндами 1 та 2 операцію 1;
3. виконати над результатом та операндом 1 операцію 2;
4. виконати над результатом та операндом 2 операцію 3;
5. піднести поелементно операнд 1 до степеня 3.



*Побудова графіків .* **(Варіанти завдань див. табл. 1.2)**



**Варіанти завдань для лабораторної роботи 4:**

*Поверхневий і контурний графіки.* **(Варіанти завдань див. табл.1. 3)**

Для формування поверхневого або контурного графіка необхідно:

- задати число точок по координатах X та Y,

- створити вкладені цикли по X та Y, обчислити функцію Z=f(X,Y),

- ввести номер графічного вікна, вивести туди графік обраного типу.

Необхідно використовувати графіки:

- тривимірний з аксонометрією, функція plot3(X,Y,Z),

- тривимірний з функціональним забарвленням, функція mesh(X,Y,Z),

- тривимірний з функціональним забарвленням і проекцією, функція meshc(X,Y,Z),

- тривимірний з функціональним забарвленням і проекцією, функція surf(X,Y,Z),

- контурний, функція contour(X,Y,Z),

- об'ємний контурний, функція contour3(X,Y,Z),

- тривимірний з висвітленням, функція surfl(X,Y,Z).

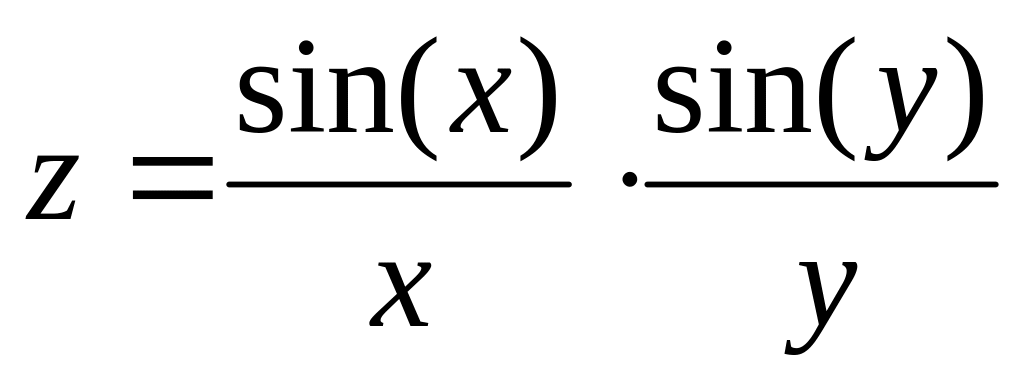
У кожному вікні можна малювати декілька графіків з накладенням один на одного. У списку параметрів для кожного графіка параметри перераховуються групами послідовно (у роботі графік для вікна один). У кожну групу входять:

- X - перша координата площадки основи,

- Y - друга координата площадки основи,

- Z - значення функції.

**Приклад виконання**

Функція 

Межі зміни аргументів -2π...2π

Варіант 1

*% Число точок і крок*

N=40;

h=pi/20;

*% Розрахунок матриці*

for n=1:2\*N+1

if n==N+1 A(n)=1; else A(n)=sin(h\*(n-N-1))/(h\*(n-N-1)); end;

end;

for n=1:2\*N+1

for m=1:2\*N+1 Z(n,m)=A(n)\*A(m);

end; end;

*% Завдання площадки*

[X,Y]=meshgrid([-N:1:N]);

*% Вивід графіка в аксонометрії у вікно 1*

figure(1); plot3(X,Y,Z);

*% вивід тривимірного графіка з функціональним забарвленням у вікно 2*

figure(2); mesh(X,Y,Z);

*% вивід тривимірного графіка з функціональним забарвленням і проекцією у вікно 3*

figure(3); meshc(X,Y,Z);

*% вивід тривимірного графіка із проекцією у вікно 4*

figure(4); surf(X,Y,Z);

*% Вивід контурного графіка у вікно*

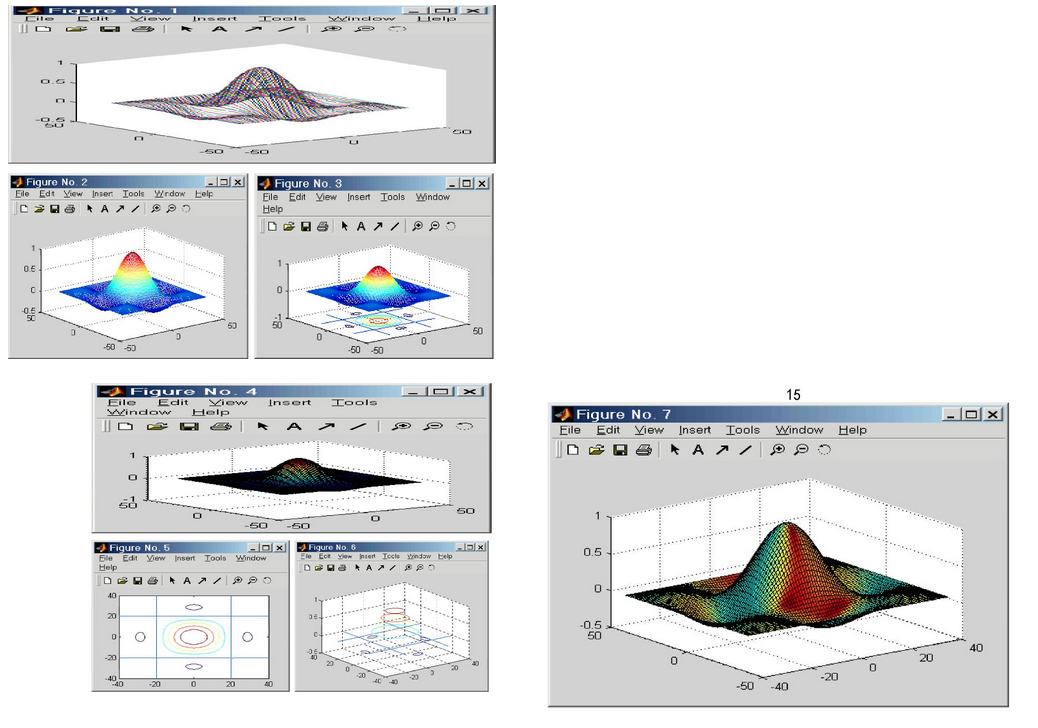
5 figure(5); contour(X,Y,Z)

