ЖДТУ	Міністерство освіти і науки України Житомирський державний технологічний університет

Затверджено науково-методичною
радою ЖДТУ
протокол від «__»_____ 20__ р. №__

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

для проведення лабораторних робіт
з навчальної дисципліни
«ЧИСЛОВІ МЕТОДИ»

для студентів освітнього рівня «БАКАЛАВР»
денної форми навчання
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
освітньо-професійна програма
«Автоматизація та комп'ютерно-інтегровані технології»
факультет інформаційно-комп'ютерних технологій
кафедра автоматизації та комп'ютерно-інтегрованих технологій
ім. проф. Б.Б. Самотокіна

Розглянуто і рекомендовано
на засіданні кафедри автоматизації та
комп'ютерно-інтегрованих технологій
ім. проф. Б.Б. Самотокіна
протокол від «___» _____ 201__ р.
№ ____

Розробник: к.т.н., доц. кафедри автоматизації та комп'ютерно-інтегрованих технологій ім. проф. Б.Б. Самотокіна Коваль А.В.

Житомир
2017 – 2018 н.р.

ЖДТУ	Міністерство освіти і науки України Житомирський державний технологічний університет

ЗМІСТ

Лабораторна робота № 1

Вивчення математичного пакету MATLAB

Мета роботи

Оптимати практичні навики використання системи для чисельного та символьного аналізу MATLAB.

Короткі теоретичні відомості до роботи

Інтерфейс MATLAB

MATLAB — це математичний пакет прикладних програм, заснований на використанні матриць. Пакет містить велику бібліотеку програм по чисельних методах, використовує двох- і тривимірну графіку, а також формати мов високого рівня. Підходящий формат дозволяє за допомогою MATLAB швидко виконувати й модифікувати програми всім, хто вивчає цю книгу й реалізує наведені в ній алгоритми.

MATLAB (для прикладу використана версія 7.9 (R2009b)) звичайно запускається з головного меню операційної системи Windows або активізацією ярлика з логотипом системи () на робочому столі Windows. Після запуску MATLAB на екрані з'являється основне вікно системи MATLAB, показане на рис. 1. Воно має звичайні для Windows засоби керування розмірами, приховання й закриття. Зазвичай в MATLAB використовується декілька вікон, кожне з яких можна налаштувати або закрити.

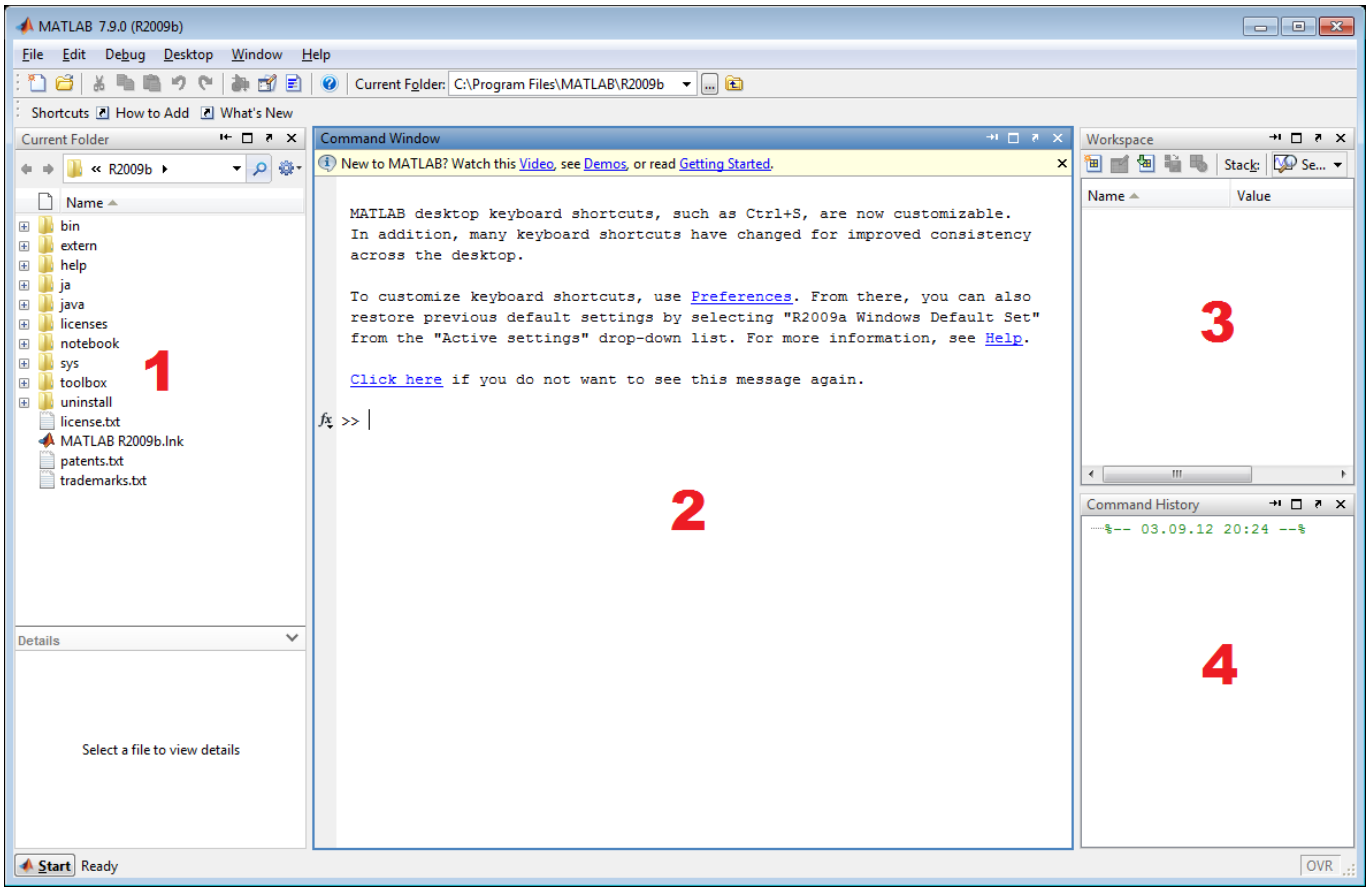


Рис. 1. Вікно системи MATLAB R2009b після запуску

У лівій частині загального вікна MATLAB показане **Вікно 1** (**Current Folder** — Поточна Папка), яке подібне до Windows Explorer. З нього користувач може доступатися до файлів в вибраній папці.

