МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Б.П. ДОВГИЙ
Є.С. ВАКАЛ
А.В. ЛОВЕЙКІН

ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ

Конспект лекцій та практичні заняття

Навчальний посібник

для студентів механіко-математичного-факультету

Київ – 2024

УДК 51-37:519.6 (0.75.08)
Д 58

Рецензенти:

д-р фіз.-мат. наук, член-кореспондент НАН України Ю.В.Крак
канд. фіз.-мат наук, ст. наук. співр. О.Б. Стеля

Рекомендовано до друку
вченою радою механіко-математичного факультету
(протокол № 14 від 02 травня 2024 року)

Довгий Б.П.
Д 58 Інформаційні системи та технології. Конспект лекцій та практичні заняття : навчальний посібник для студентів механіко-математичного факультету / Б.П.Довгий, Є.С.Вакал, А.В. Ловейкін. – К.: 2024. – 225 с.

Навчальне видання містить лекційний матеріал, який складається з основних, базових знань про процеси перетворення інформації в області використання інформаційних систем для розв'язання математичних задач як у числовому так і в символьному вигляді.

За змістом видання (як лекції, так і практичні заняття) поділено на теми згідно з навчальною програмою. Головна увага приділяється питанням практичного застосування інформаційних систем та технологій для розв'язання задач, що виникають в галузі математики. Як базовий інструмент навчання, обрано пакет MATLAB, в якому реалізовано низку стандартних і спеціальних математичних операцій.

Для студентів, аспірантів і викладачів механіко-математичного факультету та факультету комп'ютерних наук та кібернетики.

УДК 51-37:519.6 (0.75.08)

ПЕРЕДМОВА

Інформаційні технології (ІТ) – це процеси, які використовують сукупність засобів і методів збирання, опрацювання і передавання даних (первинної інформації) для отримання інформації нової якості про стан об’єкта, процесу або явища (інформаційного продукту). ІТ є процесом, який складається із чітко регламентованих правил виконання операцій, дій, етапів різного ступеня складності над даними, які зберігаються в комп’ютерах.

Інформаційна система (ІС) – це організаційно-впорядкована взаємопов’язана сукупність засобів і методів ІТ, які використовуються для збереження, обробки і отримання інформації в інтересах досягнення поставленого завдання. Таке поняття ІС передбачає використання ЕОМ і засобів зв’язку як основних технічних засобів обробки інформації, що реалізують інформаційні процеси передачі інформації, необхідної для прийняття рішення, розв’язання задачі із будь-якої предметної області.

Як бачимо, визначення ІС і ІТ є досить широкими поняттями, які відображають сучасне уявлення про процеси перетворення інформації. В залежності від конкретної області використання ІС можуть дуже сильно відрізнятися за своїми функціями, архітектурою, реалізацією.

У навчальному посібнику основну увагу приділено задачам, що виникають в галузі математики. В зв'язку з цим розглядаються питання застосування відповідних інформаційних систем і технологій, які дозволяють:

- представляти дані максимально наближено до математичної області;
- просто, і в той же час ефективно, розв’язувати широкий спектр різноманітних задач математики, як в чисельному так і в символьному вигляді;
- відображати отримані результати у вигляді різноманітних таблиць, діаграм, 2D, 3D-графіків і аналітичних виразів.

Сучасні інтегровані середовища, що ґрунтуються на цих засадах, є універсальними системами комп'ютерної математики, серед яких найвідомішими на сьогоднішній день є MATLAB, Mathcad, Maple, Mathematica, Statistica та Derive. У посібнику як базовий інструмент навчання обрано математичний пакет MATLAB, в якому реалізовано низку стандартних і спеціальних математичних операцій [1, 3–4, 6, 8 –10, 12, 14–15, 17–19, 23–24]. Для їх глибшого і змістовнішого розуміння навчальне видання доповнено коротким теоретичним матеріалом із відповідних розділів наближених обчислень.

ЛЕКЦІЯ № 01

ОСНОВНІ ОБ'ЄКТИ СИСТЕМИ MATLAB

Математичний пакет MATLAB має власну мову програмування, інтерактивне середовище для розробки алгоритмів, візуалізації та аналізу числових розрахунків.

Для практичної роботи будемо використовувати програмний комплекс MATLAB 7.6.0 (R2008a). Виконання програм відбувається в режимі *інтерпретації* операторів.

Правила іменування. У системі MATLAB для іменування об'єктів використовуються *ідентифікатори*, які обов'язково починаються з букви і можуть містити літери латинського алфавіту, цифри й символ підкреслення "_". Ідентифікатор може мати скільки завгодно символів, але значущими є тільки перші (початкові) 31 символ. Доцільно використовувати для позначення об'єктів програми змістовні імена. MATLAB не допускає використання кирилиці в іменах об'єктів.

Типи даних. Майже всі дані в системі MATLAB, зі структурної точки зору, є масивами, отже:

скалярна змінна	–	матриця розмірності 1×1
вектор-рядок з *N* елементів	–	матриця розмірності 1×*N*
вектор-стовпець з *N* елементів	–	матриця розмірності *N*×1

Для маніпуляції об'єктами в системі MATLAB використовуються *дескриптори* функцій – скалярні величини, які зберігають вказівник на функцію і дозволяють передати її іншій функції або викликати локальну функцію ззовні основної функції.

1. Для *числових* типів даних існує наступна типізація (клас) і відповідні імена функцій *перетворення* типів:

`single, double`	– дійсний/комплексний 32 біт, 64 біт;
`int8, int16, int32, int64`	– знаковий цілочисельний 8 біт, 16 біт, 32 біт, 64 біт;
`uint8,uint16,uint32, uint64`	– беззнаковий цілочисельний 8 біт, 16 біт, 32 біт, 64 біт.

За замовчуванням використовується тип `double`, який має найбільшу точність представлення дійсного числа з діапазону від 10^{-308} до 10^{308} (приблизно 16 значущих цифр) і тому є універсальним типом. Однак, при необхідності економити пам'ять ЕОМ, можна вказувати самостійно бажаний тип, наприклад,

```
A = uint8([1,2,3,4,5])
```

Ще одним варіантом економії пам'яті ЕОМ, у випадку роботи з сильно розрідженими масивами, є використання класу `sparse`. Перетворення "повної" матриці `A` (з багатьма нулями) в "розріджену" матрицю `B` здійснюється за допомогою функції `B=sparse(A)`. Зворотне перетворення виконується функцією `full(B)`. Система MATLAB має великий арсенал вбудованих функцій для роботи з `sparse`-матрицями. Враховуючи певну специфіку використання цього типу, ми не будемо докладно зупинятися на його вивченні.

Константи числових типів даних записуються як і в інших алгоритмічних мовах і тип константи визначається при її записі, наприклад,

```
17, -25                    % цілі
435.123, -2.345, .0001     % дійсні (запис з фіксованою крапкою)
```

```
1.248e-12, 1.26E8          % дійсні (запис в експоненціальній формі)
13-2.5i, -2.0345j          % комплексні
```

2. Для *логічного* типу використовується клас `logical` (і відповідна функція перетворення типу) , наприклад, означення логічної змінної:

```
a = logical(1);B = 1>0;B(2) = 1<0;B(3) = true;B(4) = false;
C = logical([1,0,0;0,1,0;0,0,1]);
```

Значенням логічних констант є 1 або 0, яким відповідає *Істина* і *Хибність*. Можна використовувати і позначення `true`, `false`. Дані цього типу мають розмір 1 байт і використовуються для побудови та обчислення *умов* й індексації масивів, наприклад, у випадку заданої вище логічної змінної `C` оператор `Y(C)=X(C)` реалізує заміну діагональних, а оператор `Y(C==0) = X(C==0)` - недіагональних елементів матриці `Y` відповідними елементами матриці `X`.

Зазначимо, що коли елементи матриці `C` визначаються із цілих чисел 1, 0 (`C=[1,0,0;0,1,0;0,0,1];`), то перший варіант запису призводить до помилки.

3. Для *рядкового* типу використовується клас `char` (і відповідна функція перетворення типу). Константи-рядки записуються за допомогою обмежувача – апострофа, наприклад,

```
s='Hello'; x='Розв''язок рівняння Бюргерса'; al='{\alpha}';
```

У результаті отримуємо змінні рядкового типу `s,x` у вигляді лінійного масиву з елементами-символами (`s=['H','e','l','l','o']`). Цю властивість можна використовувати при конкатенації рядків – `S=[s,' !',x]`. Команда `a=double(s)` перетворює символьний масив в числовий, що дає `A = 72 101 108 108 111`, де елементи масиву `A` є кодами символів в певній *кодовій сторінці* (наприклад, "Windows-1251").

Команда `c = char(A)` виконує обернене перетворення числового вектора в рядок і дає `c = 'Hello'`. Для визначення кодової сторінки, встановленої на комп'ютері, необхідно виконати команду `slCharacterEncoding`, результатом виконання якої може бути, наприклад, таке повідомлення

```
ans =
windows-1251
```

З огляду на метод збереження рядків, при групуванні рядків в масив, кожний рядок буде (при необхідності) доповнений справа пропусками (`' '`) до максимальної довжини рядка, наприклад, оператор `A = ['a';'ab';'abcd']` утворює масив `A` з рядками однакової довжини, рівної 4 : `A = ['a   ';'ab  ';'abcd']`.

4. Для *структурного* типу (тип *запис*) використовується клас `struct` (і відповідна функція перетворення типу). Цей тип даних групує дані різних типів з використанням контейнерів даних, які називаються *полями*. Доступ до полів структури виконується за допомогою нотації у формі *structName.fieldName*. Наприклад, ланцюжок команд

```
field1 = 'p1'; field2 = 'p2';
value1 = [1 2 3 4 5]; value2 = 'текст';
S = struct(field1,value1,field2,value2);
```

утворює структуру *S*. Для доступу до значення поля *p1*, можна використати один із наступних еквівалентних синтаксисів: *S.p1*, *S.(field1)*, *S.('p1')*. Створити структуру можна й іншим способом, наприклад, команди:

```
S(2).p1 = [1.2,3,-4,1,34]; S(2).p2 = '4'; S(2).p3 = C;
```

не тільки перетворюють *S* в масив, але і додають нове поле з іменем *p3*, при цьому значення *S(1).p3* порожнє! Різні поля можуть зберігати різні типи даних, у тому числі й структури, але в конкретному полі масиву структур має бути визначений фіксований тип.

При виведенні структури (скаляра) відображаються як імена полів, так і значення. При виведенні структури (масиву) виводиться лише інформація про те, що це – масив-структура, його розмірність й імена полів структури. Якщо необхідно вивести тільки імена полів, то можна використовувати функцію *fieldnames* з одним аргументом – іменем масиву структур, наприклад, *fieldnames(S)*. Для виведення значень відповідного поля масиву структур необхідно вказати *ім'я_масиву.ім'я_поля*, наприклад, *S.p2* .

Для видалення деякого поля з усіх елементів масиву структур, потрібно застосувати функцію *rmfield*, наприклад, *S = rmfield(S,'p3')* – видалення поля з ім'ям *p3* з масиву структур *S*.

Структури досить часто використовуються у вбудованих функціях MATLAB (див., наприклад, *spline*, *ode45*, *bvpinit*, *pdepe*).

5. Для типу *комірка* використовується клас *cell* (і відповідна функція перетворення типу). Цей тип даних групує дані різних типів і структур даних з використанням універсальних контейнерів даних, які доступні за допомогою індексу. Наприклад, команда

```
A={3+2i,[1 2 3],[1 2 3;4 5 6;7 8 9],'Hello'};
```

створить масив комірок із об'єктів різного типу. Таким чином, конструктор *{}* діє подібно конструктору *[]* для масивів, але він об'єднує в масив комірок дані різного типу. Ще приклад: для зберігання матриць однакової розмірності можна використовувати тривимірний масив, а для зберігання матриць різної розмірності – масив комірок

```
M{1}=magic(4); M{2}=magic(3);
M{3}=[1 2 3 4 5]; M{4}=[1 2 3;4 5 6];
```

Масив комірок зручно використовувати і для зберігання рядків, наприклад, оператор *A = {'a','ab','abcd'}* створює масив комірок *A* з рядками різної довжини. Звичайно, масиви комірок можуть бути і багатовимірні. Наприклад, за командою

```
B = cell(2,2);
```

задається матриця *B* розмірності 2×2 із комірок із порожніми (невизначеними) значеннями.

Таку ініціалізацію доцільно виконувати у випадку створення масиву комірок із наступним заповненням значень елементів у циклі, що підвищує швидкість виконання програмного коду.

MATLAB відображає масив комірок у скороченій формі. Для виведення всіх

значень комірок, необхідно застосувати функцію *celldisp*, наприклад, *celldisp(A)*. Для відображення значення конкретної комірки – *A{2}*, а для виведення конкретного значення в комірці – *A{2}(2,3)*. Необхідно звернути увагу на різне відображення результатів виконання команд *A{2} і A(2):*

```
A{2}
ans =
      1 2 3
A(2)
ans =
      [1x3 double]
```

Отримані результати показують, що індекс у фігурних дужках *A{k}* означає вміст (значення) відповідної комірки, а в круглих *A(k)* – це комірка з відповідним вмістом.

Для виведення структури масиву комірок у вигляді графічного зображення використовується функція *cellplot*, наприклад, *cellplot(A)*.

Система MATLAB не очищає масив комірок при виконанні оператора присвоєння і при цьому в пам'яті може залишатись стара інформація в незаповнених комірках. Тому корисно перед виконанням оператора присвоєння *A = …;* скористатись командою *clear A;*.

У системі MATLAB тип *cell* зручно використовувати при програмуванні функції зі змінною кількістю аргументів. Для цього аргументом функції вказується ключове слово *varargin*, яке інтерпретується в тілі функції як масив комірок. Дану ситуацію буде проілюстровано при вивченні функцій.

В пунктах 1–5 розглянуто основні типи даних, які надалі будуть використовуватися. Зазначимо, що для визначення типу конкретного об'єкту, необхідно застосовувати функцію *class* до імені цього об'єкту.

З появою нових релізів системи MATLAB з'являються нові типи даних, які корисні в деяких конкретних випадках. Звичайно, при цьому MATLAB доповнюється множиною інструментів для роботи з цими типами даних. В нових релізах також можуть оновлюватись і доповнюватись вбудовані функції для покращення ефективності роботи пакету.

Змінні. *Змінна* – це певна поіменована величина, яка здатна зберігати деякі, звичайно різні за значеннями, дані відповідних типів. Тип змінної заздалегідь не декларується, а визначається *виразом*, значення якого присвоюється або переприсвоюється змінній. Крім імен змінних і поіменованих констант користувача, в пакеті є *системні змінні*, які мають фіксовані імена і значення. Основні системні змінні, що використовуються в пакеті MATLAB, наведено в таблиці:

Основні системні змінні	
Ім'я	**Призначення**
i, j	уявна одиниця комплексного числа
pi	число π
eps	похибка операцій над числами з плаваючою крапкою
realmin	найменше число з плаваючою крапкою
realmax	найбільше число з плаваючою крапкою

inf	значення машинної нескінченності (∞)
ans	змінна, що зберігає результат останньої операції
NaN	вказівка на нечисловий характер даних (Not-a-Number), позначення невизначеного результату (наприклад, 0/0; *inf* / *inf*)
nargin	кількість вхідних аргументів у функції

Основні системні змінні	
Ім'я	**Призначення**
nargout	кількість вихідних аргументів у функції
computer	тип комп'ютера
flops	кількість операцій з плаваючою крапкою
version	номер версії MATLAB

Системні змінні можна перевизначити на термін поточної сесії (виконання програми), а відновити початкове значення можна оператором `clear`, наприклад,

```
eps = 1e-6;
…
clear eps;
```

Побудова виразів. *Вирази* використовуються для запису послідовності певних обчислень (дій/операцій) над даними для отримання значення конкретного типу. Зазвичай, побудова виразу кожного типу визначається строго, за індуктивними правилами. При цьому чітко визначається пріоритет виконання операцій, як для групи операцій (по типам), так і всередині групи.

В MATLAB визначено наступний пріоритет (у порядку ″старший″ – ″молодший″) за обчисленнями:

1. *арифметичні*;
2. *умови* (в пріоритеті):
   2.1. *відношення* (*предикати*);
   2.2. *логічні операції*.

Порядок виконання операцій в групі може бути змінено за допомогою круглих дужок.

При побудові виразів використовуються не тільки операції, але й *спеціальні символи,* наведені нижче:

Операції і спеціальні символи	
Позначення	**Призначення**
:	операція ″двокрапка″ (має декілька варіантів використання)
( )	дужки для групування операцій або аргументів; зазначення індексів
[ ]	конструктор для формування масивів
{ }	конструктор для формування масивів комірок; зазначення індексів у комірках
.	крапка (має декілька варіантів використання)
..	посилання на батьківський каталог директорії
...	текст рядка команди продовжується на наступний фізичний рядок
пропуск	розділовий знак (між словами, даними, змінними, константами)
end	в індексі масиву означає останній елемент/стовпець/рядок
,	розділовий знак

;	блокування виведення інформації на екран або поділ рядків у масиві
%	знак початку коментаря

В *арифметичних операціях* використовуються числові дані однакової розмірності. Якщо один із операндів є скаляром, а інший ні, то скаляр *поширюється* до розмірів іншого операнду. Набір арифметичних операцій у MATLAB складається з наступних операцій:

Арифметичні операції	
Позначення операції	**Призначення**
+	додавання
-	віднімання
*	добуток
.*	поелементний добуток
^	піднесення до степеня
.^	поелементне піднесення до степеня
\	“ліве” матричне ділення (якщо AX = B, де X – невідоме, тоді X = A \ B)
/	“праве” матричне ділення (якщо XA = B, де X – невідоме, тоді X = B / A)
.\	“ліве” поелементне ділення
./	“праве” поелементне ділення
'	операція транспонування для дійсних масивів і транспонування з одночасним комплексним спряженням для комплексних масивів
.'	операція транспонування

Для групи арифметичних операцій встановлено наступний пріоритет:

Пріоритет арифметичних операцій	
Рівень	**Операція**
1	" .' ", " .^ ", " ' ", " ^ "
2	унарні " + “, “ – ”
3	" .* ", " ./ ”, " .\ ", " * ", " / ", " \ "
4	бінарні " + ", " – "
5	оператор "двокрапка" " : "

Усередині кожного рівня оператори мають однаковий пріоритет і обчислюються зліва направо.

Операції відношення (предикати) мають два операнди і використовуються для порівняння двох величин, у тому числі і масивів. Вони можуть записуватись у вигляді бінарної операції або звернення до функції і представлені в наступній таблиці:

Операції/функції відношення			
Позначення операції	**Функція**	**Призначення**	**Приклад**
==	*eq*	дорівнює	*x==y* або *eq*(*x*,*y*)
~=	*ne*	не дорівнює	*x~=y* або *ne*(*x*,*y*)
<	*lt*	менше	*x<y* або *lt*(*x*,y)
>	*gt*	більше	*x>y* або *gt*(*x*,*y*)
<=	*le*	менше або дорівнює	*x<=y* або *le*(*x*,*y*)
>=	*ge*	більше або дорівнює	*x>=y* або *ge*(*x*,*y*)

При обчисленні предиката ми отримуємо результат 1 – *Істина* або 0 – *Хибність*.

Зазначимо, що порівняння комплексних чисел операціями "<", "<=", ">", ">=" реалізується шляхом порівняння їх дійсних частин.

Логічні операції і функції застосовуються поелементно до даних і використовуються для побудови булевих виразів (умов):

Логічні операції і функції			
Позначення операції	**Функція**	**Призначення**	**Приклад**
~	*not*	заперечення	*~x* або *not*(*x*)
&	*and*	кон'юнкція	*x&y* або *and*(*x*,*y*)
|	*or*	диз'юнкція	*x|y* або *or*(*x*,*y*)
	xor	виключаюче "або"	*xor*(*x*,*y*)
	all	*Істина,* якщо всі елементи вектора ≠0	*all*(*x*)
	any	*Істина,* якщо не всі елементи вектора =0	*any*(*x*)
	exist	перевірка існування змінної чи функції	*exist*('*x*')
	find	пошук індексу ненульового елемента (індексів, що задовольняють певні умові)	*find*(*x*)
	isempty	*Істина,* якщо масив порожній	*isempty*(*x*)
	isinf	*Істина,* якщо елемент = ∞	*isinf*(*Inf*)
	isletter	*Істина,* якщо символом є буква	*isleter*('*a*')
	isnan	*Істина,* якщо елемент не є числовим	*isnan*(*NaN*)
	isreal	*Істина,* якщо елемент дійсний	*isreal*(1.2)
	isstr	*Істина,* якщо елементом є текстовий рядок	*isstr*('*qwerty*')

У виразах можливе звернення до *функцій*, які можуть бути *вбудованими* і *М–функціями* (користувача). Зовнішні функції створюються користувачем і містять свої визначення у М–файлах. Вбудовані функції зберігаються у відкомпільованому ядрі системи MATLAB, внаслідок чого вони виконуються гранично швидко. Пакет MATLAB містить значну кількість елементарних і спеціальних функцій. Зі списком елементарних функцій можна ознайомитись за допомогою команди `help elfun`, а зі списком спеціальних функцій – за допомогою команди `help specfun`.

Перелік деяких вбудованих функцій наведено нижче:

Тригонометричні та гіперболічні функції			
Тригонометрична функція	**Обернена Тригонометрична функція**	**Гіперболічна функція**	**Обернена гіперболічна функція**
sin(*x*)	*asin*(*x*)	*sinh*(*x*)	*asinh*(*x*)
cos(*x*)	*acos*(*x*)	*cosh*(*x*)	*acosh*(*x*)
tan(*x*)	*atan*(*x*)	*tanh*(*x*)	*atanh*(*x*)
cot(*x*)	*acot*(*x*)	*coth*(*x*)	*acoth*(*x*)

Трансцендентні функції	
Назва	**Призначення**
exp(*x*)	експонента *x*
log (*x*)	натуральний логарифм *x*
log2(*x*)	логарифм *x* за основою 2
log10(*x*)	логарифм *x* за основою 10
pow2(*x*)	експонента *x* за основою 2
sqrt(*x*)	корінь квадратний з *x*
nextpow2(*x*)	найближча степінь експоненти за основою 2 ≥ *x*

Функції роботи з комплексними числами	
Назва	**Призначення**
abs(*z*)	модуль *z* (|*z*|)
angle(*z*)	аргумент *z* (*arg*(*z*))
conj(*z*)	спряжене до *z* (*z**)
imag(*z*)	уявна частина *z* (*Im*(*z*))
real(*z*)	дійсна частина z (*Re*(*z*))

Функції заокруглення і модульна арифметика	
Назва	**Призначення**
abs(*x*)	абсолютна величина *x* (|*x*|)
fix(*x*)	ціла частина *x* ([*x*])
floor(*x*)	округлення *x* до найближчого меншого цілого (результат – ціле значення)
ceil(*x*)	округлення *x* до найближчого більшого цілого (результат – ціле значення)
round(*x*)	округлення *x* до цілого (результат – дійсне значення)
mod(*x*,*y*)	залишок від цілочисельного ділення *x* на *y* з урахуванням знаку
rem(*x*,*y*)	залишок від цілочисельного ділення *x* на *y*. Порівняйте: *mod*(5.345,2), *rem*(5.345,2); *mod*(–5.5,2), *rem*(–5.5,2)
sign(*x*)	знак числа

Теоретико-числові функції	
Назва	**Призначення**
factor(*n*)	розклад числа *n* на прості множники
isprime(*n*)	*Істина*, якщо число *n* просте
primes(*n*)	формування списку простих чисел, які не перевищують *n*
gcd(*n*,*m*)	найбільший спільний дільник *n*,*m*
lcm(*n*,*m*)	найменше спільне кратне *n*,*m*
rat(*n*)	подання числа *n* у вигляді ланцюгового дробу
rats(*n*)	наближене число у вигляді раціонального дробу вигляду *a*/*b*
perms(*1*:*n*)	можливі перестановки елементів заданого числового вектора
nchoosek(*n*,*k*)	число сполучень (C_n^k)

Функції для операцій з елементами матриць	
Назва	**Призначення**
diag(*A*)	створення або видобування діагоналі матриці. Якщо *A* є вектором, то створюється квадратна матриця, в якій елементи вектора *A* розміщені на головній діагоналі. Якщо *A* є матрицею, створюється вектор-стовпець, що складається з елементів головної діагоналі матриці *A*
fliplr(*A*)	відбиття стовпців матриці *A* відносно вертикальної осі
flipud(*A*)	відбиття рядків матриці *A* відносно горизонтальної осі
isreal(*A*)	*Істина*, якщо всі елементи матриці *A* дійсні
reshape(*A*,*m*,*n*)	зміна розмірності матриці без зміни кількості її елементів. (див. *help reshape*) Утворюється матриця розмірності *m*×*n*. Нехай *A* – матриця розмірності 3×4, тоді можна утворити матриці: *reshape*(*A*,1,12); *reshape*(*A*,12,1); *reshape*(*A*,2,6); *reshape*(*A*,3,[ ]); *reshape*(*A*,[ ],4);
rot90(*A*,*k*)	поворот матриці *A* на 90**k* градусів, де *k* = 1, –1, 2, –2,…. Якщо *k* відсутнє, то приймається *k* = 1
tril(*A*,*k*)	створення лівої трикутної матриці, починаючи з діагоналі (*головна* + *k*), де при *k*>0 – відлік вправо, а при *k*<0 – вліво від головної діагоналі. Якщо *k* відсутнє, то приймається *k*=0.
triu(*A*,*k*)	створення правої трикутної матриці, починаючи з діагоналі (*головна* + *k*),

	де при $k>0$ – відлік вправо, а при $k<0$ – вліво від головної діагоналі. Якщо k відсутнє, то приймається $k=0$
:	виділення стовпця/рядка в матриці або генерація елементів вектора
[n,m]=size(A)	визначення розмірності матриці *A (*кількість рядків і стовпців)
ndims(*A*)	визначення кількості вимірів у просторовому представленні масиву *A*
s=length(A)	максимальний розмір матриці *A* (*s = max(size(A))*) – максимум із кількості рядків і стовпців

Функції для роботи з рядками символів	
Назва	**Призначення**
abs(‘*i*’)	код символу *i*
blanks(*n*)	створення рядка з *n* пробілів
deblank(*s*)	видалення пробілів у кінці рядка
double(‘*xyz*’)	перетворення символів рядка ‘*xyz*’ у числове значення
strcmp(*s*,*t*)	порівняння рядків *s* і *t*
upper(*s*)	переведення всіх символів рядка *s* у верхній регістр
lower(*s*)	переведення всіх символів рядка *s* у нижній регістр
setstr(*n*)	перетворення числа *n* у рядок
int2str(*n*)	перетворення цілого числа *n* у рядок
num2str(*x*)	перетворення дійсного числа *x* у рядок
str2num(*s*)	перетворення рядка *s* у число
findstr(*s*,*p*)	пошук місця рядка *p* у рядку *s*
strrep(*s*,*c*,*p*)	заміна в рядку *s* рядка *c* на рядок *p*
strcat(*s*,*c*)	горизонтальне об’єднання рядків
strvcat(*s*,*c*)	вертикальне об’єднання рядків
eval(*s*)	обчислення рядкових виразів

Функції часу та дати	
Назва	**Призначення**
clock	поточний час і дата у векторній формі [рік, місяць, день, година, хвилина, секунда]
date	поточна дата у формі рядка
now	поточний час і дата у формі числа
datestr(*d*,*n*)	перетворення числового формату дати *d* в один із 19 вихідних рядкових форматів дати і часу (*n* – номер формату від 0 до 18)
etime(*t2*,*tl*)	визначення тривалості проміжку часу (в сек), заданого векторами *t1* і *t2*

Враховуючи певну специфіку спеціальних математичних функцій, наведемо тільки їх відповідні назви:

- функції Бесселя (`bessel, besseli, besselj, besselk, bessely`);
- бета-функція (`beta, betainc, betaln`);
- еліптичні функції Якобі (`ellipj, ellipke`);
- гама-функції (`gamma, gammainc, gammaln`);
- функції Лежандра (`legandre`).

Для детального ознайомлення з відповідною функцією можна використати команду `help` <*name_function*>, де <*name_function*> – ім’я функції.

ЛЕКЦІЯ № 02

ОСНОВНІ СТРУКТУРИ КЕРУВАННЯ АЛГОРИТМІЧНОЇ МОВИ MATLAB

Враховуючи, що алгоритми, створювані користувачем, як правило, складаються із серії команд і можуть виконуватись не один раз, використання інтерактивного середовища для запуску і виконання команд у командному вікні не є найкращим варіантом роботи з пакетом MATLAB. Отже потрібно створювати програми. Для цього в системі MATLAB є своя алгоритмічна мова [15, 18, 23].

Ланцюг команд. Команди мови, як і команди у командному вікні, записуються по одній у фізичний рядок. Якщо після запису тексту команди відсутній розділювач " ; ", то після виконання цієї команди на екран буде виведено результат. За наявності знаку " ; " здійснюється блокування виведення. Для запису декількох команд в одному фізичному рядку використовується розділювач " , " (без блокування виведення) або розділювач " ; ". Частину довгого виразу можна перенести на наступний рядок за допомогою символу "…" (краще поділити на окремі частини і реалізувати їх додатковими операторами присвоєння !).

Оператор присвоєння. В системі MATLAB можна задавати змінним певні значення. З цією метою використовується оператор присвоєння, який має синтаксис:

x = E, де *x* – ім'я змінної, а *E* – вираз.

Семантика присвоєння реалізується у наступний спосіб – обчислюємо значення (позначимо його через *e*) виразу *E* у правій частині оператора, яке і стає значенням змінної *x*. В MATLAB пам'ять під змінні заздалегідь не виділяється, тому, якщо змінна *x* не існує, то вона створюється. У випадку існування *x* – попереднє значення змінної *x* стає недосяжним (або модифікується), а тип *x* приводиться до *class(e)*.

Приклад 1. Знаходження коренів сіткової функції.

```
%%pl2_1.m
close all          % закриття усіх графічних вікон
clear all          % очистка робочої області
clc                % очистка командного вікна
X = [-2 : 0.1 : 3];
Y = (X + 1) .*  ...   % приклад продовження команди на наступний рядок
    (X - 2);
% Знаходження коренів сіткової функції Y
n = length(X);  w = 1 : n - 1;
% пошук індексу масиву значень функції Y, для якого виконується умова,
% тоді значення елементу масиву X з цим індексом дає корінь
X(find(Y(w) .* Y(w + 1) < 0 | Y(w) == 0))
% Графічний розв'язок
% Виведення графіку функції і осі абсцис
plot(X, Y, X, zeros(n),'LineWidth',2), grid on
```

Розгалуження. У мові MATLAB ця *керуюча структура* представлена декількома варіантами, які мають свою специфіку виконання (семантику). Незалежно від варіанту розгалуження його запис обов'язково має закінчуватися ключовим словом *end*. Тоді команда, розташована після нього, буде виконуватися наступною.

а) Синтаксис *спрощеного оператора розгалуження* –

```
if F
    P;
```

```
end
```
Семантика: якщо умова *F* істинна, то виконується група команд *P*, у протилежному випадку оператор не виконує жодної команди.

b) Синтаксис *оператора розгалуження* має вигляд:

```
if F
    P;
else
    Q;
end
```
Семантика: якщо після обчислення умови *F* отримали значення *Істина*, то виконується група команд *P*, а при значенні *Хибність* – група команд *Q*.

Зазначимо, що оператори розгалуження (як **a** так і **b**) можуть бути вкладеними.

c) Для спрощення запису вкладеності операторів розгалуження застосовується *оператор каскадного розгалуження*, який має наступний синтаксис:

```
if F_1
    P_1;
elseif F_2
    P_2;
    …
elseif F_n
    P_n;
else
    Q;
end
```
Семантика: послідовно перевіряємо значення умов *F_k* і перша з умов, яка істинна, приводить до виконання групи команд *P_k*. Якщо всі умови *F_k* хибні і наявна альтернативна частина *else*, то виконується група команд *Q*.

Приклад 2. Використання різних варіантів розгалуження.

```
%%pl2_2.m
% Приклади варіантів розгалужень
close all, clear all, clc
x = input('? x=');
% Спрощений оператор розгалуження
if ~isreal(x)
    x = real(x)
end

% Вкладене розгалуження
if x < -1
    y = -1;
else
    if x < 1
        y = x;
    else
        y = 1;
    end
end
disp(strcat('x=',num2str(x),'    y=',num2str(y)));

% Каскадне розгалуження
if x < -1
```

```
    y = -1;
elseif x < 1
    y = x;
else
    y = 1;
end
disp(strcat('x=',num2str(x),'    y=',num2str(y)))
% Каскадне розгалуження (інша форма запису)
if x < -1,     y = -1;
elseif x < 1,  y =  x;
else,          y =  1;
end
display(strcat('x=',num2str(x),'    y=',num2str(y)))
```

d) В деяких випадках зручно користуватись *оператором вибору*, який має такий синтаксис:

```
switch    E
    case    e_1
        P_1;
    case    e_2
        P_2;
   …
    case    e_n
        P_n;
    otherwise
        Q;
end
```

Тут `E` має бути виразом/змінною зі значенням скаляра або рядка символів, а `e_k` – скаляром, або рядком символів, або масивом комірок із скалярів чи рядків символів.

Семантика такої структури: послідовно перевіряється виконання умов `E == e_1`, `E == e_2`, …, і перша з умов `E == e_k`, яка істинна, призводить до виконання групи команд `P_k`. Якщо жодна з цих умов не справджується і наявна альтернативна частина `otherwise`, то виконується група команд `Q`.

Приклад 3. Використання оператора вибору для перевірки уведеного символу.

```
%%pl2_3.m
% Приклад з оператором вибору
clear all, close all, clc
% вводимо у вигляді запису рядкової константи ('d'|'2'|'('|'.'|'*')
x = input('Введіть символ =');
if isstr(x)
    x = x(1);
end
y = x;
if isletter(y)
    y = lower(y);
end
disp(strcat('Ви ввели символ="',x,'"'));
% Використання оператора вибору
switch y
    case {'a','b','c','d','e','f','g','h','i','j','k','l','m',...
```

```
            'n','o','p','q','r','s','t','u','v','w','x','y','z'}
        disp('Це буква англійського алфавіту');
    case {'0','1','2','3','4','5','6','7','8','9'}
        disp('Це цифра');
    case {'(',')','[',']','{','}'}
        disp('Це дужка');
    case '.'
        disp('Це крапка');
    otherwise
        disp('Інший символ');
end
```

Цикл. Для повторення виконання певної послідовності команд мова MATLAB має структуру керування, яка називається *циклом* або *ітерацією*. Існує декілька варіантів оператора циклу.

a) Синтаксис *оператора циклу з передумовою*:

```
while F      % заголовок циклу
    P;       % "тіло" циклу
end          % кінець циклу
```

Семантика такої структури: Поки значення умови `F` є істинним, виконується група команд `P` тіла циклу. Як тільки значення `F` стає хибним, то здійснюється вихід з циклу, і керування передається командам, розташованим після оператора циклу, тобто після ключового слова `end`.

b) Синтаксис *оператора циклу за діапазоном значень (цикл з лічильником)*:

```
for k = e1 : e2 : e3  % заголовок циклу
    P;                % "тіло" циклу
end                   % кінець циклу
```

Тут `k` – змінна циклу (*лічильник*), `e1,e3` – початкове і кінцеве значення діапазону, `e2` – крок (при значенні `+1` його можна не вказувати) на який змінюється `k` після кожного виконання групи команд `P`. Цикл закінчується, якщо значення лічильника виходить за межі вказаного діапазону.

Наведений синтаксис діапазону відповідає арифметичній прогресії, проте можна використовувати й інші варіанти завдання діапазону:

- одновимірні масиви з числовими, символьними, рядковими елементами;
- одновимірні масиви комірок;
- двовимірні масиви.

Зверніть увагу (приклад №4) на значення лічильника циклу `h`, які він отримує у випадку використання матриці в якості діапазону (варіанти завдання `H` і `H(:)`).

Приклад 4. Використання різних варіантів оператора циклу

```
%%pl2_4.m
% Приклади операторів циклу
clear all, close all, clc

% Цикл з передумовою
% Обчислення константи eps
e = 1;
while 1 + e ~= 1
    e = e./2;
end
e = 2 .* e;
disp({'обчислене eps=', e, 'eps=', eps})
```

```
pause; clc

n = 5; H = zeros(n);
% Цикл за діапазоном значень (з "лічильником")
for i = 1 : n
    for j = 1 : n
        H(i, j) = 1 ./ (i + j - 1);
    end
end
disp('Матриця Гільберта (обчислена)')
H
disp('Використання вбудованої функції hilb')
H = hilb(n)
pause; clc

disp('Діапазон - одновимірний масив чисел H')
H = [-34, 25.8, 123e2]
for h = H
    h
end
pause; clc

disp('H - числова матриця, діапазон = H')
H = [-34, 25.8; 123e2, 2; -4, 234.5]
for h = H
    h
end
disp('H - числова матриця, діапазон = H(:)')
for h = H(:)
    h
end
pause; clc

disp('H - масив комірок, діапазон = H')
H = {-34, 2+5.8i, 'abc123'};
for h = H
    h
end
disp('H - масив комірок, діапазон = H(:)')
for h = H(:)
    h
end
```

При програмуванні ланцюга команд у тілі циклу можуть використовуватися (зазвичай в комбінації зі спрощеним оператором розгалуження) наступні команди:

`continue` – пропуск усіх наступних команд (до `end`) і передача керування на першу команду після заголовка циклу;

`break` – завершення циклу і передача керування наступній команді після `end`.

Введення, виведення, деякі допоміжні команди. Для інтерактивного введення даних (під час виконання програми на MATLAB) використовують команду *введення*:

```
X = input('Текст')
```

або ланцюг команд `input('Текст'), X = ans.`

Семантика команди:

1. виводиться на екран інформація `Текст`;

2. вводиться з клавіатури довільний вираз MATLAB, при цьому константи вводяться так, як вони визначаються в мові;
3. значення цього виразу присвоюється змінній *X* (у другому варіанті системна змінна *ans* теж отримує це значення).

У найпростішому випадку виведення значення на екран здійснюється записом виразу, після якого відсутній символ " ; ". Проте існує можливість виведення даних за допомогою *команд виведення*:

```
disp(E);
display(E);
```

Тут через *E* позначено вираз. У результаті виконання цих команд на екран (у різному вигляді) виводиться значення *E*.

При оформленні діалогу з користувачем корисні команди:

format – впливає на спосіб представлення інформації на екрані, наприклад:

```
>> -123.234
format short              ans = -123.23          (за угодою)
format long               ans = -1.232340000000000e+002
format short e            ans = -1.2323e+002
format long g             ans = -123.234
```

(детальнішу інформацію можна отримати за командою *help format*);

pause – призупиняє виконання програми до натиснення довільної клавіші на клавіатурі;

pause(n) – призупиняє виконання програми на *n* секунд (*n≠0*).

М-сценарії та їх виклик. Зазначимо, що програми, створені користувачем, умовно можна поділити на дві групи:

- *файли-сценарії*, які не мають вхідних параметрів (будемо називати їх *М-файли*);
- *файли-функції*, які мають вхідні параметри (*М-функції*).

М-функції розглядаються в наступних розділах посібника. Зараз зосередимося на М-файлах.

Текст для М-файла (позначимо його ім'я *myfile*) можна підготувати у різний спосіб:

- у командному вікні командою *edit myfile* (буде використано вбудований редактор текстів);
- за допомогою відповідного пункту меню або піктограми на панелі інструментів (також буде використано вбудований редактор текстів);
- у середовищі довільного (зовнішнього по відношенню до MATLAB) текстового редактору (при збереженні треба вказати і розширення файла *myfile.m*).

Текст файла бажано починати з коментаря з інформацією про його призначення, після якого записати такі команди:

```
close all         % закриття всіх графічних вікон
clear all         % очистка робочої області
clc               % очистка командного вікна
```

Зазначимо, що у М-файлі доступна вся інформація, яка є в робочій області.

Виклик М-файла відбувається введенням його імені (без розширення), тобто аналогічно виклику команди користувача у командному вікні, наприклад, *pl2_4*.

ЛЕКЦІЯ № 03

РОБОТА З МАТРИЦЯМИ, МАТРИЧНИЙ АНАЛІЗ

Усі дані MATLAB представляються у вигляді масивів. Дуже важливо розуміти, як їх використовувати. Без цього неможлива ефективна робота в MATLAB, зокрема побудова графіків, розв'язання задач лінійної алгебри, обробка даних тощо. У цьому підрозділі описані основні операції над масивами.

Створення та використання масивів в системі MATLAB

Для створення масиву (стандартно двовимірної дійсної матриці) можна використати:

1. *конструктор масиву* `[]`, в якому записуються необхідні елементи. В загальному випадку це вирази зі скалярними даними або масивами, але можуть бути і порожні елементи. Елементи рядка розділяються символом "*пропуск*" або "`,`", а елементи стовпця – символом "`;`";
2. *операцію* "`:`" для формування *діапазону* значень `e1:e2:e3` – послідовності елементів у вигляді арифметичної прогресії з початковим елементом `e1`, кроком `e2` і кінцевим елементом `e3`. При використанні дійсних параметрів в означенні послідовності слід мати на увазі, що значення `e3` визначається з урахуванням наближених обчислень (може не належати послідовності);
3. *вбудовані функції:*
   - `linspace(e1,e3,n)` – для формування вектор-рядка з послідовності `n` елементів у вигляді арифметичної прогресії з початковим елементом `e1` і кінцевим елементом `e3` (який належить послідовності), а рівномірний крок визначається за формулою `(e3-e1)/(n-1);`
   - `logspace(e1,e3,n)` – для формування вектор-рядка з послідовності `n` елементів у вигляді геометричної прогресії з початковим елементом 10^{e1} і кінцевим елементом 10^{e3}.

   Якщо параметр `n` опущено, то вважається, що `n=100`;
4. *вбудовані функції:*
   - `eye(n,m,тип)` – для створення одиничної матриці ($m=n$) або матриці з одиницями на головній діагоналі ($m \neq n$);
   - `zeros(n,m,тип)` – для створення матриці *з нулів*;
   - `ones(n,m,тип)` – для створення матриці *з одиниць*,

   де `n,m` – кількість рядків, стовпців. При створенні квадратної матриці параметр `m` можна не вказувати. Параметр `тип` – необов'язковий. Зазвичай його використовують для означення типу вихідних даних і вказують у вигляді константи-рядка. Якщо його не вказано, то за замовчуванням вважаємо, що маємо тип `'double'`.
5. *вбудовані функції* для генерації спеціальних матриць: `compan`, `hadamard`, `hankel`, `hilb`, `invhilb`, `magic`, `pascal`, `rosser`, `toeplitz`, `wilkinson` (див. `help name`, де `name` - ім'я необхідної функції);
6. *вбудовану функцію* `rand(n,m)` для генерації матриці розмірності `n×m` із псевдовипадкових чисел із діапазону від 0 до 1;

7. *вбудовану функцію* `meshgrid` для генерації матриць `X,Y` координатної сітки на базі одновимірних векторів `x,y` (використовуються при побудові 3D-графіків). Після виклику `[X,Y]=meshgrid(x,y)` отримуємо матриці наступного виду:

$$\mathrm{X}=\left[\underbrace{x;x;...;x}_{m}\right],x=\left[x_1,x_2,...,x_n\right];\quad \mathrm{Y}=\left[\overbrace{y,y,...,y}^{n}\right],y=\left[y_1;y_2;...;y_m\right].$$

Приклад 5. Використання різних варіантів створення масивів.

```
%%pl3_1.m
% Приклади створення масивів
close all, clear all, clc
b = [1; -1; 3], A = [11, 3, 2; -1 10 -1; 2, 2 7]
B = [], B = [diag(b), eye(3); diag(diag(A)), A]
y = 0 : 0.501 : 1.1
z = 0 : 1 : 5
x = linspace(0, 1.1, 3), y = linspace(-1, 1.3, 5), z = linspace(-2, 2)
p = logspace(0, 4, 5)
A = eye(2),  A = eye(2, 1),  A = eye(3, 2, 'int8')
A = zeros(2),  A = zeros(2, 3), A = zeros(3, 2, 'int8')
A = ones(2), A = ones(2, 3), A = ones(3, 2, 'uint8')
C = compan([1 2 3 4]), H = hadamard(4), A = hankel(C(1,:))
H = hilb(5), I = invhilb(5)
M = magic(5), P = pascal(5)
R = rosser(),T = toeplitz(C(1,:)), W = wilkinson(5)
A = rand(3,4)
[X, Y] = meshgrid(x, y)
```

Для створення масивів розмірності більше 2 можна використовувати:

1. функції `zeros, ones, rand` з відповідним набором параметрів. Наприклад, `A=rand(3,2,4)`.
2. процедуру "нарощування" існуючого масиву відповідними елементами. Наприклад, для матриці `A=[1,2;3,4]` після виконання команди `A(1,1,2)=0` отримаємо матрицю `A` розмірності 2×2×2 (причому матриця `A(:,:,2)` - нульова), а після виконання команди `A(:,:,3)=[0,20;70,2]` - матрицю `A` розмірності 2×2×3. Допустима і зворотна дія, наприклад, після команди `A(:,:,2)=[]` ми видалили нульову матрицю і отримали матрицю `A` розмірності 2×2×2.
3. функцію `cat(d,A,B)` – конкатенація масивів `A,B`, де результуючий масив має розмірність `d`. При цьому `cat(2,A,B)` еквівалентне `[A,B]`, а `cat(1,A,B)` еквівалентне `[A;B]`. Наприклад, після виконання команд

```
A = magic(3),  B = pascal(3),  C = cat(4,A,B)
```

отримаємо матрицю C розмірності 3×3×1×2.

Відлік (нумерація) елементів масиву за розмірностями починається з 1.

Для використання масиву з заданим іменем:

1. вказуємо ім'я масиву `A` в матричних операціях при побудові виразів і при отриманні матриці-результату, наприклад, `A=[-pi:pi/20:pi]; Y=sin(A);`
2. вказуємо *A(індексний_вираз)* – для *доступу* до елемента(ів) матриці `A` при використанні в побудованому виразі або для *зміни значення* елемента (значень елементів) `A` при використанні в лівій частині команди присвоєння. Слід

пам’ятати, що відсутність елемента у випадку *доступу*, приводить до виникнення фатальної помилки, а у випадку *зміни значення* – до добудови ("нарощування") масиву `A`.

Уведене тут позначення *індексний_вираз* має наступні смислові варіанти:

- *один* скалярний вираз/змінна/константа зі значенням `k` – для вказівки фіксованого (конкретного) `k`-го елементу одновимірного масиву `A`, наприклад, `A(1)`, `A(k)`, `A(k+1)`;
- *декілька* скалярних виразів/змінних/констант, які розділені символом "`,`" (*список індексів*) зі значеннями `k1,k2,…,km` – для вказівки фіксованого (конкретного) елементу `m`-вимірного масиву `A`, наприклад, `A(1,2)`, `A(k,j)`, `A(k+1,j-1,m)`;
- в *списку індексів* використовується *знак операції* "`:`" – для вказівки усіх елементів цього рядка/стовпця/матриці/… масиву `A` (перші два варіанти для двовимірного, а наступні – для багатовимірних). Наприклад, якщо `A` матриця розмірності 10×20, то `A(:,2)` – одновимірний вектор-стовпець із елементів другого стовпця матриці `A`, а `A(5,:)` – одновимірний вектор-рядок із елементів п’ятого рядка матриці `A`;
- *end* (спеціальний символ) – для вказівки останнього елементу в даній вимірності масиву `A`. Може комбінуватись зі знаком операції "`:`". Наприклад, для одновимірного масиву: `A(end)`, `A(end-1)`, `A(k:end)`; для двовимірного масиву: `A(1,end)`, `A(:,end-2)`, `A(end,k)`, `A(end,end)`, `A(end)`;
- *один знак операції* "`:`" – для вказівки всіх елементів масиву `A` (якщо `A` двовимірний, то перебір виконується по стовпцях, а у випадку тривимірності, то спочатку по стовпцях матриці `A(:,:,1)`, потім по стовпцях матриці `A(:,:,2)` і т. д.). Наприклад, порівняйте для матриці `A=[1,2;3,4]` вирази `A(:)` і `A(:,:)`.
- *масив зі значеннями цілого типу (індексний масив)* – для вказівки тих елементів масиву `A`, індекси яких дорівнюють значенню елементів індексного масиву. Наприклад, для матриці `A=rand(5,6)` команди `A([3:8])=0; A(2:3,4:6)=0;` обнуляють наступні елементи: `A(3,1)`, `A(4,1)`, `A(5,1)`, `A(1,2)`, `A(2,2)`, `A(3,2)`, `A(2,4)`, `A(2,5)`, `A(2,6)`, `A(3,4)`, `A(3,5)`, `A(3,6)`;
- *масив зі значеннями логічного типу (масив логічного індексування)* – для вказівки всіх елементів масиву `A`, індекси яких відповідають індексам масиву логічного індексування (МЛІ) і відповідний елемент МЛІ є *Істина*. Приклади: для отриманої в попередньому прикладі матриці, команда `b=A(0<A&A<0.5)` утворить вектор-стовпець `b` з елементів матриці `A`, для яких виконується умова `0<A&A<0.5`, а після виконання команди `B=A(A(:,1)>0, A(5,:)>0)` отримаємо матрицю `B` розмірності 2×5 (проаналізуйте утворення її елементів!).

Вбудовані функції MATLAB для роботи з матрицями

Розглянемо вбудовані функції системи MATLAB, які використовуються для знаходження числових характеристик матриць, виконання операцій над матрицями і векторами, розв’язування основних задач лінійної алгебри, представлення матриць в певних формах і вирішення питань проблеми власних значень. Імена більшості з цих

функцій співпадають з математичними позначеннями, тому легко запам'ятовуються.

Для розуміння змісту деяких вбудованих функцій MATLAB цієї групи, доцільно ознайомитись з певним теоретичним матеріалом лінійної алгебри і методів обчислень, викладеним в Додатку до цієї теми.

Визначник квадратної матриці знаходиться функцією `det(A)`.

Норма вектора /матриці знаходиться функцією `norm(A,p)`, де необов'язковий параметр `p` може приймати такі допустимі значення: `1,2,inf,'fro'`, де `'fro'` – фробеніусова норма. Якщо `p` опускаємо, то `p=2`. Нижче представлено таблицю обчислення узгоджених норм для векторів/матриць через інші вбудовані функції MATLAB, що дає чітке уявлення про визначення цієї вбудованої функції

Для матриць:	**Для векторів:**
`norm(A) = norm(A,2) = max(svd(A))`	`norm(A) = norm(A,2) = sqrt(sum(A.^2))`
`norm(A,1) = max(sum(abs(A)))`	`norm(A,1) = sum(abs(A))`
`norm(A,inf) = max(sum(abs(A')))`	`norm(A,inf) = max(abs(A))`
`norm(A,'fro') = sqrt(sum(diag(A'*A)))`	`norm(A,-inf) = min(abs(A))`

Число обумовленості квадратної матриці знаходиться функцією `cond(A,p)`, де необов'язковий параметр `p` приймає такі самі значення, як і у функції `norm`.. Для оцінки числа обумовленості використовується функція `rcond(A)`, яка обчислює значення `k1`, обернене значенню `cond(A,1)`. Якщо матриця `A` добре обумовлена, то значення `k1` близьке до одиниці, для погано обумовленої матриці значення `k1` близьке до нуля.

Ранг матриці знаходиться функцією `rank(A)`.

Слід квадратної матриці знаходиться функцією `trace(A)`.

Ортонормований базис матриці знаходиться функцією `orth(A)`. Виклик `H=orth(A)` дає матрицю `H`, для якої (математично) виконується рівність `H'*H = E`. Тут і надалі термін "математично" означає, що всі операції виконуються точно і похибки заокруглень відсутні. Тому, якщо необхідно виконати перевірку деякої математичної рівності `Л=П`, то записуємо її, наприклад, у вигляді, `abs(Л-П)<1e-P`, де `P` число із діапазону ≈[12,16].

Нуль-простір (ядро) для матриці знаходиться функцією `null(A)`. Виклик `Z=null(A)` утворює матрицю `Z`, для якої (математично) виконуються рівності `Z'*Z=E, A*Z=0`.

Скалярний добуток векторів знаходиться функцією `dot(X,Y)`, де `X,Y` – вектори однакової розмірності.

Векторний добуток векторів знаходиться функцією `cross(X,Y)`, де `X,Y` – вектори розмірності 3.

Тензорний добуток векторів (добуток Кронекера) знаходиться функцією `kron(X,Y)`, де `X,Y` – прямокутні матриці відповідної розмірності.

Обернена матриця для квадратної матриці знаходиться функцією `inv(A)`. Виклик `B=inv(A)` дає матрицю `B`, для якої (математично) виконується рівність `B*A = A*B = E`.

Псевдообернена матриця (по Муру-Пенроузу) – знаходиться функцією `pinv(A)`. Виклик `B=pinv(A)` для квадратної (прямокутної) матриці `A` дає матрицю `B`, для якої (математично) виконуються рівності `A*B*A = A, B*A*B = B`. Тут `A*B`, `B*A` – ермітові матриці.

LU–розклад матриці `A` виконується за допомогою виклику `[L,U,P]=lu(A)` для квадратної матриці `A`. Тут `L` – нижня трикутна матриця з одиницями на головній діагоналі, `U` – верхня трикутна матриця, `P` – матриця перестановок. Для наявних матриць (математично) виконується рівність `P*A = L*U`. *Зауваження*: виклик $[L_1,U_1]=lu(A)$ знаходить $L_1 \neq L$ і вона не нижня трикутна, але $A = L_1 * U_1$.

QR–розклад матриці `A` виконується за допомогою виклику `[Q,R,P]=qr(A)` або $[Q_1,R_1]=qr(A)$ для квадратної матриці `A`. Тут `Q` – унітарна матриця (`Q*Q'=E`), `R` – верхня трикутна матриця, `P` – матриця перестановок. Для наявних матриць (матема-тично) виконується рівність `A*P=Q*R` або $Q_1 * R_1 = A$. (*зауваження*: $Q_1 \neq Q$, $R_1 \neq R$).

Розклад Холецького. Якщо матриця `A` *симетрична* (елементи дійсні) або *ермітова* (елементи комплексні), то можна застосувати розклад Холецького за допомогою виклику функції `[R,p]=chol(A)`. Якщо `A` – додатно означена матриця, то `p=0`, а `R` – верхня трикутна і виконується рівність `R'*R=A`, в іншому випадку `p>0` і `R` – верхня трикутна матриця порядку `q=p-1` така, що `R'*R=A(1:q,1:q)`.

Сингулярний розклад квадратної матриці `A` виконується за допомогою виклику `[U,D,V]=svd(A)` (SVD – Singular value decomposition). Тут `D=diag(d)` – діагональ-на матриця, де `d` – вектор невід'ємних чисел ($d_1 \geq d_2 \geq \ldots \geq d_n$); `U`, `V` – унітарні матриці. Функція `svd` досить часто використовується в інших вбудованих функціях цієї групи функцій MATLAB.

Приведення матриці до (верхньої) форми Хессенберга виконується функцією `[P,H]=hess(A)`. Якщо матриця `A` симетрична або ермітова, то матриця Хессенберга `H` буде тридіагональною. Необов'язковий параметр `P` – унітарна матриця перетворень. Для наявних матриць (математично) виконується рівність `A=P*H*P`, `P'*P=E`.

Жорданова форма матриці знаходиться функцією `jordan(A)` (*зауваження*: відсутня в `Octave` [16]).

Приведення матриці до ступінчастого вигляду виконується функцією `[R,rb]=rref(A)`. Застосовується метод Гаусса з вибором провідного елемента по стовпцю (у випадку невиродженої квадратної матриці `R(:,1:end-1)=E`). Наприклад, послідовність команд

```
A = [1,1,1;1,2,2;1,2,3];b = [1;2;3];R = rref([A,b]),X =
                        R(:,4)
```

знаходить розв'язок наступної СЛАР

$$\begin{cases} 1x_1 + 1x_2 + 1x_3 = 1, \\ 1x_1 + 2x_2 + 2x_3 = 2, \\ 1x_1 + 2x_2 + 3x_3 = 3. \end{cases}$$

Власні значення і ***вектори*** квадратної матриці знаходяться функцією `eig(A)`. Виклик `[H,L]=eig(A)` визначає матрицю `H` власних векторів і діагональну матрицю `L` власних чисел, для яких (математично) виконується рівність `A*H = H*L`. Для матриць `H,L` виконуються співвідношення:

`n=length(A); norm(H(:,k))=1`, `k=1,…,n;`
`abs(L(1,1))≥…≥abs(L(n,n))`

Характеристичний поліном квадратної матриці знаходиться функцією `poly(A)`. Виклик `P=poly(A)` дає вектор-рядок `P` (стандартне зображення полінома в MATLAB), який і представляє характеристичний поліном виду

$$P = p_1\lambda^n + p_2\lambda^{n-1} + ... + p_n\lambda + p_{n+1}.$$

Наступні М-файли ілюструють деякі варіанти роботи з матрицями.

Приклад 6. Реалізувати метод відбиття (*QR*–розклад матриці, див. Додаток)

```
%%pl3_2.m
% Якщо для матриці A всі головні мінори <>0,
% то методом відбиття можна отримати розклад Q*R=A,
% де Q - ортогональна (QQ'=E), а R - верхня трикутна матриці
% err - код закінчення: при err=0 - розклад знайдено
close all, clear all, clc
A = rand(6);
n = length(A); R = A; Q = eye(n); err = 0;
for k = 1 : n-1
   if R(k,k) == 0
      err = 1; break
   end
   y = R(k:n, k); km = k-1; m = n-km;
   if any(y(2:m))
      al = norm(y); y(1) = y(1)-al;
      y = y./norm(y); U = eye(m)-2.*y*y';
      U = [eye(km), zeros(km,m); zeros(m,km), U];
      Q = U*Q; R = U*R; R(k+1:n,k) = 0;
   end
end
if ~err
   Q = Q', R,  abs(Q*R - A) < 1e-14  % перевірка
   [Q, R] = qr(A)  % використання вбудованої функції
end
```

Приклад 7. Різні варіанти розв'язання СЛАР

```
%%pl3_3.m
% Приклади розв'язання СЛАР Ax=b
close all, clear all, clc
xx = [1; -2; 3]; % відомий розв'язок (модельна задача)
n = length(xx);
% формування вхідних данних: матриця A=A' & A>0
A=[ 7,  1, 1;...
    1, 10, 2;...
    1,  2, 4]
b = A * xx    %  вектор правої частини СЛАР
% Варіанти
x1 = A \ b;               % 1-й : матричне ліве ділення
[Q, R] = qr(A);
x2 = R \ (Q' * b);        % 2-й : QR-розклад
[L, U, P] = lu(A);
x3 = U \ (L \ (P * b));   % 3-й : LU-розклад
R = rref([A, b]);
x4 = R(:, n+1);           % 4-й : приведення до ступінчастої форми
x5 = NaN.*ones(n,1);
if all(all(A == A'))      % 5-й : розклад Холецького
   [R, p] = chol(A);
   if p == 0
      x5 = R \ (R' \ b);
   end
end
opts.SYM = true; opts.POSDEF = true;
```

```
x6 = linsolve(A, b, opts);% 6-й : використання вирішувача – солвера
%       для СЛАР
x7 = inv(A) * b;          % 7-й : використання оберненої матрицю
% візуальне порівняння отриманих розв'язків :
[x1, x2, x3, x4, x5, x6, x7, xx]
```

Приклад 8. Знаходження власних значень і власних векторів матриць

```
%%pl3_4.m
close all, clear all, clc
P = pascal(5) % побудова матриці Паскаля
m = length(P) - 1;
p = poly(P)   % характеристичний поліном матриці P
% використання конструктора [], операцій
% горизонтального і вертикального об'єднання масивів
% для побудови запису матриці P у формі Фробеніуса
F = [-p(2:end); [eye(m), zeros(m,1)]]
F = compan(p) % за допомогою вбудованої функції MATLAB
r = roots(p)  % власні числа матриці Паскаля
% функція jordan(M) для Octave не визначена !
%J = jordan(P) % Жорданова форма для матриці Паскаля
% власні вектори і власні числа матриці Паскаля
[H1, L1] = eig(P);  %  для P
H1, L1        %  вивести на екран
l1 = sort(diag(L1), 'descend');  % по спаданню значень
[H2, L2] = eig(F);  %  для P в формі Фробеніуса
H2, L2        %  вивести на екран
l2 = sort(diag(L2), 'descend');
[r, l1, l2]   %  візуальне порівняння
l1 == l2      %  порівняння логічне
% порівняння з урахуванням наближених обчислень
abs(l1 - l2) < 1e-14
```

Враховуючи певну специфіку використання *функцій від матриць* (не слід плутати із застосуванням функцій до кожного елемента матриці), наведемо тільки їх виклик або назву:

- матрична експонента (e^A) (`expm(A), expm1, expm2, expm3`);
- матричний логарифм (`logm(A)`);
- матричний квадратний корінь ($A^{1/2}$) (`sqrtm(A)`);
- матричний поліном (`polyvalm(p,A)`);
- обчислення довільної матричної функції `f` (`funm(A,@f)`).

ДОДАТОК

п.1. *LU*-розклад матриці *A* для розв'язання СЛАР *Ax=b*

Теорема 3.1. Якщо $\det(A)\neq 0$, то існує матриця *перестановок* P, така, що $P\cdot A$ має відмінні від 0 *кутові мінори* :

$$\Delta_1 = a_{1,1},\ \ \Delta_2 = \begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{vmatrix},\ \ ...,\ \ \Delta_n = \det(A),\ \ \Delta_i \neq 0,\ \ i=\overline{1,n}. \qquad (3.1)$$

Елементарна матриця перестановок визначається як: $P_{k,j} = (E)_{j\leftrightarrow k\ \text{rows}}$.

Теорема 3.2. Нехай всі кутові мінори матриці A відмінні від 0, тоді матрицю A однозначно можна представити у вигляді добутку двох трикутних матриць:

$$A = L \cdot U, \quad L = \begin{pmatrix} l_{1,1} & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ l_{n,1} & l_{n,2} & \cdots & l_{n,n} \end{pmatrix}, \quad U = \begin{pmatrix} 1 & u_{1,2} & \cdots & u_{1,n} \\ 0 & 1 & \cdots & u_{2,n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}. \tag{3.2}$$

Таким чином, розв'язування СЛАР $A\vec{x} = \vec{b}$ можна реалізувати у вигляді послідовності наступних кроків:

$$A \ \Rightarrow \ [P,\ L,\ U], \tag{3.3}$$

$$U \cdot \vec{x} = \vec{f}, \quad \vec{f} = L^{-1} P \cdot \vec{b}\,. \tag{3.4}$$

Знаходження розкладу (3.3) називається *прямим ходом*, а (3.4) – *зворотним ходом методу Гаусса*. З точки зору ефективності обчислень у алгоритм (3.3), (3.4) потрібно додати процедуру зменшення впливу *похибок заокруглень*.

Для реалізації (3.3)–(3.4) використовуються:

А) Алгоритм без явного знаходження матриць *P*, *L*.

Для k – стовпця матриці $A^{(k-1)}$, $(A^{(0)} = A)$, де $k = 1,2,...,n-1$,

1) знаходимо j_k – номер рядка, в якому знаходиться *провідний елемент*
$\left|a_{j_k,k}^{(k-1)}\right| = \max\limits_{k \le i \le n}\left(\left|a_{i,k}^{(k-1)}\right|\right) \,\&\, a_{j_k,k}^{(k-1)} \neq 0;$
2) якщо $j_k \neq k$, то виконуємо перестановку рядків j_k, k множенням на відповідну елементарну матрицю перестановок
$P_k = P_{k,j_k}, \quad P_k \cdot A^{(k-1)} = P_k \cdot \vec{b}^{(k-1)};$
3) виключаємо x_k з рядків $i = \overline{k+1, n}$ множенням на відповідну елементарну трикутну матрицю
$L_k \cdot P_k \cdot A^{(k-1)} = L_k \cdot P_k \cdot \vec{b}^{(k-1)};$
де

$$L_k = \begin{pmatrix} 1 & 0 & \cdots & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots & \cdots & \cdots & \vdots \\ 0 & \cdots & l_{k,k} & 0 & \cdots & \cdots & 0 \\ 0 & \cdots & l_{k+1,k} & 1 & \ddots & \cdots & 0 \\ 0 & \cdots & l_{k+1,k} & 0 & 1 & \ddots & 0 \\ 0 & \cdots & \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & \ldots & l_{n,k} & 0 & \cdots & \cdots & 1 \end{pmatrix}, \quad l_{i,k} = \begin{cases} \dfrac{1}{a_{k,k}^{(k-1)}}, i = k, \\ \dfrac{-a_{i,k}^{(k-1))}}{a_{k,k}^{(k-1)}}, i = \overline{k+1, n}. \end{cases}$$

У результаті об'єднання проведених обчислень (прямий хід методу Гаусса), отримуємо (3.4), де

$$U = L_{n-1} \cdot P_{n-1}(\cdot \ldots (L_2 \cdot P_2 (L_1 \cdot P_1 \cdot A)) \ldots),$$

$$\vec{f} = \vec{b}^{(n-1)} \equiv P_{n-1} \cdot \ldots \cdot P_2 \cdot P_1 \cdot \vec{b}.$$

Знаходження компонент x_k (зворотний хід методу Гаусса) здійснюється за формулою: $x_k = f_k - \sum\limits_{i=k+1}^{n} u_{k,i} x_i, \quad k = n, n-1, ..., 1.$

Б) Алгоритми з явним знаходження матриць *P*, *L*, *U* – *методи факторизації*.

1) Знаходимо розклад матриці $A \ \Rightarrow \ [P,\ L,\ U]$;
2) Отримуємо дві СЛАР з трикутними матрицями

$$(L \cdot U) \cdot \vec{x} = \vec{f}, \quad \vec{f} = P \cdot \vec{b}, \quad \begin{cases} L \cdot \vec{y} = \vec{f}, \\ U \cdot \vec{x} = \vec{y}. \end{cases} \tag{3.5}$$

Як методи факторизації, так і метод Гаусса, мають певні варіанти реалізації, пов'язані зі специфікою матриці СЛАР :

a) якщо в (3.5) $P = E$, то такий варіант має назву *методу Холецького*;

b) якщо матриця A є ермітовою $A = A^*$ або симетричною $A = A^{'}$, то такий варіант має назву *методу квадратного кореня*;

c) якщо A – *тридіагональна* матриця

$$\begin{cases} a_i x_{i-1} - c_i x_i + b_i x_{i+1} = -f_i, \\ i = \overline{1,n}, \quad a_1 = b_n = 0, \end{cases} \tag{3.6}$$

для якої виконуються умови

$$\begin{cases} |c_i| \geq |a_i| + |b_i|, \quad i = \overline{1,n}; \\ a_i \neq 0, \quad i = \overline{2,n}; \\ b_i \neq 0, \quad i = \overline{1,n-1}, \end{cases} \tag{3.7}$$

то такий варіант методу Гаусса має назву *методу монотонної правої прогонки*, і реалізація прямого ходу відбувається за рекурентними формулами:

$$\alpha_1 = \beta_1 = 0; \;\; z = c_i - a_i \alpha_i, \;\; \alpha_{i+1} = b_i / z, \;\; \beta_{i+1} = (f_i + a_i \beta_i)/z, \;\; i = \overline{1,n}, \tag{3.8}$$

а зворотного ходу – за рекурентними формулами:

$$x_n = \beta_{n+1}; \;\; x_{i-1} = \alpha_i x_i + \beta_i, \;\; i = n, n-1, ..., 2. \tag{3.9}$$

При виконанні умов (3.7) прогонка (3.8),(3.9) є *стійкою* (виконується умова $|\alpha_i| \leq 1, \;\; i = \overline{2,n}$).

Для випадку (3.6) є варіант *немонотонної прогонки* (пошук оптимального елемента по рядку), аналогічно існують ці варіанти для п'ятидіагональної матриці. Розглядаються також реалізації прогонки, які носять назви: *потокова*, *зустрічна*, *циклічна*, *для складних систем*, *матрична* [12, 20, 21].

п.2. ***QR*-розклад матриці *A* для розв'язання СЛАР *Ax=b***

Лема. 3.1. Довільна дійсна квадратна матриця може бути представлена у вигляді добутку *ортогональної* і *правої трикутної* матриці:

$$A = Q * R \equiv U^{\mathrm{T}} * R, \quad U = U_{n-1} * \ldots * U_2 * U_1; \qquad (U_i^{\mathrm{T}} * U_i = U_i * U_i^{\mathrm{T}} = E). \tag{3.10}$$

Метод послідовної побудови матриці U – *метод відбиття*. Розглянемо алгоритм цього методу (спрощений варіант, при умові, що всі кутові мінори не дорівнюють нулеві). Маємо реалізацію :

I. *1-й крок*. Матриця $A = A_{n,n} = A^{(0)}$. Вибираємо вектор-стовпець $\vec{y} = \left(a_{1,1}, a_{2,1}, ..., a_{n,1}\right)^{\mathrm{T}}$.

а) якщо $a_{2,1} = a_{3,1} = ... = a_{n,1} = 0$, то $A^{(1)} = A^{(0)}, \quad U_1 = E$;

б) якщо $a_{2,1}, a_{3,1}, ..., a_{n,1} \neq 0$, то $A^{(1)} = U_1 * A^{(0)}, \quad U_1 = E - 2\vec{\omega} * \vec{\omega}^{\mathrm{T}}$, де

$$\vec{\omega}^{\mathrm{T}} * \vec{\omega} = (\vec{\omega}, \vec{\omega}) = 1, \quad \vec{\omega} = \frac{\vec{y} - \alpha \vec{e}_1}{\left\| \vec{y} - \alpha \vec{e}_1 \right\|}, \quad \alpha = \|\vec{y}\| = \sqrt{(\vec{y}, \vec{y})}; \tag{3.11}$$

II. Виконали *l-1 -й крок*;

III. *l-й крок*. Для матриці розмірності *n-l*+1 $\begin{pmatrix} a_{l,l}^{(l-1)} & \ldots & a_{l,n}^{(l-1)} \\ \vdots & \ddots & \vdots \\ a_{n,l}^{(l-1)} & \ldots & a_{n,n}^{(l-1)} \end{pmatrix}$ вибираємо вектор-стовпець $\vec{y} = \left(a_{l,l}^{(l-1)}, a_{l+1,l}^{(l-1)}, ..., a_{n,l}^{(l-1)}\right)^{\mathrm{T}}$ і виконуємо дії, аналогічні 1-му кроку:

а) якщо $a_{l+1,l}^{(l-1)} = a_{l+2,l}^{(l-1)} = ... = a_{n,l}^{(l-1)} = 0$, то $A^{(l)} = A^{(l-1)}, \quad U_l = E$;

б) якщо $a_{l+1,l}^{(l-1)}, a_{l+2,l}^{(l-1)}, ..., a_{n,l}^{(l-1)} \neq 0$, то $A^{(l)} = U_l * A^{(l-1)}$, а блочна матриця U_l визначається наступним чином:

$$U_l = \begin{pmatrix} E_{l-1} & 0 \\ 0 & U_{(n-l+1)} \end{pmatrix}, \quad U_{(n-l+1)} = E_{(n-l+1)} - 2\vec{\omega}_{(n-l+1)} * \vec{\omega}^{\mathrm{T}}{}_{(n-l+1)},$$

де для обчислення вектора $\vec{\omega}_{(n-l+1)}$ використовуємо формули (3.11), але з урахуванням розмірності n-l+1.

Після виконання n-1 кроків отримаємо верхню трикутну матрицю R:

$$R = A^{(n-1)} = ... = U_{n-1} * U_{n-2} * .... * U_1 * A = U * A, \quad U^{\mathrm{T}} * R = A.$$

QR-розклад використовується як для розв'язання СЛАР, так і для побудови ітераційних процесів вирішення повної проблеми власних значень.

п.3. Оцінка похибки розв'язування СЛАР. Число *cond(A)*

Припустимо, що похибки, які вносяться у вхідні дані, малі.

Означення 3.1. Під *обумовленістю* обчислювальної задачі розуміють чуттєвість (залежність) її розв'язку до вхідних даних, а коефіцієнт, який зв'язує ці похибки, називають *мірою обумовленості задачі*.

Визначимо його для СЛАР :

$$A\vec{x} = \vec{b}, \quad \det(A) \neq 0. \tag{3.12}$$

Для (3.12) можна отримати

$$\vec{x} = A^{-1} * \vec{b}. \tag{3.13}$$

Вважаючи A^{-1}, $\vec{b}$ змінними, формально продиференцюємо (3.13) :

$$\begin{aligned} d\vec{x} = dA^{-1} * \vec{b} + A^{-1} * d\vec{b} = \left[A * A^{-1} = E,\ dA * A^{-1} + A * dA^{-1} = 0,\ dA^{-1} = -A^{-1} * dA * A^{-1}\right] = \\ = -A^{-1} * \left(dA * \left(A^{-1} * \vec{b}\right) - d\vec{b}\right) = -A^{-1} * \left(dA * \vec{x} - d\vec{b}\right). \end{aligned} \tag{3.14}$$

В останній рівності перейдемо до норм, використаємо відповідні нерівності, в результаті чого отримаємо:

$$\|d\vec{x}\| \leq \left\|A^{-1}\right\| \cdot \left(\|dA\| \cdot \|\vec{x}\| + \|d\vec{b}\|\right). \tag{3.15}$$

Враховуючи співвідношення

$$\vec{b} = A \cdot \vec{x}, \quad \|\vec{b}\| \leq \|A\| \cdot \|\vec{x}\|, \quad \|\vec{x}\| \geq \frac{\|\vec{b}\|}{\|A\|},$$

і вводячи відносні похибки вхідних даних і розв'язку

$$\varepsilon_A = \frac{\|dA\|}{\|A\|}, \quad \varepsilon_b = \frac{\|d\vec{b}\|}{\|\vec{b}\|}, \quad \varepsilon_x = \frac{\|d\vec{x}\|}{\|\vec{x}\|},$$

отримаємо:

$$\begin{aligned} \frac{\|d\vec{x}\|}{\|\vec{x}\|} \leq \left\|A^{-1}\right\| \cdot \left(\|dA\| + \frac{\|d\vec{b}\|}{\|\vec{x}\|}\right) \leq \left\|A^{-1}\right\| \cdot \left(\|dA\| + \frac{\|d\vec{b}\|}{\|\vec{b}\|} \cdot \|A\|\right) \leq \left\|A^{-1}\right\| \cdot \|A\| \cdot \left(\frac{\|dA\|}{\|A\|} + \frac{\|d\vec{b}\|}{\|\vec{b}\|}\right), \\ \varepsilon_x \leq cond(A) \cdot (\varepsilon_A + \varepsilon_b), \end{aligned} \tag{3.16}$$

де $\mu_A = cond(A) = \|A\| \cdot \left\|A^{-1}\right\|$ – *число обумовленості* матриці *A*.

Зауваження 1. Матриці з $\mu_A >> 1$ – *погано обумовлені.*

Зауваження 2. Детальніші дослідження дають наступний вираз для (3.16):

$$\varepsilon_x \le \frac{\mu_A}{1-\mu_A\cdot\varepsilon_A}\cdot\left(\varepsilon_A+\varepsilon_b\right).$$

Зауваження 3. Для μ_A маємо наступні співвідношення:

1^o $\mu_A \ge 1$;

2^o $\mu_A \ge \dfrac{\max\limits_i\left|\lambda_i(A)\right|}{\min\limits_i\left|\lambda_i(A)\right|}$;

3^o $\mu(A*B)\le\mu(A)\cdot\mu(B)$.

Зауваження 4. Простою ознакою поганої обумовленості матриці є велика різниця норм вектор-стовпців / вектор-рядків. Загальне правило дій у цьому випадку таке: за допомогою множення стовпців / рядків на числа (степені 2) урівноважуємо ці норми, масштабуємо невідомі / перетворюємо праві частини. Після цього застосовуємо методи типу Гаусса. Якщо цього не достатньо, потрібно використовувати *регуляризацію*.

п.4. Алгебраїчна проблема власних значень

Означення 3.2. Нехай для матриці $A=\left(a_{i,j}\right)_{i,j=1}^{n}$ існує нетривіальний розв'язок однорідної системи :

$$A*\vec{h}=\lambda\cdot\vec{h}, \tag{3.17}$$

тоді числа λ_i називаються *власними числами* матриці А, а відповідні їм розв'язки $\vec{h}_i$ – *власними векторами*.

Означення 3.3. Задачу відшукання повного набору $\left\{\lambda_i(A),\ \vec{h}_i\right\}$ називають *повною проблемою власних значень* (ППВЗ) для матриці А. Відповідно, визначення частини спектру $\lambda_i(A)$ (найчастіше максимального / мінімального за модулем) і можливо відповідних власних функцій $\vec{h}_i$ називають *частковою проблемою власних значень*.

Часткова проблема власних значень. Розглянемо *степеневий* ітераційний метод/процес (СІП), який базується на відношенні Релея і для застосування якого необхідно, щоб матриця *A* мала *n* лінійно незалежних векторів.

Алгоритм. 1. Задаємо довільний вектор, відмінний від нуля ($\forall\ \vec{x}^{(0)}\ne 0$).

2. В циклі по k=0,1,…, *kmax* повторюємо наступні дії :

а) $\vec{Y}=A*\vec{x}^{(k)},\quad y=\sqrt{(\vec{Y},\vec{Y})},\quad \lambda^{(k)}=y,\quad \vec{x}^{(k+1)}=\vec{Y}/y$;

б) після $k>m$ ітерацій перевіряємо виконання умови закінчення СІП $\left|\lambda^{(k)}-\lambda^{(k-1)}\right|<\varepsilon$.

Тут позначено:

kmax – максимальна кількість ітерацій (для запобігання зациклюванню СІП),

m – кількість ітерацій без перевірки різниці наближень до максимального за модулем власного числа (0<*m*<*kmax*), $0<\varepsilon<1$ – бажана точність.

Результатом СІП є власне значення $\max\limits_i\left|\lambda_i(A)\right|=\lambda^{(k)}$ і відповідний йому власний вектор $\vec{h}=\vec{x}^{(k+1)}$. Збіжність СІП лінійна із знаменником $q=\left|\lambda_2/\lambda_1\right|$. Збіжність СІП відсутня при:

а) $\lambda_1\ne\lambda_2$ & $\left|\lambda_1\right|=\left|\lambda_2\right|$;

б) для власного значення λ_1. жорданова клітина має розмірність більше 1.

Для варіанту $A=A'$ СІП має назву методу *скалярних добутків* і в наведеному

вище алгоритмі у пункті 2 змінюється дія а) на наступну:

$$\vec{Y} = \frac{\vec{x}^{(k)}}{\left\|\vec{x}^{(k)}\right\|}, \quad \vec{x}^{(k+1)} = A * \vec{Y}, \quad \lambda^{(k)} = \left(\vec{Y}, \vec{x}^{(k+1)}\right),$$

що забезпечує квадратичну збіжність.

Для знаходження $\min\limits_i |\lambda_i(A)|$ використовується варіант СІП з оберненою матрицею $A * \vec{Y} = \vec{x}^{(k)}, \quad y = \sqrt{(\vec{Y}, \vec{Y})}, \quad \beta^{(k)} = y, \quad \vec{x}^{(k+1)} = \vec{Y}/y$ і $\min\limits_i |\lambda_i(A)| = \frac{1}{\beta^{(k)}}$.

Повна проблема власних значень. Існують досконалі алгоритми і програми, що базуються на різних модифікаціях *QR-алгоритмів* за схемою

$$A^{(0)} = A, \quad A^{(k)} = Q^{(k+1)} * R^{(k+1)}, \quad A^{(k+1)} = R^{(k+1)} * Q^{(k+1)}, \quad \lim_{i\to\infty}\left(A^{(i)}\right) = \Lambda,$$

де Λ – *права канонічна форма Жордана*. Для варіанту $A = A^{\mathrm{T}}$ або $A = A^*$ алгоритм має назву *методу обертань* (*Якобі*).

п.5. Сингулярний розклад матриці А

Сингулярний розклад матриці A означає, що потрібно знайти ненульові вектори $\vec{v}$, $\vec{u}$ і невід'ємні числа d, для яких виконується наступна системи рівнянь:

$$\begin{cases} A * \vec{v} = d \cdot \vec{u}, \\ A^{\mathrm{T}} * \vec{u} = d \cdot \vec{v}. \end{cases} \tag{3.18}$$

Вектори $\vec{v}$, $\vec{u}$ називаються *правими і лівими сингулярними* векторами, числа d – *сингулярними* числами.

Допустимо, що вектори $\vec{v}$ утворюють стовпці унітарної матриці V, а $\vec{u}$ утворюють стовпці унітарної матриці U, числа d – діагональну матрицю D. Тоді рівняння (3.18) еквівалентні двом матричним рівнянням:

$$\begin{cases} A * V = U * D, \\ A^{\mathrm{T}} * U = V * D \end{cases}$$

або одному матричному розкладу:

$$A = U * D * V^{\mathrm{T}}, \tag{3.19}$$

який і називається *сингулярним розкладом матриці* A.

Це представлення для квадратної матриці можна знайти і з інших міркувань.

Теорема 3.3. (Полярний розклад Келі) Довільну квадратну матрицю A можна представити у вигляді добутку симетричної S і ортогональної матриці W:

$$A = S * W; \quad S = S^{\mathrm{T}}, \quad W * W^{\mathrm{T}} = E. \tag{3.20}$$

У свою чергу симетрична матриця S може бути приведена до діагонального виду за допомогою деякого ортогонального перетворення:

$$S = U * C * U'; \quad C = diag(c_i), c_i = \pm d_i, d_i \geq 0; \quad C = D * T; \quad T = diag(t_i), t_i = \pm 1, \; T * T' = E. \tag{3.21}$$

Підставимо (3.21) у (3.20) і отримаємо :

$$A = U * D * T * U^{\mathrm{T}} * W = U * D * V^{\mathrm{T}}, \quad V^{\mathrm{T}} = T * U^{\mathrm{T}} * W.$$

п.6. Ітераційні методи розв'язання СЛАР

При розв'язанні СЛАР великої розмірності ($n >\sim 100$) прямі методи стають неефективними. У таких випадках доцільніше застосовувати наближені ітераційні методи [3–5, 13, 21].

Нехай маємо СЛАР (3.12), де A має обернену матрицю A^{-1}. Представимо матрицю A у вигляді $A = A_1 + D + A_2$, де $D = diag\left[a_{11}, \ldots, a_{nn}\right]$, A_1 – нижня трикутна, A_2 –

верхня трикутна матриці.

$$(A_1 + D + A_2)\vec{x} = \vec{f},$$
$$D\vec{x} = -(A_1 + A_2)\vec{x} + \vec{f},$$
$$\vec{x} = -D^{-1}(A_1 + A_2)\vec{x} + D^{-1}\vec{f}. \quad (3.22)$$

Метод Якобі. Схема методу :

$$\vec{x}^{k+1} = -D^{-1}(A_1 + A_2)\vec{x}^{(k)} + D^{-1}\vec{f}. \quad (3.23)$$

Якщо $A = A^{\mathrm{T}}, A > 0$ і $|a_{ii}| > \sum_{j=u\neq i}^{n} |a_{ij}|$, $i = \overline{1,n}$, то ітераційний процес (3.23) збігається.

Метод Зейделя. Схема методу:

$$\vec{x}^{(k+1)} = -D^{-1}A_1\vec{x}^{(k+1)} - D^{-1}A_2\vec{x}^{(k)} + D^{-1}\vec{f}. \quad (3.24)$$

Умова закінчення цих ІП :

$$\left\|\vec{x}^{(k+1)} - \vec{x}^{(k)}\right\|_c < \varepsilon.$$

Канонічна форма однокрокових ітераційних методів має вигляд:

$$B_{(k+1)}\frac{\vec{x}^{(k+1)} - \vec{x}^{(k)}}{\tau_{k+1}} + A\vec{x}^{(k)} = \vec{f}. \quad (3.25)$$

Якщо $B_{(k+1)} = E$, то ітераційний процес (3.25) називають *явним*, інакше – *неявним*. Якщо $B_{(k+1)} = B$, то ітераційний процес називається с*таціонарним.*

Метод простої ітерації. Схема методу

$$\frac{\vec{x}^{(k+1)} - \vec{x}^{(k)}}{\tau} + A\vec{x}^{(k)} = \vec{f}. \quad (3.26)$$

Метод Річардсона. Схема методу

$$\frac{\vec{x}^{(k+1)} - \vec{x}^{(k)}}{\tau_{k+1}} + A\vec{x}^{(k)} = \vec{f}. \quad (3.27)$$

Якщо $A = A^{\mathrm{T}}$, $A > 0$, то відомі алгоритми побудови оптимальних ітераційних параметрів для (3.26), (3.27), які базуються на значеннях границь спектра власних значень матриці СЛАР. Для (3.26) – це *оптимальний лінійний ІП*, а для (3.27) – *метод Річардсона з чебишовським набором параметрів*.

Метод релаксації (узагальнення методу Зейделя). Схема методу

$$(D + \omega \cdot A_1)\frac{\vec{x}^{(k+1)} - \vec{x}^{(k)}}{\omega} + A\vec{x}^{(k)} = \vec{f}. \quad (3.28)$$

При $0 < \omega < 1$ (3.28) називають *методом нижньої релаксації*, при $1 < \omega < 2$ – *методом верхньої релаксації*, при $\omega = 1$ – методом Зейделя.

