МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

Р.В.Сачок

Конспект лекцій

з кредитного модуля

" Основи програмування для проектування нафтопереробного обладнання "

для студентів

напряму 050503 Машинобудування

Затверджено на засіданні кафедри машин і апаратів хімічних та нафтопереробних виробництв

Протокол № 12 від 13.04.2016 р.
Завідувач кафедри МАХНВ

_____ Я.М. Корнієнко

КИЇВ – 2016

Конспект лекцій з кредитного модуля «Основи програмування для проектування нафтопереробного обладнання» для студентів напрямку підготовки 050503 Машинобудування: [Електронний ресурс]: / Укладач: Сачок Р.В., К.: НТУУ "КПІ", 2016 – 61 с.

Гриф надано Вченою радою
інженерно-хімічного факультету НТУУ „КПІ”
(Протокол № 4 від «30» травня 2016 р.)

Навчальне видання

Конспект лекцій з кредитного модуля «Основи програмування для проектування нафтопереробного обладнання» для студентів напрямку підготовки 050503 Машинобудування

Укладач: *Сачок Роман Володимирович*

Рецензент: *О.Л. Сокольський*(НТУУ "КПІ", кафедра ХПСМ ІХФ)

Відповідальний редактор: *Степанюк Андрій Романович*

ЗМІСТ

ВСТУП

На сьогоднішній день широкого поширення набули інтегровані середовища математичних обчислень, що дозволяють вирішувати складні математичні задачі для розв'язку, втому числі диференціальних рівнянь, що є одним з необхідних навичок розрахунків нафтопереробних процесів і процесів хімічних виробництв загалом. Однією з найбільш поширених систем цього типу є MATLAB.

Розробка пакету була почата в 1972 році за ініціативою відомих американських фахівців в області обчислювальної математики Дж. Форсайта і Дж. Уілкинсона. Спочатку MATLAB створювався як пакет програм, що реалізовують найбільш ефективні числові алгоритми лінійної алгебри.

Наповнення пакету проходило також і у напрямі розширення можливостей графічного представлення результатів обчислень, полегшення виведення результатів на друк і т.д. З появою ПЕОМ і ОС типа Windows, розробники MATLAB'у (фірма Mathworks) створила достатньо зручне середовище.

В сучасному вигляді MATLAB є інтегрованим середовищем програмування, що включає мову програмування високого рівня, засоби редагування, налагодження і виконання програм. Відзначимо також, що мова програмування MATLAB'у з одного боку має спрощений синтаксис, що полегшує його освоєння недосвідченим користувачем, а з іншого боку дозволяє досвідченому користувачеві створювати закінчені додатки, що використовують розроблені функції в інших середовищах програмування.

Різним аспектам роботи в системі MATLAB присвячено різноманітну літературу [1]–[11]. В більшості своїй представлена література носить довідковий характер, окрім [6]–[9], що спираються на відповідні учбові курси.

1 Команди і функції пакету MATLAB

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

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

1.1 Корисні команди пакету

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

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

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

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

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

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

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

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

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

—*hex* - шістнадцятирічний формат.

—+ - відображається тільки знак.

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

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

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

1.2 Змінні. Вектора і матриці

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

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

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

Основним об'єктом MATLAB'а є матриця. Вектором називається матриця, один з розмірів якої дорівнює одиниці, скаляром - матриця 1x1.

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

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

Для формування матриць і векторів використовуються символи «[…]». Наприклад матрицю 2x3 можна записати у вигляді *a* = [1 2.3 4; 3.4 2 3]. Різні елементи в такому записі розділяються пропуском (або комою), рядки відділяються один від одного символом ';'. Різне число чисел в рядках або стовпцях породжує помилку, про яку MATLAB повідомляє користувача.

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

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

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

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

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

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

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

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

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

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

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

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

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

c ='Hello!'

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

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

1.3 Прості арифметичні операції

Символьні позначення основних операцій, що проводяться над змінними:

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

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

Послідовність операцій визначається відповідно до звичайних правил алгебри, причому для виділення першочергових операцій традиційно

використовуються круглі дужки (...).

1.4 Робоча область і операції над нею

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

•*who* - виводить на екран список змінних, що зберігаються в робочій області

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

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

Grand total is 12 elements using 112 bytes

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

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

• *save* - зберігає всі змінні робочої області в бінарному файлі *matlab.mat.*

• *save « ім'я файлу»* - зберігає всі змінні робочої області в бінарному файлі з розширенням *.mat.* Цю ж дію можна зробити також в результаті виклику пункту підміню *Save Workspace as* в пункті *File* головного меню.

• *save «ім'я файлу» «змінні»* - зберігає змінні в бінарному mat-файлі із заданим ім'ям. Як змінні задається список їх імен розділених пропуском.

• *save « ім'я файлу» « змінні» – ascii* - зберігає змінні у файлі, в стандартних кодах ASCII. Ім'я файлу може мати довільне розширення.

• *load* - відновлює змінні з файлу *matlab.mat* .

• *load «ім'я файлу»* - відновлює змінні в робочу область з *mat-файлу* із заданим ім'ям. Цю ж дію можна зробити також в результаті виклику пункту підміню *Load Workspace* в пункті *File* головного меню.

• *load « ім'я файлу» -ascii* - читає в робочу область з файлу, матрицю записану

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

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

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

1.5 Робоча папка

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

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

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

Проглянути вміст цієї теки дозволяють команди:

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

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

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

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

1.6 Програмування в MATLAB'і. Сценарії і функції

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

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

1.6.1 Сценарії

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

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

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

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

1.6.2 Функції

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

function [outpars]=progname(inpars)

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

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

Виконання функції припиняється в наступних випадках:

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

```
function [e,f]=my1prog(a,b)
% function [e,f]=my1prog(a,b)
% It is my first function in MA TLAB
% Inputs: scalar constants a and b
% Outputs: scalar constants e and f
% Calculation formulas is in text of file.
c=3.5;
d=a/c*cos(b);
e=a* (c*sin(b)+cos(pi-b));
f=e/d;
```

Тепер звернення до функції з командного вікна може мати вигляд

[e1,f1] = my1prog(a,b);

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

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

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

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

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

1.7 Найбільш вживані стандартні функції пакету

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

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

1.7.1 Елементарні математичні функції

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

- *y = exp(x)* - поелементна експонента ($y_{ij} = e^{x_{ij}}$)
- *y = sin(x)* - синус
- *y = cos(x)* - косинус
- *y = tan(x)* - тангенс

• *y = asin(x)* - арксинус
• *y = acos(x)* - арккосинус
• *y = atan(x)* - арктангенс
• *y = sqrt(x)* - квадратний корінь
• *y = abs(x)* - абсолютна величина
• *y = log(x)* - натуральний логарифм
• *y = log10(x)* - десятковий логарифм
• *y = sign(x)* - сігнум-функція

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

1.7.2 Функції округлення і їм супутні.

• *y = round(x)* - округлення елементів *x* до найближчого цілого (округлення в традиційному сенсі)

• *y = ceil(x)* - округлення елементів *x* до найближчого цілого убік збільшення

• *y = fix(x)* - округлення елементів *x* до найближчого цілого убік нуля (служить для виділення цілої частині чисел)

• *y = floor(x)* - округлення елементів *x* до найближчого цілого убік зменшення

• *y = rem(x, y)* - залишок. *rem*(*x, y)= x — y * N*, де *N = fix(x/y).*
• *y = gcd(x,y)* - найбільший спільний дільник для елементів матриць *x* і *y*.

1.7.3 Функції над комплексними числами

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

• *y = angle(x)* - аргумент комплексного числа.
• *y = conj(x)* - комплексне сполучення.
• *y = imag(x)* - уявна частина.
• *y = real(x)* - дійсна частина.
• *y = sign(x)* - сигнум-функція, для комплексного *x, sign(x)= x./abs(x).*
Тут же згадаємо функції перетворення типів змінних:

• *num2str* - перетворить числову змінну в рядок символів;

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

1.7.4 Формування векторів і матриць

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

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

а *diag([5, 2],–1*) – матрицю:

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

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

Для повороту векторів і матриць використовуються наступні функції:

- *fliplr(A)* - переставляє стовпці матриці *A* симетрично щодо вертикальної осі

—*flipud(A)* - переставляє рядки матриці *A* симетрично щодо горизонтальної осі*;*

—*rot90(A, n)* - *n* разів повертає матрицю *A* на 90° проти годинникової стрілки. За відсутності параметра *n* проводиться поворот матриці на 90°. При нецілому значенні *n* функція не проводить ніяких дій, не видаючи ні попереджень ні повідомлень про помилку.