Вікно 2 (**Command Window** — Командне Вікно) є головним вікном, в якому вводяться команди MATLAB і куди виводяться результати обчислень. Командне вікно містить **позначення >>**, після якого можна вводити команди користувача. Для виконання введеної команди необхідно натиснути Enter. Для швидкого

введення команд можна використати відповідну функцію Командного вікна MATLAB, натиснувши на позначку ***fx*** на лівому полі вікна (рис. 2).

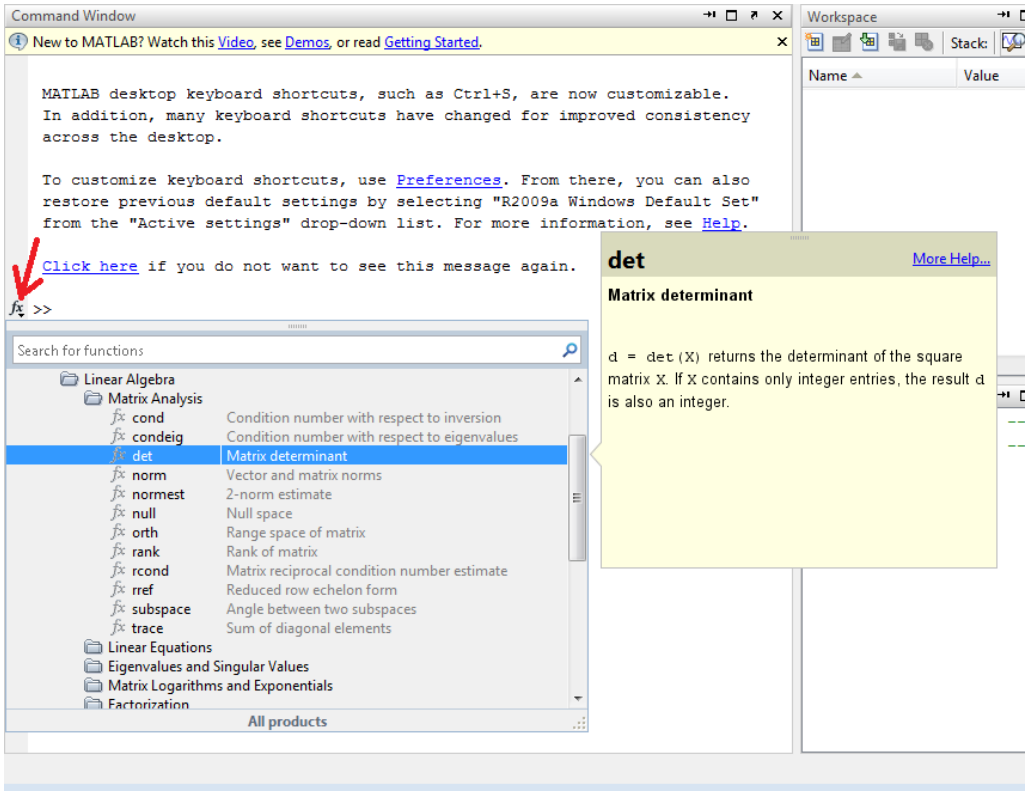


Рис. 2. Швидкий доступ до команд MATLAB

Вікно 3 Workspace (Робоча Область) відображає всі змінні визначені користувачем в процесі роботи.

Під ними розташоване **Вікно 4** (**Command History** — Історія Команд), що містить список виконаних команд. Клацнувши мишею по будь-якій команді і цьому вікні, її можна перенести в поточний рядок (>>) Командного вікна MATLAB.

Якщо якесь з необхідних вікон "зникло", його можна відкрити знову з меню MATLAB. На рис. 3 показані відповідні пункти меню **Desktop**.

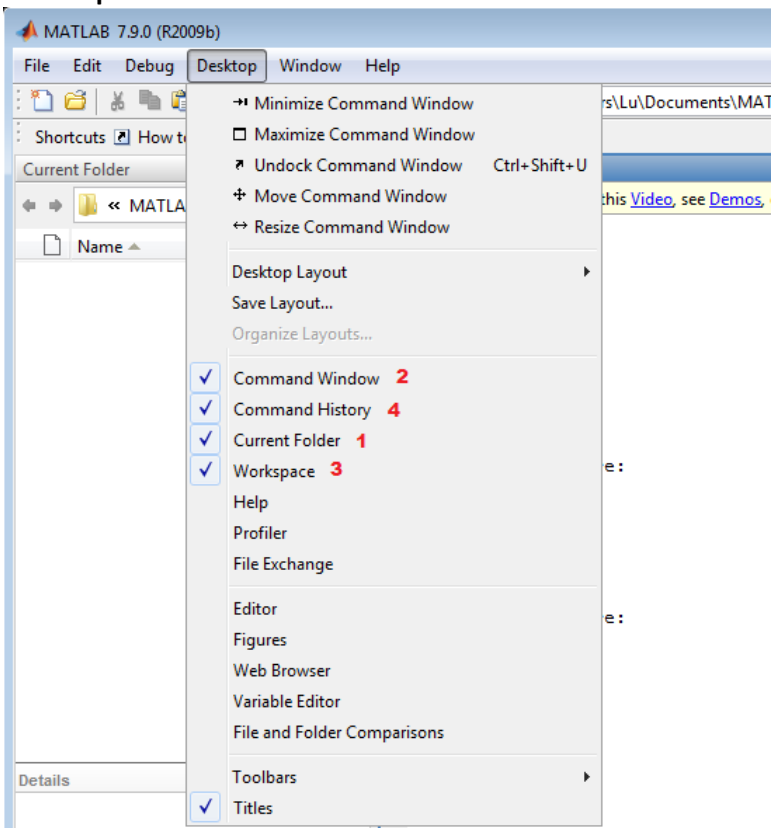


Рис. 3. Меню Desktop

Операції редагування Command Window

При роботі з MATLAB у командному режимі діє найпростіший текстовий редактор. Його робота повинна бути зрозуміла будь-якому користувачу, який стикався з текстовими редакторами. Обмежимося вказівкою команд редагування, які представлені в табл. 1.

Таблиця 1

Команди редактора Command Window MATLAB

Комбінація клавіш	Призначення
→	Переміщення курсору вправо на один символ
←	Переміщення курсору вліво на один символ
Ctrl+→	Переміщення курсору вправо на одне слово
Ctrl+←	Переміщення курсору вліво на одне слово
Home	Переміщення курсору в початок рядка
End	Переміщення курсору в кінець рядка
↑ і ↓	Перегляд попередніх команд вперед або назад для підстановки в рядок вводу
Del	Стирання символу праворуч від курсору
Backspace	Стирання символу ліворуч від курсору
Ctrl+k	Стирання до кінця рядка
Esc	Очищення рядка вводу
Ins	Включення/виключення режиму вставки
PgUp	Перегортання сторінок сесії нагору
PgDown	Перегортання сторінок сесії вниз

Зверніть особливу увагу на застосування клавіш ↑ і ↓. Вони використовуються для підстановки після маркера рядка введення **>>** раніше введених команд, наприклад для їхнього виправлення, дублювання або доповнення. Виправити раніше введені команди іншим чином неможливо, навіть, якщо вони були помилковими. Вводити команди MATLAB можна тільки в рядку з позначкою **>>**.