Якщо $A = A^{\mathrm{T}}$, $A > 0$, то (3.28) збігається при $0 < \omega < 2$.

Запишемо (3.28) у покомпонентній формі:

$$\begin{cases} x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega\dfrac{f_i}{a_{ii}} - \omega\left(\sum\limits_{j=1}^{i-1}\dfrac{a_{ij}}{a_{ii}}x_j^{(k+1)} + \sum\limits_{j=i+1}^{n}\dfrac{a_{ij}}{a_{ii}}x_j^{(k)}\right), i = \overline{1,n}; \\ k = 0,1,\ldots, \left\|\vec{x}^{(k+1)} - \vec{x}^{(k)}\right\|_{\mathrm{C}} < \varepsilon. \end{cases} \quad (3.29)$$

Методи побудовані на основі варіаційних підходів. Нехай в СЛАР (3.12) $A = A^{\mathrm{T}}$, $A > 0$. Розглянемо стаціонарний ітераційний процес у вигляді (3.25) з $B = E$. Позначимо для кроку k нев'язку через

$$\vec{r}^{(k)} = A\vec{x}^{(k)} - \vec{f}, \quad (3.30)$$

тоді (3.25) можна переписати у вигляді

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \tau_{k+1}\vec{r}^{(k)}. \qquad (3.31)$$

Методом мінімальних нев'язок (ММН) називається ітераційний процес (3.31), в якому τ_{k+1} вибирається з умови $\min\left\|\vec{r}^{(k+1)}\right\|$ при заданому $\left\|\vec{r}^{(k)}\right\|$.

Помножимо (3.31) на A і віднімемо з обох частинах рівності $\vec{f}$:

$$A\vec{x}^{(k+1)} - \vec{f} = A\vec{x}^{(k)} - \vec{f} - \tau_{k+1}A\vec{r}^{(k)}, \quad \vec{r}^{(k+1)} = \vec{r}^{(k)} - \tau_{k+1}A\vec{r}^{(k)},$$

$$\left\|\vec{r}^{(k+1)}\right\|^2 = (\vec{r}^{(k)} - \tau_{k+1}A\vec{r}^{(k)}, \ \vec{r}^{(k)} - \tau_{k+1}A\vec{r}^{(k)}) = \left\|\vec{r}^{(k)}\right\|^2 - 2\tau_{k+1}(\vec{r}_{(k)}, A\vec{r}^{(k)}) + \tau_{k+1}^2\left\|A\vec{r}^{(k)}\right\|^2.$$

Звідси бачимо, що $\left\|r^{(k+1)}\right\|$ досягає мінімуму, якщо

$$\tau_{k+1} = \frac{(A\vec{r}^{(k)}, \vec{r}^{(k)})}{\left\|A\vec{r}^{(k)}\right\|^2}. \qquad (3.32)$$

Алгоритм переходу від наближення $\vec{x}^{(k)}$ до наступного $\vec{x}^{(k+1)}$:

$$\vec{r}^{(k)} = A\vec{x}^{(k)} - \vec{f}, \quad \vec{y} = A\vec{r}^{(k)},$$
$$\tau_{k+1} = \frac{(\vec{y}, \vec{r}^{(k)})}{(\vec{y}, \vec{y})}, \quad \vec{x}^{(k+1)} = \vec{x}^{(k)} - \tau_{k+1}\vec{r}^{(k)}. \qquad (3.33)$$

Метод найшвидшого спуску (МНС) відрізняється від ММН тільки формулою обчислення τ_{k+1}. Це значення знаходять з умови $\min\left\|\vec{z}^{(k+1)}\right\|_A$ при заданому $\vec{z}^{(k)} = \vec{x}^{(k)} - \vec{x}^{\text{exac}}$, що еквівалентно ортогональності нев'язок $\vec{r}^{(k)}, \vec{r}^{(k+1)}$. Помножимо (3.31) скалярно на $\vec{r}^{(k)}$, отримаємо

$$0 = (\vec{r}^{(k)}, \vec{r}^{(k)}) - \tau_{k+1}(A\vec{r}^{(k)}, \vec{r}^{(k)}).$$

Звідси

$$\tau_{k+1} = (\vec{r}^{(k)}, \vec{r}^{(k)})/(A\vec{r}^{(k)}, \vec{r}^{(k)}). \qquad (3.34)$$

Швидкість збіжності ММН, МНС так сама, як і у процеса (3.26) з оптимальним параметром τ.

Приклади реалізації деяких ітераційних процесів для розв'язання СЛАР розглянуто в М-файлі *pl3_7.m*.

```
%%pl3_7.m
% Приклади реалізації і використання ітераційних процесів
close all, clear all, clc
n = 10;
% Створюєм деяку матрицю
A = ones(n); x = logical(eye(n)); A(x) = [11:5:56];
b = ones(n,1);   %  створюєм вектор правих частин
ep = 1e-10; mki = 10000;  %  точність і max кількість ітерацій
x = A \ b;   %  розв'язання СЛАР Ax=b (матричне ліве ділення)
% Ітераційним м. Зейделя
xz = b; [xz, kz] = m_zejdel(A, b, xz, ep, mki);
% Ітераційним м. верхньої релаксації
w = 1.0125;    %  параметр релаксації
xr = b; [xr, kr] = m_relaxv(A, b, w, xr, ep, mki);
% Ітераційний метод мінімальних нев'язок
xn = b; [xn, kn] = m_minn(A, b, xn, ep, mki);
disp(' Кількість ітерацій :')
disp(strcat('м.Зейделя         =',int2str(kz)))
disp(strcat('м.верхн.релаксації=',int2str(kr)))
disp(strcat('м.min  нев''язок   =',int2str(kn)))
disp('Розв''язки :')
```

```
S = sprintf('%c%c%c%c','  матр.лів.діл \','   метод Зейделя',...
                       ' верх.релаксації',' мінім. нев''язок');
disp(S)
for k = 1 : n
    S = sprintf('%16.6e%16.6e%16.6e%16.6e',x(k),xz(k),xr(k),xn(k));
    disp(S)
end
[S,er] = sprintf('%s%16.6e%16.6e%16.6e','Абсолют.похибка:',...
           norm(x-xz), norm(x-xr), norm(x-xn));
disp(S)
function [x, k] = m_zejdel(A, f, x, e, mki)
%%M_ZEJDEL   Метод Зейделя.
% Ітераційний метод Зейделя розв'язання СЛАР A*x=f
%  (реалізація матричної-векторної форми)
n = length(A); Df = 1./diag(A);
DA1 = -Df.* tril(A, -1); DA2 = -Df.* triu(A, 1); Df = Df.* f;
xn = x;
for k = 1 : mki
   xn = DA1 * xn + DA2 * x + Df; t = true; i = 0;
   while t & i < n
      i = i + 1; t = abs(xn(i) - x(i)) < e;
   end
   x = xn;
   if t
      break
   end
end
function [x, k] = m_relaxv(A, f, w, x, e, mki)
%%M_RELAXV      Метод релаксації.
% Розв'язання СЛАР A*x=f ітераційним методом релаксації
%    (матрично-векторна форма)
A1 = tril(A, -1); O = diag(diag(A));
O = inv(O + w .* A1);
for k = 1 : mki
   r = w .* (f - A * x);
   r = O * r; x = x + r;
   if norm(r, inf) < e
      break
   end
end
function [x, k] = m_minn(A, f, x, e, mki)
%% M_MINN       ітераційний метод min нев'язок для СЛАР.
% Ітераційний метод min нев'язок розв'язання СЛАР  A*x=f
n = length(A);
for k = 1 : mki
   r = A * x - f; y = A * r;
   t = dot(y, r) ./ dot(y, y); r = t .* r;
   t = true; i = 0;
   while t & i < n
      i = i + 1; t = abs(r(i)) < e;
   end
   x = x - r;
   if t
      break
   end
end
```

ЛЕКЦІЯ № 04

4.1. ПОЛІНОМИ ТА ДІЇ НАД НИМИ

У MATLAB поліном задається і зберігається у вигляді вектора, елементами якого є коефіцієнти полінома. Перелік вбудованих функцій для роботи з поліномами наведено нижче:

Функції для роботи з поліномами	
Назва	**Призначення**
polyval(P,x)	обчислення значення полінома *n*-го степеня при заданому значенні змінної *x*: $y=P_1*x^n+...+P_n*x+P_{n+1}$, ($P_1$<>0 !)
roots(P)	обчислення коренів полінома *P*
conv(P,Q)	обчислення добутку поліномів *P,Q*
[Q,R]=deconv(B,A)	ділення поліномів *B,A*; частка від ділення повертається у вигляді вектора *Q*, залишок – у вигляді вектора *R*, так що виконується співвідношення *B=A*Q+R*
polyder(P)	обчислення похідної від полінома *P*
[Q,R]=polyder(B,A)	похідна від відношення поліномів *B/A* повертається у вигляді відношення поліномів *Q/R*
polyfit(x,Y,n)	апроксимація дискретної функції *Y(x)* поліномом степеня *n*
[R,P,K]=residue(B,A)	розклад *B(x)/A(x)* на прості дроби: $B(x)/A(x) = R_1/(x-P_1) + R_2/(x-P_2)+...+ R_n/(x-P_n)+K(x)$. Якщо корінь P_j має кратність *t*, будуть утворені вирази $R_j/(x-P_j) + R_{j+1}/(x-P_j)^2+...+ R_{j+t-1}/(x-P_j)^t$

4.2. АНАЛІЗ ДАНИХ

Система MATLAB містить значну кількість функцій для використання в різних розділах математичної статистики для аналізу даних. Крім цього, до складу системи входить спеціалізований пакет *Statistics Toolbox*. Зупинимося лише на тих функціях для аналізу та статистичної обробки даних, які найчастіше використовуються для даних, заданих у вигляді числових масивів:

Функції для аналізу і обробки числових масивів	
Назва	**Призначення**
max(x)	максимальна компонента масиву
min(x)	мінімальна компонента масиву
mean(x)	середнє значення елементів масиву
median(x)	серединне значення елементів масиву (*медіана*)
std(x, flag), std(x)=std(x,0)	стандартне відхилення елементів масиву: $std(x,\ flag)=\sqrt{\sigma^2}$, де $\sigma^2=\frac{1}{\eta(flag)}\sum_{i=1}^{n}(x_i-\bar{x})^2$, $\bar{x}=mean(x)$, $\eta(flag)=\begin{cases}n-1, & flag=0,\\ n, & flag=1.\end{cases}$
sum(x)	сума елементів масиву
prod(x)	добуток елементів масиву
cumsum(x)	сумування з накопиченням: $y_i=\sum_{j=1}^{i}x_i$
cumprod(x)	добуток з накопиченням: $y_i=\prod_{j=1}^{i}x_i$
sort(x), sort(x, 'ascend'); sort(x, 'descend')	сортування елементів масиву за зростанням; сортування елементів масиву за спаданням

Зауваження: Для двовимірного масиву *x*, приведені в таблиці функції *f(x)* дають вектор з елементами *f(x(:,i)), i=1,2,…,m*, де *[n,m]=size(x)*.

4.3. ЗАДАЧІ ЧИСЛОВОГО АНАЛІЗУ

До бібліотеки MATLAB входять функції, що реалізують низку методів апроксимації та інтерполяції даних, числового диференціювання та інтегрування, оптимізації, розв'язання нелінійних рівнянь [3, 4, 8, 14, 17, 19, 24]. Перед означенням кожної групи функцій подамо короткий теоретичний матеріал з методів обчислень.

Алгоритми наближення функції поліномами

Нехай на відрізку [a, b] задано вузли (різні) – *вузли інтерполяції*

$$a = X_1 < X_2 < ... < X_n = b. \tag{4.1}$$

Крім того, в цих вузлах відомі значення функції $f(X_i) = Y_i$. Необхідно побудувати *інтерполяційний поліном* (ІП)

$$P_{n-1}(x) = \sum_{k=1}^{n} p_k x^{k-1} \tag{4.2}$$

такий, що

$$P_{n-1}(X_i) = Y_i, \quad i = \overline{1,n}. \tag{4.3}$$

Враховуючи (4.2), (4.3), отримуємо для невідомих коефіцієнтів p_k, $k = \overline{1,n}$ СЛАР із визначником Вандермонда, який не дорівнює нулю (з огляду на властивість вузлів (4.1)), а отже ІП існує та єдиний. Крім "поліноміальної" форми представлення (4.2) існують й інші форми запису ІП.

Форма Лагранжа дозволяє представити поліном (4.2) через значення функції у вузлах інтерполяції $f(X_i)$ у вигляді

$$L_{n-1}(x) = \sum_{k=1}^{n} f(X_k) l_k(x). \tag{4.4}$$

Для знаходження поліномів $l_k(x)$ маємо: $\sum_{k=1}^{n} f(X_k) l_k(X_i) = f(X_i)$, $i = \overline{1,n}$, а отже $l_k(X_i) = \begin{cases} 1, k = i \\ 0, k \neq i \end{cases}$, звідки отримуємо $l_k(x) = \dfrac{\omega(x)}{(x - X_k)\omega'(X_k)}$, де уведено позначення $\omega(x) = \prod_{i=1}^{n} (x - X_i)$.

Форма Ньютона є представленням полінома (4.2) у вигляді різницевого аналога формули Тейлора.

Означення 4.1. *Поділені різниці* 1-*го порядку* для функції $f(x)$ визначають наступним чином

$$f(X_i; X_j) \equiv f_{i;j} = \frac{f(X_i) - f(X_j)}{X_i - X_j} = f_{j;i}, \quad (i \neq j). \tag{4.5}$$

Поділені різниці вищих порядків визначаються рекурентно:

$$f(X_i; X_j; X_k) \equiv f_{i;j;k} = \frac{f_{i;j} - f_{j;k}}{X_i - X_k}, \ \ldots, f_{i;j;\ldots;s;k} = \frac{f_{i;j;\ldots;s} - f_{j;\ldots;s;k}}{X_i - X_k}. \tag{4.6}$$

Інтерполяційний поліном (4.2) у формі Ньютона має вид

$$N_{n-1}(x) = f(X_1) + (x - X_1) f_{1;2} + (x - X_1)(x - X_2) f_{1;2;3} + ... + (x - X_1)(x - X_2)...(x - X_{n-1}) f_{1;2;\ldots;n}. \tag{4.7}$$

Похибка інтерполяції визначається як $r_{n-1}(x) = f(x) - P_{n-1}(x)$. Якщо $f(x) \in C^{(n)}[a,b]$, то маємо $|r_{n-1}(x)| = |f(x) - P_{n-1}(x)| \leq \dfrac{M_n \cdot |\omega(x)|}{n!}$, $M_n = \max\limits_{x \in [a,b]} |f^{(n)}(x)|$.

Збіжності $|r_{n-1}(x)| \underset{n\to\infty}{\longrightarrow} 0$ у загальному випадку немає, вона суттєво залежить від вузлів сітки і гладкості функції $f(x)$.

До набору вбудованих функцій MATLAB для поліноміальної апроксимації і інтерполяції входять:

Функції апроксимації і інтерполяції даних	
Назва	**Призначення**
P=polyfit(x,y,n)	інтерполяційний поліном *P* степеня *n* для функції *y*, заданій на сітці *x*
w=griddata(x,y,z,t,r,M)	двовимірна інтерполяція функції *z(x,y)* на нерівномірній сітці; *t, r* – точки сітки, на якій визначається вихідний масив *w*, *M* – метод: *'linear'* (за замовчуванням), *'cubic'*

У системі MATLAB можна скористатися символьними обчисленнями і отримати ІП як у формі (4.4) так і у формі (4.7) (див. приклади відповідного розділу). ІП у формі (4.2) легко одержати, використовуючи вбудовану функцію `polyfit`.

Приклад 9. Побудувати інтерполяційний поліном у формі Ньютона.

Нижче наведено М-файл, який ілюструє роботу розглянутих функцій і використовує функцію користувача `ip_N` для побудови ІП у формі Ньютона:

```
%%pl4_1.m
% Приклади апроксимації і інтерполяції даних
clear all, close all, clc
a = 0; b = 10; n =10;
x = linspace(a, b, n+1);
y = sin(x);
P = polyfit(x, y, n)      %  Інтерполяційний поліном
t = linspace(a, b, 10*n+1);
w = polyval(P, t);        %  Інтерполяція даних
%  Графіки порівняння вихідної функції і інтерполяції
plot(x, y, t, w, t, sin(t))
% Побудова інтерполяційного поліному у формі Ньютона
[N, err] = ip_N(x, y);
if ~err
   N
end
```

```
function [N, err] = ip_N(X, Y)
% Побудова інтерполяційного поліному у формі Ньютона
% вхідні параметри:
%   X - вектор вузлів інтерполювання (відсутні кратні)
%   Y - вектор значень функції у вузлах інтерполювання
% вихідні параметри:
%   N - вектор, який є представленням інтерполяційного поліному Nm(t)
% err = 1- при перевірці вхідних даних знайдена помилка і
%          виконання функції аварійно закінчено
%       0- перевірка вхідних даних пройшла успішно.
%
N = sort(X); err=0;
if nargin<2
   err = 1;
   error('ip_N:NotEnoughInputs',...
         'Not enough input arguments. See ip_N.');
   return
end
```

```
n = length(X); nm = n - 1;
% перевірка, чи є кратні вузли інтерполювання ?
c = 0; i = 1;
while ~c & i<n
   j = i;  i = i + 1; c = N(j) == N(i);
end
clear N;
if c
   err = 1;
   error('ip_N:NotEnoughInputs',...
         ' Є кратні вузли інтерполювання. See ip_N.');
   return
end
% обчислення поділених різниць і
%   поліномів Wi(x)= x-X(i), i=1,...,n-1;
for i = 1 : nm
   for j = n : -1 : i+1
      Y(j) = (Y(j-1)-Y(j))./(X(j-i)-X(j));
   end
   W(i,:) = [1; -X(i)];
end
% побудова інтерполяційного поліному Nm(x)
j = 1; N(j) = Y(n);
for i = nm : -1 : 1
   N = conv(W(i,:), N);
   j = j+1; N(j) = N(j) + Y(i);
end
return
```

Числове диференціювання

Постановка задачі. Нехай

$$L_m(u(x)) = \sum_{j=0}^{m} a_j u^{(j)}(x)$$

лінійний диференціальний оператор m-го порядку. Необхідно його наближено обчислити в деякій точці $x_i \in \omega_h$, де ω_h – рівномірна або нерівномірна сітка.

Схема розв'язання. Заміняємо наближено $u(x) \approx L_n(x) \in \mathrm{P}_n(x) \;\Rightarrow\; u^{(j)}(x) \approx L_n^{(j)}(x)$.

Найпростіші формули числового диференціювання отримують, вибираючи певний *шаблон S* із точок рівномірної сітки ω_h:

Шаблон	Похідна	Позначення і обчислення	Назва похідної	Точність
$S[x_{i-1}, x_i]$	$u'(x_i)$	$u_{\bar{x},i} = \frac{u_i - u_{i-1}}{h}$	*ліва різницева похідна*	$O(h^1)$
$S[x_i, x_{i+1}]$	$u'(x_i)$	$u_{x,i} = \frac{u_{i+1} - u_i}{h}$	*права різницева похідна*	$O(h^1)$
$S[x_{i-1}, x_i, x_{i+1}]$	$u'(x_i)$	$u_{\tilde{x},i} = \frac{u_{i+1} - u_{i-1}}{2h}$	*центральна різницева похідна*	$O(h^2)$
$S[x_{i-1}, x_i, x_{i+1}]$	$u''(x_i)$	$u_{\bar{x}x,i} = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}$	*друга різницева похідна*	$O(h^2)$

Для отримання вищої точності можна збільшити кількість точок шаблону або скористатися формулою *Рунге-Ромберга*.

Нехай для обчислення $\eta(x)$ використовується деяка наближена розрахункова

формула $\xi(x,h)$ з порядком точності p

$$\eta(x) = \xi(x,h) + O(h^p) = \xi(x,h) + \varphi(x)\cdot h^p + O(h^{p+1}). \quad (4.8)$$

Використовуючи дві зв'язані сітки з кроками $h_1 = h$ і $h_2 = k \cdot h$ можна отримати для (4.8):

$$\varphi(x)\cdot h^p = \frac{\xi(x,k\cdot h) - \xi(x,h)}{1-k^p}, \quad (4.9)$$

$$\eta(x) = \xi(x,h) + \frac{\xi(x,k\cdot h) - \xi(x,h)}{1-k^p}. \quad (4.10)$$

Формула (4.9) – *перша формула Рунге-Ромберга*. Вона використовується для перевірки, чи знайдено результат з необхідною точністю ε. Формула (4.10) – *друга формула Рунге-Ромберга*. Вона використовується як для підвищення точності обчислень, так і для побудови нових розрахункових формул з порядком не нижче ніж $p+1$.

У системі MATLAB числове диференціювання представлено такими функціями:

Функції числового диференціювання	
Назва	**Призначення**
dx=diff(x)	обчислення скінчених різниць, тобто *dx=[x(2)-x(1), x(3)-x(2), ..., x(end)-x(end-1)]*
[*px,py*]=*gradient*(*z,hx,hy*)	обчислення дискретного аналога градієнта функції двох змінних *z*(*x,y*). Тут *hx,hy* – кроки диференціювання по *x* і по *y*, *px, py* – матриці, що містять компоненти градієнта
L=del2(Z), L=del2(Z,hx,hy)	обчислення "дискретного Лапласіана" для матриці *Z* (*L* =0.25*Δ*Z*)

Приклад 10. Обчислити різницеві похідні, дискретний аналога градієнта функції двох змінних та оператора Лапласа.

```
%%pl4_2.m
% Приклади числового диференціювання
close all, clear all, clc
n = 101; x = linspace(0, 2*pi, n);
n1 = n - 1; h = x(2) - x(1); hh = h.*h;
y = sin(x); dy = cos(x);

% різницева перша похідна
Dy = diff(y);
dydx = Dy./h;           %  для точок x(1:end-1) - права різниця
dydx(n) = dydx(n1);     %  для точки x(end)     - ліва різниця
disp(strcat('1-ша похідна : h  =',num2str(h),...
       ' абсолютна похибка=',num2str(norm(dydx-dy,inf))))
plot(x, dy, x, dydx)
grid on, title('1-ша різницева похідна')
pause, close all, clear Dy dydx

% різницева друга похідна
%  для точок x(2:end-1) - центральна різниця
d2ydx2 = zeros(1,n-2);
for k = 2 : n1
   k1 = k-1;
   d2ydx2(k1) = (y(k1)-2*y(k)+y(k+1))./hh;
end
disp(strcat('2-га похідна : h^2=',num2str(hh),...
       ' абсолютна похибка=',num2str(norm(d2ydx2-(-y(2:n1)),inf))))
plot(x, -y, x(2:n1), d2ydx2)
```

```
grid on, title('2-га різницева похідна')
pause, close all, clear x y d2ydx2

f = @(x,y)(16 - x.^4 - y.^4);
x = linspace(-2, 2, 21); hx = x(2) - x(1);
y = linspace(-2, 2, 21); hy = y(2) - y(1);
[X, Y] = meshgrid(x, y); Z = f(X, Y);
[Px, Py] = gradient(Z, hx, hy);
contour(Z, 10), hold on
quiver(Px, Py), hold off
s = char(f); title(s)
pause, close all, clear Px Py
surfc(X, Y, Z), grid on, title(s)
pause, close all
L = 4.*del2(Z, hx, hy);
surfc(X, Y, L), grid on
s = ['{\Delta}',s]; title(s)
pause, close all
```

Отримаємо результат (графіки див. при запуску в MATLAB):

```
1-ша похідна : h  =0.062832 абсолютна похибка=0.031406
2-га похідна : h^2=0.0039478 абсолютна похибка=0.00032894
```

Числове інтегрування

Постановка задачі. Необхідно наближено обчислити визначений інтеграл

$$I=\int_a^b f(x)dx \quad \text{або} \quad I=\int_a^b \rho(x)u(x)dx, \tag{4.11}$$

де $\rho(x)>0$ – вагова функція, неперервна на (a,b), $f(x)|u(x)$ – підінтегральні функції достатньої гладкості на $[a,b]$.

Більшість методів числового інтегрування ґрунтуються на заміні (4.11) скінченною сумою – *квадратурними формулами* (КФ)

$$I_n=\sum_{k=1}^{n} A_k f_k \quad \text{або} \quad I_n=\sum_{k=1}^{n} A_k u_k\,, \tag{4.12}$$

де A_k – *квадратурні коефіцієнти*, x_k – *вузли* КФ.

Означення 4.2. Якщо граничні точки $x=a$, $x=b$ входять до системи вузлів КФ, то формули (4.12) називаються формулами *закритого* типу, інакше – *відкритого* типу.

Означення 4.3. КФ називається *чисельно стійкою*, якщо всі A_k невід'ємні.

Означення 4.4. *Похибкою* КФ називається величина $|I-I_n|=R_n$. Вона залежить від квадратурних коефіцієнтів і вузлів $\{A_k\}$, $\{x_k\}$, а також гладкості функції $f|u$.

Формули (4.12) в цілому містять $2n$ коефіцієнтів, тому є можливість будувати різні КФ, що ґрунтуються на певній стратегії:

- різні методи заміни підінтегральної функції $f|u$, як на всьому інтервалі $[a,b]$, так і на його складових $[x_{k-1},x_k]$;
- фіксація вибраного набору параметрів $\{x_k\}$ або $\{A_k\}$ з подальшим визначенням решти параметрів, що задовольняють певним додатковим умовам;
- забезпечення найменшої оцінки залишку R_n для всіх функції $u(x)$ в деякому класі U;

- забезпечення *точного* ($R_n = 0$) обчислення (4.11) формулою (4.12) для всіх функції $u(x)$ з деякого класу U.

Найпростіші формули числового інтегрування отримують, вибираючи на $[a,b]$ фіксовану рівномірну сітку $\omega_h = \{x_k = a + (k-1)h, k = \overline{1,n}, h = (b-a)/(n-1)\}$ і користуючись адитивністю (4.11)

$$I = \int_a^b f(x)dx = \sum_{k=2}^{n} \int_{x_{k-1}}^{x_k} f(x)dx. \qquad (4.13)$$

Назва	**Обчислення** $I_n(k) = \int_{x_{k-1}}^{x_k} f(x)dx$	**Застосування складеної КФ** $I_n = \sum_{k=2}^{n} I_n(k)$	**Точність на [a,b]**
1. *КФ прямокутників а) лівих*	hf_{k-1}	$h\sum_{k=1}^{n-1} f_k$	$O(h^1)$
б) правих	hf_k	$h\sum_{k=2}^{n} f_k$	$O(h^1)$
г) центральних (КФ середніх)	$hf_{k-0.5}$, $x_{k-0.5} = 0.5(x_{k-1} + x_k)$	$h\sum_{k=2}^{n} f_{k-0.5}$	$O(h^2)$
2. *КФ трапецій*	$0.5h(f_{k-1} + f_k)$	$h\left(0.5(f_1 + f_n) + \sum_{k=2}^{n-1} f_k\right)$	$O(h^2)$
3. *КФ Сімпсона*	$\frac{0.5h}{3}(f_{k-1} + 4f_{k-0.5} + f_k)$	$\frac{h}{3}\left(f_1 + f_n + 4\sum_{k=2,4,\ldots}^{n-1} f_k + 2\sum_{k=3,5,\ldots}^{n-2} f_k\right)$	$O(h^4)$

Для отримання значення (4.11) із заданою точністю $0 < \varepsilon < 1$ використовують адаптивні алгоритми (як рекурентні так і рекурсивні), які будуються на апостеріорній оцінці похибки R_n за допомогою першої формули Рунге-Ромберга.

У системі MATLAB числове інтегрування представлено вбудованими функціями:

Функції числового інтегрування	
Назва	**Призначення**
*s=h.*trapz(y)*	метод трапецій для дискретної функції *y* з постійним кроком *h*
s=trapz(x,y)	метод трапецій для дискретної функції *y*, заданої у вузлах сітки *x*
s=quad(@f,a,b,tol)	метод Сімпсона
s=quadl(@f,a,b,tol)	метод Сімпсона (підвищеної точності) для функції $f(x)$, описаної у М-файлі *f.m*; *a,b* – інтервал інтегрування; *tol* – точність рекурсивного адаптивного алгоритму (необов'язковий параметр, за замовчуванням рівний 1*e*-6)
s=dblquad(@f,a,b,c,d)	метод Сімпсона для функції $f(x,y)$, описаної у М-файлі *f.m*; x ∈ *a,b*, y ∈ *c,d*

Приклад 11. Обчислити визначений інтеграл (4.11) для a=1, b=2, $f(x) = x\sin(x)$ за формулами чисельного інтегрування, використовуючи вбудовані функцій MATLAB.

```
function pl4_3
%%PL4_3   Приклади числового інтегрування
close all, clear all, clc
%pkg load symbolic  % тільки для Octave !
```

```
a = 1; b = 2;
x = linspace(a, b, 11); h = x(2) - x(1);
n = length(x); y = f(x); n1 = n - 1;
disp(' КФ прямокутників :')
Pl = h .* sum(y(1:n1));                    % лівих
Pr = h .* sum(y(2:n));                     % правих
Pz = h .* sum(f(0.5.*(x(1:n1)+x(2:n)))); % центральних
disp([Pl Pr Pz])
disp(' КФ трапецій :')
It1 = h .* trapz(y);          %  з рівномірним кроком
x = [1,1.15,1.2,1.253,1.35,1.5,1.65,1.8,1.9,1.93,2];
y = f(x); It2 = trapz(x, y);  %  з нерівномірним кроком
% реалізація м.трапецій з нерівномірним кроком
It3 = 0;
for i = 1 : n1
    ip = i+1; It3 = It3 + (x(ip)-x(i)).*(y(i)+y(ip));
end
It3 = 0.5.*It3;
It4 = 0.5.*dot(diff(x),y(1:n1)+y(2:n)); % ще один варіант
disp([It1 It2 It3 It4])
disp(' КФ Сімпсона (рекурсивний адаптивний алгоритм):')
Is = quad(@f, a, b, 1e-4);    % точність користувача  1e-4
Isd = quadl(@f, a, b);        % точність за угодою  1e-6
Isr = i_simp(a, b, @f, 1e-6, 1000); % рекурентний адаптивний
disp([Is Isd Isr])
disp(' Символьне обчислення визначеного інтегралу :')
syms z; F = f(z); I = vpa(int(F, z, a, b), 10)
disp(' Всі результати :')
Z = vpa([Pl, Pr, Pz, It1, It2, Is, Isd], 10)
disp(' абсолютна похибка :')
disp(abs(Z - I))

function s = f(t)
% означення підінтегральної функції
s = t .* sin(t);
```

Отримаємо результати (MATLAB):

```
КФ прямокутників :
       1.3905       1.4882        1.441
КФ трапецій :
       1.4393       1.4388       1.4388       1.4388
КФ Сімпсона (рекурсивний адаптивний алгоритм):
       1.4404       1.4404       1.4404
Символьне обчислення визначеного інтегралу :
I =
1.440422421
Всі результати :
Z =
 [ 1.390478757, 1.488191144, 1.440966217, 1.439334951, 1.438761144, 1.440422419, 1.440422421]
абсолютна похибка :
[ .49943664e-1, .47768723e-1, .543796e-3, .1087470e-2, .1661277e-2,      .2e-8,          0.]
```

ЛЕКЦІЯ № 05

5.1. НАБЛИЖЕННЯ ФУНКЦІЙ ЛІНІЙНИМИ І КУБІЧНИМИ СПЛАЙНАМИ

Ураховуючи, що в загальному випадку відсутня збіжність системи інтерполяційних поліномів (ІП) на всьому відрізку наближення, було запропоновано будувати не один ІП на всьому відрізку $[a, b]$, а використовувати *кусково-поліноміальну інтерполяцію*. У цьому випадку на кожному інтервалі Δ_i застосовують поліном зазвичай низького степеня. Коефіцієнти кожного із поліномів цієї системи визначають із умов інтерполяції на сітці та деяких додаткових умов – відомостей про функцію, що інтерполюється.

Побудований у такий спосіб інтерполянт називають *сплайн-функцією* або просто *сплайном*. Перевага сплайнів перед звичайною інтерполяцією полягає у простоті обчислень полінома, стійкості процесу обчислень, збіжності.

Назва "сплайн" походить від англійського *spline*, що означає гнучку металеву лінійку, яка використовувалась як лекало (інструмент кресляра для проведення гладкої лінії через задані точки на площині). Профіль лінійки, розташований між двома сусідніми точками, математично описує кубічний поліном.

Нехай на відрізку $[a, b]$ задана *сітка* з $n > 1$ вузлами

$$a = X_1 < X_2 < ... < X_{n-1} < X_n = b. \tag{5.1}$$

Уведемо позначення:

$$\begin{aligned} &\Delta_{[a,b]} = \bigcup_{i=1}^{n-1} \Delta_i, \\ &\Delta_i = [X_i, X_{i+1}], \\ &h_i = X_{i+1} - X_i,\ i = 1,...,n-1. \end{aligned} \tag{5.2}$$

На сітці відомі значення деякої функції $u(x)$:

$$y_i = u(X_i),\ \ i = 1,...,n.$$

Означення 5.1. Функцію $S_{m,q}(x),\ x \in [a, b]$ називають *сплайном степеня* $m > 0$ *дефекту* $q\ (0 \le q \le m)$, якщо

а) на кожному відрізку $\Delta_i,\ i = 1,...,n-1$ вона є поліномом степеня m, тобто

$$S_{m,q}(x) = \sum_{k=0}^{m} s_{i,k}(x - X_i)^k,\ x \in \Delta_i; \tag{5.3}$$

б) функція $S_{m,q}(x)$ неперервна на $[a, b]$ разом зі своїми похідними до степеня $m - q$ включно:

$$S_{m,q}(x) \in C^{m-q}([a,b]). \tag{5.4}$$

Означення 5.2. Сплайн $S_{m,q}(x),\ x \in [a, b]$ називають *інтерполяційним* для функції $u(x)$ і позначають $S_{m,q}(x;u)$, якщо він задовольняє умову інтерполяції у вузлах сітки

$$S_{m,q}(X_i) = y_i,\ i = 1,...,n. \tag{5.5}$$

Представлення (5.3) можна записати і в інших формах, наприклад, "поліноміальній" або у формі ″лагранжіана″.

Побудова інтерполяційного лінійного сплайна

Ураховуючи, що для ІС першого порядку виконуються умови (5.4)–(5.5), дефект лінійного сплайна дорівнюватиме одиниці. Для ІС $S_{1,1}(x;u), x \in \Delta_{[a,b]}$ використовуватимемо "сплайнове" зображення

$$S_{1,1}(x;u) = a_i(x - X_i) + b_i,\ x \in \Delta_i, i = 1,...,n-1. \tag{5.6}$$

Система (5.6) визначається коефіцієнтами $\{a_i, b_i\}_{i=1}^{n-1}$, кількість яких дорівнює 2(n–1). Для їх знаходження скористаємося зображенням (5.6) та умовами (5.4), (5.5). У результаті маємо систему

$$\begin{cases} b_i = y_i,\ \ i = 1,...,n-1, \\ a_{n-1}h_{n-1} + b_{n-1} = y_n,\ \ i = n, \\ a_{i-1}h_{i-1} + b_{i-1} = b_i,\ \ i = 2,...,n-1 \end{cases}$$

із кількістю рівнянь 2(n–1). Її розв'язок легко знайти в явній (алгебраїчній) формі:

$$\begin{cases} b_i = y_i, \\ a_i = \dfrac{b_{i+1} - b_i}{h_i} \equiv \dfrac{y_{i+1} - y_i}{X_{i+1} - X_i} \equiv y_{i;i+1},\ i = 1,...,n-1. \end{cases} \tag{5.7}$$

Тут ми формально позначили $b_n = y_n$, а також використали стандартне позначення $y_{i;i+1} = y(X_i; X_{i+1})$ для *поділених різниць першого порядку*.

Зауважимо, що подання одних коефіцієнтів (a_i) за допомогою інших (b_i) є типовим при побудові сплайнів.

Досить часто при застосуванні сплайн-функцій необхідно представити ІС на відрізку $[X_i, X_{i+1}]$ сітки у формі "лагранжіана":

$$\begin{aligned} &S_{1,1}(x;u) = y_i\rho_{i,1}(x) + y_{i+1}\rho_{i,2}(x),\ x \in \Delta_i, \\ &\rho_{i,1}(x) = (X_{i+1} - x)h_i^{-1}; \rho_{i,2}(x) = (x - X_i)h_i^{-1}. \end{aligned} \tag{5.8}$$

Якщо увести позначення

$$t = (x - X_i)h_i^{-1},\ x \in \Delta_i,\ i = 1,...,n-1,\ t \in [0,1], \tag{5.9}$$

то сплайн (8) набуває вигляду

$$\begin{aligned} &S_{1,1}(x;u) = y_i\psi_1(t) + y_{i+1}\psi_2(t); \\ &x \in \Delta_i,\ i = 1,...,n-1,\ t \in [0,1]; \\ &\psi_1(t) = 1 - t,\ \psi_2(t) = t. \end{aligned} \tag{5.10}$$

Побудова інтерполяційного кубічного сплайна

Оскільки для сплайна третього порядку має виконуватись умова (5.4) із гладкістю до другої похідної включно, дефект кубічного сплайна дорівнюватиме 1.

Для ІС $S_{3,1}(x;u), x \in \Delta_{[a,b]}$ використовуватимемо "сплайнове" зображення

$$\begin{aligned} &S_{3,1}(x;u) = a_i(x - X_i)^3 + b_i(x - X_i)^2 + c_i(x - X_i) + d_i, \\ &\qquad x \in \Delta_i,\ i = 1,...,n-1. \end{aligned} \tag{5.11}$$

Легко визначити зміст коефіцієнтів у (5.11):

$$d_i = S_{3,1}(X_i;u), c_i = S'_{3,1}(X_i;u), b_i = \frac{1}{2}S''_{3,1}(X_i;u), a_i = \frac{1}{6}S'''_{3,1}(X_i;u).$$

Система (5.11) визначається коефіцієнтами $\{a_i, b_i, c_i, d_i\}_{i=1}^{n-1}$, кількість яких дорівнює

4(n–1). Для їх знаходження використаємо:

1) умову інтерполяції (5.5), що дає систему (n – 1) + 1 рівнянь

$$\begin{cases} d_i = y_i, \ i = 1,...,n-1, \\ a_{n-1}h_{n-1}^3 + b_{n-1}h_{n-1}^2 + c_{n-1}h_{n-1} + d_{n-1} = y_n, \ i = n; \end{cases} \tag{5.12}$$

2) умови гладкості сплайна (5.4) в усіх внутрішніх вузлах сітки

$$S_{3,1}^{(p)}(X_i - 0; u) = S_{3,1}^{(p)}(X_i + 0; u), \ p = 0,1,2, \ i = 2,...,n-1,$$

що дає систему 3(n–2) рівнянь

$$a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i = d_{i+1}, \tag{5.13}$$

$$3a_i h_i^2 + 2b_i h_i + c_i = c_{i+1}, \tag{5.14}$$

$$6a_i h_i + 2b_i = 2b_{i+1}, \ i = 1,...,n-2; \tag{5.15}$$

3) дві додаткові умови, необхідні для утворення замкненої системи рівнянь. Найчастіше використовують такі додаткові умови – *типи крайових умов для кубічного сплайна*:

I. $S'_{3,1}(a;u) = A, \ S'_{3,1}(b;u) = B;$ (5.16)

II. $S''_{3,1}(a;u) = A, \ S''_{3,1}(b;u) = B;$ (5.17)

III. $S_{3,1}^{(p)}(a;u) = S_{3,1}^{(p)}(b;u), \ p = 1,2;$ (5.18)

IV. $S'''_{3,1}(X_j + 0; u) = S'''_{3,1}(X_j - 0; u), \ j = 2, n-1.$ (5.19)

Система (5.12)–(5.15), доповнена одним з рівнянь (5.16)–(5.19), може бути розв'язана безпосередньо, але краще (з огляду на кількість операцій і об'єм оперативної пам'яті) визначити, наприклад, $\{a_i, b_i\}_{i=1}^{n-1}$ через $\{c_i\}_{i=1}^{n-1}$. При цьому ми отримаємо для $\{c_i\}_{i=1}^{n-1}$ СЛАР з тридіагональною матрицею, яка має діагональну перевагу.

Вбудовані функції системи MATLAB для роботи зі сплайнами.

Для кусково-поліноміальної апроксимації табличних функцій $y(x)$ призначена функція *interp*1:

yy = *interp*1(*x*, *y*, *xx*, *method*),

де x – абсциси функції, що апроксимується; y – ординати функції, що апроксимується; *xx* – абсциси точок, в яких обчислюють значення апроксимуючих поліномів, що повертаються у вигляді вектора *yy*; *method* – спосіб апроксимації, що задається як рядок символів (насправді достатньо вказати лише перший символ).

Параметр *method* може набувати одного зі значень:

• '*nearest*' – інтерполяція за сусідніми точками, за якої значення у довільній точці дорівнює значенню у найближчій вузловій точці (апроксимація кусковими поліномами нульового степеня («сходинками»): для довільного проміжного значення *xx* необхідно знайти найближче табличне x_i і за *yy* прийняти відповідне табличне значення y_i);

• '*linear*' – лінійна інтерполяція (апроксимація ламаними – кусковими поліномами першого степеня);

• *'spline'* – кубічний сплайн (апроксимація сплайнами – кусковими поліномами третього степеня);

• *'pchip'* або *'cubic'* – апроксимація кусковими поліномами Ерміта третього степеня.

Якщо параметр *method* опущено, то за замовчуванням використовують *'linear'*.

За апроксимації методом *'spline'* у вузлових точках – неперервні перша і друга похідні, а у внутрішніх вузлах, сусідніх із кінцевими, – неперервна також і третя

похідна (умова (5.19)).

Назви методів *'spline'* та *'pchip'* є іменами функцій, до яких звертається функція *interp*1 (інші два методи *'nearest'* та *'linear'* опрацьовуються безпосередньо в *interp*1). Звернення до цих функцій має два різні формати:

1) *yy = spline*(*x*, *y*, *xx*) та *yy = pchip*(*x*, *y*, *xx*);

2) *pp = spline*(*x*, *y*) та *pp = pchip*(*x*, *y*).

У першому варіанті вхідні та вихідні параметри мають той самий зміст, що й при зверненні до *interp*1.

У другому варіанті обидві функції повертають *pp* – структуру, так звану кусково-поліноміальну форму (*ppform*, *pieceweise polinomial form*) – спеціальний внутрішній формат системи MATLAB для представлення сплайнів. Ці структури можуть задаватися як аргументи при зверненні до функції *ppval* для обчислення значень апроксимуючих функцій у формі *yy = ppval*(*pp*, *xx*).

У системі MATLAB також реалізована багатовимірна інтерполяція, яка представлена наступними процедурами для інтерполяції функцій 2-х, 3-х і *n* змінних:

- `UI=interp2(X1,X2,U,XX1,XX2,method);`
- `UI=interp3(X1,X2,X3,U,XX1,XX2,XX3,method);`
- `UI=interpn(X1,X2,...,Xn,U,XX1,XX2,...,XXn,method);`

де `U` – функція з описом `U(X1,...,Xn)`;

`method:=` '`nearest`' | '`linear`' | '`spline`' | '`cubic`' (деталі див. відповідний `help`).

Розглянемо практичне застосування вбудованих функції для роботи зі сплайнами.

Приклад 12. Наблизити функції `f(x)`, для яких відсутня збіжність системи інтерполяційних поліномів на всьому відрізку наближення :

`1) f(x)=1/(1+25*x^2)`

`2) f(x)=abs(x)`

за допомогою $S_{0,1}(x; f)$ – сплайна нульового степеня і зобразити графік табуляції вихідної функції `f(T)` і сплайна $S_{0,1}(T; f)$ у точках масиву табуляції `T`.

```
%%pl5_1.m
% Приклади апроксимації і інтерполяції даних
% з використанням інтерполяційного сплайна порядку 0
% Виконати на рівномірній сітці наближення
% 1) функції f(x)=1/(1+25*x^2) на відрізку [-1,1];
% 2) функції f(x)=|x| на відрізку [-1,1].
clear all, close all, clc
SP = 51; a = -1; b = 1; X = linspace(a, b, SP);
T = linspace(a, b, 201);  % масив табуляції

% 1) приклад К.Рунге
f = @(x)(1./(1+25.*x.*x));
NF(1) = {'f(x)'}; F(1,:) = f(T); y = f(X);
NF(2) = {strcat('S0(x;f) -',int2str(SP))};
F(2,:) = interp1(X, y, T, 'nearest');
plot(T, F), grid on
title('Наближення f(x)=1/(1+25x^2) ІП S0(x;f)')
legend(NF,'Location','Best')
```

```
pause, close all

% 2) приклад С.Н.Бернштейна
f = @(x)(abs(x));
F(1,:) = f(T); y = f(X);
F(2,:) = interp1(X, y, T, 'nearest');
plot(T, F), grid on
title('Наближення f(x)=|x| ІП S0(x;f)')
legend(NF,'Location','Best')
pause, close all
```

Приклад 13. Розв'язати задачу з попереднього прикладу за допомогою $S_{3,1}(x; f)$ – кубічного сплайна.

```
%%pl5_2.m
% Приклади апроксимації і інтерполяції даних
% з використанням кубічного інтерполяційного сплайна
% Виконати на рівномірній сітці наближення
% 1) функції f(x)=1/(1+25*x^2) на відрізку [-1,1];
% 2) функції f(x)=|x| на відрізку [-1,1].
clear all, close all, clc
SP = 21; a = -1; b = 1; X = linspace(a, b, SP);
T = linspace(a, b, 201);  % масив табуляції

% 1) приклад К.Рунге
f = @(x)(1./(1+25.*x.*x));
NF(1) = {'f(x)'}; F(1,:) = f(T); y = f(X);
NF(2) = {strcat('S3(x;f) -',int2str(SP))};
F(2,:) = spline(X, y, T);
plot(T, F), grid on
title('Наближення f(x)=1/(1+25x^2) ІП S3(x;f)')
legend(NF,'Location','Best')
pause, close all

% 2) приклад С.Н.Бернштейна
f = @(x)(abs(x));
F(1,:) = f(T); y = f(X);
F(2,:) = spline(X, y, T);
plot(T, F), grid on
title('Наближення f(x)=|x| ІП S3(x;f)')
legend(NF,'Location','Best')
pause, close all
```

Приклад 14. Для дискретно-заданої функції побудувати кубічний сплайн і вивести на екран *pp* – структуру.

```
%%pl5_3.m
% Побудова кубічного сплайна для дискретної функції
clear all, close all, clc
x = -3 : 3; y = sign(x);
plot(x, y, 'o'), axis( [-3.5, 3.5, -1.5, 1.5] )
xlabel('x'), ylabel('y')
pp = spline(x, y)
X = pp.breaks
C = pp.coefs
```

Результат виведення на екран *pp* – структури (з доданими коментарями у правому стовпці для пояснення полів):

```
pp =

      form: 'pp'                 форма сплайна
    breaks: [-3 -2 -1 0 1 2 3]   масив точок розбиття
     coefs: [6x4 double]         масив коефіцієнтів сплайна
    pieces: 6                    кількість поліномів, що складають сплайн
     order: 4                    кількість коефіцієнтів в поліномі
       dim: 1                    розмірність


X =

    -3    -2    -1     0     1     2     3

C =

       0.25        -0.75          0.5           -1
       0.25            0        -0.25           -1
      -0.25         0.75          0.5           -1
      -0.25            0         1.25            0
       0.25        -0.75          0.5            1
       0.25  5.5511e-017        -0.25            1
```

Коротка інформація про деякі інші вбудовані функції для роботи з сплайнами:

- побудова кубічних сплайнів з спеціальними крайовими умовами – функції *csape*, *csapi*;
- обчислення похідних від поліноміальних сплайнів – функція *fnder*;
- побудова згладжуючих сплайнів – функція *csaps*.

Є також можливість застосування сплайнів при параметричній інтерполяції для заданих табличних функцій $x = x(t),\ y = y(t)$. Для детальнішого ознайомлення з основними питаннями теорії поліноміальних сплайнів та їх застосувань у прикладних задачах і способами використання вбудованих функцій пакету MATLAB для роботи зі сплайнами можна рекомендувати, наприклад, [7, 22].

5.2. МІНІМІЗАЦІЯ ФУНКЦІЙ І РОЗВ'ЯЗАННЯ СИСТЕМ НЕЛІНІЙНИХ РІВНЯНЬ

Задача знаходження розв'язків системи нелінійних рівнянь

$$\begin{cases} f_i(x_1, x_2, ..., x_n) = 0, \\ i = 1,2,...,n. \end{cases} \tag{5.20}$$

з n невідомими $\vec{\mathbf{x}} = [x_1, x_2, ..., x_n]$, де $\vec{\mathbf{x}} \in$ деякій метричній множині X, еквівалентна задачі мінімізації без обмежень функції (пошуку локальних мінімумів $\vec{\mathbf{x}}_*$ або абсолютного мінімуму $\vec{\mathbf{x}}_\bullet$):

$$F(\vec{\mathbf{x}}_*) \to \min \tag{5.21a}$$

або

$$F(\vec{\mathbf{x}}_\bullet) = \inf_{\mathbf{x}} (F(\vec{\mathbf{x}})), \tag{5.21b}$$

де

$$F(\vec{\mathrm{x}}) \equiv F(x_1, x_2, ..., x_n) \equiv \sum_{i=1}^{n} | f_i(x_1, x_2, ..., x_n) |^2 . \tag{5.22}$$

Звичайно, задача пошуку мінімуму (або максимуму) функції *n* невідомих і сама по собі має велике практичне значення.

Відомо, що розв'язок задачі (5.21а) можна знайти, розв'язавши нелінійну систему рівнянь

$$\begin{cases} \dfrac{\partial F(x_1, x_2, ..., x_n)}{\partial x_i} = 0, \\ i = 1,2,...,n. \end{cases} \tag{5.23}$$

Для розв'язання систем виду (5.20) використовуються ітераційні методи, зокрема, *простої ітерації, типу Зейделя, Ньютона, спуску* та інші. Але для задачі мінімізації (5.21) будують свої числові методи, без приведення до форми (5.23). Ці методи також ітераційні, серед них є досить велика група методів *спуску* (рух в напрямку спадання функції $F(\vec{\mathbf{x}})$), які будуються в наступному виді:

$$\vec{\mathbf{x}}^{(\mathrm{k}+1)} = \vec{\mathbf{x}}^{(\mathrm{k})} + \alpha_k \cdot \vec{\mathbf{p}}^{(\mathrm{k})}, \tag{5.24}$$

тут $\vec{\mathbf{p}}^{(\mathrm{k})}$ – вектор, який визначає напрямок руху з точки $\vec{\mathbf{x}}^{(\mathrm{k})}$, α_k – число, яке визначає довжину кроку в напрямку $\vec{\mathbf{p}}^{(\mathrm{k})}$. Серед представників цієї групи методів – *спуск по координатам, найшвидший спуск, балковий метод, спряжений метод, випадковий пошук, метод Ньютона з регулюванням кроку* та інші.

Обмежимось цими загальними поняттями і не будемо розглядати алгоритми існуючих ітераційних методів мінімізації, як і ітераційних методів розв'язання нелінійних рівнянь і нелінійних систем рівнянь. Приклади програмної реалізації деяких з цих методів наведено в файлах `pl5_5.m, pl5_6.m`

В пакеті MATLAB відсутня пряма реалізація ітераційних методів, але є функції, що реалізують низку методів оптимізації і розв'язання нелінійних рівнянь:

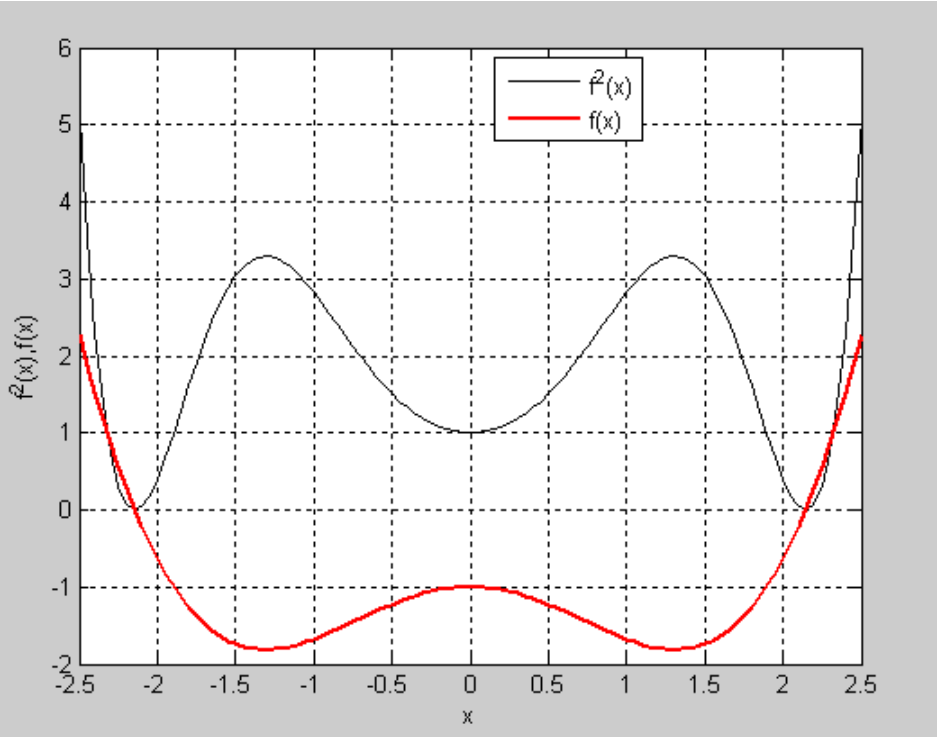
Функції мінімізації функцій і знаходження коренів	
Назва	**Призначення**
fmin('f ',xn, xk); *fminbnd('f ',xn, xk)*	знаходження мінімуму функції однієї змінної на інтервалі [*xn, xk*]; знаходження мінімуму функції однієї змінної на інтервалі [*xn, xk*] (у *V7.6*)
fmins('f ',x0); *fminsearch('f ',x0)*	знаходження мінімуму функції декількох змінних в околі точки *x0*; знаходження мінімуму функції декількох змінних в околі точки *x0* (у *V7.6*)
fzero('f ',x0); *fzero('f ',[xn, xk])*	знаходження нуля функції поблизу точки *x0*, наприклад: *fzero('x^3–27',1);* пошук нуля функції на інтервалі [*xn, xk*], де *f*(*xn*)**f*(*xk*)<0 (кращий варіант !)

Приклад 15. Розглянемо простий приклад застосування деяких з цих функцій – знайти найбільший додатний корінь нелінійного рівняння $f(x) \equiv x^2 - 2x\sin(x) - 1 = 0$.

```
%%pl5_4.m
% Знаходження кореня нелінійного рівняння f(x)=0.
close all, clear all, clc, warning off
f = @(x)(x.*x-2.*x.*sin(x)-1); % опис функції для якої шукаємо корінь
F = @(x)(f(x).^2);             % опис функції для мінімізації
```

```
x = linspace(-2.5,2.5,101);
plot(x,F(x),'k'), hold on
plot(x,f(x),'r','LineWidth',2), hold off, grid on
legend('f^{2}(x)','f(x)','Location','Best')
xlabel('x'), ylabel('f^{2}(x),f(x)'), clear x

D = input('Інтервал, де знаходиться корінь=');
z = input('Наближення=');
x = fminbnd(F,D(1),D(2));
fprintf('Мінімізація  : F(%.6f)=%.6e\n', x, F(x));
x = fzero(f, z);  % 1-й варіант виклику
fprintf('Функція fzero: f(%.6f)=%.6e\n', x, f(x));
%з розширеним переліком результатів і виведенням процесу пошуку
[x, fx, ef] = fzero(f, D, optimset('Display','iter'));
fprintf('f(%.6f)=%.6e Код завершення=%2d\n', x, fx, ef);
```

Результат роботи цього М-сценарію – графік функцій $f(x)$ і $F(x)=f^2(x)$:



і виведена інформація в командному вікні :

```
Інтервал, де знаходиться корінь=[1.8,2.4]
Наближення=2.25
Мінімізація  : F(2.145212)=6.674724e-009
Функція fzero: f(2.145196)=0.000000e+000

 Func-count     x          f(x)             Procedure
    2             1.8       -1.26585        initial
    3          2.07285      -0.337401       interpolation
    4          2.15397       0.0436669      interpolation
    5          2.14467    -0.00257528       interpolation
    6          2.14519  -1.76832e-005       interpolation
    7           2.1452   9.19673e-011       interpolation
    8           2.1452              0       interpolation

Zero found in the interval [1.8, 2.4]
f(2.145196)=0.000000e+000 Код завершення= 1
```

```
function p15_5
%% Приклади знаходження коренів нелінійного рівняння
close all, clear all, clc, warning off
global F dF
F = [1, 0, -4, 5, -20];  %  поліном (ліва частина рівняння)
dF = polyder(F);         %  похідна від полінома
e = 1;
while e
     a = input('? a='); b = input('? b='); h = input('? h=');
     X = [a : h : b]; plot(X, f(X)); grid on
     e = input('? продовжим шукати область коренів (1-да|0-ні) =');
end
close all
X0 = input('? Початкове наближення='); n = length(X0); e = 1e-10;
disp(strcat('Точність для м.Ньютона=',num2str(e)));
for k = 1 : n
   x = X0(k);
   [x, it] = m_njut1(@fdf, x, e, 1000);
   disp(strcat('корінь=',num2str(x,13),' ітерацій=',num2str(it)))
   if isreal(X0(k))
      x = fzero(@f, X0(k)); disp(strcat(' fzero=',num2str(x,13)))
   end
end
R = sort(roots(F)); format long e
disp('Корені полінома :'), disp(R)

function s = f(x)       % підфункція функції p15_5
global F
s = polyval(F, x);

function s = fdf(x)     % підфункція функції p15_5
global F dF
s(1) = f(x);
s(2) = polyval(dF, x);
```

```
function [x, k] = m_njut1(fdf, x, e, mki)
%% M_NJUT1   м.Ньютона для знаходження кореня  f(x)=0.
% Метод Ньютона для знаходження кореня  f(x)=0
for k = 1 : mki
   o = fdf(x);  o = o(1)./o(2); x = x - o;
   if abs(o) < e
      break
   end
end
```

```
function p15_6
% Приклади знаходження розв'язків
%  систем нелінійних рівнянь  F(X)=0.
close all, clear all, clc, warning off
ezplot('cos(0.4*y+x^2)+x^2+y^2-1.6',[-2,2]), hold on
ezplot('1.5*x^2-y^2/0.36-1',[-2,2]), grid on
X0 = input('? Початкове наближення=');
kn = size(X0); n = kn(1);
% Створення структури для зберігання коренів :
X = zeros(1, n); S = struct('x', X, 'y', X, 'kit', X);
e = 1e-10; mki = 1000;  % параметри ітераційного процесу
disp(strcat('Точність для м.Ньютона=', num2str(e),...
            ' Обмеж.кільк.іт=', int2str(mki)));
```

```
for k = 1 : n
   X = (X0(k, :))';
   [X, it] = m_njuts(X, e, mki, @F, @dF);
   S.x(k) = X(1); S.y(k) = X(2); S.kit(k) = it;
end
s1 = '            Розв''язок системи НР\n';
s2 = '----------x-------|----------y-------|--іт-\n';
s = strcat(s1, s2);
fprintf(s);
for k = 1 : n
   fprintf('%17.8f |%17.8f |%5d\n', S.x(k), S.y(k), S.kit(k));
end

function f = F(X)       % підфункція функції p15_6
% Обчислення лівої частини НС F(X)
x2 = X(1).^2; y = X(2); y2 = y.*y;
f = [cos(0.4 .* y + x2) + x2 + y2 - 1.6;...
     1.5 .* x2 - y2 ./ 0.36 - 1];

function J = dF(X)      % підфункція функції p15_6
% Обчислення якобіана від F(X)
x = X(1); x2 = x.*x; y = X(2); y2 = y.*y;
s = -sin(0.4 .* y + x2);
J = [2 .* x .* (1 + s), 2.*y + 0.4 .* s;...
                3 .* x, -y./0.18];

function [X, k] = m_njuts(X, e, mki, ffun, dffun)
%% M_NJUTS   м.Ньютона для знаходження коренів системи F(X)=0.
% Знаходження розв'язку нелінійної системи рівнянь F(X) = 0
% ітераційним методом Ньютона  (dFi(X(n)/dXj)(X(n+1)-X(n))-F(X(n)),
% де значення X(0) відоме.
% Умова закінчення ітерацій : norm(X(n+1)-X(n))<e.
% вхідні дані :
%   X      - початкове наближення до розв'язку;
%   e      - точність (0<e<1);
%   mki    - максимальна кількість кроків ітерацій;
%   ffun   - вектор-стовбець (функція F(X));
%   dffun  - матриця Якобі (dFi(X)/dXj).
% вихідні дані:
%   X      - фінішне наближення до розв'язку;
%   k      - кількість виконаних ітерацій.
%
if nargin<5
  error('M_NJUTS:Not Enough Inputs',...
        'Not enough input arguments.  See M_NJUTS.');
  return
end
if e <= 0 | e >= 1
  error('M_NJUTS:error e <= 0 | e >= 1.',...
        'e <= 0 | e >= 1.  See M_NJUTS.');
  return
end
k = 0; F = true;
while F & k < mki
   k = k + 1; O = dffun(X) \ ffun(X);
   X = X - O; F = norm(O) > e;
end
```

ЛЕКЦІЯ № 06

М-ФУНКЦІЇ ТА ЇХ ВИКОРИСТАННЯ

Опис функції. Опис функції здійснюється в окремому файлі і його ім'я має співпадати з ім'ям функції, яка в ньому розташована. Синтаксис цього файла (*ff.m*):

```
function [R1 R2 … Rn] = ff(x1,x2,…,xn)
%FF   Короткий зміст функції ff
% Детальний опис функції ff
%xi – вхідні формальні параметри
%Ri – вихідні формальні параметри
% коментар
%
<оператори>;  return
end
```

Для *фізичного* закінчення опису тіла функції можна використовувати ключове слово *end*, або просто завершення тексту файлу. Для *логічного* закінчення – використовується ключове слово *return*.

Усі поіменовані дані, які можуть використовуватись у тілі функції, є *локальними* або *формальними вхідними | вихідними параметрами*. Усі вихідні параметри мають бути визначені в тілі функції. Якщо деяка змінна *y* є вхідним і вихідним параметром, то вона повинна фігурувати в обох списках параметрів. Сукупність вхідних параметрів може бути порожньою – у цьому випадку можна опустити й дужки ( ). У випадку порожньої сукупності вихідних параметрів опускають і знак = . Якщо вихідний параметр один, то можна опустити [ ].

Інформацію, розміщену в тексті коментаря М-функції, можна відобразити командою *help ff*, а весь текст М-функції – надрукувати командою *type ff*. Перший рядок коментаря (при належному оформленні!) має певний зміст. Він використовується командою *lookfor ff* для пошуку функції за назвою *ff*.

При програмуванні функцій буває корисно використовувати вбудовані функції:
`nargin` – кількість вхідних параметрів поточної функції;
`nargin(fun)` – кількість вхідних параметрів функції з ім'ям `fun`;
`nargout` – кількість вихідних параметрів поточної функції;
`nargout(fun)` – кількість вихідних параметрів функції з ім'ям `fun`.

Для створення змінної кількості вхідних | вихідних параметрів використовується ключове слово `varargin | varargout`, яке записують після позиційних вхідних | вихідних параметрів :

```
function [R1, R2, varargout] = fun(x1, x2, varargin)
```

або взагалі без позиційних вхідних | вихідних параметрів :

```
function [varargout] = fun(varargin)
```

У тілі функції `fun` до `varargin | varargout` можна звертатись як до масиву комірок, наприклад, передавати `varargin{:}` при зверненні до іншої функції.

У випадку визначення простої функції від аргументів `x1,x2,…,xn`, тобто

такої, значення якої обчислюється у вигляді деякого виразу *Expr(x1,x2,…,xn)*, опис можна подати так:

а) за допомогою дескриптора *@* (інша назва *анонімна* функція)

```
ff = @(x1,x2,…,xn)(Expr(x1,x2,…,xn));
```

б) за допомогою команди *inline* (*inline*-функція)

```
ff = inline('Expr(x1,x2,…,xn)');
```

Зазначимо, що при описі обчислювальної частини функції (в тілі, у виразі *Expr*) потрібно завжди записувати крапку для вказівки поелементних арифметичних операцій. Крапку можна не ставити між числом (константою) і змінною у випадку запису арифметичної операції.

Для збереження інформації між викликами даної М-функції використовується атрибут *persistent*. Наприклад, *persistent t1 t2 t3*.

При описі функції в М-файлі допускаються такі можливості :

А) після опису *основної* функції *pl6_1* (саме вона і тільки вона доступна ззовні), можуть бути розташовані *підфункції,* опис яких починається ключовим словом *function*, а закінченням тіла є кінець файла *pl6_1.m* або останній оператор перед початком опису нової підфункції або ключове слово *end*.

Продемонструємо схему опису і використання підфункцій.

```
function pl6_1(varargin)     % основна функція pl6_1 (файл pl6_1.m)
   n = length(varargin);
   if n == 1
      e = varargin{1};
   elseif n == 2
      e = f1(varargin{2});
   elseif n == 3
      e = f2(varargin{3});
   else
      [e, es] = f(varargin{4:n});
      es
   end
   e
end      % кінець опису pl6_1 (необов'язковий, якщо відсутні вкладені функції)

function [rf, varargout] = f(varargin) % підфункція f
   rf = [varargin{:}];
   varargout = {rf};
   rf = sum(rf);
end      % кінець опису f (необов'язковий, якщо відсутні вкладені функції)

function rf1 = f1(a)                   % підфункція  f1
   rf1 = a;
end      % кінець опису f1 (необов'язковий, якщо відсутні вкладені функції)

function rf2 = f2(a)                   % підфункція  f2
   rf2 = a;
end      % кінець опису f2 (необов'язковий, якщо відсутні вкладені функції)
```

Виклик в основній програмі має вигляд:

```
M = {1, 22, 33, 444, 555, 666};
pl6_1(M{1}); pl6_1(M{1:2}); pl6_1(M{1:3}); pl6_1(M{:})
```

Виклик підфункцій (*f1*, *f2*, *f*) можливий тільки з основної функції *pl6_1* або підфункцій, розміщених у даному файлі *pl6_1.m* .

Б) в описі *основної* функції можуть бути розташовані (в довільному місці) *вкладені*

функції, опис яких починається ключовим словом `function`, а закінчення тіла обов'язково фіксується ключовим словом `end`:

```
function e = p16_2(a, b, c) % основна функція p16_2 (файл p16_2.m)
   function rf = f(x)
      rf = f1(c) + x;
      function rf1 = f1(x)
         rf1 = f2(x) + b;
      end                  % кінець опису rf1, наявність обов'язкова!
      function rf2 = f2(x)
         rf2 = a + x;
      end                  % кінець опису rf2, наявність обов'язкова!
   end                     % кінець опису rf, наявність обов'язкова!
   e = f(5);
end                        % кінець опису p16_2, наявність обов'язкова!
```

Виклик в основній програмі має вигляд :

```
p16_2(1, 2, 3)
```

Використовувати (поєднувати) обидва ці варіанти в одній М-функції допускається тільки при умові, що останнім оператором основної функції і підфункцій обов'язково є `end`.

Локальні та глобальні змінні. *Локальними* називаються змінні, які використовуються у даній основній функції, підфункціях або у вкладених функціях, причому область видимості обмежується тілом основної функції, окремим тілом кожної підфункції або поширюється на всі вкладені функції.

Глобальними називаються змінні, оголошені за допомогою атрибута `global`, наприклад,

```
global g1 g2 g3
```

Означення `global` має бути розміщено до першого використання відповідних змінних у тілі функції або в робочому просторі.

Виклик функції відбувається подібно іншим мовам програмування :

```
[F1 F2 … Fn] = ff(e1, e2, …, en);
ff(e1, e2, …, en);
```

Тут через `F1 F2 … Fn` позначено змінні для отримання результатів, а `e1, e2, …, en` – вирази, які є *фактичними* параметрами.

При першому зверненні до зовнішньої функції система MATLAB виконує її пошук за іменем М-файла на диску. Після того, як функцію знайдено, виконується її синтаксичний аналіз. У випадку знайденої помилки виводиться повідомлення в командне вікно, а при успішному завершенні аналізу створюється псевдокод, який завантажується в робочий простір і виконується. При повторному зверненні до функції вона буде знайдена в робочому просторі, а отже синтаксичний аналіз не проводиться і функція виконується скоріше. Видалити псевдокод із робочого простору можна командою

```
clear  ім'я_функції
```

Допускається використання функцій і змінних з однаковими іменами. Але тоді потрібно враховувати наступний порядок вибору системою MATLAB таких імен :

1. ім’я розшукується в поточному робочому середовищі (змінна з цим ім’ям, вкладена функція, анонімна функція або *inline*-функція);
2. розшукується підфункція в поточному М–файлі;
3. розшукується *приватна* функція. Приватні функції розміщені в директорії з ім’ям `private` і доступні тільки із функцій, розміщених у ній або у її батьківській директорії. Ця директорія не вказується в шляху доступу;
4. розшукується вбудована функція MATLAB;
5. виконується пошук М-функції в поточній директорії і шляхах доступу MATLAB.

Для визначення статусу імені *name* (змінної, функції) потрібно використати функцію *exist('name')*. Вона повертає, зокрема, такі числові значення:

- 0 – ім’я відсутнє;
- 1 – ім’я використовується як змінна робочого середовища;
- 2 – функція розташована в шляху пошуку файлів;
- 5 – функція є вбудованою (приватні функції і підфункції не ідентифікуються!);
- 7 – ім’я є директорією.

Повний перелік можливостей цієї функції можна отримати за допомогою команди *help exist*.

MATLAB має можливості використовувати імена функцій як аргументи (*процедурний тип* даних). При описі функції такі імена записуються звичайним способом, а при виклику – до фактичного імені додається дескриптор (*@*) або записується ім’я у вигляді текстової константи, наприклад, опис деякої функції *f*:

```
function […] = f(…,fun,…)
```

Тоді

```
e=Expr(fun(…),…);
```

– виклик функції *fun* (…) у складі виразу *Expr*;

```
E=Expr1(f(…,@myfun,…),…);
```

або

```
E=Expr1(f(…,'myfun',…),…);
```

– виклик функції *f* у складі виразу *Expr1*, де в якості фактичного параметру, який відповідає *fun*, використовуємо ім’я *myfun*:

Довільна функція MATLAB, наприклад, з ім’ям *myfun*, крім звичайного виклику *myfun(x1,x2,…,xn)*, може бути викликана вбудованою функцією *feval*. Перший фактичний вхідний аргумент має бути ім’ям *myfun* (у вигляді *'myfun'* або *@myfun*), а наступні фактичні вхідні аргументи – це перелік *x1,x2,…,xn* (через кому), наприклад,

```
a = myfun(x, y, z);
a = feval('myfun', x, y, z);
a = feval(@myfun, x, y, z);
```

Зазначимо відмінність функцій *eval* і *feval*. Перша приймає рядок з записаним виразом, який необхідно обчислити, а друга – ім’я функції у вигляді рядка, значення якої необхідно обчислити.

Приклад 16. Для кращого розуміння особливостей використання анонімних і *inline*-функцій при зверненні до функції *feval*, а також у тілі зовнішньої М-функції і у тілі М-файла, розглянемо М-функцію *pl6_3a* і М-файл *pl6_3b*, тіло яких

складають однакові команди, проте робота деяких з цих команд відбувається по різному. Потрібно звернути увагу на пріоритет використання імен функцій в М-функції і М-файлі, а також на форму використання параметра-функції в підфункції *P1* і в зовнішній функції *P1*. В коментарях операторів указано, що форма запису деяких з них має недопустимий синтаксис (можливо в даній версії MATLABa), а деякі з операторів виконують звернення до зовнішніх функції, при наявності співпадаючих імен в робочій області. Підсумовуючи, можна зробити висновок про необхідність ретельного тестування програм, які створює користувач, особливо при використанні внутрішніх і зовнішніх функцій з співпадаючими іменами.

```
function pl6_3a      % зовнішня функція
clear all, close all, clc
f1 = @(x)(x.*x);      % анонімна функція
f2 = inline('2.*x');  % inline-функція
f1(5), feval(f1, 5)

% для виклику у вигляді feval(@f1, 5) - синтаксична помилка !
feval('f1', 5)        % у директорії пошуку є М-функція f1.m
feval(@sin, pi), feval('sin', pi)
f2(5), feval(f2, 5)

% для виклику в виді feval(@f2, 5) - синтаксична помилка !
feval('f2', 5)        % у директорії пошуку є М-функція f2.m
P1(f1, 1)             % виклик підфункції
%  P1(@f1, 2)  -  синтаксична помилка !
%  P1('f1', 3) -  виклик зі зверненням до М-функції f1.m !
P1(f2, 1)             % виклик підфункції
%  P1(@f2, 2)% -  синтаксична помилка !
%  P1('f2', 3) -  виклик зі зверненням до М-функції f2.m !
end

function e = P1(f,x)    %  опис  підфункції
   e = f(x);            % виклик функції, заданої вхідним аргументом
end


%% М-файл pl6_3b.m
clear all, close all, clc
f1 = @(x)(x.*x);      % анонімна функція
f2 = inline('2.*x');  % inline-функція
f1(5), feval(f1, 5)

% для виклику у вигляді feval(@f1, 5) - синтаксична помилка !
feval('f1', 5)        % у директорії пошуку є М-функція f1.m
feval(@sin, pi), feval('sin', pi)
f2(5), feval(f2, 5)

% для виклику у вигляді feval(@f2, 5) - синтаксична помилка !
feval('f2', 5)        % у директорії пошуку є М-функція f2.m
%  P1(f1, 1)   -  помилка при виклику зовнішньої функції
%  P1(@f1, 2)  -  синтаксична помилка !
%  P1('f1', 3) -  виклик зі зверненням до М-функція f1.m !
P1(f2, 1)             % виклик зовнішньої функції
%  P1(@f2, 2)  -  синтаксична помилка !
%  P1('f2', 3) -  виклик зі зверненням до М-функція f2.m !

function e = P1(f,x)    %  зовнішня функція
   e = f(x);   % виклик функції, заданої вхідним аргументом
```

```
end
```

Для опису *рекурсивної* функції (яка прямо або побічно викликає сама себе) в MATLAB відсутні якісь спеціальні позначення рекурсивності і її виклик записується звичайним чином.

Приклад №17. Для рівняння $x = \sqrt[5]{x^6/3 + 0.25}$ знайти наближений розв'язок, використовуючи ітераційний метод Ейткена-Стеффенсена (написати відповідну зовнішню М-функцію).

Маємо типову ситуацію, при якій необхідно реалізувати діалогове введення даних для пошуку області розташування коренів рівняння $f(x) \equiv x - \sqrt[5]{x^6/3 + 0.25} = 0$, а потім реалізувати вказаний ітераційний метод для рівняння виду $x = \varphi(x)$, де $\varphi(x) = \sqrt[5]{x^6/3 + 0.25}$. При реалізації метода Ейткена-Стеффенсена у вигляді зовнішньої функції (один раз написали і в подальшому можемо її використовувати!) потрібно продумати її інтерфейс, який задовольнить найвибагливішого користувача (з огляду набору вхідних і вихідних параметрів). Також потрібно подбати і про можливі помилки при написанні виклику цієї функції.

Нижче наведено текст програми і результати її роботи.

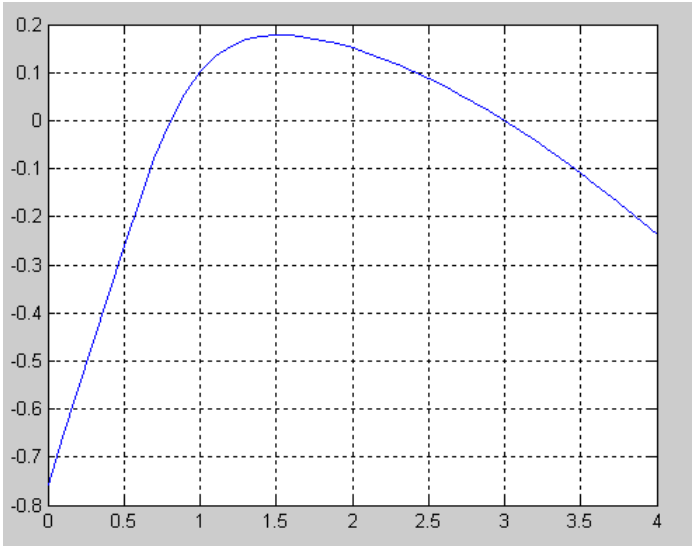
```
function pl6_4
%%PL6_4 Приклади опису і виклику функцій
clear all, close all, clc
x0 = search_region_root(@f);                % виклик підфункції
disp('Обчислення кореня методом Ейткена-Стеффенсена');
% 1) виклик зовнішньої функції з "стандартним" набором
%    вихідних і вхідних параметрів
ep = 1e-8; x = x0; [x, it] = m_estef1(@fi, x, ep, 500);
disp('Виклик:  [x, it] = m_estef1(@fi, x, ep, 500)')
disp(strcat('Корінь=', num2str(x), ' ітерацій=', int2str(it),...
            ' точність=', num2str(ep)))
% 2) виклик зовнішньої функції з "стандартним" набором
%    вихідних і скороченим вхідних параметрів
x = x0; [x, it] = m_estef1(@fi, x, ep);
disp('Виклик:  [x, it] = m_estef1(@fi, x, ep)')
disp(strcat('Корінь=', num2str(x), ' ітерацій=', int2str(it),...
            ' точність=', num2str(ep)))
% 3) виклик зовнішньої функції з мінімальним набором
%    вихідних і вхідних параметрів
x = x0; x = m_estef1(@fi, x);
disp('Виклик:  x = m_estef1(@fi, x)')
disp(strcat('Корінь=', num2str(x)))
% 4) виклик зовнішньої функції без вихідних
%    і з мінімальним набором вхідних параметрів
x = x0; m_estef1(@fi, x);
disp('Виклик:  m_estef1(@fi, x)')
disp(strcat('Корінь=', num2str(ans)))
% 5) виклик зовнішньої функції з "розширеним" набором
%    вихідних і мінімальним набором вхідних параметрів
x = x0*10; [x, it, e_tol, er, ou] = m_estef1(@fi, x);
disp('Виклик:  [x, it, e_tol, er, ou] = m_estef1(@fi, x)')
disp(strcat('Корінь=', num2str(x), ' ітерацій=', int2str(it),...
            ' точність=', num2str(e_tol),...
            ' Error=', int2str(er), ' ou=', int2str(ou)))
end
```

```
function x0 = search_region_root(f)        % підфункція
% Графічний пошук області існування кореня рівняння f(x)=0
% і завдання початкового наближення до кореня x0.
e = 1;
while e
   a = input('? a='); b = input('? b='); h = input('? h=');
   X = [a : h : b]; plot(X, f(X)), grid on
   e = input('? продовжимо дослідження областей з коренями (1-так|0-ні) =');
end
x0 = input('? Початкове наближення=');
end

function e = fi(x);                        % підфункція
% завдання правої частини рівняння x = fi(x).
e = (x .^ 6 / 3.0  + 0.25).^0.2;
end

function e = f(x);                         % підфункція
% завдання функції f(x) = x-fi(x)
e = x - fi(x);
end

function [x, varargout] = m_estef1(f, x, varargin)
%%M_ESTEF1 м.Ейткена-Стеффенсена обчислення кореня x = f(x).
%  Метод Ейткена-Стеффенсена знаходження кореня
%  рівняння виду x = f(x).
% Вхідні параметри :
%  f - функція від скалярної змінної f(x);
%  x - початкове наближення;
%  varargin - список змінних параметрів:
%               e - точність (за замовчуванням =eps);
%             mki - max кількість ітерацій (mki=1000);
% Вихідні параметри :
%  x - останнє значення наближення;
%  varargout - список змінних параметрів:
%               k - кількість виконаних ітерацій;
%               e - використана точність;
%             err - true (|x(k)-x(k-1)|>e або k>mki
%                   false (помилка відсутня);
%          решта записаних вихідних параметрів = NaN.

% перевірка кількості вхідних параметрів
varargout = {};
if nargin < 2
    error('M_ESTEF1: Мало вхідних параметрів !');
    return
end
k = length(varargin);
switch k            % визначення вхідних змінних параметрів
    case 0
        e = eps; mki = 1000;
    case 1
        e = varargin{1}; mki = 1000;
    otherwise       % k >= 2
        e = varargin{1}; mki = varargin{2};
end
% перевірка вхідних даних e і mki
if 0 >= e | e >= 1 | mki < 1
```

```
        err = true;  return
    end

    % реалізація ітераційного процесу
    k = 0; err = true;
    while k < mki
        x1 = f(x); x2 = f(x1); o = x - x1;
        o = o .* o ./(x - 2 .* x1 + x2);
        x = x1 - o; k = k + 3;
        if abs(o) < e
            err = false; break
        end
    end
    % визначення вихідних змінних параметрів
    if nargout > 1
        varargout(1) = {k};      % кількість ітерацій
        if nargout > 2
            varargout(2) = {e};  % точність
            if nargout > 3
                varargout(3) = {err};
                if nargout > 4
                    varargout(4 : nargout) = {NaN};
                end
            end
        end
    end
```

Результат виконання М-функції *p16_4*:

```
? a=0
? b=4
? h=0.1
```




```
? продовжимо дослідження областей з коренями (1-да | 0-ні) =0
? Початкове наближення=0.7
Обчислення кореня методом Ейткена-Стеффенсена
Виклик:  [x, it] = m_estef1(@fi, x, ep, 500)
Корінь=0.80686 ітерацій=12 точність=1e-008
Виклик:  [x, it] = m_estef1(@fi, x, ep)
Корінь=0.80686 ітерацій=12 точність=1e-008
Виклик:  x = m_estef1(@fi, x)
```

```
Корінь=NaN
Виклик:  m_estef1(@fi, x)
Корінь=NaN
Виклик:  [x, it, e_tol, er, ou] = m_estef1(@fi, x)
Корінь=NaN ітерацій=1002 точність=2.2204e-016 Error=1 ou=NaN
```

ЛЕКЦІЯ № 07

7.1. ПОБУДОВА 2-D, 3-D ГРАФІКІВ

Побудова графіків функцій

Одна з переваг системи MATLAB – широкий спектр засобів графіки: від команд побудови простих графіків функцій однієї змінної в декартовій системі координат і до комбінованих презентаційних графіків з елементами анімації.

1. "Елементарна" графіка

Двовимірні графіки

plot(x,y,'кмс')	Графік в лінійному масштабі, *x,y* – координати точок; коди: *к* – кольору, *м* – форми маркера, *с* – стилю лінії
loglog	Графік в логарифмічному масштабі (аналогічно *plot*)
semilogx	Графік в напівлогарифмічному масштабі по осі x (аналогічно *plot*)
semilogy	Графік в напівлогарифмічному масштабі по осі y (аналогічно *plot*)
polar	Графік в полярних координатах (аналогічно *plot*)

Символи управління кольором, стилем лінії, типом маркера					
1-й символ		**2-й символ**		**3-й символ**	
колір лінії		*форма маркера*		*стиль лінії*	
y	yellow (жовтий)	Пробіл	відсутність маркера		відсутність лінії
m	magenta (малиновий)	.		–	суцільна
c	cyan (бірюзовий)	*O*		:	пунктирна
r	red (червоний)	*X*		–.	штрих-пунктирна
g	green (зелений)	+		– –	штрихова
b	blue (синій)	*			
w	white (білий)	*S*			
k	black (чорний)	*D*			
		V			
		^			
		<			
		>			
		P			
		H			

При побудові графіків використовуються додаткові команди

axis([xl,xr,yl,yr])	Масштабування і виведення осей координат
grid on|off	Управління відображенням сітки
title('текст')	Розміщення тексту над графіком (зверху)
legend()	Опис (пояснення) елементів графіка
text()	Розміщення тексту на графіку
xlabel('текст') (*ylabel('текст')*)	Розміщення тексту вздовж *X*-осі (*Y*-осі)
subplot(n,i,j)	Розбиття графічного вікна на *n* частин і позиціювання розміщення графіка в *i, j* комірці

Нижче наведено програму побудови графіків трьох функцій з різним стилем

зображення кожної з них.

```
x=-2*pi:0.1*pi:2*pi;
yl=sin(x);
y2=sin(x).^2;
y3=x/6;
plot(x,yl,'-k',x,y2,'-.+r',x,y3,'--ob')
axis([-6 6 -1 1])
```

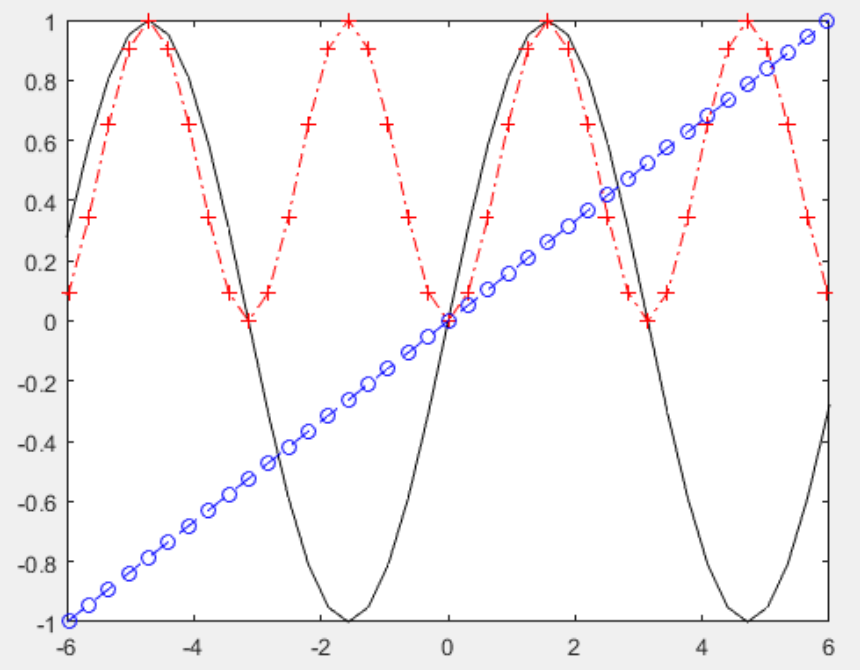


Рис. 7.1. Графіки функцій *y=sinx, y=sin²x, y=x* на відрізку [-6, 6]

Знайти для рівняння $\frac{\partial^2 u(x,t)}{\partial t^2} = a^2 \frac{\partial^2 u(x,t)}{\partial x^2}$ розв'язок задачі Коші $u(x,0) = f_1(x)$, де $a = 2$, $f_1(x) = 0.5(|x-1| + |x+1|) - |x|$ і зобразити цей роз'язок («біжучу хвилю»).