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

1.7.5 Операції над векторами і матрицями

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

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

• *y = sum(x)* - сума компонент вектора ;
• *y = prod(x)* - добуток компонент вектора ;
• *y = min(x)* - мінімальний компонент вектора;
• *y = max* (*x)* - максимальний компонент вектора;
• *y = mean* (*x)* - середнє арифметичне компонент вектора;
• *y = std(x)* - середнє квадратичне відхилення компонент вектора;

• *y = median(x)* - медіанна компоненту вектора. Під медіаною в цьому випадку розуміється величина, отримана в результаті наступних операцій: все *n* компонент вектора розташовуються в порядку зростання і в новому векторі вибирається компоненту з номером (*n* + 1)*/2* для непарних *n* і середнє арифметичне компонент з номерами *n/2* і *n/2* + 1 для парних *n;*

• *y = finite(x)* - повертає 1, якщо всі компоненти вектора *x* кінцеві і 0 інакше;

• *y = isNaN(x)* - повертає 1, якщо хоч би один з компонентів вектора *x* приймає значення *NAN* і 0 інакше;

• *y = all(x)* - повертає 1, якщо всі компоненти вектора *x* не рівні і 0 інакше;

• *y = any* (*x)* - повертає 1, якщо хоч би одна компоненту вектора *x* не рівні 0 і 0 інакше;;

• *y = find(x)* - повертає номери ненульових компонент вектора. У випадку,

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

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

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

1.7.6 Норми векторів і матриць

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

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

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

•*N* = 1 - перша норма вектора *v*

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

•*N = inf* - нескінченна (чебишевська) норма вектора *v*

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

•*N* = *-inf* - норма вектора *v* вигляду

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

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

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

y = norm(A, N), де *A* - матриця

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

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

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

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

•*N* = *inf* - нескінченна (рядкова) норма матриці *A*

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

•*N* = *'fro'* - фробеніусова норма матриці *A*

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

1.7.7 Елементарні операції над матрицями

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

- *[N, M] = size(A)* - розміри матриці *(A* - матриця розміру *N* x *M;*
- *L = length(A)* - максимальний розмір матриці *(L = max(size(A)));*
- *y = det(A)* - визначник матриці;
- *y = trace(A)* - слід матриці;
- *y = inv(A)* - обернення матриці;
- *y = pinv(A)* - псевдообернення матриці *(y =* $(A^T A)^{-1} A^T$*);*
- *y = sqrtm(A)* - корінь квадратний з матриці;
- *y = expm* (*A)* - експонента від матриці;
- *y = logm* (*A)* - логарифм матриці.
- *v = poly* (*A)* - обчислює вектор коефіцієнтів характеристичного

полінома (коефіцієнти розташовані у векторі в порядку убування ступенів).

• *[V, D] = eig(A)* - обчислення власних чисел і власних векторів матриці *A.* Діагональні елементи матриці *D* (жорданової канонічної форми) є власними числами матриці *A,* а стовпці матриці *V* є власними векторами. За наявності єдиного вихідного параметра *L = eig(A)* - вихід *L* - вектор-стовпець власних чисел матриці;

• *[U, S, V] = svd(A)* - обчислення сингулярного розкладання матриці *A.* Матриця *A* представляється у вигляді добутку трьох матриць $A = USV^T$. Тут матриці *U* і *V* - ортогональні матриці, *S* - діагональна матриця сингулярних чисел

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

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

•*y = rank (A)* - ранг матриці;

Тут слід уточнити деякі подробиці про те, що в MATLAB'і розуміється під рангом. MATLAB оперує з чисельними оцінками і обчислювальними операціями. Відповідно до цього при достатньо малих, хоч і ненульових значеннях детермінанта чисельне обернення матриці виявляється неможливим. Таким чином при обчисленнях вводиться узагальнене поняття рангу таким чином: рангом вважається кількість сингулярних чисел матриці, що перевищують заданий поріг *tol = max(size(A))* ||A||₂ * eps.*

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

• *c = cond(A)* - Обчислює число обумовленості матриці $\mu = \left\|A^{-1}\right\|_2 * \left\|A\right\|_2 = \sigma_{\max}(A)/\sigma_{\min}(A)$

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

Стандартна програма *eig* обчислення власних чисел автоматично проводить балансування матриць. Для відмови від цієї процедури необхідно додати в звернення другий аргумент - текстову константу

'nobalance' ([*V,D*] = *eig(A,' nobalance').*

- *Z = null (A)* - обчислення ортонормального базису нульового підпростору матриці. Базис нульового підпростору утворюють стовпці матриці *Z.* Обернення *Z = null(A', r')* приводить до обчислення "раціонального" базису, коли компонентами векторів є раціональні числа.
- *Q = orth(A)* - обчислення ортонормального базису матриці. Стовпці матриці *Q* утворюють той же підпростір, що і стовпці матриці *A,* причому $Q^TQ = I$.

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

```
a=hilb(6)
a =
   1.0000   0.5000   0.3333   0.2500   0.2000   0.1667
   0.5000   0.3333   0.2500   0.2000   0.1667   0.1429
   0.3333   0.2500   0.2000   0.1667   0.1429   0.1250
   0.2500   0.2000   0.1667   0.1429   0.1250   0.1111
   0.2000   0.1667   0.1429   0.1250   0.1111   0.1000
   0.1667   0.1429   0.1250   0.1111   0.1000   0.0909
b=a
b(1,6)=b(1,6)*1.01
(b^(-1)–a^(-1))./a^(-1)

ans =

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

eig(a)
```

ans =

0.0000
0.0000
0.0006
0.0163
0.2424
1.6189

eig(b)

ans =

1.6191
0.2420
0.0165
0.0006
0.0000
-0.0000

d=cond(a)
d =

1.4951e+007
det(a)

ans =

5.3673e-018

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

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

Розглянемо приклад

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

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

   > b=a;
   > b(5,1)=b(5,1)*1.01;
   > [eig(a),eig(b)]
   > ans =
ans =

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

> cond(a)
   ans =
   2.0253e+018
```

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

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

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

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

2.1 "Логічні"змінні

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

Аналогічно визначені логічні операції:

- & - логічне І (AND). Результатом операції *A&B* є матриця того ж розміру, кожен компонент якої рівний 1, якщо обидва відповідні компоненти матриць *A* і *B* не рівні 0, і рівний 0 інакше (тобто, якщо хоча б один з компонент дорівнює 0).
- | - логічне АБО (OR). Результатом операції *A*|*B* є матриця того ж розміру, кожен компонент якої рівний 1, якщо хоч би один з відповідних компонент матриць *A* і *B* не рівні 0, і рівний 0 інакше (тобто, якщо обидва компоненти рівні 0).

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

Наприклад

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

a =

    1    2

    3    4

>> b=[4 2; 3 1]


b =


    4    2

    3    1

>> c= a>=b

c =

    0    1

    1    1


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

d =


    1    1

    1    0
```

2.2 Умовний оператор

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

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

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

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

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

Для ясності приведемо два наступні приклади

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

    1     0
    0     1
>> b=[1 1; 1 1]

b =

    1     1
    1     1

>> d=[1 1; 1 0]

d =

    1     1
    1     0

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

c =

    1     1
    1     1
```

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


c =


    1     0

    0     1
```

Також ці операції використовуються звичайним способом на випадок скалярних величин, і саме таке традиційне звернення до оператора порівняння можна зустріти в програмах *if a <b*
c=a; elseif
a==b,
c=1;
else
c=b;
end

2.3 Організація циклів

При програмуванні в пакеті MATLAB можливе використання двох видів циклів

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

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

for var = Вираз,
Інструкції
end

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

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


    1     2     3
    4     5     6
```

>> for i = a; i, end

i =

1
4

i =

2
5

i =

3
6

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

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

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

a =

2	3	4
3	4	5
4	5	6

(3)

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

while Логічна змінна, Інструкції
end

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

Традиційний приклад:

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

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

Менш традиційним слід рахувати приклад вигляду

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

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

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

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

2.4 Перемикач *switch-case-otherwise-end*

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

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

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

Як приклад розглянемо сценарій asw.m:

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

```
case {6,7,8,'June','July','August'}
disp('Summer')
case {9,10,11,'September','October','November'}
disp('Autumn')
otherwise
disp('It is not a month.')
   end
```

Звернення до цього сценарію приводить до наступних результатів:

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

3 Спеціальні можливості при зверненні до функцій

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

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

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

3.1 Звернення до функції по імені

У ряді випадків (наприклад при рішенні нелінійних рівнянь або чисельній інтеграції звичайних диференціальних рівнянь) необхідно передати функції як аргумент ім'я деякої іншої функції (наприклад, ім'я функції обчислення

правих або лівих частин рівнянь), яка буде викликана по цьому імені. Ім'я функції або змінної у такому разі зберігається у вигляді символьних одновимірних масивах (векторах). Такий масив можна передати функції як формальний параметр.

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

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

eval(command)

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

```
for k=5:10

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

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

[outpars]=feval(FunName,inparl,inpar2...)