Відзначимо одну корисну команду керування Командним вікном:

`clc` – очищає екран і розміщає курсор у лівому верхньому куті порожнього екрана;

Обчислення і режимі командного рядка

Інтерпретуюча мова програмування системи MATLAB створена таким чином, що будь-які (часом досить складні) обчислення можна виконувати в режимі *прямих обчислень*, тобто без підготовки і написання спеціальної програми користувачем. При цьому MATLAB виконує функції суперкалькулятора й працює в *режимі командного рядка*.

Робота із системою носить діалоговий характер і відбувається за правилом "поставив запитання – одержав відповідь". За позначкою **>>** користувач набирає на клавіатурі вираз, що обчислюється, редагує його (якщо потрібно) у командному рядку й завершує введення натисканням клавіші **ENTER**. Перелічимо основні правила, які необхідно знати для роботи з MATLAB у командному режимі.

- для вказівки місця введення вихідних команд MATLAB використовується символ >>;
- команди вводяться за допомогою найпростішого рядкового редактора (див. попередній пункт);
- MATLAB виводить результат обчислень в рядках результатів обчислень. Вони відрізняються тим, що не починаються з знака **>>**;
- для блокування виводу результату обчислень деякого виразу після виразу треба встановити знак ; (крапка з комою);
- якщо не зазначена змінна для значення результату обчислень, то MATLAB призначає свою змінну з іменем **ans**;
- знаком присвоювання є звичний математикам знак рівності **=**, а не комбінований знак **:=**, як у деяких інших мовах програмування й математичних системах;

• вбудовані функції (наприклад, **sin**) записуються малими літерами, і їх аргументи вказуються в *круглих дужках*;

На рис. 4 показаний приклад виконання простих обчислень в командному режимі. Тут рядки введені користувачем починаються з знака **>>** (наприклад, **>> a=10**), відповіді MATLAB ідуть нижче без такої позначки. Остання команда завершується знаком '**;**', тому результат обчислень не виведено, проте в віконці Workspace справа видно, що системна змінна **ans** набула значення **-10** (**a - b**).

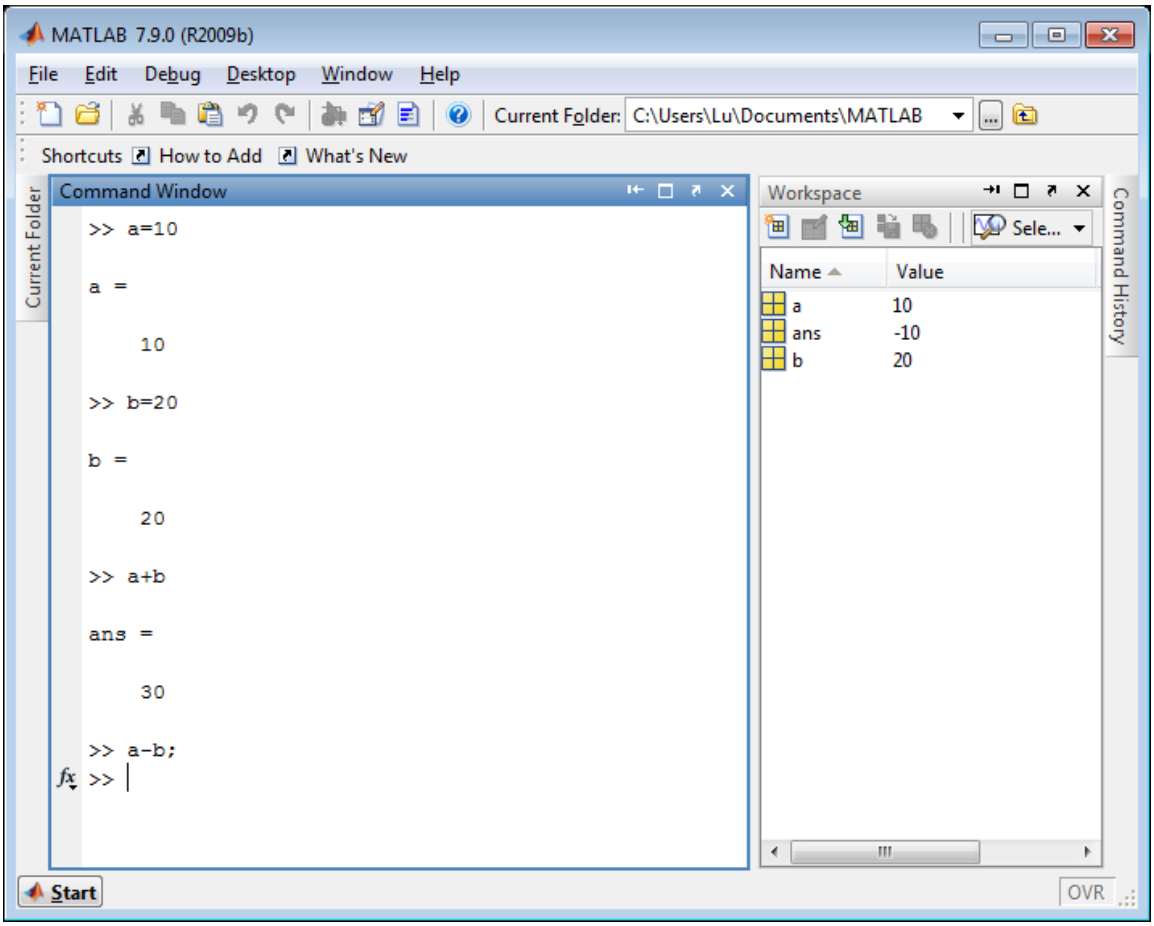


Рис. 4. Прості обчислення в командному режимі

Арифметичні операції

+	Додавання
–	Вирахування
*	Множення
/	Ділення
^	Піднесення до степеня
pi, e, i	Константи

Повний список операторів можна одержати, використовуючи команду **help ops**.

Приклад

```
>> (2+3*pi)/2
ans = 5.7124
```

Зверніть увагу! В цьому і наступних прикладах текст (команди і дані), які вводить користувач позначені зеленим жирним шрифтом та починаються з позначки >> (яку вводити непотрібно). Синім шрифтом показані відповіді MATLAB на відповідні запити.