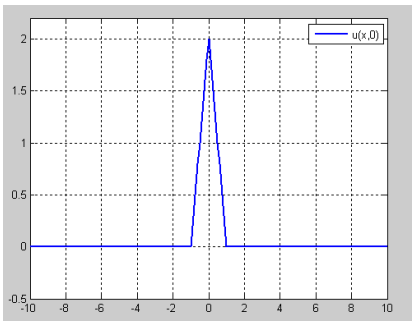
Програма :

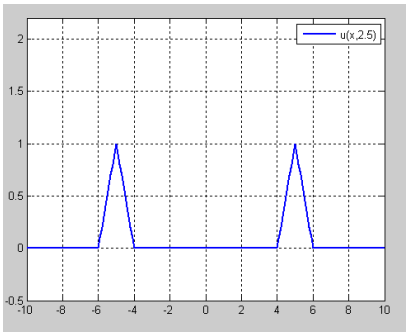
```
function f = F1(x)
% Профіль струни в початковий момент часу (t=0)
   f = 0.5*(abs(x+1)+abs(x-1))-abs(x);
end
```

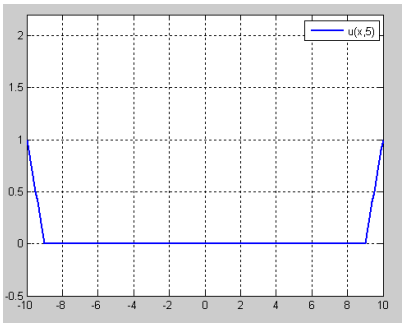
```
%% pl7_1n.m
% Побудова графіка бігучої хвилі по нескінченній струні
clear all, close all, clc
% знаходимо загальний розв'язок одновимірного хвильового рівняння :
maple('eq:=pdsolve(diff(u(x,t),t,t)-a^2*diff(u(x,t),x,x)=0, u(x,t))');
maple 'q2:=subs([_F1=F1,_F2=F1, a=2],eq)'; % виконуємо підстановки
u = maple('rhs(q2)');                % u - рядкового типу
U = @(x,t)eval(u);                   % eval - обчислює рядковий вираз
xb = -10; xe = 10; Nx = 201;
x = linspace(xb,xe,Nx);              % задаємо сітку по x
tb = 0; te = 5; Nt = 51;
T = linspace(tb,te,Nt);              % задаємо сітку по t
save pl7_1n Nx Nt                    % запис Nx,Nt в *.mat файл
fp = fopen('pl7_1n.bin','wb');       % відкриття бінарного файлу
n = fwrite(fp,x,'double');           % збереження x для експорту
disp(n==Nx);
```

```
n = fwrite(fp,T,'double');          % збереження T для експорту
disp(n==Nt);
SNi1 = int2str(Nt/2); SNi2 = int2str(Nt);  % номера стоп-кадрів
for i = 1:Nt                        % номер кадра анімації
   t = T(i); y = U(x,t);            % числовий розв'язок в точці t
   n = fwrite(fp,y,'double');       % збереження u(:,t) для експорту
   plot(x,y,'LineWidth',2); axis([x(1),x(end), -0.5, 2.2]), grid on
   legend(strcat('u(x,',num2str(t),')'))
   M(i) = getframe;                 % зберігаємо графік в масиві кадрів
   switch int2str(i)                % стоп-кадр для t = tb,(tb+te)/2, te
       case {'1',SNi1,SNi2}
           pause
   end
end
fclose(fp);                         % закриття бінарного файлу
movie(M);                           % анімація
```

Отримані «стоп-кадри» :







Тривимірні графіки

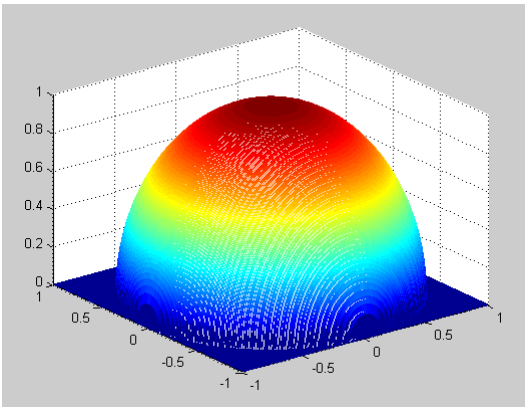
plot3(x,y,z)	Побудова ліній і точок в тривимірному просторі
contour	Зображення ліній рівня для тривимірної поверхні
contourc	Формування масиву опису ліній рівня
contour3	Зображення тривимірних ліній рівня
mesh(X,Y,Z)	Тривимірна сітчаста поверхня
meshc	Тривимірна сітчаста поверхня з проекцією ліній постійного рівня
meshz	Тривимірна сітчаста поверхня з площиною відліку на нульовому рівні
surf	Затінена сітчаста поверхня
surfc	Затінена сітчаста поверхня з проекцією ліній постійного рівня
surfl	Затінена сітчаста поверхня з підсвічуванням

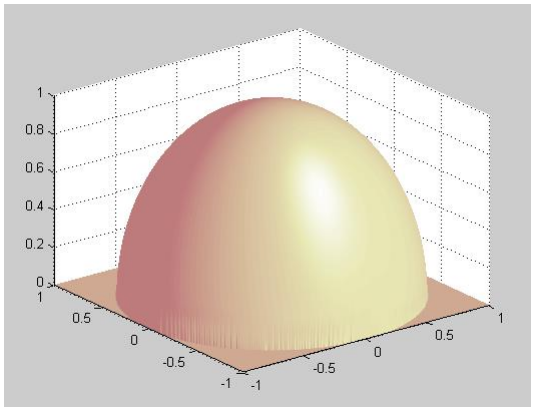
Побудувати графік функції $z(x,y)=\begin{cases}\sqrt{1-x^2-y^2}, & x^2+y^2\le 1;\\ 0, & x^2+y^2>1.\end{cases}$, де $x,y\in[-1,1]$.

Програма :

```
%%pl7_2.m
% Побудова 3-D графіка
clear all, close all, clc

[X Y] = meshgrid(linspace(-1,1,251));
XY = X.^2+Y.^2;                  % обчислення допоміжної функції
C = XY<=1;                       % логічний масив
XYZ = zeros(size(XY));           % нульова матриця
XYZ(C) = 1 - XY(C);              % логічне індексування
```

```
Z = sqrt(XYZ);                    % обчислення функції z(x,y)
mesh(X,Y,Z), pause, close all
surfl(X,Y,Z), shading interp, colormap pink
```





Крім цих функцій, є багато додаткових функцій, що реалізують *управління кольором, завдання палітри кольорів, управління підсвічуванням, управління кутом перегляду, створення "твердої" копії та збереження графіка* тощо.

2. "Спеціальна" графіка

Двовимірні графіки

area	Зафарбування областей графіка
bar	Стовпчикова діаграма
barh	Стовпчикова діаграма з горизонтальним розташуванням
comet	Рух точки по траєкторії
compass	Графік векторів-стрілок, які виходять із початку координат
errorbar	Графік із зазначенням інтервалу похибки
ezplot	Побудова графіків з використанням діалогового вікна
feather	Графік векторів-стрілок, які виходять із рівновіддалених точок горизонтальної осі
fill	Зафарбування многокутника
hist	Побудова гістограми
pareto	Діаграма Парето
pie	Кругова діаграма
plotmatrix	Графік матриці
quiver	Графік поля напрямків
ribbon	Зображення ліній на тривимірному графіку
stairs	Ступінчастий графік
stem	Графік дискретних значень

В програмах `pl7_3.m`, `pl7_4.m` (тексти розміщено в кінці лекції) розглянуто приклади використання деяких функцій "спеціальної" 2-D і відповідно 3-D графіки.

Тривимірні графіки

bar3	Тривимірна стовпчаста діаграма
bar3h	Тривимірна стовпчикова діаграма з горизонтальним розташуванням
comet3	Рух точки по траєкторії в тривимірному просторі
contourf	Графік ліній рівня із зафарбованими областями

fill3	Зафарбування багатокутника в тривимірному просторі
pie3	Секторна діаграма
quiver3	Графік поля напрямків у тривимірному просторі
slice	Перетин функцій від трьох змінних
stem3	Графік дискретних значень у тривимірному просторі
trimesh	Тривимірна поверхня з трикутними комірками
trisurf	Тривимірна сітчаста поверхня з трикутними комірками
waterfall	Тривимірна поверхня без відображення ребер сітки

А ще доступні: *робота з графічними образами, анімаційні можливості, об'ємні графічні об'єкти, дескрипторна графіка, створення і управління осями координат, об'єкти дескрипторної графіки, операції над графічними об'єктами, графічний інтерфейс користувача (GUI), засоби проектування GUI, діалогові панелі, створення меню і кнопок інструментальної панелі і т. д.*

7.2. РОБОТА З ФАЙЛАМИ

Використання бінарних і текстових (форматованих) файлів для введення, виведення даних

1. Відкриття і закриття файлу

fp=fopen(namef,mode)	Зв'язати змінну *fp* з файлом, що має ім'я *namef* на зовнішньому пристрої та відкрити його для обробки згідно з режимом *mode*. Рядок *mode* може мати значення: '*r*' - читання (існуючого); '*w*' – запис (у новий); '*a*' – доповнення існуючого. Наявність '+' означає можливість заміни записів у файлі (прямий доступ). Значення '*b*' - бінарний, '*t*'' – текстовий файл.
fclose(fp)	Закрити файл *fp*.

2. Бінарні файли

[A,count]= *fread(fp,size,precssion)*	Виконує читання в масив *A* даних бінарного файла, виозначеного змінною *fp*. Аргумент *count* отримує число успішно прочитаних елементів. Допустимі значення size: *N* = читається *N* елементів в вектор-стовпець; *Inf* = читається до кінця файла; [*M,N*] = читаються елементи і записуються по стовпцях у матрицю розмірності *M*N* (допустимо задавати кількість стовпців *N* як *inf*). Якщо аргумент *size* опущено, то виконується читання усіх даних. Аргумент *precssion* визначає кількість біт для представлення даних: MATLAB C or Fortran Description *'uchar' 'unsigned char' unsigned integer, 8 bits.* *'schar' 'signed char' signed integer, 8 bits.* *'int8' 'integer*1' integer, 8 bits.* *'int16' 'integer*2' integer, 16 bits.* *'int32' 'integer*4' integer, 32 bits.* *'int64' 'integer*8' integer, 64 bits.* *'uint8' 'integer*1' unsigned integer, 8 bits.* *'uint16' 'integer*2' unsigned integer, 16 bits.* *'uint32' 'integer*4' unsigned integer, 32 bits.* *'uint64' 'integer*8' unsigned integer, 64 bits.* *'single' 'real*4' floating point, 32 bits.* *'float32' 'real*4' floating point, 32 bits.* *'double' 'real*8' floating point, 64 bits.* *'float64' 'real*8' floating point, 64 bits.* Якщо аргумент *precssion* опущено, то *'uint8'*
count=	Записує елементи матриці *A* у файл *fp*. Дані записуються по стовпцях. *count* – лічильник успішно записаних даних.

fwrite(fp,A,precssion)	*fp* – ціле, файлова змінна при відкритті файла, або 1 для стандартного файла виведення, або 2 – для файла помилок. *precssion* визначає розмір і точність даних (як у *fread*).

3. Позиціонування файла

[message,errnum]=ferror(fp)	Отримати інформацію про помилку *I/O*.
st=feof(fp)	Чи досягнуто (st = 1) кінець файла *fp*.
st=fseek(fp,offset,origin)	Встановити маркер файла *fp* у задану позицію. Маркер переміщується на *offset* байт вправо(>=0) | вліво (<0) від позиції, вказаної в *origin*, яка має значення : *'bof'* або -1 початок файла; *'cof'* або 0 поточна позиція; *'eof'* або 1 кінець файла.
pos=ftell(fp)	Визначити в байтах позицію *pos* маркера від початку файла *fp*. При pos=-1 – помилка.
frewind(fp)	Встановити маркер у початок файла *fp*.

4. Текстові (форматовані) файли

[A,count]= *fscanf(fp,format,size)*	Прочитати в матрицю *A* данні із файла *fp* під управлінням *format*. Значення *count*, *size* аналогічні оператору *fread*. Рядок *format* може містити пробіли, звичайні символи або символи специфікаторів перетворення (використання як у мові С): %*d* - знакове ціле; %*u* – ціле; %*f* – дійсне число з фіксованою крапкою; %*e* – дійсне число з плаваючою крапкою (експоненціальна форма); %*g* – дійсне число з фіксованою|плаваючою крапкою; %*s* – рядок; %*q* - рядок, обмежений подвійними лапками; %*c* - символи, включаючи символи пробілів; %[...] - найбільший рядок, з символами із множини, вказаної в [ ]; %[^...] - найбільший рядок, в який не входять символи із множини, вказаної в [ ]. Ширина поля регулюється для цілих даних (%5*d* - ціле із 5 цифр). Формати %*f*, %*e* підтримують форму %<ширина>.<точність>. Екранні символи *n*,*r*,*t*,*b*,*f* також мають звичайний смисл.
count= *fprintf(fp,format,A,…)*	Записати данні *A*, … у файл *fp* під управлінням форматів, визначених у рядку *format*. Значення аргументу *count* аналогічно оператору *fwrite*.
line=fgetl(fp)	Прочитати рядок *line* із файла *fp*, видаливши символ кінця рядка (-1 при читанні кінця файла).
line=fgets(fp)	Аналогічно *fgetl*, але символ кінця рядка включається.

Інколи корисно використовувати функцію `sprintf` – форматне перетворення даних *A, B, C*, ... у рядок символів *S* :

```
[S, errmsg] = sprintf(format, A, B, C, ...)
```

Значенням необов'язкового аргументу `errmsg` є повідомлення про помилку (за наявності), або порожній рядок, при її відсутності.

Функції `fprintf` і `sprintf` – "векторизовані", тобто, коли вхідними даними є масив, то рядок формату `format` циклічно застосовується до елементів стовпця масиву, поки не будуть вичерпані всі елементи. Наприклад,

```
x =0 : 0.1 : 1;  y = [x, exp(-x)];
f=fopen('ex1file.txt', 'w');
fprintf(f, '%6.2f%13.6f\n', y);
fclose(f);
type ex1file.txt
```

Імпорт, експорт файлів

load namef *load namef ac*p*	Прочитати змінні з (бінарного) *namef.mat* файла; Прочитати змінні з "шаблонними" іменами *ac*p* із *namef.mat* файла. При завантаженні MAT-файла, нові значення однойменних змінних будуть записані замість існуючих.
save namef *save namef a ab ac*p -append*	Записати змінні з вказаними іменами *a ab* і з "шаблонними" іменами *ac*p* у бінарний *namef.mat* файл. За наявності опції *append* можна додати вказані дані до існуючого MAT-файла.
A=csvread('namef.csv')	Прочитати числові елементи, які розділені "," з текстового файла *namef.csv* і записати в масив *A*.
csvwrite('namef.csv',A)	Записати числовий масив *A* у текстовий файл *namef.csv*, розділяючи елементи за допомогою ",".
A=dlmread('namef.csv','D')	Прочитати числові елементи, які розділені "D" з текстового файла *namef.csv* і записати в масив *A*.
dlmwrite('namef.csv',A,'D')	Записати числовий масив *A* у текстовий файл *namef.csv*, розділяючи елементи за допомогою "D".

Зазначимо, що для роботи з графічними форматами файлів використовуються функції `imread`, `imwrite`**,** звуковими файлами – `wavwrite, wavread`**.**

Збереження змінних робочої області. Команда `save` дозволяє зберегти вміст робочої області (Workspace) у бінарному MAT-файлі, який у подальшому можна завантажити командою `load`. Команда `save` також доступна як опція *Save Workspace* меню *File*. Аналогічним цілям служать і функції `save|load`, наприклад,

```
save('mfile.mat', 'v1', 'v2', 'av1', 'bv1');
load('mfile.mat', 'v1', 'v2');
```

Функція menu. Для створення меню використовується команда `menu` у вигляді:

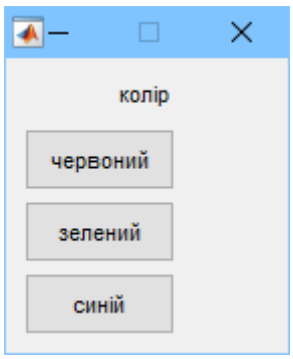
```
choice=menu('заголовок','вибір 1','вибір 2',…,'вибір n');
```

яка реалізує виведення графічного вікна меню із заголовком `заголовок` і кнопками вибору `вибір i`. Змінна `choice` отримує число 1,2,…,*n* – номер вибраної кнопки меню.

Проілюструємо роботу функції `menu` для побудови графіка деякої функції на заданому інтервалі кольором, вибраним із меню

```
k=menu ('колір','червоний', 'зелений','синій')
color=['r';'g';'b'];
x=-2:0.1:2;
y=x.*x;
plot(t,y,color(k))
```

Функція menu виводить на екран вікно у вигляді:



При натисненні на одну з кнопок із назвою кольору, функція *menu* повертає номер цієї кнопки (1,2 або 3), і цей номер стає значенням змінної *k*. Вектор-стовпець *color* містить атрибути кольору для побудови графіка. За номером *k* потрібний атрибут вибирається із масиву *color* і вказується в команді *plot*. У результаті, обравши потрібний пункт меню, одержимо графік функції, побудований вибраним кольором.

Тексти програм *pl7_3.m*, *pl7_4.m*:

```
%%p17_3.m
% Приклади використання "спеціальної" графіки
clear all, close all, clc, warning off

f1 = @(x)(9-x.*x);
f2 = @(x)((x-0.7).*(x-0.5)-5);
X  = linspace(-3,4,8)';
Y1 = f1(X); Y2 = f2(X);

stairs(X,Y1,'LineWidth',2), grid on
title('Функція STAIRS','FontSize',13), pause(3)

stem(X,Y2,'LineWidth',2), grid on
title('Функція STEM','FontSize',13), pause(3)

area(X,Y1,'FaceColor','g','EdgeColor','r','LineWidth',3)
title('Функція AREA','FontSize',13), pause(3)

area(X,Y1,'FaceColor',[.5 .9 .6],'EdgeColor','b',...
     'LineWidth',2), hold on
area(X,Y2,'FaceColor',[.9 .85 .7],'EdgeColor','y',...
     'LineWidth',2), hold off
title('Функція AREA (суперпозиція)','FontSize',13), pause(3)

bar(X,Y1), grid on
title('Функція BAR (вертикальна)','FontSize',13), pause(3)

barh(X,Y2), grid on
title('Функція BARH (горизонтальна)','FontSize',13), pause(3)

t = fix(1000.*Y1(Y1>0));
m = length(t); z = cell(1,m);
for i = 1:m
    z(i) = {strcat('name',int2str(i))};
end
pareto(t,z)
```

```
title('Функція PARETO (діаграма Парето)','FontSize',13), pause(3)

hist(Y2), grid on
title('Функція HIST','FontSize',13), pause(3)

pie(Y2)
title('Функція PIE','FontSize',13), pause(3)

compass(X,Y1)
title('Функція COMPASS','FontSize',13), pause(3)

feather(X,Y1)
title('Функція FEATHER','FontSize',13), pause(3)

X  = linspace(-3,4,1401)';
Y1 = f1(X); Y2 = f2(X);
comet(X,Y1)
title('Це виконувалась функція COMET','FontSize',13), pause(3)

I = Y2<=Y1;
t = X(I); z = Y2(I);
m = length(t); J = m-1:-1:2;
t = [t;t(J)]; z = [Y1(I);z(J)];
fill(t,z,'y')
title('Функція FILL','FontSize',13), pause(3), close all, clear all
```

%%p17_4.m

```
% Приклади використання "спеціальної" 3-D графіки
clear all, close all, clc, warning off

Z = linspace(0,2*pi,501);
u = sqrt(Z); Y = 10*Z;
X = cos(Y).*u; Y = sin(Y).*u; clear u
comet3(X,Y,Z)
title('Це виконувалась функція COMET3','FontSize',13), pause(1)

[X,Y,Z] = peaks;
waterfall(X,Y,Z)
title('Функція WATERFALL','FontSize',13), pause(3)

contourf(Z,10)
title('Функція CONTOURF (вар.1)','FontSize',13), pause(3)

[c,h] = contourf(Z); clabel(c,h), colorbar
title('Функція CONTOURF (вар.2)','FontSize',13)
pause(3), clear c h

[X,Y,Z] = peaks(11);
[U,V,W] = surfnorm(X,Y,Z);
quiver3(X,Y,Z,U,V,W)
title('Функція QUIVER3','FontSize',13), pause(3)

quiver3(X,Y,Z,U,V,W), hold on
surf(X,Y,Z), hold off
title('Функція QUIVER3+SURF','FontSize',13)
pause(3), clear U V W
```

```
[X,Y,Z] = meshgrid(linspace(-2,2,41));
V = X .* exp(-X.^2 - Y.^2 - Z.^2);
slice(X,Y,Z,V,[],[],[-0.6,1])
title('Функція SLICE','FontSize',13)
pause(3), close all, clear all
```

ЛЕКЦІЯ № 08

РОЗВ'ЯЗАННЯ ЗАДАЧІ КОШІ ДЛЯ ЗДР

Розглянемо задачу Коші для ЗДР

$$\begin{cases} \dfrac{d\vec{u}(x)}{dx} = \vec{f}(x,\vec{u}(x)), \quad x \in (x_0, x_{end}], \\ \vec{u}(x_0) = \vec{u}_0 \end{cases} \tag{8.1}$$

і для її наближеного розв'язку будемо використовувати *чисельні* методи:

a) *різницеві* (*однокрокові* і *багатокрокові*);

b) *типу Рунге-Кутта*.

Як перші, так і другі методи бувають *явними* і *неявними*.

Різницеві однокрокові методи. Розглянемо спочатку скалярний варіант (8.1). Вводимо за змінною x сітку з певним кроком (рівномірним або нерівномірним), виконуємо апроксимацію лівої і правої частини рівняння (8.1) на шаблоні $S[x_n, x_{n+1}]$.

Явний метод Ейлера. Схема методу

$$y_{n+1} = y_n + h_n \cdot f(x_n, y_n), \quad y_0 = u_0, \quad n = 0,1,\ldots. \tag{8.2}$$

Неявний метод Ейлера. Схема методу

$$y_{n+1} = y_n + h_n \cdot f(x_{n+1}, y_{n+1}), \quad y_0 = u_0, \quad n = 0,1,\ldots. \tag{8.3}$$

Симетрична схема. Схема методу

$$y_{n+1} = y_n + 0.5h_n \cdot (f(x_n, y_n) + f(x_{n+1}, y_{n+1})), \quad y_0 = u_0, \quad n = 0,1,\ldots. \tag{8.4}$$

Для автоматичного вибору кроку використовують метод *Рунге* (враховуючи, що методи (8.2)–(8.3) мають точність $O(h)$, а метод (8.4) – точність $O(h^2)$).

Методи типу Рунге-Кутта. Характерною особливістю цих методів є те, що обчислення проводяться не тільки в точках сітки, але і в деяких проміжних точках (використовуючи спеціальні квадратурні формули)

$$y_{n+1} - y_n = \int_{x_n}^{x_{n+1}} f(\xi, u(\xi))d\xi = h_n \sum_{i=1}^{m} A_i \cdot K_i, \quad y_0 = u_0, \quad n = 0,1,\ldots.$$

Сім'я методів Рунге-Кутта **2-го порядку точності** (m=2). Схема методу

$$\begin{aligned} &A_1 = 1-\gamma, \quad A_2 = \gamma, \quad \theta = 0.5h_n/\gamma, \\ &K_1 = f(x_n, y_n), \quad K_2 = f(x_n + \theta, y_n + \theta K_1), \quad \gamma \in (0,1]. \end{aligned} \tag{8.5}$$

Найчастіше використовуються такі значення: $\gamma = 0.5$, $\gamma = 1$.

Методи Рунге-Кутта **4-го порядку точності** (m=4) – найпопулярніші з усіх методів. Схема методу

$$\begin{aligned} &A_1 = 1/6, \quad A_2 = 1/3, \quad A_3 = 1/3, \quad A_4 = 1/6, \\ &K_1 = f(x_n, y_n), \quad K_2 = f(x_n + 0.5h_n, y_n + 0.5h_n K_1), \\ &K_3 = f(x_n + 0.5h_n, y_n + 0.5h_n K_2), \quad K_4 = f(x_{n+1}, y_n + h_n K_3). \end{aligned} \tag{8.6}$$

Поняття жорстких систем диференціальних рівнянь

Стійкість умовна і абсолютна. Якщо розглядати модельну задачу Коші для (8.1) з $f(x,u(x)) = \lambda \cdot u(x), \ x > 0, \ \lambda < 0$, то стійкість означає виконання нерівності $|u(x+h)| \le |u(x)|, \ h > 0$.

При використанні різницевого методу, потрібно вимагати, щоб для його розв'язку виконувалась нерівність, аналогічна диференціальному розв'язку.

Означення 1. Різницевий метод є *абсолютно стійким* при $\forall \ h > 0$, і *умовно*

стійким, якщо він стійкий при певних обмеженнях на крок *h*.

Для явного методу Ейлера: $|y_{n+1}| \le |1+\lambda h||y_n|$ і маємо обмеження

$$0 < h \le \frac{2}{|\lambda|}. \tag{8.7}$$

Аналогічне обмеження на крок можна отримати для методу Рунге-Кутта 2-го порядку точності ($\gamma = 1$), а для 4-го –

$$|\lambda| \cdot h \le 2.78.$$

Для неявного методу Ейлера $|y_{n+1}| \le \left|\frac{1}{1-\lambda h}\right| |y_n|$ і завжди маємо

$$\frac{1}{1-\lambda h} < 1 \text{ при } 0 < h \,\&\, \lambda < 0.$$

Із наведених прикладів бачимо, що явні методи умовно стійкі, а неявні – безумовно стійкі, але вимагають розв'язання нелінійних рівнянь.

Розглянуті методи без зусиль переносяться на системи ЗДР, але виникає проблема з їх стійкістю (обмеження типу (8.7)), пов'язана з різномасштабністю процесів, які описуються системою ЗДР. Проілюструємо це на простому прикладі системи двох рівнянь

$$\begin{cases} \frac{du_1(x)}{dx} + a_1 u_1(x) = 0, \\ \frac{du_2(x)}{dx} + a_2 u_2(x) = 0, \end{cases}$$
$$x > 0; \;\; a_{1,2} > 0, \;\; a_2 >> a_1$$

або записаній у векторній формі

$$\frac{d\vec{u}(x)}{dx} = A \cdot \vec{u}(x), \;\; A = \begin{pmatrix} -a_1 & 0 \\ 0 & -a_2 \end{pmatrix}, \;\; \vec{u}(x) = \begin{pmatrix} u_1(x) \\ u_2(x) \end{pmatrix}. \tag{8.8}$$

Ураховуючи, що $a_{1,2} > 0, \;\; a_2 >> a_1$, з деякого x ми повинні мати (аналітично) $\vec{u}(x) = \begin{pmatrix} u_1(x) \\ u_2(x) \end{pmatrix} \approx \begin{pmatrix} u_1(x) \\ 0 \end{pmatrix}$.

Проте чисельно, якщо використовувати явний метод Ейлера, ми маємо обмеження на крок у вигляді нерівності (8.7), що приводить до $h \cdot a_2 \le 2$.

Означення 2. Нехай матриця A – стала, квадратна, розмірності N. Система (8.8) називається *жорсткою*, якщо:

1) $\mathrm{Re}(\lambda_k) < 0, k = 1,2,...,N$ – система асимптотично стійка за Ляпуновим;

2) $S = \dfrac{\max\limits_k |\mathrm{Re}(\lambda_k)|}{\min\limits_k |\mathrm{Re}(\lambda_k)|}$ – *число жорсткості* системи велике.

У системі MATLAB є вбудовані функції (їх часто називають *солвери – вирішувачі*), які реалізують різні методи, наприклад,

- Гіра для жорстких ЗДР – `ode15s`;
- типу Рунге-Кутта 2-го, 3-го порядку для нежорстких ЗДР – `ode23`;
- Розенброка 2-го порядку для жорстких ЗДР – `ode23s`;
- типу Рунге-Кутта 4-го, 5-го порядку для нежорстких ЗДР – `ode45`.

Решта солверів має імена `ode113`,`ode23t`,`ode23tb` і `ode15i` (для розв'язання систем виду $\vec{f}\left(x, \vec{u}(x), \frac{d\vec{u}(x)}{dx}\right) = 0$). Методика використання, з точки зору

вхідних/вихідних параметрів, всіх солверів однакова і має такий шаблон виклику:

```
[X, Y] = solver(odef, [x0, xend], Y0, options);
```

де :

- *X* – сітка розв'язку, *Y* – матриця розв'язку (стовпчики *Y*(:, k) = $u_k(X)$);
- *solver* – ім'я вбудованої функції MATLAB;
- *odef* – функція користувача (умовно позначимо її *fun*) з описом правої частини ЗДР (8.1), яка записується у вигляді дескриптора (*@fun*) або рядка з її ім'ям ('*fun*');
- *[x0,xend]* – відрізок, на якому шукаємо розв'язок ЗДР, якщо подаємо його у вигляді вектора *[x0:h:xend]* – то задаємо точки, в яких отримуємо розв'язок ЗДР (тільки для вихідних параметрів у формі *[X,Y]* !);
- *Y0* – вектор початкових значень задачі Коші ЗДР;
- *options* – необов'язковий набір параметрів управління солверами.

Інтерфейс солверів допускає звернення до них з одним вихідним аргументом:

```
sol = solver(odef, [x0, xend], Y0);
```

Такий виклик солвера приводить до утворення *структури* *sol*, в якій *sol.x* – сітка розв'язку, а *sol.y* – матриця розв'язку (рядки *sol.y*(k, :) = u_k(*sol.x*)). Крім цих полів, в *sol* міститься ще деяка інформація (див. *help* відповідного солвера).

Для реалізації сервісних операцій при роботі з солверами для ЗДР, MATLAB містить дві групи команд:

1. для підтримки опцій солвера :

odeset	створити/змінити опції солвера
odeget	отримати опції солвера

2. для формування вихідної інформації солвера в певних форматах, прийнятних для подальшої візуалізації результату:

odeplot	інформація солвера, як матриці від часу
odephas2	інформація для побудови двовимірної фазової площини
odephas3	інформація для побудови тривимірної фазової площини
odeprint	діалогове вікно виведення на друк

Розглянемо приклади застосування деяких з цих солверів.

Приклад 18. Знайти наближений розв'язок задачі Коші для системи ЗДР на відрізку [0,25] (рівняння Лотки-Вольтерри для моделювання процесу "хижак-жертва"):

$$\begin{cases} u'(t) = u*(1-\alpha*v), \\ v'(t) = v*(\beta*u-1), \\ u(0)=1, \\ v(0)=1. \end{cases}$$

Нижче наведено текст відповідної *M*-функції, результати роботи якої зображено на рис. 8.1–8.2.

```
function p18_1
%% Приклади розв'язання задачі Коші для ЗДР
% (рівняння Лотки-Вольтерри для моделювання
```

```
% процесу "хижак-жертва")

clear all, close all, clc
global alf bet

alf = 1; bet = 2.5;
Y0 = [1, 1];        % вектор початкових умов
[T, Y] = ode45(@f81, [0 25], Y0); % розв'язання системи

plot(T, Y(:,1), T, Y(:,2),'r','LineWidth',2), grid on
legend('u','v','Location','Best')
title('Модель "хижак-жертва" ( {\alpha} = 1, {\beta} = 2.5 ) ' )
xlabel('t'), ylabel('u,v'), pause, close all
plot(Y(:,1), Y(:,2), 'b','LineWidth',2), hold on

alf = 2.5; bet = 1;
[T, Y] = ode45(@f81, [0 25], Y0);
plot(Y(:,1), Y(:,2), 'r','LineWidth',2), grid on
title('Фазові траєкторії')
xlabel('u'), ylabel('v')
ch=legend('{\alpha}=1, {\beta}=2.5',...
    '{\alpha}=2.5,{\beta}=1','Location','Best');
set(ch,'Fontsize',12);


function F = f81(t, U)
% підфункція правої частини системи
    global alf bet
    F = [U(1).*(1 - alf .*U(2));...
         U(2).*(bet.*U(1) - 1)];
```

Результат виконання М-функції *p18_1.m*:

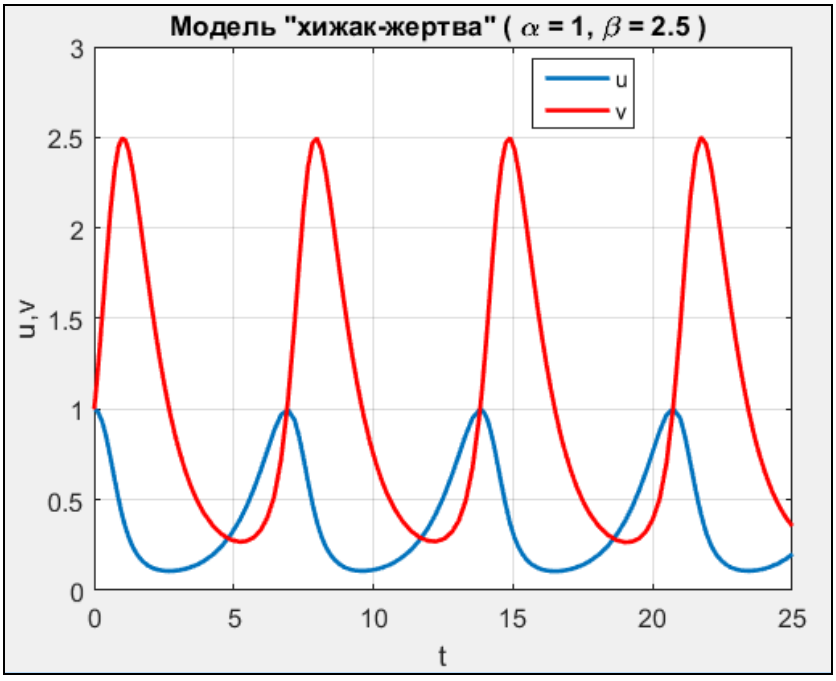


Рис. 8.1. Графік розв'язку системи задачі Коші для рівнянь Лотки-Вольтерри (параметри $\alpha = 1$; $\beta = 2.5$)

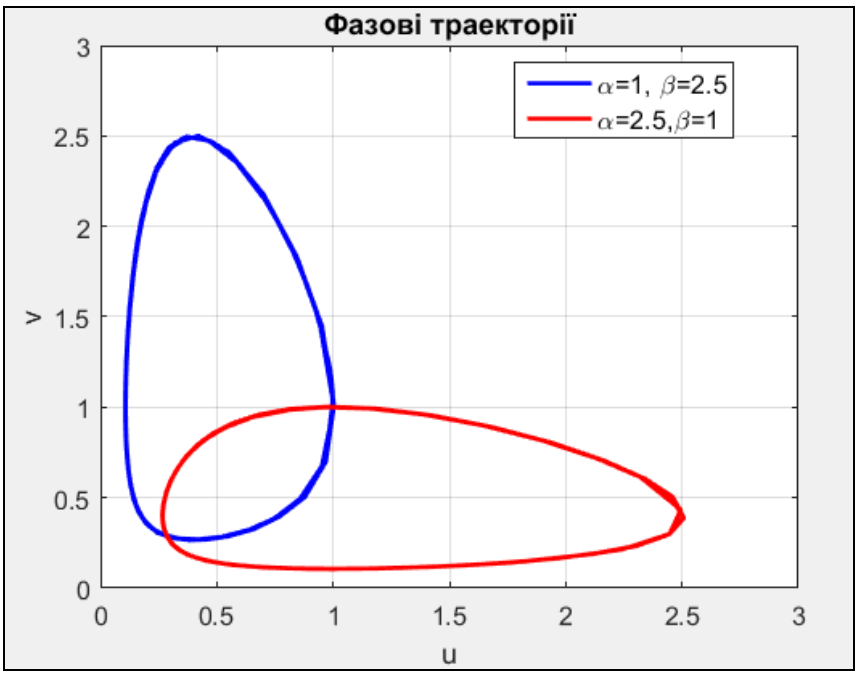


Рис. 8.2. Фазові траєкторії системи рівнянь Лотки-Вольтерри

Приклад 19. Знайти наближений розв'язок початково-крайової задачі для рівняння Кортевега-де Фріза (КдФ) з періодичними крайовими умовами

$$\begin{cases} \dfrac{\partial u(x,t)}{\partial t} + \dfrac{\partial}{\partial x}\left(\dfrac{u^2(x,t)}{2}\right) + b\dfrac{\partial^3 u(x,t)}{\partial x^3} = 0, \\ x \in [l,r],\ 0 < t \le T; \\ u(x,0) = u0(x); \\ u(l+\xi,t) = u(r+\xi,t),\ \xi \in [0, r-l] \end{cases}$$

при наступних параметрах: $b = 0.1,\ l = 0,\ \mathrm{r} = 2\pi,\ T = 30,\ u0(x) = \sin(x)$. Для цієї задачі є декілька законів збереження, наприклад,

перший – закон збереження імпульсу: $\int_l^r u(\varsigma,t)d\varsigma = \int_l^r u0(\varsigma)d\varsigma = C_1$,

другий – закон збереження енергії: $\int_l^r u^2(\varsigma,t)d\varsigma = \int_l^r u0^2(\varsigma)d\varsigma = C_2$.

Використаємо метод *прямих* (*Pome*) і наближений розв'язок цієї задачі зведемо до розв'язання задачі Коші. Для цього введемо рівномірну сітку по просторовій змінній

$$\overline{\omega}_h = \{x_k = l + (k-1)h,\ k = \overline{1,n},\ r - l = (n-1)h\},$$

а на прямих $x = x_k$ невідомі функції $U_k(t) = u(x_k,t)$. Всі диференціальні оператори в рівнянні КдФ замінимо на різницеві:

$$\frac{\partial}{\partial x}\left(\frac{u^2(x,t)}{2}\right) \approx \frac{(U_{k+1}(t) + U_k(t) + U_{k-1}(t)) * (U_{k+1}(t) - U_{k-1}(t))}{6h};$$

$$\frac{\partial^3 u(x,t)}{\partial x^3} \approx \frac{(U_{k+2}(t) - 2 * (U_{k+1}(t) - U_{k-1}(t)) - U_{k-2}(t))}{2h^3},$$

врахуємо періодичність крайових умов і отримаємо задачу Коші для системи ЗДР:

$$\begin{cases} \dfrac{d\vec{U}(t)}{dt} = \vec{F}\left(t, \vec{U}(t)\right), \quad 0 < t \le T; \\ \vec{U}(0) = u0\left(\vec{X}\right), \quad \vec{X} = [x_1, x_2, ..., x_n], \end{cases}$$

де

$$\vec{F}\left(t, \vec{U}(t)\right) = \begin{cases} f(k), & k = 1, \quad kp = 2, kpp = 3, km = n-1, kmm = n-2; \\ f(k), & k = 2, \quad kp = 3, kpp = 4, km = 1, kmm = n-1; \\ f(k), & k = \overline{3, n-2}, \quad kp = k+1, kpp = k+2, km = k-1, kmm = k-2; \\ f(k), & k = n-1, \quad kp = n, kpp = 2, km = n-2, kmm = n-3; \\ f(k), & k = n, \quad kp = 2, kpp = 3, km = n-1, kmm = n-2; \end{cases}$$

$$f(k) = A * \left(U_{kp}(t) + U_k(t) + U_{km}(t)\right) * \left(U_{kp}(t) - U_{km}(t)\right) +$$
$$+ B * \left(U_{kpp}(t) - 2 * \left(U_{kp}(t) - U_{km}(t)\right) - U_{kmm}(t)\right);$$

$$A = -1/6h, \; B = -b/2h^3.$$

Нижче наведено текст відповідного М-файла і результати його виконання (перевірка виконання закону збереження, рис. 8.3, 8.4).

```
function pl8_2
%% Приклади використання солверів для жорстких СЗДР
%  Застосування м.Роте для початково-крайової задачі
%  для рівняння Кортевега-де Фріза з періодичними
%  крайовими умовами на відрізку x Є [l,r].

clear all, close all, clc
global A B n3 n2 n1 n

% параметри крайової задачі :
T = 30; l = 0; r = 2.*pi; b = 0.1;
% завдання просторової сітки :
n = 93; x = linspace(l, r, n);
% завдання сітки отримання розв'язку :
m = 75; Tt = linspace(0, T, m);

U0 = @(x)(sin(x));    % функція початкових умов
Y0 = U0(x);           % вектор початкових умов

% допоміжні величини :
n1 = n-1; n2 = n-2; n3 = n-3; h = x(2)-x(1);
A = -1./(6.*h); B = -b./(2.*h.^3);

[t, Y] = ode23s(@pl9_2f, Tt, Y0);  % розв'язання задачі

% побудова сіток для виведення графіка розв'язку
[Mx, Mt] = meshgrid(x, t');
% виведення графіка наближеного розв'язку u(x,t)
surfl(Mx, Mt, Y)
shading interp, colormap pink
title('КЗ для рівняння Кортевега-де Фріза')
xlabel('x'), ylabel('t'),zlabel('u')
pause
contour(Mx, Mt, Y, 10)
xlabel('x'), ylabel('t')
colormap hsv
```

```
disp(strcat('Виконання 1-го закону збереження(імпульсу)=',...
            num2str(abs(h.*trapz(Y0-Y(m,:))),13)))
disp(strcat('Виконання 2-го закону збереження (енергії)=',...
            num2str(abs(h.*trapz(Y0.^2-Y(m,:).^2)),13)))
end


function F = pl8_2f(t, U)      %  опис підфункції
% функція правої частини системи
    function f = fs   % вкладена функція
        f = A.*(U(kp)+U(k)+U(km)).*(U(kp)-U(km))+ ...
            B.*(U(kpp)-2.*(U(kp)-U(km))-U(kmm));
    End

    global A B n3 n2 n1 n

    F = zeros(n, 1);
    kmm = n2; km = n1; k = 1; kp = 2; kpp = 3; F(k) = fs();
    kmm = n1; km = n;  k = 2; kp = 3; kpp = 4; F(k) = fs();
    for k = 3 : n2
        kmm = k-2; km = k-1; kp = k+1; kpp = k+2; F(k) = fs();
    end
    kmm = n3; km = n2; k = n1; kp = n; kpp = 2; F(k) = fs();
    kmm = n2; km = n1; k = n;  kp = 2; kpp = 3; F(k) = fs();
end
```

Результат виконання М-функції *pl8_2.m*:

```
Виконання 1-го закону збереження(імпульсу)=3.403969715368e-006
Виконання 2-го закону збереження (енергії)=3.357457026136e-005
```

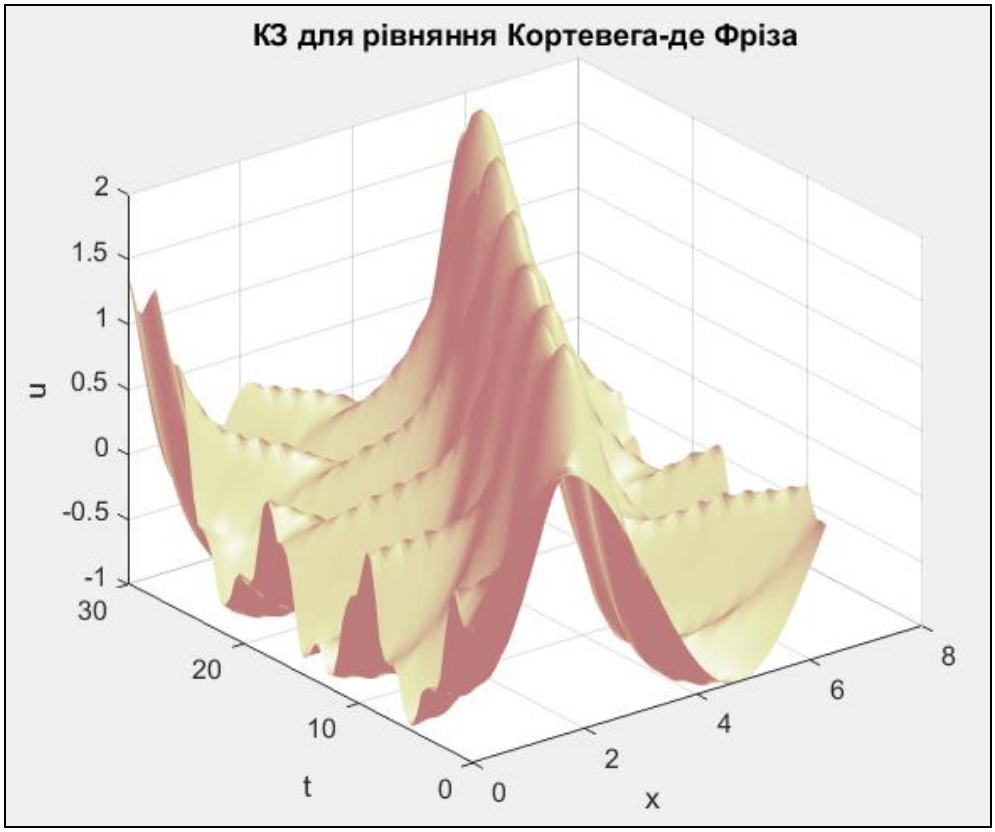


Рис. 8.3. Графік розв'язку КЗ для рівняння Кортевега-де Фріза (КдФ) з періодичними крайовими умовами

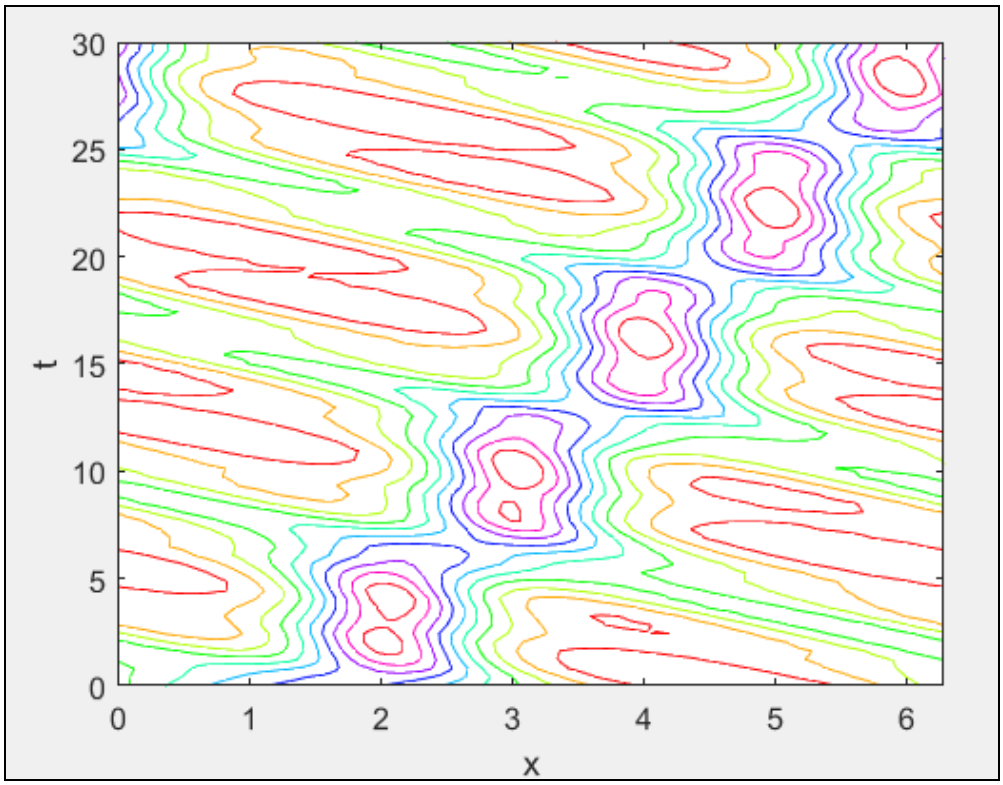
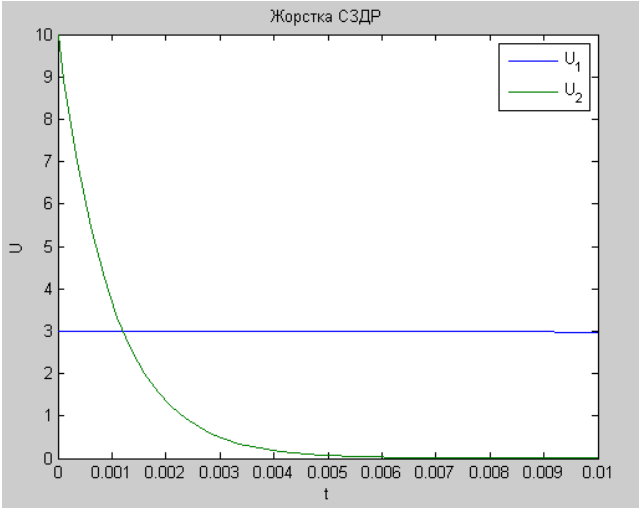


Рис. 8.4. Зображення ліній рівня для наближеного розв'язку *u*(*x*,*t*)

Приведемо приклад розв’язання (8.8) для A = [-1,1; 2,1000] – простої жорсткої СЗДР. Нижче наведено текст відповідної М-функції і результат її виконання:

```
function pl8_3
%% Приклад розв'язання задачі Коші для "жорсткої" системи ЗДР
clear all, close all, clc, warning off
A = [-1, 1; 2, -1e3];  % матриця A
f83 = @(t,U)(A*U);     % опис правої частини СЗДР
Y0 = [3; 10];          % вектор початкових умов
a = 0; b = 1e-2; h = 5e-4;
U = ode23s(f83, [0:h:b], Y0);
plot(U.x, U.y);
axis([a,b,0,10]); title('Жорстка СЗДР');
legend('U_1','U_2','Location','Best');
xlabel('t'); ylabel('U');
```



ЛЕКЦІЯ № 09

ВИКОРИСТАННЯ СОЛВЕРІВ MATLAB ДЛЯ РОЗВ'ЯЗАННЯ КРАЙОВИХ ЗАДАЧ ДЛЯ ЗДР

Розглянемо крайову задачу для ЗДР, записану у вигляді системи диференціальних рівнянь першого порядку, розв'язаних відносно похідної

$$\vec{U}'(x) = \vec{F}(x, \vec{U}(x)), \quad x \in [a,b] \tag{9.1}$$

і набору додаткових крайових умов

$$\vec{G}(\vec{U}(a), \vec{U}(b)) = 0. \tag{9.2}$$

Тут уведено позначення :

$\vec{U}(x) = [u_1(x), u_2(x), ..., u_n(x)]$ – вектор-функція шуканого розв'язку;

$\vec{F}(x, \vec{U}(x)) = [f_1(x, \vec{U}(x)), f_2(x, \vec{U}(x)), ..., f_n(x, \vec{U}(x))]$ – вектор-функція з описом правих частин системи (9.1), де $f_i(...)$ – відомі функції;

$\vec{G}(\vec{U}(a), \vec{U}(b)) = [g_1(\vec{U}(a), \vec{U}(b)), g_2(\vec{U}(a), \vec{U}(b)), ..., g_n(\vec{U}(a), \vec{U}(b))]$ – вектор-функція з описом крайових умов задачі (9.1), де $g_i(...)$ – відомі функції.

Для задачі (9.1) – (9.2) будемо вважати, що питання існування та єдиності розв'язку досліджені. Для наближеного розв'язку (9.1) – (9.2) використовують різні наближені методи:

- *редукція до задачі Коші* (*балістичний метод*);
- чисельно-аналітичні – *метод Трефца*; *проекційні*, *варіаційні*; *асимптотичні* (розвинення у ряд за малим параметром);
- *метод скінченних різниць* (у варіанті різних побудов різницевої задачі).

У системі MATLAB є вбудована функція `bvp4c` для розв'язання задач типу (9.1) – (9.2). Стандартна схема її використання має вигляд :

```
a = «a»; b = «b»; m = «m»;
x_init = «сітка по x»; % наприклад, linspace(a, b, m);
u_init = «[A(m,n)]»;
sol_init = bvpinit(x_init, u_init);
sol = bvp4c(@F, @G, sol_init);
% Виведення результатів :
N = «N»; x = linspace(a, b, N);
Y = deval(sol, x);
```

Функція `bvpinit` задає деяку "підказку":

`x_init` – вектор-рядок розмірності *m*, який задає сітку за змінною *x*, наприклад, `linspace(a,b,m);`

`u_init` – матриця розмірності *m*×*n* або вектор-стовпець розмірності *n*. Елементи матриці *A(m,n)* задають значення функцій $u_1(x), u_2(x), ..., u_n(x)$ у вузлах сітки. Якщо це вектор-стовпець, то ці значення для кожної функції $u_i(x)$ є сталими.

Вихідний параметр `bvpinit` – структура `sol_init`, з полями `x,y`.

Перший вхідний параметр в `bvp4c` задає вказівник на функцію користувача з описом правої частини системи (9.1). Ця функція повинна бути описана наступним чином:

```
function dfdx = F(x, y)
```

```
dfdx = [f1(…); f2(…); …; fn(…)];
```

Тут *y* – вектор значень функцій $u_1(x)$, $u_2(x)$, ..., $u_n(x)$ у точці *x*. Вихідний параметр *dfdx* – вектор-стовпець значень функцій $f_1(...)$, $f_2(...)$, ..., $f_n(...)$.

Другий вхідний параметр в *bvp4c* задає вказівник на функцію користувача з описом граничних умов (9.2). Ця функція повинна бути описана наступним чином:

```
function bc = G(ya, yb)
bc = [g1(…); g2(…); …; gn(…)];
```

Тут *ya,yb* – вектори, які задають значення функцій $u_1(x)$, $u_2(x)$, ..., $u_n(x)$ у граничних точках *a* і *b* відповідно. Вихідний параметр *bc* – вектор-стовпець зі значень функцій $g_1(...)$, $g_2(...)$, ..., $g_n(...)$.

Вихідний параметр *bvp4c* – структура *sol*, що описує знайдений розв'язок і містить наступні поля :

- ***x*** – сітка розв'язку, побудована *bvp4c* (вектор розмірності 1×*M*);
- ***y*** – наближений розв'язок y(x) в точках сітки x (матриця розмірності *n*×*M*);
- ***yp*** – значення y'(x) в точках сітки x (матриця розмірності *n*×*M*);
- ***solver*** – рядок зі значенням *'bvp4c'*.

Функція *deval* обчислює значення знайденого розв'язку *sol.y* в точках *X* – сітки виведення графіка(ів). Вихідний параметр функції *deval* – матриця *Y*, де $Y(i,:) = u_i(X)$, $i = \overline{1,n}$.

Приклад 20. Знайти наближений розв'язок модельної крайової задачі

$$\begin{cases} u''(x) = 6x(1-2x), \\ \quad x \in (0,1); \\ u(0) = 0,\ u(1) = 0 \end{cases}$$

і порівняти точний $u(x) = x^3(1-x)$ і знайдений розв'язок.

Нижче наведено текст відповідної *M*-функції з використанням *bvp4c*, результати роботи якого зображено на рис. 9.1. Можна порівняти отриманий розв'язок з розв'язком за методом *редукції до задачі Коші* (див. *M*-функцію *pl9_1r*).

```
function pl9_1
%% Розв'язання крайової задачі :
% u"(x) = 6x(1-2x),
% u(0) = 0, u(1) = 0.
%    Модельна задача: u(x)=x^3*(1-x)
clear all, close all, clc
% Створюємо структуру sol_init початкових даних із m точок,
% рівномірно розподілених на відрізку [0, 1] зі значеннями y=[0; 0]:
a = 0; b = 1; m = 51; xx = linspace(a, b, m); yy = [0; 0];
sol_init = bvpinit(xx, yy);
sol = bvp4c(@F, @G, sol_init);  % Розв'язуємо задачу
% Виведення результатів :
N = 101; x = linspace(a, b, N);
Y = deval(sol, x);
u = @(x)(x .^ 3 .* (1 - x));    % Завдання точного розв'язку
xx = linspace(a, b, 21); U = u(xx);
plot(x, Y(1,:),'k -', xx, U,'*', x, Y(2,:),'r :');
axis([a, b, -1.2, 0.7]); grid on;
title('Розв''язання КЗ для ЗДР');
xlabel('x'); ylabel('Y(x)');
```

```
legend('y(x)', 'u(x)', 'y''(x)');
U = Y(1, 1:5:end) - U;
disp(strcat('Похибка в нормі C=',num2str(norm(U,inf))))

function dydf = F(x, y)
dydf = [y(2); 6.*x.*(1-2.*x)];

function bc = G(ya, yb)
bc = [ya(1); yb(1)];
```

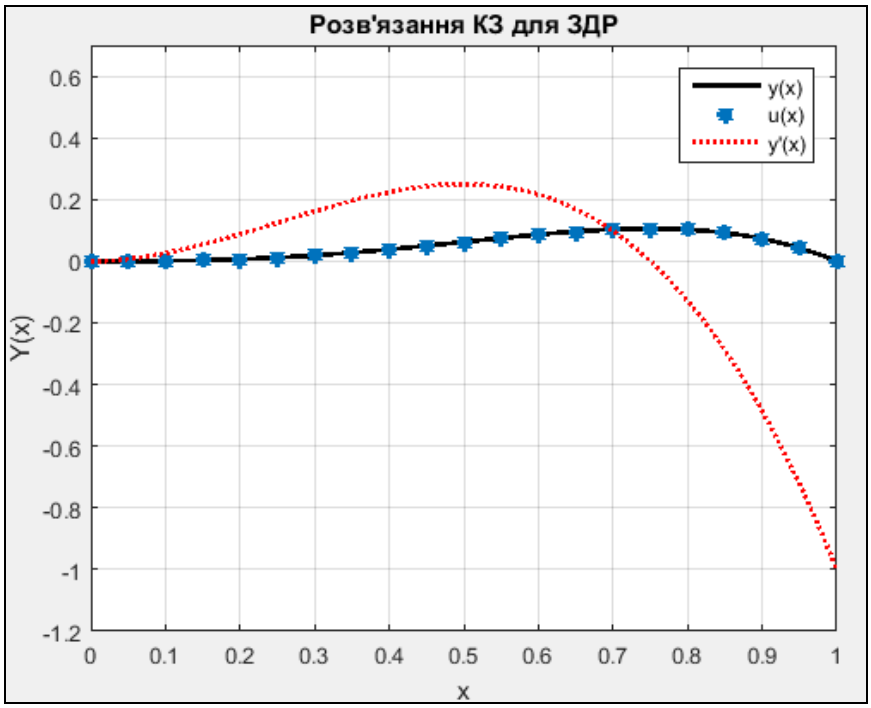


Рис. 9.1. Графік розв'язку модельної крайової задачі

```
Похибка в нормі C=1e-008
```

Приклад 21. Розв’язати крайову задачу для нелінійного рівняння Треша:

$$\begin{cases} u''(x) = \alpha * sh(\alpha * u(x)), \ x \in (0,1); \\ u(0) = 0, \ u(1) = 1 \end{cases}$$

при наступних варіантах параметру $\alpha = [1.0, 5.0, 10.0, 15.0, 17.0]$.

Нижче наведено текст функції *pl9_2.m*, а її робота – показана на рис. 9.2.

```
function pl9_2
%% Розв'язання КЗ для нелінійного рівняння Треша:
% u"(x) = alfa*sinh(alfa*u(x)),
%     u(0) = 0, u(1) = 1.
%  за допомогою вбудованих функцій MATLAB.
clear all, close all, clc
global alfa
a = 0; b = 1; m = 51;
xx = linspace(a, b, m); yy = [0; 1];
sol_init = bvpinit(xx, yy);
ALFA = [1.0, 5.0, 10.0, 15.0, 17.0]; na = length(ALFA);
N = 101; x = linspace(a, b, N); Y = zeros(na, N);
for i = 1 : na
    alfa = ALFA(i); fprintf('alfa=%6.2f\n', alfa);
    sol = bvp4c(@F, @G, sol_init);
    y = deval(sol, x);
    Y(i,:) = y(1,:);
```

```
end
plot(x, Y,'LineWidth',2), grid on
axis([-0.1,1.1,-0.1,1.1])
s = 'Розв''язання КЗ для рівняння Треша';
title(s,'FontWeight','bold');
xlabel('x'); ylabel('u(x)');
ch=legend(strcat('{\alpha}=',num2str(ALFA(:))),'Location','Best');
set(ch,'Fontsize',10);

function dydf = F(x, y)
global alfa
dydf = [y(2); alfa.*sinh(alfa.*y(1))];

function bc = G(ya, yb)
bc = [ya(1); yb(1) - 1];
```

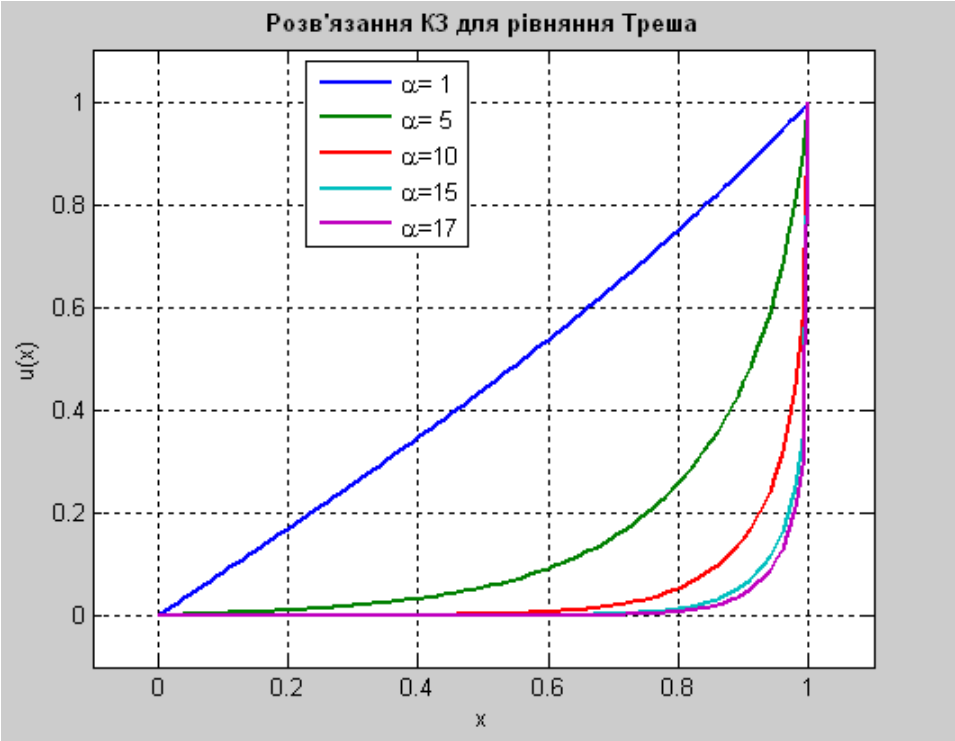


Рис. 9.2. Графіки розв'язку крайової задачі для нелінійного рівняння Треша при різних значеннях параметра α

Для більших значень α, нажаль, використання *bvp4c* не приносить результату. Таке "просте" на вигляд нелінійне рівняння є одним із тестових для перевірки методів розв'язання нестійких нелінійних крайових задач. Отже не для всіх задач можна використовувати стандартні програми.

Приклад 22. Розв'язати модельну крайову задачу:

$$\begin{cases} \left(K(x)*u'(x)\right)' = 0,\ x \in (0,1), \\ u(0)=1,\ u(1)=0, \\ [u(t)] \equiv \left(u(t+0)-u(t-0)\right)=0, \\ [K(t)u'(t)]=0;\quad t \in (0,1), \end{cases}$$

де $$K(x)=\begin{cases} K_1,\ 0<x<t; \\ K_2,\ t<x<1. \end{cases}$$

$K_{1,2} > K_0,\ K_0 > 0;\ t = x_m + 0.5h,\ 1 < m < N.$

Порівняти наближений розв'язок, отриманий з рівномірним кроком, із відомим точним розв'язком задачі:

$$u(x)=\begin{cases}1-\alpha * x, & 0 \le x \le t;\\ \beta * (1-x), & t < x \le 1.\end{cases},\ \alpha = 1/(\gamma + (1-\gamma) * t),\ \beta = \alpha * \gamma,\ \gamma = K_1/K_2.$$

Нижче наведено текст відповідного *M*-файла, результати роботи якого зображено на рис. 9.3–9.4.

```
function pl9_3
%% Розв'язання модельної крайової задачі :
%      ( K(x)u'(x) )' = 0,
%       u(0) = 1, u(1)= 0.
%  з розривним коефіцієнтом K(x).
clear all, close all, clc
global t al be K1 K2
a = 0; b = 1; N = 51;  K1 = 2; K2 = 5; m = 33;
xx = linspace(a, b, N); t = xx(m) + 0.5.*(xx(2)-xx(1));
ga = K1./K2; al = 1./(ga + (1 - ga).*t); be = al.*ga;
%Створюєм структуру sol_init початкових даних із m точок,
%  рівномірно розподілених на відрізку [a, b]
%  зі значеннями y=[1; 0]:
yy = [1; 0]; sol_init = bvpinit(xx, yy);
sol = bvp4c(@F, @G, sol_init);   % Розв'язуєм задачу
% Виведення результатів :
n = 101; x = linspace(a, b, n);  Y = deval(sol, x);
nn = 21; xx = linspace(a, b, nn); u = xx;
for k = 1 : nn
    u(k) = U(xx(k));
end
plot(x, Y(1,:),'k -', xx, u, '*','LineWidth',2)
axis([a-0.1, b+0.1, -0.1, 1.1]), grid on
s = 'КЗ для (K(x)u''(x))''=0 (K-розривна)';
title(s,'FontWeight','bold')
xlabel('x'), ylabel('Y(x)')
legend('Y','u(x)','Location','Best'), pause
u = Y(1, 1:5:end) - u;
disp(strcat('Похибка в нормі C=',num2str(norm(u,inf))))
plot(xx, abs(u), 'LineWidth', 2), grid on
s = 'Абсолютна похибка'; title(s,'FontWeight','bold')
xlabel('x'), ylabel('|Y-u(x)|')

function k = K(x)
% підфункція з описом коефіцієнта K
global t K1 K2
if x <= t
    k = K1;
else
    k = K2;
end

function dydf = F(x, y)
% підфункція опису правої частини системи
dydf = [y(2)./K(x); 0];

function bc = G(ya, yb)
% підфункція опису системи крайових умов
bc = [ya(1) - 1; yb(1)];

function u = U(x)
% підфункція опису точного розв'язку u(x)
```

```
global t al be
if x <= t
    u = 1 - al.*x;
else
    u = be.*(1 - x);
end
```

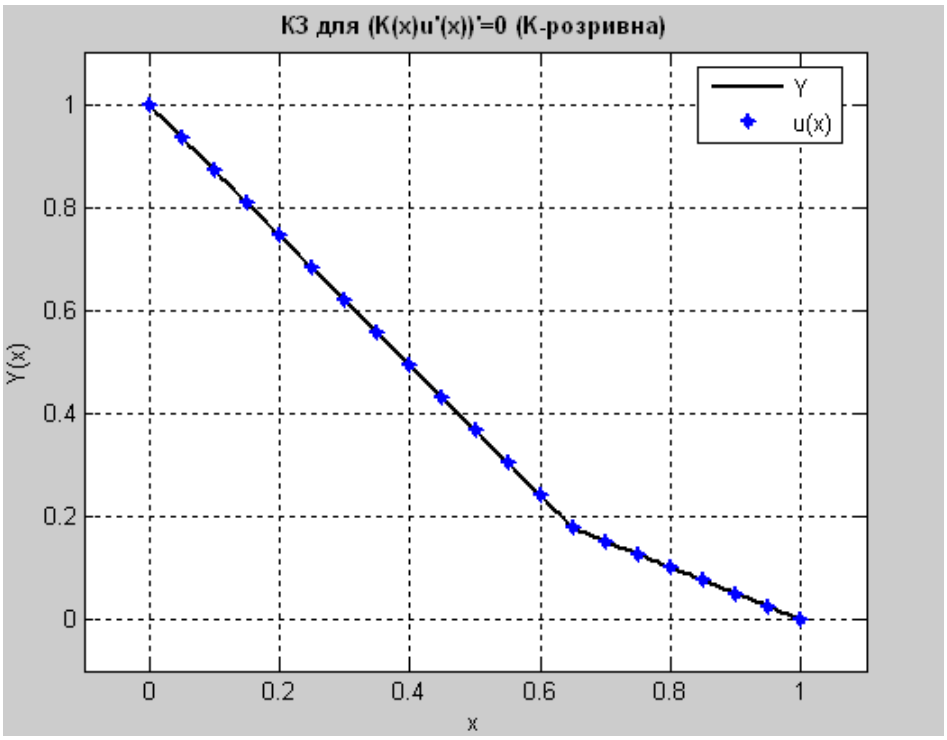


Рис. 9.3.Графік розв'язку КЗ з розривним коефіцієнтом

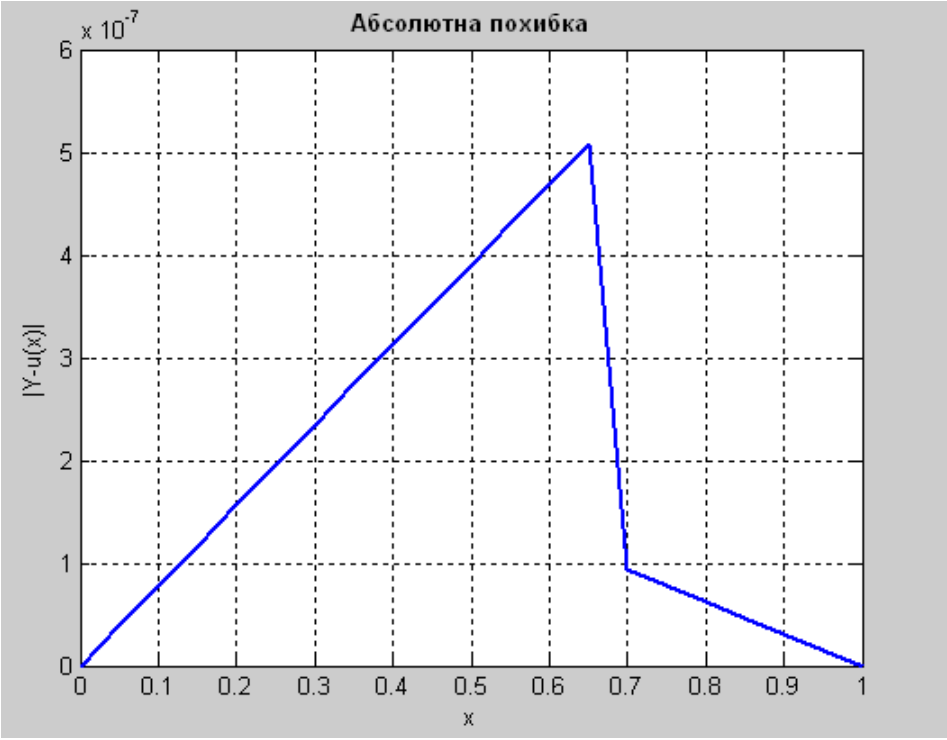


Рис. 9.4. Графік абсолютної похибки розв'язку КЗ з розривним коефіцієнтом

```
Похибка в нормі C=1.1294e-006
```

ЛЕКЦІЯ № 10

ВИКОРИСТАННЯ СОЛВЕРІВ MATLAB ДЛЯ РОЗВ'ЯЗАННЯ ОДНОВИМІРНИХ КРАЙОВИХ ЗАДАЧ ДЛЯ СИСТЕМ ПАРАБОЛІЧНИХ І ЕЛІПТИЧНИХ ДРЧП

Досить часто необхідно розв'язувати початково-крайові задачі для одновимірних параболічних рівнянь (систем рівнянь), які можуть комбінуватись з параболічними рівняннями. Наявність нелінійності в таких задачах, суттєво ускладнює як вибір наближеного методу розв'язання, так і сам процес отримання числового розв'язку.

Розглянемо наступну систему рівнянь

$$\vec{C}\left(x,t,\vec{U},\frac{\partial\vec{U}}{\partial x}\right)\frac{\partial\vec{U}}{\partial t}=x^{-m}\,\frac{\partial}{\partial x}\left(x^m\,\vec{f}\left(x,t,\vec{U},\frac{\partial\vec{U}}{\partial x}\right)\right)+\vec{S}\left(x,t,\vec{U},\frac{\partial\vec{U}}{\partial x}\right), \qquad (10.1)$$

$$x\in(a,\ b),\quad t\in(t0,\ tf]$$

для якої виконуються початкові умови

$$\vec{U}(x,t_0)=\vec{U}_0(x) \qquad (10.2)$$

і набір крайових умов в граничних точках $x=a$ (при $m>0$ повинно бути $a\geq 0$), $x=b$ скінченного інтервалу $[a,\ b]$

$$\vec{p}(x,t,\vec{U})+\vec{q}(x,t)\vec{f}\left(x,t,\vec{U},\frac{\partial\vec{U}}{\partial x}\right)=0\,. \qquad (10.3)$$

Тут уведено позначення:

- $\vec{U}(x)=[u_1(x,t),u_2(x,t),...,u_n(x,t)]$ – вектор-функція шуканого розв'язку;
- $m=0,1,2$ – параметр, що характеризує плоску, циліндричну, або сферичну симетрію задачі;
- $\vec{C}\left(x,t,\vec{U},\frac{\partial\vec{U}}{\partial x}\right)$ – відома діагональна матриця (теплоємність), елементи якої приймають нульові або додатні значення (в солвері $\vec{C}$ задається як вектор-стовпець). Елемент, рівний нулеві, відповідає еліптичному рівнянню, інакше – параболічному рівнянню. В системі (10.1) має бути принаймні одне параболічне рівняння. Елемент *c(...)*, який відповідає параболічному рівнянню, може перетворюватися у нуль в ізольованих значеннях *x*, якщо ці значення – вузли сітки. Розриви розподілу *c(...)* і/або *s(...)* за просторовою координатою допустимі при умові, що точки сітки представлені в кожному підінтервалі, де знаходяться розриви;
- $\vec{f}\left(x,t,\vec{U},\frac{\partial\vec{U}}{\partial x}\right)$, $\vec{S}\left(x,t,\vec{U},\frac{\partial\vec{U}}{\partial x}\right)$ – відомі вектор-функції (потік, джерела) з описом коефіцієнтів рівняння системи (10.1);
- $\vec{p}(x,t,\vec{U})$, $\vec{q}(x,t)$ – відомі вектор-функції, причому елементи *q(...)* або тотожно дорівнюють нулеві або не дорівнюють нулю при жодних значеннях *x, t*. Крім того, із цих двох коефіцієнтів, тільки *p(...)* може залежати від *U(x,t)*.

У системі MATLAB є вбудована функція `pdepe` для розв'язку задач (10.1) – (10.3). Найпростіший синтаксис виклику цієї функції :

```
sol = pdepe(m, @pdefun, @icfun, @bcfun, xmesh, tspan);
```

Аргументи функції `pdepe`:

- *m* – параметр зі значеннями 0 | 1 | 2. Він описує симетрію задачі;
- *pdefun* – функція користувача, яка обчислює значення коефіцієнтів *C, f, S* з (10.1). Ця функція викликається за формою

```
[C, f, S] = pdefun(x, t, u, dudx);
```

Її вхідними параметрами є скалярні величини *x*, *t* і вектори *u*, *dudx*, які наближено відповідають розв'язку *U* та його частинній похідній по *x*;

- *icfun* – функція користувача, яка обчислює початкові умови (10.2). Ця функція викликається за формою

```
u = icfun(x);
```

- *bcfun* – функція користувача, яка обчислює значення функцій *p* та *q* граничних умов (10.3) і викликається за формою

```
[pl, ql, pr, qr] = bcfun(xl, ul, xr, ur, t);
```

Тут позначено :

ul – наближений розв'язок на лівому кінці відрізка *xl = a*;
ur – наближений розв'язок на правому кінці відрізка *xr = b;*
pl, *ql* – масиви функцій *p*, *q* для лівої граничної точки *xl*;
pr, *qr* – масиви функцій *p*, *q* для правої граничної точки *xr*;

- *xmesh* – вектор $[x_1, x_2, ..., x_N]$, $N > 3$, що задає координати сітки по *x* $a = x_1 < x_2 < ... < x_N = b$;
- *tspan* – вектор $[t0, t1, ..., tf]$ моментів часу для фіксації шуканої функції за координатою *t*. Елементи *tspan* повинні задовольняти умову $t0 < t1 < ... < tf$ і кількість елементів *tspan* повинна бути не менше 3.

Функція *pdepe* формує розв'язок у вигляді багатовимірного масиву *sol* :

$$U_i(x,t) = sol(:,:,i),$$

при цьому перший індекс (*j*) відповідає номеру часового шару t_j, другий індекс (*k*) визначає номер вузла x_k, а третій індекс (*i*) дає номер компоненти вектора розв'язку. Таким чином, елемент *sol(j,k,i)* дає значение розв'язку $u_i(x_k, t_j)$. Інакше кажучи, елемент *Ui(j,k)=sol(j,k,i)* є наближенням *Ui* при *(t,x)* = (*tspan(j), xmesh(k)*).

Зробимо короткий огляд необов'язкових аргументів функції *pdepe*.

Вхідні параметри :

- *options* – формують значення параметрів функції *odeset*, яка використовуєть-ся в *ode*-солверах, наприклад, *RelTol, AbsTol, NormControl, InitialStep, MaxStep* (детальну інформацію див. функцію *odeset*).

Вихідні параметри :

- *tsol* – вектор-стовпець часових значень, указаних в *tspan*, до першої термінальної події;
- *sole(j,.,.)* – розв'язок в *te(j)*;
- *te* – вектор часових значень, коли відбуваються події;
- *ie* – вектор, значення якого вказують, яка подія сталася.

Як бачимо, для «звичайних задач», реально може бути використано тільки `options`.

Розглянемо деякі приклади використання М-функції `pdepe`.