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

Наприклад, при зверненні

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

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

3.2 Багатовимірні масиви

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

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

```
>> a=[1 2; 3 4]
a =
1 2
3 4
>> a(:,:,2)=3
```

a(:,:,1) =
1 2
3 4
a(:,:,2) =
3 3
3 3

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

a = ones(n1, n2 .. ., nn)

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

•функція

n = ndims(a)

дозволяє визначити число размерностей багатовимірного масиву *a.*

•функція

n = size(a)

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

•Функція *B = cat(dim,A1,A2,A3,A4...)* дозволяє об'єднати масиви *A1,A2,A3,A4...* у один уздовж розмірності *dim*. Розглянемо простій приклад

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

B3(:,:,1) =
1 1
1 1
B3(:,:,2) =
1 0

0 1

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

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

3.3 Комірки і операції над ними

Масиви комірок використовуються в MATLAB'і для зберігання і передачі даних різних типів. Створити масив комірок можна двома способами: з використанням функції *cell* або виведенням об'єктів у фігурні дужки. Наприклад, якщо в робочу область пам'яті введена матриця *a* розміру 2x2, наступна команда вводить масив комірок розміру 2 x 3 :
>>b={min(a) max(a) 3; 5 1 'text' }
b =
[1 . 2double] [1 . 2double] [3]

[5] [1] 'text'

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

```
>>b{2,3}
ans =
text
>>b{:,1}
ans =
1 2
ans =

   5
```

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

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

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

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

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

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

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

Приведемо приклад опису такої функції:

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

...

varargout=my2fun(b, varargin);

...

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

3.4 Глобальні змінні

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

Команда

global ім'я1,ім'я2 ...

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

Приведемо приклад використання глобальних змінних

Звернення до цієї функції з командного вікна приводить до результату *»*

mylfun(3,4)

5 7

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

```
function z=my1fun(a,b)
global c
c=b;
d=my2fun(a);
disp ([c,d])
. . .
function d=my2fun(b);
d=my3fun(b);
. . .
function d=my3fun(b);
global c
d=b+c;
c=5;
```

Звернення до цієї функції з командного вікна приводить до результату

» mylfun(3,4)

5 7

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

4 Графічні функції пакету MATLAB

4.1 Двовимірна графіка

4.1.1 Креслення графіків в декартових координатах

Основною функцією малювання графіків в пакеті MATLAB є функція *plot.* Звернення до цієї функції мають вигляд

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

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

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

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

  Наприклад, послідовне виконання команд

  t=0:0.05:pi;

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

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

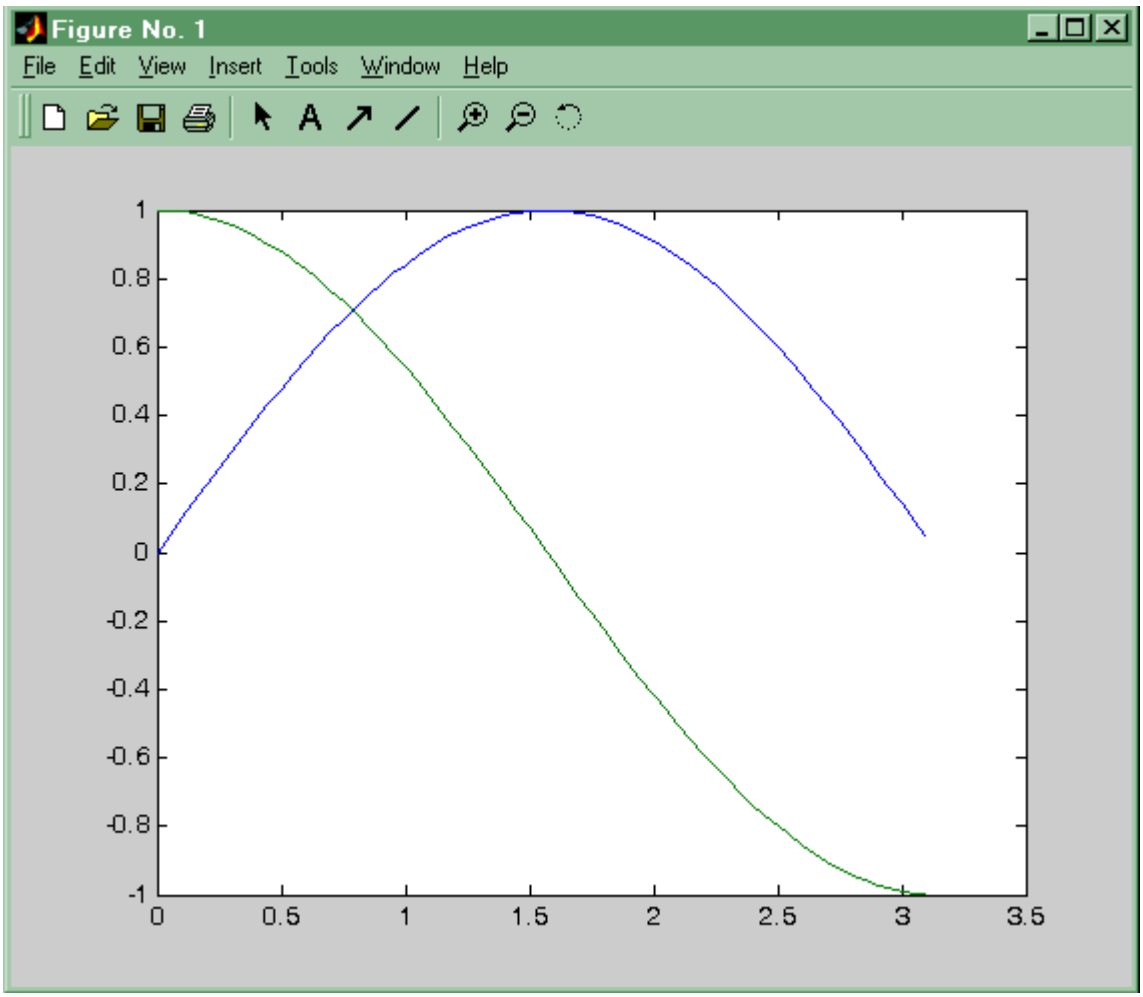

Рис. 1: Результат дії функції *plot*

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

• *plot(x, y, s*).

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

Таблиця 1 – значення змінної s

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

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

Перший з них - звернення

plot(x1, y1, s1, x2, y2, s2, x3, y3, s3. ..).

В цьому випадку в одних і тих же координатних осях будуть побудовані графіки *y1* від *x1* відповідно до рядкової константи *s1*, графіки *y2* від *x2* відповідно до рядкової константи s*2* і т.д. При цьому звичайно ж повинна дотримуватися відповідність розмірів серед пар масивів *x1* і *y1*, *x2 і y2* і т.д., але число точок в різних парах може не співпадати. Як приклад, наведемо наступну послідовність команд