Для присвоювання виразам імен використовується знак рівності.

Приклад

```
>> a=3-floor(exp(2.9))
a = -15
```

Крапка з комою, розташована наприкінці виразу, забороняє вивід символу на екран комп'ютера.

Приклад

```
>> b=sin(a);  %Зауваження, b не виведене.
>> 2*b^2
ans = 0.8457
```

Константи і системні змінні

Константа – це попередньо визначене числове або символьне значення, представлене унікальним іменем (ідентифікатором). Числа (наприклад, 1, –2 і 1.23) є безіменними *числовими константами*.

Інші види констант в MATLAB прийнято називати *системними змінними*, оскільки, з одного боку, вони задаються системою при її завантаженні, а з іншого – можуть перевизначатися. Основні системні змінні, застосовувані в системі MATLAB, зазначені нижче:

- **i** або **j** –уявна одиниця (корінь квадратний з –1);
- **pi** – число π = 3,1415926…;
- **eps** – погрішність операцій над числами із плаваючою комою (2^–52);
- **realmin** – найменше число із плаваючою крапкою (2^–1022);
- **realmax** – найбільше число із плаваючою крапкою (2^1023)
- **inf** – значення машинної нескінченності;
- **ans** – змінна, що зберігає результат останньої операції і може бути використана для відображення останнього результату в Командному вікні;
- **NaN** – вказівка на нечисловий характер даних (Not_a_Number).

Приклади застосування системних змінних:

Приклад

```
>> 2*pi
ans = 6.2832
>> eps
ans = 2.2204e-016
>> realmin
ans = 2.2251e-308
>> realmax
ans = 1.7977e+308
>> 1/0
Warning: Divide by zero.
ans = Inf
>> 0/0
Warning: Divide by zero.
ans = NaN
```

Як відзначалося, системні змінні можуть *перевизначатися*. Можна задати системної змінної eps інше значення, наприклад **eps=0.0001**. Однак важливо те, що їх значення за замовчуванням задаються відразу після завантаження системи. Тому невизначеними, на відміну від звичайних змінних, системні змінні не можуть бути ніколи.

Символьна константа – це рядок символів, укладених в апострофи, наприклад:

```
'Hello my friend!'
'Привіт'
'2+3'
```

Якщо в апострофи поміщене математичне вираження, то воно *не обчислюється* й розглядається просто як рядок символів. Так що '2+3' не буде повертати число 5. Однак за допомогою спеціальних функцій перетворення символьні вираження можуть бути перетворені в такі, що обчислюються.

Знищення визначень змінних

У пам'яті комп'ютера змінні займають певне місце, називане *робочою областю* (вікно Workspace вже згадувалось вище). Для очищення робочої області використовується функція `clear` у різних формах, наприклад:

- **clear** – знищення визначень усіх змінних;
- **clear x** – знищення визначення змінної x;

• **clear a, b, c** – знищення визначень декількох змінних.

Знищена (стерта в робочій області) змінна стає невизначеною. Використовувати невизначені змінні не можна, і такі спроби будуть супроводжуватися видачею повідомлень про помилку. Приведемо приклади завдання й знищення змінних:

Приклад

```
>> x=2*pi
x = 6.2832
>> V=[1 2 3 4 5]
V = 1 2 3 4 5
>> MAT
??? Undefined function or variable 'MAT'.
>> MAT=[1 2 3 4; 5 6 7 8]
MAT =
1 2 3 4
5 6 7 8
>> clear V
>> V
??? Undefined function or variable 'V'.
>> clear
>> x
??? Undefined function or variable 'x'.
>> M
??? Undefined function or variable 'M'.
```

Зверніть увагу на те, що спочатку вибірково стерта змінна **V**, а потім командою **clear** без параметрів стерті всі інші змінні. Невизначені змінні використовуються при виконанні символьних обчислень. Спеціально система MATLAB для виконання таких обчислень не призначена. Однак вони можливі за допомогою пакета розширення символьної математики Symbolic Math Toolbox.

Формати чисел

Часто необхідно вивести відповідь у вікні MATLAB з більшою чи меншою кількістю знаків або в різних форматах представлення чисел. Для установки певного *формату* представлення чисел використовується команда

```
>> format name
```

де **name** – ім'я формату. Для ілюстрації різних форматів розглянемо вектор, що містить два елементи_числа:

```
x=[4/3 1.2345e-6]
```

У різних форматах представлення елементи вектора будуть мати такий вигляд:

```
format short        1.3333                      0.0000
format short e      1.3333E+000                 1.2345E-006
format long         1.333333333333338           0.000001234500000
format long e       1.333333333333338E+000      1.234500000000000E-006
format bank         1.33                        0.00
```

Завдання формату позначається тільки на *формі виводу* чисел. Обчислення однаково відбуваються у форматі подвійної точності, а введення чисел можливе в будь-якому зручному для користувача виді.

За замовчуванням виводиться приблизно п'ять десяткових значущих цифр (**format short**). Команда **format long** дозволяє вивести приблизно 15 десяткових значущих цифр.

Приклад

```
>> format long
>> 3*cos(sqrt(4.7))
ans = -1.686868922368934
```

Вбудовані функції

Нижче коротко перераховані деякі функції, наявні в MATLAB. Наступний приклад ілюструє, як можна комбінувати функції й арифметичні операції. Опис інших функцій можна знайти, використовуючи легко доступну довідку в діалоговому режимі.

```
abs(#)    cos(#)    exp(#)    log(#)    log10(#)    cosh(#)
sin(#)    tan(#)    sqrt(#)   round(#)  acos(#)     tanh(#)
```

Приклад

```
>> 3*cos(sqrt(4.7))
ans = -1.6869
```

Текстові коментарі в програмах

Оскільки MATLAB використовується для досить складних обчислень, важливе значення має наочність їх опису. Вона досягається, зокрема, за допомогою текстових коментарів. *Текстові коментарі в програмах* уводяться за допомогою символу %, наприклад так:

```
% It is factorial function
```

У нових версіях MATLAB відпала проблема введення коментарів із символами кирилиці. Так що подібний коментар також цілком прийнятний:

```
% Це функція обчислення факторіала
```

Звичайно перші рядки m_файлів служать для опису їх призначення, яке виводиться на екран дисплея після команди

```
>> help Ім'я_файлу
```

Уважається правилом гарного тону вводити в m_файли досить докладні текстові коментарі. Без таких коментарів навіть розроблювач програмних модулів швидко забуває про суть власних розв'язків.