Приклад 23. Розв'яжемо крайову задачу, яка містить еліптичну і параболічну складову системи (10.1):

$$\begin{cases} \dfrac{\partial}{\partial x}\left(\dfrac{\partial U_1(x,t)}{\partial x}\right) = 10 * U_2(x,t), \\ \dfrac{\partial U_2(x,t)}{\partial t} = \dfrac{\partial}{\partial x}\left(\dfrac{\partial U_2(x,t)}{\partial x}\right) + U_1(x,t) * U_2(x,t), \\ \qquad x \in (0,1), \quad t \in (0,1]; \\ \dfrac{\partial U_1(0,t)}{\partial x} = t, \quad U_1(1,t) = 1 - t, \\ U_2(0,t) = 0, \quad U_2(1,t) = 0, \\ U_2(x,0) = x(1-x). \end{cases} \tag{10.4}$$

Для використання солвера `pdepe` і опису відповідних М-функцій `pdefun`, `icfun`, `bcfun` нам необхідно визначитись з функціями $\vec{C}(...), \vec{f}(...), \vec{S}(...), \overrightarrow{U_0}(...), \vec{p}(...), \vec{q}(...)$, а це означає, що нашу задачу потрібно привести до форми, вказаної в описі солвера `pdepe`. Виконаємо це приведення (тут маємо $m = 0$) і спочатку запишемо в скалярному представлені (з позначеннями відповідних елементів задачі (10.1)-(10.3)):

$$\begin{cases} \underbrace{(0)}_{C_1} * \dfrac{\partial U_1(x,t)}{\partial t} = \dfrac{\partial}{\partial x}\left(\underbrace{\dfrac{\partial U_1(x,t)}{\partial x}}_{f_1(...)}\right) + \underbrace{(-10 * U_2(x,t))}_{S_1(...)}, \\ \underbrace{(1)}_{C_2} * \dfrac{\partial U_2(x,t)}{\partial t} = \dfrac{\partial}{\partial x}\left(\underbrace{\dfrac{\partial U_2(x,t)}{\partial x}}_{f_2(...)}\right) + \underbrace{(U_1(x,t) * U_2(x,t))}_{S_2(...)}, \\ \qquad a = 0, b = 1, \quad x \in (0,1), \quad t \in (0,1]; \\ U_1(x,0) = 0, \quad U_2(x,0) = x(1-x); \\ \underbrace{(-t)}_{pl_1(...)} + \underbrace{(1)}_{ql_1(...)} * \underbrace{\dfrac{\partial U_1(0,t)}{\partial x}}_{f_1(...)} = 0, \qquad \underbrace{(U_2(0,t))}_{pl_2(...)} + \underbrace{(0)}_{ql_2(...)} * \underbrace{\dfrac{\partial U_2(0,t)}{\partial x}}_{f_2(...)} = 0, \\ \underbrace{(U_1(1,t) + t - 1)}_{pr_1(...)} + \underbrace{(0)}_{qr_1(...)} * \underbrace{\dfrac{\partial U_1(0,t)}{\partial x}}_{f_1(...)} = 0, \ \underbrace{(U_2(1,t))}_{pr_2(...)} + \underbrace{(0)}_{qr_2(...)} * \underbrace{\dfrac{\partial U_2(1,t)}{\partial x}}_{f_2(...)} = 0. \end{cases}$$

Звідси маємо необхідні векторні і матричні функції, які є в формі (10.1)-(10.3), для нашої задачі, де позначення [•; •] – вектор-стовпець MATLABa :

$$\vec{C}\left(x,t,\vec{U},\frac{\partial \vec{U}}{\partial x}\right) = [0;\ 1], \ \vec{f}\left(x,t,\vec{U},\frac{\partial \vec{U}}{\partial x}\right) = \left[\frac{\partial U_1}{\partial x};\ \frac{\partial U_2}{\partial x}\right],$$

$$\vec{S}\left(x,t,\vec{U},\frac{\partial \vec{U}}{\partial x}\right) = [-10 * U_2(x,t);\ U_1(x,t) * U_2(x,t)];$$

$$\vec{U}_0(x) = [0;\ x(1-x)];$$

$$\vec{p}\left(0,t,\vec{U}\right)=[-t;U_2(0,t)],\ \vec{p}\left(1,t,\vec{U}\right)=[U_1(1,t)+t-1;U_1(1,t)];$$
$$\vec{q}(0,t)=[1;0],\ \vec{q}(1,t)=[0;0].$$

Для М-функцій *pdefun*, *icfun*, *bcfun* в нашій програмі оберемо наступні імена: *koefpde*, *initpde*, *boundpde*, відповідно Нижче наведено текст нашої М-функції та результати її роботи (рис. 10.1–10.2).

```
function pl10_1
% Розв'язання крайової задачі :
%      0 = U1" - 10*U2(x,t),
% dU2/dt = U2" + U1(x,t)*U2(x,t),
%   U1'(0,t)= t, U1(1,t) = 1-t,
%   U2(0,t) = 0, U2(1,t) = 0.
%   U2(x,0) = x*(1-x).
% за допомогою вбудованих функцій MATLAB.

close all, clear all, clc

m = 0; % плоска задача
xl = 0; xr = 1; x = linspace(xl, xr, 21); % сітка по x
t = linspace(0, 1, 21);                   % вектор по часу

% Розв'язуємо задачу
sol = pdepe(m, @koefpde, @initpde, @boundpde, x, t);

[X, T] = meshgrid(x, t);
% графіки розв'язку U1(x,t), U2(x,t)
u = sol(:, :, 1);
mesh(X, T, u); colormap gray; view(-8, 12);
title('Функція U1(x,t)');
xlabel('x'); ylabel('t'); pause;

u = sol(:, :, 2);
mesh(X, T, u); colormap('default');
title('Функція U2(x,t)');
xlabel('x'); ylabel('t');


function [c, f, s] = koefpde(x, t, u, DuDx)
% коефіцієнти рівняння
c = [0; 1];
f = [DuDx(1); DuDx(2)];
s = [-10 .* u(2);
     u(1) .* u(2)];


function u0 = initpde(x)
% початкова умова
u0 = [0;
      x .* (1 - x)];


function [pl, ql, pr, qr] = boundpde(xl, ul, xr, ur, t)
% коефіцієнти крайових умов
pl = [-t;
      ul(2)];
ql = [1; 0];
```

```
pr = [ur(1) + t - 1;
      ur(2)];
qr = [0; 0];
```

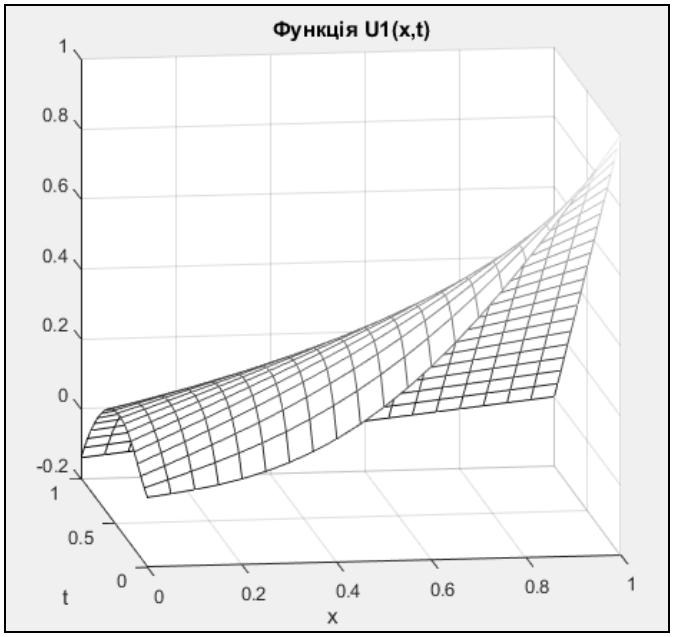


Рис. 10.1. Графік функції $U_1(x, t)$

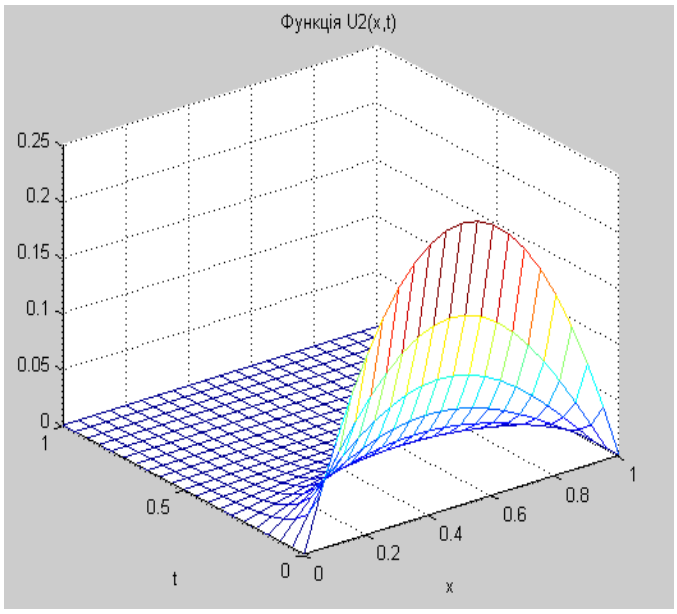


Рис. 10.2. Графік функції $U_2(x, t)$

Приклад 24. Розв'язати крайову задачу для нелінійного рівняння Бюргерса :

$$\begin{cases} \dfrac{\partial U(x,t)}{\partial t} + L * \dfrac{\partial}{\partial x}\left(\dfrac{U^2(x,t)}{2}\right) = K * \dfrac{\partial}{\partial x}\left(\dfrac{\partial U(x,t)}{\partial x}\right), \\ x \in (-3,3), \quad t \in (0,1]; \quad L = 1, \; K = 1; \\ U(x,0) = \exp\left(-20x^2\right); \\ U(-3,t) = 0, \quad U(3,t) = 0. \end{cases} \tag{10.5}$$

Аналогічно попередній задачі, нашу задачу необхідно привести до форми

запису (10.1)-(10.3), визначитись з функціями $\vec{C}(...), \vec{f}(...), \vec{S}(...), \overrightarrow{U_0}(...), \vec{p}(...), \vec{q}(...)$, при описі функцій *koefpde, initpde, boundpde* врахувати, що маємо тільки одне рівняння, а отже всі вектор-функції є скалярами. Нижче наведено текст відповідної М-функції і результати її роботи (рис. 10.3–10.4).

```
function pl10_2
% Розв'язання крайової задачі для нелінійного рівняння Бюргерса:
% du/dt + 0.5*L*(u^2)' =  K*u"(x,t),
% x є (0,1), t є (0,1]; L,K є const,
%     u(-3,t) = 0,  u(3,t) = 0,
%      u(x,0) = exp(-20x^2).
% за допомогою вбудованих функцій MATLAB.

close all, clear all, clc

m = 0; % плоска задача

global K L
K = 1; L = 0.5;
xl = -3; xr = 3; x = linspace(xl, xr, 61); % сітка по x
t = linspace(0, 1, 21);                    % вектор по часу

% Розв'язуємо задачу
sol = pdepe(m, @koefpde, @initpde, @boundpde, x, t);
u = sol(:, :, 1);

% графік розв'язку
[X, T] = meshgrid(x, t);
mesh(X, T, u); shading interp; colormap gray;
title('Чисельний розв''язок'); xlabel('x'); ylabel('t');
view(31, 48); pause;

% проста анімація
fig = plot(x, u(1, :), 'erase', 'xor');
for k = 1 : length(t)
    set(fig, 'xdata', x, 'ydata', u(k, :));
    pause(.3);
end


function [c, f, s] = koefpde(x, t, u, DuDx)
% коефіцієнти рівняння
global K L
c = 1;
f = K .* DuDx;
s = -L .* u .* DuDx;


function u0 = initpde(x)
% початкова умова
u0 = exp(-20 .* x .* x);


function [pl, ql, pr, qr] = boundpde(xl, ul, xr, ur, t)
% коефіцієнти крайових умов
pl = ul; ql = 0;
```

```
pr = ur; qr = 0;
```

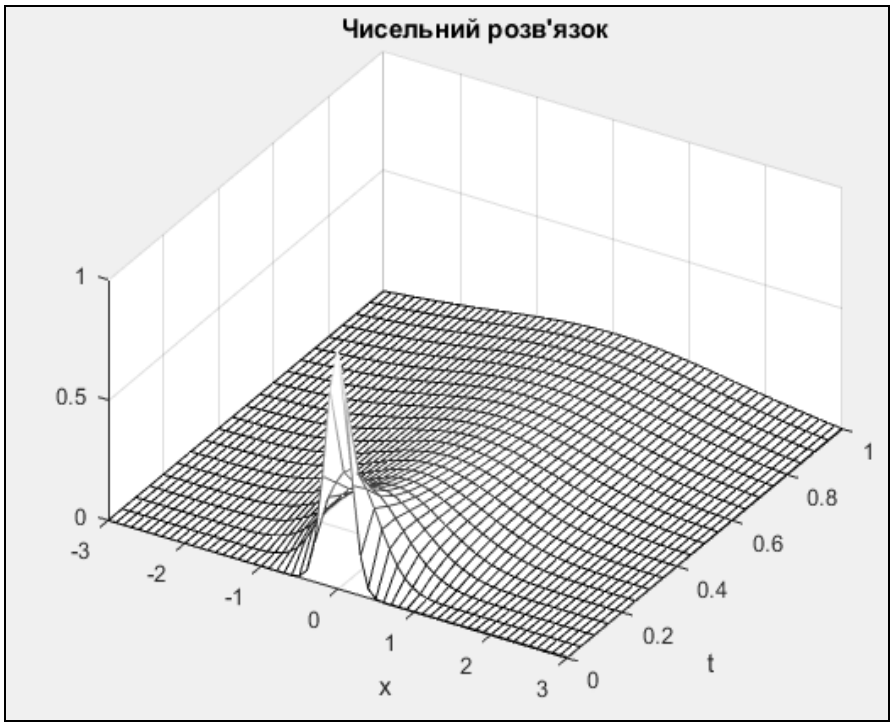


Рис.10.3. Графік наближеного розв’язку нелінійного рівняння Бюргерса

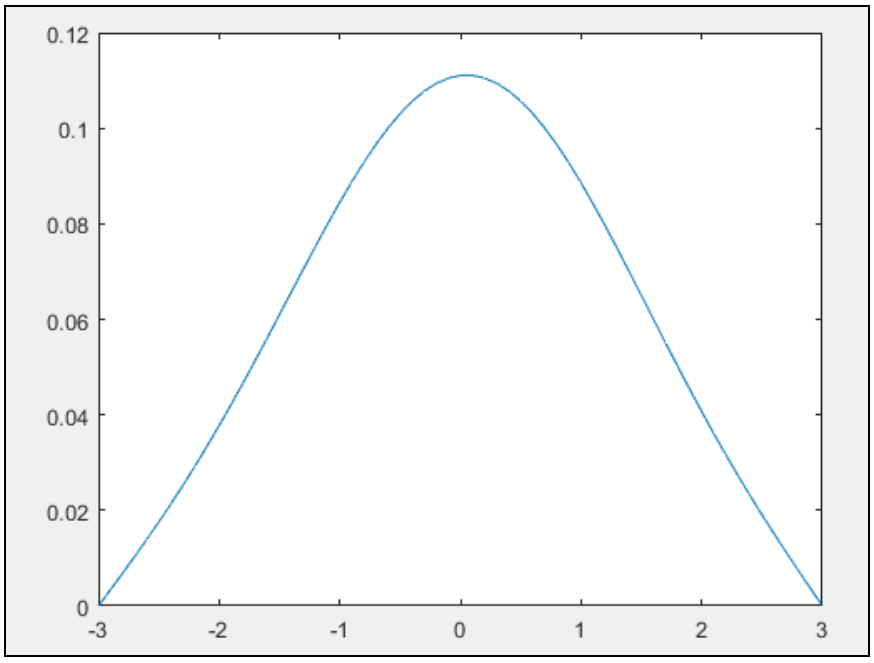


Рис.10.4. Кінцевий графік “анімації” (поданий в відповідному масштабі осей).

Задачі, які розглянуто в **Прикладі 23** і **24**, не є модельними, чи такими, що ми знаємо аналітичний розв’язок, тому важко судити про точність наближеного розв’язку.

Розв’яжемо просту задачу :

$$\begin{cases} u_t' = u_{xx}'' + 2t\exp(-2t)\cos(x) + x + 1, \\ x \in (0,b), \quad b = 0.5\pi, \quad t \in (0,1]; \\ u(x,0) = \cos(3x); \quad x \in [0,b]; \\ u_x'(0,t) = t, \quad u(b,t) = (1+b)t, \quad t \in [0,T]. \end{cases} \tag{10.6}$$

яку ви могли розглядати в курсі «Методи математичної фізики» і знаходили розв'язок методом Фур'є :

$$u(x,t) = 2(\exp(-t) - (t+1)\exp(-2t))\cos(x) + \exp(-9t)\cos(3x) + (x+1)t$$

Створимо М-функцію *pl10_3* :

```
function pl10_3
% Розв'язання мішаної крайової задачі для
% неоднорідного рівняння теплопровідності:
%  du/dt = u"(x,t)+2t*exp(-2t)cos(x)+x+1,
%   x є (0,b), b=0.5*pi; t є (0,T], T=1;
%    u'(0,t) = t, u(b,t) = (1+b)*t,
%       u(x,0) = cos(3x).
% за допомогою вбудованих функцій MATLAB.
% Розв'язок, отриманий м.Фур'є :
%u(x,t)=2(exp(-t)-(t+1)exp(-2t))cos(x)+exp(-9t)cos(3x)+(x+1)t

close all, clear all, clc

global L
b = 0.5*pi; T = 1; L = b+1;

m = 0; % плоска задача
xl = 0; xr = b; x = linspace(xl, xr, 51); % сітка по x
t = linspace(0, T, 33);                   % вектор по часу
% Розв'язуємо задачу
sol = pdepe(m,@koefpde,@initpde,@boundpde,x,t);
U = sol(:,:,1); % скалярне рівняння => функція одна

s = 'Числовий розв''язок U(X,T)';
[X, T] = meshgrid(x, t);
surfl(X, T, U), colormap('default')
title(s, 'FontWeight', 'bold')
xlabel('x'), ylabel('t')
view(-44, 13)
pause

% u(x,t) за методом Фур'є :
u = @(x,t)(2.*(exp(-t)-(t+1).*exp(-2.*t)).*cos(x)+...
           exp(-9.*t).*cos(3.*x)+(x+1).*t);

s = 'Абсолютна похибка |U(X,T)-u(x,y)|';
U = abs(U - u(X,T));
fprintf([s,' в нормі C =%.6e\n'], norm(U,inf))
surfl(X, T, U), colormap('default')
title(s, 'FontWeight', 'bold')
xlabel('x'), ylabel('t')
view(-44, 13)

function [c, f, s] = koefpde(x, t, u, DuDx)
% коефіциенти рівняння
c = 1;
f = DuDx;
s = 2.*t; s = s.*exp(-s).*cos(x)+x+1;
```

```
function u0 = initpde(x)
% початкова умова u0(x)
u0 = cos(3.*x);

function [pl, ql, pr, qr] = boundpde(xl, ul, xr, ur, t)
% коефіцієнти крайових умов
global L
pl = -t;  ql = 1;
pr = ur - L.*t;  qr = 0;
```

Результати її роботи (рис. 10.5–10.6)

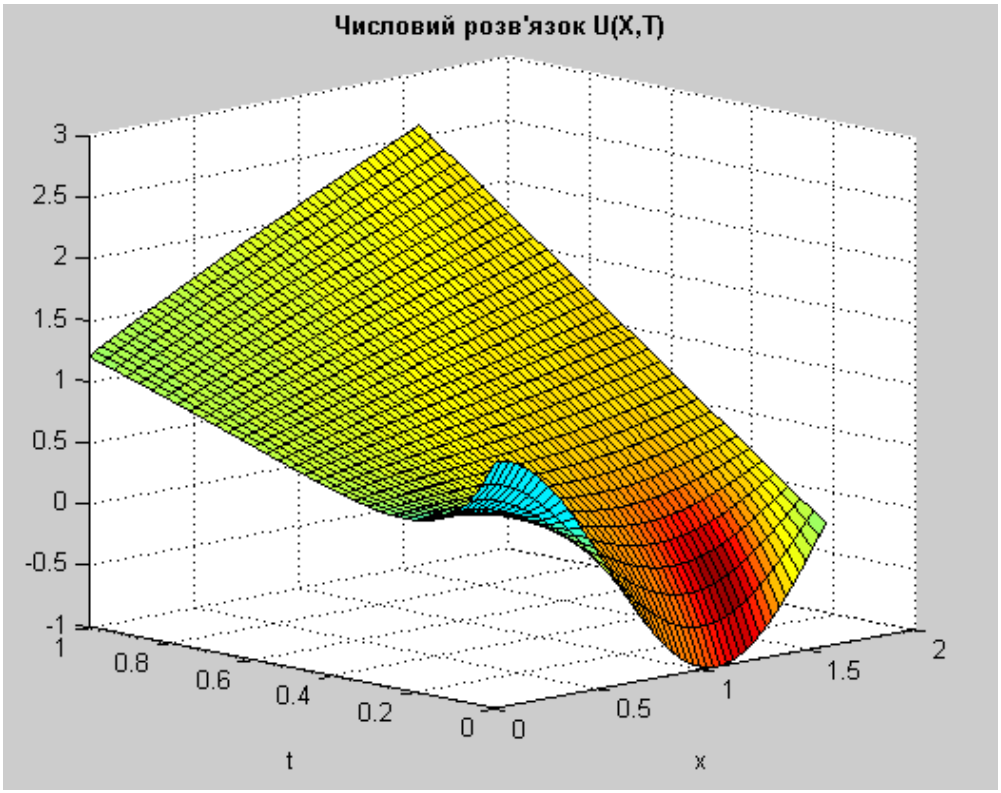


Рис.10.5. Наближений розв’язок задачі U(X,T) солвером *pdepe*.

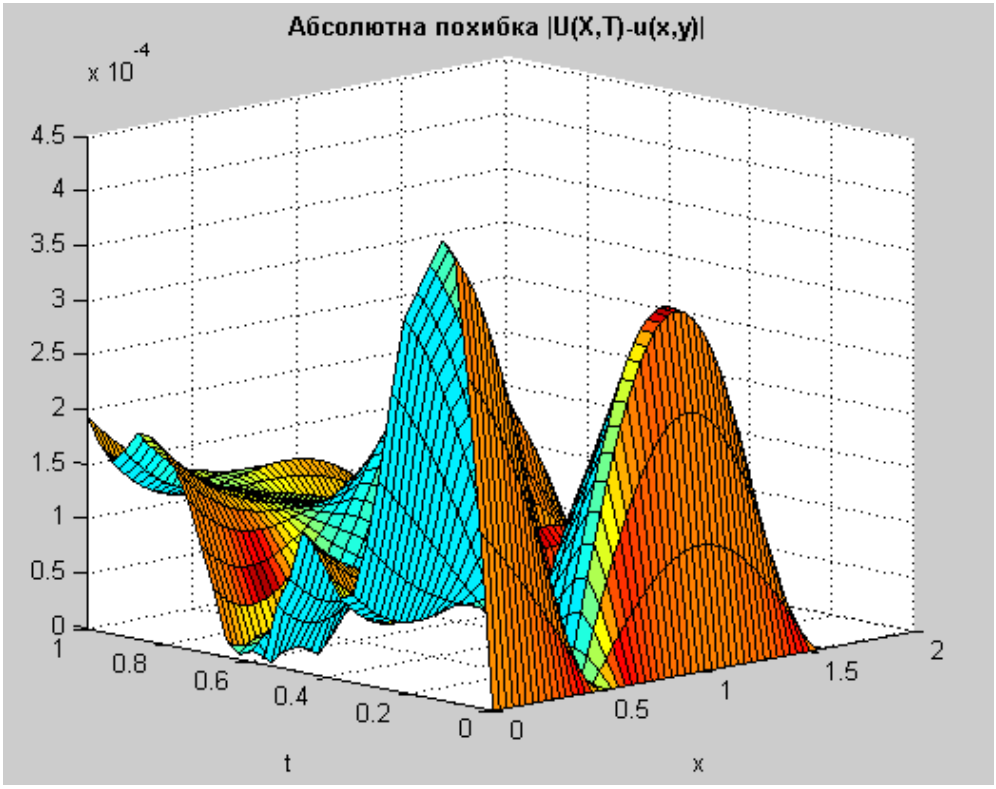


Рис.10.6. Абсолютна похибка числового і розв'язку за методом Фур'є.

```
Абсолютна похибка |U(X,T)-u(x,y)| в нормі C =1.119714e-002
```

Як бачимо, результат отримано досить непоганий, враховуючи, що кроки сітки $h_x = 0.0308$, $h_t = 0.03125$.

Розв'яжемо ще одну задачу для однорідного рівняння теплопровідності (у варіанті сферичної симетрії) :

$$\begin{cases} \dfrac{\partial u(r,t)}{\partial t} = \left(\dfrac{a}{r}\right)^2 \dfrac{\partial}{\partial r}\left(r^2 \dfrac{\partial u(r,t)}{\partial r}\right), \\ \quad r \in (0, re), \quad t \in (0, \mathrm{T}]; \\ u(r,0) = f_0 * \left(1 - \dfrac{r}{re}\right); \quad r \in [0, re]; \\ r^2 \dfrac{\partial u(r,t)}{\partial r}\Big|_{r\to 0} = 0, \quad u(re,t) = 0, \quad t \in [0, \mathrm{T}]. \end{cases} \tag{10.7}$$

яку ви могли розглядати в курсі «Методи математичної фізики» і знаходили розв'язок методом Фур'є :

$$u(r,t) = \begin{cases} \dfrac{8 f_0 re}{\pi^3 r} \sum\limits_{i=1,(2)}^{\infty} \sin\left(\dfrac{\pi * i * r}{re}\right) \dfrac{\exp(-a^2 \lambda_i t)}{i^3}, & r \neq 0; \\ \dfrac{8 f_0}{\pi^2} \sum\limits_{i=1,(2)}^{\infty} \dfrac{\exp(-a^2 \lambda_i t)}{i^2}, & r = 0; \end{cases} \qquad \lambda_i = \left(\frac{\pi * i}{re}\right)^2.$$

Створимо М-функцію *p110_4*:

```
function p110_3
% Розв'язання мішаної крайової задачі для
function p110_4
% Розв'язання мішаної крайової задачі для
% однорідного рівняння теплопровідності
```

```
%     (сферична симетрія):
%     du/dt = (a/r)^2*(r^2*u')',
%       r є (0,re), t є (0,T];
%     u'(0,t) = 0, u(re,t) = 0,
%       u(r,0) = f0*(1-r/re).
% за допомогою вбудованих функцій MATLAB.
% Порівняння з розв'язком, отриманим м.Фур'є.

close all, clear all, clc
global a re f0 ep aa
a = 2; rb = 0; re = 3; T = 1; f0 = 1;
ep = 1e-10; aa = 1./(a.*a);

m = 2; % сферична симетрія
rl = rb; rr = re;
% будуємо рівномірні сітки :
r = linspace(rl, rr, 31);   %  по r
t = linspace(0, T, 21);     %  по t

% Розв'язуємо задачу
sol = pdepe(m, @koefpde, @initpde, @boundpde, r, t);
U = sol(:,:,1); % скалярне рівняння => функція одна

[R, T] = meshgrid(r, t);

surfl(R, T, U), colormap('default')
s = 'Числовий розв''язок U(R,T)';
title(s, 'FontWeight', 'bold')
xlabel('r'), ylabel('t')
%view(-44, 13)
pause(5), close all

[M,N] = size(U); uf = zeros(M,N);
for i = 1 : N
    ri = r(i);
    for j = 1 : M
        uf(j,i) = u(ri,t(j));
    end
end
s = 'Абсолютна похибка |U(R,T)-u(r,t)|';
U = abs(U - uf);
fprintf([s,' в нормі C =%.6e\n'], norm(U,inf))
surfl(R, T, U), colormap('default')
title(s, 'FontWeight', 'bold')
xlabel('r'), ylabel('t')
%view(-44, 13)

function [c, f, s] = koefpde(r, t, u, DuDr)
% коефіциенти рівняння
global aa
c = aa;  f = DuDr;  s = 0;

function u0 = initpde(r)
% початкова умова u0(r) = f0*(1-r/re)
global re f0
u0 = f0.*(1 - r./re);

function [pl, ql, pr, qr] = boundpde(xl, ul, xr, ur, t)
% коефіциенти крайових умов
pl = 0; ql = 1;  pr = ur; qr = 0;

function S = u(r,t)
```

```
% Метод Фур'е
global a re f0 ep
at = -t.*(a.*pi./re).^2;
j = -1; S = 0; Mj = 600;
if r ~= 0
    p = pi.*r./re;
    while j <= Mj
        j = j+2; jj = j.*j;
        e = exp(at.*jj)./(jj.*j).*sin(p.*j);
        S = S + e;
        if abs(e) <= ep
            break
        end
    end
    S = S.*8.*f0.*re./(r.*pi.^3);
else
    while j <= Mj
        j = j+2; jj = j.*j;
        e = exp(at.*jj)./jj;
        S = S + e;
        if abs(e) <= ep
            break
        end
    end
    S = S.*8.*f0./pi.^2;
end
```

Результати її роботи (рис. 10.7–10.8)

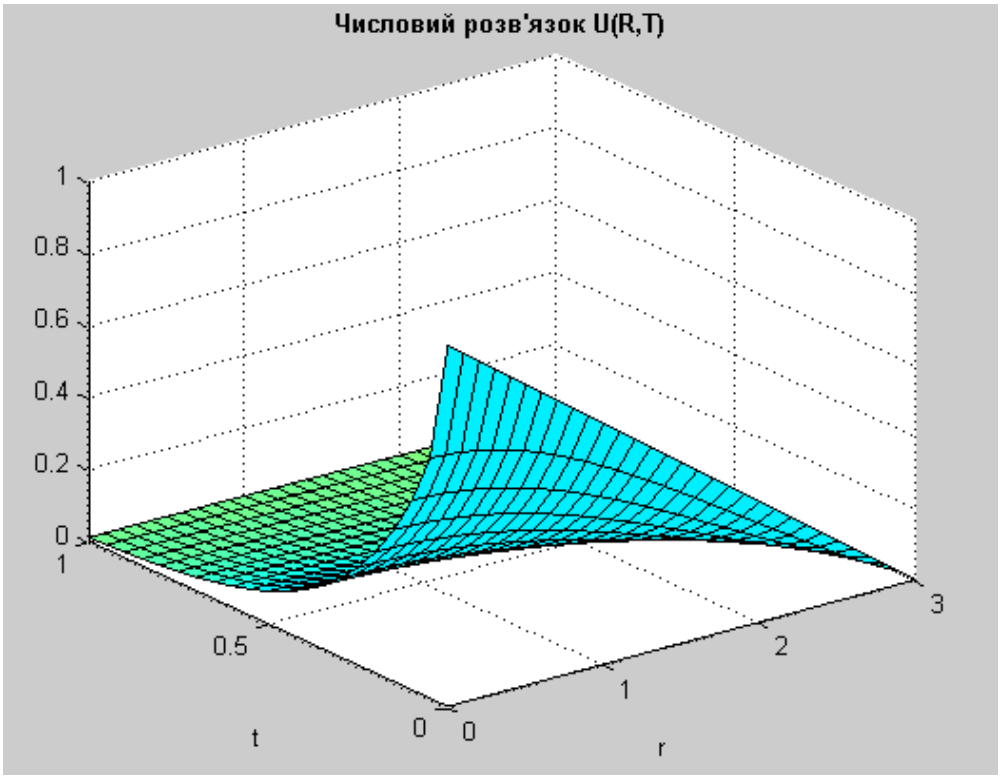


Рис.10.7. Наближений розв’язок задачі U(X,T) солвером *pdepe*.

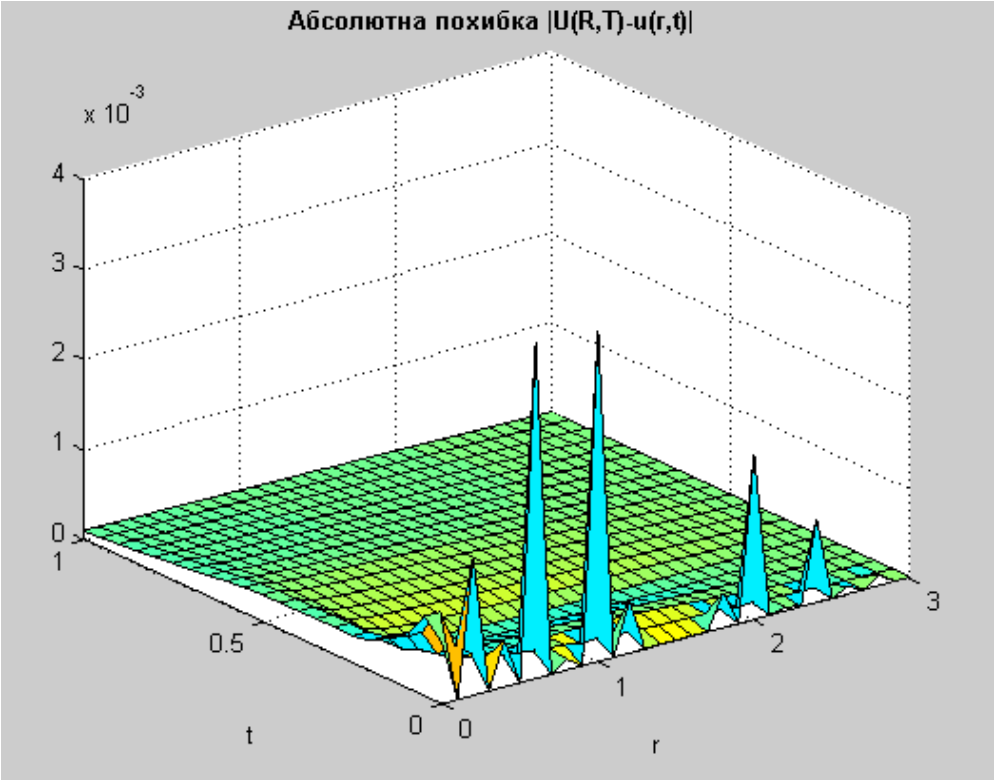


Рис.10.8. Абсолютна похибка числового і розв'язку за методом Фур'є.

```
Абсолютна похибка |U(R,T)-u(r,t)| в нормі C =1.421268e-002
```

Приклади застосування солвера *pdepe* можна знайти, наприклад, у посібнику [3, с. 108-116)].

Математичне моделювання багатьох прикладних задач приводить до диференціальних рівнянь в частинних похідних (ДРЧП) та їх систем. Одним із найпоширеніших наближених методів їх розв'язання є метод скінченних елементів (МСЕ). Тому, крім створення власних, спеціалізованих програм, можна використовувати пакети розширення MATLAB, які багатократно збільшують ефективність використання МСЕ для наближеного розв'язання задач ДРЧП.

Одним із таких пакетів є пакет PDE ToolBox, призначений для розв'язання ДРЧП та їх систем за допомогою МСЕ.

Використання функцій пакету PDE ToolBox вимагає від користувача глибоких знань предметної області задач ДРЧП і МСЕ, але старанно продуманий інтерфейс їх використання суттєво полегшує процедуру наближеного числового розв'язання цих складних математичних задач та його графічне зображення і аналіз отриманих результатів.

Зупинимося на переліку деяких задач, розв'язання яких можна здійснити засобами цього пакету:

- стаціонарні і нестаціонарні задачі теплопровідності;
- задачі дифузії;
- задачі електростатики ;
- двовимірні задачі теорії пружності;
- задачі дифракції;
- розповсюдження акустичних і електромагнітних хвиль;
- знаходження власних коливань конструкцій;

- ламінарні течії рідин і газів;
- задачі теорії пластин і оболонок.

Розглянемо застосування деяких функцій з пакету PDE ToolBox (тут використано приклади, наведені в документації) до розв'язання наступних типових задач для ДРЧП :

еліптичного типу	***гіперболічного типу***	***параболічного типу***
$\begin{cases}\Delta u(x,y)=-1,\ (x,y)\in\Omega;\\ u\big\|_{\partial\Omega}=0.\end{cases}$ $\overline{\Omega}=\overline{G}\setminus g,$ $g=\{-1\le x<0\ \&\ 0<y\le 1\}.$	$\begin{cases}u''_{tt}=\Delta u(x,y,t),\ (x,y)\in G,\ t>0;\\ u\big\|_{\partial G_1}=0,\ u'_x\big\|_{\partial G_2}=0,\ t\ge 0;\\ u\big\|_{t=0}=arctg(\cos(0.5\pi * x)),\\ u'_x\big\|_{t=0}=3\sin(\pi * x)\exp(\cos(\pi * y)),\\ \qquad (x,y)\in G.\end{cases}$ $\partial G=\partial G_1\bigcup\partial G_2,$ $\partial G_1=\{\|x\|=1\&\|y\|\le 1\},\partial G_2=\{\|y\|=1\&\|x\|<1\}.$	$\begin{cases}u'_t=\Delta u(x,y,t),\ (x,y)\in G,\ t>0;\\ u\big\|_{\partial G}=0,\ t\ge 0;\\ u\big\|_{t=0}=\begin{cases}1,\ (x,y)\in R;\\ 0,\ (x,y)\in\overline{G}\setminus R,\end{cases}\\ \qquad (x,y)\in G.\end{cases}$ $R=\{x^2+y^2<0.25\}.$

Тут скрізь $\overline{G}=G\bigcup\partial G=\{\|x\|\le 1\&\|y\|\le 1\}$ – квадрат. Створимо наступний М-файл :

```
%% pl10_5
%Застосування деяких функцій з пакету PDE ToolBox
% до розв'язання задач ДРЧП :
% еліптичного типу
close all, clear all, clc
[p, e, t] = initmesh('lshapeg', 'Hmax', 0.2);
[p, e, t] = refinemesh('lshapeg', p, e, t);
uu = assempde('lshapeb', p, e, t, 1, 0, 1);
%pdesurf(p,t,uu);
pdeplot(p,e,t,'xydata',uu(:,end),'zdata',uu(:,end),...
            'mesh','on','colorbar','off');
pause(2)

% гіперболічного типу
close all, clear all, clc
[p, e, t] = initmesh('squareg');
x = pi.*p(1,:)'; y = pi.*p(2,:)';
u0 = atan(cos(0.5.*x));
ut0 = 3.*sin(x).*exp(cos(y));
nt = 21; tlist = linspace(0, 5, nt);
uu = hyperbolic(u0, ut0, tlist, 'squareb3', p, e, t, 1, 0, 0, 1);
umax = max(uu(:)); umin = min(uu(:));
for i = 1 : nt
    pdeplot(p,e,t,'xydata',uu(:,i),'zdata',uu(:,i),...
            'mesh','on','colorbar','off');
    axis([-1 1 -1 1 umin umax]); caxis([umin umax]);
    pause(1)
end

% параболічного типу
close all, clear all, clc
[p, e, t] = initmesh('squareg');
[p, e, t] = refinemesh('squareg', p, e, t);
u0 = zeros(size(p,2), 1);
ix = find(sqrt(p(1,:).^2+p(2,:).^2)<0.25);
u0(ix) = ones(size(ix));
nt = 11; tlist = linspace(0, 0.07, nt);
```

```
uu = parabolic(u0, tlist, 'squareb1', p, e, t, 1, 0, 1, 1);
umax = max(uu(:)); umin = min(uu(:));
for i = 1 : nt
    pdeplot(p,e,t,'xydata',uu(:,i),'zdata',uu(:,i),...
            'mesh','on','colorbar','off');
    axis([-1 1 -1 1 umin umax]); caxis([umin umax]);
    pause(1.5)
end
```

Результати його роботи (рис. 10.9–10.11)

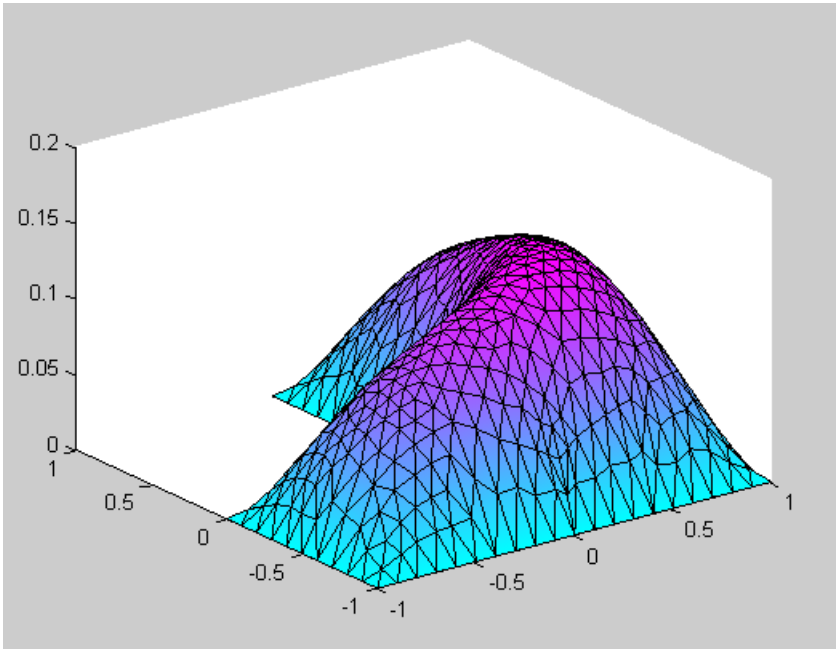
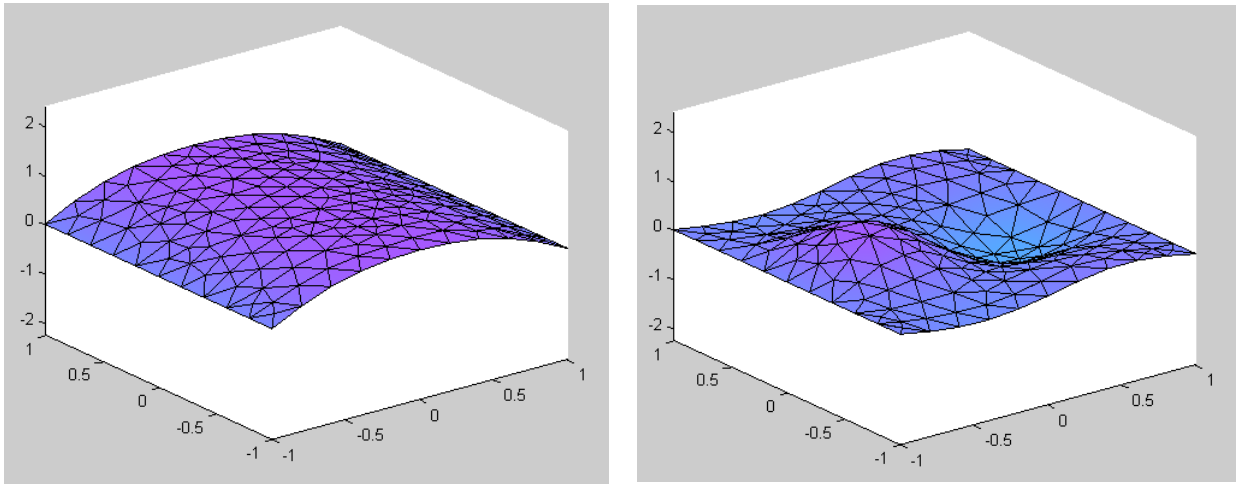
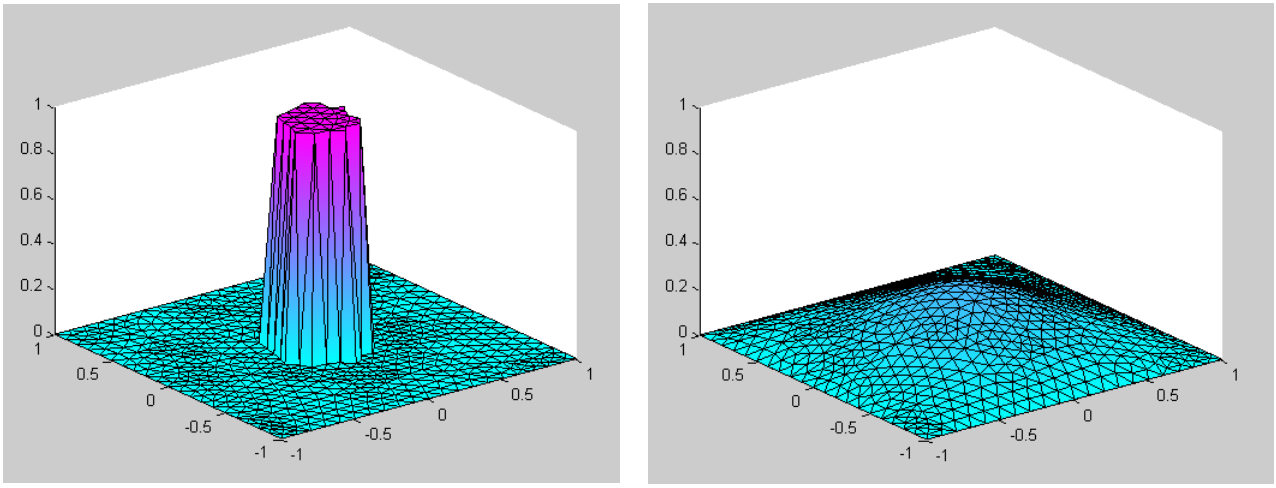


Рис.10.9. Числовий розв'язок крайової задачі для ДРЧП еліптичного типу.



а) $t = 0$ **б) $t = 5$**

Рис.10.10. Числовий розв'язок мішаної задачі для ДРЧП гіперболічного типу.



а) $t = 0$ **б) $t = 0.07$**

Рис.10.11. Числовий розв'язок мішаної задачі для ДРЧП параболічного типу.

В цих демо-прикладах ми, звичайно, не ставили собі мету навчитись використовувати функції `assempde`, `hyperbolic`, `parabolic` і супутні їм функції пакету PDE ToolBox.

Зауважимо, що пакет PDE ToolBox має не тільки можливості виклику відповідних вбудованих функцій, а і свій власний графічний інтерфейс - `pdetool`. Використання його спрощує процес розв'язання задач, адже тут можна інтерактивно означити:

- геометрію області розв'язку, її однозв'язність чи наявність внутрішніх «вирізів»;
- підібрати необхідну розбивку області на елементи (триангуляція області);
- означити необхідні крайові умови на границі області;
- означити коефіцієнти рівняння або рівнянь системи;
- вибрати необхідний солвер для розв'язання задачі;
- керувати виведенням результатів, елементами візуалізації, анімацією.

ЛЕКЦІЯ № 11

СИМВОЛЬНІ ОБЧИСЛЕННЯ В СИСТЕМІ MATLAB

1. Основні поняття та базові операції символьних обчислень

Пакет прикладних програм Symbolic Math Toolbox надає системі MATLAB принципово нові можливості – вирішення завдань у символьному (аналітичному) вигляді, реалізація точної арифметики довільної розрядності. Пакет базується на застосуванні ядра символьної математики однієї з найпотужніших систем комп'ютерної алгебри – Maple. Пакет Symbolic забезпечує: виконання символьного диференціювання та інтегрування, обчислення сум і добутків, розкладання в ряди Тейлора і Маклорена, операції з многочленами, обчислення їх коренів, розв'язання в аналітичному вигляді нелінійних рівнянь і задач для ЗДР, різноманітні символьні перетворення, підстановки і багато іншого. Пакет має команди прямого доступу до ядра системи Maple. Він дозволяє створювати процедури із синтаксисом мови програмування системи Maple і встановлювати їх у системі MATLAB.

Для ознайомлення з пакетом Symbolic можна використати наступні команди:

- *symintro* – початкове знайомство із Symbolic;
- *symcalcdemo* – демонстрація символьних обчислень;
- *symlindemo* – демонстрація застосування пакету Symbolic у задачах лінійної алгебри;
- *symvpademo* – демонстрація операцій арифметики з довільною розрядністю;
- *symrotdemo* – вивчення питань поворотів на площині;
- *symeqndemo* – демонстрація розв'язання рівнянь у символьному вигляді.

Для кожного з цих прикладів наявний М-файл, запуск якого відбувається за допомогою однієї із вказаних команд. Також можна роздрукувати ці файли (*type команда*), або взяти на редагування (*edit команда*) і копіюючи певні фрагменти команд, які зацікавили користувача, виконати їх у командному вікні.

Для отримання переліку доступних у MATLAB команд системи Symbolic Math Toolbox можна звернутися до довідкової системи або ввести команду

```
help symbolic
```

Щоб отримати довідку про символьну команду з пакета Symbolic (при збігу числових і символьних імен) треба використати префікс *sym* перед іменем команди

```
help sym/<ім'я_команди>
```

Щоб отримати довідку про команду системи Maple, необхідно звернутися до довідкової системи за допомогою спеціальної команди

```
mhelp <ім'я_команди>
```

Символьні об'єкти. Для роботи з командами системи Maple у MATLAB визначено новий тип *sym* – символьний об'єкт (symbolic object). Для проведення аналітичних операцій необхідно, щоб відповідні змінні були попередньо оголошені. Група символьних змінних може бути утворена за допомогою команди *syms*. Наприклад, команда

```
syms s1 s2;
```

описує змінні *s1* і *s2*.

Інший спосіб опису символьного об'єкта надає команда *sym*

`>> s3=sym('s3')`	s3 = s3

Ця команда може задати нову символьну змінну, зв'язавши її з певним виразом

`>> s4=sym(exp(s1)*s2+sqrt(2))`	*s4 =* *exp(s1)*s2+2^(1/2)*

або перетворити число у символьну форму у відповідності із параметром `flag`. Тому функція `sym` у загальному вигляді в має такий виклик :

`X = sym(S);` або `X = sym(S, flag);`

Другий аргумент `flag` задається у формі :

- `'f'|'floating point'` – число у форматі MATLAB (із плаваючою точкою);
- `'r'|'rational'` – число в раціональній формі (за угодою);
- `'e'|'estimate error'` – число в раціональній формі з оцінкою відхилення від точного значення за допомогою константи *eps*;
- `'d'|'decimal'` – число в розширеній десятковій формі із кількістю знаків, що визначається поточною величиною параметра `digits` (встановлюється функцією `digits(k)`).;

За допомогою команд `sym`,`syms` реалізується довизначення властивостей символьних змінних. Для цього використовуються атрибути `'real'|'positive'` – діапазон дійсних чисел | діапазон додатніх (`real`) чисел. Опишемо *x* як дійсну змінну `x=sym('x','real')` або `syms x real`. Ці значення параметру впливають на результати операцій, порівняйте:

```
x = sym('x','positive'), y = sqrt(-x),
x = sym('x','real'), y = sqrt(-x)
```

Для зняття накладених обмежень на можливі значення виконаємо команду `sym('x','unreal')` або `syms x unreal`.

У MATLAB відсутня математична нотація для подання формул, подібна тій, що використовується в системі Maple. Для виведення аналітичного виразу *s* в ″математичній формі″ застосовується функція *pretty(s)*, але її використання не є ефективним (в сенсі відображення формули).

Виявити символьні змінні у виразі дозволяє команда `findsym`:

`>> findsym(s4)`	*ans =* *s1, s2*

Символьні операції дозволяють знаходити значення виразів і символьних констант із будь-якою точністю. Зокрема, значення раціональних дробів, функцій від цілих і раціональних аргументів, від числа π тощо можна отримати з будь-якою кількістю значущих цифр результату. Для цього використовується функція `vpa` :

`>> vpa('sqrt(2)',50)`	*ans =* *1.4142135623730950488016887242096980785696718753769*

Другий (необов’язковий) параметр функції `vpa` задає кількість значущих цифр, що виводяться. За замовчуванням утримуються 32 значущі цифри.

Після оголошення символьних змінних можна утворювати символьні вирази (СВ), наприклад, використовуючи необхідні функції, операції "+", "–", "*", "/", "^" та "(", ")". Зауважимо, що MATLAB завжди залишається перш за все матричним процесором, отже для символьних змінних можна вільно застосовувати векторний і матричний запис і відповідні вбудовані оператори і функції. Так, наприклад, для створення символьної матриці необхідно створити символьні змінні, що будуть елементами матриці, а потім створити матрицю, явно визначивши її рядки і стовпці:

`>> syms a b c;` `>> A=[a b c;b c a;c a b]`	*A =* *[a, b, c]* *[b, c, a]*

	[*c*, *a*, *b*]

Інший спосіб побудови матриці:

`>> A=sym('[a b c;b c a;c a b]')`	*A* = [*a*, *b*, *c*] [*b*, *c*, *a*] [*c*, *a*, *b*]

Команду *sym* можна використати також для створення СВ із абстрактною функцією *ff(x)* (опис такої функції відсутній у поточному М-файлі або в підключених директоріях). Отриманий вираз можна використати для побудови нових СВ. Наприклад, створимо СВ, що повертає вираз правої скінченної різниці (для довільних *x*, *h*), а для числового значення цього СВ використаємо підстановку конкретних значень *x*, *h* і реальну функцію *ff*.

Оформимо потрібну послідовність команд у вигляді М-функції:

`function pl11_1` `clear; clc; z = sym('ff(x)')`	`z =` `ff(x)`
`syms x h;` `rdz=(subs(z,x,x+h)-z)/h`	`rdz =` `(ff(x+h)-ff(x))/h`
`nh=0.01;nx=pi;ff=@(t)(sin(t));` `s=subs(subs(rdz,h,'nh'),x,'nx')`	`s =` `(ff(nx+nh)-ff(nx))/nh`
`es=vpa(s)`	`es =` `(ff(nx+nh)-1.*ff(nx))/nh`
`en=eval(s)`	`en =` `-0.99998`

Символьні операції з виразами. В таблиці наведено деякі прості операції, за допомогою яких здійснюються аналітичні перетворення над символьними об'єктами.

Аналітичні операції з виразами	
Назва	**Призначення**
`expand(S)`	розкриття дужок у виразі *S*
`factor(S)`	розклад виразу *S* на множники
`simplify(S)`	спрощення виразу *S*
`e=simple(S);` `[e,how]=simple(S)`	спрощення виразу *S* з переліком варіантів
`subs(S,ov,nv)`	заміна у виразі *S* кожного входження символьної змінної *ov* змінною *nv*
`[r,s]=subexpr(t,s)`	перетворення СВ *t* у вираз *r* за допомогою підстановки *s*
`collect(S,var)`	перетворення виразу *S* в поліном з обчисленням коефіцієнтів при степенях незалежної змінної *var*
`[n,d]=numden(S)`	перетворення виразу *S* в раціональну форму як відношення двох незвідних поліномів *n* і *d* з цілочисельними коефіцієнтами
`horner(P)`	перетворення символьного полінома *P* за схемою Горнера

Нижче наведено текст М-функції *pl11_2*, що використовує деякі з цих функції (зверніть увагу на отримані значення *sn*, *ss*):

`function pl11_2`
`clear; clc; syms x y;`
`a = 5; b = -1; c = 4;`
`S = (x+a)*(x+b)^2*(y-c)`
	`S =`
	`(x+5)*(x-1)^2*(y-4)`

`S1 = expand(S)`	`S1 =` `x^3*y-4*x^3+3*x^2*y-12*x^2-9*x*y+36*x+5*y-20`
`S2 = subs(S1,y,1)`	`S2 =` `-3*x^3-9*x^2+27*x-15`

`S3 = diff(S2,x)`	`S3 =` `-9*x^2-18*x+27`
`S4 = factor(S3)`	`S4 =` `-9*(x+3)*(x-1)`
`S5 = collect(S4,x)`	`S5 =` `-9*x^2-18*x+27`
`S6 = horner(S5)`	`S6 =` `27+(-18-9*x)*x`
`sn = subs(S5,x,1/3)`	`sn =` `20`
`ss = subs(S5,x,'1/3')`	`ss =` `20`

Приклад 25. Для функції $u(x)=\sin(\pi * x),\ x\in[0,1]$ за m=3 точками побудувати інтерполяційний поліном у формі Лагранжа, потім перетворити його у форму Ньютона і побудувати відповідні графіки.

Нижче наведено текст відповідного М-файла

```
%%  pl11_3.m
clear; clc
syms x;
a = 0; b = 1; m = 3;
X = linspace(a, b, m); Y = sin(pi .*  X);
% Побудова інтерполяційного поліному у формі Лагранжа
w = prod(x - X); w1 = diff(w, x);
L = 0;
for k = 1 : m
   L = L + Y(k) .* (w / (x - X(k)) / subs(w1, x, X(k)));
end
L
% Побудова інтерполяційного поліному у формі Ньютона
N = expand(L); N = horner(N)
Nn = vpa(N,10)
XX = linspace(a, b, 10 * m + 1);
YL = subs(L, x, XX); YNn = subs(Nn, x, XX);
plot(XX, YL, XX, YNn, X, Y, 'b*');
legend('ІП ф.Лагранжа','ІП ф.Ньютона(ч)',...
   'точний розв''язок','Location','Best');
xlabel('x'); ylabel('sin({\pi}*x)');
grid on;
```

Отримаємо такий результат (звертаємо увагу на значення N і Nn):

```
 L =
 -4*x*(x-1)+4967757600021511/2028240960365167042394725128601 6*x*(x-1/2)
 N =
 (16225927682921335842382041026 6617/40564819207303340847894502572032-
81129638414606676728031405122553/2028240960365167042394725128601 6*x)*x
 Nn =
 (4.000000000-4.000000000*x)*x
```

Символьні операції математичного аналізу. Основні команди, що реалізують низку важливих операцій математичного аналізу, наведено в таблиці:

Команди математичного аналізу	
Назва	**Призначення**
`diff(S),diff(S,y),` `diff(S,n),diff(S,x,n)`	Обчислення *n*–тої похідної (за замовчуванням – першої) від СВ S за незалежною змінною x
`jacobian(F,v)`	Обчислення матриці Якобі для вектор-функції F (стовпця)

	за вектором змінних *v* (рядком)
`int(F)`, `int(F,x)`, `int(F,a,b)`, `int(F,x,a,b)`	Обчислення інтегралу від символьної функції *F*. Тут *x*, *a*, *b* – необов'язкові змінні: *x* – змінна інтегрування, *a*, *b* – межі інтегрування
`limit(F)`, `limit(F,a)`, `limit(F,x,a)`, `limit(F,x,a,'left')`, `limit(F,x,a,'right')`	Обчислення границі символьно заданої функції *F*. Тут *a* – граничне значення незалежної змінної (за замовчуванням *a* = 0). Необов'язкові параметри '*left*', '*right*' указують, що обчислюється границя функції зліва або справа при прямуванні *x* до *a*
`symsum(S)`, `symsum(S,x)`, `symsum(S,a,b)`, `symsum(S,x,a,b)`	Обчислення суми нескінченного ряду за змінною *x*. У випадку скінченного ряду задаються початкове і кінцеве значення номера члена ряду *a* і *b*
`taylor(F)`, `taylor(F,a)`, `taylor(F,n)`, `taylor(F,x)`, `taylor(F,n,x,a)`	Розкладення символьно заданої функції *F* у ряд Тейлора. Для будь-якого варіанту розкладу можна задавати точку *a* і змінну *x*, відносно яких записується розклад, а також число членів ряду (розклад містить члени ряду до *n*–1-го порядку). За замовчуванням обчислюється розклад у точці *a* = 0 до п'ятого степеня (*n* = 6)
`g=finverse(f)`, `g=finverse(f,x)`	Знаходження функції, оберненої до *f*, відносно заданої змінної *x*. Для функції однієї змінної параметр *x* може бути опущено
`compose(f,g)`	Суперпозиція функцій. Результатом є функція *f*(*g*(*y*)), де *f*=*f*(*x*), *g*=*g*(y)

Приклад 26. За допомогою метода Ньютона знайти наближений розв'язок нелінійної системи рівнянь

$$\begin{cases} x^2 y^2 = 1, \\ x - \dfrac{y}{2} = 2. \end{cases} \qquad (11.1)$$

Нижче наведено текст М-функції `pl11_4` і зовнішньої функції `njuts`:

```
function pl11_4
% Розв'язання системи нелінійних рівнянь,
% заданої в символьному виді за допомогою
% наближеного методу Ньютона.
close all, clear all, clc
syms x y;
global v
v = [x, y];
% Побудова графіка для системи рівнянь
F = funf(v);
ezplot(F(1),[-5,5,-5,5]); hold on;
ezplot(F(2),[-5,5,-5,5]); grid on; pause;
X0 = input('?наближення до коренів ([x0,y0;...])=\n');
[n,m] = size(X0);
% уточнення наближень до коренів
ep = 1e-10; mki = 100;
s = ['Ітер Корені: x',blanks(12),'y',blanks(7),'max(Fi(x,y))'];
fprintf('%s\n', s);
for k = 1 : n
    X = X0(k,:)';
```

```
    [X, kit, FX] = njuts(@funf_J, X, ep, mki);
    fprintf('%4u %13.8f %13.8f %.8e\n', kit, X, max(FX));
end

function F = funf(z)
% Опис нелінійних рівнянь системи
    F = [z(1)^2*z(2)^2-1;
         z(1)-z(2)/2-2];

function [F, dF] = funf_J(z)
% Обчислення функції і якобіана
    F = funf(z);
    dF = jacobian(F, z);


function [X, k, varargout] = njuts(f_J, X, e, mki)
% Метод Ньютона знаходження розв'язку системи
%   для символьно заданої нелінійної СЛ   F(X)=0
global v
varargout = {};
n = length(X); [F, J] = f_J(v);
for k = 1 : mki
   A = J; b = F;
   for i = 1 : n
      A = subs(A, v(i), X(i));
      b = subs(b, v(i), X(i));
   end
   d = A\b; X = X - d;
   if norm(d) < e
      if nargout > 2
         b = F;
         for i = 1 : n
            b = subs(b, v(i), X(i));
         end
         varargout = {b};
      end
      return
   end
end
```

Отримаємо такий результат (графік на рис.11.1):

```
?наближення до коренів ([x0,y0;...])=
[-0.2,-4.5; 0.2,-3.4; 1.8,-0.7; 2.2,0.5]
Ітер Корені: x            y       max(Fi(x,y))
   4   -0.22474487   -4.44948974 0.00000000e+000
   5    0.29289322   -3.41421356 0.00000000e+000
   5    1.70710678   -0.58578644 0.00000000e+000
   4    2.22474487    0.44948974 2.22044605e-016
```

Приклад 27. Знайти похибку апроксимації ($u_{\bar{x}x,i} - u''(x_i)$) функції $u''(x_i)$ другою різницевою похідною $u_{\bar{x}x,i} = \dfrac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}$, використовуючи розклад $u(x_i \pm h)$ в ряд Тейлора по h.

```
%%  pl11_5.m
clear; clc
syms x h;
zmh = sym('u(x-h)');
```

```
z0h = sym('u(x)');
zph = sym('u(x+h)');
sm = taylor(zmh, 7, x, h);
sp = taylor(zph, 7, x, h);
Oh = simplify((sm-2*z0h+sp)/h^2-diff(z0h,x,2));
Oh = subs(Oh, x, 'Xi'); Oh = collect(Oh);
z = char(Oh); k = findstr(z,'h'); z = z(k(end-1)+3:end);
Oh = simple(sym(z))
```

Отримаємо такий результат:

```
Oh =
1/12*@@(D,4)(u)(Xi)*h^2
```

Приклад 28. Проілюструємо застосування функцій математичного аналізу. Нижче наведено текст відповідного М-файла.

```
%%  pl11_6.m
clear; clc
syms x a b n;
% Г.М.Фихтенгольц "Курс диф.и инт.исч", т.1, стр.65
u = sqrt(n)*(sqrt(n+1)-sqrt(n))
limu = limit(u,n,inf)
clear u limu
% Г.М.Фихтенгольц "Курс диф.и инт.исч", т.1, стр.153
u = a^(1/x), ua = subs(u, a, 6);
limul = limit(ua, x, 0, 'left')
limur = limit(ua, x, 0, 'right')
u = subs(u, a, [3, 6, 13, 17, 25]);
for i = 1 : length(u)
   ezplot(u(i), [-15,15]); hold on;
end
grid on; pause; close;
clear u ua limul limur i
% Г.М.Фихтенгольц "Курс диф.и инт.исч", т.1, стр.238
u = x^2*cos(a*x)
up50 = diff(u, x, 50)
clear u up50
% Г.М.Фихтенгольц "Курс диф.и инт.исч", т.2, стр.30
u = 1/sqrt((x-a)*(b-x))
iu = simplify(int(u, x))
U = simplify(diff(iu)) % перевірка
clear u iu U
% Г.М.Фихтенгольц "Курс диф.и инт.исч", т.2, стр.136
u = sin(x)/(1+cos(x)^2)
iu = simplify(pi/2*int(u, x, 0, pi))
clear u iu
% Г.М.Фихтенгольц "Курс диф.и инт.исч", т.2, стр.325
u = n*x^(n-1)
S = symsum(u, n, 1, inf)

ezplot(S, [-15,15]); grid on; pause; close;
clear u S
% просто приклади
u = exp(x), Ou = finverse(u), S = compose(Ou, u)
ezplot(u, [-4, 4]); grid on; pause; close;
ezplot(Ou, [0.01, 4]); grid on; pause; close;
ezplot(S); grid on; pause; close;
clear u Ou S
```

ЛЕКЦІЯ № 12

СИМВОЛЬНІ ОБЧИСЛЕННЯ В СИСТЕМІ MATLAB (продовження)

2. Символьні операції з матрицями

Для алгебраїчних операцій з СВ використовуються команди з іменами, аналогічними тим, що застосовуються для числових даних. Проте, виявив, що аргументом команди є символьний об'єкт, MATLAB запускає відповідну процедуру пакету Symbolic Math Toolbox.

Функції для операцій з елементами матриць	
Назва	**Призначення**
diag	створення або видобування діагоналі матриці
tril	формування лівої трикутної матриці
triu	формування правої трикутної матриці
inv	утворення оберненої матриці
det	обчислення визначника (детермінанта) квадратної матриці
rank	обчислення рангу матриці
jordan	обчислення канонічної форми Жордана
rref	зведення матриці до верхньої трикутної форми
null	нуль-простір матриці
eig	обчислення власних значень і власних векторів матриці
svd	сингулярний розклад матриці
poly	обчислення характеристичного полінома матриці
expm	обчислення матричної експоненти

Враховуючи, що ці функцій неодноразово використовувалися із числовими даними, покажемо застосування деяких з них.

Приклад 29. Знайти:

а) загальне рівняння площини в просторі, яка проходить через точки $M_i(x_i, y_i, z_i),\ \ i = 1, 2, 3$;

б) нормальний вектор площини;

в) відповідь на питання: чи проходить площина через точку $O(0, 0, 0)$.

Нижче наведено текст відповідного М-файла.

```
%%  pl12_1.m
clear; clc
syms x y z;
n = 3; M = [1, 2, 3; -4, 5, 6;7, -8, 9];
% Чи лежать точки на одній прямій ?
p = (M(3,:)-M(1,:))./(M(2,:)-M(1,:));
if p-p(1)==0
   disp('Точки лежать на одній прямій !')
else
   p = [x,y,z]; A = [p,1;M,ones(n,1)];
   d = det(A); P = sym(strcat(char(d),'=0'));
   disp('Рівняння площини :'); disp(P);
   N = zeros(1,n); A = eye(n); D = subs(d, p, N);
   for k = 1 : n
      N(k) = simplify(subs(d, p, A(k,:)) - D);
   end
```

```
    disp('Координати нормального вектора площини :'); disp(N);
    if D
       b = ' не';
    else
       b = '';
    end
    b = strcat('Площина',b,' проходить через O(0,0,0)');
    disp(b);
end
```

Отримаємо такий результат:

```
Рівняння площини :
48*x+48*y+32*z-240=0
Координати нормального вектора площини :
     48    48    32
Площина не проходить через O(0,0,0)
```

Приклад 30. Застосування деяких функцій для роботи з матрицями.

Нижче наведено текст відповідного М-файла.

```
%%  pl12_2.m
clear; clc
syms a;
% Показати, що Жорданові матриці
%   для матриць A і A' однакові
A = [2, 0, 1, 5; 1, 0, a, 3; 1, 0, 2, -1;1, 1, 1, 1]
Ja = jordan(A)
B = A.';
Jb = jordan(B)
if all(Ja == Jb)
    disp('Так, однакові)
else
    disp('Ні, неоднакові)
end
clear Ja Jb
% Показати, що для трикутної матриці A розмірності n
%  з нульовими діагональними елементами маємо A^n = 0
B = triu(A, 1);
C = B^length(B)
```

3. Розв’язання рівнянь і систем рівнянь в символьному виді

Для аналітичного розв’язання алгебраїчних рівнянь використовується команда `solve`, виклик якої має вигляд

```
solve(ex1,ex2,...,exN,var1,var2,...,varN)
```

Тут `ex1,ex2,...,exN` – завдання рівнянь у виглядів рядків, `var1,var2,...,varN` – невідомі (символьні змінні). При розв’язанні системи рівнянь, для збереження результату, створюється структура, поля якої відповідають іменам невідомих. Можна також використовувати варіант виклику з невідомими в якості вихідних параметрів:

```
[var1,var2,...,varN] = solve(ex1,ex2,...,exN)
```

Приклад 31. Знайти розв'язок :

а) системи (11.1);

б) системи

$$\begin{cases} \cos(0.4y + x^2) + x^2 + y^2 = 1.6, \\ 1.5x^2 - \dfrac{y^2}{0.36} = 1. \end{cases} \tag{11.2}$$

в) системи

$$\begin{cases} x^2 + 2y^2 - 3z^3 = 3, \\ 8x^3 - 3x^2 y + z = 77, \\ xyz^2 + 8/x = 2. \end{cases} \tag{11.3}$$

Нижче наведено текст М-файла для знаходження розв'язків систем (11.1)–(11.3).

```
%%  pl12_3.m
close all, clear all, clc

% Система а)
e1 = 'x^2*y^2-1';
e2 = 'x-y/2-2';
D = [-5, 5, -5, 5];
f1 = ezplot(e1, D); hold on;
f2 = ezplot(e2, D); grid on
set(f1,'color',' red','LineWidth',2);
set(f2,'color','blue','LineWidth',2);
title([e1,'; ',e2]);
pause(5)
[x, y] = solve(e1, e2);
s1 = ' Розв''язок системи ';
s2 = '----------x-------|----------y-------\n';
s = strcat(s1, ' а)\n', s2);
fprintf(s);
n = length(x);
for k = 1 : n
 fprintf('%17.8f |%17.8f\n', double(x(k)), double(y(k)));
end
pause(5), close all

% Система б)
e1 = 'cos(0.4*y+x^2)+x^2+y^2-1.6';
e2 = '1.5*x^2-y^2/0.36-1';
D = [-2, 2, -2, 2];
f1 = ezplot(e1, D); hold on
f2 = ezplot(e2, D); grid on
set(f1,'color',' red','LineWidth',2);
set(f2,'color','blue','LineWidth',2);
title([e1,'; ',e2]);
S = solve(e1, e2, 'x', 'y');
s = strcat(s1,' б)\n',s2);
fprintf(s);
n = length(S.x);
for k = 1 : n
 fprintf('%17.8f |%17.8f\n', double(S.x(k)), double(S.y(k)));
end
pause(5), clear all, close all

% Система в)
[x, y, z] = solve('x^2+2*y^2-3*z^3-3=0',...
                  '8*x^3-3*x^2*y+z-77=0',...
                  'x*y*z^2+8/x-2=0');
s1 = ' Розв''язок системи в)\n';
```

```
s2 = '----------x-------|----------y-------|----------z-------\n';
s = strcat(s1, s2);
fprintf(s);
n = length(x);
for k = 1 : n
 fprintf('%17.8f |%17.8f |%17.8f\n', double(x(k)),...
 double(y(k)), double(z(k)));
end
```

Нижче наведено результат роботи М-файла *pl12_3*.

```
          Розв'язок системи а)
------------x-------|------------y-------
         1.70710678 |       -0.58578644
         0.29289322 |       -3.41421356
         2.22474487 |        0.44948974
        -0.22474487 |       -4.44948974
```

Аналізуючи графік і чисельний розв’язок системи (11.1) робимо висновок, що всі її корені знайдено.

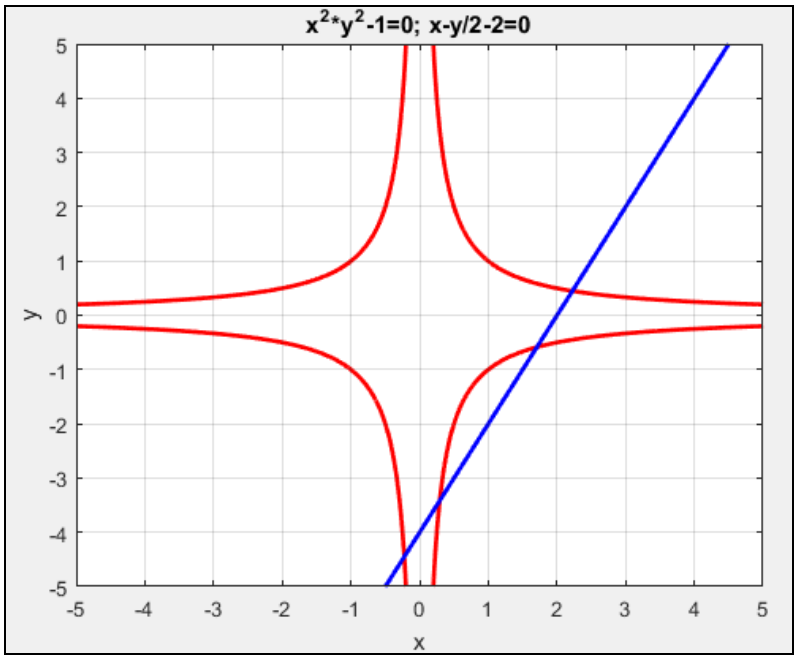


Рис. 11.1. Графічне розв'язання системи (11.1)

```
          Розв'язок системи б)
------------x-----|-----------y--------
      -0.87456548 |       -0.23027589
       0.87456548 |       -0.23027589
```

Аналіз графіка і чисельного розв’язка системи (11.2) показує, що не всі корені можуть бути знайдені вбудованою функцією *solve*.

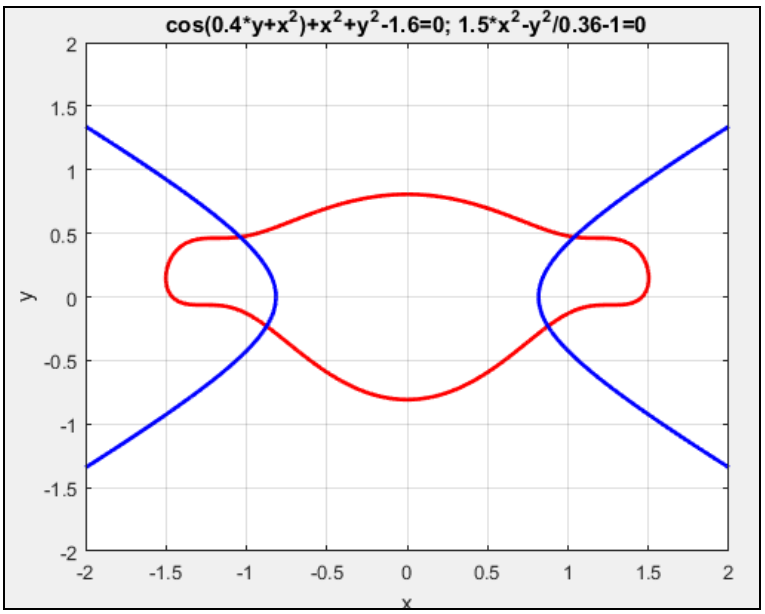


Рис. 11.2. Графічне розв'язання системи (11.2)

Частина (всього 25 рядків) зі знайдених розв'язків системи (11.3) (перший корінь легко перевіряється !):

```
          Розв'язок системи в)
--------x------|-----------y------|-----------z--------
    2.00000000 |      -1.00000000 |       1.00000000
    0.00127683 |    -827.46179873 |      76.99595295
    2.21246671 |       0.64475439 |      -0.32685410
    2.05779696 |      -0.56761935 |      -0.78278359
```

4. Розв'язання звичайних диференціальних рівнянь і СЗДР

Для розв'язання ЗДР і систем ЗДР в MATLAB використовується функція `dsolve`. Команда

`dsolve(e1,e2,…,c1,c2,…)`

знаходить загальний розв'язок ЗДР, а за наявності додаткових умов (початкових або крайових) – частинний.

ДР задаються рівностями `e1,e2,…` у вигляді рядків (обов'язкові вхідні параметри), потім – додаткові умови (необов'язкові вхідні параметри). За замовчуванням незалежною змінною вважається *t*, але можна використовувати й іншу змінну, додавши її в кінець списку параметрів команди `dsolve`.

Для позначення похідних у рівняннях використовується символ `D`, а комбінації `D2,D3,…` застосовуються для позначення другої, третьої, … похідних (`D2u,D3u,…`).

Додаткові умови задаються у вигляді рівностей, наприклад,

$$'u(a)=\mu_a',\ 'Du(a)=\mu_a',\ 'u(b)+Du(b)=\mu_b',$$

де `u(t)` – шукана функція від незалежної змінної; `a`, `b`, μ_a, μ_b – константи. Якщо кількість умов менша, ніж кількість ДР, то в розв'язку будуть присутні довільні константи з іменами `C1,C2,….` .

Розв'язок (вихідний параметр `res`) це змінна (або масив), яка в залежності від розв'язків (декілька для одного рівняння) і кількості ДР може бути :

- «скаляр» | вектор-стовпець у випадку одного рівняння,
- вектор-рядок | структура, де елементи | поля – це імена шуканих функцій у випадку системи ([`u,v,w`] | `res.u,res.v,res.w`).

Нижче наведено приклади застосування функції *dsolve* для розв'язання СЗДР.

```
%%  pl12_4.m
clear; clc
%   A.M.Самойленко,C.A.Кривошея,Н.А.Перестюк
% "Дифференциальные уравнения:примеры и задачи",стр.172 Пр.2.72
S = dsolve('Dy=u','Du=v','Dv=3/t*v-6/t^2*u+6/t^3*y+sqrt(t)'),
S.y
%    -#- ,стр.128 Пр.2.16
S = dsolve('Dy=u','Du=-(u^2+u^4)/(2*y)','y(0)=1','u(0)=2');
simplify(S.y)
%    -#- ,стр.276 Пр.4.31
[x, y] = dsolve('Dx=1-2/t*x','Dy=(1+2/t)*x+y-1');
x
y
```

Отримаємо наступні результати (зазначимо, що для Пр.2.16 знайдено і комплексний розв'язок):

```
ans =
8/15*t^(7/2)+1/6*t^3*C1+1/2*t^2*C2+t*C3
ans =
                      4/5+1/5*(15*t+1)^(2/3)
1/2*(16+(-84*t-12*i*t*3^(1/2)-8)^(2/3))/(7+i*3^(1/2))
x = =
1/3*t+1/t^2*C2
y =
-1/3*(t^3+3*C2-3*exp(t)*C1*t^2)/t^2
```

5. Засоби візуалізації символьних обчислень

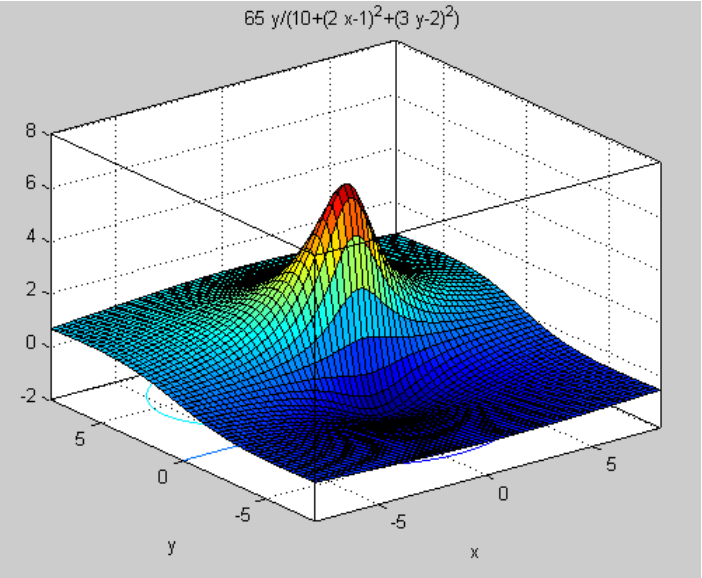
Розширення MATLAB командами Maple призвело до появи нових графічних функцій :

Графічні команди пакета Symbolic Math	
Назва	**Призначення**
ezplot	побудова графіка
funtool	графічний калькулятор функції однієї змінної
ezpolar	побудова графіка в полярних координатах
ezcontour	побудова контурних графіків (ліній рівня)
ezcontourf	побудова контурних графіків (ліній рівня) з заповненням
ezmesh	побудова сітчастої поверхні
ezmeshc	побудова сітчастої поверхні з проекцією ліній рівня
ezsurf	побудова поверхні
ezsurfc	побудова поверхні з проекцією ліній рівня
ezplot3	побудова тривимірних кривих

Деякими з цих функцій (*ezplot*) ми активно користувалися у попередніх прик-ладах. Приведемо приклад на використання функції *ezsurfc :*

```
%%  pl12_6.m
% побудова поверхні з проекцією ліній рівня
clear all, close all, clc
X = linspace(-8,8,81);
ezsurfc('65*y/(10+(2*x-1)^2+(3*y-2)^2)',[X,X]);
box on;
```

і його виконання :



Використання інших вбудованих функцій можна знайти в посібниках [2-4, 6–10]. Серед цих функцій є інтерактивна – графічний калькулятор функцій однієї змінної. Виклик `funtool` виводить три графічні вікна: в перших двох відбувається виведення графіків функцій *f*, *g*. Третє вікно містить інтерфейс користувача, а саме :

- рядки введення:
  - опису функцій *f*, *g*,
  - інтервалу представлення незалежної змінної *x*,
  - параметра перетворень *a*;
- систему кнопок для виконання різних операцій побудови і перетворення функцій.

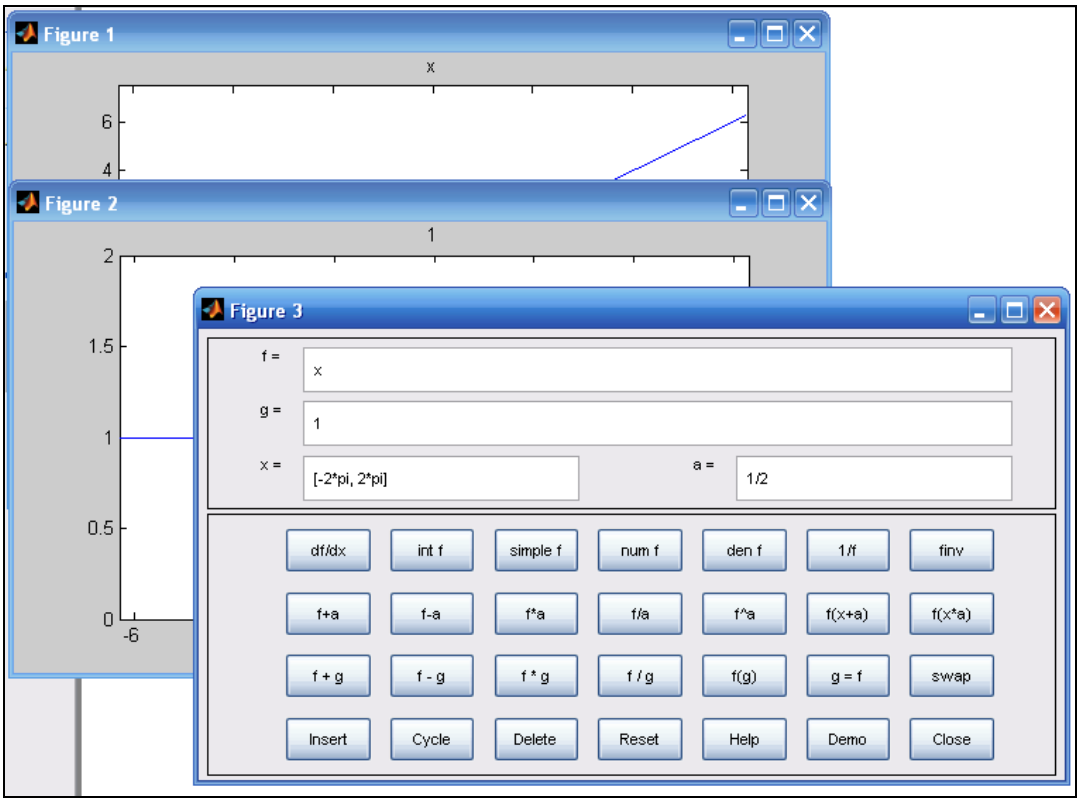


Рис. 11.3. Вікна графічного калькулятора

Практичне заняття № 01

1.1. ВСТУП ДО СИСТЕМИ MATLAB

Призначення і сфера застосування математичного пакета MATLAB. Для розв'язання прикладних задач важливим є застосування ефективних засобів автоматизації обчислень від простих до досить громіздких, а також відображення отриманої інформації в різноманітних формах, зокрема у вигляді таблиць, діаграм, 2D, 3D-графіків і аналітичних виразів. Сучасне програмне забезпечення ПЕОМ має досить різноманітні, потужні математичні пакети, наприклад, MATLAB, MATHEMATICA, MAPLE, Mathcad, які використовуються у математичних обчисленнях і аналітичних перетвореннях. Пакет MATLAB є зручною в користуванні системою, містить велику кількість вбудованих функцій і спеціалізованих додатків, застосування яких дає змогу чисельно і аналітично розв'язувати складні задачі математики, фізики, хімії, інженерії, фінансів [1-4,8,14, 15, 19, 23, 24]. Крім вбудованих засобів, пакет можна доповнювати власними розробками, використовуючи як власну мову MATLAB (M-програми), так і зовнішні алгоритмічні мови, наприклад, C#, JAVA, FORTRAN. Підсумовуючи, охарактеризуємо спектр задач, які розв'язуються за допомогою пакета MATLAB:

- проведення математичних досліджень, що вимагають обчислень і аналітичних розрахунків;
- розробка і аналіз алгоритмів;
- математичне моделювання, комп'ютерний експеримент;
- аналіз і обробка даних;
- візуалізація, наукова та інженерна графіка;
- розробка графічних і розрахункових додатків.

Перше знайомство з роботою в системі. Для практичної роботи будемо використовувати програмний комплекс MATLAB 7.0.0 (R14), робота в якому досить легка та комфортна і може проводитись в двох режимах – *інтерактивному* (виконання команд користувача і отримання результата) і *програмному* (запуск відповідного програмного M-файлу і виконання його в режимі інтерпретації).

Завантаження встановленого пакета в ОС Windows відбувається стандартним чином (клацнувши мишею по ярлику програми MATLAB), при цьому на екрані відкривається система меню, панель інструментів, рядок стану та три основних вікна: *робоча область* (*Workspace*), *вікно історії* (*Command History*), *командне вікно* (*Command Window*) (рис. 1.1).

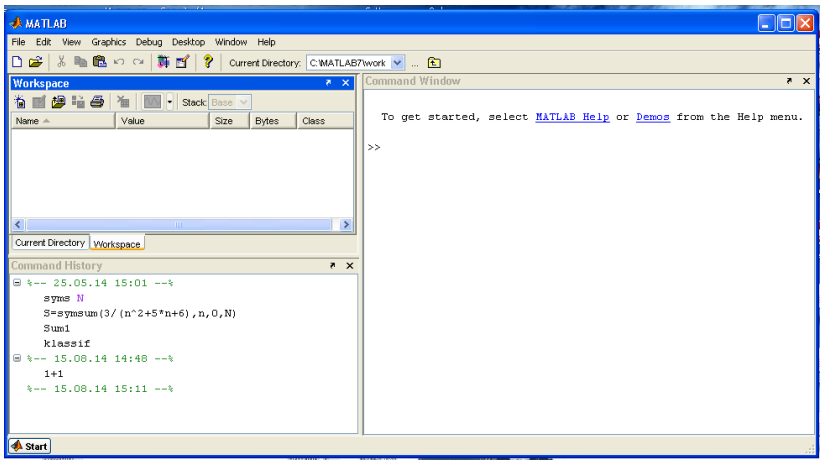


Рис. 1.1. Головне вікно системи

При відкритті файла з розширенням ***.m***, також відбувається завантаження

(встановленого!) пакета MATLAB, при цьому додатково до перерахованого вище, на екрані монітора появляється *вікно вбудованого редактора текстів* з завантаженим файлом і відбувається налаштування системи на поточну директорію, з якої було завантаження файла.

Наявність, перелік і структуру вікон можна змінювати з метою створення зручнішого інтерфейсу при розв'язанні конкретних задач. Наприклад, можна тимчасово прибрати з екрану вікно робочої області і вікно історії, або додатково до стандартних вікон активізувати вікно *поточного каталогу* (*Current Directory*) (рис. 1.2).

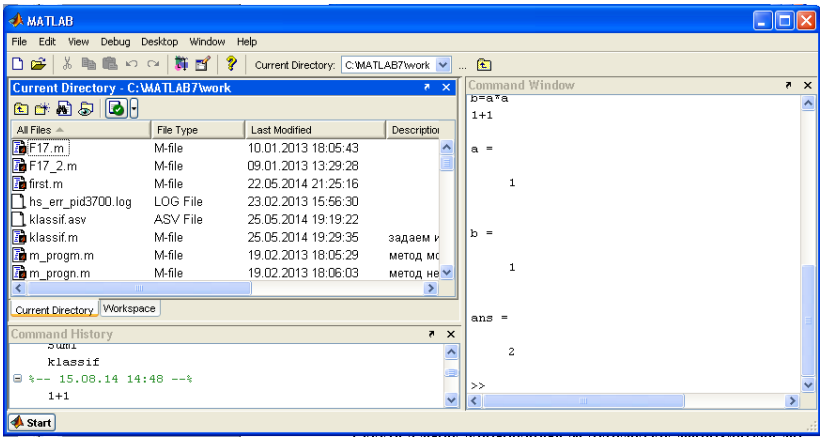


Рис. 1.2. Вікно поточного каталогу

Робота з меню здійснюється за допомогою маніпулятора миша та набору певних клавіатурних комбінацій ("гарячих клавіш"). Через меню виконуються найбільш загальні дії системи MATLAB. Меню має стандартний вигляд і організацію, властиві багатьом програмним продуктам ОС Windows. Уміння працювати з меню може істотно полегшити діалог користувача з комп'ютером.

Панель інструментів містить кнопки, що дозволяють створювати нові або відкривати існуючі файли, виконувати операції з буфером обміну, звертатися до довідкової системи MATLAB тощо. Кнопки панелі інструментів забезпечують виконання більшості необхідних команд розв'язання математичних задач. Спливаючі підказки при зверненні до кнопок повідомляють про їх зміст.

У рядку стану розташована кнопка [**Start**], яка забезпечує зручний доступ до всіх спеціальних додатків MATLAB. Поруч із цією кнопкою розташований індикатор, що відображає стан обчислювального ядра MATLAB. У момент проведення розрахунків у цьому рядку відображається слово **Busy** (Зайнято).

У вікні робочої області відображаються імена змінних, уведених користувачем, та їх поточні значення і певні характеристики.

Вікно історії використовується для зберігання та відображення раніше виконаних команд користувача. При необхідності можливе їх подальше використання.

Інтерактивний сеанс роботи в командному вікні MATLAB прийнято іменувати *сесією* (session). Сесія, по суті, є поточним документом, що відображає роботу користувача із системою. У ній є рядки введення даних, результатів і повідомлень про помилки. За допомогою команд сесії користувач взаємодіє із середовищем пакета. Поіменовані дані поточної сесії, розташовані в робочій області пам'яті (але не саму сесію), можна записати на диск у вигляді файла формату ***.mat***, використовуючи команду *save* (зберегти). Командою *load* (завантажити) можна відновити з диска дані робочої області. Фрагменти сесії можна зберегти в певному документі за допомогою команди *diary*.