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

В результаті у відкритому графічному вікні MATLAB'а буде побудовано

зображення, приведене на рис. 2

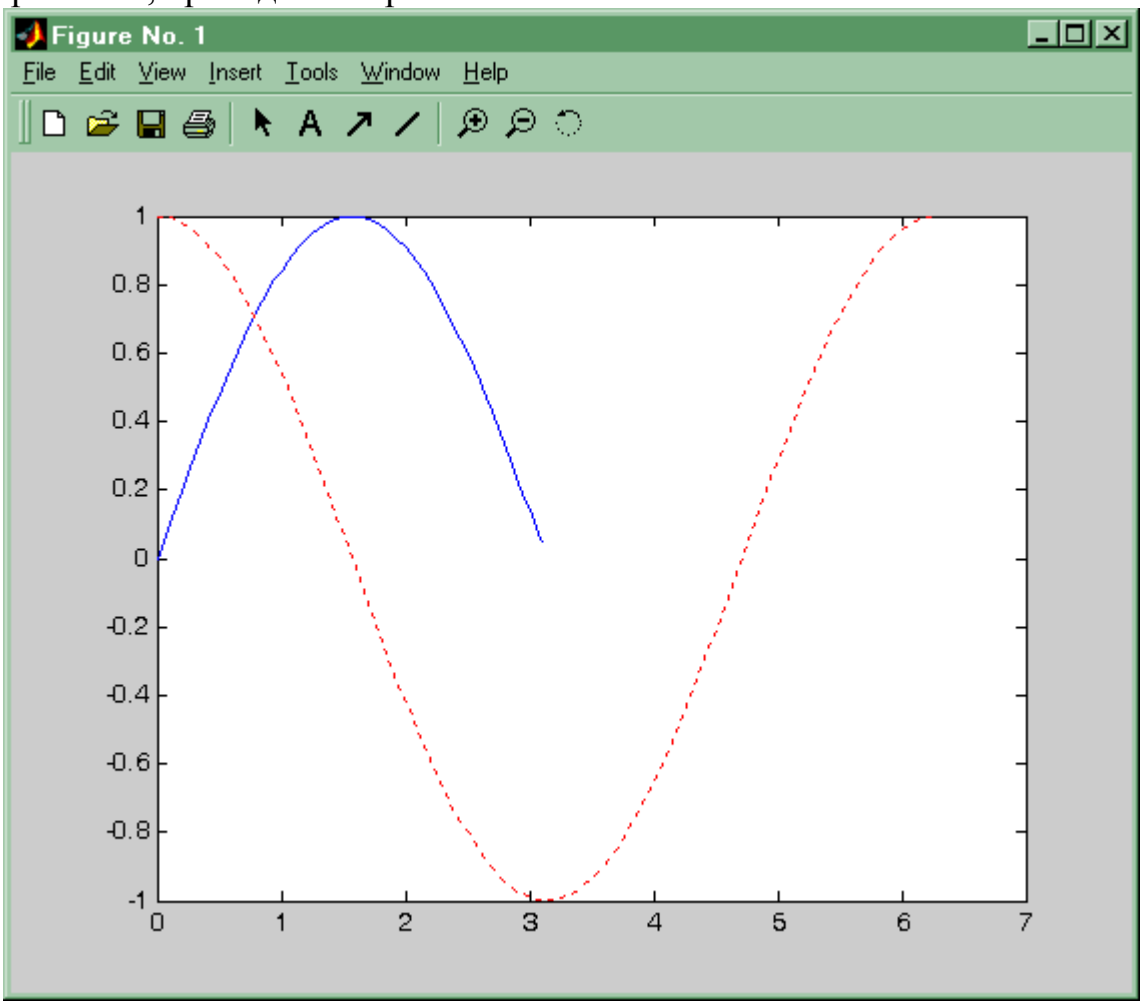

Рис. 2 Ще один приклад дії функції *plot*

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

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

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

```
t=0:0.05:pi;
hold on
plot(t,sin(t), 'b-')
```

*t=0:0.05:2*pi;*
plot(t,cos(t), 'V:');
hold off

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

На закінчення відзначимо, що при зверненні

comet(x,y,p)

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

t=0:0.0001:pi;
*comet(sin(2*t),cos(t))*

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

4.1.2 Графіки в інших системах координат

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

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

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

Аналогічна побудова графіків в полярних координатах, що задаються кутом *phi* і радіусом *rho,* здійснює функція

polar(phi,rho,s)

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

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

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

Результат виконання цих команд приведений на рис. 3.

4.1.3 Сітка, написи і пояснення на графіках

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

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

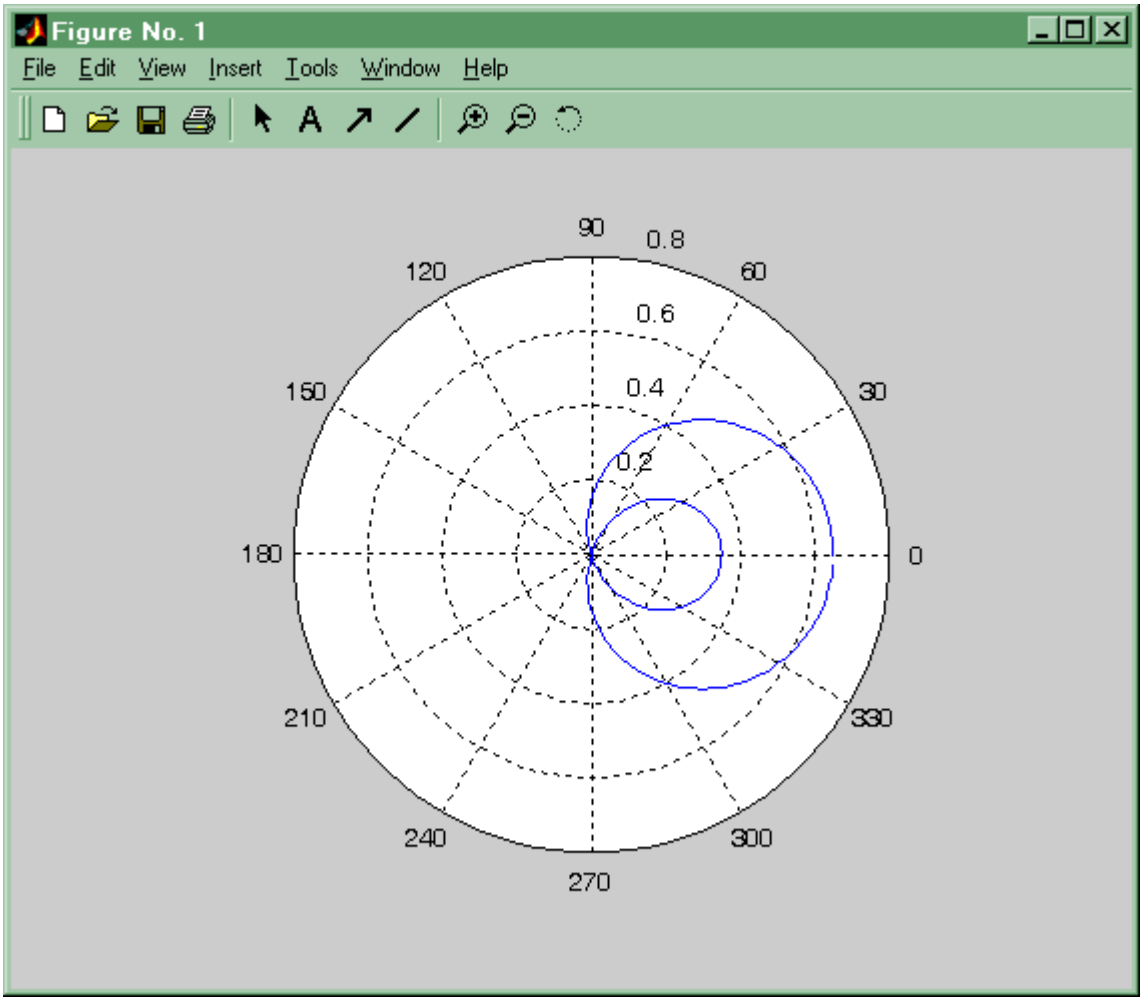

Рис. 3 Результат дії функції *polar*

Зробити написи на малюнках допомагають наступні функції

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

відповідає порядку виведення графіків. Можливі також наступні форми звернення до функції:

—*legend(M)*, де *M* - масив рядків однакової довжини.

—*legend off* видаляє пояснення.

—*legend(stext1, stext2..., n)* - цілий аргумент *n* встановлює граничне значення позицій для розміщення пояснення, *n* = - 1 - пояснення розміщується поза областю графіка, *n* = 0 - пояснення розміщується в області графіка, якщо місця для цього достатньо.

Продемонструємо використання цих функцій на наступному прикладі

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

Результати виконання цих команд наведено на рис. 4.