Повідомлення про помилки й виправлення помилок

Велике значення при діалозі із системою MATLAB і налагодженню програм у ній має *діагностика помилок*. Розглянемо ряд прикладів, що пояснюють техніку діагностики. Уведемо, приміром, помилковий вираз

```
>> sqr(2)
```

і натиснемо клавішу **ENTER**. Система повідомить про помилку:

```
??? Undefined function or variable 'sqr'.
```

Це повідомлення говорить про те, що не визначена змінна або функція, і вказує, яка саме, – **sqr**. У цьому випадку, зрозуміло, можна просто набрати правильний вираз. Однак у випадку громіздкого виразу краще скористатися допоміжними функціями редактора команд (див. табл. 1). Для цього досить нажати клавішу ↑ для перегляду попередніх рядків. У результаті в рядку введення з'явиться вираз

```
>> sqr(2)
```

з курсором у його кінці. В MATLAB можна тепер нажати клавішу **Tab**. Система введе підказку, аналізуючи вже введені символи (рис. 5). Із запропонованих системою операторів вибираємо **sqrt**.

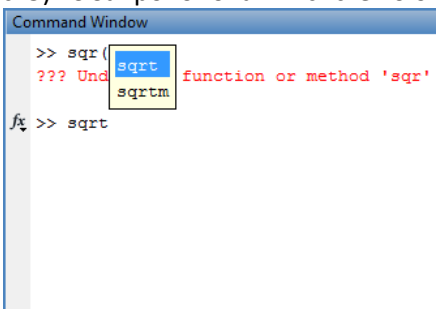


Рис. 5. Підказка відомих MATLAB функцій

Тепер з допомогою клавіші ↑ знову вибираємо потрібний рядок і, користуючись клавішею ←, установлюємо курсор після букви **r**. Тепер натиснемо клавішу **t**, а потім клавішу **ENTER**. Вираження прийме наступний вид:

```
>> sqrt(2)
ans = 1.4142
```

Якби був тільки один варіант закінчення введених символів, то після натискання клавіші **Tab** система б закінчила наше введення без введення **ENTER**. Обчислення дають очікуваний результат – значення квадратного кореня із двох.

У системі MATLAB функції та змінні визначені зовні (користувачем або в додатковому модулі) використовуються точно так само, як і вбудовані функції й оператори. Ніяких додаткових вказівок на їхнє

застосування робити не треба. Досить лише подбати про те, щоб використовувані визначення дійсно існували у вигляді файлів з розширенням .**m**. Втім, якщо ви забудете про це або введете ім'я неіснуючого визначення, то система відреагує на це звуковим сигналом (дзвінком) і виводом повідомлення про помилку:

```
>> hsin(1)
??? Undefined function or variable 'hsin'.
>> sinh(1)
ans = 1.1752
```

У цьому прикладі ми забули (навмисно), яке ім'я має зовнішня функція, що обчислює гіперболічний синус. Система підказала, що функція або змінна з іменем **hsin** не визначена – ні як внутрішня, ні як m-функція. Зате далі ми бачимо, що функція з іменем **sinh** є в складі функцій системи MATLAB – вона задана у вигляді М-функції, що зберігається на жорсткому диску. Тим часом в останньому прикладі ми не давали системі ніяких вказівок на те, що слід шукати саме зовнішню функцію! І це обчислення пройшло так само просто, як обчислення вбудованої функції, такий як **sin**.

Іноді в ході виводу результатів обчислень з'являється скорочення **NaN** (від слів *Not a Number* – не число). Воно позначає невизначеність, наприклад виду **0/0** або **Inf/Inf**, де **Inf** – системна змінна зі значенням машинної нескінченності. Можуть з'являтися й різні попередження про помилки (англійською мовою). Наприклад, при діленні на 0 кінцевого числа з'являється попередження **«Warning: Devide by Zero.»** (**«Увага: ділення на нуль»**). Діапазон чисел, представлених у системі, лежить від 10^{-308} до 10^{+308}.

Загалом кажучи, в MATLAB треба відрізняти *попередження* про помилку від *повідомлення* про неї. *Попередження* (звичайно після слова Warning) не зупиняють обчислення й лише попереджають користувача про те, що помилка здатна вплинути на хід обчислень. *Повідомлення* про помилку (після знаків ???) зупиняє обчислення.

Матриці

Усі змінні в MATLAB інтерпретуються як матриці або масиви. Матриці можна вводити безпосередньо:

Приклад

```
>> A=[1 2 3; 4 5 6; 7 8 9]
A=
    1 2 3
    4 5 6
    7 8 9
```

Крапка з комою використовується для поділу рядків матриці. Відзначимо, що елементи матриці слід розділяти одиночним пробілом. Альтернативний спосіб уведення матриці — рядок за рядком.

Приклад

```
>> A=[1 2 3
4 5 6
7 8 9]
A =
    1 2 3
    4 5 6
    7 8 9
```

Матриці можна формувати, використовуючи вбудовані функції.

Приклад

```
>> Z=zeros(3,5)     %Створення нульової матриці розміру 3x5
Z =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0

>> X=ones(3,5)      %Створення матриці розміру 3x5,
                    %що складається з одиниць
X =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

>> Y=0:0.5:2  %Визначення матриці розміру 1 x 5. Такий запис означає:
```

```
                %перший елемент 0, наступні елементи задані з кроком 0,5
                %останній елемент 2
Y=
     0 0.5000 1.0000 1.5000 2.0000

>> cos(Y)       %Створення матриці розміру 1 x 5,
                %кожний елемент якої є cos від Y
ans=
     1.0000 0.8776 0.5403 0.0707 -0.4161
```

Елементами матриці можна маніпулювати декількома способами.

Приклад

```
>> A=[1 2 3
4 5 6
7 8 9];

>> A(2,3)       %Виділення одного елемента матриці A
                %індекси елементу вказуються в круглих дужках через кому
ans=
     6
>> A(1:2,2:3) %Виділення підматриці A
                %(двокрапка позначає діапазон індексів елементів)
ans=
     2 3
     5 6


>> A( [1 3], [1 3])      %Інший спосіб виділення підматриці A
ans=
     1 3
     7 9
>> A(2,2)=tan(7.8) %Присвоєння нового значення елементу матриці A
A =

     1.0000    2.0000    3.0000
     4.0000   18.5068    6.0000
     7.0000    8.0000    9.0000
```

Додаткові команди для матриць можна знайти, використовуючи довідку в діалоговому режимі або документацію до пакета прикладних програм.