При роботі з MATLAB у командному режимі діє найпростіший рядковий редактор. Команди, які використовуються в редакторі, мають, як правило, відповідні комбінації клавіш для їх виклику. Ці комбінації наведені нижче:

Комбінації клавіш для команд *рядкового редактора*	
Клавіша	**Призначення**
<→> або <Ctrl+B>	на позицію вправо
<←> або <Ctrl+F>	на позицію вліво
<Ctrl + →> або <Ctrl+R>	на слово вправо
<Ctrl + ←> <Ctrl+L>	на слово вліво
<Home> або <Ctrl+A>	на початок рядка
<End> або <Ctrl+E>	на кінець рядка
<↑> або <Ctrl+P>	на попередню команду вгору
<↓> або <Ctrl+N>	на наступну команду вниз
<Del> або <Ctrl+D>	видалити символ справа
<BkSp> або <Ctrl+H>	видалити символ зліва
<Ctrl+K>	видалити текст до кінця рядка
<Esc>	очистити рядок уведення
<Ins>	вибрати режим вставки/заміни
<PgUp>	на сторінку сесії вгору
<PgDn>	на сторінку сесії вниз

Раніше введені команди автоматично утворюють список, який виводиться у вікні історії. Якщо з'явилася необхідність повторити раніше виконану команду, то треба відшукати її в цьому вікні і двічі клацнути по ній лівою кнопкою миші. Для активізації вікна історії необхідно вибрати вкладку з однойменною назвою і клацнути по ній лівою кнопкою миші. Якщо клацнути на команді вікна історії лівою кнопкою миші, то дана команда стає поточною (на синьому тлі). Можна виділити потрібну послідовність команд за допомогою комбінації клавіш <Shift+↑>, <Shift+↓>. Вставлений у командний рядок набір команд відправляється на виконання натисненням клавіші <Enter>. До цього вміст набору можна редагувати, використовуючи звичайні прийоми редагування, спільні для Windows-програм, у тому числі за допомогою миші. Можна вносити в команди необхідні зміни, видаляти зайві команди і додавати нові.

1.2. ОПЕРАЦІЙНЕ СЕРЕДОВИЩЕ MATLAB І ОСНОВНІ КОМАНДИ

Основні пункти меню системи, інструментальна панель і опції MATLAB. Основне меню системи складається з 6 пунктів, кожний з яких у свою чергу містить декілька підпунктів різного ступеня вкладеності :

- *File* – робота з файлами;
- *Edit* – редагування сесії та документів М-файлів;
- *View* – управління вікнами та панеллю інструментів;
- *Web* – доступ до Інтернет-ресурсів;
- *Windows* – відображення відкритих додаткових вікон, перехід до потрібного вікна;
- *Help* — доступ до довідкових підсистем.

Розглянемо призначення деяких із них.

Серед команд меню *File*:

- *New* – відкриття підменю з пунктами:
  - *M-file* – відкриття вікна редагування М-файлів, аналог команди *edit*;

- *Figure* – відкриття порожнього вікна графіки;
- *Model* – відкриття порожнього вікна для створення Simulink-моделі;
- *GUI* – відкриття вікна розробки елементів графічного інтерфейсу користувача;

- *Open* – відкриття вікна завантаження файла;
- *Close Command Windows* – закриття вікна командного режиму;
- *Import data* – відкриття вікна імпорту файлових даних;
- *Save Workspace As* – відкриття вікна запису робочої області у вигляді файла із заданим ім'ям;
- *Set Path* – відкриття вікна встановлення шляхів доступу файлової системи;
- *Preferences* – відкриття вікна налаштування елементів інтерфейсу;
- *Print* – відкриття вікна друку поточного документа;
- *Print Selection* – відкриття вікна друку виділеної частини документа.
- *Exit* – завершення роботи в системі.

Серед команд меню *Edit*:

- *Undo* – скасування результату попередньої операції;
- *Redo* – скасування дії останньої операції *Undo*;
- *Cut* – вирізання виділеного фрагмента і перенесення його в буфер обміну;
- *Copy* – копіювання виділеного фрагмента в буфер обміну;
- *Paste* – вставка фрагмента з буфера обміну в поточну позицію курсору;
- *Select All* – виділення всієї сесії;
- *Delete* – видалення виділеного об'єкта;
- *Clear Command Windows* – очистка тексту сесії зі збереженням створених об'єктів, аналог команди *clc*;
- *Clear Command History* – очистка вікна історії команд;
- *Clear Workspace* – очистка вікна робочої області.

Вікно редактора М-файлів. Для підготовки і редагування М-файлів (текстів програм на внутрішній алгоритмічній мові системи) використовується спеціальний системний редактор текстів або зовнішній текстовий редактор. Редактор можна викликати командою *edit* з командного рядка або через пункт меню *File*. Після закінчення роботи з текстом М-файла його необхідно записати на диск, використовуючи команду *File* → *Save as* у меню редактора.

При створенні тексту файла редактор виконує ретельну синтаксичну перевірку. Редактор має й інші важливі засоби – він дозволяє встановлювати в тексті М-файла спеціальні мітки переривання (breakpoints). При досягненні цих міток обчислення припиняються, і користувач може оцінити проміжні результати обчислень, наприклад, значення певних змінних, перевірити правильність виконання циклів і т.д. Для зручності роботи з редактором рядки програми в ньому нумеруються в послідовному порядку. Редактор є багатовіконним. Вікно кожної програми оформлене як вкладка.

Управління командним вікном. Розглянемо деякі команди управління вікном командного режиму:

- *clc* – очистка екрана і розміщення курсору в лівому верхньому куті порожнього вікна;
- *home* – повернення курсору в лівий верхній кут вікна;
- *echo on* – увімкнення режиму виведення на екран тексту файлу М-сценарію;
- *echo off* – вимкнення попереднього режиму виведення на екран тексту файлу М-сценарію;
- *echo* – зміна режиму виведення на протилежний;
- *echo on all* – увімкнення режиму виведення на екран тексту для всіх М-файлів;
- *echo off all* – вимкнення режиму виведення на екран тексту М-файлів;
- *more on* – увімкнення режиму посторінкового виведення (відображення) (корисний при перегляді великих М-файлів);

- *more off* – вимкнення режиму посторінкового виведення, в цьому випадку для перегляду великих файлів треба користуватися смугою прокручування.

MATLAB у ролі калькулятора. Система MATLAB створена таким способом, що будь-які (часом досить складні) обчислення можна виконувати в режимі прямих обчислень (інтерактивному режимі), тобто без підготовки програми. Це перетворює MATLAB у надзвичайно потужний калькулятор, здатний здійснювати не тільки звичайні для калькуляторів обчислення (наприклад, виконувати арифметичні операції й обчислювати елементарні функції), але й операції з векторами й матрицями, комплексними числами, рядами й поліномами і т.д. Можна майже миттєво задати й вивести графіки різних функцій — від простої одновимірної до складної тривимірної фігури. Робота із системою в режимі прямих обчислень носить діалоговий характер і відбувається за правилом "поставив запитання, отримав відповідь". Користувач набирає певний запит на клавіатурі, потім (якщо потрібно) редагує його в командному рядку і передає на виконання натисненням клавіші <Enter>.

Деякі правила роботи в системі MATLAB:

- ознакою готовності системи до введення запиту є наявність курсору після символу **">>"** у командному вікні;
- дані вводяться і редагуються за допомогою найпростішого рядкового редактора;
- записи запитів розрізняються залежно від використання великих і малих літер;
- якщо в кінці запиту (команди) поставити знак **";"**, то результат (відповідь) не відображається у командному вікні. Якщо цього не зробити (в тому числі і в М-файлі), то після успішного обчислення буде здійснене виведення результатів на екран;
- якщо не зазначена змінна для отримання результату обчислень, то MATLAB призначає таку змінну з ім'ям *ans* ;
- знаком присвоювання є **"="** ;
- результат обчислень виводиться в рядках відображення (без знака **">>"**);
- у загальному випадку для подання результату в командному вікні застосовується така форма: <*ім'я_змінної*> = <*результат*>;
- вбудовані функції записуються малими літерами, і їхні аргументи вказуються в круглих дужках, наприклад, *sin*(*x*);
- у деяких випадках математичний вираз, що вводиться, може виявитись настільки довгим, що для нього не вистачить одного рядка. Тоді для його перенесення на новий рядок використовується символ "…" (три або більше крапки), наприклад:

$$s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 \ldots$$

$$- 1/8 + 1/9 - 1/10 + 1/11 - 1/12;$$

- цей прийом може бути корисним і для створення наочних документів (запобігання заходу рядків у невидиму область вікна або текстового документа, який друкується на принтері). Загалом кажучи, максимальне число символів в одному рядку командного режиму – 4096, а в М-файлі – не обмежене, проте з

такими довгими рядками працювати незручно. У ранніх версіях пакета рядок мав не більше 256 символів;

- зайві пробіли в командному рядку ігноруються, якщо вони не порушують синтаксис мови MATLAB. Якщо ж спробувати поставити пробіл всередині синтаксичної конструкції, MATLAB виведе повідомлення про помилку і вкаже позицію, в якій ця помилка була допущена;
- під час проведення обчислень командне вікно блокується до їх завершення. Зазвичай це непомітно для користувача, так як обчислення в MATLAB виконуються дуже швидко. Але іноді через помилки у побудові обчислювального алгоритму, перевитрати машинних ресурсів комп'ютера або з якоїсь іншої причини, час обчислень стає або неприйнятно великим, або взагалі нсскінченним («зациклювання»). У такій ситуації обчислення необхідно перервати, натиснувши комбінацію клавіш <Ctrl-Break>.

Вихід із системи. Завершення роботи з системою MATLAB здійснюється за допомогою команд *quit, exit* або натисненням комбінації клавіш <Ctrl+0>.

Завдання для самостійної роботи

1. Створити свою (індивідуальну) папку для роботи в системі MATLAB і додати її до системних директорій MATLAB.
2. Відобразити на екрані значення системних змінних і констант *eps, pi, realmin, realmax.*
3. Присвоїти змінній *x* значення 5 без виведення значення на екран.
4. Вивести значення змінної *x* на екран.
5. Присвоїти змінній *y* значення 15 з виведенням значення на екран.
6. Знайти значення виразу $x+y$.
7. Знайти значення виразу $x+i*y$.
8. Знайти значення виразу $x-j*y$.
9. Знайти значення виразу $x+i*y$ і записати його в змінну *u* (використовуючи редагування попередньої команди).
10. Знайти значення виразу $u*ans$.
11. Проаналізувати результати виконання команд 0/0, -1/0, exp(710).
12. Ввести вектор *V* з елементами 1, 2, 3, 4
13. Обчислити функції синуса та експоненти від вектора *V.*
14. Ввести матрицю *A* розмірністю 1×4
15. Ввести матрицю *B* розмірністю 4×1
16. Знайти матрицю *C=A*B.*
17. Знайти вектор *Y=C*V.*
18. Проаналізувати стан вікон історії та робочої області після виконання завдань 1–17.
19. Відкрити вікно редактора М-файлів і перенести в нього інформацію з вікна історії.
20. Зберегти отриманий текст у М-файлі з ім'ям *tem11.m.*
21. Очистити командне вікно.
22. Очистити вікна історії та робочої області.
23. Завантажити М-файл з ім'ям *tem11.m* у вікно редагування й замінити в ньому четвертий рядок на *XY=x+y.* Зберегти файл під ім'ям *tem11_.m.*
24. Виконати М-файл з ім'ям *tem11_.m.*

25. Використовуючи системний редактор текстів, створити М-файл з ім'ям *tem12.m*, у якому ввести вектор-рядок *X* розмірністю 10 і вектор-стовпець *Y* розмірністю 10. Виконати нормування векторів і знайти косинус кута між векторами *X* та *Y*. Зберегти цей файл і виконати його.
26. Створити командою *edit* М-файл з ім'ям *tem13.m*, у якому ввести вектори *a*, *b*, *c* розмірністю 3×1. Знайти (використовуючи математичне означення) векторний добуток векторів *a* та *b* (вектор *x*) і число $H = |abc|$ – змішаний добуток векторів *a*, *b*, *c*. Зберегти цей файл і виконати його.
27. Створити командою *edit* М-файл з ім'ям *tem14*.m, у якому ввести вектор X, із цілочисельними компонентами з проміжку [5, 10]. Знайти значення виразу

$$S = \frac{A^{\frac{2}{3}}\sqrt{C^3}}{(B+D)^2},$$

де $A=(sin(X)cos(X))^2$, $B=(ln(X+2))/X$, $C=(exp(X/X^2))^{-2}$, $D=A+B$.

28. Оформити за допомогою команд керування командним вікном усі наступні пункти вправи у вигляді М-сценарію з ім'ям свого прізвища.
29. Записати кілька прикладів обчислення предикатів

$$ab \neq 0; a > b; |a| \geq 0; \big||a| - |b|\big| \leq |a+b|; |a+b| \leq |a| + |b|; b^2 - 4ac \geq 0$$

у двох варіантах – з використанням операції та функції. Звернути увагу на отримані значення.

30. Записати кілька прикладів обчислення булевих виразів

$$p \vee q;\ p \,\&\, q \,\&\, r;\ q \supset p;\ x < y \vee x = y;\ y \geq x \,\&\, y + x \geq 0 \,\&\, y \leq 1;$$

$$(x-1)^2 + (y-2)^2 \geq 9 \,\&\, (x-3)^2 + (y+1)^2 \leq 25;\ 0 < x \leq 1$$

у двох варіантах – з використанням операції та функції.

31. За вектором $Z = [1\text{-}2i,\ 1+3i,\ 1+2i,\ 2,\ 5i,\ 0]$ створити матрицю: перший рядок – це модуль відповідних компонент вектора *Z*, другий – аргумент, третій – дійсна частина, четвертий – уявна частина. Визначити, скільки часу (у секундах) витрачається на знаходження цієї матриці.
32. Для вектора *Z* із завдання 31 знайти скалярний добуток *Z* на *Z**.
33. Знайти найбільший спільний дільник і найменше спільне кратне заданих цілих чисел *m* і *n*.
34. Задати довільне дійсне число *Y*. Знайти його зображення у вигляді ланцюгового та раціонального дробу.
35. Задано одновимірний масив дійсних чисел *X* = [–1.9, –0.2, 3.4, 5.6, 7.0]. Побудувати масиви чисел *X*1, *X*2, *X*3, *X*4, елементами яких будуть значення компонент масиву *X*, що

    а) заокруглені до найближчого цілого, більшого або рівного відповідному елементу *X;*

    б) заокруглені до найближчого цілого, меншого або рівного відповідному елементу *X;*

    в) заокруглені до найближчого цілого;

    г) отримані відкиданням дробової частини числа.
36. Для двох заданих дійсних чисел *x* і *y* обчислити залишок від ділення *x* на *y*.

Практичне заняття № 02

ВИКОРИСТАННЯ ОСНОВНИХ СТРУКТУР КЕРУВАННЯ МОВИ MATLAB ДЛЯ НАПИСАННЯ М-ФАЙЛІВ

Аудиторне заняття

1. Записати синтаксис
   - операторів: присвоєння, введення, виведення;
   - керуючих структур: ланцюг операторів, розгалуження, вибору, циклів.

   Означити семантику цих об'єктів алгоритмічної мови MATLAB.
2. За допомогою команди edit створити М-файл з ім'ям *tem21a.m*, в якому ввести: вектор-рядок X розмірності 10 і вектор-стовпець Y розмірності 10. Без використання вбудованих функцій MATLAB виконати нормування векторів ($\|\bullet\|_C$) і знайти косинус кута між векторами X і Y. Передбачити верифікацію вхідних даних. Зберегти цей файл і виконати його тестування.

   Текст М-файлу *tem21a.m*

```
%%tem21a.m
close all, clear all, clc
%X = input('? вектор-рядок X розмірності 10 :=');
%Y = input('? вектор-стовпець Y розмірності 10 :=');
X = [1, -2, 3, 4, 5, 6,-7, 21, -15, 12];
Y = [2; -1; -1; 1; 2; 3; 567; -7; -8; 10];
disp('X :'); disp(X);
disp('Y :'); disp(Y);
[i, nX] = size(X); [nY, j] = size(Y);
if i ~=1
    error('X - НЕ вектор-рядок')
elseif j ~= 1
    error('Y - НЕ вектор-стовпець')
elseif nX ~= nY
    error('X,Y мають різні кількості елементів')
end

I = 1 : nX;  % масив індексів
x = -1; y = -1;
for i = I
    z = abs(X(i));
    if z > x
        x = z;
    end
    z = abs(Y(i));
    if z > y
        y = z;
    end
end
X = X./x; Y = Y./y;
disp('Вектори після нормування'), X, Y

s = 0; x = 0; y = 0;
for i = I
    Xi = X(i); Yi = Y(i);
    s = s + Xi.*Yi;
    x = x + Xi.*Xi;
    y = y + Yi.*Yi;
```

```
end
if x & y
     cosXY = s./sqrt(x.*y);
else
     cosXY = inf;
end
disp('косинус кута між векторами X і Y =')
disp(cosXY)
```

Тестування і виконання *tem21a.m* зробити самостійно.

3. Використовуюючи системний редактор текстів створити М-файл з ім'ям *tem22a.m*, в якому ввести довільне ціле число *N*. Якщо це число не просте, то знайти його розклад на прості множники і масив простих чисел, які не перевищують *N*.

Текст М-файлу *tem22a.m*

```
%%tem22a.m
close all, clear all, clc
N = input('? ціле число N :=');
if isreal(N) & N == fix(N)
    if N < 0
        N = -N; disp('Перетворили в ціле додатнє!')
    end
    if isprime(N)
        disp(strcat('Число=',int2str(N),' - просте'))
    else
        disp(strcat('Число=',int2str(N),' - не просте'))
        disp('Розклад на прості множники'); P = factor(N)
        disp('Масив простих чисел, які його не перевищують')
        Z = primes(N); disp(Z);
    end
else
    disp(strcat('Число=',num2str(N),' - не ціле'))
end
```

Тестування і виконання *tem22a.m* зробити самостійно.

4. Створити М-файл з ім'ям *tem23a.m*, в якому ввести натуральне число *N* (використати захищене введення !) і комплексне число *C*. Якщо $C \neq 0$, то знайти розв'язок рівняння $Z^N = C$. Отриманий масив Z надрукувати.

Текст М-файлу *tem22a.m*

```
%%tem23a.m
close all, clear all, clc
while true
    N = input('? натуральне число N :=');
    if N >= 0 & isreal(N) & N == fix(N)
         break
    end
end
while 1
    C = input('? комплексне число C<>0 :=');
    if C ~= 0 & imag(C) ~= 0
         break
    end
end
r = abs(C).^(1/N); fi = angle(C); p = 2.*pi;
K = [1 : N].';
```

```
disp('Варіант реалізації №1 (звичайний операторний)')

Z = zeros(N,1);
for k = K
     a =  (fi + p.*(k-1))./N;
     Z(k) = r.*(cos(a)+sin(a).*i);
end
disp('Розв''язок Z^N=C'), disp(Z)
disp('Перевірка Z^N ?= C'), disp(Z.^N)
clear Z

disp('Варіант реалізації №2 (векторизований)')
Z = r.*exp(i.*(fi + p.*(K-1))./N);
disp('Розв''язок Z^N=C'), disp(Z)
disp('Перевірка Z^N ?= C'), disp(Z.^N)
```

Тестування зробити самостійно, а приклад виконання *tem23a.m* :

```
? натуральне число N :=4
? комплексне число C<>0 :=i
Варіант реалізації №1 (звичайний операторний)
Розв'язок Z^N=C
      0.92388 +    0.38268i
     -0.38268 +    0.92388i
     -0.92388 -    0.38268i
      0.38268 -    0.92388i
Перевірка Z^N ?= C
 -2.2204e-016 +          1i
            0 +          1i
  4.4409e-016 +          1i
 -1.1102e-015 +          1i
Варіант реалізації №2 (векторизований)
Розв'язок Z^N=C
      0.92388 +    0.38268i
     -0.38268 +    0.92388i
     -0.92388 -    0.38268i
      0.38268 -    0.92388i
Перевірка Z^N ?= C
 -2.2204e-016 +          1i
            0 +          1i
  4.4409e-016 +          1i
 -1.1102e-015 +          1i
>>
```

Завдання для самостійної роботи

5. Створити М-файл з ім’ям *tem24.m*, в якому ввести: вектор X розмірності 10 і ціле число класу `uint8` $p \in \{1, 2, 3\}$ (використати захищене введення). За допомогою оператора вибору і циклу з лічильником знайти (без використання вбудованих функцій MATLAB):

$$normX = \begin{cases} \max\limits_i \left(|X_i| \right), & p = 1, \\ \sum\limits_i |X_i|, & p = 2, \\ \sqrt{\sum\limits_i X_i^2}, & p = 3. \end{cases}$$

6. Створити М-файл з ім'ям *tem25.m*, в якому ввести квадратну матрицю А розмірності $n = 5$. Обчислити масив комірок m з наступними матрицями :

$$m_1 = (a_{1,1}),\ \ m_2 = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix},\ \ ...,\ \ m_n = A..$$

7. Створити М-файл з ім'ям *tem26.m*, в якому ввести квадратну матрицю А розмірності $n = 4$. Позначимо $P\langle k, j\rangle = (E)_{j\leftrightarrow k}$ ðÿäêè - елементарна матриця перестановок, де E – одинична матриця. Обчислити масив p з наступними матрицями :

$$p_1 = P\langle 1, 2\rangle,\ \ p_2 = P\langle 1, 3\rangle,\ \ ...,\ \ p_{n-1} = P\langle 1, n\rangle.$$

8. Створити М-файл з ім'ям *tem27.m*, в якому спочатку викликати М-файл з ім'ям *tem26.m,* а потім обчислити масив B з наступними матрицями :

$$B_1 = A,\ \ B_j = p_{j-1}A,\ \ \ j = \overline{2, n}.$$

9. Створити М-файл з ім'ям *tem28.m*, в якому ввести квадратну матрицю А розмірності $n = 4$. Позначимо $P\langle k, j\rangle = (E)_{j\leftrightarrow k}$ ðÿäêè - елементарна матриця перестановок, де E – одинична матриця. Обчислити масив p з наступними матрицями :

$$p_i = P\langle i, k_i\rangle,\ \ i = \overline{1, n-1},$$

де k_i - індекс (номер) рядка, в якому знаходиться ненульовий елемент, для якого виконується умова $\max\limits_{i\le j\le n}(|A_{i,j}|)$.

10. Створити М-файл з ім'ям *tem29.m*, в якому спочатку викликати М-файл з ім'ям *tem28.m,* а потім обчислити масив B з наступними матрицями :

$$B_1 = A,\ \ B_j = p_{j-1}B_{j-1},\ \ \ j = \overline{2, n}.$$

11. В СКМ MATLAB поліном моделюється вектор-рядком з ненульовим головним коефіцієнтом і розташуванням усіх (в т.ч. нульових) коефіцієнтів по спаданню степенів. Для прикладу, запис

```
P = [1,0,0,2.35,-20,-4.21e3];
```

означає поліном $P_5(x) = x^5 + 2.35x^2 - 20x - 4.21e3$. Створити М-файл з ім'ям *tem210.m* для виконання такої роботи: вводиться деякий вектор-рядок, який перетворюється в MATLAB-поліном. Для MATLAB-полінома необхідно знайти (назвемо його `str_poly` представленням полінома) рядок виду (для нашого прикладу)

$$1 * x\text{^}5 + 2.35 * x\text{^}2 - 20 * x - 4210\ .$$

12. В термінах попередньої задачі, створити М-файл з ім'ям *tem211.m*, в якому для MATLAB-полінома необхідно знайти (назвемо його `str_poly_g` представленням полінома по схемі Горнера) рядок виду (для нашого прикладу)

$$-4210 + x*(-20 + x*(2.35 + x*(0 + x*(0 + x*(1))))).$$

13. В термінах задачі №11, створити М-файл з ім'ям *tem212.m*, в якому для MATLAB-полінома необхідно знайти (назвемо його `str_poly_gs` представленням полінома по схемі Горнера) рядок виду (для нашого прикладу)

$$-4210 + x*(-20 + x*(2.35 + x\text{^}3*(1)))\ .$$

Практичне заняття № 03

АЛГОРИТМИ РОЗКЛАДУ МАТРИЦЬ. ОБУМОВЛЕНІСТЬ МАТРИЦЬ. ПРОБЛЕМА ВЛАСНИХ ЗНАЧЕНЬ

Аудиторне заняття

1. Нехай в системі лінійних алгебраїчних рівнянь (СЛАР) матриця **A** є тридіагональна, має діагональну перевагу і записана в компактній формі (за допомогою трьох векторів $\vec{\mathbf{a}}, -\vec{\mathbf{c}}, \vec{\mathbf{b}}$):

$$\begin{cases} a_i x_{i-1} - c_i x_i + b_i x_{i+1} = -f_i, \\ i = \overline{1,n}, \quad a_1 = b_n = 0, \end{cases}$$

$$\left(|c_i| \ge |a_i| + |b_i|\right)_{i=\overline{1,n}} \,\&\, \left(a_i \ne 0\right)_{i=\overline{2,n}} \,\&\, \left(b_i \ne 0\right)_{i=\overline{1,n-1}}.$$

Для модельної задачі (відомий точний розв'язок $\vec{\mathbf{x}}$) :

- Отримати **A** – «повну» матрицю СЛАР, вектор $(-\vec{\mathbf{f}})$ – праву частину і надрукувати розширену матрицю СЛАР;
- Розглянути розрахункові формули методу *монотонної правої прогонки* (мМПП), алгоритм і існуючу програмну реалізацію у вигляді зовнішньої М-функції `m_progm` ;
- Записати розрахункові формули методу *монотонної лівої прогонки* [12];
- Модифікувати файл `m_progm.m` в файл `m_progml.m` для реалізації методу монотонної лівої прогонки (мМЛП);
- Використовуючи `m_progm`, `m_progml`, операцію *лівого матричного ділення* знайти y_{r}, y_{l}, $y_{\backslash}$ – відповідні чисельні розв'язки СЛАР, а також абсолютні похибки цих розв'язків по відношенню до точного розв'язку;
- Використовуючи вхідні вектори $(\vec{\mathbf{a}}, -\vec{\mathbf{c}}, \vec{\mathbf{b}})$, отримані вихідні параметри мМПП (вектор $\vec{\alpha}$) обчислити визначник матриці **A** і перевірити це значення за допомогою `det(A)`;
- Використовуючи вхідні вектори $(\vec{\mathbf{a}}, -\vec{\mathbf{c}}, \vec{\mathbf{b}})$, отримані вихідні параметри мМПП (вектор $\vec{\alpha}$) знайти матриці **L**, **U**, де $\mathbf{A} = \mathbf{L} * \mathbf{U}$;
- Знайти :
  `rcond(A,inf)`, μ_A=`cond(A,inf)`, μ_L=`cond(L,inf)`, μ_U=`cond(U,inf)`, $\breve{\mu}_L = \max_i |\lambda_i(L)| \Big/ \min_i |\lambda_i(L)|$ і перевірити виконання умов $\mu_A \le \mu_L * \mu_U$, $\mu_L \ge \breve{\mu}_L$;
- Для матриці **A** знайти власні вектори (матрицю *H*) і власні числа (матрицю *L*), а також перевірити виконання *A*H=H*L*;
- Для матриці **A** знайти характеристичний поліном *P(x)* у варіанті MATLAB-полінома і у рядковому представленні – по спаданню степенів (елементів виду `p*x^k`).

Почнемо з мМПП. Прямий хід методу Гаусса послідовного виключення x_1, x_2, …, x_{n-1} (перехід від початкової матриці **A** до верхньої трикутної **U**) схематично зображений в верхній частині рис. 3.1. Відповідно до цього, розрахункові формули мМПП (див. Лекцію №3) будуть складати дві групи :
спочатку знаходимо *прогоночні коефіцієнти* $(\alpha_i, \beta_i)_{i=\overline{1,n-1}}$ за рекурентними формулами

$$\alpha_1 = \beta_1 = 0;$$
$$\left\{ z = c_i - a_i\alpha_i,\ \ \alpha_{i+1} = b_i / z,\ \ \beta_{i+1} = (f_i + a_i\beta_i)/z,\ \ i = \overline{1, n-1} \right.,$$

а потім компоненти вектора розв'язку $(x_i)_{i=n,n-1,\ldots,1}$ за рекурентними формулами

$$x_n = (f_n + a_n\beta_n)/(c_n - a_n\alpha_n);$$
$$\left\{ x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1},\ \ i = n-1, n-2, \ldots, 1. \right.$$

Після цього алгоритм записується елементарно (записати самостійно!), а його програмна реалізація представлена зовнішньою М-функцією `m_progm` (див. відповідний текст програми).

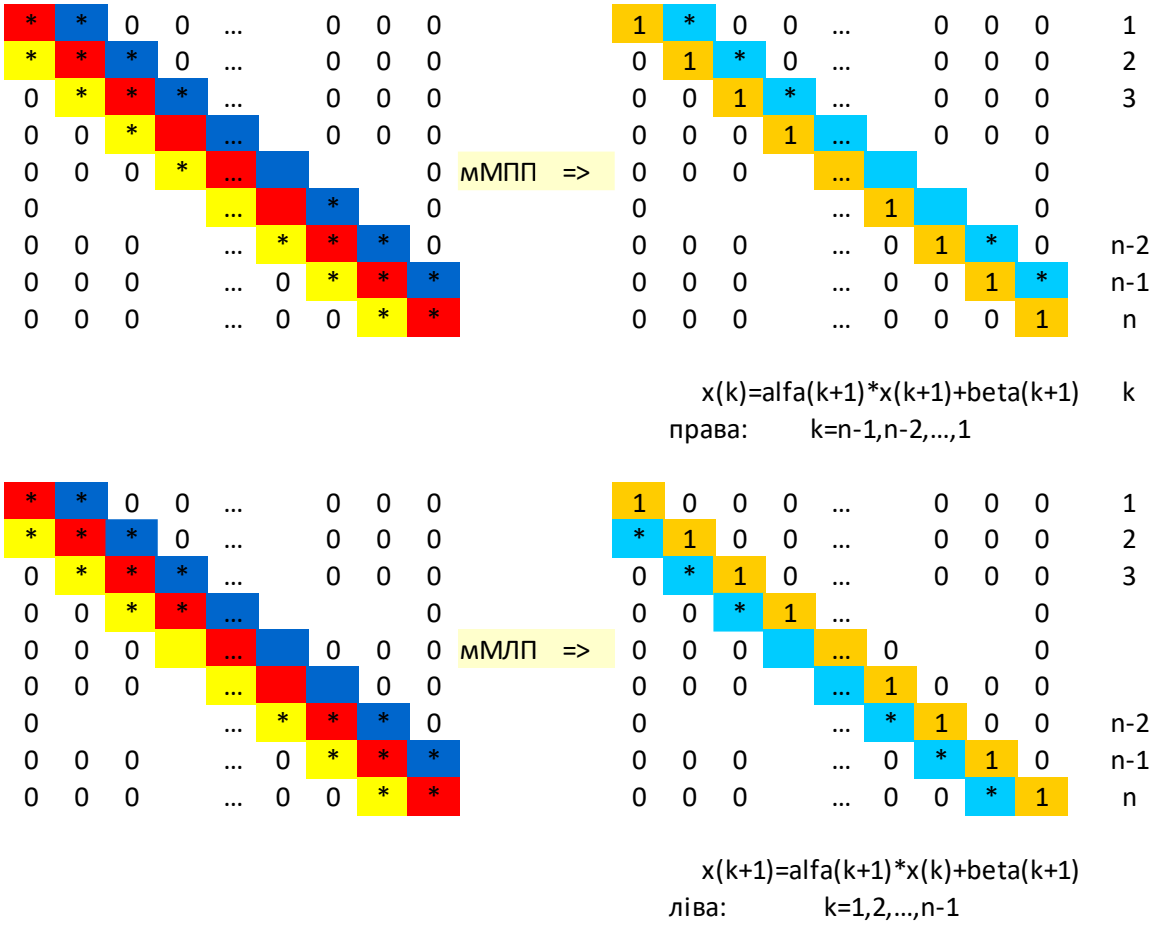


Рис. 3.1. Схема перетворення $\mathbf{A \to U}$ в мМПП і $\mathbf{A \to L}$ в мМЛП.

Для мМЛП прямий хід методу Гаусса – це виключення x_n, x_{n-1}, …, x_2 (перехід від початкової матриці **A** до нижньої трикутної **L**) схематично зображений в нижній частині рис. 3.1. Відповідно, розрахункові формули мМЛП будуть :

$$\alpha_n = a_n / c_n,\ \ \beta_n = f_n / c_n;$$
$$\left\{ z = c_i - b_i\alpha_{i+1},\ \ \alpha_i = a_i / z,\ \ \beta_i = (f_i + b_i\beta_{i+1})/z,\ \ i = n-1, n-2, \ldots, 2; \right.'$$
$$x_1 = (f_1 + b_1\beta_2)/(c_1 - b_1\alpha_2);$$
$$\left\{ x_{i+1} = \alpha_{i+1}x_i + \beta_{i+1},\ \ i = 1, 2, \ldots, n-1. \right.$$

Модифікація тексту програми `m_progm.m` в файл `m_progml.m` відбувається зміною імені функції (в заголовку) і відповідними змінами операторів реалізації вище приведених розрахункових формул (див. відповідний текст програми).

Використовуючи розрахункові формули мМПП, легко знаходимо елементи матриць **L** і **U**:

$$\mathbf{L}=\left(l_{i,j}\right)_{i,j=1}^{n},\quad l_{i,j}=\begin{cases}0, & \left((j<i-1)\vee(j>i)\right)\&\left(i=\overline{2,n}\right);\\ a_i\alpha_i-c_i, & (j=i)\&\left(i=\overline{1,n}\right);\\ a_j, & (j=i-1)\&\left(i=\overline{2,n}\right).\end{cases}$$

$$\mathbf{U}=\left(u_{i,j}\right)_{i,j=1}^{n},\quad u_{i,j}=\begin{cases}0, & \left((j<i)\vee(j>i+1)\right)\&\left(i=\overline{1,n-1}\right);\\ 1, & (j=i)\&\left(i=\overline{1,n}\right);\\ -\alpha_j, & (j=i+1)\&\left(i=\overline{1,n-1}\right).\end{cases}$$

Враховуючи, що $\det(\mathbf{A})=\det(\mathbf{L})*\det(\mathbf{U})=\det(\mathbf{L})$, а отже $\det(\mathbf{A})=\prod_{i=1}^{n}(a_i\alpha_i-c_i)$.

Решта вимог, які сформульовані в умові задачі, легко реалізуються вбудованими функціями MATLAB.

Текст М-файла *tem31a.m* з тестовим завданням векторів $\vec{\mathbf{a}},-\vec{\mathbf{c}},\vec{\mathbf{b}},\vec{\mathbf{x}}$ і тексти зовнішніх М-функцій користувача *m_progm.m*, *m_progml.m*:

```
%% tem31a.m
close all, clear all, clc, warning off

n = 6; nm = n-1;
% модельний розв'язок СЛАР :
X = linspace(-5,5,n); x = X';
%Будуємо СЛАР :
% компактна форма матриці СЛАР
a = [0,linspace(1,10,nm)]; % піддіагональ
c = linspace(10,100,n);    % -(діагональ)
b = [linspace(2,11,nm),0]; % наддіагональ
% "повна" матриця А
A = diag(-c); A(1,2) = b(1); A(n,nm) = a(n);
for k = 2 : nm
   A(k,k-1) = a(k); A(k,k+1) = b(k);
end
% права частина СЛАР для модельного х :
f = A * x;
disp(' СЛАР :'); disp([A, f]);
disp('Точн.розв''язок (модельний):'); disp(X);

%м.правої монотонної прогонки (мПМП)
[y, alfa, err] = m_progm(a, c, b, -f);
disp('Розв''язок м.ПравМонотПрог :'); disp(y);
disp(strcat('похибка =',num2str(norm(y-X,inf))));

%м.лівої монотонної прогонки (мЛМП)
[y, alf, err] = m_progml(a, c, b, -f);
disp('Розв''язок м.ЛівоїМонотПрог:'); disp(y);
disp(strcat('похибка =',num2str(norm(y-X,inf))));
clear alf

disp('Розв''язок СЛАР ( x = A\f ):');
y = A\f; disp(y');
disp(strcat('похибка =',num2str(norm(y-x,inf))));

%  обчислимо визначник А
l = a.*alfa - c; dp = prod(l); d = det(A);
disp('  det за мПМП функц det(A)'); disp([dp, d]);

%Представлення L,U через отримані вектори мПМП, де A=L*U:
```

```
L = diag(l); U = eye(n);
for k = 1 : nm
    j = k + 1;
    L(j,k) = a(j);
    U(k,j) = -alfa(j);
end
disp('Матриця L'); disp(L);
disp('Матриця U'); disp(U);
disp('Перевірка |A-L*U|<1e-14 :');
disp(abs(A-L*U) < 1e-14);

%Співвідношення між числами обумовленості
condA = cond(A,inf);
condL = cond(L,inf); condU = cond(U,inf);
disp('  cond(L,inf)  cond(U,inf)');
disp([condL, condU]);
disp('     rcond(A)  cond(A,inf)<=cond(L,inf)*cond(U,inf)');
disp([rcond(A), condA, condL*condU]);
l = abs(l); cndL = max(l)./min(l);
disp('  cond(L,inf) >=      cndL');
disp([condL, cndL]);
clear alfa l L U

%Власні вектори і власні числа A :
[H, L] = eig(A); l = diag(L)';
disp('Власні вектори A (стовпці)'); disp(H);
disp('Власні числа A'); disp(l);
disp('Похибка |A*H-H*L| :');
disp(max(max(abs(A*H-H*L))));
clear H L l

disp('Характеристичний поліном матриці A :');
P = poly(A); disp(P); % MATLAB-поліном
S = ''; j = 0;
for k = n :-1: 0
    j = j+1; d = P(j); dp = '';
    if j > 1
        if d > 0
            dp = '+';
        end
    end
    if k > 1
        e = strcat('*x^', int2str(k));
    elseif k == 1
        e = '*x';
    else
        e = '';
    end
    S = strcat(S, dp, num2str(d), e);
end
disp(S);              % поліном у вигляді рядка


function [y, alfa, err]  =  m_progm(a, c, b, f)
% метод монотонної (правої) прогонки для розв'язання
%       СЛАР з тридіагональною матрицею
%    a(k)*y(k-1) - c(k)*y(k) + b(k)*y(k+1) = - f(k),
%                  a(1) = b(n) = 0,
% при виконанні умов :
%             |c(k)| >= |a(k)| + |b(k)|,
%                 k= 1,2,...,n.
%  Вхідні  дані:
```

```
%      a - піддіагональ;
%     -c - діагональ матриці;
%      b - наддіагональ;
%     -f - права частина.
%  Вихідні дані:
%           y - розв'язок;
%   alfa - прогоночні коефіцієнти;
%    err - 1= умови застосування методу не виконуються і
%            розв'язок не знайдено,
%          0= розв'язок знайдено.

n = length(c); err = 0; k = 0;
while k < n & ~err
  k = k + 1;
  err = err  |  abs(c(k)) < abs(a(k)) + abs(b(k));
end
alfa = zeros(1, n); y = alfa;
if err
  return
end

n1 = n - 1;
% пряма прогонка:
beta = alfa;
for k = 1 : n1
  j = k + 1; z = c(k) - a(k) .* alfa(k);
  alfa(j) = b(k) ./ z; beta(j) = (f(k) + a(k) .* beta(k)) ./ z;
end
% зворотна прогонка:
y(n) = (f(n) + a(n) .* beta(n)) ./ (c(n) - a(n) .* alfa(n));
for k = n1 : -1 : 1
  j = k+1; y(k) = alfa(j) .* y(j) + beta(j);
end
return


function [y, alfa, err]  =  m_progml(a, c, b, f)
% метод монотонної (лівої) прогонки для розв'язання
%       СЛАР з тридіагональною матрицею
%    a(k)*y(k-1) - c(k)*y(k) + b(k)*y(k+1) = - f(k),
%                  a(1) = b(n) = 0,
% при виконанні умов :
%            |c(k)| >= |a(k)| + |b(k)|,
%                k= 1,2,...,n.
%  Вхідні  дані:
%      a - піддіагональ;
%     -c - діагональ матриці;
%      b - наддіагональ;
%     -f - права частина.
%  Вихідні дані:
%           y - розв'язок;
%   alfa - прогоночні коефіцієнти;
%    err - 1= умови застосування методу не виконуються і
%            розв'язок не знайдено,
%          0= розв'язок знайдено.

n = length(c); err = 0; k = 0;
while k < n & ~err
  k = k + 1;
  err = err  |  abs(c(k)) < abs(a(k)) + abs(b(k));
end
alfa = zeros(1, n); y = alfa;
```

```
if err
  return
end

n1 = n - 1;
% пряма прогонка:
beta = alfa; alfa(n) = a(n) ./ c(n); beta(n) = f(n) ./ c(n);
for k = n1 : -1 : 2
  j = k + 1; z = c(k) - b(k) .* alfa(j);
  alfa(k) = a(k) ./ z; beta(k) = (f(k) + b(k) .* beta(j)) ./ z;
end
% зворотна прогонка:
y(1) = (f(1) + b(1) .* beta(2)) ./ (c(1) - b(1) .* alfa(2));
for k = 1 : n1
  j = k+1; y(j) = alfa(j) .* y(k) + beta(j);
end
return
```

Виконання *tem31a.m* в СКМ MATLAB:

```
СЛАР :
         -10           2           0           0           0           0          44
           1         -28        4.25           0           0           0       74.75
           0        3.25         -46        6.5           0           0       42.75
           0           0         5.5         -64        8.75           0      -43.25
           0           0           0        7.75         -82          11      -183.25
           0           0           0           0          10        -100        -470
Точн.розв'язок (модельний):
    -5    -3    -1     1     3     5
Розв'язок м.ПравМонотПрог :
          -5          -3          -1           1           3           5
похибка =4.4409e-016
Розв'язок м.ЛівоїМонотПрог:
          -5          -3          -1           1           3           5
похибка =8.8818e-016
Розв'язок СЛАР ( x = A\f ):
          -5          -3          -1           1           3           5
похибка =2.2204e-016
  det за мПМП функц det(A)
  6.3834e+009  6.3834e+009
Матриця L
         -10           0           0           0           0           0
           1       -27.8           0           0           0           0
           0        3.25      -45.503           0           0           0
           0           0         5.5      -63.214           0           0
           0           0           0        7.75      -80.927           0
           0           0           0           0          10      -98.641
Матриця U
           1        -0.2           0           0           0           0
           0           1     -0.15288           0           0           0
           0           0           1     -0.14285           0           0
           0           0           0           1     -0.13842           0
           0           0           0           0           1     -0.13592
           0           0           0           0           0           1
Перевірка |A-L*U|<1e-14 :
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
  cond(L,inf)  cond(U,inf)
       10.864       1.4828
     rcond(A)  cond(A,inf)<=cond(L,inf)*cond(U,inf)
     0.086087       11.964       16.109
  cond(L,inf) >=       cndL
       10.864       9.8641
Власні вектори A (стовпці)
       0.9984     -0.11271    -0.013528   -0.0016528   0.00019182 -1.2528e-005
     0.056333      0.97682      0.23512     0.043167   -0.0067765   0.00059587
    0.0051659       0.1798     -0.92415     -0.34733     0.083913    -0.010811
   0.00053451     0.027942     -0.29357      0.84591     -0.44401     0.098042
```

```
  5.8433e-005    0.0040741    -0.064549        0.389      0.79244     -0.45403
  6.4845e-006   0.00056065    -0.011686        0.103      0.40965      0.88551
Власні числа A
      -9.8872      -27.333      -44.762      -62.234      -80.656      -105.13
Похибка |A*H-H*L| :
  4.9738e-014
Характеристичний поліном матриці A :
            1          330        42311  2.6631e+006  8.5068e+007  1.2642e+009  6.3834e+009

1*x^6+330*x^5+42310.625*x^4+2663057.5*x^3+85067786.6602*x^2+1264211394.6172*x+6383414850.156
3
>>
```

Завдання для самостійної роботи

2. Нехай матриця A має: **a)** діагональну перевагу; **b)** тридіагональна. Записати розрахункові формули для знаходження матриць L, U, де $A = L \cdot U$ за методом *Холецького*, враховуючи наявність великого числа нулів матриці A.
3. Нехай матриця A має: **a)** діагональну перевагу; **b)** ермітова | симетрична; **c)** тридіагональна. Записати розрахункові формули для знаходження матриць $L, D,$ де $A = L \cdot D \cdot L^* \,|\, A = L \cdot D \cdot L'$ за методом *квадратного кореня* [12], враховуючи наявність великого числа нулів матриці A.

В пп.4-11 матриця A симетрична і має діагональну перевагу. Позначимо:

$$\lambda_M = \max_i |\lambda_i(A)|, \quad A \cdot \vec{h}_M = \lambda_M \vec{h}_M \text{ і } \lambda_m = \min_i |\lambda_i(A)|, \quad A \cdot \vec{h}_m = \lambda_m \vec{h}_m.$$

4. Написати М-файл для реалізації *степеневого ітераційного процесу* (СІП) пошуку значення λ_M з точністю ep, а також $\vec{h}_M$.
5. Написати М-файл для реалізації СІП пошуку значення λ_M з точністю ep, а також $\vec{h}_M$ з точністю ep.
6. Написати М-файл для реалізації методу *скалярних добутків* пошуку значення λ_M з точністю ep, а також $\vec{h}_M$.
7. Написати М-файл для реалізації методу *скалярних добутків* пошуку значення λ_M з точністю ep, а також $\vec{h}_M$ з точністю ep.
8. Написати М-файл для реалізації СІП пошуку значення λ_m з точністю ep, а також $\vec{h}_m$.
9. Написати М-файл для реалізації СІП пошуку значення λ_m з точністю ep, а також $\vec{h}_m$ з точністю ep.
10. Написати М-файл для реалізації методу *скалярних добутків* пошуку значення λ_m з точністю ep, а також $\vec{h}_m$.
11. Написати М-файл для реалізації методу *скалярних добутків* пошуку значення λ_m з точністю ep, а також $\vec{h}_m$ з точністю ep.
12. Не використовуючи точне знаходження числа обумовленості $cond(A)$, оцінити «чи погано обумовлена» матриця A і виконати необхідні перетворення в СЛАР для покращення цієї характеристики для наступних систем:

12.01	$\left(\begin{array}{ccc\|c} 1 & -3 & -1 & 4 \\ -2\cdot10^8 & 7 & 2 & -9 \\ 3\cdot10^8 & 2 & -4 & 2 \end{array}\right)$	**12.11**	$\left(\begin{array}{ccc\|c} 1 & -3 & -1 & 1 \\ -2\cdot10^6 & 5 & 2 & 4 \\ 3\cdot10^6 & 1 & -4 & 2 \end{array}\right)$

12.02	$\left(\begin{array}{ccc\|c} 3 & -3 & -1 & -3 \\ 1 & -10^{8} & 2 & 9 \\ 2 & 3\cdot 10^{8} & 4 & 2 \end{array}\right)$	**12.12**	$\left(\begin{array}{ccc\|c} 1 & -3 & 1 & 3 \\ 2 & -10^{5} & 2 & 5 \\ 2 & 3\cdot 10^{5} & 4 & 12 \end{array}\right)$
12.03	$\left(\begin{array}{ccc\|c} 3 & -3 & -1 & 1 \\ 1 & -10^{-3} & 2 & 3 \\ 2 & 4 & 3\cdot 10^{8} & -2 \end{array}\right)$	**12.13**	$\left(\begin{array}{ccc\|c} 5 & 3 & -1 & 1 \\ 2 & -10^{-3} & 2 & 3 \\ 1 & 4 & 3\cdot 10^{6} & -2 \end{array}\right)$
12.04	$\left(\begin{array}{ccc\|c} 5 & -3 & -1 & -3 \\ -1 & 8 & 2 & 4 \\ 2 & 3 & 10^{8} & 1 \end{array}\right)$	**12.14**	$\left(\begin{array}{ccc\|c} -5 & 6 & -1 & 1 \\ -1 & 7 & 2 & 6 \\ 2 & 3 & 10^{5} & 13 \end{array}\right)$
12.05	$\left(\begin{array}{ccc\|c} 1 & -3 & -1 & 4 \\ 2\cdot 10^{-4} & 7 & 3 & 5 \\ 6 & 3\cdot 10^{8} & -4 & 2 \end{array}\right)$	**12.15**	$\left(\begin{array}{ccc\|c} 1 & -3 & -1 & 4 \\ 2\cdot 10^{-3} & 7 & 3 & 1 \\ 2 & 3\cdot 10^{5} & 4 & 3 \end{array}\right)$
12.06	$\left(\begin{array}{ccc\|c} 3 & -3 & -1 & 5 \\ -10^{-4} & 1 & 3 & 6 \\ 2 & 10^{8} & -4 & 2 \end{array}\right)$	**12.16**	$\left(\begin{array}{ccc\|c} 2 & -4 & 1 & 4 \\ -10^{-3} & 10 & 3 & 6 \\ 1 & 10^{6} & -4 & 12 \end{array}\right)$
12.07	$\left(\begin{array}{ccc\|c} 3 & -3 & -1 & 1 \\ 1 & -10^{8} & 3\cdot 10^{8} & 12 \\ 2 & 4 & -2 & 2 \end{array}\right)$	**12.17**	$\left(\begin{array}{ccc\|c} 13 & -3 & -1 & 12 \\ 1 & -10^{4} & 3\cdot 10^{5} & 120 \\ 2 & 14 & -2 & 23 \end{array}\right)$
12.08	$\left(\begin{array}{ccc\|c} 5 & -3 & -1 & -3 \\ -1 & 8 & 2 & 4 \\ 2 & 3 & 10^{8} & 1 \end{array}\right)$	**12.18**	$\left(\begin{array}{ccc\|c} 5 & -3 & 10 & 3 \\ 1 & 80 & 2 & 40 \\ 20 & 3 & 10^{4} & 10 \end{array}\right)$
12.09	$\left(\begin{array}{ccc\|c} 3 & 3 & 1 & 1 \\ 1 & -10^{-2} & 5 & 12 \\ 2 & 4 & 10^{8} & 2 \end{array}\right)$	**12.19**	$\left(\begin{array}{ccc\|c} 30 & 3 & 1 & 1 \\ 1 & -10^{-2} & 5 & 12 \\ 2 & 40 & 10^{4} & 20 \end{array}\right)$
12.10	$\left(\begin{array}{ccc\|c} 5 & -3 & -1 & -3 \\ -1 & 8 & 2 & 4 \\ 2 & 3 & 10^{8} & 1 \end{array}\right)$	**12.20**	$\left(\begin{array}{ccc\|c} 5 & -3 & -1 & -3 \\ 11 & 8\cdot 10^{3} & 2 & 40 \\ 2 & 30 & 10^{4} & 1 \end{array}\right)$

Практичне заняття № 04

РОБОТА З МАТРИЦЯМИ, МАТРИЧНИЙ АНАЛІЗ

Аудиторне заняття

1. Написати М-файл *tem41a.m* для перевірки, чи має матриця A відмінні від 0 кутові мінори. Якщо відповідь "так", то зберегти цю матрицю в файлі *tem41a.mat*.

Текст М-файла :

```
%%tem41a.m
close all, clear all, clc
%A = input('? матриця 5*5 :=');
A = [ 1, -2,  3,  4,   5;...
      5,  6, -7, 21, -15;...
     -2,  4, 56, -7,  12;...
     10, 23,  1,  3,   0;...
      0,  2,  5,  7,  89]
n = length(A);
u = true;
for k = 1 : n
    i = 1:k;      % індексний масив
    d = det(A(i,i));
    if d == 0
        u = false; break
    end
end
if u
    disp('Матрицю A зберегли в файл tem41a.mat');
    save tem41a A
end
```

Виконання *tem41a.m* :

```
A =
     1    -2     3     4     5
     5     6    -7    21   -15
    -2     4    56    -7    12
    10    23     1     3     0
     0     2     5     7    89
Матрицю A зберегли в файл tem41a.mat
>>
```

2. Написати М-файл *tem42a.m* для розв'язання СЛАР $A\vec{x} = \vec{b}$ за формулами *Крамера*. Матрицю A прочитати з файла *tem41a.mat*, а вектор $\vec{b}$ згенерувати з псевдовипадкових чисел з діапазону [0,10].

Текст М-файла :

```
%%tem42a.m
close all, clear all, clc
n = exist('tem41a.mat','file');
if n == 2   % файл існує
    load tem41a
    A
    n = length(A);
    b = 10.*rand(n,1)
    d = det(A)
    if d
        i = 1:n; x = double(i');
```

```
        for k = i
            B = A; B(i,k) = b;
            dk = det(B);
            x(k) = dk./d;
        end
        disp('Розв''язок СЛАР :'); disp(x);
    else
        disp('Матриця вироджена!');
    end
else
    disp('Файл tem41a.mat відсутній!');
end
```

Виконання *tem42a.m* :

```
A =
     1    -2     3     4     5
     5     6    -7    21   -15
    -2     4    56    -7    12
    10    23     1     3     0
     0     2     5     7    89
b =
       8.1472
       9.0579
       1.2699
       9.1338
       6.3236
d =
    -3949938
Розв'язок СЛАР :
       4.1352
      -1.4114
      0.25182
   -0.0025999
     0.088826
>>
```

3. Нехай виконуються умови Теореми 3.2. (див. Лекцію №3). Написати М-файл *tem43a.m* для знаходження матриць L, U, де $A = L \cdot U$, за методом *Холецького*, використовуючи рекурентний процес [12]. Записати отриманий результат (L, U) в файл *tem43a.mat*.

Алгоритм методу Холецького, записаний у вигляді рекурентного процесу має наступний вигляд :

1. для матриці порядку $k = 2$

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} = \mathbf{L} * \mathbf{U};\ \ \mathbf{L} = \begin{pmatrix} l_{1,1} & 0 \\ l_{2,1} & l_{2,2} \end{pmatrix},\ \ \mathbf{U} = \begin{pmatrix} 1 & u_{1,2} \\ 0 & 1 \end{pmatrix} \Rightarrow$$

$$l_{1,1} = a_{1,1},\ \ u_{1,2} = a_{1,2} / a_{1,1},\ \ l_{2,1} = a_{2,1},\ \ l_{2,2} = \det(\mathbf{A}) / a_{1,1}.$$

2. <u>циклічно</u>, для матриці порядку $k = 3,\ldots,n$, з урахуванням $\mathbf{A}_{k-1} = \mathbf{L}_{k-1} * \mathbf{U}_{k-1}$, маємо

$$\mathbf{A}_k = \left(\begin{array}{c|c} \mathbf{A}_{k-1} & \vec{\mathbf{r}}_{k-1} \\ \hline \vec{\mathbf{s}}_{k-1} & a_{k,k} \end{array}\right) = \mathbf{L}_k * \mathbf{U}_k;\ \ \mathbf{L}_k = \left(\begin{array}{c|c} \mathbf{L}_{k-1} & \vec{\mathbf{0}}^{\mathrm{T}} \\ \hline \vec{\mathbf{l}}_{k-1} & l_{k,k} \end{array}\right),\ \ \mathbf{U}_k = \left(\begin{array}{c|c} \mathbf{U}_{k-1} & \vec{\mathbf{u}}_{k-1} \\ \hline \vec{\mathbf{0}} & 1 \end{array}\right);$$

$$\vec{\mathbf{s}}_{k-1} = \begin{pmatrix} a_{k,1} & \cdots & a_{k,k-1} \end{pmatrix},\ \ \vec{\mathbf{r}}_{k-1} = \begin{pmatrix} a_{1,k} \\ \vdots \\ a_{k-1,k} \end{pmatrix},\ \ \vec{\mathbf{l}}_{k-1} = \begin{pmatrix} l_{k,1} & \cdots & l_{k,k-1} \end{pmatrix},\ \ \vec{\mathbf{u}}_{k-1} = \begin{pmatrix} u_{1,k} \\ \vdots \\ u_{k-1,k} \end{pmatrix} \Rightarrow$$

$$\vec{\mathbf{l}}_{k-1} = \vec{\mathbf{s}}_{k-1} * \mathbf{U}_{k-1}^{-1},\ \ \vec{\mathbf{u}}_{k-1} = \mathbf{L}_{k-1}^{-1} * \vec{\mathbf{r}}_{k-1},\ \ l_{k,k} = a_{k,k} - \vec{\mathbf{l}}_{k-1} * \vec{\mathbf{u}}_{k-1}.$$

Текст М-файла :

```
%%tem43a.m
close all, clear all, clc
n = exist('tem41.mat','file');
if n == 2   % файл існує
    load tem41a
    disp('Матриця A'), disp(A)
    n = length(A);
    if n > 1
        z = A(1,2)./A(1,1);
        U = eye(2); U(1,2) = z;
        i = 1:2;
        L = tril(A(i,i)); L(2,2) = A(2,2)-A(2,1).*z;
        for k = 3:n
            km = k-1; i = 1 : km;
            u = L\A(i,k); l = A(k,i)/U;
            z = A(k,k)-l*u; i =0.*i;
            L = [L, i';l, z];
            U = [U, u;i, 1];
        end
    else
        L = A; U = [1];
    end
    disp('Матриця L'), disp(L)
    disp('Матриця U'), disp(U)
    disp('Матриця L*U'), disp(L*U)
    save tem43a L U
    disp('Матриці L, U зберегли в файл tem43a.mat')
else
    disp('Файл tem41a.mat відсутній!');
end
```

Виконання *tem43a.m* :

```
Матриця A
     1    -2     3     4     5
     5     6    -7    21   -15
    -2     4    56    -7    12
    10    23     1     3     0
     0     2     5     7    89

Матриця L
            1            0            0            0            0
            5           16            0            0            0
           -2            0           62            0            0
           10           43       30.125      -40.173            0
            0            2         7.75         6.75       99.115

Матриця U
            1           -2            3            4            5
            0            1       -1.375       0.0625         -2.5
            0            0            1     0.016129      0.35484
            0            0            0            1      -1.1652
            0            0            0            0            1

Матриця L*U
     1    -2     3     4     5
     5     6    -7    21   -15
    -2     4    56    -7    12
    10    23     1     3     0
     0     2     5     7    89

Матриці L, U зберегли в файл tem43a.mat
>>
```

Завдання для самостійної роботи

4. Написати М-файл *tem44.m* в якому виконати імпорт з файла *tem43a.mat* данних (L, U) розкладу матриці A і з їх допомогою знайти обернену матрицю для матриці A. Розглянути наступні варіанти розв'язку:
   а) використовуючи операцію «\»;
   б) використовуючи функції `solvL(L,B)`, `solvU(U,B)`, де L – ліва трикутна $n\times n-$матриця, U – права трикутна $n\times n-$матриця, B – матриця розмірності $n\times m$ (m може бути і 1). Ці функції написати самостійно, для цього першим рядком М-файлів `solvL.m` і `solvU.m` має бути рядок (заголовок функції) `function [X,err]=solvL(L,B)` і `function [X,err]=solvU(U,B)` відповідно.
   Порівняти отримані результати з результатом вбудованої функції `inv(A)`.
5. Нехай маємо прямокутну матрицю $B=[A,\vec{b}]$ – запис СЛАР $A\vec{x}=\vec{b}$ у вигляді розширеної матриці, при цьому матриця A – тридіагональна. Написати М-файл *tem45.m* у якому за допомогою відповідних формул методу прогонки матриця B приводиться до ступінчастого вигляду (відповідно матриця A – до верхньої трикутної).
6. Написати М-файл *tem46.m* в якому виконати генерацію квадратної матриці A порядку $n\geq 10$ з псевдовипадковими числами. Перше "нарощення" матриці A до розмірності >2 (використати оператор присвоєння для елементу $A(:,:,1)$) виконати за наступним алгоритмом: головна діагональ множиться на (100), піддіагональ множиться на (5), наддіагональ множиться на (8), а інші елементи не змінюються. Елемент $A(:,:,2)$ знаходиться з матриці $A(:,:,1)$ видаленням всіх елементів, які менші 1. Вказівка: використати логічне індексування.
7. Написати М-файл *tem47.m* в якому квадратна матриця A читається з файлу *tem41a.mat* і використовується для створення тридіагональної матриці T (у вигляді векторів $\{\vec{a},-\vec{c},\vec{b}\}$) наступним чином: $\vec{a}$ – піддіагональ A, $-\vec{c}$ – діагональ A, $\vec{b}$ – наддіагональ A. Використовуючи метод прогонки знайти визначник T. Вказівка: $\det(T)=\prod_{i=1}^{n}(a_i\alpha_i-c_i)$.
8. Написати М-файл *tem48.m* в якому квадратна матриця A генерується псевдовипадковими числами. Виконати над нею послідовні перетворення: а) приведення до (верхньої) форми Хессенберга; б) транспонування; с) приведення до (верхньої) форми Хессенберга. Перевірити: чи буде отримана матриця тридіагональна ?
9. Написати М-файл *tem49.m*, в якому будується квадратна матриця A розмірнос-ті $n\in[5,10]$ за допомогою функції *hilb(...).* Використовуючи степеневий ітераційний процес оцінити число обумовленості μ_A, враховуючи $\max_i|\lambda_i(A)|\Big/\min_i|\lambda_i(A)|\leq\mu_A$. Перевірити цю властивість за допомогою вбудованої функції *cond(...).*
10. Написати М-файл *tem410.m*, в якому будується квадратна матриця $\hat{A}$ розмір-ності $n\in[5,10]$ за допомогою функції *magic(...).* По елементам матриці $\hat{A}$ будується матриця $A=\hat{A}*B'$, а матриця $\hat{A}$ видаляється з пам'яті. Використо-вуючи степеневий ітераційний процес (варіант скалярних добутків) оцінити число обумовленості μ_A, враховуючи $\max_i|\lambda_i(A)|\Big/\min_i|\lambda_i(A)|\leq\mu_A$. Перевірити цю властивість за допомогою вбудованої функції *cond(...).*

11. Написати М-файл *tem411.m*, в якому для $n \in [6,10]$ будується квадратна матриця за наступними формулами:

$$A = wilkinson(n);\ A = A*A';\ a_{i,i} = a_{i,i} + 10*(i-1),\ i = \overline{1,n}.$$

Використовуючи знайдену для неї Жорданову форму побудувати характеристичний многочлен (без використання функції *poly(...)*). Отриманий результат перевірити за допомогою вбудованої функції *poly(...)*.

12. Написати М-файл *tem412.m* в якому виконати *LU*-розклад матриці A за допомогою вбудованої функції *lu(...)*. Отриманий розклад використати для обчислення матриці A'. Порівняти отриманий результат зі значенням вбудованої функції `inv(A)`.
13. Написати М-файл *tem413.m* в якому виконати *QR*-розклад матриці A за допомогою вбудованої функції *qr(...)*.Отриманий розклад використати для обчислення матриці A'. Порівняти отриманий результат зі значенням вбудованої функції `inv(A)`.
14. Нехай маємо прямокутну матрицю $B = [A, \vec{b}]$ – запис СЛАР $A\vec{x} = \vec{b}$ у вигляді розширеної матриці. Написати М-файл *tem414.m* у якому знаходиться розв'язок цієї СЛАР за допомогою вбудованої функції *rref(...)*. Перевірити, чи буде цей розв'язок співпадати з розв'язком, отриманим методом прогонки для СЛАР $H*\vec{y} = \vec{f}$, де $[P,H] = hess(A'*A)$; $\vec{f} = P*A'*\vec{b}$; $\vec{x} = P'*\vec{y}$?.
15. Написати М-файл *tem415.m* в якому виконати сингулярний розклад матриці A. Перевірити, чи виконуються умова $A*V == U*D \,\&\, A'*U == V*D$. Навести приклади вбудованих функцій MATLAB, де використовується функція *svd(...)*.
16. Написати М-файл *tem416.m* в якому вводиться матриця A симетрична або ермітова (перевірити !). Використати вбудовану функцію `chol(A)` для перевірки, чи є A додатно означена матриця ($A > 0$) ? Якщо "ні", то використовуючи зміну окремого елемента в діалоговому режимі, перетворити її в додатно означену. Для матриці $A > 0$ показати верхню трикутну матрицю R розкладу Холецького. Зауваження: при зміні окремого елемента передбачити введення значень індексів i, j і значення $A(i, j)$. Якщо $i \neq j$, то враховуємо вид матриці A симетрична | ермітова і змінюємо $A(i, j)$, $A(j, i)$ відповідним чином.
17. Написати М-файл *tem417.m* в якому квадратна матриця A розмірності $n \in [5,10]$ обчислюється наступним чином: $A = pascal(n)$. Визначити, який знак порівнян-ня $\{\leq, =, \geq\}$ необхідно поставити між *cond(A)* і *cond(A'A)*. Аналогічне питання вирішити для матриці A і її *LU*-розкладу.
18. Написати М-файл *tem418.m* в якому квадратна матриця A розмірності $n \in [5,10]$ обчислюється наступним чином: $A = wilkinson(n)$. Визначити, який знак порівняння $\{\leq, =, \geq\}$ необхідно поставити між *cond(A)* і *cond(A'A)*. Аналогічне питання вирішити для матриці A і її *QR*-розкладу.

Практичне заняття № 05

НАБЛИЖЕННЯ ФУНКЦІЙ. ЗАДАЧІ ЧИСЛОВОГО АНАЛІЗУ

Аудиторне заняття

Форма Лагранжа для ІП дозволяє представити ций поліном через значення функції у вузлах інтерполяції $f(X_i)$ у вигляді

$$L_{n-1}(x)=\sum_{k=1}^{n} f(X_k)l_k(x).$$

В подальшому будемо говорити, що це представлення ІП в формі "*лагранжіана*" (адже існують і інші форми !). Для знаходження поліномів $l_k(x)$ (n-1)-степеня використаємо, що це інтерполяційний поліном, тобто:

$$L_{n-1}(X_i)\equiv\sum_{k=1}^{n} f(X_k)l_k(X_i)=f(X_i),\ \ i=\overline{1,n},$$

а отже має виконуватись співвідношення $l_k(X_i)=\begin{cases}1, k=i\\0, k\neq i\end{cases}$. З огляду, що $l_k(X_i)=0,\ k\neq i$, тобто відомі корені полінома $l_k(x)$, отримуємо його вид

$$l_k(x)=c_k(x-X_1)(x-X_2)\cdots(x-X_{k-1})(x-X_{k+1})\cdots(x-X_n),$$

а враховуючи $l_k(X_i)=1,\ k=i$ – отримуємо значення коефіцієнта c_k

$$(c_k)^{-1}=(x_k-X_1)(x_k-X_2)\cdots(x_k-X_{k-1})(x_k-X_{k+1})\cdots(x_k-X_n).$$

Якщо ввести поліном $\omega(x)=\prod_{i=1}^{n}(x-X_i)$, то запис $l_k(x)$ спрощується і має вид

$$l_k(x)=\frac{\omega(x)}{(x-X_k)\omega'(X_k)}.$$

Запишемо формулу $L_{n-1}(x)$ детально, в «розгорнутому» вигляді

$$\begin{aligned}L_{n-1}(x)=&f(X_1)\frac{(x-X_2)(x-X_3)\cdots(x-X_n)}{(X_1-X_2)(X_1-X_3)\cdots(X_1-X_n)}+\\&+f(X_2)\frac{(x-X_1)(x-X_3)\cdots(x-X_n)}{(X_2-X_1)(X_2-X_3)\cdots(X_2-X_n)}+\cdots+\\&+f(X_n)\frac{(x-X_1)(x-X_2)\cdots(x-X_{n-1})}{(X_n-X_1)(X_n-X_2)\cdots(X_n-X_{n-1})}.\end{aligned}$$

Зауваження.

1. Впорядкованість вузлів інтерполяції $\vec{\mathbf{X}}$ не є обов'язкова.
2. Якщо виконати розкриття всіх дужок в лагранжіанній формі, то отримаємо ІП в поліноміальній формі.
3. Додавання, при необхідності, ще одного вузла $\tilde{X}$ в сітку $\vec{\mathbf{X}}$ приводить до перерахунку усіх поліномів $l_k(x)$.

Форма Ньютона є представленням ІП у вигляді різницевого аналога формули Тейлора. Згадаємо деякі означення (див. Лекцію № 4).

Означення. *Поділені різниці* 1-*го порядку* ($ЇР_1$) для функції $f(x)$ визначаються наступним чином

$$f(X_i; X_j) \equiv f_{i;j} = \frac{f(X_i) - f(X_j)}{X_i - X_j} = f_{j;i}, \quad (i \neq j),$$

а поділені різниці вищих порядків ($ЇР_2$, $ЇР_3$,…) визначаються рекурентно:

$$f(X_i; X_j; X_k) \equiv f_{i;j;k} = \frac{f_{i;j} - f_{j;k}}{X_i - X_k}, \ \ldots, f_{i;j;\ldots;s;k} = \frac{f_{i;j;\ldots;s} - f_{j;\ldots;s;k}}{X_i - X_k}.$$

Поділені різниці зручно обчислювати по схемі, записуючи їх в таку таблицю :

X_i	$f_i = f(X_i)$	$ЇР_1$	$ЇР_2$	$\cdots$	$ЇР_{n-1}$
X_1	f_1	–	–	–	–
X_2	f_2	$f_{1;2}$	–	–	–
X_3	f_3	$f_{2;3}$	$f_{1;2;3}$	$\cdots$	–
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
X_{n-1}	f_{n-1}	$f_{n-2;n-1}$	$f_{n-3;n-2;n-1}$	$\cdots$	–
X_n	f_n	$f_{n-1;n}$	$f_{n-2;n-1;n}$	$\cdots$	$f_{1;2;\ldots;n-1;n}$

(*)

Інтерполяційний поліном у формі Ньютона має вид

$$N_{n-1}(x) = f(X_1) + (x - X_1) f_{1;2} + (x - X_1)(x - X_2) f_{1;2;3} + \ldots + (x - X_1)(x - X_2)\ldots(x - X_{n-1}) f_{1;2;\ldots;n} \qquad (**)$$

і, як бачимо, використовує елементи діагоналі цієї таблиці.

Розглянемо процес отримання представлення ІП в ньютонівській формі. Нехай маємо $P_{n-1}(x)$ – деякий довільний поліном (n-1)-степеня. Поліном $P_{n-1}(x) - P_{n-1}(X_1)$ ділиться націло на $(x - X_1)$ і маємо, згідно означення ПР :

$$\frac{P_{n-1}(x) - P_{n-1}(X_1)}{x - X_1} = P_{n-2}(x; X_1) \text{ – це } ЇР_1 \text{ і є поліном } (n\text{-}2)\text{-степеня.}$$

Аналогічно, поліном $P_{n-2}(x; X_1) - P_{n-2}(X_1; X_2)$ ділиться націло на $(x - X_2)$ і маємо, згідно означення ПР :

$$\frac{P_{n-2}(x; X_1) - P_{n-2}(X_1; X_2)}{x - X_2} = P_{n-3}(x; X_1; X_2) \text{ – це } ЇР_2 \text{ і є поліном } (n\text{-}3)\text{-степеня.}$$

Продовжуючи ці обчислення, отримаємо, що поліном $P_0(x; X_1; \ldots; X_{n-1})$ буде $ЇР_{n-1}$ і є константа, а всі подальші поділені різниці – будуть нулями, тобто такий ланцюжок обчислень обривається на $ЇР_n = 0$.

Отже ми отримали наступні формули :

$P_{n-1}(x) = P_{n-1}(X_1) + (x - X_1) * P_{n-2}(x; X_1);$

$P_{n-2}(x; X_1) = P_{n-2}(X_1; X_2) + (x - X_2) * P_{n-3}(x; X_1; X_2);$

…

$P_1(x; X_1; \ldots; X_{n-2}) = P_1(X_1; \ldots; X_{n-1}) + (x - X_{n-1}) * P_0(x; X_1; \ldots; X_{n-1});$

$P_0(x; X_1; \ldots; X_{n-1}) = P_0(X_1; \ldots; X_n) + (x - X_n) * 0.$

Якщо виконати послідовні підстановки отриманих значень починаючи з останнього рівняння і до першого, то отримаємо

$$P_{n-1}(x) = P_{n-1}(X_1) + (x - X_1) * (P_{n-2}(X_1; X_2) + (x - X_2) * (\ldots + (x - X_{n-1}) * P_0(X_1; \ldots; X_n))).$$

Візьмемо тепер $P_{n-1}(x)$ – інтерполяційний поліном для $f(x)$, а отже $P_{n-1}(X_1) = f(X_1)$ і поділені різниці полінома $P_{n-1}(x)$ і поділені різниці $f(x)$ співпадають. Враховуючи, що поліноми $P_{n-1}(x)$, $P_{n-1}(x)$ є поліноми (n-1)-степеня і співпадають по значенням в точках $\vec{\mathbf{X}}$, то маємо $P_{n-1}(x) \equiv P_{n-1}(x)$, а отже отримуємо для ІП формулу

$$N_{n-1}(x) = f_1 + (x - X_1) * (f_{1;2} + (x - X_2) * (\ldots + (x - X_{n-1}) * f_{1;2;\ldots;n})), \qquad (***)$$

представлену в виді, оптимальному для розрахунку (схема Горнера).

ІП в формі Ньютона зручно використовувати як при «ручних» розрахунках, так і на ЕОМ. Для підвищення точності розрахунків значення вузлів $\bar{\mathbf{X}}$ бажано упорядкувати так, щоб точка обчислення була поблизу X_1. Якщо в розрахунки додаємо ще один вузол інтерполяції $\tilde{X}$ («підключаємо» нову точку), то розмістивши її в кінці таблиці (*), виконуємо перерахунок тільки тих ПР, в які вона входить. При виконанні контролю точності розрахунків (новий доданок по модулю менше деякого ε) ми обираємо для розрахунків $N_{n-1}(x)$ у виді (**), а якщо використовуємо ВСІ вузли $\bar{\mathbf{X}}$, то обераємо $N_{n-1}(x)$ у виді (***).

ІП у формі Ньютона зручно використовувати і для інтерполювання $x=\varphi(y)$ – оберненої функції до функції $y=f(x)$. Для цього необхідно поміняти місцями стовпчики X_i і $f(X_i)$ в таблиці (*), знайти ПР виду $X_{i;j;\ldots;k}$ і побудувати ІП :

$$\tilde{N}_{n-1}(y)=X_1+(y-f_1)*(X_{1;2}+(y-f_2)*(\ldots+(y-f_{n-1})*X_{1;2;\ldots;n})).$$

Цей підхід дає змогу будувати певний ітераційний процес знаходження кореня рівняння $f(x)=0$.

Розглянемо MATLAB-реалізацію побудови ІП за формулами Лагранжа і Ньютона, а також порівняємо їх результати роботи з обчисленнями за допомогою вбудованих функцій *polyfit*, *interp1(…,'nearest')*, *interp1*, *spline*.

Маємо наступні зовнішні М-функції *pr05_1*, *ip_L*, *ip_N* :

```
%%pr05_1.m
close all, clear all, clc, warning off

f = @(x)(0.5.*x.*(cos(x)+sqrt(x)));

a = 0; b = 4; n = 7; t = 2.34;
x = linspace(a, b, n+1); y = f(x);

% Побудова ІП за формулами Лагранжа
[L, err] = ip_L(x, y);
if ~err
   disp('ІП за формулами Лагранжа');
   disp(L);
end

% Побудова ІП за формулами Ньютона
[N, err] = ip_N(x, y);
if ~err
   disp('ІП за формулами Ньютона');
   disp(N);
end
% Використання функцій MATLAB
P = polyfit(x, y, n);
disp('ІП функція MATLAB');
disp(P);

% Обчислення в точці t
ft = [f(t), polyval(L,t), polyval(N,t), polyval(P,t)];
disp('     f(t)      L(t)      N(t)      P(t)');
disp(ft);
```

```
ft(2:4) = [interp1(x,y,t,'nearest'),...
           interp1(x,y,t), spline(x,y,t)];
disp('     f(t)  S0(t;f)  S1(t;f)  S3(t;f)');
disp(ft); clear ft

T = linspace(a, b, 10*n+1); Y = f(T);
PT = polyval(P, T); S0T = interp1(x,y,T,'nearest');
S1T = interp1(x,y,T); S3T = spline(x,y,T);

%  Графіки порівняння вихідної функції і інтерполяцій
subplot(2,2,1);
plot(T,Y,'k -',T,PT,x,y,'o'); grid on; xlabel('x'); ylabel('y');
title('f(T), P(T)');
subplot(2,2,2);
plot(T,Y,'k -',T,S0T,x,y,'o'); grid on; xlabel('x'); ylabel('y');
title('f(T), S_{0,1}(T;f)');

subplot(2,2,3);
plot(T,Y,'k -',T,S1T,x,y,'o'); grid on; xlabel('x'); ylabel('y');
title('f(T), S_{1,1}(T;f)');

subplot(2,2,4);
plot(T,Y,'k -',T,S3T,x,y,'o'); grid on; xlabel('x'); ylabel('y');
title('f(T), S_{3,1}(T;f)');



function [L, err] = ip_L(X, y)
    % Побудова інтерполяційного поліному у формі Лагранжа
    % вхідні параметри:
    %   X - вектор вузлів інтерполювання (відсутні кратні)
    %   y - вектор значень функції у вузлах інтерполювання
    % вихідні параметри:
    %   L - вектор, який є представленням інтерполяційного поліному Lm(t)
    % err - 1= при перевірці вхідних даних знайдена помилка і
    %               виконання функції аварійно закінчено
    %        0= перевірка вхідних даних пройшла успішно.

    L = sort(X); err = 0;
    if nargin<2
         err = 1;
         error('ip_L:NotEnoughInputs',...
                'Not enough input arguments. See ip_L.')
    end
    n = length(X);
    % перевірка, чи є кратні вузли інтерполювання ?
    c = 0; i = 1;
    while ~c & i<n
         j = i;  i = i+1; c = L(j)==L(i);
    end
    if c
         err = 1;
         error('ip_L:NotEnoughInputs',...
                ' Є кратні вузли інтерполювання. See ip_L.')
    end
    % задання Lm(x)=0 і обчислення поліномів Wi(x)= x-X(i), i=1,...,n
    I = 1:n;
    for i = I
         W(i,:) = [1; -X(i)];  L(i) = 0;
    end
    % побудова інтерполяційного поліному Lm(x)
    for i = I
```

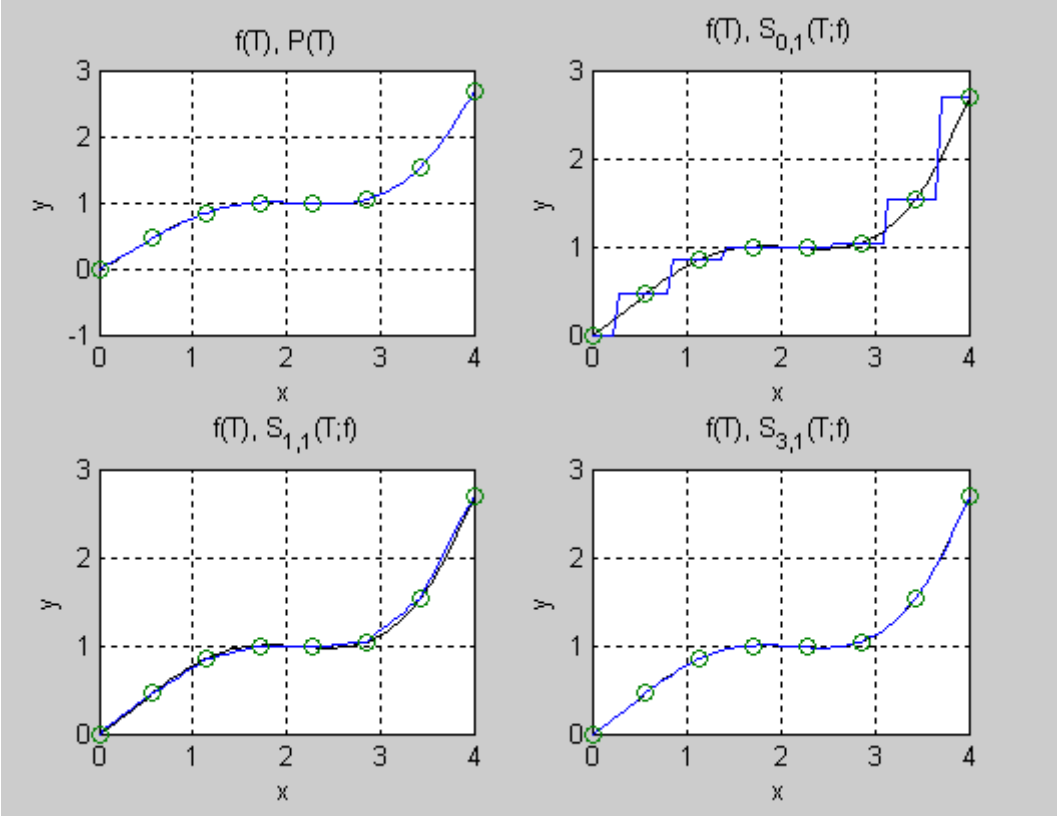
```
        c = y(i);  l = [1];
        for j = I
            if j ~= i
                c = c./(X(i)-X(j)); l = conv(l, W(j,:));
            end
        end
        l = c.*l;  L = L+l;
    end
end


function [N, err] = ip_N(X, Y)
    % Побудова інтерполяційного поліному у формі Ньютона
    % вхідні параметри:
    %   X - вектор вузлів інтерполювання (відсутні кратні)
    %   Y - вектор значень функції у вузлах інтерполювання
    % вихідні параметри:
    %   N - вектор, який є представленням інтерполяційного поліному Nm(t)
    % err = 1- при перевірці вхідних даних знайдена помилка і
    %           виконання функції аварійно закінчено
    %       0- перевірка вхідних даних пройшла успішно.

    N = sort(X); err=0;
    if nargin<2
        err = 1;
        error('ip_N:NotEnoughInputs',...
              'Not enough input arguments. See ip_N.')
    end
    n = length(X); nm = n - 1;
    % перевірка, чи є кратні вузли інтерполювання ?
    c = 0; i = 1;
    while ~c & i<n
        j = i;  i = i + 1; c = N(j) == N(i);
    end
    clear N;
    if c
        err = 1;
        error('ip_N:NotEnoughInputs',...
              ' Є кратні вузли інтерполювання. See ip_N.')
    end
    % обчислення поділених різниць і
    %   поліномів Wi(x)= x-X(i), i=1,...,n-1;
    for i = 1 : nm
        for j = n : -1 : i+1
            Y(j) = (Y(j-1)-Y(j))./(X(j-i)-X(j));
        end
        W(i,:) = [1; -X(i)];
    end
    % побудова інтерполяційного поліному Nm(x)
    j = 1; N(j) = Y(n);
    for i = nm : -1 : 1
        N = conv(W(i,:), N);
        j = j+1; N(j) = N(j) + Y(i);
    end
end
```

Виконання функції *pr05_1* :

```
ІП за формулами Лагранжа
 0.00019802 -0.0048962   0.025888   0.037937   -0.42227   0.48125   0.65186    0
ІП за формулами Ньютона
 0.00019802 -0.0048962   0.025888   0.037937   -0.42227   0.48125   0.65186    0
```

```
ІП функція MATLAB
 0.00019802 -0.0048962   0.025888   0.037937   -0.42227   0.48125   0.65186 -3.3047e-015
     f(t)      L(t)      N(t)      P(t)
      0.97595       0.97597       0.97597       0.97597
     f(t)   S0(t;f)   S1(t;f)   S3(t;f)
      0.97595       0.97863        0.9848       0.97594
>>
```

і отримали такі графіки :



Завдання для самостійної роботи

1. Побудувати за формулами Лагранжа І П n-го степеня (n=5, 7, 9) для функції y та обчислити його значення в заданій точці x.

№	Функція	Відрізок	Точка *x*
1.1.	$y=\frac{x^2+5}{lg(x+1)+6}$	[0, 5]	2.5
1.2.	$y=\frac{x^2+5\cos x}{6+sin(x+1)}$	[0, 5]	3.0
1.3.	$y=\frac{x^2+\sin x}{3+\cos(x-1)}$	[0, 4]	1.5
1.4.	$y=\frac{x}{3}\left(\sin x+\sqrt{x}\right)$	[0, 4]	2.0
1.5.	$y=x^2+6x$	[3, 7]	4.5
1.6.	$y=\frac{2x}{7x+4}$	[1, 6]	3.0
1.7.	$y=\frac{5x}{x^3+5}$	[1, 9]	6.0

1.8.	$y=\frac{3}{x^2+1}$	[1, 9]	5.0
1.9.	$y=\frac{2x+4}{x+4}$	[2, 6]	3.5
1.10.	$y=\frac{x^2+4}{x^3+3}$	[1, 6]	2.5

2. Знайти наближене значення функції y, заданої таблично, для заданого значення аргументу x, за допомогою І П розрахованого за формулами Лагранжа.