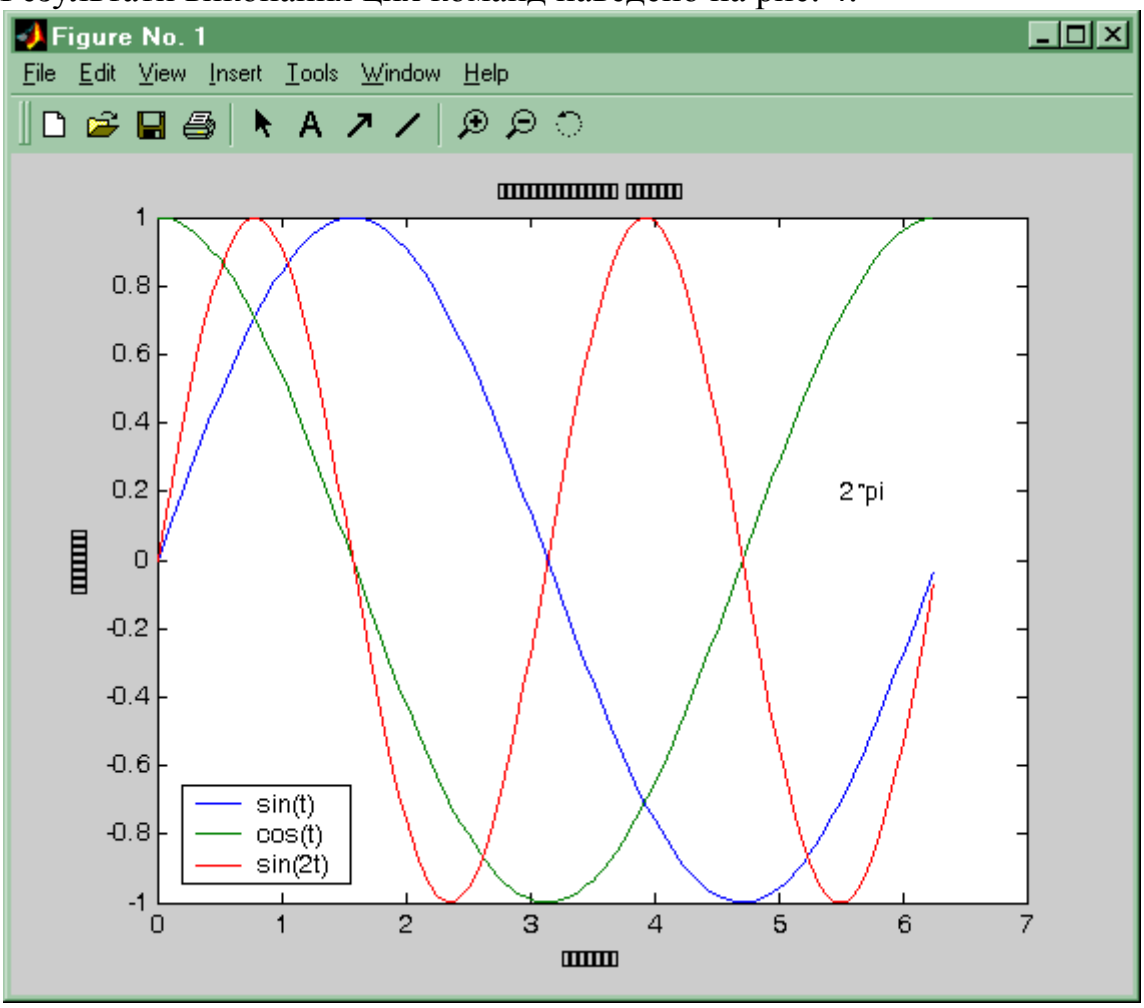

Рис. 4 Зразок написів до графіків

4.1.4 Декілька графіків в одному вікні

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

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

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

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

Продемонструємо відповідні можливості на наступному прикладі

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

title('Траекторія в декартових координатах')

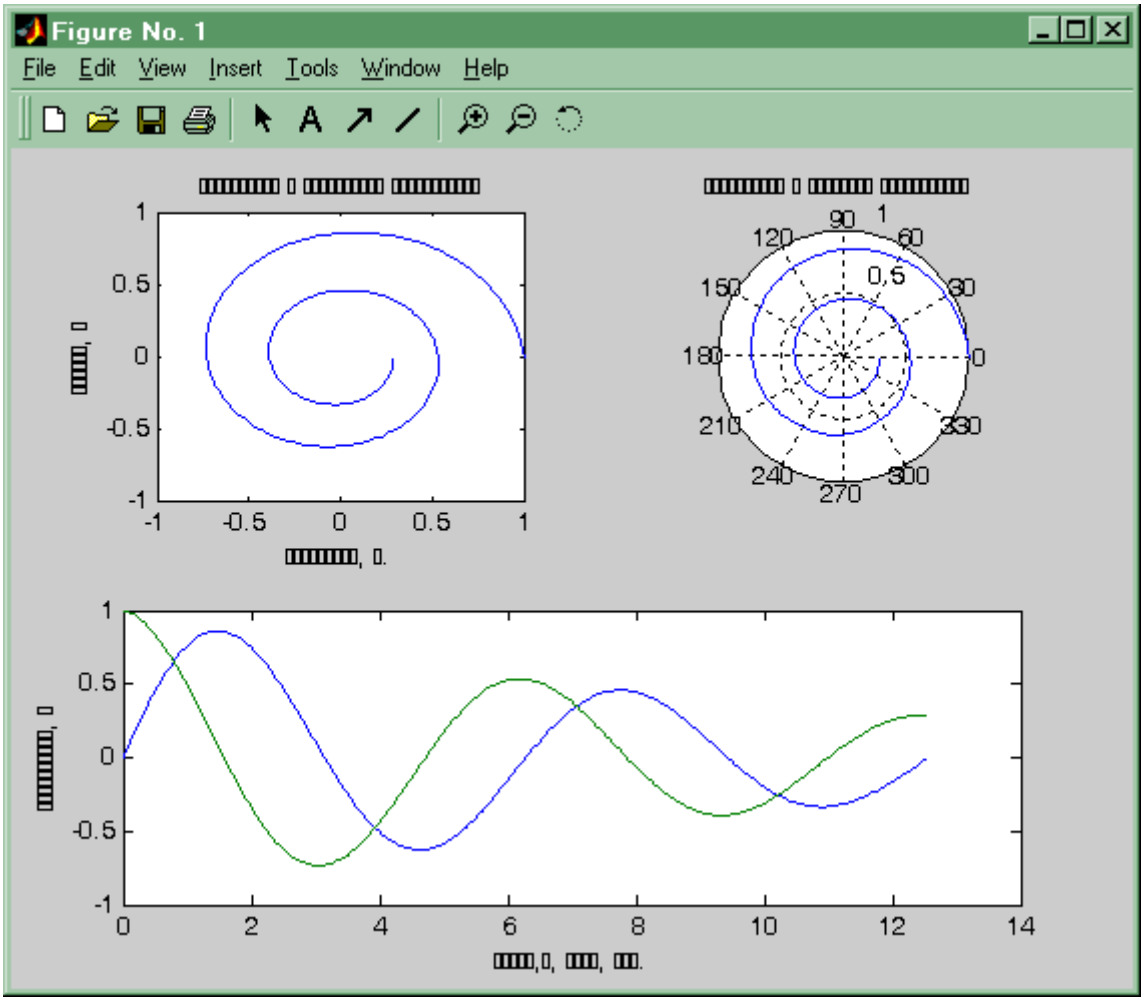

Рис. 5 Розбиття графічного вікна на "підвікна"

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

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

4.1.5 Графічні вікна і управління ними

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

- Для створення нового графічного вікна використовуємо функцію *figure*. За відсутності формальних параметрів звернення *h = figure* створює нове графічне вікно і робить його активним. Тепер всі викликані функції побудови графіків будуватимуть зображення в цьому графічному вікні. Звернення *h = figure(N)* для будь-якого цілого числа *N* створює нове

графічне вікно з номером *N*, якщо такого вікна не існувало, і робить його активним. У обох випадках функція повертає покажчик на вказане графічне вікно. Якщо *h* покажчик на графічне вікно звернення *figure(h)* робить це графічне вікно активним.

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

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

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

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

4.1.6 Діаграми, гістограми, вектора

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

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

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

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

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

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

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

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

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

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

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

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

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

- **Ступінчастий графік,** *stairs*(*x, Y, s)* - представляє залежність *Y* від *x* у вигляді ступінчастої функції. Форми звернення до неї ті ж, що і для функції *plot.*

- **Графік дискретних відліків.** *stem*(*x, Y, s)* - представляє дискретні значення залежності *Y* від *x,* сполучені вертикальною лінією з віссю абсцис. Форми звернення схожі з функцією *plot.*

- **Графік з інформацією про погрішності.** *errorbar(x, Y, L, U)* - представляє дискретні значення залежності *Y* від *x* з вказівкою довірчих інтервалів значень функції. Верхні і нижні межі цих інтервалів задаються аргументами *L* і *U.* При зверненні *errorbar(x, Y, V)* межі довірчих інтервалів встановлюються *L = Y — V nU = Y+V.*

- **Графік проекцій векторів на площину.** Для побудови векторів на площині використовується функція *feather.* При зверненні *feather(U, V)* на координатній площині будуть побудовані вектора, почала яких знаходяться в крапках з нульовими ординатами і абсцисами *X0 =* 1 : *length(U)*, а закінчення в крапках з координатами (*X0 + U,V).* Звернення *feather(Z)* для комплексного вектора *Z* еквівалентно зверненню *feather(real(Z),imag(Z)).*

- **Графік векторів в полярних координатах.** Для зображення векторів використовується також функція *compass. compass*(*U, V)* будує на площині вектора, почала яких знаходяться в крапці з нульовими координатами, а закінчення в крапках з координатами *(U,V).* Звернення *compass(Z)* для комплексного вектора *Z* еквівалентно зверненню *compass(real(Z), imag(Z)).*

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

4.2 Тривимірна графіка

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

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

plot3(x1, y1, z1, s).