Операції з матрицями

+	Додавання
–	Вирахування
*	Множення
/	Ділення
^	Піднесення до степеня
'	Сполучення і транспонування

Приклад

```
>> B=[1 2;3 4]
B =
     1  2
     3  4
>> C=B'  %C дорівнює транспонованій матриці B
C=
     1  3
     2  4
>> 3*(B*C)^3  %3 (BC)^3
ans=
     13080  29568
     29568  66840
```

Описаний спосіб завдання матриць дозволяє виконати операцію *конкатенації* – об'єднання малих матриць у більшу матрицю. Наприклад, створимо спочатку "магічну" матрицю розміру 3×3:

Приклад

```
>> A=magic(3)
A =
     8 1 6
     3 5 7
     4 9 2
```

Тепер можна побудувати матрицю, що містить чотири матриці:

```
>> B=[A A+16;A+32 A+16]
B =
     8  1  6  24 17 22
     3  5  7  19 21 23
     4  9  2  20 25 18
    40 33 38 24 17 22
    35 37 39 19 21 23
    36 41 34 20 25 18
```

Отримана матриця має вже розмір 6×6. Обчислимо суму її стовпців:

```
>> sum(B)
ans = 126 126 126 126 126 126
```

Цікаво, що вона однакова для всіх стовпців. А для обчислення суми рядків використовуємо команду

```
>> sum(B')
ans = 78 78 78 174 174 174
```

Тут запис **B'** означає транспонування матриці **B**, тобто заміну рядків стовпцями. Цього разу сума виявилася різною. Це відкидає початкове припущення, що матриця **B** теж є магічною. Для істинно магічної матриці суми стовпців і рядків повинні бути однаковими:

```
>> D=magic(6)
D =
    35  1  6 26 19 24
     3 32  7 21 23 25
    31  9  2 22 27 20
     8 28 33 17 10 15
    30  5 34 12 14 16
     4 36 29 13 18 11
>> sum(D)
ans = 111 111 111 111 111 111
>> sum(D')
ans = 111 111 111 111 111 111
```

Більше того, для магічної матриці однаковою є й сума елементів по основних діагоналях (головної діагоналі й головної антидіагоналі).

Видалення стовпців і рядків матриць

Для формування матриць і виконання ряду матричних операцій виникає необхідність видалення окремих стовпців і рядків матриці. Для цього використовуються порожні квадратні дужки – [ ]. Проробимо це з матрицею **M**:

```
>> M=[1 2 3; 4 5 6; 7 8 9]
M =
     1 2 3
     4 5 6
   7 8 9
```

Вилучимо другий стовпець, використовуючи оператор **:** (просто двокрапка позначає всі рядки, якщо стоїть на місці першого індексу і стовпчики, якщо введена замість другого індексу):

```
>> M(:,2)=[]
M =
     1 3
     4 6
   7 9
```

А тепер, використовуючи оператор **:** (двокрапка), вилучимо другий рядок:

```
>> M(2,:)=[ ]
M =
    1 3
    7 9
```

Операції з масивами

Однією з найбільш корисних властивостей пакета MATLAB є кількість операцій, які можна виконувати над окремими елементами матриці. Така операція була продемонстрована вище, коли елементи матриці розміру 1 x 5 записувалися як cos від елементів матриці, що вводиться. Операції додавання, віднімання множення матриці на скаляр завжди здійснюються поелементно, але це не відноситься до операцій множення, ділення і піднесення матриці до степеня. Дані три операції можна робити поелементно, як і попередні, але з покажчиком поелементної операції (крапка): **.***, **./** і **.^**. Важливо зрозуміти, як і коли саме їх можна використовувати. Операції над масивами є вирішальними для ефективної побудови й виконання пакетом MATLAB програм і графіків.

Приклад

```
>> A=[1 2;3 4];
>> A^2   %Добуток матриць A*A
ans=
    7 10
    15 22
>> A.^2 %Квадрат кожного елементу A
ans=
    1 4
    9 16
>> cos(A./2)  %Ділення кожного елементу A на 2
              %і обчислення cos від кожного елементу
ans=
    0.8776  0.5403
    0.0707 -0.4161
```

Графіки

Пакет MATLAB дозволяє будувати дво- і тривимірні графіки кривих і поверхонь. Додаткові можливості й опис графіків у пакеті MATLAB можна знайти, використовуючи довідку в діалоговому режимі або документацію до пакета прикладних програм.

Команду **plot(X, Y)** використовують для побудови графіка двомірних функцій. X — вектор абсцис графіка функції, Y— вектор ординат графіка. Їх розміри повинні співпадати.

Наступний приклад показує, як створити програму побудови графіка функцій **y = cos(x)** і **y = cos²(x)** на інтервалі **[0; π]**.

Приклад

```
>> x=0:0.1:pi;
>> y=cos(x);
>> z=cos(x).^2;
>> plot(x, y, x, z, 'o')
```

У першому рядку програми задається область визначення **[0; π]** із кроком 0,1. У наступних двох рядках задаються дві функції від **x**. Відмітимо, що перші три рядки закінчуються крапкою з комою. Крапка з комою необхідна, щоб заборонити вивід матриць x, y и z на екран. У четвертому рядку втримується команда **plot**, яка й будує графік у окремому вікні Figure 1 (рис. 6).

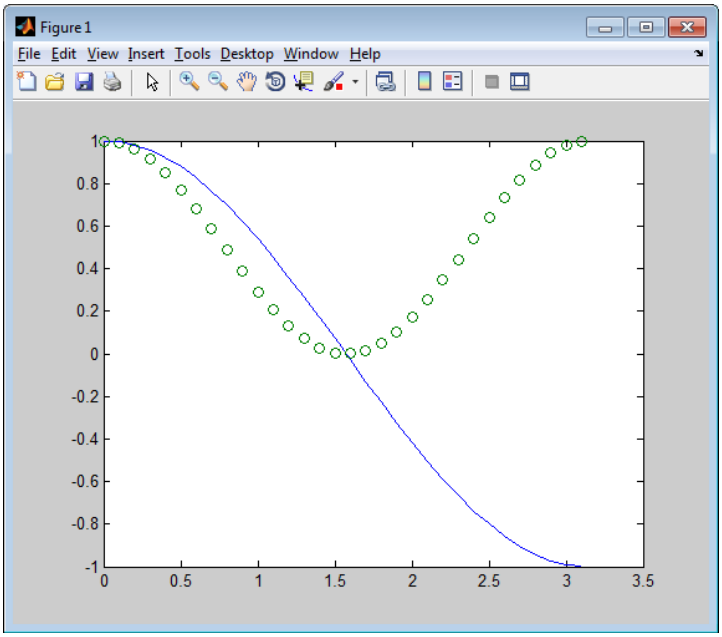


Рис. 6. Побудова графіків

Перші параметри команди **plot**, **x** і **y**, задають абсциси і ординати точок графіка функції **y = cos(x)**. Третій і четвертий параметри, **x** і **z**, задають координати точок графіка функції **y = cos²(x)**. Останній параметр — **'o'**, він вказує символ, яким будуть позначені точки (x_k, z_k) другого графіка, де $z_k = \cos^2(x_k)$.