2.1. x=0.72

i	0	1	2	3	4	5
x_i	0.43	0.48	0.55	0.62	0.70	0.75
y_i	1.63597	1.73234	1.87686	2.03345	2.22846	2.33973

2.2. x=0.21

i	0	1	2	3	4	5
x_i	0.02	0.08	0.12	0.17	0.23	0.30
y_i	1.02316	1.09590	1.14725	1.21483	1.30120	1.40976

2.3. x=0.49

i	0	1	2	3	4	5
x_i	0.35	0.41	0.47	0.51	0.56	0.64
y_i	2.73951	2.30080	1.96864	1.78776	1.59502	1.34310

2.4. x=0.58

i	0	1	2	3	4	5
x_i	0.41	0.46	0.52	0.60	0.65	0.72
y_i	2.57418	2.32513	2.09336	1.86203	1.74926	1.62098

2.5. x=0.95

i	0	1	2	3	4	5
x_i	0.68	0.73	0.80	0.88	0.93	0.99
y_i	0.80866	0.89492	1.02964	1.20966	1.34087	1.52368

2.6. x=0.33

i	0	1	2	3	4	5
x_i	0.11	0.15	0.21	0.29	0.35	0.40
y_i	9.05421	6.61659	4.69170	3.35106	2.73951	2.36522

2.7. x=0.26

i	0	1	2	3	4	5
x_i	0.05	0.10	0.17	0.25	0.30	0.36
y_i	0.050042	0.100335	0.171657	0.255342	0.309336	0.376403

2.8. x=0.19

i	0	1	2	3	4	5
x_i	0.1	0.2	0.3	0.4	0.5	0.6
y_i	0.09983	0.19867	0.29552	0.38942	0.47943	0.76484

2.9. x=1.22

i	0	1	2	3	4	5	6

x_i	1.10	1.18	1.23	1.31	1.37	1.42	1.49
y_i	2.17520	2.30254	2.38631	2.50946	2.21730	2.22361	2.23470

2.10. x=2.34

i	0	1	2	3	4	5	6
x_i	2.01	2.07	2.33	2.35	2.47	2.49	2.50
y_i	1.27520	1.40254	1.48631	1.60946	1.31730	1.32361	1.33470

3. Побудувати за формулами Ньютона I П для функції у та обчислити його значення в заданій точці x.

№	Функція	Відрізок	Точка x
3.1.	$y=\frac{1}{4x^2+1}$	[1.1, 1.9]	1.48
3.2.	$y=\frac{1}{\sqrt{x^2+1}}$	[1.2, 2.0]	1.50
3.3.	$y=\frac{1}{x^2+7x+6}$	[1.4, 2.2]	1.66
3.4.	$y=\frac{1}{x^2+\sqrt{x}+3}$	[1.1, 2.0]	1.46
3.5.	$y=\frac{1}{3x^3+4}$	[1.0, 1.4]	1.22
3.6.	$y=\frac{1}{x^5+2x+1}$	[1.4, 2.3]	1.84
3.7.	$y=\frac{1}{x\sqrt{x^2+7}}$	[1.4, 2.3]	1.63
3.8.	$y=\frac{1}{\sqrt{2x^2+5}}$	[1.2, 2.1]	1.48
3.9.	$y=\frac{1}{3x\sqrt{x+4}}$	[1.2, 2.1]	1.58
3.10.	$y=\frac{1}{\sqrt{x^3+3x^2+4}}$	[1.1, 2.0]	1.56

4. Знайти наближене значення функції у, заданої таблично, для заданого значення аргументу x, за допомогою I П побудованого за формулами Ньютона.

4.1. x=0.32

i	0	1	2	3	4	5	6
x_i	0.298	0.303	0.310	0.317	0.323	0.330	0.339
y_i	3.25578	3.17639	3.12180	3.04819	2.98755	2.91950	2.83598

4.2. x=3.52

i	0	1	2	3	4	5	6
x_i	0	1	2	3	4	5	6
y_i	5.2	8.0	10.4	12.4	14.0	15.2	16.0

4.3. x=0.44

i	0	1	2	3	4	5	6
x_i	0	0.2	0.3	0.4	0.7	0.9	1.0

y_i	132.651	140.877	157.464	166.375	195.112	216.000	234.122

4.4. x=2.26

i	0	1	2	3	4	5	6
x_i	0.5	1.0	2.0	2.5	3.0	3.5	4.0
y_i	1.21065	2.15123	4.21023	5.12565	6.11134	7.15002	8.21012

4.5. x=0.63

i	0	1	2	3	4	5	6
x_i	0.593	0.598	0.605	0.613	0.619	0.627	0.632
y_i	0.532050	0.535625	0.540598	0.546235	0.550431	0.555983	0.559428

4.6. x=0.72

i	0	1	2	3	4	5	6	7	8
x_i	0.698	0.706	0.714	0.727	0.736	0.747	0.760	0.769	0.782
y_i	2.22336	2.24382	2.26446	2.29841	2.32221	2.35164	2.38690	2.41162	2.44777

4.7. x=0.13

i	0	1	2	3	4	5	6	7
x_i	0.100	0.108	0.119	0.127	0.135	0.146	0.157	0.169
y_i	1.12128	1.13160	1.14594	1.15648	1.16712	1.18191	1.19689	1.21344

4.8. x=0.26

i	0	1	2	3	4	5	6	7	8
x_i	0.235	0.240	0.250	0.255	0.265	0.280	0.295	0.300	0.305
y_i	1.20800	1.21256	1.22169	1.22628	1.23547	1.24933	1.26328	1.26795	1.27263

4.9. x=0.10

I	0	1	2	3	4	5	6	7
x_i	0.095	0.102	0.104	0.107	0.110	0.112	0.116	0.120
y_i	1.09131	1.23490	1.27994	1.35142	1.42815	1.48256	1.60033	1.73205

4.10. x=0.13

I	0	1	2	3	4	5	6	7
x_i	0.103	0.108	0.115	0.120	0.128	0.136	0.141	0.150
y_i	2.01284	2.03342	2.06070	2.07918	2.10721	2.13354	2.14922	2.17609

5. Провести поліноміальну інтерполяцію аналітично заданих функцій, поданих у завданні 1, використовуючи вбудовані функції MATLAB.

6. Здійснити кусково-поліноміальну інтерполяцію таблично заданих функцій, поданих у завданні 4, використовуючи вбудовані функції MATLAB.

7. Обчислити першу і другу різницеві похідні (у внутрішніх вузлах сітки) для функцій, заданих у завданнях 1, 3.

8. Побудувати сплайни $S_{0,1}(x;u)$, $S_{1,1}(x;u)$, $S_{3,1}(x;u)$ для функції у та обчислити його значення в заданій точці x. Варіанти завдань – задані у 1, 3.

9. Побудувати сплайни $S_{0,1}(x;u)$, $S_{1,1}(x;u)$, $S_{3,1}(x;u)$ для дискретної функції, заданої у 2. Вивести отримані поліноми на кожному з відрізків сітки.

10. Обчислити визначені інтеграли за формулами трапецій і Сімпсона, використовуючи вбудовані функцій MATLAB.

	Формула трапецій	Формула Сімпсона
8.1.	$\int_{0.8}^{1.6} \frac{dx}{\sqrt{2x^2+1}}$	$\int_{1.2}^{2} \frac{lg(x+2)}{x}dx$
8.2.	$\int_{1.2}^{2.7} \frac{dx}{\sqrt{x^2+3.2}}$	$\int_{1.6}^{2.4} (x+1)\sin x dx$
8.3.	$\int_{1}^{2} \frac{dx}{\sqrt{2x^2+1.3}}$	$\int_{0.2}^{1} \frac{tg(x^2)}{x^2+1}dx$
8.4.	$\int_{0.2}^{1.2} \frac{dx}{\sqrt{x^2+1}}$	$\int_{0.6}^{1.4} \frac{\cos x}{x+1}dx$
8.5	$\int_{0.8}^{1.4} \frac{dx}{\sqrt{2x^2+3}}$	$\int_{0.4}^{1.2} \sqrt{x}\cos(x^2)dx$
8.6.	$\int_{0.4}^{1.2} \frac{dx}{\sqrt{2+0.5x^2}}$	$\int_{0.8}^{1.2} \frac{\sin 2x}{x^2}dx$
8.7.	$\int_{1.4}^{2.1} \frac{dx}{\sqrt{3x^2-1}}$	$\int_{0.8}^{1.6} \frac{lg(x^2+1)}{x}dx$
8.8	$\int_{1.2}^{2.4} \frac{dx}{\sqrt{0.5+x^2}}$	$\int_{0.4}^{1.2} \frac{\cos x}{x+2}dx$
8.9.	$\int_{0.4}^{1.2} \frac{dx}{\sqrt{3+x^2}}$	$\int_{0.4}^{1.2} (2x+0.5)\sin x dx$
8.10.	$\int_{2}^{3.5} \frac{dx}{\sqrt{x^2-1}}$	$\int_{1.3}^{2.1} \frac{\sin(x^2-1)dx}{2\sqrt{x}}$

11. Скласти програми для обчислення визначених інтегралів із завдання 4 за формулами трапецій, Сімпсона, лівих, правих і центральних прямокутників.

Практичне заняття № 06

ВИКОРИСТАННЯ М-ФУНКЦІЙ

Аудиторне заняття

Розглянемо опис і виклик М-функцій, зокрема зовнішніх, підфункцій, вкладених, анонімних і *inline*-функції, а також роботу з вбудованими рядковими функціями на базі задачі № 23 і вимог, які ставляться до реалізації цієї задачі. В цьому прикладі також будемо використовувати різні варіанти завдання вхідних і вихідних параметрів, в тому числі і змінну їх кількість.

Маємо наступні зовнішні М-функції *tem61a, search_region_root :*

```
function tem61a                     % зовнішня функція
    close all, clear all, clc, warning off
    %disp('Введіть рядковий вираз для многочлена з елементами b*x**k');
    %p_str = input('? p= ');
    p_str = '  -412.99*x**4+153*x**0  +17.301*x**6 +230.86*x**5+1*x**7   '
    p_str = strrep(p_str, ' ','')
    P = str_to_poly(p_str);
    if ~length(P)
        disp('Рядковий вираз для многочлена ПУСТИЙ !'); return
    end
    disp('P(x)='); disp(P);
    p_str = strrep(p_str, '**','.^');
    p = inline(p_str);
    Fp = @(x)(feval(p,x).^2);
    D = search_region_root(P);
    [n,m] = size(D); Xr = zeros(1,n);
    for k = 1 : n
         Xr(k) = fminbnd(Fp, D(k,1), D(k,2));
    end
    disp('Знайдено корені P/min(Fp)'); disp(Xr);
    Xr = roots(P);
    disp('Всі корені полінома P'); disp(Xr);
end

function P = str_to_poly(p_str)    % підфункція
%  перетворення рядка p в MATLAB-поліном P
    function j = poz(s, c, j)      % вкладена функція
        i = findstr(s, c);
        if length(i)
            j = min(j,i(1));
        end
    end

    s = p_str; Koef = []; Power = []; e = '*x**'; k = 0;
    while length(s)
        i = findstr(s, e); n = length(i);
        if n > 0
            k = k+1; j = i(1);
            Koef(k) = str2num(s(1:j-1));
            s = s(j+4:end);
            if n == 1
                Power(k) = str2num(s); s = ''; break
            end
            j = inf; j = poz(s,'+',j); j = poz(s,'-',j);
            if j < inf
```

```
                Power(k) = str2num(s(1:j-1));
                s = strtrim(s(j:end));
            else
                error('Недопустимий розділовий знак між елементами b*x**k')
            end
        else
            error('Відсутній елемент *x** в записі многочлена')
        end
    end
    if k == 0
        P = []; return
    end
    k = max(Power); P = zeros(1,k+1); j = 0;
    for i = Power
        j = j+1; P(k-i+1) = Koef(j);
    end
end


function varargout = search_region_root(P_f, varargin)      % зовнішня функція
% Графічний пошук області існування коренів полінома P або функції P_f(x).
% varargin{1} := t_in - типізація P_f.
%                Допустимі значення 1|2, де
%             1:= P_f - MATLAB-поліном (за угодою);
%             2:= P_f - функція виду f(x).
% varargin{2} := k_out - значення варіанта списка вихідного параметра.
%                Допустимі значення 1|2|3, де
%             1:= список є {D} - масив діапазонів коренів (за угодою);
%             2:= список є {x0} - масив початкового наближення до коренів;
%             3:= список є {D, x0} - діапазони коренів і початкові наближення.

    if nargin == 0
        error('Пустий список вхідних параметрів')
    end
    % ініціалізація   t_in, k_out :
    if nargin == 1
        t_in = 1; k_out = 1;
    else
        t_in = varargin{1};
        if t_in < 0 | t_in > 2
            error('Недопустиме значення типізації 1-го вхідного параметра')
        end
        if nargin == 2
            k_out = 1;
        elseif nargin == 3
            k_out = varargin{2};
            if k_out < 0 | k_out > 3
                e='Недопустиме значення варіанта списка вихідного параметра';
                error(e)
            end
        else
            error('Неправильно задано список вхідних параметрів')
        end
    end

    disp('Введіть область [a,b] і крок h для побудови графіка');
    e = 1;
    while e
        a = input('? a='); b = input('? b='); h = input('? h=');
        D = [a : h : b]; close all;
        if t_in == 1
            Y = polyval(P_f, D);
```

```
        else
            Y = P_f(D);
        end
        plot(D, Y), grid on
        e = input('? продовжимо шукати області з коренями (1-да | 0-ні) =');
    end
    clear D Y
    if k_out == 1 | k_out == 3
        D = input('? Масив з діапазонами коренів=');
        varargout(1) = {D};
    end
    if k_out > 1
        x0 = input('? Масив з наближенням до коренів=');
        varargout(k_out-1) = {x0};
    end
end
```

Результати тестового запуску *tem6a* для значення *p_str*, вказаного в п.**b.** вимог задачі №23, будуть наступні (реалізація Octave і графік не приводимо) :

```
p_str =   -412.99*x**4+153*x**0  +17.301*x**6 +230.86*x**5+1*x**7
p_str = -412.99*x**4+153*x**0+17.301*x**6+230.86*x**5+1*x**7
P(x)=
     1.0000    17.3010   230.8600  -412.9900         0         0         0   153.0000
Введіть область [a,b] і крок h для побудови графіка
? a=-0.8
? b=1.55
? h=0.01
? продовжимо шукати області з коренями (1-да | 0-ні) =0
? Масив з діапазонами коренів=[-0.75,-0.5;0.9,1;1.4,1.5]
Знайдено корені P/min(Fp)
  -0.7196   0.9722   1.4715
Всі корені полінома P
   -9.4424 + 13.1000i
   -9.4424 - 13.1000i
    1.4715 +       0i
    0.9722 +       0i
   -0.7196 +       0i
   -0.0701 +  0.7517i
   -0.0701 -  0.7517i
>>
```

Завдання для самостійної роботи

1. При виконанні умови Теореми 3.2. (Лекція №3) написати М-функцію розв'язання СЛАР *Ax=b*. Звертання до цієї функції повинно мати вигляд: *x=sol_hol(A,b)* або [*x,L,U*]=*sol_hol(A,b)* (тобто мати реалізацію зі змінною кількістю вихідних параметрів (*x* | *x,L,U*)). Функцію *sol_hol* реалізувати за допомогою таких підфункцій: [*L,U*]=*hol(A)*, *x=sol_L(L,b)*, *x=sol_U(U,b)*, де функція *hol* – реалізує метод Холецького знаходження матриць *L,U* з використанням рекурентного процесу [12]; функція *sol_L* – реалізує розв'язання СЛАР *Lx=b* з нижньою трикутною матрицею; функція *sol_U* – реалізує розв'язання СЛАР *Ux=b* з верхньою трикутною матрицею. При реалізації *sol_L*, *sol_U* операцію \ і функцію *inv()* не використовувати.
2. При виконанні умови Теореми 3.2. (Лекція №3) написати М-функцію розв'язання СЛАР *Ax=b*. Звертання до цієї функції повинно мати вигляд: *x=sol_hol1(A,b)* або [*x,L,U*]=*sol_hol1(A,b)* (тобто мати реалізацію зі змінною кількістю вихідних параметрів (*x* | *x,L,U*)). Функцію *sol_hol1* реалізувати за допомогою

підфункції [L,U]=hol(A) і зовнішньої функції x=sol_tr(t,A,b), де функція hol – реалізує метод Холецького знаходження матриць L,U з використанням рекурентного процесу [12]. Функція sol_tr – реалізує розв'язання СЛАР Ax=b з трикутною матрицею, де A – нижня|верхня при t=true|false. При реалізації sol_tr операцію \ і функцію inv() не використовувати.

3. Написати М-функцію розв'язання СЛАР Ax=b, де A=A'. Звертання до цієї функції має бути: x=sol_sqrt(A,b) або [x,L,d]=sol_sqrt(A,b) (тобто мати реалізацію зі змінною кількістю вихідних параметрів (x | x,L,d)). Функцію sol_sqrt реалізувати за допомогою таких підфункцій: [L,d]=m_srqt(A), x=sol_L(L,b), x=sol_U(U,b), де функція m_srqt – реалізує метод квадратного кореня знаходження матриці L (нижня трикутна) і вектора d=diag(D) розкладу A=L*D*L'. Функції sol_L, sol_U – означені в **п.1** цієї теми.
4. Написати М-функцію розв'язання СЛАР Ax=b, де A=A'. Звертання до цієї функції має бути: x=sol_sqrt1(A,b) або [x,L,d]=sol_sqrt1(A,b) (тобто мати реалізацію зі змінною кількістю вихідних параметрів (x | x,L,d)). Функцію sol_sqrt1 реалізувати за допомогою підфункції [L,d] = m_sqrt(A) і зовнішньої функції x=sol_tr(t,A,b), де функції m_sqrt і sol_tr – означені в **п.3** і **п.2** цієї теми, відповідно.
5. Написати М-функцію spin_max для знаходження максимального по модулю власного числа l_max і, при необхідності, відповідного власного вектора h_max матриці A ($A \neq A^{'}$), використовуючи степеневий ітераційний метод. При знаходженні тільки l_max виклик функції має бути:

   l_max=spin_max(A,x,e,m,kmax),

   де e – точність $0<\varepsilon<1$ при визначенні $\left|\lambda^{(k)}-\lambda^{(k-1)}\right|<\varepsilon$, m,kmax – див.Лекцію №3. А у випадку обох значень:

   [l_max,h_max]=spin_max(A,x,e,m,kmax,eh,kmaxh),

   де eh – точність $0<\tilde{\varepsilon}<1$ при визначенні $\left|h^{(i)}-h^{(i-1)}\right|<\tilde{\varepsilon}$, kmaxh – максимальна кількість уточнень вектора h. Степеневий ітераційний метод реалізувати підфункцією [l_max,x] = m_sip(A,x,e,m,kmax).
6. Написати М-функцію spin_maxsd для знаходження максимального по модулю власного числа l_max і, при необхідності, відповідного власного вектора h_max матриці A ($A=A^{'}$), використовуючи метод скалярних добутків. При знаходженні тільки l_max виклик функції має бути:

   l_max=spin_maxsd(A,x,e,m,kmax),

   де e – точність $0<\varepsilon<1$ при визначенні $\left|\lambda^{(k)}-\lambda^{(k-1)}\right|<\varepsilon$, m,kmax – див.Лекцію №3. А у випадку обох значень:

   [l_max,h_max]=spin_maxsd(A,x,e,m,kmax,eh,kmaxh),

   де eh – точність $0<\tilde{\varepsilon}<1$ при визначенні $\left|h^{(i)}-h^{(i-1)}\right|<\tilde{\varepsilon}$, kmaxh – максимальна кількість уточнень вектора h. Метод скалярних добутків реалізувати підфункцією [l_max,x] = m_sd(A,x,e,m,kmax).
7. Виконати **п.3** цієї теми для варіанту ермітової матриці $A=A^{*}$.
8. Виконати **п.4** цієї теми для варіанту ермітової матриці $A=A^{*}$.

9. Написати М-функцію `spin_min` для знаходження мінімального по модулю власного числа `l_min` і, при необхідності, відповідного власного вектора `h_min` матриці `A` ($A \neq A^{'}$), використовуючи степеневий ітераційний метод. При знаходженні тільки `l_min` виклик функції має бути:

`l_min=spin_min(A,x,e,m,kmax)`,

де `e` – точність $0 < \varepsilon < 1$ при визначенні $\left|\lambda^{(k)} - \lambda^{(k-1)}\right| < \varepsilon$, `m,kmax` – див.Лекцію №3. А у випадку обох значень:

`[l_min,h_min]=spin_min(A,x,e,m,kmax,eh,kmaxh)`,

де `eh` – точність $0 < \tilde{\varepsilon} < 1$ при визначенні $\left|h^{(i)} - h^{(i-1)}\right| < \tilde{\varepsilon}$, `kmaxh` – максимальна кількість уточнень вектора h. Степеневий ітераційний метод реалізувати підфункцією `[l_min,x] = m_sip(A,x,e,m,kmax)`. Розв'язання СЛАУ $A \cdot \vec{Y} = \vec{x}^{(k)}$ реалізувати за схемою: один раз виконати *LU*-розклад матриці, а в подальшому використовувати підфункції (вкладені в `m_sip`) `sol_L, sol_U` (означені в **п.1** цієї теми).

10. Написати М-функцію `spin_minxsd` для знаходження мінімального по модулю власного числа `l_min` і, при необхідності, відповідного власного вектора `h_min` матриці `A` ($A = A^{'}$), використовуючи метод скалярних добутків. При знаходженні тільки `l_min` виклик функції має бути:

`l_min=spin_minsd(A,x,e,m,kmax)`,

де `e` – точність $0 < \varepsilon < 1$ при визначенні $\left|\lambda^{(k)} - \lambda^{(k-1)}\right| < \varepsilon$, `m,kmax` – див.Лекцію №3. А у випадку обох значень:

`[l_min,h_min]=spin_minsd(A,x,e,m,kmax,eh,kmaxh)`,

де `eh` – точність $0 < \tilde{\varepsilon} < 1$ при визначенні $\left|h^{(i)} - h^{(i-1)}\right| < \tilde{\varepsilon}$, `kmaxh` – максимальна кількість уточнень вектора h. Метод скалярних добутків реалізувати підфункцією `[l_min,x] = m_sd(A,x,e,m,kmax)`. Розв'язання СЛАУ $A \cdot \vec{Y} = \vec{x}^{(k)}$ реалізувати за схемою: один раз виконати *LU*-розклад матриці, а в подальшому використовувати підфункції (вкладені в `m_sd`) `sol_L, sol_U` (означені в **п.1** цієї теми).

11. Написати М-функцію `ip_Lagr` для знаходження $L_{n-1}(x)$ – інтерполяційного полінома (ІП) в формі Лагранжа по формулі (5.4) Лекції № 5 або обчислень значень (за його допомогою) в точках сітки `T`. Побудову поліномів $\omega(x)$, $L_{n-1}(x)$ реалізувати окремими підфункціями `w_fun`, `L_fun`, а полінома $l_k(x)$ – вкладеною функцією в `L_fun`. У випадку тільки побудови ІП виклик функції має бути:

`P=ip_Lagr(X,Y)`,

де `P` – поліном $L_{n-1}(x)$ в сенсі MATLABa. А при обчислені $L_{n-1}(x)$ в точках сітки `T` виклик функції має бути:

`P=ip_Lagr(X,Y,T)`,

де `P` – масив значень $L_{n-1}(T)$.

12. Написати М-функцію `ip_Njut` для знаходження $N_{n-1}(x)$ – інтерполяційного полінома (ІП) в формі Ньютона по формулі (5.7) Лекції № 5 або обчислень значень (за його допомогою) в точках сітки `T`. Побудову масива розділених різниць і полінома $N_{n-1}(x)$ реалізувати окремими підфункціями `dr_fun`, `N_fun`, а

додавання поліномів (в сенсі MATLABa) – вкладеною функцією в `N_fun`. У випадку тільки побудови ІП виклик функції має бути:

`P=ip_Njut(X,Y)`,

де `P` – поліном $N_{n-1}(x)$ в сенсі MATLABa. А при обчислені $N_{n-1}(x)$ в точках сітки `T` виклик функції має бути:

`P=ip_Njut(X,Y,T)`,

де `P` – масив значень $N_{n-1}(T)$.

13. Написати М-функцію `dy = dy_dx(h,y)` для наближеного знаходження $u'(x)$ з локальною точністю $O(h^2)$ в точках рівномірної сітки з кроком `h` на триточковому шаблоні по формулам :

$$u'(x_i)=\begin{cases}\dfrac{-3u_1+4u_2-u_3}{2h},\ i=1;\\ \dfrac{u_{i+1}-u_{i-1}}{2h},\quad i=\overline{2,n-1};\\ \dfrac{u_{n-2}-4u_{n-1}+3u_n}{2h},\ i=n\end{cases}$$

Для реалізації формул в точках з індексами $i=\{1\},\{2\div(n-1)\},\{n\}$ написати відповідні підфункції `diff_b()`, `diff_c(i)`, `diff_e()` (перша і третя без параметрів!). Написати тестуючий М-файл для деякої функції, яка вводиться в діалоговому режимі.

14. Написати М-функцію `dy = d2y_dx2(h,y)` для наближеного знаходження $u''(x)$ з локальною точністю $O(h^2)$ в точках рівномірної сітки з кроком `h` на триточковому шаблоні по формулам :

$$u''(x_i)=\begin{cases}\dfrac{u_1-2u_2+u_3}{h^2},\quad i=1;\\ \dfrac{u_{i-1}-2u_i+u_{i+1}}{h^2},\ i=\overline{2,n-1};\\ \dfrac{u_{n-2}-2u_{n-1}+u_n}{h^2},\ i=n\end{cases}$$

Для реалізації формул в точках з індексами $i=\{1\},\{2\div(n-1)\},\{n\}$ написати відповідні підфункції `diff_b()`, `diff_c(i)`, `diff_e()` (перша і третя без параметрів!). Написати тестуючий М-файл для деякої функції, яка вводиться в діалоговому режимі.

15. Написати М-функцію `runge_romberg(x1,y1,p,x2,y2)`, зі змінною кількістю вихідних параметрів (`p_err | [p_err, new_y]`), де:

`x1,y1` – рівномірна сітка (з кроком `h`) і значення деяких обчислень на цій сітці;

`p` – порядок точності використаної розрахункової формули ($O(h^p)$);

`x2,y2` – рівномірна сітка (з кроком `k*h`) і значення деяких обчислень на цій сітці. Вказана функція може визначати: `p_err` – масив похибок (формула (5.9) Лекції №5) і, при необхідності, `new_y` – нове (уточнене) значення в точках сітки `x1` (формула (5.10) Лекції №5). Написати тестуючий М-файл.

16. Написати М-файл розв'язання СЛАР $Ax=b$ за допомогою зовнішньої М-функції `[x,k]=m_jac(A,b,eps,x,mki)`, яка реалізує матричний варіант методу Якобі. Результат перевірити за допомогою солвера СЛАР.
17. Написати М-файл розв'язання СЛАР $Ax=b$ за допомогою зовнішньої М-функції `[x,k]=m_1jac(A,b,eps,x,mki)`, яка реалізує покомпонентний варіант методу Якобі. Результат перевірити за допомогою вбудованої функції приведення до ступінчатої форми.
18. Написати М-файл розв'язання СЛАР $Ax=b$ за допомогою зовнішньої М-функції `[x,k]=m_1zej(A,b,eps,x,mki)`, яка реалізує покомпонентний варіант методу Зейделя (додаткових векторів не використовувати!). Результат перевірити за допомогою допомогою солвера СЛАР.
19. Написати М-файл розв'язання СЛАР $Ax=b$ за допомогою зовнішньої М-функції `[x,k]=m_zej(A,b,eps,x,mki)`, яка реалізує матричний варіант методу Зейделя. Результат перевірити за допомогою вбудованої функції приведення до ступінчатої форми.
20. Написати М-файл розв'язання СЛАР $Ax=b$ за допомогою зовнішньої М-функції `[x,k]=m_1rel(A,b,eps,x,mki,w)`, яка реалізує покомпонентний варіант методу верхньої релаксації (додаткових векторів не використовувати!). Результат перевірити за допомогою допомогою солвера СЛАР.
21. Написати М-файл розв'язання СЛАР $Ax=b$ за допомогою зовнішньої М-функції `[x,k]=m_vmns(A,b,eps,x,mki)`, яка реалізує метод найшвидшого спуску. Результат перевірити за допомогою вбудованої функції приведення до ступінчатої форми.
22. Написати М-файл розв'язання нелінійної системи рівнянь

1	$\begin{cases} \dfrac{x^2}{5.678}+\dfrac{y^2}{17.23}=1, \\ (x-1.32)^2+y^2=2. \end{cases}$	**11**	$\begin{cases} \dfrac{x^2}{7.8}+\dfrac{y^2}{12.3}=1, \\ 1.45x+5.1y=2. \end{cases}$
2	$\begin{cases} \dfrac{x^2}{6.8}-\dfrac{y^2}{19.9}=1, \\ x+y^2=5.2. \end{cases}$	**12**	$\begin{cases} \dfrac{x^2}{3.8}+\dfrac{y^2}{15.3}=1, \\ y-x^2=2. \end{cases}$
3	$\begin{cases} \dfrac{x^2}{7.8}+\dfrac{y^2}{27.3}=1, \\ x+y^2=2. \end{cases}$	**13**	$\begin{cases} \dfrac{x^2}{5.678}-\dfrac{y^2}{17.23}=1, \\ (x-1.2)^2+y=1. \end{cases}$
4	$\begin{cases} \dfrac{x^2}{5.678}-\dfrac{y^2}{17.23}=1, \\ (x-1.32)^2+y^2=2. \end{cases}$	**14**	$\begin{cases} \dfrac{x^2}{6.8}+\dfrac{y^2}{7.3}=1, \\ -1.32x+2.5y=2. \end{cases}$
5	$\begin{cases} \dfrac{x^2}{7.5}+\dfrac{y^2}{7.3}=2, \\ (x-1.8)^2+y^2=1. \end{cases}$	**15**	$\begin{cases} \dfrac{x^2}{5.678}+\dfrac{y^2}{17.23}=1, \\ (x-1.32)^2+y^2=2. \end{cases}$
6	$\begin{cases} \cos(0.4y+x^2)+x^2+y^2=1.6, \\ 1.5x^2-\dfrac{y^2}{0.36}=1. \end{cases}$	**16**	$\begin{cases} \sin(0.2y+x^2)+x^2+y^2=1.7, \\ x^2-\dfrac{y^2}{0.6}=1. \end{cases}$

7	$\begin{cases}\cos(0.6y+x^2)+x^2+y^2=2.3,\\ 1.405x^2-\dfrac{y^2}{0.8}=1.\end{cases}$	**17**	$\begin{cases}\sin(0.4y+x^2)+x^2+y^2=2.6,\\ 1.9x^2-\dfrac{y^2}{0.867}=1.\end{cases}$
8	$\begin{cases}x^2+2y^3=3,\\ 8x^3-3x^2y=77.\end{cases}$	**18**	$\begin{cases}x^3+2y^3=3,\\ 8y^2-5x^2=7.\end{cases}$
9	$\begin{cases}-x^2+2y^3=3,\\ 8x^3-3xy=17.\end{cases}$	**19**	$\begin{cases}2x^2+2y^3=5,\\ 4x^3-3x^2y=14.\end{cases}$
10	$\begin{cases}x^3+3y^3=3,\\ 8x-3x^2y=7.7.\end{cases}$	**20**	$\begin{cases}3x^2-y^3=6,\\ 8x^3-3xy=12.\end{cases}$

за допомогою зовнішньої М-функції, яка реалізує метод Ньютона у вигляді

```
function [X, k, err] = m_ns_njut(f_J, X, e, mki)
```

Тут `f_J` – зовнішня функція, яка при глобальній змінній `fun=0` обчислює тільки `{F(X)}`, а при `fun=1` – обчислює пару `{F(X),J(X)}`. `J(X)` – це якобіан $J(X)=\left(\dfrac{\partial F_i(X)}{\partial X_j}\right)_{i,\,j=1}^{n}$, а зміст решти вхідних | вихідних параметрів цілком зрозумілий. Надрукувати отримані корені ($X^{(end)}$), кількість виконаних ітерацій ($k=end$) для заданої точності (e) і значення функції $F(X^{(end)})$.

23. Написати М-функцію для знаходження дійсних коренів многочлена :

1	$1.2x^4-4.5x^2+5.1x-20.3$	**11**	$x^6-5.5x^4+6x-2.14$
2	$1.2x^6-5x^2+5.3x-23$	**12**	$2x^4+5x^2+3.1x-10.5$
3	$x^4-6.7x^2-3.1x-12.03$	**13**	$3x^6-2.5x^4+x^2-1.4$
4	$2x^6-8x^2+10x-13$	**14**	$2.5x^4+5x^2-3.2x-13.5$
5	$1.2x^4-4.5x^2+5.1x-20.3$	**15**	$x^6-5.5x^4+6x-2.4$
6	$1.2x^6-5x^2+5.3x-23$	**16**	$2x^4-6x^2-3.8x+1.5$
7	$3x^4+6.7x^2+3.1x-13$	**17**	$x^6-3.5x^4-2x^2+1.4$
8	$3x^6-3x^2+4x-3$	**18**	$x^4+7x^2+2x-3.5$
9	$2x^4+6x^2-15.2x-2.3$	**19**	$0.5x^6-1.5x^4+x^2-6.4$
10	$0.3x^6-5.8x^3+5x+3$	**20**	$2x^4-2x^2-3x-12.5$

використовуючи вбудовану функцію мінімізації функції однієї змінної. Знайдені корені перевірити за допомогою функції `roots`.

Реалізувати наступні вимоги :

a. розв'язок задачі подати у вигляді М-функції `tem623`.

b. сам вираз многочлена `p` вводимо як рядок з елементами `b***x**k`, де `b` – коефіцієнт (ціле | дійсне число); «`***x**`» – обов'язковий вираз; `k` – степінь (натуральне число). Наприклад,

```
p_str='  -412.99*x**4+153*x**0 +17.301*x**6+230.86*x**5+1*x**7    ';
```

c. для подальшої роботи з цим рядком необхідно видалить всі наявні пропуски;

d. для перетворення многочлена-рядка `p_str` в MATLAB-поліном `P` написати підфункцію `str_to_poly` з наступним заголовком :

```
function P = str_to_poly(p_str)
```

e. опис многочлена `p(x)`, як функцію від `x`, подати у вигляді `inline`-функції;

f. опис функції для мінімізації `Fp(x)` подати у вигляді анонімної функції, де її значення обчислюється з використанням функції `feval`(…) і функції `p(x)`;

g. для діалогового графічного пошуку області існування коренів полінома `P` і знаходження `D` – масива діапазонів з коренями, написати зовнішню функцію `search_region_root`.

Зауваження. Подібну функцію, яка виконувала діалоговий графічний пошук коренів, ми розглядали (див. Приклад 17 в Лекції №6). Це була підфункція в функції `pl6_4` з таким заголовком :

```
function x0 = search_region_root(f)
```

де вхідний параметр `f` – ім'я функції виду `f(x)`, а вихідний параметр `x0` – це масив початкових наближень для коренів.

Отже маємо ситуацію, що для `search_region_root` є різні варіанти вхідних і вихідних параметрів. Як наслідок, її функціонал необхідно написати так, щоб були враховані всі розглянуті можливості.

Цього можна досягти, якщо описати функцію `search_region_root` з наступним заголовком :

```
function varargout = search_region_root(P_f, varargin)
```

і з відповідною реалізацією тіла функції.

Розберем деякі деталі цього опису, зокрема зміст вхідних і вихідних параметрів і як це буде впливати на реалізацію тіла функції.

Вхідні параметри :

`P_f` – це ім'я MATLAB-полінома або ім'я функція виду `f(x)`.

`varargin{1}` – це параметр `t_in`, який визначає типізацію `P_f`.

Допустимі значення: 1 | 2, де

1 означає, що `P_f` є MATLAB-поліном (за угодою);

2 означає, що `P_f` є функція виду `f(x)`.

`varargin{2}` – це параметр `k_out`, що визначає вид результата `varargout`.

Допустимі значення: 1 | 2 | 3, де

1 означає, що список є `{D}` – масив діапазонів коренів (за угодою);

2 означає, що список є `{x0}` – масив початкових наближень до коренів;

3 означає, що список `varargout` містить обидва масиви `{D, x0}`.

При реалізації тіла функції потрібно подбати про можливі помилки користувача з означенням вхідних | вихідних параметрів та їх значень. Список результатів має формувитись в змінній `varargout` (масив типу `cell`). Старанно протестуйте функцію `search_region_root`!

Якщо ми все правильно реалізуємо, то виклик нашої функції буде мати наступний вид (враховуючи означення параметрів за угодою) :

```
D = search_region_root(P);
```

Практичне заняття № 07

ПОБУДОВА ГРАФІКІВ У MATLAB

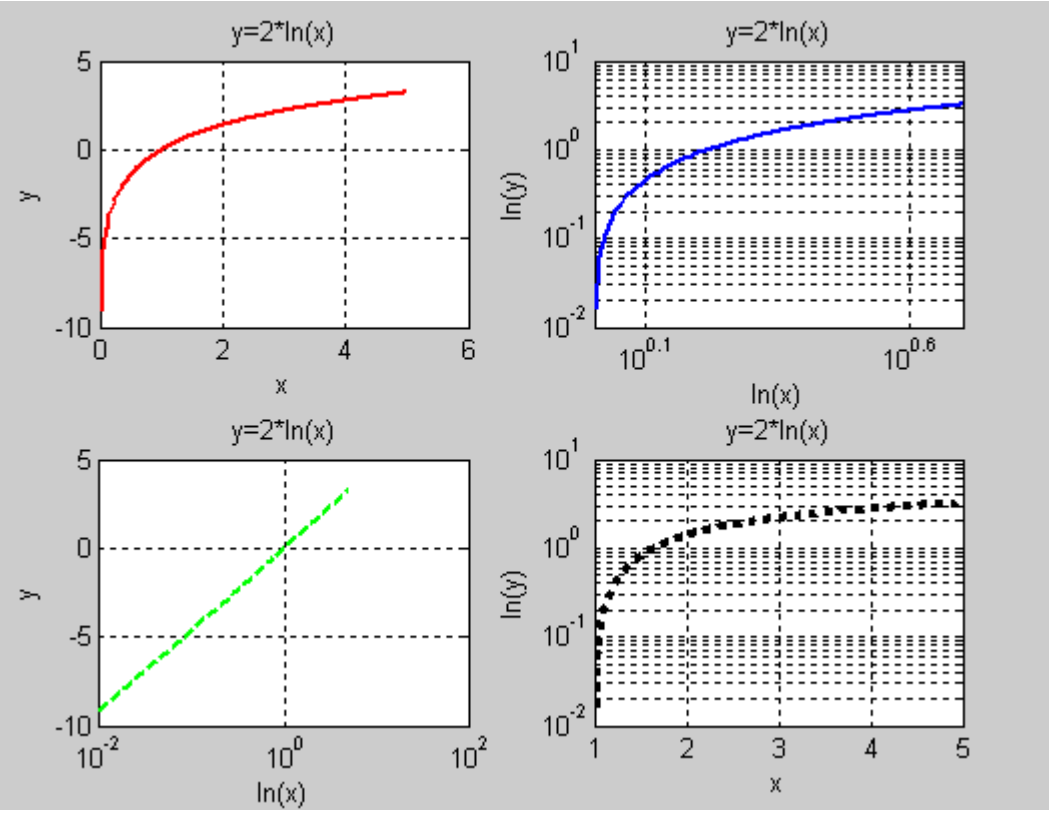
Аудиторне заняття

1. Приклади побудови деяких 2-D графіків.

Маємо наступну М-функцію *tem71a*

```
%% tem71a.m
% Побудова 2-D графіків
clear all, close all, clc, warning off

f = @(x)(2*log(x));
x = linspace(1e-2, 5, 201);
y = f(x);

s = 'y=2*ln(x)';
subplot(2,2,1);
plot(x,y,'r','LineWidth',2), grid on
title(s),
xlabel('x'), ylabel('y')

subplot(2,2,2);
loglog(x,y,'b','LineWidth',2), grid on
title(s),
xlabel('ln(x)'), ylabel('ln(y)')

subplot(2,2,3);
semilogx(x,y,'g--','LineWidth',2), grid on
title(s),
xlabel('ln(x)'), ylabel('y')

subplot(2,2,4);
semilogy(x,y,'k:','LineWidth',3), grid on
title(s),
xlabel('x'), ylabel('ln(y)')
pause(5), clear all, close all

a = 10; b = 5;
f1 = @(t)(a.*(1+cos(t))); % Кардіоїда
f2 = @(t)(b+a.*cos(t));   % Равлик Паскаля
fi = linspace(0, 2*pi, 201);

r = f1(fi);
polar(fi,r), grid on
title('Кардіоїда'),
pause(5)

r = f2(fi);
polar(fi,r), , grid on
title('Равлик Паскаля')
```

Виконання функції *tem71a* (тільки перший графік):

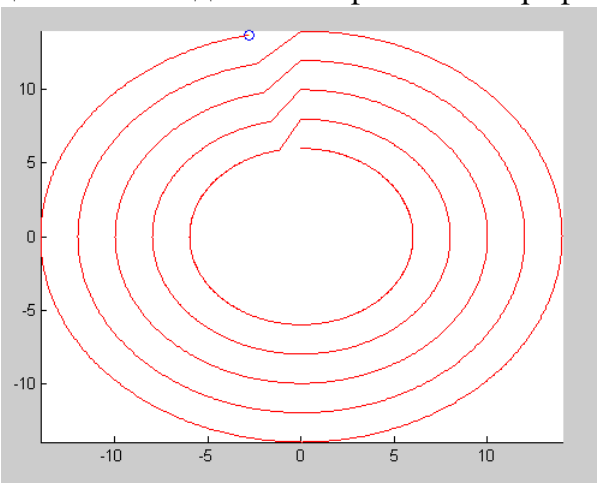


2. Побудувати анімований графік "спіраль електроплитки".

Маємо наступну М-функцію *tem72a*

```
function tem72a
% Побудова анімованого графіка "спіраль електроплитки"
    clear all, close all, clc
    R = [6 : 2 : 14];           % радіуси кругів
    d = 0.2;                    % "зсув" переходу між кругами
    t = [0 : 0.01 : 2.*pi-d];   % градусна сітка
    [n, r] = size(t);
    X = []; Y = [];             % (x,y)-координати спіралі
    for r = R
        % (x,y)-координати окремого круга спіралі
        x = r .* sin(t);
        y = r .* cos(t);
        X = [X,x]; Y = [Y,y];   % конкатенація кругів
    end
    comet(X,Y)
end
```

Виконання функції *tem72a* дає такий фінальний графік:



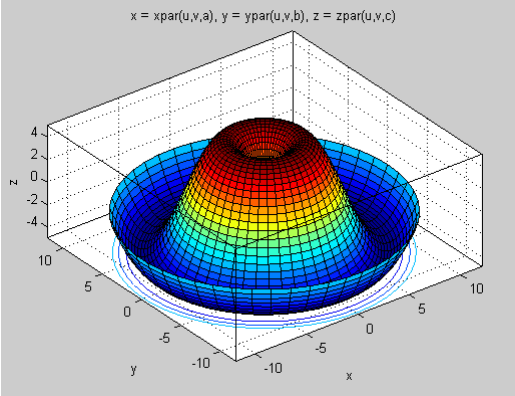
3. Побудувати 3-D графік – поверхню типу сомбреро. Рівняння поверхні задамо в параметричній формі і за "заготовку" нашої поверхні оберемо *еліптичний параболоїд*, який змінимо в частині завдання координати *z* (*z = c*sin(u)*), а точку (*x0,y0*) покладемо (0,0) і будемо будувати "кругле" сомбреро, тому покладемо *a = b*. Отримаємо графіки, використовуючи вбудовані MATLAB- функції *ezsurfc, ezsurf, ezmeshc, ezmesh*.

Маємо наступну М-функцію *tem73a*

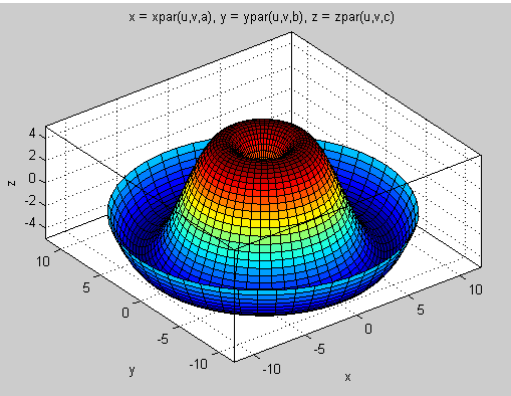
```
function tem73a
% Поверхня типу сомбреро
    clear all, close all, clc

    % завдання параметрів
    a = 2; b = 2; c = 5;
    % параметричне завдання координат поверхні :
    %    x = x(u,v), y = y(u,v), z = z(u,v)
    x = @(u,v)xpar(u,v,a);
    y = @(u,v)ypar(u,v,b);
    z = @(u,v)zpar(u,v,c);
    % завдання діапазонів зміни параметрів u,v :
    %    d = [A, B, C, D],  A <u< B, C <v< D
    d = [0, 6, 0, 2.*pi];

    disp('ezsurfc N=60')
    ezsurfc(x, y, z, d);  % сітка N*N, за угодою N = 60
    axis equal;           % пропорційність по осям однакова
    box on;               % рисунок розміщуємо в "бокс"
    pause(5); close all

    disp('ezsurf N=51')
    ezsurf(x, y, z, d, 51);
    axis equal; box on;
    pause(5); close all

    disp('ezmeshc N=101')
    ezmeshc(x, y, z, d, 101);
    axis equal; box on;
    pause(5); close all

    disp('ezmesh N=31 + що під куполом сомбреро?')
    ezmesh(x, y, z, d, 31);
    axis equal; box on;
    view(-25,-50)  % завдання кута огляду Az=-25, El=-50
    %pause(5); close all
end

function x = xpar(u, v, a)
    x = a.*cos(v).*u;
end

function y = ypar(u, v, b)
    y = b.*sin(v).*u;
end

function z = zpar(u, v, c)
    z = c.*sin(u);
end
```

Виконання функції *tem73a* дає такі графіки :

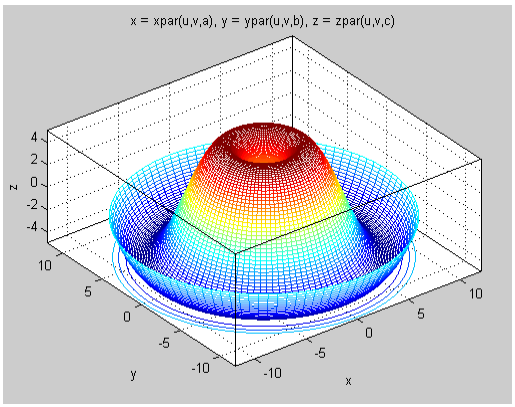
ezsurfc N=60 **ezsurf N=51**

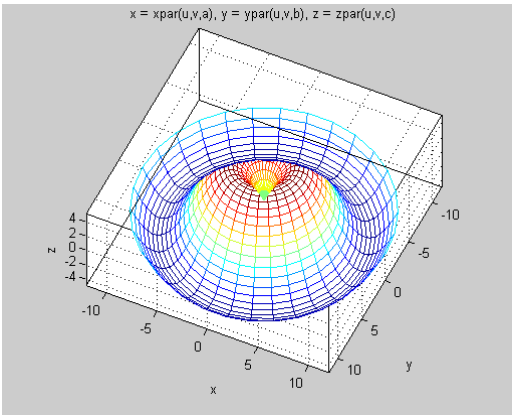


`ezmeshc N=101`



`ezmeshc N=31 + що під куполом сомбреро?`





Завдання для самостійної роботи

1. Побудувати анімований графік для наступних плоских кривих :

1.1 $\begin{cases} x = a \cdot \cos^3(t), \\ y = a \cdot \sin^3(t) \end{cases}$ – *астроїда*;

1.2 $\begin{cases} x = a \cdot t - b \cdot \sin(t), \\ y = a - b \cdot \cos(t) \end{cases}$ – *циклоїда*;

1.3 $\begin{cases} x = (a+c) \cdot \sin\left(\frac{a}{c}t\right) - b \cdot \sin\left(\frac{a-c}{c}t\right), \\ y = (a+c) \cdot \cos\left(\frac{a}{c}t\right) - b \cdot \cos\left(\frac{a-c}{c}t\right) \end{cases}$ – *епіциклоїда*;

1.4 $\begin{cases} x = (c-a) \cdot \sin\left(\frac{a}{c}t\right) - b \cdot \sin\left(\frac{c-a}{c}t\right), \\ y = (c-a) \cdot \cos\left(\frac{a}{c}t\right) + b \cdot \cos\left(\frac{c-a}{c}t\right) \end{cases}$ – *гіпоциклоїда*;

1.5 $\begin{cases} x = \frac{3a \cdot t}{t^3+1}, \\ y = \frac{3a \cdot t^2}{t^3+1}, \quad (-\infty < t < \infty) \end{cases}$ – *Декартів лист*;

1.6 $\begin{cases} x = a \cdot \left(\cos(t) + \ln\left(tg\left(\frac{t}{2} \right) \right) \right), \\ y = a \cdot \sin(t) \end{cases}$ – *трактриса*.

2. Для поверхонь другого порядку, заданих в параметричній формі :

2.1 $\begin{cases} x = x0 + a \cdot \sin(u) \cdot \cos(v), \\ y = y0 + b \cdot \sin(u) \cdot \sin(v), \\ z = z0 + c \cdot \cos(u) \end{cases}$ – *еліпсоїд* $(0 \le u < \pi,\ 0 \le v < 2\pi)$;

2.2 $\begin{cases} x = x0 + a \cdot ch(u) \cdot \cos(v), \\ y = y0 + b \cdot ch(u) \cdot \sin(v), \\ z = z0 + c \cdot sh(u) \end{cases}$ – *однополий гіперболоїд* $(-\infty < u < \infty,\ 0 \le v < 2\pi)$;

2.3 $\begin{cases} x = x0 \pm a \cdot ch(u), \\ y = y0 + b \cdot sh(u) \cdot \sin(v), \\ z = z0 + c \cdot sh(u) \cdot \cos(v) \end{cases}$ – *двополий гіперболоїд* $(-\infty < u < \infty,\ 0 \le v < 2\pi)$;

2.4 $\begin{cases} x = x0 + a \cdot u \cdot \cos(v), \\ y = y0 + b \cdot u \cdot \sin(v), \\ z = c \cdot u^2 \end{cases}$ – *еліптичний параболоїд* $(-\infty < u < \infty,\ 0 \le v < 2\pi)$

2.5 $\begin{cases} x = x0 + a \cdot (u + v), \\ y = y0 + b \cdot (v - u), \\ z = 4 \cdot u \cdot v \end{cases}$ – *гіперболічний параболоїд* $(-\infty < u < \infty,\ -\infty < v < \infty)$;

2.6 $\begin{cases} x = x0 + a \cdot u \cdot \cos(v), \\ y = y0 + b \cdot u \cdot \sin(v), \\ z = z0 + c \cdot u \end{cases}$ – *дійсний конус* $(-\infty < u < \infty,\ 0 \le v < 2\pi)$;

2.7 $\begin{cases} x = v, \\ y = y0 + a \cdot \cos(u), \\ z = z0 + b \cdot \sin(u) \end{cases}$ – *еліптичний циліндр* $(0 \le u < 2\pi,\ -\infty < v < \infty)$;

2.8 $\begin{cases} x = x0 + a \cdot sh(u), \\ y = v, \\ z = z0 + b \cdot ch(u) \end{cases}$ – *гіперболічний циліндр* $(-\infty < u < \infty,\ -\infty < v < \infty)$;

2.9 $\begin{cases} x = u^2 / 2p, \\ y = u, \\ z = a \cdot v \end{cases}$ – *параболічний циліндр* $(-\infty < u < \infty,\ -\infty < v < \infty)$.

побудувати графіки за допомогою функцій : `ezsurfc, ezsurf, ezmeshc, ezmesh`.

Зауваження:

1. Описати відповідні підфункції `xpar,ypar,zpar` означення параметричних поверхонь і використати їх в анонімних функціях `x,y,z` наступного виду:

```
x = @(u,v)xpar(u,v,a);
y = @(u,v)ypar(u,v,b);
z = @(u,v)zpar(u,v,c);
```

які в свою чергу будуть аргументами функцій побудови графіків.

2. Якщо функція опису поверхні має ще додаткові параметри, то для їх передачі використати глобальні змінні.

Практичне заняття № 08

АЛГОРИТМИ РОЗВ'ЯЗАННЯ ЗАДАЧІ КОШІ ДЛЯ ЗДР. ПОБУДОВА ГРАФІКІВ

Аудиторне заняття

Спочатку розглянемо деякі нюанси, пов'язані з викликом *ode*-солверів *ode23, ode45*, *ode15s*, *ode23s*. З матеріалів лекції ми знаємо, що загальний синтаксис виклику цих функцій є наступний (із вхідних параметрів тут тільки обов'язкові !) :

(*1) `[X, Y] = solver(odef, [x0, xend], Y0);`

(*2) `[X, Y] = solver(odef, [x0:h:xend], Y0);`

або

(**) `sol = solver(odef, […], Y0);`

Якщо виклик записано у формі (*1), то вектор *X* – це отримана (сформована) солвером *сітка розв'язку*, для якої виконується умова (`X(1)==x0` & `X(end)==xend`). Виклик, записаний у формі (*2), також сформує вектор *X*, але він повністю співпадає з вектором `[x0:h:xend]` – тобто, виконується умова (`X==[x0:h:xend]`). Вихідні параметри `[X, Y]` обов'язкові при формі виклику (*), тобто, як для форми (*1), так і для форми (*2). При цьому вихідний параметр **Y** – це вектор (одне рівняння) або матриця (якщо система *N* ЗДР). Рядки змінної **Y** містять дискретний розв'язок задачі Коші в точках сітки розв'язку `X`.

Якщо виклик записано у формі (**), то в структурі `sol` є поле `x` – це сітка розв'язку, отримана *ode*-солвером в процесі розв'язку. Для неї виконується умова (`X(1)==x0` & `X(end)==xend`), причому незалежно від `[…]` – форми опису діапазону, де шукаємо розв'язок. Тепер розберемося з `sol.y` – це поле, яке містить дискретний розв'язок задачі Коші в точках `sol.x`. Змінна `sol.y` буде вектор-рядком (для випадку, коли у нас одне рівняння) або матрицею (для випадку системи *N* ЗДР) і рядки змінної `sol.y` зберігають наближені розв'язки для невідомих функцій $u_i(x)$.

Порівнюючи спосіб зберігання розв'язку в **Y**, для виклику (*) і в `sol.y`, для виклику (**), бачимо, що вони різняться, і про це небхідно пам'ятати при роботі з *ode*-солверами. Наприклад, для задачі Коші для системи *N* ЗДР, маємо :

$$\mathbf{Y}(k, i) = u_i(X(k)),\ i = 1,2,\ldots,N;$$

$$sol.\mathbf{y}(i, k) = u_i(X(k)),\ i = 1,2,\ldots,N.$$

Розібрані ситуації демонструє наступна програма (для `N = 2`) :

```
function tem80a
    clear all, close all, clc
    a = 0; b = 3; n = 16;
    xv = linspace(a, b, n).';
    y0 = [0, 1];
    xd = [a, b];

    [x, y] = ode23(@fun, xd, y0);
    [Nx, mx]   = size(x);
    [Nxd, mxd] = size(xd);
    [Ny, my]   = size(y);
```

```
    e = all(Nx == Nxd);
    fprintf('Виклик (*1): Nx=%3u Nxd=%3u Ny=%3u my=%3u (Nx==Nxd)=%u\n',...
             Nx, Nxd, Ny, my, e);

    [x, y] = ode23(@fun, xv, y0);
    [Nx, mx]   = size(x);
    [Nxv, mxv] = size(xv);
    [Ny, my]   = size(y);
    e = all(Nx == Nxv);
    fprintf('Виклик (*2): Nx=%3u Nxv=%3u Ny=%3u my=%3u (Nx==Nxv)=%u\n',...
             Nx, Nxv, Ny, my, e);

    U = ode23(@fun, xd, y0);
    U

    U = ode23(@fun, xv, y0);
    U

    odeset()
end

function dydx = fun(x, y)
    dydx = [y(2);
            sin(y(1).^2)+x.*y(2)-3.*y(1)];
end
```

Результат її виконання :

```
Виклик (*1): Nx= 35 Nxd=  1 Ny= 35 my=  2 (Nx==Nxd)=0
Виклик (*2): Nx= 16 Nxv= 16 Ny= 16 my=  2 (Nx==Nxv)=1

U =
     solver: 'ode23'
    extdata: [1x1 struct]
          x: [1x35 double]
          y: [2x35 double]
      stats: [1x1 struct]
      idata: [1x1 struct]

U =
     solver: 'ode23'
    extdata: [1x1 struct]
          x: [1x35 double]
          y: [2x35 double]
      stats: [1x1 struct]
      idata: [1x1 struct]

          AbsTol: [ positive scalar or vector {1e-6} ]
          RelTol: [ positive scalar {1e-3} ]
     NormControl: [ on | {off} ]
     NonNegative: [ vector of integers ]
       OutputFcn: [ function_handle ]
       OutputSel: [ vector of integers ]
          Refine: [ positive integer ]
           Stats: [ on | {off} ]
     InitialStep: [ positive scalar ]
         MaxStep: [ positive scalar ]
             BDF: [ on | {off} ]
        MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
        Jacobian: [ matrix | function_handle ]
        JPattern: [ sparse matrix ]
      Vectorized: [ on | {off} ]
            Mass: [ matrix | function_handle ]
MStateDependence: [ none | {weak} | strong ]
       MvPattern: [ sparse matrix ]
    MassSingular: [ yes | no | {maybe} ]
    InitialSlope: [ vector ]
          Events: [ function_handle ]
```

В цій демо-програмі також був здійснений виклик (без вхідних і вихідних

параметрів) функції `odeset()`, яка (в такій формі виклику) інформує нас про імена і значення параметрів управління `ode`-солверами (в {} – значення за угодою!). Ця функція, але з параметрами, слугує і для зміни значень опцій. Враховуючи кількість цих опцій і їх призначення, бачимо, що для більшості практичних задач, значення за угодою цілком нас влаштовує, тому вхідний параметр `options`, при виклику `ode`-солвера, не є необов'язковий.

Причиною зміни параметрів `RelTol, AbsTol` може бути бажання користувача підвищити точність на кожному кроці розрахунків, яка визначається в `ode`-солверах по формулі :

`abs(e(i)) ≤ max(RelTol*abs(y(i)), AbsTol(i))`

В цьому випадку необхідно змінити значення опцій `RelTol,AbsTol`, наприклад :

```
options = odeset();   % отримання опцій
options = odeset('AbsTol', 1e-8, 'RelTol', 1e-6);   % зміна опцій
odeget(options, 'AbsTol')   % показати значення зміненої опції AbsTol
```

Тепер перейдемо до розв'язання деяких задач.

1. Нехай маємо для функції $u(t)$ задачу Коші :

$$\begin{cases} 4u'' + 5u' + 9u = \cos(u), \\ u(0) = 4,\ u'(0) = -3. \end{cases}$$

Щоб використати `ode`-солвери MATLAB спочатку цю задачу потрібно перетворити в стандартну систему звичайних диференціальних рівнянь першого порядку. Позначимо $u_1(t) = u(t),\ u_2(t) = u'(t)$, тоді маємо :

$$\begin{cases} u_1'(t) = u_2(t) & \equiv f_1(u_1, u_2), \\ u_2'(t) = 0.25\cos(u_1) - 1.25u_2 - 2.25u_1 & \equiv f_2(u_1, u_2); \\ u_1(0) = 4,\ u_2(0) = -3. \end{cases}$$

або в стандартній векторній формі :

$$\begin{cases} \dfrac{d\vec{\mathbf{U}}(t)}{dt} = \vec{\mathbf{f}}\left(t, \vec{\mathbf{U}}(t)\right), \\ \vec{\mathbf{U}}(0) = \overrightarrow{\mathbf{U0}}. \end{cases},$$

де $\vec{\mathbf{U}}(t) = (u_1, u_2)^{\mathrm{T}}$; $\vec{\mathbf{f}}\left(t, \vec{\mathbf{U}}(t)\right) = \left(f_1(u_1, u_2),\ f_2(u_1, u_2)\right)^{\mathrm{T}}$; $\overrightarrow{\mathbf{U0}} = (4, -3)^{\mathrm{T}}$.

Для отриманої задачі Коші знайдемо наближений числовий розв'язок і побудуємо наступні графіки (зобразимо в підграфіках 2×2):

1) розв'язок $u(t)$, а також зобразимо $u'(t)$;
2) фазову траєкторію розв'язку ($u_2(u_1)$);
3) поле напрямків;
4) фазову траєкторію + поле напрямків.

Згадаємо деякі терміни з курсу диференціальних рівнянь.

Фазову траєкторію, яка відповідає $\vec{\tilde{\mathbf{U}}}(t)$ – розв'язку задачі Коші, можна подати параметричним рівнянням в декартових кординатах (x, y) :

$$\begin{cases} x = \tilde{u}_1(t), \\ y = \tilde{u}_2(t) \end{cases}.$$

Очевидно, що програмна реалізація буде за допомогою `plot(…)`.

Поле напрямків для автономної системи ЗДР, будується з сукупності векторів $\{\vec{N}_k\}$, де $\vec{N}_k$ – вектор утворений дотичною в точці $M_k(x, y)$, яка належить деякій інтегральній кривій задачі Коші з допустимими початковими умовами. Таким чином, технічна побудова поля напрямків зводиться до:

1) завдання сукупності точок $\{M_k(x,y)\}$ на можливих інтегральних кривих даної задачі. Позначимо ці координати $(\{X\},\{Y\})$;

2) знаходження значень $F_1 = f_1(\{X\},\{Y\})$, $F_2 = f_2(\{X\},\{Y\})$;

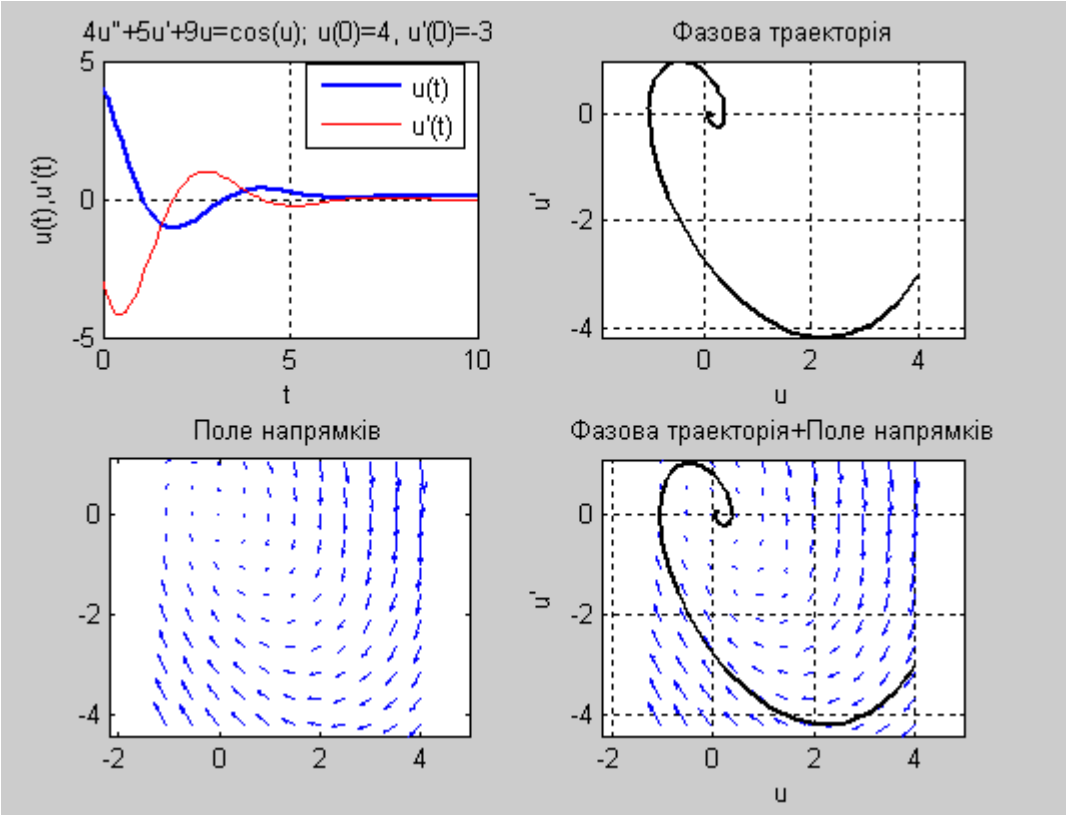
3) використання вбудованої функції MATLAB *quiver(…)* .

В тексті М-файлу *tem81a* максимальним чином збережені позначення, які ми використовували в наших вищеописаних викладках і міркуваннях.

```
%% tem81a.m
clear all, close all, clc, warning off

% Розв'язання задачі Коші
%   4u"+5u'+9u=cos(u);
%    u(0)=4,u'(0)=-3.

f1 = @(u1,u2)(u2);                  % опис функції f1(u1,u2)
f2 = @(u1,u2)(0.25.*cos(u1)-1.25.*u2-2.25.*u1); % f2(u1,u2)

f = @(t,U)[f1(U(1),U(2)); f2(U(1),U(2))];       % f(t,U)

U0 = [4,-3];                        % вектор початкових умов

% Розв'язання задачі Коші для системи ЗДР
[T, U] = ode45(f, [0 10], U0);
u = U(:,1); du = U(:,2);

clear U U0

% Побудова графіків u(t), u'(t)
subplot(2,2,1);
plot(T, u, 'b', 'LineWidth', 2), hold on
plot(T, du, 'r'), hold off,
grid on
s = 'u(t)';
sn = 'u''(t)';
legend(s, sn, 'Location', 'Best')
title('4u"+5u''+9u=cos(u); u(0)=4, u''(0)=-3')
xlabel('t'), ylabel([s,',',sn])

clear T

s = 'Фазова траєкторія';
subplot(2,2,2);
plot(u, du, 'k', 'LineWidth', 2)
axis equal
grid on
title(s)
xlabel('u'), ylabel('u''')

sn = 'Поле напрямків';
subplot(2,2,3);
m1 = min(u); M1 = max(u);
x = linspace(m1,M1,11);
m2 = min(du); M2 = max(du);
y = linspace(m2,M2,11);
[X, Y] = meshgrid(x, y);
F1 = f1(X, Y);
F2 = f2(X, Y);

clear x y m1 m2 M1 M2

quiver(X, Y, F1, F2)
```

```
axis equal
title(sn)

s = [s,'+',sn];
subplot(2,2,4);
quiver(X, Y, F1, F2)
axis equal
hold on
plot(u, du, 'k', 'LineWidth', 2)
axis equal
hold off
grid on
title(s)
xlabel('u'), ylabel('u''')
```

Результати виконання *tem81a*:



2. Розглянемо модельну задачу, на якій продемонструємо зміну точності для наближеного розв'язку і отримання розв'язку на заданній сітці виведення результатів.

$$\begin{cases} u'(t) = -(1+u^2)/(1+t^2), \ t \in (t0, T); \\ u(1) = 3. \end{cases}$$

$$t0 = 1, \ T = 11; \ T_{out} = \{T_k = t0 + (k-1)\Delta t, \ k = \overline{1, M}, \ T - t0 = (M-1)\Delta t\}, \ M = 11.$$

$$u(t) = (t+2)/(2t-1).$$

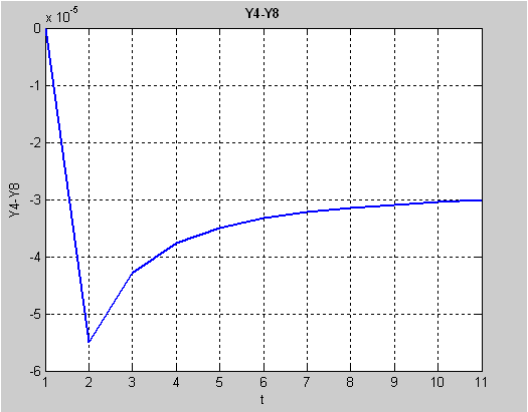
Напишемо наступну М-функцію *tem82a*:

```
function tem82a
    clear all, close all, clc, warning off
    U = @(t)((t+2)./(2.*t-1));
    t0 = 1; T = 11;
    M = 11; Tout = linspace(t0,T,M); % сітка виведення розв'язку
    Y0 = 3;  % початкові умови
```

```
    opt = odeset();
    opt = odeset('AbsTol', 1e-4, 'RelTol', 1e-4);
    A = odeget(opt, 'AbsTol');
    R = odeget(opt, 'RelTol');
    fprintf('AbsTol=%.3e RelTol=%.3e\n', A, R); clear A R
    Y4 = zeros(1,M);
    y0 = Y0; Y4(1) = y0;
    for k = 2 : M
        [t, Y] = ode23(@f, [Tout(k-1) Tout(k)], y0, opt);
        y0 = Y(end); % для наступного відрізка розв'язання
        Y4(k) = y0;  % збереження розв'язку в t=Tout(k)
    end
    opt = odeset('AbsTol', 1e-8, 'RelTol', 1e-8);
    A = odeget(opt, 'AbsTol');
    R = odeget(opt, 'RelTol');
    fprintf('AbsTol=%.3e RelTol=%.3e\n', A, R); clear A R
    Y8 = zeros(1,M);
    y0 = Y0; Y8(1) = y0;
    for k = 2 : M
        [t, Y] = ode23(@f, [Tout(k-1) Tout(k)], y0, opt);
        y0 = Y(end);
        Y8(k) = y0;
    end
    u = U(Tout);
    fprintf('  t      u(t)              Y4-u(t)            Y8-u(t)\n');
    s = '%5.2f %18.10e %18.10e %18.10e\n';
    for k = 1 : M
        fprintf(s,Tout(k),u(k),Y4(k)-u(k),Y8(k)-u(k));
    end
    plot(Tout,Y4-Y8,'b','LineWidth',2);
    grid on; xlabel('t'); ylabel('Y4-Y8');
    title('Y4-Y8','FontWeight','bold');
end

function F = f(t, U)
    % права частина системи
    a = U(1);
    F = -(1+a.*a)./(1+t.*t);
end
```

Результати виконання *tem82a*:

```
AbsTol=1.000e-004 RelTol=1.000e-004
AbsTol=1.000e-008 RelTol=1.000e-008
  t      u(t)              Y4-u(t)            Y8-u(t)
 1.00  3.0000000000e+000  0.0000000000e+000  0.0000000000e+000
 2.00  1.3333333333e+000 -5.4932017857e-005 -6.6583465408e-009
 3.00  1.0000000000e+000 -4.2965415763e-005 -6.4069640704e-009
 4.00  8.5714285714e-001 -3.7661220560e-005 -6.4590515159e-009
 5.00  7.7777777778e-001 -3.4935066101e-005 -6.6703739199e-009
 6.00  7.2727272727e-001 -3.3310360888e-005 -6.9312090512e-009
 7.00  6.9230769231e-001 -3.2241088074e-005 -7.1921586464e-009
 8.00  6.6666666667e-001 -3.1487303675e-005 -7.4391663940e-009
 9.00  6.4705882353e-001 -3.0928717791e-005 -7.6747094235e-009
10.00  6.3157894737e-001 -3.0498854031e-005 -7.8997220987e-009
11.00  6.1904761905e-001 -3.0158151017e-005 -8.1087617732e-009
```



3. Розглянемо ще один приклад, який буде корисним при виконанні самостійних робіт, це розв'язання "жорстких" задач для систем ЗДР.

Якщо обмежитись автономними системами, то для задачі Коші :

$$\begin{cases} \dfrac{d\vec{\mathbf{U}}(t)}{dt} = \vec{\mathbf{F}}\left(\vec{\mathbf{U}}(t)\right), \\ \vec{\mathbf{U}}(0) = \overrightarrow{\mathbf{U0}}. \end{cases},$$

де $\vec{\mathbf{U}}(t) = (u_1, u_2, \ldots, u_N)^{\mathrm{T}}$, $\vec{\mathbf{F}}\left(\vec{\mathbf{U}}(t)\right) = \left(f_1(\vec{\mathbf{U}}(t)), \ldots, f_N(\vec{\mathbf{U}}(t))\right)^{\mathrm{T}}$, ознакою "жорсткості" системи буде наявність великих по модулю власних чисел матриці Якобі

$$\mathbf{J} = \left(\frac{\partial F_i}{\partial U_j} \right)_{i,\, j=1}^{N}.$$

Особливо часто "жорсткі" системи виникають при моделюванні явищ, де розкид часових характеристик закладено в самій фізичній природі явища і пов'язано це з наявністю в розв'язках т.з. *примежового шару*. Такі задачі досить часто зустрічаються в електротехніці, хімічній кінетиці, дослідженні роботи ядерних реакторів, біології, економіці, екології, соціології.

Нехай маємо для функцій $u_1(t),\ u_2(t),\ u_3(t)$ задачу Коші :

$$\begin{cases} u_1'(t) = -4 \cdot 10^{-2} * u_1 + 10^4 * u_2 * u_3, \\ u_2'(t) = 10^{-2} * u_1 - 10^4 * u_2 * u_3 - 10^7 * u_2^2, \\ u_1'(t) = 10^7 * u_2^2, \\ \qquad\qquad t \in (0, T), \quad T = 100; \\ \vec{\mathbf{u}}(0) = (1, 0, 0)^{\mathrm{T}}. \end{cases}$$

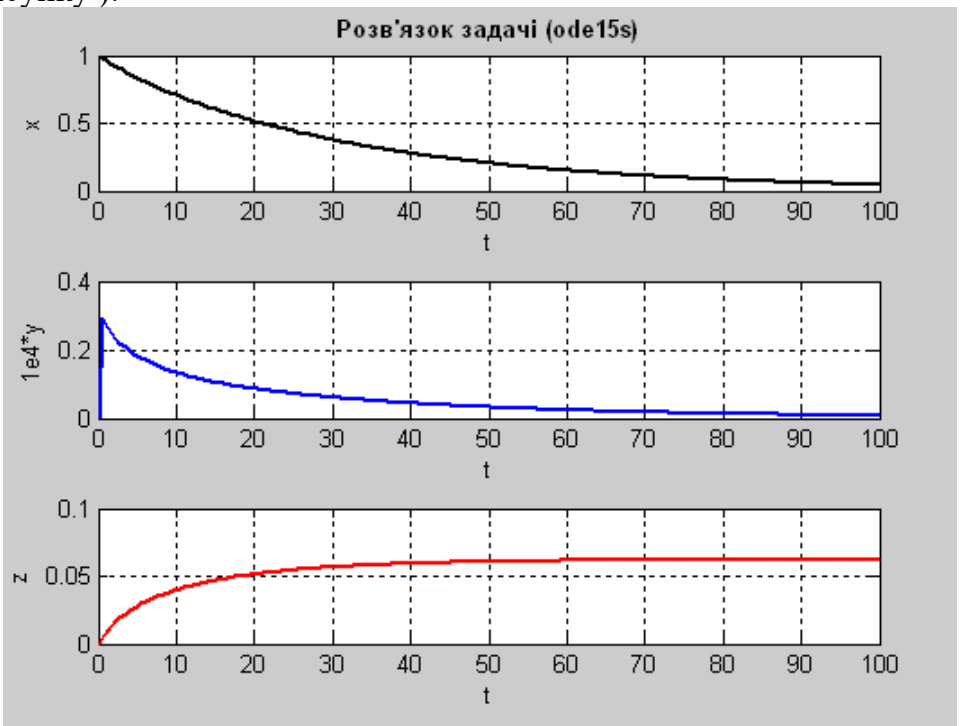
Це модель Робертсона опису хімічної реакції деяких речовин і фізично вектор-функція $\vec{\mathbf{u}}(t)$ описує концентрації речовин. Для нашого прикладу застосування *ode*-солверів MATLABa її розв'язання, числові параметри вибрані нами довільно.

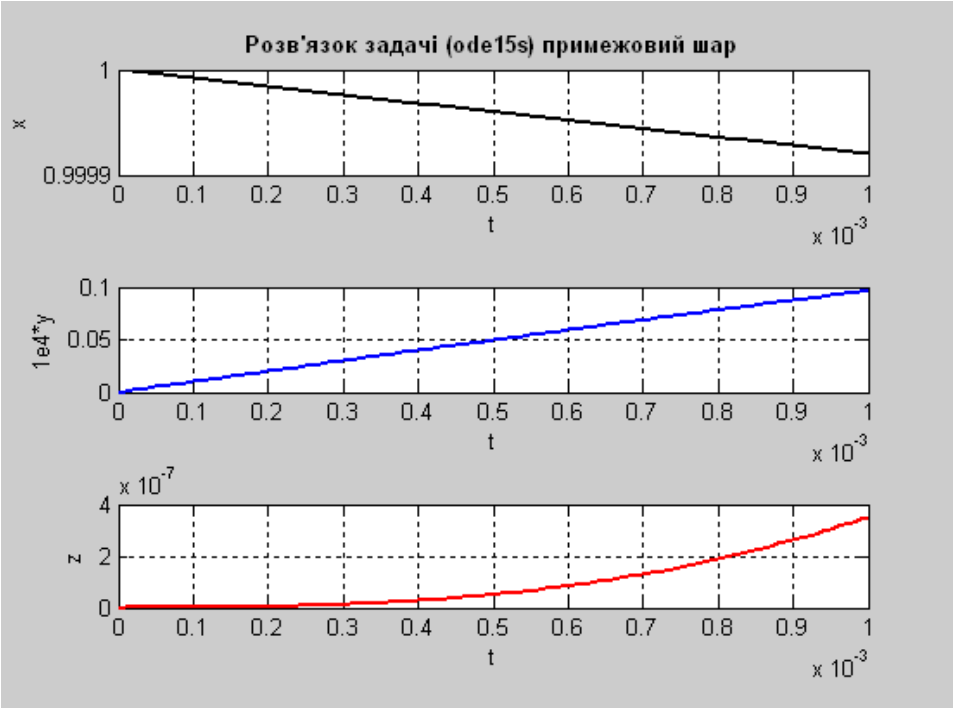
Текст М-функції *tem83a* :

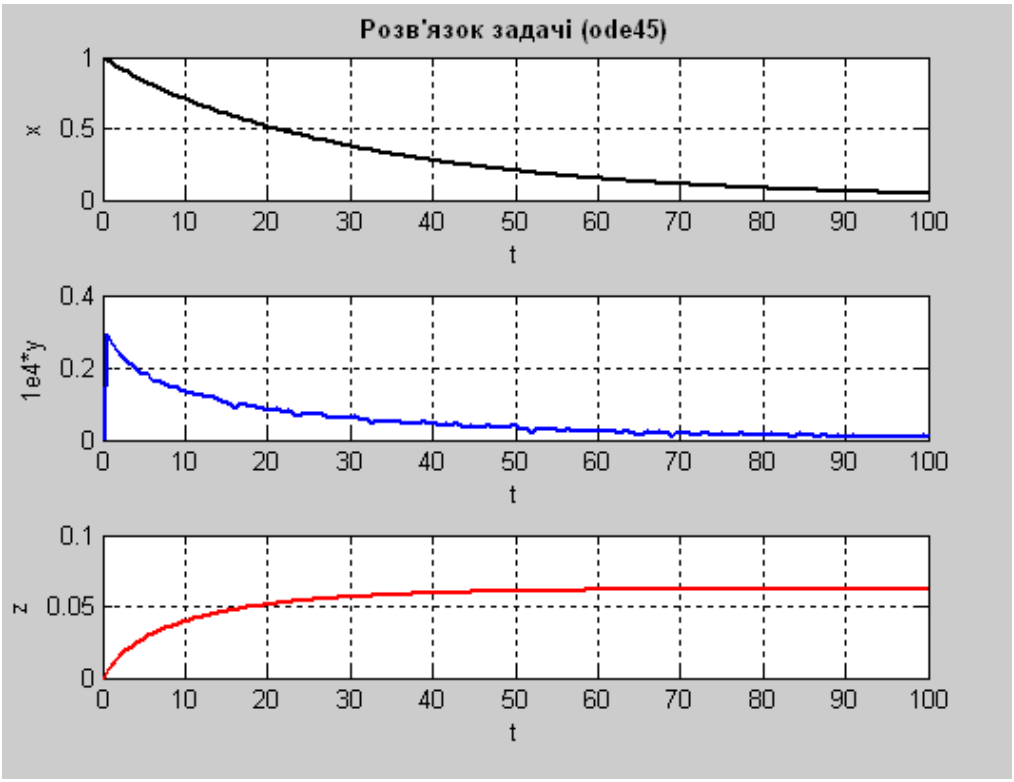
```
function tem83a
% Приклад розв'язання "жорсткої" СДУ
% Рівняння хімічної кінетики
%   (модель Робертсона)
    clear all, close all, clc, warning off

    Y0 = [1, 0, 0];  % вектор початкових умов
    [T, Y] = ode15s(@f, [0 : 0.5 : 100], Y0);
    s = 'Розв''язок задачі'; s1 = [s,' (ode15s)'];
    putsol(T, Y, s1, 5)

    [T, Y] = ode15s(@f, [0:1e-5:1e-3], Y0);
```

```
    putsol(T, Y, [s1,' примежовий шар'],15)
    [T, Y] = ode45(@f, [0 : 0.5 : 100], Y0);
    putsol(T, Y, [s,' (ode45)'],5)
end

function R = f(t, U)
    % права частина системи
    a = 1e-2.*U(1); c = 1e4.*U(2);
    b = c.*U(3); c = 1e3.*c.*U(2);
    R = [-4.*a + b; a - b - c; c];
end

function putsol(T, Y, txt, p)
    % Виведення графіків
    subplot(3,1,1)
    plot(T, Y(:,1), 'k', 'LineWidth', 2)
    grid on, xlabel('t'), ylabel('x')
    title(txt, 'FontWeight', 'bold')
    subplot(3,1,2)
    plot(T, 1e4.*Y(:,2), 'b', 'LineWidth', 2)
    grid on, xlabel('t'), ylabel('1e4*y')
    subplot(3,1,3)
    plot(T, Y(:,3), 'r', 'LineWidth', 2)
    grid on, xlabel('t'), ylabel('z')
    pause(p), close all
end
```

Результати виконання *tem83a* (зверніть увагу на швидкий ріст $u_2(t)$ на другому рисунку):







Порівнюючи графіки розв'язку для компоненти $u_2(t)$, отримані *ode15s* і *ode45*, бачимо відмінності.

Ще один відомий приклад "жорсткої" СДР – рівняння Ван-дер-Поля

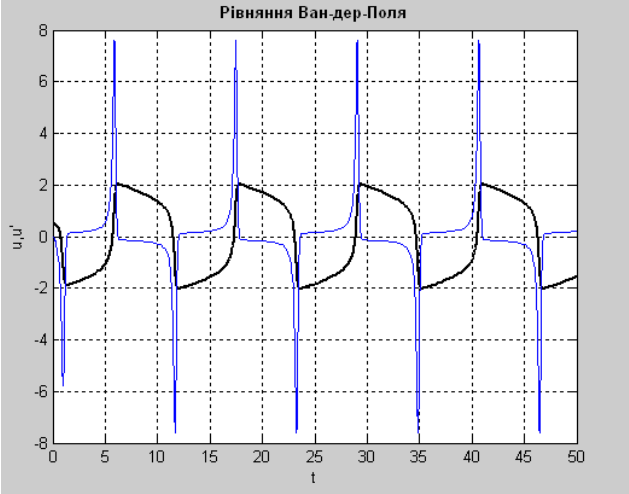
$$u''(t) + \mu * (u^2 - 1) * u' + u = 0, \quad t \in (0, \mathrm{T});$$

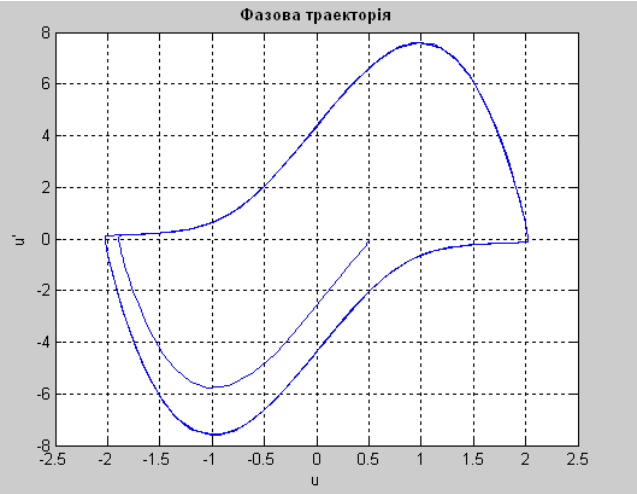
$$u(0) = u0, \quad u'(0) = u1.$$

Маємо М-функцію *tem84a* :

```
function tem84a
% Приклад розв'язання "жорсткої" СДР
%  (рівняння Ван-дер-Поля)
%  u"(t)+mu*(u^2-1)u'+u = 0,
```

```
%       t є (0,T];
%   u(0) = u0, u'(0) = 0.

    clear all, close all, clc, warning off

    global mu
    mu = 5; u0 = 0.5; T = 50;

    Y0 = [u0, 0];  % вектор початкових умов
    [T, Y] = ode15s(@f, [0 T], Y0);

    plot(T, Y(:,1), 'k', 'LineWidth', 2), hold on
    plot(T, Y(:,2), 'b'), grid on
    xlabel('t'), ylabel('u,u'''')
    s = 'Рівняння Ван-дер-Поля';
    title(s, 'FontWeight', 'bold')
    hold off, pause(5)

    plot(Y(:,1),Y(:,2)), grid on
    xlabel('u'), ylabel('u'''')
    s = 'Фазова траєкторія';
    title(s, 'FontWeight', 'bold')
end

function R = f(t, U)
    % права частина системи
    global mu
    e = U(1);
    R = [U(2);...
         mu.*(1-e.*e).*U(2)-e];
end
```





Завдання для самостійної роботи

1. Знайти розв'язок задачі Коші для моделі "хижак-жертва" Холлінга-Теннера :

$$\begin{cases} x'(t) = a * (1 - x/k1) * x - b * x * y/(k2 + x), \\ y'(t) = (c - d * y/x) * y. \end{cases}$$

де $0 < t \le 20$, а решта параметрів задана в таблиці

№	$x(0)$	$y(0)$	*a*	*b*	*c*	*d*	*k1*	*k2*
1	17	6	1.5	1.6	0.1	0.2	7	1
2	15	9	1.3	1.3	0.15	0.22	8	2
3	22	10	0.9	0.88	0.13	0.3	10	1

4	30	5	0.98	1.45	0.05	0.12	9.56	3
5	7	14	1.23	0.78	0.2	0.3	5	7
6	9	12	1	1	0.2	0.2	8	2
7	14	9	1.02	1.03	0.5	0.2	9	1
8	17	6	1.5	1.6	0.1	0.2	7	1
9	15	9	1.3	1.3	0.15	0.22	8	2
10	18	6	1.19	0.9	0.3	0.13	12	3
11	30	7	0.8	1.5	0.12	0.1	12.5	3.7
12	7	16	1.3	0.8	0.12	0.23	11	1.7
13	19	3	1.1	1.2	0.12	0.21	12	2
14	16	2	1.2	1.3	0.3	0.2	19	1
15	23	4.5	0.9	1.2	0.1	0.1	9.1	3.1
16	7.6	19.4	1.23	0.78	0.2	0.3	5	6
17	9.5	12	1	1	0.22	0.2	8	2.1
18	13	9	1.02	1.03	0.32	0.1	9	1
19	17.4	6	1.5	1.6	0.14	0.02	7	1.1
20	19.5	7	1.1	1.43	0.5	0.32	12.8	1.2

Побудувати наступні графіки :

1) розв'язок $x(t), y(t)$**;**
2) фазову траєкторію розв'язку $y(x)$;
3) поле напрямків;
4) фазову траєкторію + поле напрямків,

причому для двох останніх – зобразити в в підграфіках 1×2.