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

*x=(1:1000)/1000*3*pi;*

*plot3(x, sin(10*x),x.^ 2);*

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

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

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

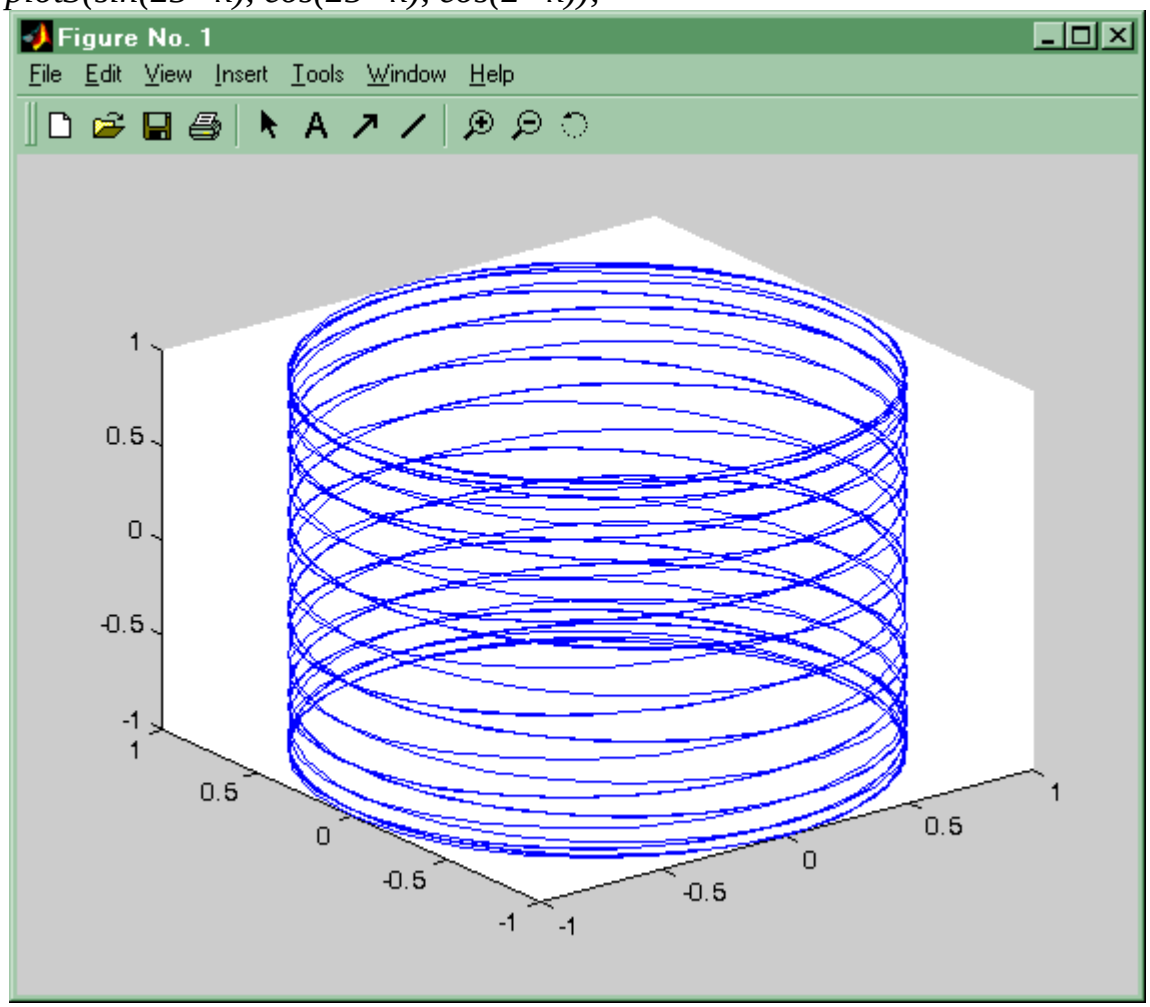

Рис. 6 Результати роботи оператора *plot3*

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

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

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

Написи до зображення і конкретних осей для тривимірних зображень

виконуються вже відомим способом, описаним в розділі 4.1.3. Для тривимірної графіки додатково введені функції *zgrid* і *zlabel* аналогічні описаним раніше *xgrid, ygrid* і *xlabel, ylabel.*

4.2.2 Поверхні в просторі

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

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

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

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

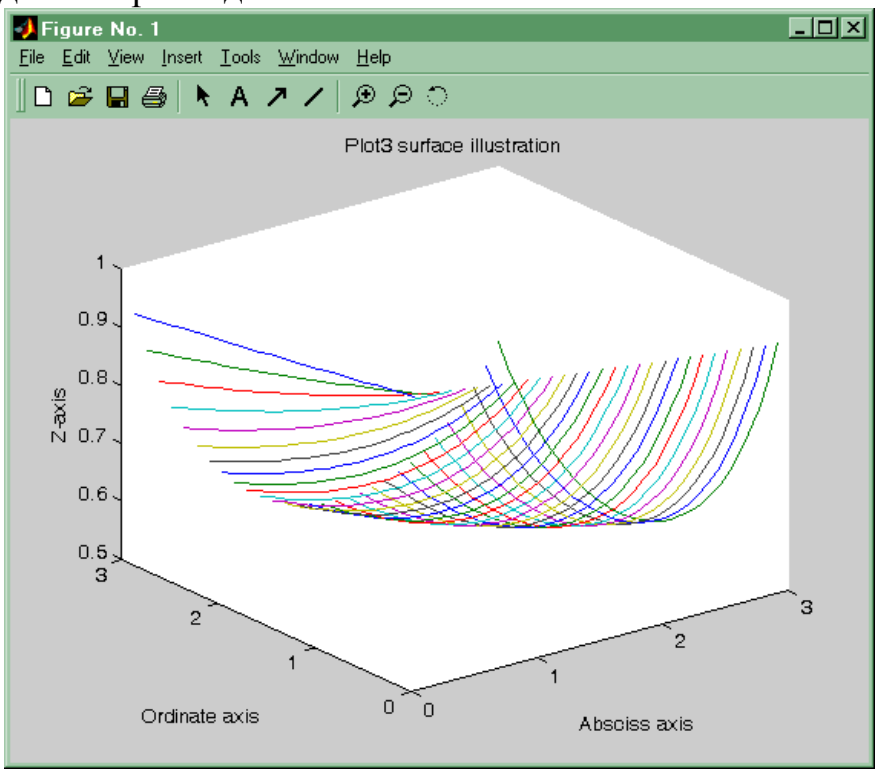

Рис. 7 Незрозумілий результат дії оператора plot3

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

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

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

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

Для всіх вищеперелічених функцій допустимі види звернень типу

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

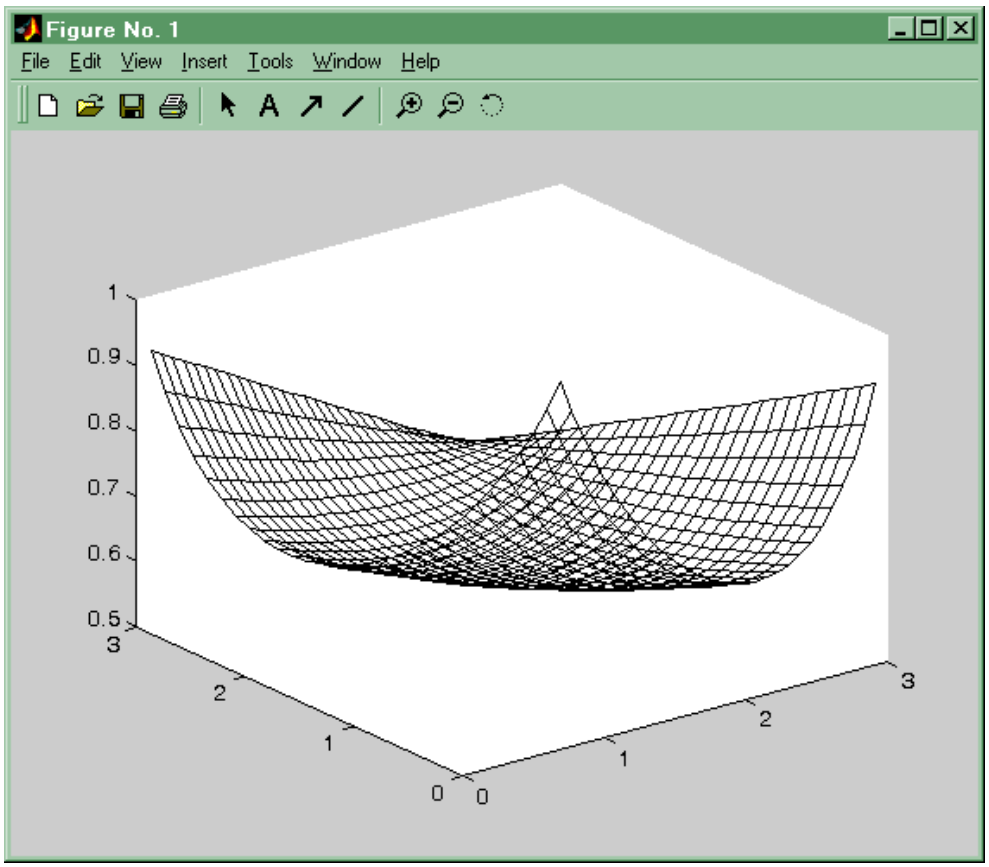

Рис. 8 Результат будування поверхні функцією plot3

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

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

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

[px,py] = gradient(Z, hx, hy);

quiver(X, Y,px,py)