У третьому рядку необхідно використовувати оператор "**.^**", тому що спочатку обчислюється значення **cos** кожного елемента матриці **x**. Потім кожний елемент матриці **cos(x)** за допомогою команди **.^** підноситься до квадрату.

plot(X,Y,S) аналогічна команді **plot(X,Y)**, але тип лінії графіка можна задавати за допомогою строкової константи **S**.

Значеннями константи **S** можуть бути наступні символи.

Таблиця 2

Колір лінії	
y	Жовтий
m	Фіолетовий
c	Блакитний
r	Червоний
g	Зелений
b	Синій
w	Білий
k	Чорний
Тип точки	
.	Точка
o	Окружність
x	Хрест
+	Плюс
*	Зірочка
s	Квадрат
d	Ромб
v	Трикутник (униз)
	Трикутник (нагору)
<	Трикутник (уліво)
>	Трикутник (вправо)
p	П'ятикутник
Тип лінії	

–	Суцільна
:	Подвійний пунктир
–.	Штрих-Пунктир
––	Штрихова

Розглянемо приклад простої програми для побудови графіків трьох функцій з різним стилем представлення кожної з них:

Приклад

```
>> x=-2*pi:0.1*pi:2*pi;
>> y1=sin(x); y2=sin(x).^2; y3=sin(x).^3;
>> plot(x,y1,'-m',x,y2,'-.+r',x,y3,'-ok')
```

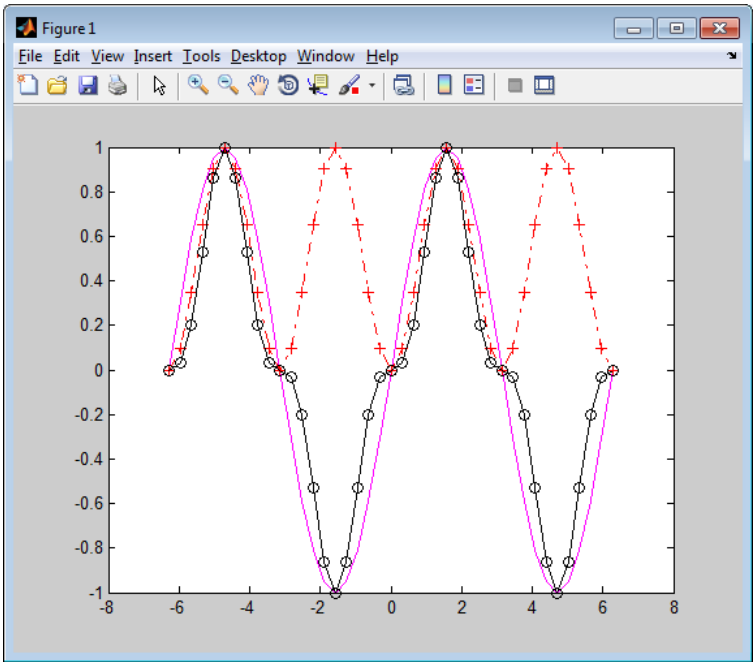


Рис. 7. Побудова графіків різним стилем ліній

Команда побудови графіка **fplot** — це корисна альтернатива команді **plot**. Форма її запису: **fplot ('name', [a, b], n)**. Вона будує графік функції **name.m** (тобто функцію з файлу користувача або додаткового модуля) по множині *n* точок на інтервалі [*a*; *b*]. За замовчуванням число *n* рівно 25.

Приклад

```
>> fplot('tanh', [-2,2]) %Графік y = tanh(x)на[-2;2]
```

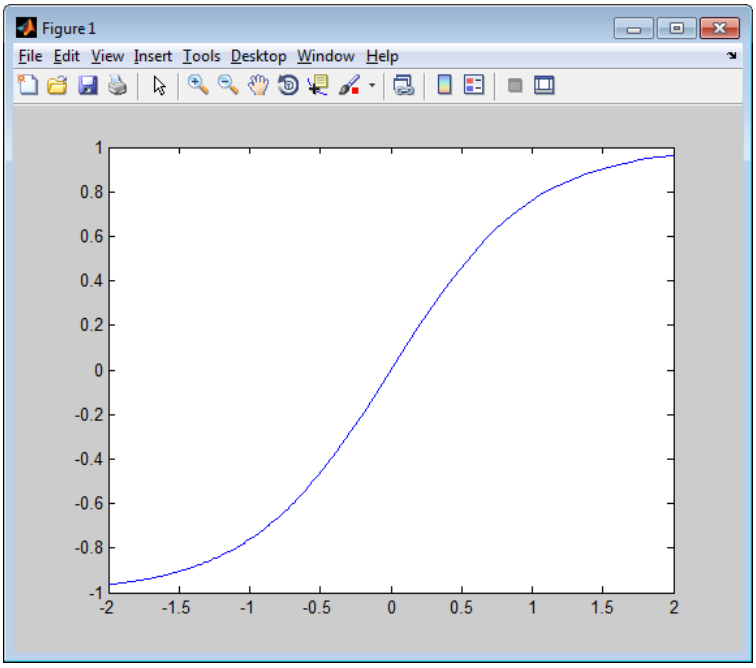


Рис. 8. Графік **tanh** функцією **fplot**

Команди **plot** і **plot3** відповідно використовуються для побудови параметричних кривих у дво- і тривимірному просторі. Вони особливо корисні для створення зорових образів розв'язків диференціальних рівнянь у двох і трьох вимірах.