2. Написати М-файл розв'язання мішаної задачі для одновимірного квазілінійного параболічного рівняння :

$$\frac{\partial u(x,t)}{\partial t} = K^2 \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t,u),\ x \in (0,1),\ t \in (0,T];$$

$$u(0,t) = \mu_1(t),\quad u(1,t) = \mu_2(t),\ t \in [0,T],\ T = 5;$$

$$u(x,0) = u0(x),\ x \in [0,1].$$

Параметри задачі задані в таблиці

№	***K***	$u0(x)$	$\mu_1(t)$	$\mu_2(t)$	$f(x,t,u)$
1	0,5	$x(1-x)$	$\sin(t)$	$2t$	$\sin(2t-x)/(1+u^2)$
2	0.6	$x^2(1-x)$	t	$\sin(\sqrt{t})$	$(2t-ux)/(1+u^2)$
3	2	$x^3(1-x)$	$1-\cos(t)$	t	$2tx/(4+u^2)$
4	1.5	$\sqrt{x}(1-x)$	$2t$	$1-\exp(-t)$	$(ut-x)/(1+4u^2)$
5	5	$x(1-x^2)$	t^2	$\sin(t)$	$\cos(tx)/(1+u^2)$
6	3.4	$x(1-x^3)$	$\sqrt{t}$	t	$\sin(xt+u^2)$
7	4	$\sqrt{x}(1-x^2)$	$\sqrt{t^3}$	$1-\cos(t)$	$(2u-tx)/(3+u^2)$
8	1.7	$\sin(x(1-x))$	$1-\exp(-t)$	$2t$	$(u+t-x)/(1+14u^2)$
9	2.1	$\sin(x^2(1-x))$	$2.5t$	t^2	$(xt+u^2)^{0.5}$
10	2.5	$\sin(x^3(1-x))$	$0.5t^2$	$\sqrt{t}$	$\cos(tx/(2+u^2))$
11	3.1	$\sqrt{x}(1-x)\sin(x)$	$\sin(t^2)$	$t^{3/2}$	$2tx(4+u^2)^{-0.5}$

12	3.2	$x(1-x^2)\cos(x)$	t	$1-\exp(-2t)$	$\cos(xt+u^2)$
13	1.76	$\sin(x)(1-x^3)$	$\sin(\sqrt{t})$	$2.5t$	$\exp(xt-u^2)$
14	2.3	$\sqrt{x}(1-x^2)\cos(x^2)$	$2t$	$0.5t^2$	$xt+2u$
15	1.6	$x(1-x)\sin(x)$	$t^{3/2}$	$\sin(t^2)$	$\sqrt[3]{xt+u^2}$
16	1.7	$x(1-x)\cos(x^2)$	$1-\exp(-2t)$	$t^{5/2}$	$\sin(x-t+u^2)$
17	1.9	$x(1-x)(1-x^3)$	$t^{5/2}$	$\sin(t)$	x^3t+2u
18	2,1	$x(1-x)sh(x)$	$\sin(t)$	$1-\cos(t)$	$\exp(-xt/(1+2u^2))$
19	2	$x(1-x)\exp(x)$	$0.5t^2$	$2t$	$2tx/(4+u^4)$
20	3,4	$x(1-x)\exp(-x)$	$1-\cos(t)$	t^2	$\cos(x-t^2u)$

Побудувати наступні графіки :

1) розв'язок $u(x,t)$ за допомогою `mesh`;
2) розв'язок $u(x,t)$ за допомогою `surf`;
3) розв'язок у вигляді ліній постійного рівня.

Вказівки.

1. Для наближеного розв'язання задачі використаємо метод *прямих* (*Pome*). Для цього побудуємо тільки просторову сітку $\overline{\omega}_h = \{x_k = (k-1)h,\ k = 1,2,...,n,\ 1 = (n-1)h\}$, а на прямих $x = x_k,\ k = 2,3,...,n-1$ введемо невідомі функції $V_{k-1}(t) = u(x_k,t)$. Диференціальний оператор по змінній x в рівнянні замінимо на різницевий (з функціями $V_{k-1}(t)$) і порядком апроксимації $O(h^2)$, врахуємо крайові умови ($u(x_1,t) = \mu_1(t),\ u(x_n,t) = \mu_2(t)$ - відомі!), а також позначимо вектор-функцію (яку маємо визначити!) :

$$\vec{V}(t) = [V_1(t) = u(x_2,t),\ V_2(t) = u(x_3,t),\ ...,\ V_{n-3}(t) = u(x_{n-2},t),\ V_{n-2}(t) = u(x_{n-1},t)].$$

В результаті цього, для $\vec{V}(t)$ отримаємо наступну задачу Коші :

$$\begin{cases} \vec{V}'(t) = \vec{F}(t,\vec{V}(t)),\ 0 < t \le T; \\ \vec{V}(0) = u0(\vec{X}),\ \ \vec{X} = [x_2, x_3, ..., x_{n-2}, x_{n-1}], \end{cases}$$

де

$$\vec{F}(t,\vec{V}(t)) = \begin{cases} A*(\mu_1(t) - 2V_1(t) + V_2(t)) + f(x_{k+1},t,V_k(t)), & k = 1; \\ A*(V_{k-1}(t) - 2V_k(t) + V_{k+1}(t)) + f(x_{k+1},t,V_k(t)), & k = \overline{2,n-3}; \\ A*(V_{k-1}(t) - 2V_k(t) + \mu_2(t)) + f(x_{k+1},t,V_k(t)), & k = n-2; \end{cases} \quad A = (K/h)^2.$$

Шукана вектор-функція розв'язку поставленої задачі буде визначатись так

$$\vec{U}(t) = [\mu_1(t),\ \vec{V}(t),\ \mu_2(t)].$$

2. Враховуючи, що "жорсткість" побудованої системи ЗДР зростає зі збільшенням n (даже при $f(x,t,u) \equiv 0$!), тому :

- кількість вузлів сітки по x обирайте $n \approx 11 \div 21$ і можна попробувати використати солвер `ode45`, але це «мала» точність по кроку h;
- для покращення точності наближеного розв'язку (по кроку h) кількість вузлів сітки по x збільшіть $\approx 4 \div 5$ рази, використайте солвер `ode15s` і порівняйте результати.

Практичне заняття № 09

ВИКОРИСТАННЯ СОЛВЕРІВ MATLAB ДЛЯ РОЗВ'ЯЗАННЯ КРАЙОВИХ ЗАДАЧ ДЛЯ ЗДР

Аудиторне заняття

1. Для рівняння (введені позначення $p1 = 0.5\pi,\ \ p2 = p1^2$) :

$$u''(x) + u'(x) + p2 * u(x) = p1 * \cos(p1 * x),\ \ x \in (0,4); \tag{9.1}$$

задано періодичні крайові умови :

$$u(0) = u(4),\ \ u'(0) = u'(4). \tag{9.2}$$

Знайти розв'язок цієї крайової задачі і побудувати графіки.

Якщо ввести позначення $U_1(x) = u(x), U_2(x) = u'(x)$, то задача (9.1)-(9.2) буде приведена до вигляду :

$$\begin{cases} U_1'(x) = U_2(x), \\ U_2'(x) = -U_2(x) - p2 * U_1(x) + p1 * \cos(p1 * x), \quad x \in (0,4). \end{cases} \tag{9.1*}$$

$$U_1(0) = U_1(4),\ \ U_2(0) = U_2(4). \tag{9.2*}$$

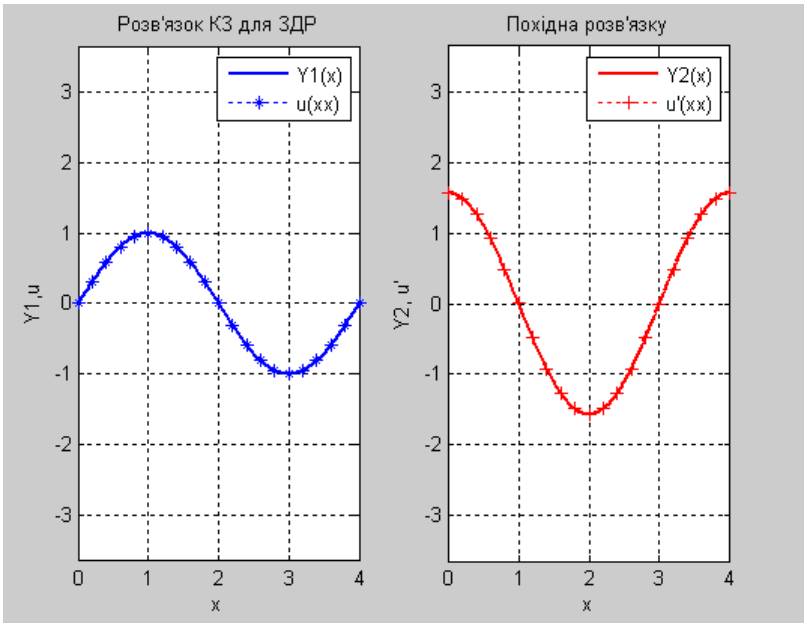
Отже можна застосувати солвер *bvp4c*, залишається тільки написати відповідну програму. Слідуючи стандартній схемі використання цього солвера і створивши *F(x,y)* – функцію для обчислення правої частини системи (9.1*), а також *G(ya,yb)* - функцію з описом крайових умов (9.2*), маємо таку М-функцію :

```
function pr09_1a
%% Розв'язання крайової задачі :
% u"(x)+u'(x)+p2*u(x) = p1*cos(p1*x),
%            x є (0,4);
%  u(0) = u(4), u'(0) = u'(4).
%   (p1 = 0.5*pi, p2 = p1^2)
% за допомогою вбудованої функції bvp4c.
% Модельна задача: u(x) = sin(p1*x), період=4.
clear all, close all, clc

global p1 p2
p1 = 0.5*pi; p2 = p1.*p1;

%Створюєм структуру sol_init початкових даних із m точок,
%рівномірно розподілених на відрізку [0, 4]
%  зі значеннями y = [0; 0]:
a = 0; b = 4; m = 101;
xx = linspace(a, b, m); yy = [0; 0];
sol_init = bvpinit(xx, yy);

% Розв'язуємо задачу :
sol = bvp4c(@F, @G, sol_init);
% Виведення результатів :
N = 401; X = linspace(a, b, N);
Y = deval(sol, X);
% Тест :  u(x) = sin(2*pi*x),
%        u'(x) = 2*pi*cos(2*pi*x)
u  = @(x)(sin(p1.*x));
du = @(x)(p1.*cos(p1.*x));
xx = linspace(a, b, 21);
U = u(xx); dU = du(xx);
subplot(1,2,1)
```

```
plot(X, Y(1,:), 'b -', 'LineWidth', 2), hold on
plot(xx, U, 'b*:'), hold on
grid on, axis equal
title('Розв''язок КЗ для ЗДР')
xlabel('x'); ylabel('Y1,u')
legend('Y1(x)', 'u(xx)')
subplot(1,2,2)
plot(X, Y(2,:), 'r -', 'LineWidth', 2), hold on
plot(xx, dU, 'r+:'), hold off
grid on, axis equal
title('Похідна розв''язку')
xlabel('x'); ylabel('Y2, u'''')
legend('Y2(x)', 'u''(xx)')
clear xx yy X Y U dU
U = norm(sol.y(1,:) -  u(sol.x),inf);
disp(strcat('||Y1-u ||c =',num2str(U)))
U = norm(sol.y(2,:) - du(sol.x),inf);
disp(strcat('||Y2-u''||c =',num2str(U)))

function dydf = F(x, y)
global p1 p2
dydf = [y(2);...
        p1.*cos(p1.*x)-p2.*y(1)-y(2)];

function bc = G(ya, yb)
bc = [ya(1) - yb(1);
      ya(2) - yb(2)];
```

Результати виконання *pr09_1a* :

```
||Y1-u ||c =4.9499e-007
||Y2-u'||c =6.7639e-007
```



2. Знайти розв’язок крайової задачі :

$$u'(x)+u(x)=1-|x-1|-sign(x-1),\quad x\in(0,2);$$
$$u(0)=u(2). \tag{9.3}$$

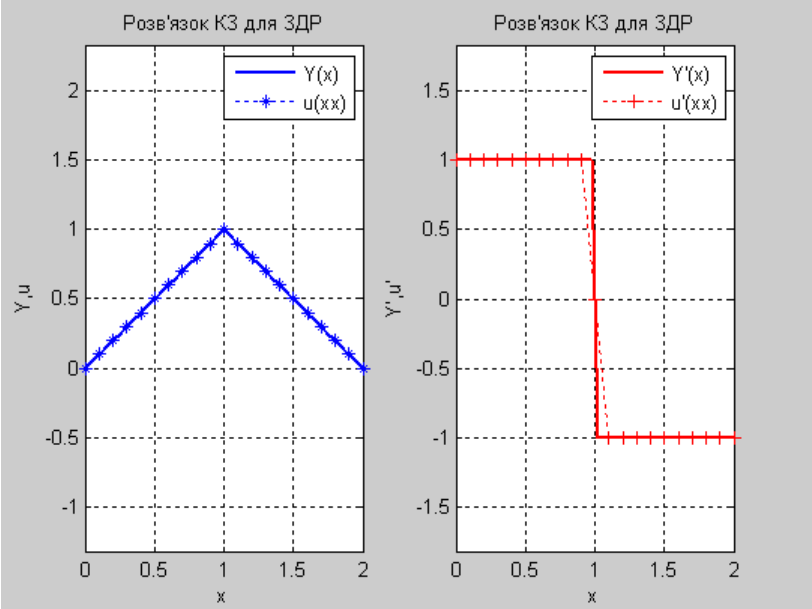
і побудувати графіки.

Функцію *pr09_1c* написати самостійно, використовуючи *pr09_1a*.

Результати виконання *pr09_1c* мають бути такі :

```
sol =
```

```
          x: [0 0.49998 0.99995 0.99998 0.99999 0.99999 1 1 1 1.2 1.4 1.7 2]
          y: [1x13 double]
         yp: [1 1 1 1 1 1 1.0173e-006 -1 -1 -1 -1 -1 -1]
     solver: 'bvp4c'

||Y1-u ||c =1.0173e-006
||Y2-u'||c =1.0173e-006
```



3. Знайти розв’язок крайової задачі :

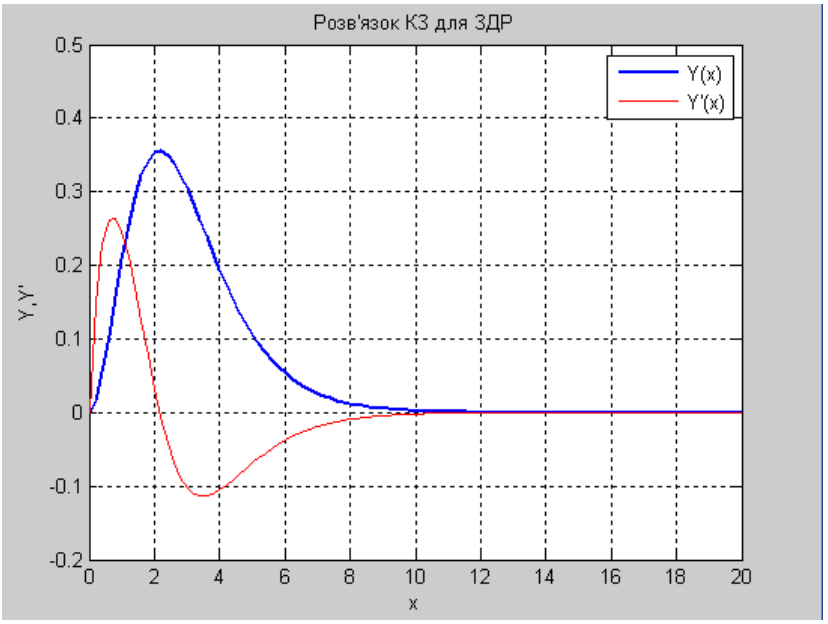
$$u'(x)+u(x)=x*\exp(u-x),\quad x\in(0,20);$$
$$u(0)=u(20). \tag{9.4}$$

і побудувати графіки наближеного розв’язку $u(x)$, $u'(x)$

Функцію *pr09_1d* написати самостійно, використовуючи *pr09_1c*.

Результати виконання *pr09_1d* мають бути такі :

```
sol =
          x: [1x40 double]
          y: [1x40 double]
         yp: [1x40 double]
     solver: 'bvp4c'
```



4. Нехай маємо для функцій $u(x), v(x), w(x)$ систему ЗДР (тут введено позначення $z = 1 + u^2(x) + v^2(x) + w^2(x)$) :

$$\begin{cases} u'(x) = \alpha * \exp(1 - u^2) * v * w/z, \\ v'(x) = \beta * \sin(p*(u + v))/z, \\ w'(x) = \gamma * \sin(p*(u - w))/z, \quad x \in (0,1) \end{cases} \tag{9.5}$$

і наступні крайові умови :

$$\begin{cases} u(0) = 1, \quad v(0) = r * w(0), \\ w(1) = r * v(1). \end{cases} \tag{9.6}$$

Для заданих параметрів $\alpha = 10, \beta = -5, \gamma = -6, p = 4.296, r = 0.96570055$, за допомогою солвера *bvp4c*, знайти розв'язок цієї крайової задачі і побудувати графіки.

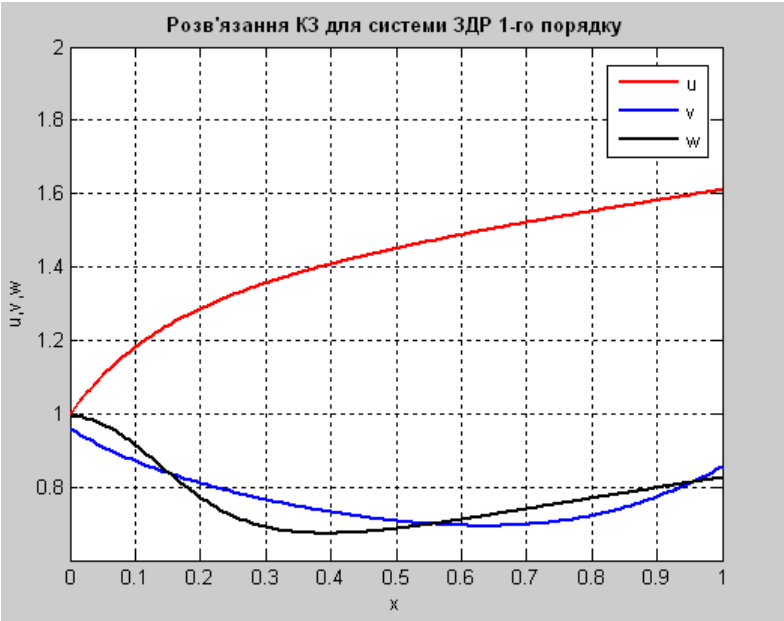
Якщо перепозначити $u(x), v(x), w(x) \to (U_i(x))_{i=1}^3$, то задача (9.5)-(9.6) буде мати вигляд, до якої можна застосувати солвер *bvp4c*, залишається тільки написати відповідну програму. Слідуючи стандартній схемі використання цього солвера і створивши *F(x,y)* – функцію для обчислення правої частини системи (9.5), а також *G(ya,yb)* - функцію з описом крайових умов (9.6), маємо таку М-функцію :

```
function pr09_2
%% Розв'язання крайової задачі :
% u'(x) = al*exp(1-u^2)*v*w/z,
% v'(x) = be*sin(p*(u+v))/z,
% w'(x) = ga*sin(p*(u-w))/z,
%       z = 1+u^2+v^2+w^2,
%         x є (0,1);
% u(0) = 1, v(0) = r*w(0),
% w(1) = r*v(1).
% за допомогою функції bvp4c.
clear all, close all, clc

global al be ga p r
al = 10; be = -5; ga = -6;
p = 4.296; r = 0.96570055;% параметри КЗ
a = 0; b = 1;  % граничні точки області
%Створюємо структуру sol_init початкових даних із m
% точок, рівномірно розподілених на відрізку [a,b]
% зі значеннями y = [1;0;0] :
m = 51; xx = linspace(a, b, m);
yy = [1; 0; 0];
sol_init = bvpinit(xx, yy);

% Розв'язуємо задачу :
sol = bvp4c(@F, @G, sol_init);
sol   % інформація про структуру sol

% Виведення результатів :
N = 201; X = linspace(a, b, N); % X - сітка виведення
Y = deval(sol, X); % обчислення отриманого розв'язку на X
plot(X, Y(1,:),'r', 'LineWidth', 2), hold on
plot(X, Y(2,:),'b', 'LineWidth', 2), hold on
plot(X, Y(3,:),'k', 'LineWidth', 2), hold off
grid on %, axis equal
s = 'Розв''язання КЗ для системи ЗДР 1-го порядку';
title(s, 'FontWeight', 'bold')
xlabel('x'); ylabel('u,v,w')
legend('u', 'v', 'w', 'Location', 'Best')

function dydf = F(x, y)
```

```
% опис правої частини системи
global al be ga p
z = 1./(1 + y(1).^2 + y(2).^2 + y(3).^2);
dydf = [al.*exp(1-y(1).^2).*y(2).*y(3).*z;...
        be.*sin(p.*(y(1)+y(2))).*z;...
        ga.*sin(p.*(y(1)-y(3))).*z];

function bc = G(ya, yb)
% опис крайових умов
global r
bc = [ya(1) - 1; ya(2) - r.*ya(3);...
      r.*yb(2) - yb(3)];
```

Результати виконання *pr09_2*:

```
sol =
         x: [1x989 double]
         y: [3x989 double]
        yp: [3x989 double]
    solver: 'bvp4c'
```



5. Нехай маємо для функцій $u(x), v(x)$ систему ЗДР :

$$\begin{cases} u''(x) = \sin(x * v' - 2u), \\ v''(x) = \cos(x*u' - u + v), \quad x \in (0, 4); \end{cases} \tag{9.7}$$

і наступні крайові умови :

$$u(0) = 1,\ v(0) = 0,\ u(4) = v(4),\ u'(4) = -v'(4). \tag{9.8}$$

За допомогою солвера *bvp4c* знайти розв'язок цієї крайової задачі і побудувати графіки $u(x), v(x)$, $u'(x), v'(x)$.

Задачу (9.6)-(9.7) спочатку необхідно привести до стандартного виду системи ДР першого порядку, розв'язаних відносно похідної. Введемо наступні позначення :

$$U_1(x) = u(x),\ U_2(x) = u'(x),\ U_3(x) = v(x),\ U_4(x) = v'(x),$$

і запишемо (9.6)-(9.7) наступним чином :

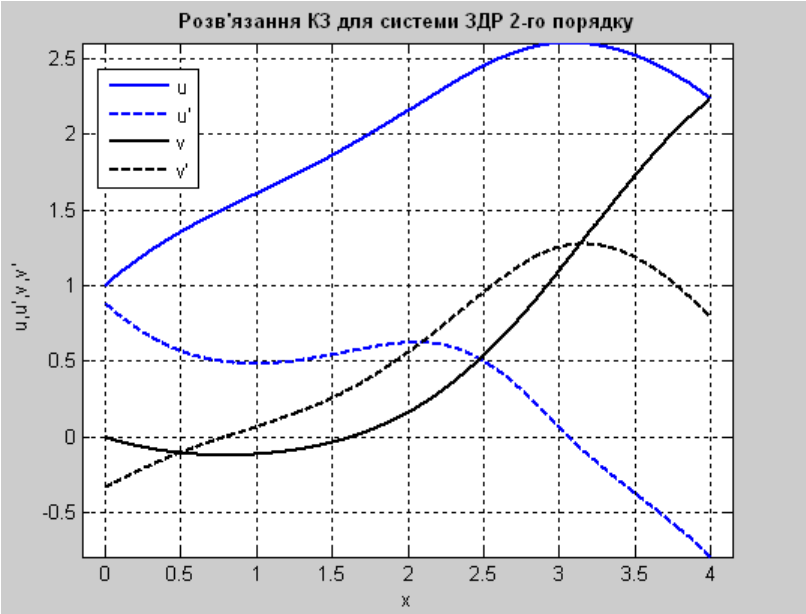
$$\begin{cases} U_1'(x) = U_2(x), \\ U_2'(x) = \sin(x * U_4(x) - 2U_1(x)), \\ U_3'(x) = U_4(x), \\ U_4'(x) = \cos(x*U_2(x) - U_1(x) + U_3(x)), \quad x \in (0, 4); \end{cases} \tag{9.7*}$$

$$U_1(0)-1=0,\ U_3(0)=0,\ U_1(4)-U_3(4)=0,\ U_2(4)+U_4(4)=0. \qquad (9.8^*)$$

Крайова задача (9.7*)-(9.8*) тепер має потрібний вигляд, а отже можна застосувати солвер *bvp4c* для її розв'язання. Напишемо відповідну програму, слідуючи стандартній схемі використання цього солвера. Для обчислення правої частини системи (9.7*), а також для опису крайових умов (9.8*), залишимо імена *F(x,y)*, *G(ya,yb)* – адже ці функції є підфункціями М-функції *pr09_3*:

```
function pr09_3
%% Розв'язання крайової задачі :
% u"(x) = sin(x*v'(x)-2*u(x)),
% v"(x) = cos(x*u'(x)-u(x)+v(x)),
%         x є (0,4);
% u(0) = 1,      v(0) = 0,
% u(4) = v(4), u'(4) = -v'(4).
% за допомогою функції bvp4c.

clear all, close all, clc

a = 0; b = 4;  % граничні точки області

%Створюємо структуру sol_init початкових даних із m
% точок, рівномірно розподілених на відрізку [a,b]
% зі значеннями y = [1;0;0;0] :
m = 201; xx = linspace(a, b, m);
yy = [1; 0; 0; 0];
sol_init = bvpinit(xx, yy);

% Розв'язуємо задачу :
sol = bvp4c(@F, @G, sol_init);

% Виведення результатів :
N = 801; X = linspace(a, b, N); % X - сітка виведення
Y = deval(sol, X); % обчислення отриманого розв'язку на X

plot(X, Y(1,:),'b -',  'LineWidth', 2), hold on
plot(X, Y(2,:),'b --', 'LineWidth', 2), hold on
plot(X, Y(3,:),'k -',  'LineWidth', 2), hold on
plot(X, Y(4,:),'k --', 'LineWidth', 2), hold off
grid on, axis equal
s = 'Розв''язання КЗ для системи ЗДР 2-го порядку';
title(s, 'FontWeight', 'bold')
xlabel('x'); ylabel('u,u'',v,v''')
legend('u', 'u'''', 'v', 'v'''', 'Location', 'Best')


function dydf = F(x, y)
% опис правої частини системи
dydf = [y(2);...
        sin(x.*y(4) - 2.*y(1));...
        y(4);...
        cos(x.*y(2) - y(1) + y(3))];

function bc = G(ya, yb)
% опис крайових умов
bc = [ya(1) - 1;...
      ya(3);...
      yb(1) - yb(3);...
      yb(2) + yb(4)];
```

Результати виконання *pr09_3*:



Після розв'язання задач (9.1)-(9.2) – (9.7)-(9.8), а також використовуючи теоретичні відомості про солвер *bvp4c* і розглянуті **Приклади 20-22** в Лекції № 9, ви легко справитесь зі своїми індивідуальними завданнями.

Завдання для самостійної роботи

Написати М-файл розв'язання крайової задачі з використанням солвера *bvp4c*:

1. $u''-2u(x)=2x\sin(u-5);\quad u'(0)+2u(0)=2,\quad u(2)=-1.$
2. $u''+xu'-u(x)=2x-\cos(u-1);\quad u'(1)=2,\quad u(2)-2u'(2)=1.$
3. $u''-\exp(-x)u(x)=2x\sqrt{3+u};\quad u(0)=1,\quad u'(1)=1.$
4. $u''+2u'-2u(x)=2x-\exp(2-u);\quad u'(0)+2u(0)=2,\quad u(1)=2.$
5. $u''-2u'+u^2=2x\sin(x);\quad u(1)=2,\quad u(2)-u'(2)=1.$
6. $u''-2u^2=2x\sin(u-5);\quad u(1)=2,\quad u'(2)=2.$
7. $u'''-2u(x)=2x-\sin(u-1);\quad u'(1)=0,\quad u(1)=1,\quad u'(2)=1.$
8. $u^{IV}-2u(x)=2x-\cos(u+1);\quad u'(0)+2u(0)=2,\quad u(2)=-1,\quad u'(2)=0,\quad u''(2)=0.$
9. $u'''+u''-u(x)=2x\sin(u);\quad u(1)=1,\quad u'(1)=0,\quad u'(2)=-1.$
10. $u'''+u'(x)=2x-\exp(-u);\quad u'(0)+2u(0)=2,\quad u(2)=1,\quad u'(2)-u(2)=0.$
11. $u^{IV}-xu(x)=2x\sin(u'-5);\quad u(1)=2,\quad u(2)=-1,\quad u'(2)=1,\quad u''(2)=0.$
12. $u'''+xu'-u(x)=2x-\cos(u-1);\quad u(1)=2,\quad u'(1)=0,\quad u(2)-2u'(2)=1.$
13. $u''-\exp(-x)u'+2u(x)=2x\sqrt{3+u};\quad u(0)=1,\quad u'(1)=1.$
14. $u^{IV}+2u''-2u(x)=2x-\exp(-u);\quad u(0)=1,\quad u'(0)=1,\quad u''(0)=0,\quad u(1)=2.$
15. $u''-2(u')^2=2x-\sin(x-1);\quad u(1)=2,\quad u(2)-u'(2)=1.$
16. $u^{IV}-u^2=2x;\quad u(1)=1,\quad u'(1)=2,\quad u(2)=2,\quad u'(2)=0.$
17. $u'''-2u'=2x\sin(u);\quad u'(1)=0,\quad u(1)=1,\quad u'(2)=1.$

18. $u^{IV} - u'' = 2x - \cos(u+1);\quad u'(0) + 2u(0) = 2,\quad u(2) = -1,\quad u'(2) = 0,\quad u''(2) = 0.$
19. $u''' + u'' - 2u' = 2x\sin(u);\quad u(1) = 1,\quad u'(1) = 0,\quad u'(2) = -1.$
20. $u''' + 2u'' - u'(x) = 2x\exp(-u);\quad u'(1) + 2u(1) = 2,\quad u(2) = 1,\quad u'(2) - u(2) = 0.$
21. $u'' - \exp(-x)u' = 2x\sqrt{3+u};\quad u(0) = 0,\quad u'(1) = 0.$
22. $u'' - u' - 2u(x) = x - \exp(-u);\quad u'(0) - u(0) = 2,\quad u(1) = 1.$
23. $u'' - 2u^2u' = 2 + x;\quad u(1) = 1,\quad u(2) - u'(2) = 1.$
24. $u''' - 2u^2 = 2x\sin(u-5);\quad u(1) = 2,\quad u'(1) = 0,\quad u'(2) = 2.$
25. $u^{IV} + u'' - u(x) = 2x\cos(u);\quad u(1) = 1,\quad u'(1) = 0,\quad u(2) = 1,\quad u'(2) = 0.$
26. $u'' - \exp(-x)u' = 2x\sqrt{3+u};\quad u(0) = u(1),\quad u'(0) - u'(1) = 0.$
27. $u'' - u' - 2u(x) = x - \exp(-u);\quad u(0) - u(1) = 2,\quad u'(0) - u'(1) = 1.$
28. $u'' - 2x^2u' = 2 + x;\quad u(1) + u'(2) = 1,\quad u'(1) + u(2) = 1.$
29. $u'' - (u')^2 - u = \sin(x+5);\quad u(1) - u'(2) = 2,\quad u'(1) - u(2) = 1.$
30. $u^{IV} + u'' - xu = 2x\cos(u);\quad u(1) = u(2),\quad u'(1) = u'(2),\quad u''(1) = u''(2),\quad u'''(1) = u'''(2).$
31. $ru''(r) + u'(r) - r^{-1}u = \sin(r+\pi);\quad u(1) = 1,\quad u'(\pi) = 1 - u(\pi).$
32. $ru''(r) + u'(r) = u * \sin(r+\pi);\quad u(\pi) = 1,\quad u'(2\pi) + u(2\pi) = 2.$
33. $ru''(r) + u'(r) - r^{-2}u = \cos(r-\pi);\quad u(\pi) = 1,\quad u'(2\pi) = 2 - u(2\pi).$
34. $ru''(r) + u'(r) = 2u;\quad u(\pi) = u(2\pi),\quad u'(\pi) = u'(2\pi).$
35. $u''(r) + r^{-1}u'(r) = u * (r+\pi);\quad u(\pi) = u(2\pi),\quad u'(\pi) - u'(2\pi) = 0.$

Побудувати графік розв'язку $u(x)$ і в окремих підграфіках наявні похідні від функції розв'язку.

Варіанти задач – для параметра k - № по списку.

Практичне заняття № 10

ВИКОРИСТАННЯ СОЛВЕРІВ MATLAB ДЛЯ РОЗВ'ЯЗАННЯ ОДНОВИМІРНИХ КРАЙОВИХ ЗАДАЧ ДЛЯ СИСТЕМ ПАРАБОЛІЧНИХ І ЕЛІПТИЧНИХ ДРЧП

Аудиторне заняття

1. Розглянемо мішану задачу для рівняння параболічного типу з циліндричною симетрією

$$\begin{aligned} &u_t'(r,t) = r * u_{rr}'' + u_r' - \frac{u}{r}, \quad r \in (0,1), t \in (0,1]; \\ &u(r,0) = J_2\left(\mu_k \sqrt{r}\right), \quad r \in [0,1]; \\ &u_r'(0,t) = 0, \quad u(1,t) = 0, \quad t \in [0,1]; \end{aligned} \tag{10.1}$$

Тут позначено: $J_2(z)$ – *функція Бесселя першого роду цілого порядку* = 2, а $\mu_k > 0$ – корінь рівняння $J_2(\mu) = 0$.

За допомогою солвера *pdepe* знайти $U(\mathrm{R},\mathrm{T})$ – наближений розв'язок цієї задачі на сітці R,T, порівняти його з розв'язком отриманим методом Фур'є

$$u(r,t) = J_2\left(\mu_k \sqrt{r}\right) * \exp\left(-\left(0.5\mu_k\right)^2 * t\right).$$

Побудувати графіки $U(\mathrm{R},\mathrm{T})$, $|U(\mathrm{R},\mathrm{T}) - u(\mathrm{R},\mathrm{T})|$ і знайти значення $\|U(\mathrm{R},\mathrm{T}) - u(\mathrm{R},\mathrm{T})\|_C$.

Єдине, що необхідно перетворити до форми задачі, означеної для солвера *pdepe*, це рівняння. Отже, спочатку поділимо рівняння на *r*, потім суму диференціальних операторів по *r* приведемо до дивергентного вигляду і отримаємо :

$$\underbrace{\left(\frac{1}{r}\right)}_{C(\ldots)} * \frac{\partial u(r,t)}{\partial t} = r^{-1} * \frac{\partial}{\partial r}\left(r * \underbrace{\frac{\partial u(r,t)}{\partial r}}_{f(\ldots)} \right) + \underbrace{\left(-\frac{u}{r^2}\right)}_{S(\ldots)}, \quad m = 1.$$

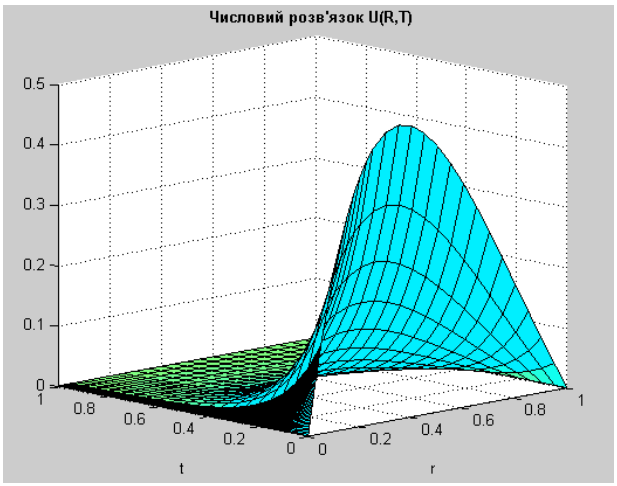
Тепер можна застосовувати солвер *pdepe*. Слідуючи стандартній схемі використання цього солвера (див Лекцію №10), маємо таку М-функцію :

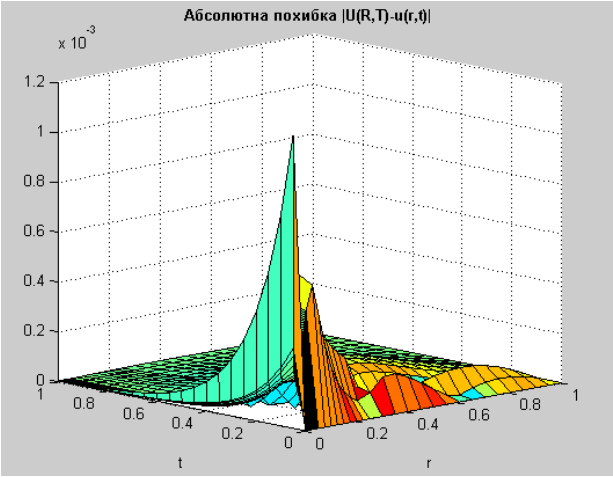
```
function pr10_1
% Розв'язання мішаної крайової задачі для
% неоднорідного рівняння теплопровідності
%     (в циліндричній симетрії):
%  du/dt = r*(r*u')' - u/r,
%  r є (0,1), t є (0,1];
%  u'(0,t) = 0, u(1,t) = 0,
%  u(r,0) = J(2,mk*sqrt(x)).
% за допомогою вбудованих функцій MATLAB.
% Розв'язок, отриманий м.Фур'є :
% u(r,t) = J(2,mk*sqrt(x))*exp(-0.25*mk^2*t).

close all, clear all, clc

global mk
mk = 5.135622302305;  % корінь рівняння J(2,m)=0

m = 1; % циліндрична симетрія

rl = 0; rr = 1;
% будуємо нерівномірну сітку по r
r = [[rl:0.001:0.05],[0.075:0.025:0.25],[0.3:0.05:rr]];
t = linspace(0, 1, 21);     % вектор по часу
% Розв'язуємо задачу
```

```
sol = pdepe(m, @koefpde, @initpde, @boundpde, r, t);
U = sol(:,:,1); % скалярне рівняння => функція одна

s = 'Числовий розв''язок U(R,T)';
[R, T] = meshgrid(r, t);
surfl(R, T, U), colormap('default')
title(s, 'FontWeight', 'bold')
xlabel('r'), ylabel('t')
view(-44, 13)
pause

% u(x,t) за методом Фур'є :
m4 = -0.25.*mk.*mk;
u = @(r,t)(besselj(2,mk.*sqrt(r)).*exp(m4.*t));

s = 'Абсолютна похибка |U(R,T)-u(r,t)|';
U = abs(U - u(R,T));
fprintf([s,' в нормі C =%.6e\n'], norm(U,inf))
surfl(R, T, U), colormap('default')
title(s, 'FontWeight', 'bold')
xlabel('r'), ylabel('t')
view(-44, 13)

function [c, f, s] = koefpde(r, t, u, DuDr)
% коефіцієнти рівняння
c = 1./r;
f = DuDr;
s = -u./(r.*r);

function u0 = initpde(r)
% початкова умова u0(r)
global mk
u0 = besselj(2,mk.*sqrt(r));

function [pl, ql, pr, qr] = boundpde(xl, ul, xr, ur, t)
% коефіцієнти крайових умов
pl = 0; ql = 1;
pr = ur; qr = 0;
```

Результати виконання *pr10_1*:

```
Абсолютна похибка |U(R,T)-u(r,t)| в нормі C =1.959678e-002
```





2. Розглянемо мішану задачу для квазілінійного рівняння параболічного типу з циліндричною симетрією

$$\frac{\partial u(r,t)}{\partial t} = r^{-1} * \frac{\partial}{\partial r}\left(r * \frac{\partial u(r,t)}{\partial r} \right) - \cos\left(u * r^{-2}\right), \quad r \in (1,2), t \in (0,1];$$

$$u(r,0) = (r^2 - 1) * (2 - r), \quad r \in [1,2]; \qquad (10.2)$$

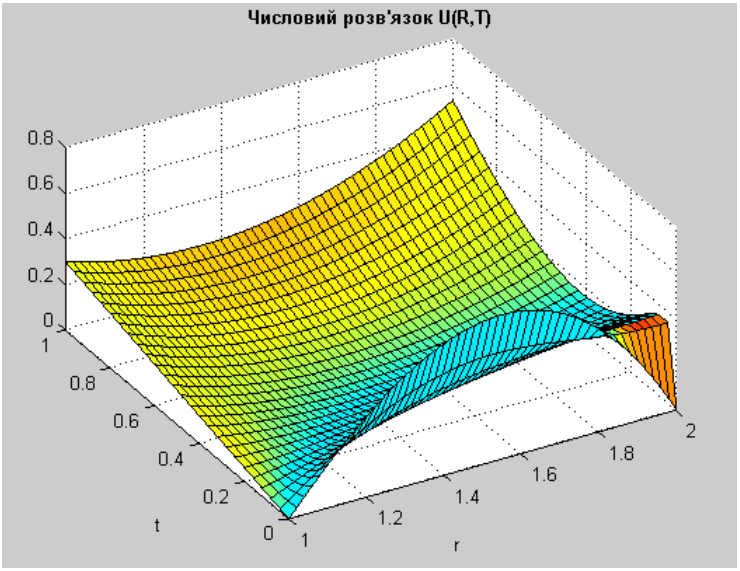
$$u(1,t) = 0.3t, \quad \frac{\partial u(2,t)}{\partial r} = t, \quad t \in [0,1];$$

За допомогою солвера *pdepe* знайти $U(\mathrm{R},\mathrm{T})$ – наближений розв'язок цієї задачі на сітці R,T. Постановка задачі (10.2) повністю співпадає з формою задачі, означеної для солвера *pdepe*. Слідуючи стандартній схемі використання цього солвера (див Лекцію №10), маємо таку М-функцію :

```
function pr10_2
% Розв'язання мішаної крайової задачі для
% неоднорідного рівняння теплопровідності
%     (в циліндричній симетрії):
%  du/dt = r*(r*u')' - cos(u/r^2),
%  r є (1,2), t є (0,1];
%  u(1,t) = 0.3t, u'(2,t) = t,
%  u(r,0) = (r^2-1)*(2-r).
% за допомогою вбудованих функцій MATLAB.

close all, clear all, clc

m = 1; % циліндрична симетрія
rl = 1; rr = 2; r = linspace(rl, rr, 51); % сітка по r
t = linspace(0, 1, 26);     % вектор по часу

% Розв'язуємо задачу
sol = pdepe(m, @koefpde, @initpde, @boundpde, r, t);
U = sol(:,:,1); % скалярне рівняння => функція одна

s = 'Числовий розв''язок U(R,T)';
[R, T] = meshgrid(r, t);
surfl(R, T, U), colormap('default')
title(s, 'FontWeight', 'bold')
xlabel('r'), ylabel('t')
view(-30, 50)

function [c, f, s] = koefpde(r, t, u, DuDr)
% коефіцієнти рівняння
c = 1;
f = DuDr;
s = -cos(u./(r.*r));

function u0 = initpde(r)
```

```
% початкова умова u0(r)
u0 = (r.^2 - 1).*(2 - r);

function [pl, ql, pr, qr] = boundpde(xl, ul, xr, ur, t)
% коефіцієнти крайових умов
pl = ul - 0.3.*t; ql = 0;
pr = -t; qr = 1;
```

Результати виконання *pr10_2*:



3. Знайти наближений розв'язок для функцій $u(x,t)$, $v(x,t)$, які задовольняють початково-крайову задачу

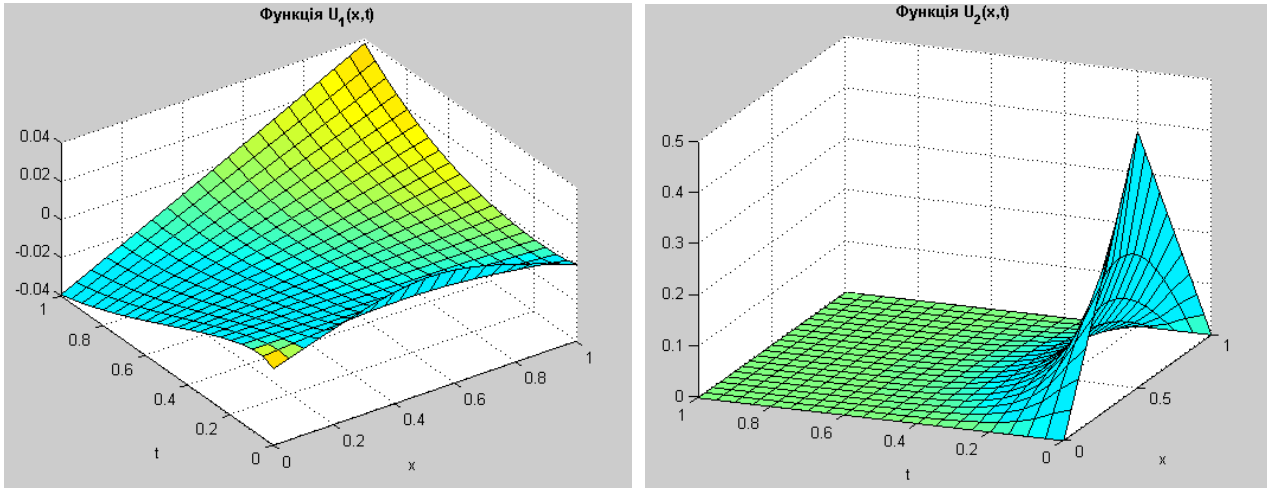
$$\begin{cases} 0=\dfrac{\partial}{\partial x}\left(K(x,t)\dfrac{\partial u}{\partial x}\right)+\left(x^2+\exp(-t)\right)*v^2, \\ \qquad K(x,t)=\exp(-0.05x^2); \\ \dfrac{\partial v}{\partial t}=\dfrac{\partial}{\partial x}\left(\dfrac{\partial v}{\partial x}\right)+v*u^2,\ x\in(0,1),\ t\in(0,1]; \\ \left(2*Ku'_x-0.5u\right)\big|_{x=0}=0.2t,\ \left(Ku'_x+0.2u\right)\big|_{x=1}=-0.03t, \\ \qquad v(0,t)=0, \qquad v(1,t)=0,\ t\in[0,1]; \\ v(x,0)=1-|1-x|,\ x\in[0,1]. \end{cases} \tag{10.3}$$

Якщо перепозначити $u(x,t)$, $v(x,t)\to U_1(x,t)$, $U_2(x,t)$, то задача (10.3) просто переписується до вигляду застосування солвера *pdepe* з m=0. Створимо відповідну M-функцію, слідуючи стандартній схемі використання цього солвера:

```
function pr10_3
% Розв'язання задачі для U1(x,t),U2(x,t)
%      0 = (K(x,t)*U1')'+(x^2+exp(-t))*u(2)^2,
% dU2/dt = U2" + U2*U1^2,
%           x є (0,1), t є (0,1];
%           K = exp(-0.05*x^2);
% (2*K*U1'-0.5*U1) =  0.2t, U2(x,t) = 0, x=0,
%   (K*U1'+0.2*U1) = -0.3t, U2(x,t) = 0, x=1, tє(0,1];
%   U2(x,0) = 0.5-|0.5-x|, xє[0,1].
% за допомогою вбудованих функцій MATLAB.

close all, clear all, clc

m = 0; % плоска задача

xl = 0; xr = 1; x = linspace(xl, xr, 21); % сітка по x
t = linspace(0, 1, 21);                   % вектор по часу
```

```
% Розв'язуємо задачу
sol = pdepe(m, @koefpde, @initpde, @boundpde, x, t);

[X, T] = meshgrid(x, t);
% графіки розв'язку U1(x,t), U2(x,t)

u = sol(:, :, 1);
surfl(X, T, u), colormap('default')
s = 'Функція U{_1}(x,t)';
title(s, 'FontWeight', 'bold')
view(-35, 50)
xlabel('x'), ylabel('t')
pause

u = sol(:, :, 2);
surfl(X, T, u), colormap('default')
s = 'Функція U{_2}(x,t)';
title(s, 'FontWeight', 'bold')
view(-68, 24)
xlabel('x'), ylabel('t')

function [c, f, s] = koefpde(x, t, u, DuDx)
% коефіцієнти рівняння
c = [0; 1];
f = [exp(-0.05.*x.*x).*DuDx(1); DuDx(2)];
s = [(x.*x+exp(-t)).*u(2).*u(2); u(1).^2.*u(2)];

function u0 = initpde(x)
% початкова умова
u0 = [0; 0.5-abs(0.5-x)];

function [pl, ql, pr, qr] = boundpde(xl, ul, xr, ur, t)
% коефіцієнти крайових умов
pl = [-0.2.*t-0.5.*ul(1); ul(2)];
ql = [2; 0];
pr = [0.2.*ur(1)+0.03.*t; ur(2)];
qr = [1; 0];
```

Результати виконання *pr10_3*:



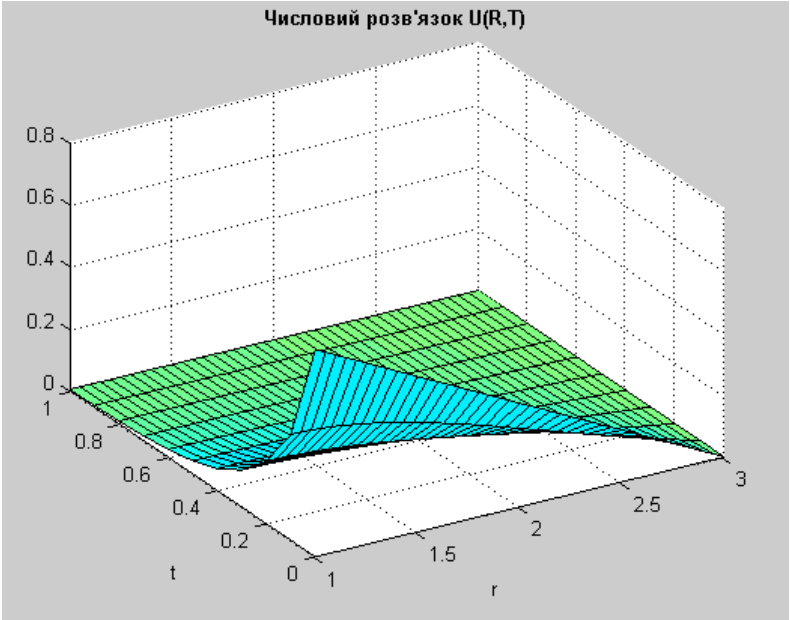
4. Розв’яжемо ще одну задачу для однорідного рівняння теплопровідності (у варіанті сферичної симетрії) :

$$\begin{cases} a^{-2} * \dfrac{\partial u(r,t)}{\partial t} = r^{-2}\,\dfrac{\partial}{\partial r}\left(r^2\,\dfrac{\partial u(r,t)}{\partial r}\right), \\ r \in (rb, re), \;\; t \in (0, \mathrm{T}]; \\ u(r,0) = f_0 * \left(1 - \dfrac{r}{re}\right), \; r \in [rb, re]; \\ -0.01 * u(rb,t) + \dfrac{\partial u(rb,t)}{\partial r} = 0, \;\; u(re,t) = 0, \;\; t \in [0, \mathrm{T}]; \\ a = 2, \; rb = 1, \; re = 3, \; \mathrm{T} = 1, \; f_0 = 1. \end{cases} \tag{10.4}$$

Задача повністю записана в формі використання солвера *pdepe* з *m*=2, тому створимо М-функцію *pr10_4*:

```
function pr10_4
% Розв'язання мішаної крайової задачі для
%  однорідного рівняння теплопровідності
% в сфері з порожниною (сферична симетрія):
%     du/dt = (a/r)^2*(r^2*u')',
%       r є (rb,re), t є (0,T];
%     u'(rb,t) = 0.01*u(rb,t), u(re,t) = 0,
%          u(r,0) = f0*(1-r/re).
% за допомогою вбудованих функцій MATLAB.

close all, clear all, clc
global re f0 aa
a = 2; rb = 1; re = 3; T = 1; f0 = 1;
aa = 1./(a.*a);

m = 2; % сферична симетрія
rl = rb; rr = re;
% будуємо рівномірні сітки :
r = linspace(rl, rr, 41);   %  по r
t = linspace(0, T, 11);     %  по t
% Розв'язуємо задачу
sol = pdepe(m, @koefpde, @initpde, @boundpde, r, t);
U = sol(:,:,1); % скалярне рівняння => функція одна

[R, T] = meshgrid(r, t);

surfl(R, T, U), colormap('default')
s = 'Числовий розв''язок U(R,T)';
title(s, 'FontWeight', 'bold')
xlabel('r'), ylabel('t')
view(-31, 38)

function [c, f, s] = koefpde(r, t, u, DuDr)
% коефіцієнти рівняння
global aa
c = aa;  f = DuDr;  s = 0;

function u0 = initpde(r)
% початкова умова u0(r) = f0*(1-r/re)
global re f0
u0 = f0.*(1 - r./re);

function [pl, ql, pr, qr] = boundpde(xl, ul, xr, ur, t)
% коефіцієнти крайових умов
pl = -0.01.*ul; ql = 1;
pr = ur; qr = 0;
```

Результати виконання *pr10_4* :



Після розв'язання задач (10.1)-(10.4), а також використовуючи теоретичні відомості про солвер `pdepe` і розглянуті приклади задач (10.4)-(10.7) в Лекції №10, можна переходити до розв'язання індивідуальних завдань.

<u>Завдання для самостійної роботи</u>

Написати М-файл розв'язання крайової задачі з використанням солвера `pdepe`. Побудувати графіки розв'язку.

В умовах задач позначено:

$$U \equiv U(x,t),\ V \equiv V(x,t);\quad \dot{V} \equiv \frac{\partial V(x,t)}{\partial t};$$

$$U' \equiv \frac{\partial U(x,t)}{\partial x},\ U'' \equiv \frac{\partial^2 U(x,t)}{\partial x^2},\ V' \equiv \frac{\partial V(x,t)}{\partial x},\ V'' \equiv \frac{\partial^2 V(x,t)}{\partial x^2}.$$

№	Крайова задача	№	Крайова задача
1	$\begin{cases} U''-2U=2x\sin(V-5), \\ \dot{V}=3V''+xt(U+V),\ \ t\in(0,2]; \\ U'(0,t)+2U(0,t)=2,\quad U(2,t)=-1; \\ V'(0,t)=t,\quad V(2,t)=2, \\ V(x,0)=x. \end{cases}$	***14***	$\begin{cases} U''+xU'-U=2x-\cos(V-1), \\ \dot{V}=3V''+xt(U-V),\ \ t\in(0,2]; \\ U'(1,t)=2,\quad U(2,t)-2U'(2,t)=1; \\ V'(1,t)=1,\quad V(2,t)=t, \\ V(x,0)=1-(1-x)^2. \end{cases}$
2	$\begin{cases} U''-\exp(-x)U(x)=2xV\sqrt{3+U}, \\ \dot{V}=3V''+xt(2U-V),\ \ t\in(0,2]; \\ U(0,t)=1,\quad U'(1,t)=1; \\ V'(0,t)=1,\quad V(1,t)=t, \\ V(x,0)=1-x^3. \end{cases}$	***15***	$\begin{cases} U''-2(U')^2=2V-\sin(xt-1), \\ \dot{V}=V''+2U-0.5V,\ \ t\in(0,2]; \\ U(1,0)=2,\quad U(2)-U'(2)=1; \\ V'(1,t)=1,\quad V(2,t)=t, \\ V(x,0)=1-\sqrt{x-1}. \end{cases}$

3	$\begin{cases} U''+2U'-2U=2x-\exp(2-V) \\ \dot V=V''+2xt-UV,\ \ t\in(0,2] \\ U'(0,t)+2U(0,t)=2,\ \ U(1,t)=2; \\ V'(0,t)=0,\ \ V(1,t)=t, \\ V(x,0)=x-\sqrt{x^3}. \end{cases}$	***16***	$\begin{cases} U''-U^2=2V, \\ \dot V=3V''+xt(U-V),\ \ t\in(0,2] \\ U'(1,t)=2,\ \ U(2,t)-2U'(2,t)=1; \\ V'(1,t)=1,\ \ V(2,t)=t, \\ V(x,0)=1-(1-x)^2. \end{cases}$
4	$\begin{cases} U''-2U'+U^2=2x\sin(x)V, \\ \dot V=V''+UV,\ \ t\in(0,2] \\ U(1,0)=2,\ \ U(2)-U'(2)=1; \\ V'(1,t)=0,\ \ V(2,t)=1+t, \\ V(x,0)=1-\sqrt{2-x}. \end{cases}$	***17***	$\begin{cases} U''-2xU=2t\sin(V), \\ \dot V=V''+(x-t)(U+V),\ \ t\in(0,2] \\ U'(0,t)+U(0,t)=1,\ \ U(2,t)=0; \\ V'(0,t)=t,\ \ V(2,t)=2-\sqrt{t}, \\ V(x,0)=x. \end{cases}$
5	$\begin{cases} U''-2U^2=2x\sin(V-5), \\ \dot V=V''+U^2-xtV,\ \ t\in(0,2] \\ U(1,t)=2,\ \ U'(2,t)=2; \\ V'(1,t)=t,\ \ V(2,t)=1-t, \\ V(x,0)=\sqrt{x-1}. \end{cases}$	***18***	$\begin{cases} U''-xU'-U=2tV, \\ \dot V=V''+x^2U(t-V),\ \ t\in(0,2] \\ U'(0,t)-U(0,t)=0,\ \ U'(1,t)=0; \\ V'(0,t)=t,\ \ V(1,t)=1-t^2, \\ V(x,0)=x^2\sqrt{2-x}. \end{cases}$
6	$\begin{cases} U''-2U(x)=2x-\sin(UV-1), \\ \dot V=V''+xU^2-tV,\ \ t\in(0,2] \\ U'(1,t)=0,\ \ U(2,t)=1; \\ V'(1,t)=t,\ \ V(2,t)=2-t, \\ V(x,0)=x\sqrt{x-1}. \end{cases}$	***19***	$\begin{cases} U''-2xtU=2x\sin(V), \\ \dot V=2V''+xt-UV,\ \ t\in(0,2] \\ U'(0,t)+2U(0,t)=2,\ \ U(1,t)=2; \\ V'(0,t)=0,\ \ V(1,t)=t, \\ V(x,0)=x-\sqrt{x^3}. \end{cases}$
7	$\begin{cases} U''-xU=2V\cos(xt), \\ \dot V=V''+xU(1-tV),\ \ t\in(0,2] \\ U(0,t)=0,\ \ U'(1,t)=0; \\ V'(0,t)=t,\ \ V(1,t)=1-t, \\ V(x,0)=x^2\sqrt{1-x}. \end{cases}$	***20***	$\begin{cases} 2U''-tU'-xU=2x\exp(-V), \\ \dot V=2V''+(x-t-V)U^2,\ \ t\in(0,2] \\ U(0,t)+U'(0,t)=1,\ \ U(1,t)=0; \\ V'(0,t)=\sqrt{t},\ \ V(1,t)=t, \\ V(x,0)=x^3(1-x). \end{cases}$
8	$\begin{cases} U''-UU'=2V, \\ \dot V=V''+U(x-tV),\ \ t\in(0,2] \\ U(0,t)=0,\ \ U'(1,t)+U(1,t)=0; \\ V'(0,t)=\sqrt{t},\ \ V(1,t)=0.5t, \\ V(x,0)=x^3\sqrt{1-x}. \end{cases}$	***21***	$\begin{cases} U''-\exp(-x)U'=2xV\sqrt{3+t}, \\ \dot V=V''+(x-t)(2U-V),\ \ t\in(0,2] \\ U'(0,t)=2,\ \ U(2,t)-2U'(2,t)=1; \\ V'(0,t)=2,\ \ V(2,t)=t, \\ V(x,0)=2-\sqrt{2x}. \end{cases}$
9	$\begin{cases} U''+U'=2x-\exp(-UV), \\ \dot V=2V''+(xt-V)U^2,\ \ t\in(0,2] \\ U(0,t)+U'(0,t)=2,\ \ U(1,t)=0; \\ V'(0,t)=0,\ \ V(1,t)=t, \\ V(x,0)=x(1-x^3). \end{cases}$	***22***	$\begin{cases} U''-2U'-U=x-\exp(-Vt), \\ \dot V=V''+(4xt-V)U,\ \ t\in(0,2] \\ U(1,t)=2,\ \ U'(2,t)=0; \\ V'(1,t)=0,\ \ V(2,t)=\sqrt{t^3}, \\ V(x,0)=(1-x)(2-x). \end{cases}$

10	$\begin{cases} U''-xU=2xt\sin(V-5), \\ \dot V=2V''+(xt-V)U, \quad t\in(0,2]; \\ U(1,t)=2, \quad U'(2,t)=0; \\ V'(1,t)=0, \quad V(2,t)=t, \\ V(x,0)=(1-x)(2-x). \end{cases}$	***23***	$\begin{cases} U''-2x^2U=2xV, \\ \dot V=V''+txU^2V, \quad t\in(0,2]; \\ U(0,t)=0, \quad U'(1,t)=1; \\ V'(0,t)=0, \quad V(1,t)=\sqrt{1+t}, \\ V(x,0)=x(2-x). \end{cases}$
11	$\begin{cases} U''+xU'-U=xtV, \\ \dot V=4V''+(xt-1)UV, \quad t\in(0,2]; \\ U'(1,t)+U(1,t)=2, \quad U(2,t)=0; \\ V'(1,t)=1, \quad V(2,t)=\sqrt{1+t}, \\ V(x,0)=\lvert 2x-3\rvert. \end{cases}$	***24***	$\begin{cases} U''-2xtU'-U=\sin(V-3), \\ \dot V=2V''+(xt-V)U, \quad t\in(0,2]; \\ U(1,t)=2, \quad U'(2,t)=0; \\ V'(1,t)=0, \quad V(2,t)=t, \\ V(x,0)=(1-x^2)(4-x^2). \end{cases}$
12	$\begin{cases} U''-\exp(-x)U'+2U=2x\sqrt{3+V^2}, \\ \dot V=V''+tU^2V, \quad t\in(0,2]; \\ U(0,t)=1, \quad U'(2,t)=0; \\ V'(0,t)=0, \quad V(2,t)=\sqrt{4+t}, \\ V(x,0)=x(3-x). \end{cases}$	***25***	$\begin{cases} U''+U'-4U=x\cos(x)V, \\ \dot V=V''+x^2U(2t-V), \quad t\in(0,2]; \\ U'(0,t)-U(0,t)=0, \quad U'(1,t)=0; \\ V'(0,t)=t, \quad V(1,t)=1-t^2, \\ V(x,0)=x^2\sqrt{2-x}. \end{cases}$
13	$\begin{cases} U''-\exp(-U)=xtV, \\ \dot V=2V''+(xt-V)U, \quad t\in(0,2]; \\ U(1,t)=2, \quad U'(2,t)=0; \\ V'(1,t)=0, \quad V(2,t)=t, \\ V(x,0)=(1-x^2)(4-x^2). \end{cases}$	***26***	$\begin{cases} U''-UU'=2x(V-5), \\ \dot V=V''+U(1-xtV), \quad t\in(0,2]; \\ U(1,t)=2, \quad U'(2,t)=2; \\ V'(1,t)=t, \quad V(2,t)=1-t, \\ V(x,0)=\sqrt{x-1}. \end{cases}$

Практичне заняття № 11

СИМВОЛЬНІ ОБЧИСЛЕННЯ В СИСТЕМІ MATLAB

Аудиторне заняття

Мета заняття :

- ❖ освоїти технологію використання символьних змінних;
- ❖ навчитись будувати символьні вирази (СВ); використовувати : операції перетворення *sym-class* ↔ *(char, double)-class*, функції *vpa(), eval()*;
- ❖ опанувати використання *аналітичних операцій* з СВ;
- ❖ опанувати використання *символьних операцій математичного аналізу*;
- ❖ розглянути приклади програм з перетворенням СВ і використанням функцій Maple.

Як і в Лекції №11, текст відповідних прикладів-програм і результати покрокового виконання відповідних операцій будемо розміщувати в таблицях.

Приклад 1. Спрощення алгебраїчних виразів (М-файл *pr11_1.m*).

Оператори програми	**Результат виконання оператора**
%% pr11_1.m *clear all, close all, clc* *% Спрощення алгебраїчних виразів :* *syms x a b* *x = 1/2*(sqrt(a/b)+sqrt(b/a))*	x = 1/2*(a/b)^(1/2)+1/2*(b/a)^(1/2)
*e = 2*b*sqrt(x^2-1)/(x-sqrt(x^2-1))*	e = 2*b*((1/2*(a/b)^(1/2)+1/2*(b/a)^(1/2))^2- 1)^(1/2)/(1/2*(a/b)^(1/2)+1/2*(b/a)^(1/2)- ((1/2*(a/b)^(1/2)+1/2*(b/a)^(1/2))^2-1)^(1/2))
[e1,how] = simple(e)	**e1 =** **a-b** **how =** **radsimp**
e2 = simplify(e)	*e2 =* *2*b*((a^2+2*(a/b)^(1/2)*(b/a)^(1/2)*b*a+b^2-* *4*b*a)/b/a)^(1/2)/((a/b)^(1/2)+(b/a)^(1/2)-* *((a^2+2*(a/b)^(1/2)*(b/a)^(1/2)*b*a+b^2-* *4*b*a)/b/a)^(1/2))*
e3 = simplify(expand(e))	*e3 =* *2*b*((a^2+2*(a/b)^(1/2)*(b/a)^(1/2)*b*a+b^2-* *4*b*a)/b/a)^(1/2)/((a/b)^(1/2)+(b/a)^(1/2)-* *((a^2+2*(a/b)^(1/2)*(b/a)^(1/2)*b*a+b^2-* *4*b*a)/b/a)^(1/2))*
e4 = expand(e2)	*e4 =* *2*b*(a/b+2*(a/b)^(1/2)*(b/a)^(1/2)+b/a-* *4)^(1/2)/((a/b)^(1/2)+(b/a)^(1/2)-* *(a/b+2*(a/b)^(1/2)*(b/a)^(1/2)+b/a-4)^(1/2))*
disp('--------') *clear x, syms x* *e = (2*x+1-1/(1-2*x))*(2*x-* *4*x^2/(2*x-1))*	e = (2*x+1-1/(1-2*x))*(2*x-4*x^2/(2*x-1))
[e1,how] = simple(e)	e1 = -8*x^3/(2*x-1)^2 how = simplify
e2 = simplify(e)	e2 = -8*x^3/(2*x-1)^2
disp('--------') *clear x, syms x* *e = ((x^3-8)/(x^2+4+2*x)+4)/(x+2)*	e =

	((x^3-8)/(x^2+4+2*x)+4)/(x+2)
[e1,how] = simple(e)	e1 = 1 how = simplify
e2 = simplify(e)	e2 = 1
ee = expand(e)	ee = 1/(x+2)/(x^2+4+2*x)*x^3- 8/(x+2)/(x^2+4+2*x)+4/(x+2)
e3 = simplify(ee)	e3 = 1
[e4,how] = simple(ee)	e4 = 1 how = simplify

Приклад 2. Розкриття дужок (М-файл *pr11_2.m*).

Оператори програми	**Результат виконання оператора**
%% pr11_2.m clear all, close all, clc % Розкриття дужок : syms x a b c e = (x-a)*(x+b)*(x^2-c)	e = (x-a)*(x+b)*(x^2-c)
e1 = expand(e)	e1 = x^4-x^2*c+x^3*b-x*b*c-a*x^3+a*x*c-b*a*x^2+b*a*c
e2 = collect(e1,x)	e2 = x^4+(-a+b)*x^3+(-b*a-c)*x^2+(a*c-b*c)*x+b*a*c
e3 = collect(e1,'x')	e3 = x^4+(-a+b)*x^3+(-b*a-c)*x^2+(a*c-b*c)*x+b*a*c
e4 = simple(e3)	e4 = -(-x+a)*(x+b)*(x^2-c)
disp('--------') clear a b c e = sin(x)-2*sin(2*x)+4*sin(3*x)	e = sin(x)-2*sin(2*x)+4*sin(3*x)
e1 = expand(e)	e1 = -3*sin(x)-4*sin(x)*cos(x)+16*sin(x)*cos(x)^2
disp('--------') e = cos(2*x)*sin(x)*sin(3*x)	e = cos(2*x)*sin(x)*sin(3*x)
e1 = expand(e)	e1 = 8*cos(x)^4*sin(x)^2-6*sin(x)^2*cos(x)^2+sin(x)^2

Приклад 3. Розклад на прості множники (М-файл *pr11_3.m*).

Оператори програми	**Результат виконання оператора**
%% pr11_3.m clear all, close all, clc % Розклад на прості множники : syms x y e = x^5-32	e = x^5-32
e1 = factor(e)	e1 = (x-2)*(x^4+2*x^3+4*x^2+8*x+16)
disp('--------') e = x^5-30	e = x^5-30
e1 = factor(e)	e1 = x^5-30
e = sym('x^5-30')	e = x^5-30
e1 = factor(e)	e1 = x^5-30
e = sym('x^5-((30)^(1/5))^5')	e = x^5-((30)^(1/5))^5
e1 = factor(e)	e1 = x^5-30

`e = sym('x^5-a^5')`	`e =` `x^5-a^5`
`e1 = factor(e)`	`e1 =` `-(-x+a)*(a^4+a^3*x+a^2*x^2+a*x^3+x^4)`
`disp('--------')` `e = 156`	`e =` `156`
`e1 = factor(e)`	`e1 =` `2 2 3 13`
`%e = syms(156)  % помилка` `%e1 = factor(e)`	
`disp('--------')` `e = 3*x^3+10*x*y-8*y^2-8*x+10*y-3`	`e =` `3*x^3+10*x*y-8*y^2-8*x+10*y-3`
`e1 = factor(e)`	`e1 =` `3*x^3+10*x*y-8*y^2-8*x+10*y-3`
`disp('--------')` `e = (x^2+5*x+4)/(x^4+5*x^2+4)`	`e =` `(x^2+5*x+4)/(x^4+5*x^2+4)`
`e1 = factor(e)`	`e1 =` `(x+4)*(x+1)/(x^2+4)/(x^2+1)`
`disp('--------')` `e = 1-2*x^2`	`e =` `1-2*x^2`
`e1 = factor(e)`	`e1 =` `1-2*x^2`
`syms a` `e = 1-a*x^2`	`e =` `1-a*x^2`
`e1 = factor(e)`	`e1 =` `1-a*x^2`
`e = 1-a^2*x^2`	`e =` `1-a^2*x^2`
`e1 = factor(e)`	**`e1 =`** **`-(a*x-1)*(a*x+1)`**
`e = 1-(3*x)^2`	`e =` `1-9*x^2`
`e1 = factor(e)`	**`e1 =`** **`-(3*x-1)*(3*x+1)`**
`e = 1-(a*x)^2`	`e =` `1-a^2*x^2`
`e1 = factor(e)`	**`e1 =`** **`-(a*x-1)*(a*x+1)`**

Приклад 4. Підстановка і приведення подібних, форма Горнера для полінома (М-файл *pr11_4.m*).

Оператори програми	**Результат виконання оператора**

`%% pr11_4.m`
`clear all, close all, clc`
`% Підстановка і приведення подібних`
`% форма Горнера для полінома`
`syms x y z`
`e = x^4-100+45*x+x^3-17*x^2-100`
	`e =`
	`x^4-200+45*x+x^3-17*x^2`

`e1 = collect(e)`	`e1 =` `x^4-200+45*x+x^3-17*x^2`
`e1 = collect(e,x)`	`e1 =` `x^4-200+45*x+x^3-17*x^2`
`e1 = collect(e,'x')`	`e1 =` `x^4-200+45*x+x^3-17*x^2`
`e1 = simplify(e)`	`e1 =` `x^4-200+45*x+x^3-17*x^2`
`e1 = simple(e)`	`e1 =` `x^4-200+45*x+x^3-17*x^2`
`e1 = horner(e)`	**`e1 =`** **`-200+(45+(-17+(x+1)*x)*x)*x`**
`e0 = expand(e1)`	`e0 =` `x^4-200+45*x+x^3-17*x^2`
`e0 = collect(e0,x)`	`e0 =` `x^4-200+45*x+x^3-17*x^2`

`e2 = subs(e,x,'y+z')`	`e2 =` `(y+z)^4-200+45*y+45*z+(y+z)^3-17*(y+z)^2`
`x = y+z`	`x =` `y+z`
`e3 = eval(e)`	`e3 =` `(y+z)^4-200+45*y+45*z+(y+z)^3-17*(y+z)^2`

`disp('--------')`
`e4 = expand(e2)`
	`e4 =`
	`y^4+4*y^3*z+6*y^2*z^2+4*y*z^3+z^4-`
	`200+45*y+45*z+y^3+3*y^2*z+3*y*z^2+z^3-17*y^2-`
	`34*y*z-17*z^2`

`e5 = collect(e4,y)`	`e5 =`
	`y^4+(1+4*z)*y^3+(-17+3*z+6*z^2)*y^2+(45-`
	`34*z+3*z^2+4*z^3)*y+z^4-17*z^2+45*z-200+z^3`

`e6 = collect(e4,z)`	`e6 =`
	`z^4+(4*y+1)*z^3+(6*y^2+3*y-`
	`17)*z^2+(45+4*y^3+3*y^2-34*y)*z+y^4-200+45*y+y^3-`
	`17*y^2`

`e7 = collect(e4)`	`e7 =`
	`y^4+(1+4*z)*y^3+(-17+3*z+6*z^2)*y^2+(45-`
	`34*z+3*z^2+4*z^3)*y+z^4-17*z^2+45*z-200+z^3`

`disp('--------')`
`clear all, syms t`
`e = (1.5-t)^3-(2*t+6)^3+(t^3-`
`2*t)^2+(2*t+8)^7`
	`e =`
	`(3/2-t)^3-(2*t+6)^3+(t^3-2*t)^2+(2*t+8)^7`

`e1 = expand(e)`	**`e1 =`** **`16775515/8+14679173/4*t+5504897/2*t^2+1146871*t^3`** **`+3585*t^6+286716*t^4+128*t^7+43008*t^5`**
`e2 = collect(e1,t)`	***`e2 =`*** ***`16775515/8+14679173/4*t+5504897/2*t^2+1146871*t^3`*** ***`+3585*t^6+286716*t^4+128*t^7+43008*t^5`***

Приклад 5. Варіанти використання символьних операцій математичного аналізу (М-файл `pr11_5.m`).

Оператори програми	**Результат виконання оператора**

`%% pr11_5.m`
`clear all, close all, clc`
`% Варіанти використання символьних`
`% операцій математичного аналізу`
`disp(' diff :')`
`clear all, syms x y`
`f = 1-2*x^3+x^7-123*x^4`
	` diff :`
	
	`f =`
	`1-2*x^3+x^7-123*x^4`

`e1 = diff(f)`	`e1 =` `-6*x^2+7*x^6-492*x^3`
`e1 = diff(f,y)`	`E1 =` `0`
`e2 = diff(f,2)`	`e2 =` `-12*x+42*x^5-1476*x^2`

`disp(' jacobian :')`
`clear all, syms x1 x2 x3`
`f = [x1*x2*x3;...`
`x1^2+2*x2^2-3*x3^2;...`
`x1*exp(x2)+x2^3];`
`J = jacobian(f,[x1,x2,x3])`
	` jacobian :`
	
	`J =`
	`[ x2*x3, x1*x3, x1*x2]`
	`[ 2*x1, 4*x2, -6*x3]`
	`[ exp(x2), x1*exp(x2)+3*x2^2, 0]`

`disp(' int :')`
`clear all, syms x`
`f = (1+x*(1+x*(2+x)))/(1+x*(1+x))/(x^2+1)`
	` int :`
	
	`f =`
	`(1+x*(1+x*(x+2)))/(1+x*(1+x))/(x^2+1)`

`e1 = int(f,x)`	`e1 =`
	`1/2*log(x^2+1)+2/3*3^(1/2)*`
	`atan(1/3*(2*x+1)*3^(1/2))`

`e2 = int(f,x,0,2)`	`e2 =`
	`-1/9*3^(1/2)*pi+1/2*log(5)+`
	`2/3*3^(1/2)*atan(5/3*3^(1/2))`

`e3 = vpa(e2)`	`e3 =`

	1.628856880838967995502413677181 6
fprintf('e4=%.10e\n', eval(e2))	e4=1.6288568808e+000
disp(' limit :') *clear all, syms x t* *e1 = limit(int(cos(t^2),t,0,x),x,0)*	limit : e1 = 0
e2 = limit(cot(x)^2-x^(-2),x,0)	e2 = -2/3
m = sym('m','positive'); *e3 = limit(x^m*log(x),x,0,'right')*	e3 = 0
e4 = limit(x(sqrt(x^2+1)-x),* *x,inf,'left')*	e4 = 1/2
disp(' symsum :') *clear all, syms n* *e1 = symsum(n^-(2),n,1,20)*	symsum : e1 = 17299975731542641/10838475198270720
fprintf('e1=%.10e\n', eval(e1))	e1=1.5961632439e+000
% ейлерова постійна : *e2 = symsum(1/n-log(1+1/n),n,1,inf)*	e2 = eulergamma
disp(' taylor :') *clear all, syms x* *f = sin(x)*	taylor : f = sin(x)
taylor(f)	ans = x-1/6*x^3+1/120*x^5
taylor(f,18,x,pi)	ans = -x+pi+1/6*(x-pi)^3-1/120*(x-pi)^5+1/5040*(x-pi)^7-1/362880*(x-pi)^9+1/39916800*(x-pi)^11-1/6227020800*(x-pi)^13+1/1307674368000*(x-pi)^15-1/355687428096000*(x-pi)^17
taylor(f,8)	ans = x-1/6*x^3+1/120*x^5-1/5040*x^7
taylor(f,pi)	ans = -x+pi+1/6*(x-pi)^3-1/120*(x-pi)^5
e = (x-1)^10	e = (x-1)^10
e1 = expand(e)	e1 = x^10-10*x^9+45*x^8-120*x^7+210*x^6-252*x^5+210*x^4-120*x^3+45*x^2-10*x+1
e2 = taylor(e1,11,x,0)	*e2 =* *1-10*x+45*x^2-120*x^3+210*x^4-252*x^5*
e2 = taylor(e1,11,x)	*e2 =* *1-10*x+45*x^2-120*x^3+210*x^4-252*x^5*
[e2,stat]=maple('taylor',e,'x=0',12); *e2*	**e2 =** **series(1-10*x+45*x^2-120*x^3+210*x^4-252*x^5+210*x^6-120*x^7+45*x^8-10*x^9+1*x^10,x)**
e3 = char(e2)	e3 = series(1-10*x+45*x^2-120*x^3+210*x^4-252*x^5+210*x^6-120*x^7+45*x^8-10*x^9+1*x^10,**x**)
e3 = strrep(e3,',x)',',x,12)')	e3 = series(1-10*x+45*x^2-120*x^3+210*x^4-252*x^5+210*x^6-120*x^7+45*x^8-10*x^9+1*x^10,**x,12**)
e4 = maple('value',e3) *if stat* *error('symbolic:sym:taylor* *:errmsg2',e2)* *end*	*e4 =* *series(1-10*x+45*x^2-120*x^3+210*x^4-252*x^5+210*x^6-120*x^7+45*x^8-10*x^9+1*x^10,x)*
e4 = maple('convert',e2,'string')	*e4 =* *"series(1-10*x+45*x^2-120*x^3+210*x^4-252*x^5+O(x^6),x,6)"*
e5 = maple('convert',e2,'polynom')	*e5 =* *1-10*x+45*x^2-120*x^3+210*x^4-252*x^5*
e6 = maple('convert',e3,'polynom')	**e6 =** **x^10-10*x^9+45*x^8-120*x^7+210*x^6-252*x^5+210*x^4-120*x^3+45*x^2-10*x+1**

Приклад 6. Аналітичні операції з СВ і використання функцій Maple.