*% Вивід об'ємного контурного графіка у вікно 6*

figure(6); contour3(X,Y,Z)

*% Вивід об'ємного графіка з висвітленням у вікно 7*

figure(7);surfl(X,Y,Z)



Варіант 2

*% Число точок і крок*

N=40;

h=pi/20;

*% Розрахунок матриці*

for n=1:2\*N+1

if n==N+1 A(n)=1; else A(n)=sin(h\*(n-N-1))/(h\*(n-N-1)); end;

end;

for n=1:2\*N+1

for m=1:2\*N+1 Z(n,m)=A(n)\*A(m);

end; end;

*% Завдання площадки*

[X,Y]=meshgrid([-N:1:N]);

*% Вивід графіка в аксонометрії в підвікно 1*

subplot(3,3,1),plot3(X,Y,Z);

*% вивід тривимірного графіка з функціональним забарвленням у підвікно 2*

subplot(3,3,2),mesh(X,Y,Z);

*% вивід тривимірного графіка з функціональним фарбуванням і проекцією в підвікно 3*

subplot(3,3,3),meshc(X,Y,Z);

*% вивід тривимірного графіка із проекцією в підвікно 4*

subplot(3,3,4),surf(X,Y,Z);

*% Вивід контурного графіка в підвікно 5*

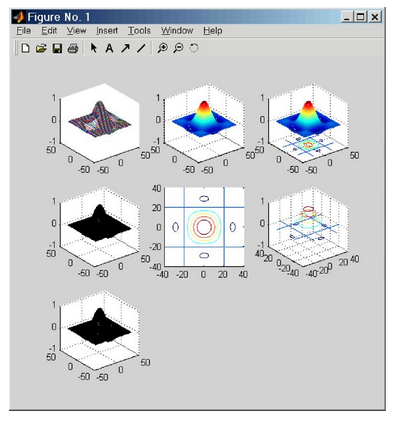
subplot(3,3,5),contour(X,Y,Z)

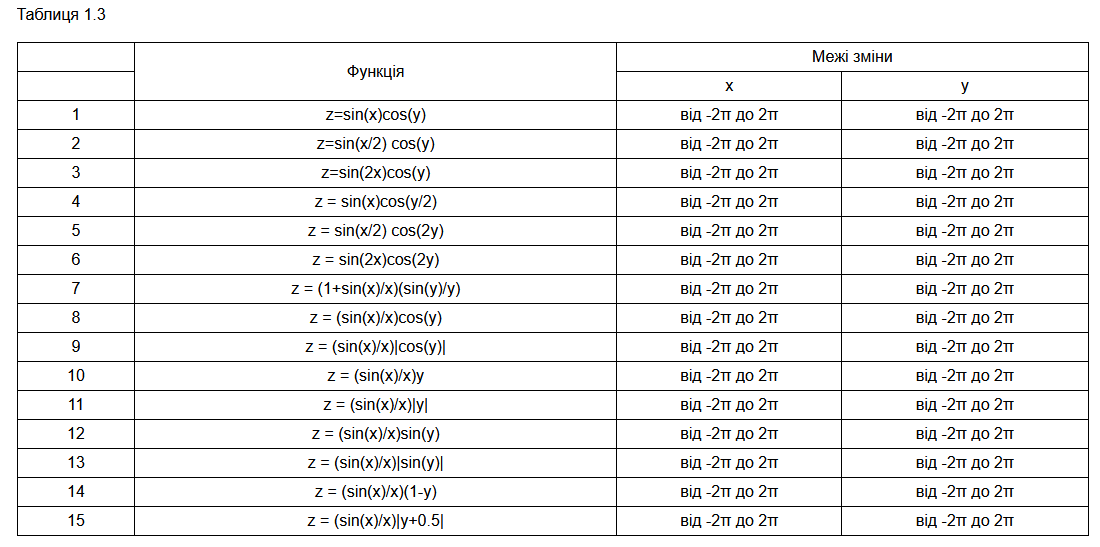
% *Вивід об'ємного контурного графіка в підвікно 6*

subplot(3,3,6),contour3(X,Y,Z)

*% Вивід об'ємного графіка з висвітленням у підвікно 7*

subplot(3,3,7),surfl(X,Y,Z)





КОНТРОЛЬНІ ЗАПИТАННЯ

1. В чому полягають переваги системи?

2. Назвіть два режими роботи в системі MatLAB.

3. Наведіть приклади пакетів програм, що входять до системи MatLAB та проаналізуйте їх вміст.

4. Опишіть можливості функції plot для візуалізації результатів моделювання.

5. Яка функція використовується для розбиття графічного вікна на декілька частин?