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

Нарешті відмітимо, що для побудови матриць *X* і *Y* по векторах *x* і *y* можна використовувати спеціальну функцію, звернення до якої має вигляд

[X, Y] = meshgrid(*x,y).*

4.2.3 Вигляд зображення з різних боків

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

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

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

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

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

- *view(2)* встановлює штатне положення "точки перегляду" для двовимірної графіки (*Az* = 0°, *El* = 90°).
- *view(3)* встановлює штатне положення "точки перегляду" для тривимірної графіки (оксанометрична проекція – *Az = -37.5°, El = 30°).*
- *[Az, El] = view* - повертає поточне значення кутів азимута і піднесення.
- *T = view* повертає поточне значення узагальненої матриці *T* перетворень, використовуваною для отримання зображення. Цю матрицю можна отримати також звернувшись до функції.
  T = viewmtx(*Az, El).* Звернення *view(T)* встановлює положення "точки перегляду", що задається матрицею *T.*

Як приклад розглянемо поверхню

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

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

Az = 60.

Розібратися в ситуації з картинкою, окрім вашої просторової уяви дозволяє функція *colorbar,* яка додає до поточного графіка шкалу палітри. При зверненні до цієї функції без аргументів або зверненні *colorbar* (*'vert')* ця шкала вертикальна. При зверненні *colorbar*(*'horiz')* - вона горизонтальна.

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

4.2.4 Друк, зберігання і експорт зображень

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

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

Проблеми виникають у випадку, якщо Вам необхідно включити зображення в звіт. При наборі звіту в редакторові MS Word можна скористатися стандартним шаблоном M-book, що поставляється спільно з системою MATLAB. Для використання інших редакторів слід експортувати зображення використовуючи Clipboard системи Windows. Для цього необхідно в меню графічного вікна вибрати позицію "Edit", а в тому, що відкрився підміню позицію "Copy Figure". Залежно від установок проводиться копіювання зображення в одному з форматів "Windows Metafile "або "Windows Bitmap". Ці зображення можна відновити в будь-якому графічному редакторові з використанням пункту підміню "Paste"flH6o комбінації клавіш Ctrl-v. Для формату "Windows Metafile" при відновленні зображення в редакторах Paint, Paintbrush, Adobe Photoshop і подібних до них спостерігається спотворення шрифтів в написах. При використанні формату "Windows Bitmap "шрифти передаються без спотворень. Для установки відповідного формату необхідно в меню графічного вікна вибрати позицію "File", а в тому, що відкрився підміню позицію "Preferences...". У вікні, що знов відкрилося, слідує вибрати сторінку із закладкою "Copying Options' в

розділі "Clipboard Format "встановити вибраний Вами формат.

5 Числовий розв'язок задачі Коші для звичайних диференціальних рівнянь

У пакеті MATLAB існує декілька можливостей розв'язку задачі Коші для звичайних диференціальних рівнянь: стандартні ODE-вирішувачі, спеціалізовані програми дослідження моделей керованих систем, засоби імітації аналогового моделювання середовища SIMULINK. Виклад засобів середовища SIMULINK може бути предметом окремого обговорення (див. наприклад [10], [5]). Дві перші можливості розглянемо тут докладніше.

5.1 Стандартні ODE-вирішувачі.

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

Вказані функції реалізують наступні методи рішення систем звичайних диференціальних рівнянь

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

Всі вказані функції призначені для числового розв'язку задачі Коші для

систем звичайних диференціальних рівнянь вигляду.

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

де *y* - вектор довжини *n.*

Вирішувачі *odel5s, ode23s, ode23t, ode23tb* призначені також для вирішення завдання Коші для систем звичайних диференціальних рівнянь неявного вигляду

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

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

Звернення до функцій числового інтегрування може приймати наступні форми:

[T, Y] = solver('Fun',tspan,y0)
[T,Y] = solver('Fun',tspan,y0, options)
[T,Y] = solver(Fun,tspan,y0,options,p1,p2 …)
[T,Y,TE,YE,IE] = solver('Fun',tspan,y0,options,p1,p2…)

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

function f = Fun(t, y) (6)
function f = Fun(t, y, flag) (7)
function f= Fun(t,y, flag,p1,p2…) (8)

- *tspan* - вектор значень моментів часу. Якщо вектор *tspan* складається з двох компонент *[t0,tfin]*, то ці значення сприймаються як початкове і фінальне значення аргументу. Для вектора більшої довжини *[t0,t1...,tfin]*, компоненти якого розташовані в порядку зростання або спадання, результати числового інтегрування видаються для вказаних значень аргументу.
- *y0* - вектор початкових значень.
- *options* - аргумент задаючий параметри для числового інтегрування. Цей аргумент задається функцією *odeset,* використання якої обговорюватиметься нижче.
- *p1,p2...* - довільні параметри, що передаються у функцію *Fun.*

- *T* - вектор значень аргументу, для якого видаються значення функції *y*.
- *Y* - матриця рішень, кожен рядок якої відповідає значенню аргументу, поверненому в *Y*.
- *TE*, *YE*, *IE* - вихідні матриці, в яких виводяться значення аргументу *t*, функції *y* і номери події ('event') для роботи опції фіксації подій.

Простий приклад розв'язку задачі Коши

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

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

за початкових умов

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

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

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

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

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

Результат отримаємо набравши в командному вікні MATLAB'а команду:

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

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

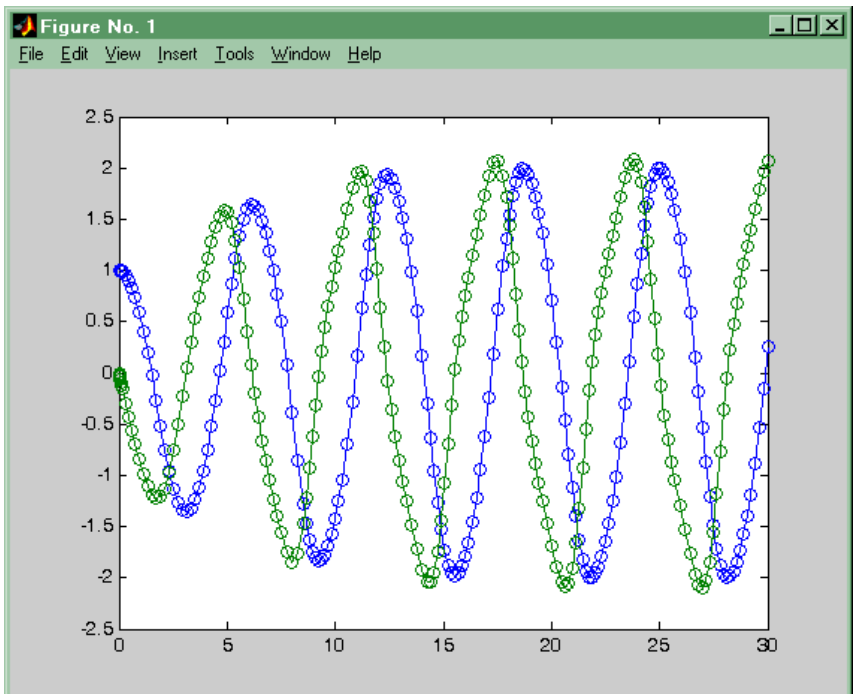

Рис. 9 Результат дії функції ode45

Щоб отримати числові значення розв'язку, використовуємо звернення:

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

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

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

Результат побудови приведений на рис. 10.