В деяких попередніх прикладах ми зустрілись з ситуацією, коли СВ містив поліноми різних степенів (неупорядковано!) і використання операції *collect()* не приводило до «очікуваного» запису результуючого полінома з впорядкуванням по зростанню/спаданню степенів. Аналогічний результат ми отримуємо, використовуючи Maple-функцію *convert* при конвертації СВ в форму *'polynom'*. Зауважимо, що це досить потужна функція з можливостями конвертації СВ в різні форми представлення. Тому, в якості вправи, напишемо власну функцію перетворення *poly_sym* – СВ типу полінома по змінній *v* у впорядкований по зростанню степенів *v* рядковий вираз *poly_str*, відкорегований *poly_sym* і, при необхідності, вектор *P* – MATLAB-поліном. Отже ця функція повинна допускати наступні форми виклику :

```
[poly_str, poly_sym] = sym_to_poly(poly_sym, v);
[poly_str, poly_sym, P] = sym_to_poly(poly_sym, v);
```

Для тестування роботи *sym_to_poly* створимо М-функцію *pr11_6.m* :

```
function pr11_6
clear all, close all, clc
% Аналітичні операції з символьними виразами.
% Використання функцій Maple.
disp(' Тест   №1')
syms x
e = (1.5-x)^3-(2*x+6)^3+(x^3-2*x)^2+(2*x+8)^7-20-x^13
e1 = expand(e)
e1 = simple(e1)
Psym = collect(e1,x)
[Pstr,Psym,P] = sym_to_poly(Psym,x);
put(Pstr, Psym, P);

disp(' Тест   №2')
clear all
syms x
e = sym(1.5)
Psym = expand(e)
[Pstr,Psym,P] = sym_to_poly(Psym,x);
put(Pstr, Psym, P);

disp(' Тест   №3')
clear all
syms x
e = 1.25+x
Psym = expand(e)
[Pstr,Psym,P] = sym_to_poly(Psym,x);
put(Pstr, Psym, P);

disp(' Тест   №4')
clear all
syms x
e = 1.25-x^10+(x-1)^5-x^7
Psym = expand(e)
[Pstr,Psym,P] = sym_to_poly(Psym,x);
put(Pstr, Psym, P);

function put(Pstr, Psym, P)
% друк
    disp('Pstr='); disp(Pstr);
    disp('Psym='); disp(Psym);
```

```
    fprintf('P=\n');
    fprintf(' %.14g',P), fprintf('\n');
```

Текст функції *sym_to_poly* :

```
function [poly_str, poly_sym, varargout] = sym_to_poly(poly_sym, v)
% Перетворення sym-полінома (poly_sym) від sym-змінної (v) в
%              str-поліном (poly_str) строго по зростанню степенів v.
% При nargout=3 отримання і MATLAB-полінома P.

   if nargin == 0
       error('SYM_TO_POLY','Пустий список вхідних параметрів')
   else
       if nargin > 2
           warning('SYM_TO_POLY: Решта (після 2-го) in-параметрів ігноровано')
       end
       % формування Psym sym-поліном по poly_sym
       c = class(poly_sym);
       if all(c(1:3) == 'sym')
           Psym = poly_sym;
       elseif all(c(1:4) == 'char')
           Psym = sym(poly_sym);
       else
           error('SYM_TO_POLY','Недопустимий тип 1-го in-параметра')
       end
       % формування x sym-змінної по v і str-змінної xs
       c = class(v);
       if all(c(1:3) == 'sym')
           x = v; xs = char(x);
       elseif all(c(1:4) == 'char')
           x = sym(v); xs = v;
       else
           error('SYM_TO_POLY','Недопустимий тип 2-го in-параметра')
       end
   end
   Pstr = char(Psym);              % поточне str-значення полінома
   Pstr = strrep(Pstr,' ','');     % видаляємо пропуски
   Psym = sym(Pstr)                % поточне sym-значення полінома

   pat = [xs,'\^','(?<po>\d*)']; % побудова шаблону пошуку
   N = regexp(Pstr,pat,'names'); % степені po для v (po>0)
   k = 1;                        % шукаємо max степінь>5
   for n = N
       k = max(k,str2num(n.po));
   end
   kp = max(6,k+1);
   [Psym, stat] = maple('taylor',Psym,[xs,'=0'],kp);
   if stat
       error('symbolic:sym:taylor:errmsg2',Psym)
   end
   Pstr = char(Psym);
   i = findstr(Pstr,',');
   poly_str = Pstr(8:i-1);
   poly_sym = sym(poly_str);
   if nargout == 3
       P = zeros(1,kp);
       for i = 0 : k
           c = maple('coeff',poly_sym,x,i); % коефіц. при x^i
           P(kp-i) = vpa(c);
       end
       while P(1)==0 & length(P)>1
           P(1) = [];
       end
```

```
        varargout = {P};
    else
        varargout = {};
    end
end
```

Результат роботи функції *pr11_6.m* (з метою економного розташування тексту результату, імена і отримані значення розташовані в одному рядку) :

```
 Тест   №1
e = (3/2-x)^3-(2*x+6)^3+(x^3-2*x)^2+(2*x+8)^7-20-x^13
e1 = 16775355/8+14679173/4*x+5504897/2*x^2+1146871*x^3+3585*x^6+286716*x^4+128*x^7+43008*x^5-x^13
e1 = 16775355/8+14679173/4*x+5504897/2*x^2+1146871*x^3+3585*x^6+286716*x^4+128*x^7+43008*x^5-x^13
Psym = 16775355/8+14679173/4*x+5504897/2*x^2+1146871*x^3+3585*x^6+286716*x^4+128*x^7+43008*x^5-x^13
Psym = 16775355/8+14679173/4*x+5504897/2*x^2+1146871*x^3+3585*x^6+286716*x^4+128*x^7+43008*x^5-x^13

Pstr=16775355/8+14679173/4*x+5504897/2*x^2+1146871*x^3+286716*x^4+43008*x^5+3585*x^6+128*x^7-1*x^13
Psym=16775355/8+14679173/4*x+5504897/2*x^2+1146871*x^3+286716*x^4+43008*x^5+3585*x^6+128*x^7-1*x^13
P= -1 0 0 0 0 0 128 3585 43008 286716 1146871 2752448.5 3669793.25 2096919.375

 Тест   №2
e = 3/2
Psym = 3/2
Psym = 3/2

Pstr=3/2
Psym=3/2
P= 1.5

 Тест   №3
e = 5/4+x
Psym = 5/4+x
Psym = 5/4+x

Pstr=5/4+1*x
Psym=5/4+1*x
P= 1 1.25

 Тест   №4
e = 5/4-x^10+(x-1)^5-x^7
Psym = 1/4-x^10+x^5-5*x^4+10*x^3-10*x^2+5*x-x^7
Psym = 1/4-x^10+x^5-5*x^4+10*x^3-10*x^2+5*x-x^7

Pstr=1/4+5*x-10*x^2+10*x^3-5*x^4+1*x^5-1*x^7-1*x^10
Psym=1/4+5*x-10*x^2+10*x^3-5*x^4+1*x^5-1*x^7-1*x^10
P= -1 0 0 -1 0 1 -5 10 -10 5 0.25
```

Завдання для самостійної роботи

1. Нехай маємо координати п’яти точок на площині. Написати зовнішню М-функцію *tem1101.m* для знаходження *загального рівняння кривої другого порядку* ($\varphi(x,y)=0$), яка проходить через ці точки. Побудувати графік цієї кривої і знайти: *a* - матрицю числових коефіцієнтів; *I, D, A – інваріанти*; $\tilde{A}$ *– семіінваріант*. Для знаходження числових коефіцієнтів написати внутрішню функцію з використанням функції *regexp*.

   Зауваження: Загальне рівняння кривої другого порядку має вид:

$$\varphi(x,y)\equiv a_{11}x^2+2a_{12}xy+a_{22}y^2+2a_{13}x+2a_{23}y+a_{33}=0,$$

   де $a_{ij}=a_{ji}$ елементи квадратної матриці

$$\mathbf{a}=\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

Числові характеристики які використовуються для класифікації кривих другого порядку є наступні інваріанти:

$$I=a_{11}+a_{22};\ \ D=\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix};\ \ A=\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix};\ \ \tilde{A}=\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}+\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix}.$$

2. Нехай маємо координати п’яти точок на площині. Написати зовнішню М-функцію *tem1102.m* для знаходження *загального рівняння кривої другого порядку* ($\varphi(x,y)=0$), яка проходить через ці точки. За допомогою `a` - матриці числових коефіцієнтів, *інваріантів* (`I,D,A`) і *семіінваріанта* ($\tilde{A}$) виконати класифікацію цієї кривої і побудувати її графік. Для знаходження числових коефіцієнтів написати внутрішню функцію з використанням функції `regexp`.
3. Нехай маємо координати п’яти точок на площині. Написати зовнішню М-функцію *tem1103.m* для знаходження *загального рівняння кривої другого порядку* ($\varphi(x,y)=0$), яка проходить через ці точки. Для цього рівняння визначити матрицю числових коефіцієнтів (`a`) і для точки $t\in\varphi(x,y)$ знайти *рівняння дотичної* до кривої (*поляру точки* `t`). Побудувати графік кривої і дотичної. Для знаходження числових коефіцієнтів написати внутрішню функцію з використанням функції `regexp`.

   Зауваження: Рівняння дотичної в точці $t\in\varphi(x,y)$ до кривої другого порядку $\varphi(x,y)=0$ має вид:

$$T(x,y)\equiv\left(\frac{\partial\varphi}{\partial x}\right)\Big|_t(x-t_x)+\left(\frac{\partial\varphi}{\partial y}\right)\Big|_t(y-t_y)=0.$$

4. Нехай маємо координати п’яти точок на площині. Написати зовнішню М-функцію *tem1104.m* для знаходження *загального рівняння кривої другого порядку* ($\varphi(x,y)=0$), яка проходить через ці точки. Для цього рівняння визначити матрицю числових коефіцієнтів (`a`) і для точки $t\in\varphi(x,y)$ знайти *рівняння нормалі* до кривої. Побудувати графік кривої і нормалі. Для знаходження числових коефіцієнтів написати внутрішню функцію з використанням функції `regexp`.

   Зауваження: Рівняння нормалі в точці $t\in\varphi(x,y)$ до кривої другого порядку $\varphi(x,y)=0$ має вид:

$$N(x,y)\equiv\left(\frac{\partial\varphi}{\partial y}\right)\Big|_t(x-t_x)-\left(\frac{\partial\varphi}{\partial x}\right)\Big|_t(y-t_y)=0.$$

5. Маємо `sym`-матрицю **A** (Таблиця 11.1). Визначити її ($n\times m$)-розмірність, побудувати: `sym`-вектори $\vec{\mathbf{b}}=\vec{\mathbf{0}}$ – вектор правих чистин, $\vec{\mathbf{X}}=[x1,x2,...,xm]$ – вектор невідомих і $\mathbf{Ab}=[\mathbf{A}|\vec{\mathbf{b}}]$ – розширену `sym`-матрицю. Для однорідної СЛР (ОСЛР) знайти загальний розв’язок (ЗР), систему фундаментальних векторів (СФВ) і фундаментальну систему розв’язків (ФСР).

   Вимога. Розв’язок виконати за допомогою набору функцій роботи з матрицями.

   Вивести таку інформацію: **Ab**, трапецієподібну форму матриці **A** ОСЛР, перетворену ОСЛР (з підставленими елементами $\vec{\mathbf{X}}$), ЗР ОСЛР, СФВ ОСЛР, ФСР ОСЛР, наприклад :

```
Розширена матриця Ab ОСЛР :
[  2, -1,  1, -5,  0]
[  1,  2, -1,  3,  0]
[  2,  4, -2,  6,  0]

Трапецієподібна форма матриці A ОСЛР :
[    1,    0,  1/5, -7/5]
[    0,    1, -3/5, 11/5]
[    0,    0,    0,    0]

Перетворена ОСЛР :
x1+1/5*x3-7/5*x4 = 0
x2-3/5*x3+11/5*x4 = 0
0 = 0

Загальний розв'язок ОСЛР :
x1=-1/5*t1+7/5*t2
x2=3/5*t1-11/5*t2
x3=t1
x4=t2

Система фундаментальних векторів ОСЛР :
[ -1/5, 3/5, 1, 0 ]
[ 7/5, -11/5, 0, 1 ]

Фундаментальна системи розв’язків ОСЛР :
[ -1/5*t1+7/5*t2, 3/5*t1-11/5*t2,            t1,             t2]
```

Таблиця 11.1

01. $\begin{pmatrix} 2 & -1 & 1 & -5 \\ 1 & 2 & -1 & 3 \\ 1 & -3 & 1 & -6 \end{pmatrix}$	**14.** $\begin{pmatrix} 2 & 4 & -1 & 2 & -1 \\ 1 & -2 & 2 & -1 & 2 \\ 7 & 2 & 4 & 1 & 4 \end{pmatrix}$
02. $\begin{pmatrix} -4 & 2 & -2 & 4 \\ 2 & -1 & 1 & 1 \\ 4 & 2 & 2 & 3 \end{pmatrix}$	**15.** $\begin{pmatrix} 2 & 3 & -1 & -2 \\ 1 & -1 & 1 & -3 \\ 3 & 7 & -5 & -1 \end{pmatrix}$
03. $\begin{pmatrix} 2 & -5 & 3 & 4 & 0 \\ 6 & -3 & -1 & 0 & -2 \\ 2 & 1 & -4 & -2 & 0 \\ 1 & -4 & 2 & 0 & -1 \end{pmatrix}$	**16.** $\begin{pmatrix} 2 & -5 & 0 & 3 & 2 \\ 0 & -3 & -1 & 0 & -4 \\ 2 & 1 & -2 & -2 & 0 \\ -1 & 0 & 2 & -1 & 0 \end{pmatrix}$
04. $\begin{pmatrix} 1 & -2 & 3 & 1 \\ 2 & 1 & 7 & -1 \\ 1 & -7 & 2 & 4 \end{pmatrix}$	**17.** $\begin{pmatrix} -3 & 2 & -2 & 1 \\ 2 & -1 & 1 & -2 \\ 1 & -2 & 1 & 0 \end{pmatrix}$
05. $\begin{pmatrix} 2 & 4 & -1 & 2 & -1 \\ 1 & -2 & 2 & -1 & 2 \\ 7 & 2 & 4 & 1 & 4 \\ 4 & 0 & -4 & -1 & 0 \end{pmatrix}$	**18.** $\begin{pmatrix} 2 & -1 & 1 & 0 & -5 \\ 1 & 1 & -1 & 0 & 3 \\ 1 & 1 & 0 & -1 & 3 \\ 3 & 2 & 1 & -2 & 0 \end{pmatrix}$
06. $\begin{pmatrix} 2 & -5 & 3 & 2 & 0 \\ 6 & -3 & 0 & -1 & -4 \\ 2 & -4 & 0 & 2 & 0 \end{pmatrix}$	**19.** $\begin{pmatrix} 2 & -5 & 3 & 0 & 2 \\ 1 & -3 & -1 & -2 & 0 \\ 2 & 1 & -2 & 0 & -1 \end{pmatrix}$
07. $\begin{pmatrix} 1 & -2 & 3 & 0 & -9 \\ 2 & -4 & -1 & -4 & 0 \\ 0 & 4 & 0 & 1 & -3 \\ 3 & -2 & -4 & 0 & 5 \end{pmatrix}$	**20.** $\begin{pmatrix} 1 & 1 & 0 & 0 & 2 \\ 1 & 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 0 \\ 2 & 2 & 1 & 0 & 1 \end{pmatrix}$

08. $\begin{pmatrix} 2 & 4 & -1 & 2 & -1 \\ 1 & -2 & 2 & -1 & 2 \\ 1 & 2 & 0 & 1 & -2 \\ 0 & 0 & -2 & 0 & -1 \end{pmatrix}$	**21.** $\begin{pmatrix} 4 & -5 & 3 & 2 & 0 \\ 1 & 1 & -1 & 0 & 3 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 2 & 0 \end{pmatrix}$
09. $\begin{pmatrix} 1 & 3 & 0 & 0 & 1 \\ 0 & -2 & 0 & 0 & 2 \\ 2 & 0 & 2 & 1 & 0 \end{pmatrix}$	**22.** $\begin{pmatrix} 2 & 0 & -1 & 0 & -1 \\ 1 & 0 & 2 & -1 & 0 \\ 0 & 2 & 0 & 1 & 0 \end{pmatrix}$
10. $\begin{pmatrix} 2 & 3 & 0 & 1 \\ 1 & 0 & 3 & 2 \\ 0 & 1 & 0 & 1 \end{pmatrix}$	**23.** $\begin{pmatrix} 2 & -5 & 3 & 0 \\ 0 & -3 & 0 & 2 \\ 0 & 1 & 3 & -1 \end{pmatrix}$
11. $\begin{pmatrix} 1 & 0 & 5 & 2 \\ 2 & 3 & 0 & 1 \end{pmatrix}$	**24.** $\begin{pmatrix} 2 & -5 & 3 & 1 & 1 \\ 1 & 1 & -1 & 0 & 2 \end{pmatrix}$
12. $\begin{pmatrix} 0 & -2 & 3 & 1 & 2 \\ 1 & 0 & 1 & -1 & 2 \end{pmatrix}$	**25.** $\begin{pmatrix} 1 & 3 & 3 & 1 \\ 2 & 6 & 0 & 2 \end{pmatrix}$
13. $\begin{pmatrix} 2 & 1 & -2 & 4 \\ 2 & 1 & 1 & 2 \\ 4 & 2 & -1 & 3 \end{pmatrix}$	**26.** $\begin{pmatrix} 1 & 3 & -1 & 1 \\ 3 & 9 & 1 & 3 \\ 3 & 6 & 4 & -1 \end{pmatrix}$

6. Маємо СЛАР, яка задана у вигляді розширеної матриці (Таблиця 11.2) і залежить від параметра t. Написати М-файл, що виконує наступну роботу: дослідження на сумісність, знаходження визначника $d=\det(A(t))$, побудову графіку `d(t)`, знаходження $[k1, k2, k3]$ або $k1$ – дійсних коренів рівняння `d(t)=0`. Для значень $t \in [k1-1, k1, (k1+k2)/2, k2, (k2+k3)/2, k3, k3+1]$ або $t \in [k1-1, k1, k1+1]$ знайти розв'язок СЛАР.

Вказівки. **1.** Щоб знайти корені `d(t)=0` – використовуємо команду `K=solve(d)`.

2. Приклад виведення результатів (графік не показано) :

```
  СЛАР :
[  t, -1, -1,  1]
[ -1,  t,  5, -1]
[ -1,  1,  t,  1]

 СЛАР сумісна

Визначник d= t^3-7*t+6
Дійсні корені det(A(t))=0
[  1,  2, -3]

Розв'язки СЛАР X(t) :
t=0 X = -3/2 -1/2 -1/2
t=1 X = 0 0 1
t=3/2 X = 6 10 -2
t=2 X = 0 0 1
t=-1/2 X = -14/15 -2/15 -2/5
t=-3 X = -1/2 1/2 0
t=-2 X = -5/12 1/12 -1/4
```

Таблиця 11.2

01	$\left(\begin{array}{ccc\|c} t & 1 & 1 & 1 \\ 1 & t & 1 & 1 \\ 1 & 1 & t & 1 \end{array}\right)$	**10**	$\left(\begin{array}{ccc\|c} t & -1 & 2 & 1 \\ 1 & t & 1 & 1 \\ 2 & 1 & t & 1 \end{array}\right)$	**19**	$\left(\begin{array}{ccc\|c} t & 4 & 1 & 1 \\ 4 & t & -2 & 1 \\ 1 & 2 & t & 1 \end{array}\right)$
02	$\left(\begin{array}{ccc\|c} t & 2 & 1 & 1 \\ 2 & t & 1 & 1 \\ 1 & 1 & t & 1 \end{array}\right)$	**11**	$\left(\begin{array}{ccc\|c} t & -1 & 2 & 1 \\ -1 & t & 1 & 1 \\ 2 & 1 & t & 1 \end{array}\right)$	**20**	$\left(\begin{array}{ccc\|c} t & 1 & 1 & 1 \\ 1 & t & -4 & 1 \\ 1 & -4 & t & 1 \end{array}\right)$
03	$\left(\begin{array}{ccc\|c} t & -1 & 2 & 1 \\ -2 & t & 1 & 1 \\ 2 & 1 & t & 1 \end{array}\right)$	**12**	$\left(\begin{array}{ccc\|c} t & -1 & 2 & 1 \\ -2 & t & 3 & 1 \\ 2 & 1 & t & 1 \end{array}\right)$	**21**	$\left(\begin{array}{ccc\|c} t & -4 & 1 & 1 \\ -4 & t & -4 & 1 \\ 1 & -4 & t & 1 \end{array}\right)$
04	$\left(\begin{array}{ccc\|c} t & -1 & 2 & 1 \\ -2 & t & 3 & 1 \\ 2 & 3 & t & 1 \end{array}\right)$	**13**	$\left(\begin{array}{ccc\|c} t & -1 & 2 & 1 \\ -1 & t & 3 & 1 \\ 2 & 2 & t & 1 \end{array}\right)$	**22**	$\left(\begin{array}{ccc\|c} t & 3 & 1 & 1 \\ 3 & t & -1 & 1 \\ 1 & 1 & t & 1 \end{array}\right)$
05	$\left(\begin{array}{ccc\|c} t & 1 & 1 & 1 \\ 1 & t & 3 & 1 \\ 1 & 3 & t & 1 \end{array}\right)$	**14**	$\left(\begin{array}{ccc\|c} t & 1 & 1 & 1 \\ 1 & t & 4 & 1 \\ 1 & 4 & t & 1 \end{array}\right)$	**23**	$\left(\begin{array}{ccc\|c} t & 4 & 1 & 1 \\ 4 & t & -1 & 1 \\ 1 & 1 & t & 1 \end{array}\right)$
06	$\left(\begin{array}{ccc\|c} t & 3 & 1 & 1 \\ 3 & t & 1 & 1 \\ 1 & 1 & t & 1 \end{array}\right)$	**15**	$\left(\begin{array}{ccc\|c} t & -2 & 1 & 1 \\ -2 & t & 1 & 1 \\ 1 & 1 & t & 1 \end{array}\right)$	**24**	$\left(\begin{array}{ccc\|c} t & 4 & 1 & 1 \\ 4 & t & -1 & 1 \\ 1 & 2 & t & 1 \end{array}\right)$
07	$\left(\begin{array}{ccc\|c} t & -4 & 1 & 1 \\ -4 & t & 1 & 1 \\ 1 & 1 & t & 1 \end{array}\right)$	**16**	$\left(\begin{array}{ccc\|c} t & 3 & 1 & 1 \\ 3 & t & -2 & 1 \\ 1 & 2 & t & 1 \end{array}\right)$	**25**	$\left(\begin{array}{ccc\|c} t & -2 & 3 & 1 \\ -2 & t & -1 & 1 \\ 3 & -1 & t & 1 \end{array}\right)$
08	$\left(\begin{array}{ccc\|c} t & -1 & 2 & 1 \\ -2 & -t & 3 & 1 \\ 2 & 3 & t & 1 \end{array}\right)$	**17**	$\left(\begin{array}{ccc\|c} -t & -1 & 2 & 1 \\ -2 & t & 3 & 1 \\ 2 & 3 & t & 1 \end{array}\right)$	**26**	$\left(\begin{array}{ccc\|c} t & -1 & 2 & 1 \\ -2 & t & 3 & 1 \\ 2 & 3 & -t & 1 \end{array}\right)$
09	$\left(\begin{array}{ccc\|c} t & 1 & 1 & 1 \\ 1 & t & 3 & 1 \\ 1 & 3 & t & 1 \end{array}\right)$	**18**	$\left(\begin{array}{ccc\|c} t & 3 & 1 & 1 \\ 3 & t & -1 & 1 \\ 1 & 1 & t & 1 \end{array}\right)$	**27**	$\left(\begin{array}{ccc\|c} t & -1 & 2 & 1 \\ -1 & t & 1 & 1 \\ 2 & 1 & t & 1 \end{array}\right)$

7. Функція $f(...)$ від деяких аргументів задана символьним виразом (Таблиця 11.3), наприклад:

```
f = sym('-5*x^2-8*y^2-14*z^2+2*x-3*z+4*y*x');
```

Аргументи можуть мати довільні позначення (імена), а коефіцієнти в виразі `f` – тільки числові. Написати М-файл, який знаходить (якщо він є!) *безумовний екстремум* `f`.

Опишемо схему послідовності дій для досягнення результату :

a) будуємо `sym`-вектор $\vec{\mathbf{X}} = [X_1, X_2, ..., X_n]$ з імен X_i, які є аргументами в `f`;

b) будуємо вектор-функцію $\vec{\mathbf{F}}(\vec{\mathbf{X}}) = \left(\frac{\partial f}{\partial X_i}\right)_{i=1}^{n}$ і $\mathbf{H}(\vec{\mathbf{X}}) = \left(\frac{\partial^2 f}{\partial X_i \partial X_j}\right)_{i,\,j=1}^{n,\,n}$ – гессіан;

c) використовуючи $\vec{\mathbf{F}}(\vec{\mathbf{X}})$, будуємо `sym`-матрицю **A** і `sym`-вектор $\vec{\mathbf{b}}$, з яких формуємо розширену матрицю $\mathbf{Ab} = [\mathbf{A}, \vec{\mathbf{b}}]$;

d) за допомогою функції `rref` знаходимо `sym`-вектор $\vec{\mathbf{X}}\mathbf{s}$ – *стаціонарні точки* (розв'язок системи $\{F_i(\vec{\mathbf{X}}) = 0,\ i = \overline{1,n}$);

e) використовуючи критерій Сільвестра для додатньо|від'ємно визначеності або невизначеності $\mathbf{H}(\vec{\mathbf{X}})$, детектуємо точку $\vec{\mathbf{X}}\mathbf{s}$ на безумовний екстремум;

f) друкуємо : сам вираз *f*, який екстремум знайшли, значення *fextr* = $f(\vec{\mathbf{X}}\mathbf{s})$, вектор $\vec{\mathbf{X}}\mathbf{s}$.

Наприклад, при наявності екстремуму

```
f = -5*x1^2-8*x2^2-14*x3^2+2*x1-3*x3+4*x2*x1
MAX : fextr=193/504
[   2/9,  1/18, -3/28]
```

```
f = 15*a^2+18*b^2+10*c^2+2*a*c-3*c+4*b
MIN : fextr=-2407/5364
[ -3/298,   -1/9, 45/298]
```

або у випадку відсутності екстремуму

```
f = 5*x^2-8*y^2+2*x-13*x*y
гессіан невизначений !
```

Таблиця 11.3

№	*Вираз f*
01	2*a^2+3*b^2+4*c^2+5*d^2+2*a*c-3*c*d+4*b-5*d-a
02	-3*v^2-5*w^2-8*u^2+5*v*w+3*w*u-2*u+5*v-3*w+5
03	-6*x1^2-6*x2^2-7*x3^2+5*x1+3*x1*x2-5*x1+4*x2-2*x3-4
04	2*x1^2+3*x2^2+3*x3^2+5*x4^2+3*x1*x2-x1*x2-2*x3*x4-7
05	-10*a1^2-a2^2-8*a3^2+4*a3+a1*a3+5*a2*a3-2*a1-7*a2-2*a3
06	-1*q^2-3*w^2-9*g^2+5*g+q*g+5*w*g-5*q-7*w-5*q
07	-10*x^2-8*y^2-7*u^2-5*v^2+x*v+5*y*x-9*x-6*y-8*u+3*v+10
08	-5*b1^2-7*b2^2-3*b3^2+2*b1*b3+6*b1*b2-7*b1-3*b2-2*b3+30
09	(x1^2+x2^2-5)/4+x3^2+(x2-1)*(x2+5)+x3
10	30-7*(a-3)^2-8*(1+b)^2-(c-3/2)^2+8*a*b
11	43+8*(a-4)^2+6*(2+b)^2+3*(c-3)^2-2*a*c
12	53+8*(u1-4)^2+6*(2+u2)^2+3*(5+u3)^2-2*u1*u2+2*u2*u3
13	5-9*(t1-2)^2-8*(1+t2)^2-7*(3+t3)^2+2*t1*t2-3*t2*t3
14	2*x1^2+x2^2-x3^2-3*x1*x2-x4^2-50
15	x1^2-7*x1*x2+x2^2+4*x2+12*x2+5
16	9*p^2-7*p*q+12*q^2+4*q+12*p+33
17	22*(x1-x2)^2+8*(x3-3*x4)^2+3*x1-9*x2+2*x3+150
18	10*w1^2-7*w1*w2+20*w2^2+4*w3^2+12*w2+5*w1-7*w3+13
19	-7*s^2+17*s*d-23*d^2-70*p^2+19*d+8*s-5*p+17
20	41+6*a^2+7*b^2+8*c^2-12*a*b+3*a+5*b+4*c
21	14-8*e^2-9*r^2-7*k^2-10*e*r+2*e+4*r+6*k
22	34-9*g^2-3*v^2-5*b^2+12*g+14*v+16*b
23	34*x^2+x*(2+y)+3*y^2+y*(3+x)+5*z^2+2*x+4*y+6*z+21
24	*c1^2+4*c1*(3+c2)+8*c2^2+15*c3^2+21*c1-6*c2+c3-52
25	15*(2-a)*(3-b)+(2*a+3*b)^2+c^2-23
26	-2*(3-a)*(4-b)+(3*a+2*b)^2+3*c^2-7
27	-2*(3-a*c)-c*(4-b)-(3*a+2*b)^2+8*c^2+27
28	-4+d1*d2-d3*(6-d2)-(5*d1+3*d2)^2-18*d3^2+8
29	9*y1^2+y2^2+y3^2-3*y1*y2+y4^2+4*y5^2+y1+y3+y5-32
30	8*u1^2+4*u2^2+u3^2-8*u1*u2+9*u4^2+7*u5^2+u5+2*u3+u4+5

Практичне заняття № 12

СИМВОЛЬНІ ОБЧИСЛЕННЯ В СИСТЕМІ MATLAB

Аудиторне заняття

Мета заняття :

- ❖ застосувати символьні обчисленя до розв'язку деяких задач обчислювальної математики;
- ❖ навчитись використовувати команду `solve` для символьного розв'язання СЛАР і систем нелінійних рівнянь, а також ознайомитись з командою `fsolve` для їх чисельного розв'язання (у разі неуспішного аналітичного розв'язання);
- ❖ навчитись використовувати команду `dsolve` для символьного розв'язання звичайних диференціальних рівнянь (ЗДР), систем ЗДР, задач Коші і крайових задач для ЗДР/систем ЗДР.

Приклад 1. Використовуючи СВ для абстрактної функції $u(x)$ і оператори :

лівої різницевої похідної $u_{\bar{x},i} = \big(u(x_i) - u(x_i - h)\big)/h$

правої різницевої похідної $u_{x,i} = \big(u(x_i + h) - u(x_i)\big)/h$

знайти :

a) $u_{\bar{x}x,i} = (u_{\bar{x}})_{x,i}$ – *другу різницеву похідну* для наближення $u''(x_i)$;

b) $u_{\bar{x}x\bar{x}x,i} = ((u_{\bar{x}x})_{\bar{x}})_{x,i}$ – *четверту різницеву похідну* для наближення $u^{\mathrm{IV}}(x_i)$;

c) $p = u_{\bar{x}x\bar{x}x,i} - u^{\mathrm{IV}}(x_i)$ – *головний член локальної похибки* наближення.

Створимо наступний М-файл `pr12_1.m` :

```
%% pr12_1.m
clear all, close all, clc
syms x h
z = sym('u(x)');              % абстрактна функція u(x)
ldz = (z - subs(z,x,x-h))/h; % ліва різницева похідна
rdz = (subs(z,x,x+h) - z)/h; % права різницева похідна
rldz = (subs(ldz,x,x+h)-ldz)/h;
rldz = simple(rldz);          % 2-га різницева похідна
disp('2-га РП='); disp(rldz);
% знайдемо праву_різницю(ліву_різницю(2-ї_РП)) :
lrldz = (rldz - subs(rldz,x,x-h))/h;
rlrldz = (subs(lrldz,x,x+h)-lrldz)/h;
rlrldz = simple(rlrldz);      % 4-та різницева похідна
disp('4-та РП='); disp(rlrldz);
p = taylor(rlrldz,12,x,h)-diff(z,4);
p = simple(p); s = char(p);
c = [char(h),'^']; i = 0;
while 1
    hi = [c,int2str(i)];
    k = findstr(s,hi);
    if k
        break
    end
    i = i+1;
end
s = s(1:k+length(hi)-1);
p = sym(s);
disp('головний член похибки='); disp(p);
```

Результат роботи *pr12_1.m* :

```
2-га РП=
(u(x+h)-2*u(x)+u(x-h))/h^2

4-та РП=
(u(x+2*h)-4*u(x+h)+6*u(x)-4*u(x-h)+u(x-2*h))/h^4

головний член похибки=
1/6*@@(D,6)(u)(x)*h^2
```

Приклад 2. Використовуючи КФ *трапецій* для абстрактної функції $u(x)$:

- **a)** побудувати нову КФ за допомогою *формул Рунге-Ромберга*;
- **b)** знайти її *квадратурні коефіцієнти*;
- **c)** дослідити її *чисельну стійкість*.

Створимо наступний М-файл *pr12_2.m* :

```
%% pr12_2.m
clear all, close all, clc
syms x h
z = sym('u(x)');          % абстрактна функція u(x)
a = x-h; b = x+h;         % границі інтегрування
p = 2;                    % порядок точності ф. трапецій
% ф. трапецій на всьому інтервалі з кроком Ho = b-a:
Ho = simplify(b-a);
ua = subs(z,x,a); ub = subs(z,x,b);
TrapHo = Ho/2*(ua + ub);
TrapHo = simple(TrapHo);
% ф. трапецій на всьому інтервалі з кроком Hn = k*Ho:
k = sym(1/2); Hn = k*Ho; c = a+Hn;
uc = subs(z,x,c);
Tac = Hn/2*(ua + uc);    % ф. трапецій на [a,c]
Tcb = Hn/2*(uc + ub);    % ф. трапецій на [c,b]
TrapHn = simple(Tac+Tcb);
clear Tac Tcb Ho Hn c
% Обчислюємо 1-у ф. Рунге-Ромберга
RR1 = simple((TrapHn-TrapHo)/(1-k^p));
clear TrapHn k p
% Обчислюємо 2-у ф. Рунге-Ромберга
RR2 = simple(TrapHo+RR1); clear RR1
NewKF = expand(RR2); clear RR2
NewKF = horner(NewKF);
disp('Нова КФ (це ф.Сімпсона)='); disp(NewKF);
% Визначимо обчислювальну стійкість :
A = [maple('coeff',NewKF,ua,1),...
     maple('coeff',NewKF,uc,1),...
     maple('coeff',NewKF,ub,1)];
clear ua uc ub a b c
A = subs(A,h,'1');
disp('Квадратурні коефіцієнти='); disp(A);
cs = all(eval(A) >= 0);
if cs
    disp('Нова КФ стійка');
else
    disp('Нова КФ нестійка');
end
```

Результат роботи *pr12_2.m* :

```
Нова КФ (це ф. Сімпсона)=
(1/3*u(x-h)+1/3*u(x+h))*h+4/3*h*u(x)

Квадратурні коефіцієнти=
[ 1/3, 4/3, 1/3]
```

Нова КФ стійка

Приклад 3. Розглянемо аналітичне розв'язання : СЛАР, нелінійних рівнянь (НР), систем НР за допомогою функції *solve*, а також ознайомимось з використанням функції *fsolve* для чисельного розв'язання систем НР.

Коротка інформація про *fsolve*. Виклик здійснюється наступним чином.

Для одного рівняння $f(x)=0$:

```
[x, fx, exitf, output] = fsolve(@f, x0, opt);
```

Тут *f* – ім'я функції; *x0* – «скаляр» | вектор-рядок з початковим(и) наближенням(и) до кореня(ів); *opt* – необов'язковий параметр з указанням необхідних опцій.

Для системи n рівнянь $\vec{\mathbf{F}}(x)=[f_1(x); f_2(x);...; f_n(x)]=\vec{\mathbf{0}}$:

```
[X, FX, exitf, output] = fsolve(@F, X0, opt);
```

Тут *F* – ім'я вектор-функції з описом рівнянь системи; *X0* –вектор-рядок з початковим наближенням до кореня (ТІЛЬКИ ДО ОДНОГО !). Отже, за наявності коренів більше ніж один, необхідно викликати *fsolve* для знаходження кожного кореня окремо, при цьому задавати відповідні початкові наближення.

Розглянемо наступні задачі :

a) знайти точний розв'язок СЛАР

$$\begin{pmatrix} 1 & 2 & 3 & -2 \\ 2 & 4 & -1 & -3 \\ 3 & 2 & -1 & 2 \\ -1 & 3 & -1 & 4 \end{pmatrix}\begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \end{pmatrix}=\begin{pmatrix} 20/7 \\ -1/21 \\ 10/21 \\ -205/42 \end{pmatrix};$$

b) знайти розв'язок однорідної СЛАР

$$\begin{pmatrix} 1 & 2 & 3 & -2 \\ 2 & 4 & -1 & -3 \\ 3 & 2 & -1 & 2 \end{pmatrix}\begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix}=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix};$$

c) знайти корені рівняння $x^3+8x^2-64=0$ (скористатись *solver* і *fsolver*);

d) за допомогою функцій *solver*, *fsolver* знайти розв'язок системи НР

$$\begin{cases} 3x+2\cos(y-1)=2, \\ 2y-x^6=4. \end{cases}$$

Створимо наступний М-файл *pr12_3.m* :

```
%% pr12_3.m
clc, warning off

disp('  п.1 Точний розв''язок СЛАР');
clear all, close all
syms x1 x2 x3 x4
% задаємо рівняння СЛАР :
A = [1*x1+2*x2+3*x3-2*x4;...
     2*x1+4*x2-2*x3-3*x4;...
     3*x1+2*x2-1*x3+2*x4;...
    -1*x1+3*x2-1*x3+4*x4];
n = length(A);
disp('Модельний розв''язок :');
X = {'1';'-1/2';'2/3';'-3/7'}; disp(X);
% будуємо вектор правої частини :
x = {x1,x2,x3,x4};
b = subs(A,x,X);
% задаємо СЛАР :
clear Ab, Ab = cell(size(A));
```

```
for i =1:n
    Ab{i} = [char(A(i)),'=',char(b(i))];
end
% розв'язуємо СЛАР :
y = solve(Ab{1},Ab{2},Ab{3},Ab{4},x1,x2,x3,x4);
disp(' СЛАР :'); disp(Ab);
disp('Отримано розв''язок :');
disp([y.x1; y.x2; y.x3; y.x4]);

disp('  п.2 Розв''язок однорідної СЛАР');
clear all, close all
syms x1 x2 x3 x4
% задаємо рівняння СЛАР :
A = [1*x1+2*x2+3*x3-2*x4;...
     2*x1+4*x2-2*x3-3*x4;...
     3*x1+2*x2-1*x3+2*x4];
n = length(A);
% задаємо СЛАР :
clear Ab, Ab = cell(size(A));
for i =1:n
    Ab{i} = [char(A(i)),'=0'];
end
% розв'язуємо СЛАР :
y = solve(Ab{1},Ab{2},Ab{3},x1,x2,x3,x4);
disp(' СЛАР :'); disp(Ab);
disp('Отримано розв''язок :');
disp([y.x1; y.x2; y.x3; y.x4]);

disp('  п.3 Розв''язок кубічного рівняння');
clear all, close all
% задаємо рівняння :
ex = 'x^3+8*x^2-64'; eqv = [ex, '=0'];
disp(' Рівняння :'); disp(eqv);
% розв'язуємо :
y = solve(eqv, 'x');
disp('Корені рівняння :'); disp(y);
% будуємо графік :
ezplot(ex, [-8.5,3]), grid on, pause(4)
X0 = input('? Початкові наближення ([X(1),X(2),...])=\n');
% розв'язуємо :
EqF = @(x)(x.^3+8.*x.^2-64);
opt = optimset('Display','off','TolX',1e-12,'TolFun',1e-12);
[X, FX, exitf, output] = fsolve(EqF, X0, opt);
f1 = '\nКод закінчення=%2d Ітерацій=%3u\n';
fprintf(f1, exitf, output.iterations);
f1 = ' %21.11e';
f2 = 'Корені рівняння  :=';
f3 = 'Значення функцій :=';
fprintf(f2); fprintf(f1, X); fprintf('\n');
fprintf(f3); fprintf(f1,FX); fprintf('\n\n');
output

disp('  п.4 Розв''язок системи нелінійних рівнянь');
disp('      а) з використанням solve');
clear all, close all
syms x y
% задаємо рівняння системи:
ex1 = '3*x-2+2*cos(y-1)'; ex2 = '2*y-x^6-4';
disp(' Рівняння :');
disp([ex1, '=0']); disp([ex2, '=0']);
% будуємо графіки функцій :
d = [-1.5,1.5,0,6];
```

```
f1 = ezplot(ex1, d); hold on
f2 = ezplot(ex2, d); hold off, grid on
set(f1,'color',' red','LineWidth',2);
set(f2,'color','blue','LineWidth',2);
title([ex1,'; ',ex2]);
% розв'язуємо :
[x, y] = solve(ex1, ex2, 'x', 'y');
disp('Корені системи рівнянь :');
n = length(x);
format long
for i = 1:n
    disp(double([x(i), y(i)]));
end
disp('      b) з використанням fsolve');
% задаємо систему рівнянь :
SistF  = @(x)([3.*x(1)+2.*cos(x(2)-1)-2;...
               2.*x(2)-x(1).^6-4]);
X0 = input('? Початкові наближення ([X(1),X(2);...])=\n');
% розв'язуємо :
opt = optimset('Display','off','TolX',1e-12,'TolFun',1e-12);
[n, m] = size(X0);
f1 = '\nКод закінчення=%2d Ітерацій=%3u\n';
f3 = ' %21.11e %21.11e\n';
f2 = ['Корені системи НР:=', f3];
f3 = ['Значення функцій :=', f3];
for i = 1 : n
    [X, FX, exitf, output] = fsolve(SistF, X0(i,:), opt);
    fprintf(f1, exitf, output.iterations);
    fprintf(f2, X); fprintf(f3, FX);
end
```

Результат роботи *pr12_3.m* та отримані графіки (рис. 12.1) :

```
  п.1 Точний розв'язок СЛАР
Модельний розв'язок :
    '1'
    '-1/2'
    '2/3'
    '-3/7'

 СЛАР :
    'x1+2*x2+3*x3-2*x4=20/7'
    '2*x1+4*x2-2*x3-3*x4=-1/21'
    '3*x1+2*x2-x3+2*x4=10/21'
    '-x1+3*x2-x3+4*x4=-205/42'

Отримано розв'язок :
    1
 -1/2
  2/3
 -3/7

  п.2 Розв'язок однорідної СЛАР
 СЛАР :
    'x1+2*x2+3*x3-2*x4=0'
    '2*x1+4*x2-2*x3-3*x4=0'
    '3*x1+2*x2-x3+2*x4=0'

Отримано розв'язок :
  -14*x3
 27/2*x3
      x3
    8*x3

  п.3 Розв'язок кубічного рівняння
 Рівняння :
x^3+8*x^2-64=0
Корені рівняння :
```

```
          -4
 -2+2*5^(1/2)
 -2-2*5^(1/2)

? Початкові наближення ([X(1),X(2),...])=
[-6.5, -4.1, 2.5]

Код закінчення= 1 Ітерацій=  4
Корені рівняння  :=   -6.47213595500e+000   -4.00000000000e+000    2.47213595500e+000
Значення функцій :=    0.00000000000e+000    0.00000000000e+000    1.42108547152e-014

output =
       iterations: 4
        funcCount: 20
        algorithm: 'trust-region dogleg'
    firstorderopt: 8.226456924967418e-013
          message: [1x76 char]

  п.4 Розв'язок системи нелінійних рівнянь
      a) з використанням solve
 Рівняння :
3*x-2+2*cos(y-1)=0
2*y-x^6-4=0
Корені системи рівнянь :
   0.306698607829648   2.000416141312054

      b) з використанням fsolve
? Початкові наближення ([X(1),X(2);...])=
[0.3,2; 1.2,3.5; 1.3,4.4]

Код закінчення= 1 Ітерацій=  3
Корені системи НР:=    3.06698607830e-001    2.00041614131e+000
Значення функцій :=    4.44089209850e-016   -4.44089209850e-016

Код закінчення= 1 Ітерацій=  3
Корені системи НР:=    1.20102469896e+000    3.50065768505e+000
Значення функцій :=    0.00000000000e+000    8.88178419700e-016

Код закінчення= 1 Ітерацій=  4
Корені системи НР:=    1.30284013612e+000    4.44521347168e+000
Значення функцій :=    0.00000000000e+000    8.88178419700e-016
```

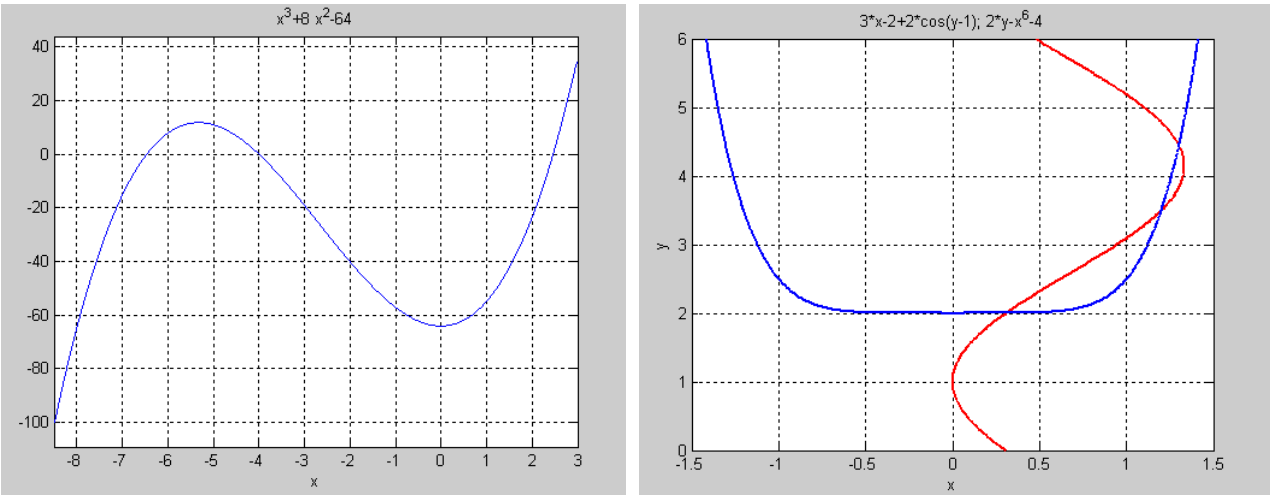


Рис. 12.1. Графіки функції: зліва – до задачі c), справа – до задачі d)

Приклад 4. Розглянемо на простих прикладах, як використовувати команду *dsolve* при символьному розв'язані ЗДР, системи ЗДР, задачі Коші і крайової задачі для ЗДР/системи ЗДР.

Напочатку згадаємо, що загальний синтаксис виклику *dsolve* є наступний :

(*) *res = dsolve('eqv1,eqv2,...', 'cond1,cond2,...', 'v');*

```
res = dsolve('eqv1','eqv2',...,'cond1','cond2',..., 'v');
```

або

(**) `[vr1, vr2,..., vrn] = dsolve(...);`

Вхідні параметри мають наступний зміст :

- `eqvi` – опис і-го диференціального рівняння, наприклад, `(D2y)^2-3*Dy=t^3` – це опис рівняння $(y''(t))^2 - 3y'(t) = t^3$;
- `condj` – опис j-тої початкової | крайової умови, наприклад, `y(0)=1,Dy(0)=0` | `y(0)=2,Dy(3)+2*y(3)=0` – це опис умов $y(0)=1, y'(0)=0$ | $y(0)=2, y'(3)+2y(3)=0$. Частина параметрів `condj` (або усі) можуть бути відсутні, тоді в розв'язку бу-дуть присутні довільні змінні з іменами `C1, C2,…, CK` у кількості $K = n-k$, де n – порядок ЗДР/системи ЗДР, а k – загальна кількість заданих умов.
- `'v'` – опис незалежної змінної, він не є обов'язковий, адже за угодою це `'t'`.

Як бачимо із синтаксиса виклику, параметри `eqvi`, `condj` можуть задаватись одним `char`-літералом і відділятись один від одного знаком коми «`,`», або окремими `char`-літералами. Необхідно пам'ятати, що загальна кількість вхідних параметрів `dsolve` обмежена числом 12.

Вихідний параметр `res` (виклик у формі (*)), в залежності від кількості отриманих розв'язків одного ДР і числа рівнянь системи, є символьний результат у виді:

- «скаляр» або вектор-стовпець знайденого(них) розв'язку(ків) для одного ДР. Декілька розв'язків може бути у випадку, якщо це рівняння нелінійне.
- структура з полями `varm`, `m=1,2,…,n` для системи n ДР. Імена полів структури лексично упорядковані по іменам невідомих функцій `varm`(v) СЗДР.

Якщо ми робимо виклик `dsolve` у формі (**), то `vr1, vr2,…, vrn` це імена змінних для отримання результа розв'язку системи n ДР. Природно ці імена обирати так, щоб вони відповідали іменам шуканих функцій СЗДР, тобто `vrm := varm`, `m=1,2,…,n`, але це вирішує користувач при написанні програми.

Зауваження.

1. MATLAB при виклику `dsolve` звертається до ядра Maple для її виконання. При цьому не всі можливості і потужності функції `dsolve`-Maple доступні. Для повного використання цієї (і інших) функції є можливість безпосередньо звернутись до Maple за таким синтаксисом :

```
maple('name_function', arg1, arg2, …);
```

Наприклад, для `dsolve` (але при цьому потрібно враховувати синтаксис Maple !)

```
% MATLAB при виклику dsolve звертається до ядра Maple :
r = dsolve('Dy=1+sin(t)','y(0)=1/2')
% пряме звертання до dsolve-Maple :
maple('dsolve','{diff(y(t),t)=1+sin(t),y(0)=1/2}','y(t)')
```

2. В Maple є пакет DEtools, який має набір функцій, що збільшує можливості по дослідженню і розв'язанню ЗДР. При завантаженні пакету, можна отримати командою `maple 'with(DEtools)'` список усіх функцій, які доступні користувачу, за виклю-ченням графічних функцій з цього пакету, які не працюють в MATLAB.

Розв'яжемо такі задачі :

a) знайти загальний розв'язок ЗДР $y''' - 2y'' - y' + 2y(t) = 0$;

b) знайти розв'язок задачі Коші для ЗДР

$$\begin{cases} y''' - 2y'' - y' + 2y(t) = 0; \\ y(0) = 1,\ y'(0) = 0,\ y''(0) = 0. \end{cases}$$

c) знайти розв'язок крайової задачі для ЗДР

$$\begin{cases} y''' - 2y'' - y' + 2y(t) = 0; \\ y(0) = 1,\ y'(0) = 0,\ y''(1) - y(1) = 0. \end{cases}$$

d) знайти розв'язок задачі Коші для ЗДР

$$\begin{cases} (u''(x))^2 - u^2/81 = 0; \\ u(0) = 1,\ u'(0) = 0. \end{cases}$$

e) знайти розв'язок задачі Коші для системи ЗДР

$$\begin{cases} x'(t) = x - 4y, \\ y'(t) = y - 9x; \\ x(0) = 1,\ y(0) = 1. \end{cases}$$

f) знайти розв'язок крайової задачі для системи ЗДР

$$\begin{cases} x'(t) = y - z, \\ y'(t) = x - z, \\ z'(t) = x + y; \\ x(0) = 1,\ y(0) = 1,\ z(2\pi/\sqrt{7}) = 0. \end{cases}$$

Створимо наступний М-файл *pr12_4.m* :

```
%% pr12_4.m
% Приклади використання функції DSOLVE.
clc
Eode = 'D3y-2*D2y-Dy+2*y=0';
Ra = dsolve(Eode);
[Ra,HOW] = simple(Ra);
Ra
pretty(Ra)

Cond0 = 'y(0)=1,Dy(0)=0';
Rb= dsolve(Eode,Cond0,'D2y(0)=0');
[Rb,HOW] = simple(Rb);
Rb

Rc = dsolve(Eode,Cond0,'D2y(1)-y(1)=0');
[Rc,HOW] = simple(Rc);
Rc

Rd = dsolve('(D2u)^2-u^2/81=0',...
            'u(0)=1,Du(0)=0','x');
[Rd,HOW] = simple(Rd);
Rd

disp('  e) :');
Cond0 = 'x(0)=1,y(0)=1';
[x y] = dsolve('Dx=x-4*y','Dy=y-9*x',Cond0);
disp('x='); disp(x);
disp('y='); disp(y);

disp('  f) :');
Sode = 'Dx=y-z, Dy=x-z, Dz=x+y';
Cond1 = 'z(2*pi/7^(1/2))=0';
Rf = dsolve(Sode, Cond0, Cond1);
disp('x='); disp(Rf.x);
```

```
disp('y='); disp(Rf.y);
disp('z='); disp(Rf.z);
```

Результат роботи *pr12_4.m* :

```
Ra =
C1*exp(t)+C2/exp(t)+C3*exp(t)^2
                                  C2                2
                       C1 exp(t) + ------ + C3 exp(t)
                                 exp(t)

Rb =
exp(t)-1/3*exp(t)^2+1/3/exp(t)

Rc =
1/2*exp(t)+1/2/exp(t)

Rd =
                      cos(1/3*x)
 1/2*exp(-1/3*x)+1/2*exp(1/3*x)
  e) :
x=
1/6*exp(7*t)+5/6*exp(-5*t)
y=
-1/4*exp(7*t)+5/4*exp(-5*t)
  f) :
x=
1/7*exp(1/2*t)*sin(1/2*7^(1/2)*t)*7^(1/2)+exp(1/2*t)*cos(1/2*7^(1/2)*t)
y=
1/7*exp(1/2*t)*sin(1/2*7^(1/2)*t)*7^(1/2)+exp(1/2*t)*cos(1/2*7^(1/2)*t)
z=
4/7*exp(1/2*t)*sin(1/2*7^(1/2)*t)*7^(1/2)
```

Завдання для самостійної роботи

1. Для символьно заданої функції $f(x1, x2)$ (Таблиця 12.1) побудувати її графік, лінії рівня (діапазон для $x1, x2$ оберіть самостійно). Знайти координати стаціонарних точок (використати функції *diff*, *solve*) і для дійсних значень координат цих точок виконати детектування на наявність екстремуму.

Таблиця 12.1

№	Функція	№	Функція
01	`-exp(sin((x1-2)^2)+cos((x2-4)^2))`	**14**	`2*x1^4+x2^4-x1^2-3*x1*x2-x2^2-50`
02	`x1*x2*(5-x1-x2)`	**15**	`x1^2-3*x1*x2+x2^4`
03	`2*x1^4+3*x2^4-x1^2-3*x2^2`	**16**	`x1^2+x2^3+10-x1-3*x2^2`
04	`2*x1^3+x2^3-10*x1*x2+30`	**17**	`x1^3+x1^2+10*x1*x2^2+x2^3`
05	`x1^4+x2^4-3*x1^2-x2^2+5*x1*x2`	**18**	`x1^3+x2^3-x1+x2^2-x1^2*x2+3`
06	`x1^2+x2^2-(x1+4)*(x2+3)+9`	**19**	`2*x1*x2+10/x1+10/x2+9`
07	`2+sqrt(x1^2+x1^3+x2^3+x2^2)`	**20**	`1+sqrt(x1^3+x2^2+x1*x2+x2+40)`
08	`x1^2/(x2^2+1)+x2^2/(x1^2+1)+10`	**21**	`1/(x2^2+1)+2/(x1^2+1)+10`
09	`(x1^3+x2^3-5)/(x1^2+x2^2+5)`	**22**	`x1^2*(1+x2^2)+x2^2*(1+x1^2)-25`
10	`30-x1^2*(1+x2^2)-x2^2*(1+x1^2)`	**23**	`30-x1^2*(3+x2)-x2^2*(2+x1)`
11	`x1+x2+x1*(3+x2)-x2^2*(2+x1)`	**24**	`x1^2*(3+x2)+x2^2*(2+x1)-x1^3-x2^3`

12	`(x1-2)*(3+x2)/(1+x1^2+x2^2)`	**25**	`(1-x1)*(1-x2)/(1+x1^2+x2^2)`
13	`1/(x1^2+2)+1/(x2^2+2)-10`	**26**	`(x1^2+x2^2)*exp(-x1-x2)`

2. ***** Нехай маємо деяку плоску криву C(*x*,*y*) (Таблиця 12.2) і точку M = [*Mx*,*My*]. Знайти T_k – точки дотику і $L_k(x,y)$ – загальне рівняння дотичних до C, проведених з точки M. Розглянути всі можливі випадки взаємного розташування C і M, передбачити систему тестів для варіантів: 0, 1-а, 2-і дотичних. Побудувати графіки: кривих C(*x*,*y*), дотичних і зобразити точки z, T_k.

   Опишемо схему послідовності дій для розв'язання задачі :

   a) задаємо систему тестів з описом вхідних даних задачі, описуємо криву C(*x*,*y*) і означимо діапазони виведення графіка кривої C за допомогою функції `ezplot`;

   b) позначимо: `t` = [`tx,ty`] – точка дотику прямої L(*x*,*y*) до C, а `tC` = [`x,y`] – довільна точка площини. Для опису шуканого рівняння дотичних, використаємо відому формулу

$$e1 := \left([C'_x(x,y), C'_y(x,y)]\big|_{tC=t}, tC - t \right) = 0 .$$

   c) використовуючи ще одне рівняння $e2 := C(x,y)\big|_{tC=t} = 0$ – належності точки дотику `t` кривій C(*x*,*y*), формуємо систему рівнянь e_1, e_2, і за допомогою `solve`, розв'язуємо систему відносно змінних `tx,ty`. Результатом є **Tx**, **Ty** – деяка система **x**, **y** координат точок `t` (зрозуміло, що **Tx**, **Ty** можуть бути і пустими).

   d) з **Tx**, **Ty** формуємо масив точок **T,** в якому залишаємо тільки дійсні, не співпадаючі між собою точки [`t`];

   e) знаходимо для кожної $t_k \in$ **T** рівняння дотичної $L_k(x,y) := e1\big|_{tC=t_k} = 0$;

   f) виконуємо друк результатів і необхідну візуалізацію (є нюанс з використанням `ezplot()` при побудові прямих, паралельних осям координат!).

Таблиця 12.2

№	*Назва*	*Завдання кривої* C(*x*,*y*)
01	**Коло**	$(x-z(1))^2 + (y-z(2))^2 = r^2$, де z=[$z_x$,$z_y$], r>0
02	**Еліпс**	$\frac{(x-z(1))^2}{a^2} + \frac{(y-z(2))^2}{b^2} = 1$, де z=[$z_x$,$z_y$], a,b>0
03	**Гіпербола**	$\frac{(x-z(1))^2}{a^2} - \frac{(y-z(2))^2}{b^2} = 1$, де z=[$z_x$,$z_y$], a,b>0
04	**Гіпербола**	$-\frac{(x-z(1))^2}{a^2} + \frac{(y-z(2))^2}{b^2} = 1$, де z=[$z_x$,$z_y$], a,b>0
05	**Парабола**	$(y-z(2))^2 = 2*p*(x-z(1))$, де z=[$z_x$,$z_y$], p>0
06	**Парабола**	$(x-z(1))^2 = 2*p*(y-z(2))$, де z=[$z_x$,$z_y$], p>0

3. Знайти розв'язок вказаної задачі для ЗДР (див. Таблиця 12.3). Виконати перевірку, що знайдений розв'язок дійсно задовольняє поставленій задачі. Побудувати графік

розв'язку (у випадку загального розв'язку для усіх присутніх там довільних змінних з іменами `C1,C2,…` підставити значення 1).

Тут ми використовуємо рівняння з книги :

А.М.Самойленко, С.А.Кривошея, Н.А.Перестюк "Дифференциальные уравнения: примеры и задачи", 1989. – 383 с.

Отже, при необхідності, можна перевірити отриманий (загальний) розв'язок за допомогою СКМ MATLAB і розв'язок з вказаного Посібника (номер задачі вказано в колонці «Загальний розв'язок»).

Таблиця 12.3

№	*Загальний розв'язок*	№	*Задача Коші*	№	*Крайова задача*
01	***2.89 1)*** $u''+3u'+2u(x)=0.$	**12**	$\begin{cases} y''+3y'+2y(t)=0, \\ y(0)=1,\ y'(0)=1. \end{cases}$	**23**	$\begin{cases} y''+3y'+2y(t)=0, \\ y(0)=1,\ y'(2)=1/2. \end{cases}$
02	***2.89 2)*** $u''-8u'(x)=0.$	**13**	$\begin{cases} y''-8y'(t)=0, \\ y(0)=1,\ y'(0)=0. \end{cases}$	**24**	$\begin{cases} y''-8y'(t)=0, \\ y'(0)-y(0)=1,\ y(2)=2. \end{cases}$
03	***2.89 3)*** $u''-8u'+16u(x)=0.$	**14**	$\begin{cases} y''-8y'+16y(t)=0, \\ y(0)=1/5,\ y'(0)=1. \end{cases}$	**25**	$\begin{cases} y''-8y'+16y(t)=0, \\ y(0)=1,\ y'(3)+2y(3)=0. \end{cases}$
04	***2.89 4)*** $u''+2u'+9u(x)=0.$	**15**	$\begin{cases} y''+2y'+9y(t)=0, \\ y(0)=0,\ y'(0)=4. \end{cases}$	**26**	$\begin{cases} y''+2y'+9y(t)=0, \\ y'(0)-2y(0)=1,\ y(\pi)=1. \end{cases}$
05	***2.90 1)*** $u'''+8u(x)=0.$	**16**	$\begin{cases} y''+8y(t)=0, \\ y(0)=1,\ y'(0)=0. \end{cases}$	**27**	$\begin{cases} y''+8y(t)=0, \\ y(0)+y'(0)=1,\ y'(\pi/\sqrt{3})=0. \end{cases}$
06	***2.90 2)*** $u'''-8u(x)=0.$	**17**	$\begin{cases} y''-8y(t)=0, \\ y(0)=0,\ y'(0)=2. \end{cases}$	**28**	$\begin{cases} y''-8y(t)=0, \\ y'(0)=1,\ y(\pi/\sqrt{3})=0. \end{cases}$
07	***2.102 1)*** $x^2u''+xu'+4u=0.$	**18**	$\begin{cases} t^2y''+ty'+4y=0, \\ y(1)=1,\ y'(1)=0. \end{cases}$	**29**	$\begin{cases} t^2y''+ty'+4y=0, \\ y(1)+y'(1)=1,\ y'(3)=0. \end{cases}$
08	***2.102 2)*** $x^2u''+xu'-9u=0.$	**19**	$\begin{cases} t^2y''+ty'-9y=0, \\ y(1)=0,\ y'(1)=1. \end{cases}$	**30**	$\begin{cases} t^2y''+ty'-9y=0, \\ y'(1)=1,\ y'(2)=1. \end{cases}$
09	***2.102 3)*** $x^2u''+6xu'+4u=0.$	**20**	$\begin{cases} t^2y''+6ty'+4y=0, \\ y(1)=1,\ y'(1)=1/2. \end{cases}$	**31**	$\begin{cases} t^2y''+6ty'+4y=0, \\ y'(1)-y(1)=0,\ y(3)=1/8. \end{cases}$
10	***2.102 4)*** $x^2u''+5xu'+5u=0.$	**21**	$\begin{cases} t^2y''+5ty'+5y=0, \\ y(1)=1,\ y'(1)=0. \end{cases}$	**32**	$\begin{cases} t^2y''+5ty'+5y=0, \\ y(1)=3,\ y(2)+y'(2)=0. \end{cases}$
11	***2.102 5)*** $x^2u''+3xu'+2u=0.$	**22**	$\begin{cases} t^2y''+3ty'+2y=0, \\ y(1)=0,\ y'(1)=1. \end{cases}$	**33**	$\begin{cases} t^2y''+3ty'+2y=0, \\ y'(1)=1,\ y(2)+y'(2)=1. \end{cases}$

4. ***** Маємо $C_i(z_i;r_i),\ r_i>0,\ i=1,2$ – кола, які задано кординатами центра ($z=[z_x,z_y]$) і радіусом, див. Таблиця 12.4. Знайти:

$L_k(x,y):=a_kx+b_ky+c_k=0,\ k\in\{1,2\}|\varnothing$ – рівняння *зовнішніх дотичних* до цих кіл;

$\mathrm{t1}_k,\mathrm{t2}_k$ – точки дотику (кординати).

Розглянути всі можливі випадки взаємного розташування кіл. Побудувати графіки кіл, дотичних і зобразити z_i, $t1_k, t2_k$. Виконати обов'язкові тести :

0 дотичних : $C_1([0,-1];1), C_2([0,-2];3)$; $C_1([1,1];1), C_2([1,1];4)$.

1-а дотична : $C_1([1,1];1), C_2([3,1];3)$; $C_1([1,3];3), C_2([1,1];1)$; $d = 1/\sqrt{2}, C_1([-d,-d];1), C_2([d,d];1)$.

2-і дотичні : $C_1([1,0];1), C_2([0,-2];3)$; $C_1([-2,-3];1), C_2([0,-1];3)$; $C_1([-2,-3];1), C_2([-1,0];3)$; $C_1([-2,0];2)$, $C_2([0,2];2)$; $C_1([-5,0];4), C_2(4,0];1)$; $C_1([0,-3];2), C_2(0,3];2)$; $C_1([0,-3];4)$, $C_2([0,7];1)$; $C_1([-2,-1];2)$, $C_2([5,4];3)$.

Опишемо схему послідовності дій для розв'язання задачі :

a) визначаємо віддаль між точками z_1, z_2 – це характеризує стан взаємного розташування кіл;

b) записуємо рівняння зовнішніх дотичних у формі $L \equiv a*x+b*y+c = 0$ (з урахуванням *умови нормування* $a^2+b^2 = 1$)

c) за допомогою L записуємо рівняння e_1, e_2 віддалей d_1, d_2 від L до центрів кіл C_1, C_2;

d) для знаходження значень a, b будуємо систему рівнянь, яка складається з різниці e_1-e_2 і умови нормування. Для розв'язання системи використовуємо `solve` і отримуємо **Ω** – масив можливих розв'язків $\{a_k, b_k\}$;

e) в масиві **Ω** залишаємо тільки дійсні, не співпадаючі між собою розв'язки;

f) використовуючи елементи **Ω**, а також рівняння e_1, e_2, знаходимо коефіцієнти c_1, c_2 і таким чином повністю визначаємо рівняння прямих $L_j(x, y)$;

g) знаходимо точки дотику прямих $L_j(x, y)$ до кіл C_1, C_2;

h) виконуємо друк результатів і необхідну візуалізацію (є нюанс з використанням `ezplot()` при побудові прямих, паралельних осям координат!).

Таблиця 12.4

№	*Функція*	№	*Функція*
01	$C_1([2,2];6)$, $C_2([5,4];3)$	**11**	$C_1([2,1];5)$, $C_2([5,4];3)$
02	$C_1([-2,-1];1)$, $C_2([5,4];5)$	**12**	$C_1([-2,1];5)$, $C_2([5,-1];2)$
03	$C_1([0,0];2)$, $C_2([4,3];3)$	**13**	$C_1([1,2];1)$, $C_2([-5,-4];5)$
04	$C_1([-2,0];2)$, $C_2([-5,4];2)$	**14**	$C_1([3,-4];6)$, $C_2([1,2];2)$
05	$C_1([-2,1];1)$, $C_2([5,4];3)$	**15**	$C_1([2,-1];4)$, $C_2([-1,4];3)$
06	$C_1([0,-1];2)$, $C_2([-1,4];4)$	**16**	$C_1([2,0];3)$, $C_2([4,2];3)$
07	$C_1([-2,-1];1)$, $C_2([1,2];2)$	**17**	$C_1([2,0];1)$, $C_2([4,2];6)$
08	$C_1([3,4];5)$, $C_2([1,5];2)$	**18**	$C_1([0,1];2)$, $C_2([-2,-4];5)$
09	$C_1([-2,-1];4)$, $C_2([5,4];1)$	**19**	$C_1([2,1];2)$, $C_2([2,4];5)$
10	$C_1([2,3];4)$, $C_2([-5,4];4)$	**20**	$C_1([2,1];2)$, $C_2([4,2];4)$

5. ****** Маємо кола $C_i(z_i; r_i)$, $r_i > 0$, $i = 1,2$, які задано кординатами центра ($z = [z_x, z_y]$) і радіусом, див. Таблиця 12.4. Знайти :

$L_k(x, y) := a_k x + b_k y + c_k = 0$, $k \in \{1,2\} | \{1\} | \emptyset$ – рівняння *внутрішніх дотичних* до C_i ;

$t1_k, t2_k$ – точки дотику (кординати).

Розглянути всі можливі випадки взаємного розташування кіл. Побудувати графіки кіл, дотичних і зобразити z_i, $t1_k, t2_k$. Виконати обов'язкові тести :

0 дотичних : $C_1([0,-1];1), C_2([0,-2];3)$; $C_1([1,1];1), C_2([1,1];4)$; $C_1([1,0];1), C_2([0,-2];3)$;
$C_1([-2,-3];1), C_2([0,-1];3)$; $C_1([-2,-3];1), C_2([-1,0];3)$.

1-а дотична : $C_1([-2,0];2), C_2([0,2];2)$; $C_1([0,1];2), C_2([0,4];2)$; $d=\sqrt{8}, C_1([0,0];1), C_2([d,d];3)$.

2-і дотичні : $C_1([0,-3];2), C_2([0,3];2)$; $C_1([-3,0];2), C_2([3,0];2)$; $C_1([-2,-1];2), C_2([5,4];3)$;
$C_1([-5,0];4), C_2([4,0];1)$; $C_1([-5,0];1), C_2(4,0];4)$; $C_1([0,-5];4), C_2(0,4];1)$;
$C_1([0,-5];1), C_2(0,4];4)$; $C_1([-2,-5];3), C_2([5,2];3)$.

Опишемо схему послідовності дій для розв'язання задачі :

a) визначаємо віддаль між точками z_1, z_2 – це характеризує стан взаємного розташування кіл;

b) виокремлюємо випадок однієї дотичної;

c) вводимо символьні позначення для точок дотику t_1, t_2 (з їх координатами);

d) для векторів $\vec{u}=t_2-t_1$, $\vec{v}=t_1-z_1$, $\vec{w}=t_2-z_2$ записуємо систему рівняннь

$$\begin{cases} e1:=\vec{v}\perp\vec{u}=0, \\ e2:=\vec{w}\perp\vec{u}=0, \\ e3:=|\vec{v}|^2-r_1^2=0, \\ e4:=|\vec{w}|^2-r_2^2=0 \end{cases}$$

розв'язуємо її (відносно координат t_1, t_2) і знаходимо $\mathbf{\Omega}_1$ = [t_1], $\mathbf{\Omega}_2$ = [t_2] – масиви координат можливих точок дотику t_1, t_2 до кіл C_1, C_2. Зауважимо, що в $\mathbf{\Omega}_{1,2}$ будуть координати точок не тільки внутрішніх дотичних, а і зовнішніх дотичних;

e) в масивах $\mathbf{\Omega}_1$, $\mathbf{\Omega}_2$ залишаємо тільки дійсні, не співпадаючі між собою розв'язки;

f) знаходимо рівняння дотичної $L_{[k\,;\,i]}(x,y)$ виду $a*x+b*y+c=0$ для пари точок [$(t_1)_k$, $(t_2)_i$], де $(t_1)_k \in \mathbf{\Omega}_1$, а $(t_2)_i \in \mathbf{\Omega}_2$. Враховуючи, що ми «не бачимо» чи є ця точка (з індексом $k|i$) точкою внутрішньої чи зовнішньої дотичної, то необхідні додаткові перевірки при їх виборі з $\mathbf{\Omega}_1$, $\mathbf{\Omega}_2$. Означимо умови детектування внутрішніх дотичних $L_j(x,y) \equiv L_{[k\,;\,i]}(x,y)$. Позначимо $\vec{v}$, $\vec{w}$ – радіус-вектори, які виходять з центрів кіл C_1, C_2 до вибраних з $\mathbf{\Omega}_1$, $\mathbf{\Omega}_2$ (поточних) точок дотику $(t_1)_k$, $(t_2)_i$. Тоді маємо такі умови :

1°. (k1 = k2), де k1, k2 – кутові коефіцієнти $\vec{v}$, $\vec{w}$;

2°. протилежність напрямків $\vec{v}$, $\vec{w}$;

3°. $|\vec{v}|=r_1 \,\&\, |\vec{w}|=r_2$.

В масивах $\mathbf{T}_{1,2}$ – «відібраних» точок, необхідно запам'ятовувати як кординати точок $t_{1,2}$, так і номер $j \Leftrightarrow [k,i]$ – «прив'язку» до рівняння прямої $L_j(x,y)$ з [$L_j(x,y)$] – це масив знайдених рівнянь дотичних.

g) Можливі ситуації ($r_1 = r_2$ і певний набір індексів $k|i$), коли в [$L_j(x,y)$] попадають і рівняння зовнішніх дотичних, які паралельні. Тому необхідно провести ще одне «відбракування» прямих з масиву [$L_j(x,y)$], згідно умови $(\neg(L_j(x,y) \parallel L_k(x,y)) \,\&\, j \neq k)$ – це непаралельність внутрішніх дотичних. Якщо така пряма $L_k(x,y)$ існує, то її необхідно видалити з масива [$L_j(x,y)$], при цьому синхронно видаляємо із масива $\mathbf{T}_{1,2}$ і точки $t_{1,2}$ з номерами $[k,i] \Leftrightarrow k$.

h) виконуємо друк результатів (елементи масивів $[L_j(x,y)]$, $\mathbf{T}_{1,2}$) і необхідну візуалізацію (є нюанс з використанням `ezplot()` при побудові прямих, які паралельні осям координат!).

6. ***** Маємо коло $C(z;r)$, визначене $z=[z_x,z_y]$ – кординатами центра і радіусом $r>0$, а також точку P. Написати програму візуалізації *графічної побудови* (змоделю-вати креслярську роботу з циркулем і лінійкою) побудови дотичних, проведених з точки P до кола $C(z;r)$.

7. ****** Маємо два кола $C_i(z_i;r_i)$, $i=1,2$, де $z=[z_x,z_y]$ – кординати центра, а $r>0$ – радіус. Написати програму візуалізації *графічної побудови* (змоделювати креслярську роботу з циркулем і лінійкою) заданих кіл та *зовнішніх дотичних* до кіл.

8. ****** Маємо два кола $C_i(z_i;r_i)$, $i=1,2$, де $z=[z_x,z_y]$ – кординати центра, а $r>0$ – радіус. Написати програму візуалізації *графічної побудови* (змоделювати креслярську роботу з циркулем і лінійкою) заданих кіл та *внутрішніх дотичних* до кіл .

Зауваження. Опис алгоритмів *графічної побудови* для задач пп.6, 7, 8 і рисунки побудови Рис.12.2, 12.3 – див. наприклад, за наступним посиланням:
https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D1%82%D0%B8%D1%87%D0%BD%D0%B0_%D0%BF%D1%80%D1%8F%D0%BC%D0%B0_%D0%B4%D0%BE_%D0%BA%D0%BE%D0%BB%D0%B0#%D0%94%D0%BE%D1%82%D0%B8%D1%87%D0%BD%D1%96_%D0%BF%D1%80%D1%8F%D0%BC%D1%96_%D0%B4%D0%BE_%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%BB%D0%B0

Результуючий рисунок графічної побудови для задачі п.6 має бути таким (позначення точок на програмних рисунках можна не відображати) :

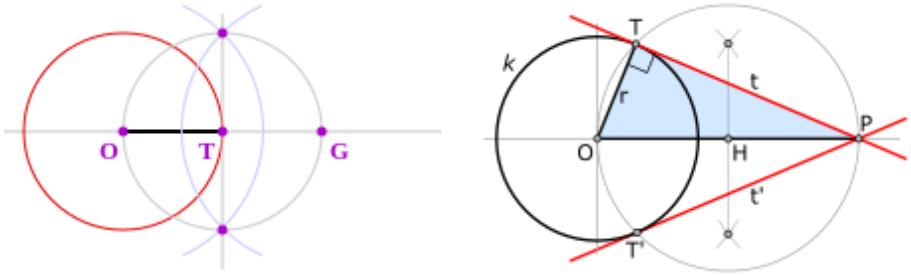


Рис. 12.2. Графіки побудови: зліва – для P = T (P ∈ C), справа – для P ∉ C

Аналогічно для задач п.7, п.8 маємо :

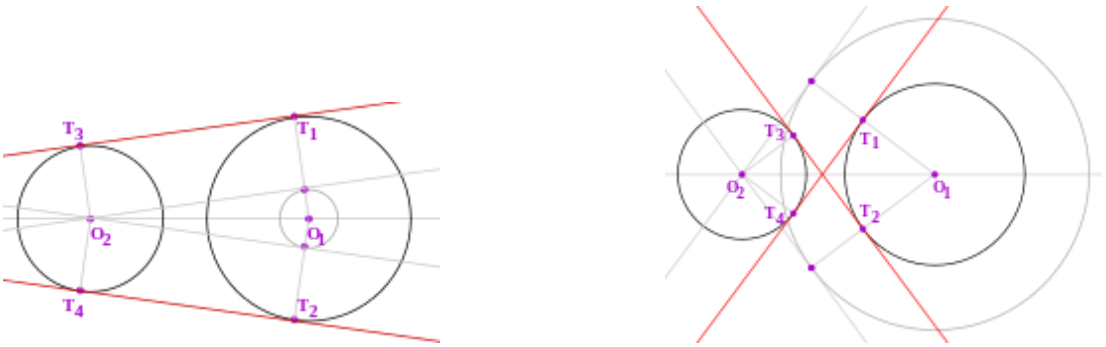


Рис. 12.3. Графіки побудови: зліва – зовнішні дотичі, справа – внутрішні дотичі

СПИСОК ЛІТЕРАТУРИ

1. *Вакал Є. С.* Класифікація рівнянь із частинними похідними з використанням системи MATLAB / Є. С. Вакал, Ю. Є. Вакал. – К.: Основа, 2017. – 104 с.
2. *Вакал Є.С.* Методи математичної фізики в прикладах і задачах : навч. посіб. для студ. мех.-мат. ф-ту / Є. С. Вакал, А. В. Ловейкін. – К.: Вид. Кравченко Я.О., 2020. – 188 с.
3. Використання математичного пакета MATLAB для розв'язування прикладних задач : навч. посіб. / Б.П. Довгий, Є.С. Вакал, Ю.Є. Вакал, А.В. Попов. – К.: Фітосоціоцентр, 2012. – 78 с.
4. Використання системи комп'ютерної математики MATLAB для розв'язування прикладних задач: навч. посіб. / Б.П. Довгий, Є.С. Вакал, Ю.Є. Вакал, А.В. Попов. – К.: ВПЦ "Київський університет", 2016. – 143 с.
5. *Гаврилюк І. П.* Методи обчислень / І. П. Гаврилюк, В. Л. Макаров. – К. : Вища шк., 1995. – 367 с.
6. *Гап'як І.В.* Методичні вказівки до застосування математичних пакетів для чисельно-аналітичного розв'язування задач: для студ. мех.-мат. ф-ту [Електронний ресурс]/ І.В. Гап'як, Б.П. Довгий, Є.С. Вакал, А.В. Ловейкін. – Режим доступа: http://www.mechmat.univ.kiev.ua/wp-content/uploads /2021/04 /hapiak_dovhyj_vakal_lovejkin_metodychni-vkazivky.pdf.– К., 2021. – 121 с.
7. *Довгий Б.П.* Сплайн-функції та їхнє застосування : навч. посіб. / Б. П. Довгий, А. В. Ловейкін, Є. С. Вакал, Ю. Є. Вакал. – К.: ВПЦ "Київський університет", 2017. – 120 с.
8. *Довгий Б.П.* Методи обчислень : методичні вказівки до лабораторних робіт із використанням пакета MATLAB / Б. П. Довгий, Є. С. Вакал, Ю. Є. Вакал. – К.: ВПЦ "Київський університет", 2017. – 60 с.
9. *Довгий Б.П.* Інформаційні системи та технології : навч. посіб. для студ. мех. мат. ф-ту / Б.П. Довгий, Є.С. Вакал. – К.: Вид. Кравченко Я А., 2021. – 111 с.
10. *Довгий Б.П.* Інформаційні системи та технології. Збірник задач для студ. мех.-мат. ф-ту [Елетронний ресурс] / Б.П.Довгий, Є.С.Вакал. Режим доступа : http://www.mechmat.univ.kiev.ua/wp-content/uploads/ 2021/04/ dovhyj_ vakal _informatsijni-sysnemy-ta-tekhnolohii_zb_zadach.pdf.
К., 2001. – 58 с.
11. *Перестюк М. О.* Збірник задач з математичної фізики / М. О. Перестюк, В. В. Маринець, В. Л. Рего. – Кам'янець-Подільський: Аксіома, 2012. – 252 с.
12. *Попов В. В.* Методи обчислень : конспект лекцій для студентів механіко-математичного факультету / В. В. Попов. – К.: ВПЦ "Київський університет", 2012. – 303 с.
13. *Попов В. В.* Лабораторні роботи та домашні завдання для самостійної роботи з дисципліни "Методи обчислень" для студ. мех.-мат. ф-ту / В. В. Попов, Б. П. Довгий, Є. С. Вакал. – К.: Укр. фітосоціолог. центр, 2006. – 32 с.
14. *Bouchaib, R.* Advanced Numerical Methods with Matlab 1: Function Approximation and System Resolution / R. Bouchaib, E.H. Abdelkhalak. – Wiley. John Wiley & Sons, LTD, 2018. – 240 p.
15. *Bober, W.* MATLAB® Essentials: A First Course for Engineers and Scientists / W. Bober. – Taylor & Francis, 2017. – 259 p.

16. *Eaton, J.W.* GNU Octave Manual Version 3 / J.W. Eaton, D. Bateman, S. Hauberg. Network Theory Ltd, 2008. – 568 p.
17. *Greenbaum, A.* Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms / A. Greenbaum, T.P.Chartier. – Princeton University Press, 2012. – 464 p.
18. *Kattan, P.* MATLAB for Beginners: A Gentle Approach / P.Kattan. – Ottawa, Ontario: Petra Books, 2010. – 288 p.
19. *Lindfield, G.* Numerical Methods Using MATLAB / G. Lindfield, J. Penny. – Academic Press, 2018. – 608 p.
20. *Samarskii, A.A.* Blow-up in quasilinear parabolic equations / A. A. Samarskii, V.A. Galactionov, S.P. Kurdyumov, A.P. Mikhailov. – Walter de Gruyte Berlin, NY, 1995. – 534 p.
21. *Samarskii, A.A.* The theory of difference schemes / A.A. Samarskii. – New York – Basel. Marcel Dekker, Inc, 2001. – 761 p.
22. *Schoenberg, I.J.* Cardinal spline interpolation. / I.J. Schoenberg. – Philadelphia: PA: Society of Industrial and Applied Mathematics, 1973. – P. 221-244.
23. *Siauw, T.* An Introduction to MATLAB® Programming and Numerical Methods for Engineers /T.Siauw, A.M.Bayen. – Elsevier, 2014. – 340 p.
24. *Valentine, D.T.* Essential MATLAB for Engineers and Scientists / D.T. Valentine, B.D. Hahn. – Elsevier, 2022. – 432 p.

ЗМІСТ

ПЕРЕДМОВА....4

ЛЕКЦІЯ №01
ОСНОВНІ ОБ'ЄКТИ СИСТЕМИ MATLAB....5

ЛЕКЦІЯ №02
ОСНОВНІ СТРУКТУРИ КЕРУВАННЯ АЛГОРИТМІЧНОЇ МОВИ MATLAB....13

ЛЕКЦІЯ №03
РОБОТА З МАТРИЦЯМИ, МАТРИЧНИЙ АНАЛІЗ....19

ЛЕКЦІЯ №04
4.1. ПОЛІНОМИ ТА ДІЇ НАД НИМИ....34
4.2. АНАЛІЗ ДАНИХ....34
4.3. ЗАДАЧІ ЧИСЛОВОГО АНАЛІЗУ....35

ЛЕКЦІЯ №05
5.1. НАБЛИЖЕННЯ ФУНКЦІЙ ЛІНІЙНИМИ І КУБІЧНИМИ СПЛАЙНАМИ. 42
5.2. МІНІМІЗАЦІЯ ФУНКЦІЙ І РОЗВЯЗАННЯ СИСТЕМ НЕЛІНІЙНИХ РІВНЯНЬ....47

ЛЕКЦІЯ №06
М-ФУНКЦІЇ ТА ЇХ ВИКОРИСТАННЯ....52

ЛЕКЦІЯ №07
7.1. ПОБУДОВА 2–D, 3–D ГРАФІКІВ....61
7.2. РОБОТА З ФАЙЛАМИ....65

ЛЕКЦІЯ №08
РОЗВ'ЯЗАННЯ ЗАДАЧІ КОШІ ДЛЯ ЗДР....71

ЛЕКЦІЯ №09
ВИКОРИСТАННЯ СОЛВЕРІВ MATLAB ДЛЯ РОЗВ'ЯЗАННЯ КРАЙОВИХ ЗАДАЧ ДЛЯ ЗДР....79

ЛЕКЦІЯ №10
ВИКОРИСТАННЯ СОЛВЕРІВ MATLAB ДЛЯ РОЗВ'ЯЗАННЯ ОДНОВИМІРНИХ КРАЙОВИХ ЗАДАЧ ДЛЯ СИСТЕМ ПАРАБОЛІЧНИХ І ЕЛІПТИЧНИХ ДРЧП....85

ЛЕКЦІЯ №11
СИМВОЛЬНІ ОБЧИСЛЕННЯ В СИСТЕМІ MATLAB.....101

ЛЕКЦІЯ №12
СИМВОЛЬНІ ОБЧИСЛЕННЯ В СИСТЕМІ MATLAB (продовження).....108

Практичне заняття № 01
1.1. ВСТУП ДО СИСТЕМИ MATLAB.....115
1.2. ОПЕРАТИВНЕ СЕРЕДОВИЩЕ MATLAB І ОСНОВНІ КОМАНДИ.....117

Практичне заняття № 02
ВИКОРИСТАННЯ ОСНОВНИХ СТРУКТУР КЕРУВАННЯ
МОВИ MATLAB ДЛЯ НАПИСАННЯ М-ФАЙЛІВ.....122

Практичне заняття № 03
АЛГОРИТМИ РОЗКЛАДУ МАТРИЦЬ. ОБУМОВЛЕНІСТЬ МАТРИЦЬ.
ПРОБЛЕМА ВЛАСНИХ ЗНАЧЕНЬ.....126

Практичне заняття № 04
РОБОТА З МАТРИЦЯМИ, МАТРИЧНИЙ АНАЛІЗ 134

Практичне заняття № 05
НАБЛИЖЕННЯ ФУНКЦІЙ. ЗАДАЧІ ЧИСЛОВОГО АНАЛІЗУ 139

Практичне заняття № 06
ВИКОРИСТАННЯ М-ФУНКЦІЙ..... 149

Практичне заняття № 07
ПОБУДОВА ГРАФІКІВ У MATLAB.....158

Практичне заняття № 08
АЛГОРИТМИ РОЗВ'ЯЗАННЯ ЗАДАЧІ КОШІ ДЛЯ ЗДР.
ПОБУДОВА ГРАФІКІВ.....163

Практичне заняття № 09
ВИКОРИСТАННЯ СОЛВЕРІВ MATLAB ДЛЯ РОЗВ'ЯЗАННЯ
КРАЙОВИХ ЗАДАЧ ДЛЯ ЗДР.....175

Практичне заняття № 10
ВИКОРИСТАННЯ СОЛВЕРІВ MATLAB ДЛЯ РОЗВ'ЯЗАННЯ ОДНОВИМІРНИХ
КРАЙОВИХ ЗАДАЧ ДЛЯ
СИСТЕМ ПАРАБОЛІЧНИХ І ЕЛІПТИЧНИХ ДРЧП.....183

Практичне заняття № 11
СИМВОЛЬНІ ОБЧИСЛЕННЯ В СИСТЕМІ MATLAB.....192

Практичне заняття № 12
СИМВОЛЬНІ ОБЧИСЛЕННЯ В СИСТЕМІ MATLAB.....205

СПИСОК ЛІТЕРАТУРИ 220

Навчальне видання

ДОВГИЙ Борис Павлович
ВАКАЛ Євген Сергійович
ЛОВЕЙКІН Андрій Вячеславович

ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ

Конспект лекцій та практичні заняття

Навчальний посібник

для студентів механіко-математичного-факультету