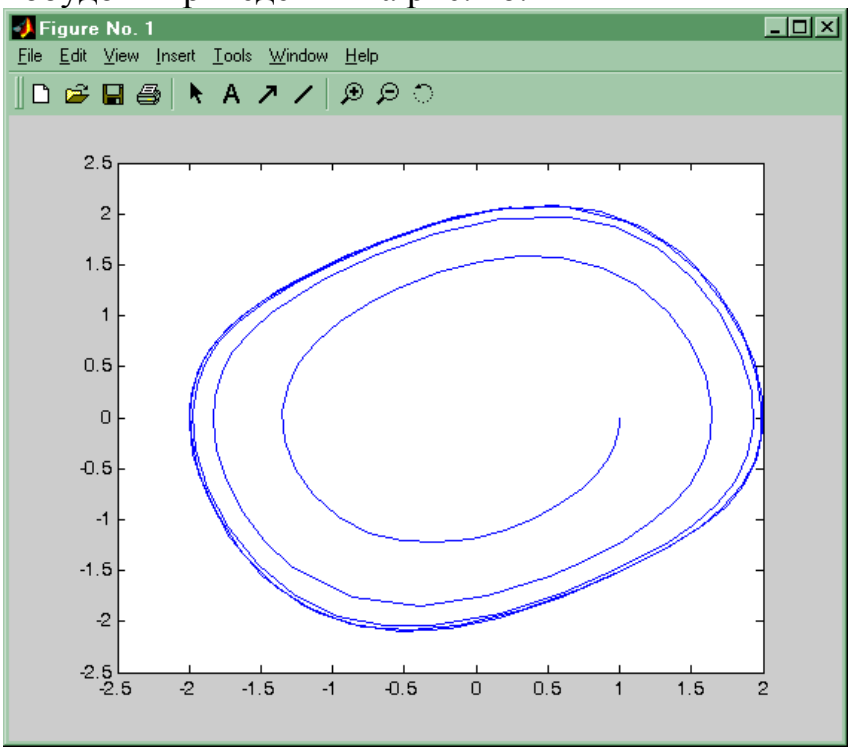

Рис. 10 Фазовий портрет розв'язків рівняння Ван дер Поля.

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

5.2 Опції вирішувача

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

1. *RelTol*- відносна погрішність. Скалярна величина за умовчанням приймається рівною 10^{-3}. Помилка на кожному кроці інтеграції оцінюється величиною $ei < max(RelTol * abs(y_i), AbsTol_i)$.
2. *AbsTol* - вектор абсолютних погрішностей (за умовчанням всі його компоненти рівні 10^{-6}).
3. *Refine* - позитивне ціле число - чинник якості, що підвищує кількість значень аргументу, що виводяться. Використовується для згладжування висновку. За замовчуванням вирішувачі окрім *ode45* використовують значення 1, а *ode45* - 4. Опція *Refine* не використовується, якщо довжина вектора *tspan* більше 2.
4. *OutputFcn* - текстова константа '*AFun',* що містить ім'я додаткової функції виведення. Вирішувач викликатиме вказану функцію після закінчення кожного кроку інтегрування. Зазвичай за умовчанням додаткова функція виведення відсутня, але за відсутності вихідних параметрів вона приймає значення '*odeplot'*. Її роботу ми і бачили в першому з простих прикладів, що наводилися. Для того, щоб на екран комп'ютера виводився графік за наявності вихідних параметрів необхідно встановити значення опції *OutputFcn* рівне '*odeplot'*. При необхідності можна використовувати також функції:

- *odephas2* - зображає двовимірний фазовий портрет;
- *odephas3* - зображає тривимірний фазовий портрет;
- *odeprint* - виводить в командне вікно проміжні результати числового інтегрування.

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

function AFun(t, z)

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

5. *OutputSel* - вектор номерів компонент вектора стану *y,* використовуваних вирішувачем при зверненні до функції додаткового виведення *OutputFcn.* За замовчуванням використовуються всі компоненти вектора стану.

6. *Stats* - відображає обчислювальну статистику після закінчення роботи вирішувача (приймає значення *on/off).*

7. *Jacobian*- ознака можливості обчислення Якобіана (*on/off).* Для використання цієї опції необхідно, щоб звернення до функції *Fun* обчислення правих частин системи рівнянь мало вигляд (7). При цьому при значенні змінної *flag* рівному '*Jacobian'* вказана програма повинна обчислювати матрицю Якобі, компоненти якої мають вигляд $\partial f_i / \partial x_j$.

   Матриця Якобі є такою, що повертається параметром функції *Fun.* Використання цієї опції може значно понизити число кроків і час числового інтегрування.

8. *JConstant* - ця опція повинна приймати значення '*on'* у випадку, якщо матриця Якобі для системи диференціальних рівнянь є постійною.

9. *Events* - ознака використання обробника подій *on/off.* Для використання цієї опції необхідно, щоб звернення до функції *Fun* обчислення правих частин системи рівнянь мало вигляд (7). При значенні змінної *flag* рівному '*Events'* вказана програма повинна повертати наступні величини

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

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

11. *MassConstant* -признак використання постійної матриці "мас"(інерційних коефіцієнтів при похідних в лівій частині рівнянь) [*on|{off}*]. Цей

параметр повинен приймати значення '*on'*, якщо при зверненні до функції *Fun* із змінній *flag* рівною '*Mass'*. Це значення інформує про те, що функція *Fun* повертає постійну матрицю інерційних коефіцієнтів ("мас").

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

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

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

5.3 Програми числового інтегрування лінійних моделей керованих систем

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

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

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

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

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

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

  [Y, T, X] = initial(A, B, C, D, x0, Tf)

  Тут *A, B,C,D* - матриці, що описують систему вигляду (10); *x*0 - вектор початкових умов, *Tf* - значення моменту часу, до якого здійснюється інтегрування. Вихідними параметрами є матриці *Y* і *X,* в яких поміщаються порядково значення векторів *y i x* для моментів часу з відповідного вектора *T*.

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

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

при нульових початкових умовах.

Звернення до цієї функції може мати вигляд:

[Y, T, X] = impulse(A, B, C, D, i, Tf)

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

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

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

Звернення до цієї функції може мати вигляд:

[Y,T,X] = step(A,B,C,D,i,Tf)

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

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

[Y, T, X] = lsim(A, B, C, D, U, T, x0)

Тут *A,B,C,D* - матриці, що описують систему вигляду (10); *U* і *T* - матриця значень векторної функції *u* і відповідних ним моментів часу. Вихідними параметрами є матриці *Y* і *X*, в яких поміщаються відрядкові значення векторів *y* і *x* для моментів часу з відповідного вектора *T*. Матриця *X* і вектор *T* з числа вихідних параметрів може бути опущений. Якщо аргументом *x*0 відсутній, програма проводить обчислення з нульовими початковими умовами.

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

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

Список литературы

1. Потемкин В.Г. Система MATLAB.// М.: Изд. "Диалог-МИФИ", 1997.
2. Потемкин В.Г. MATLAB 5 для студентов.// М.: Изд. "Диалог-МИФИ", 1998.
3. Потемкин В.Г. Система инженерных и научных расчетов MATLAB 5.x.// в 2-х томах. М.: Изд. "Диалог-МИФИ", 1999.
4. Дьяконов В.П. Справочник по применению системы PC MATLAB.// М.: Физматлит, 1993.
5. Дьяконов В.П., Абраменкова И.В. MATLAB 5.0/5.3. Система символьной математики. // М.: Нолидж, 1999.
6. Мартынов Н.Н., Иванов А.П. MATLAB 5.X. Вычисления, визуализация, программирование.-М.: КУДИЦ-ОБРАЗ, 2000.
7. Конев В.Ю., Мироновский Л.А. Основные функции пакета MATLAB.// СПб, Изд. СПГААП, 1992.
8. Коробова Н.Л., Загашвили Ю.В. Комплекс автоматизированного проектирования MATLAB-CTRL.// СПб, Изд. СПГААП, 1993.
9. Андрушевский Б.Р., Фрадков А.Л. Элементы математического моделирования в программных средах MATLAB и Scilab.// СПб, изд. Наука, 2001.
10. Гультяев А. MATLAB 5.2. Имитационное моделирование в среде Windows. // СПб.: КОРОНАпринт, 1999.
11. Медведев В.С. Потемкин В.Г. Control System Toolbox. MATLAB 5 для студентов.// М.: Изд. "Диалог-МИФИ", 1999.
12. Селезнев А.В. Эффективное программирование в среде MATLAB.// в кн. Тезисы докладов всероссийской научной конференции "Проектирование научных и инженерных приложений в среде MATLAB"// М.: ИПУ РАН. 2002. с.199-200.
13. PC MATLAB: User's Guide. MathWorks Inc., 1998.
14. Голуб Дж., Ван Лоун Ч. Матричные вычисления.// М.: Мир. 1999.
15. Воеводин В.В., Кузнецов Ю.А. Матрицы и вычисления.// М.: Наука. 1984.