Приклад

Побудуємо еліпс $c(t) = (2\cos(t), 3\sin(t))$, де $0 < t < 2\pi$, використовуючи наступні команди

```
>> t=0:0.2:2*pi;
>> plot(2*cos(t),3*sin(t))
```

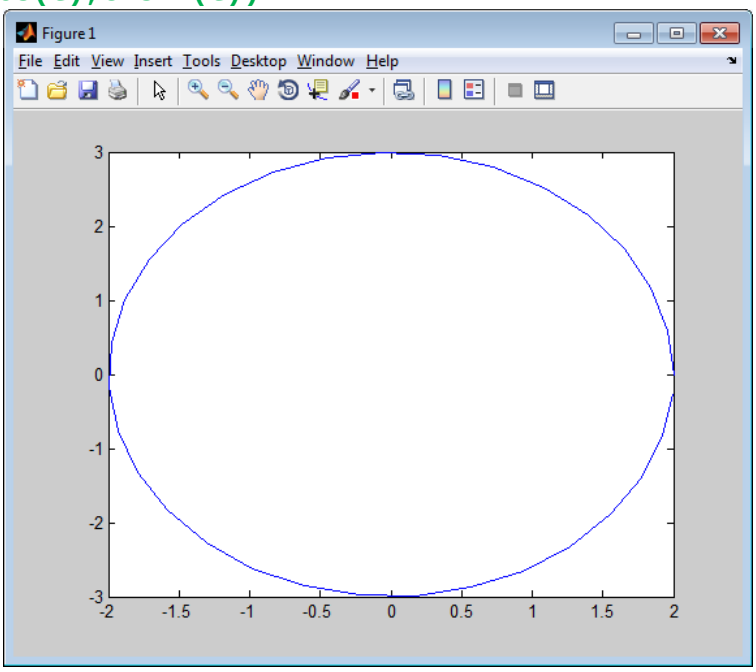


Рис. 9. Параметрична крива

Приклад

Графік кривої $c(t) = (2\cos(t), t^2, 1/t)$, $0.1 < t < 4\pi$, одержуємо за допомогою наступних команд.

```
>> t=0.1:0.1:4*pi;
>> plot3(2*cos(t), t.^2, 1./t)
```

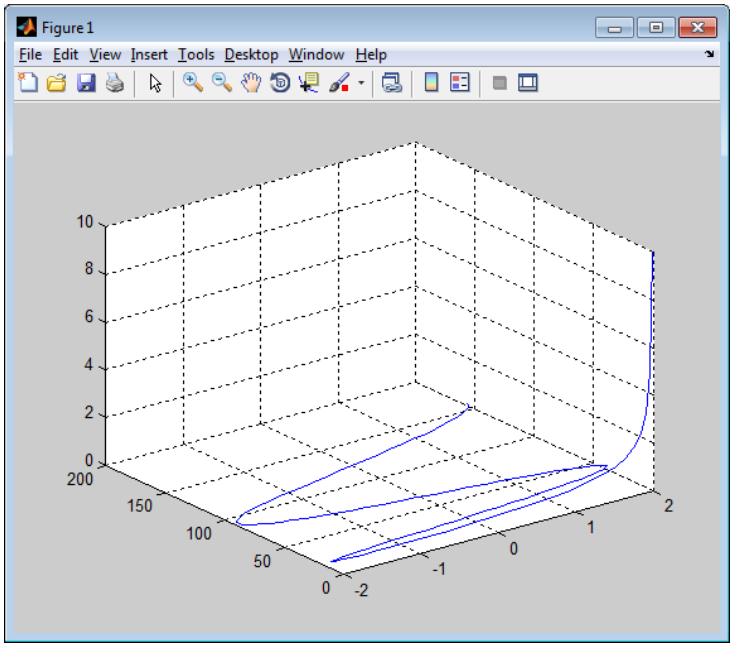


Рис. 10. Параметрична крива у трьох вимірах

Для установки написів біля осей *x*, *y* і *z* використовуються наступні команди:

```
xlabel('String')
ylabel('String')
zlabel('String')
```

Відповідний напис задається символьною константою або змінною типу 'String'.

Часто виникає необхідність додавання тексту в певне місце графіка, наприклад для позначення тієї або іншої кривий графіка. Для цього використовується команда **`text`**:

- **`text(X,Y,'string')`** додає у двовимірний графік текст, заданий строковою константою 'string', так що початок тексту розташований у точці з координатами (*X*,*Y*). Якщо X і Y задані як одномірні масиви, то напис міститься в усі позиції [x(i),y(i)];
- **`text(X,Y,Z,'string')`** додає в тривимірний графік текст, заданий строковою константою 'string', так що початок тексту розташований у позиції, заданої координатами X, Y і Z.

У наведеному нижче прикладі напис «Графік функції sin(x^3)» розміщається під кривою графіка в позиції ((4, 0.7):

Приклад

```
>> x=-10:0.1:10; plot(x,sin(x).^3)
>> text(-4,0.7,'Graphic sin(x)^3')
```

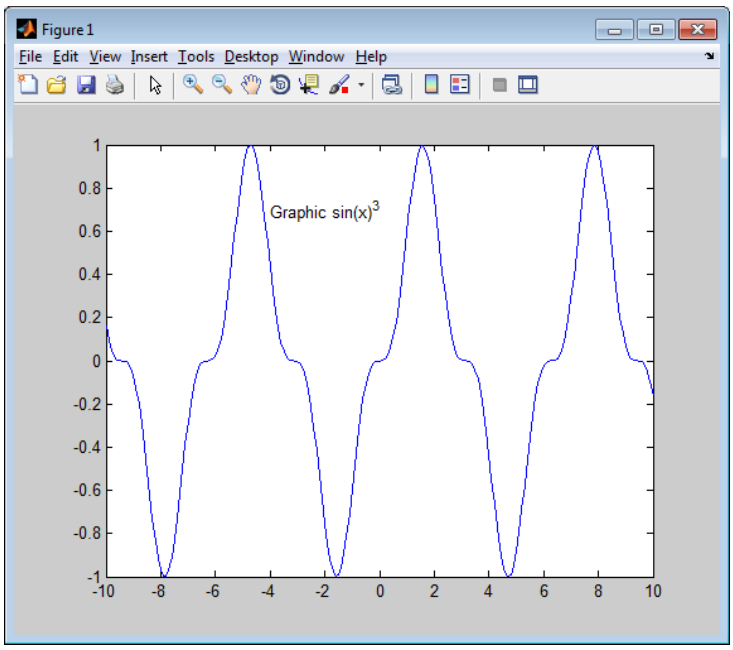


Рис. 11. Текст на графіку

Пояснення у вигляді відрізків ліній з довідковими написами, розташовуване усередині графіка або біля нього, називається *легендою*. Для створення легенди використовуються різні варіанти команди **legend**:

legend(string1,string2,string3,…) додає до поточного графіка легенду у вигляді рядків, зазначених у списку параметрів.

Програма, наведена нижче, будує графік трьох функцій з легендою, розміщеною в поле графіка:

Приклад

```
% Програма побудови графіка трьох функцій
% з виводом їх позначень – легендою
>> x=-2*pi:0.1*pi:2*pi;
>> y1=sin(x); y2=sin(x).^2; y3=sin(x).^3;
>> plot(x,y1,'-m',x,y2,'-.+r',x,y3,'-ok')
>> legend('Function 1','Function 2','Function 3');
```

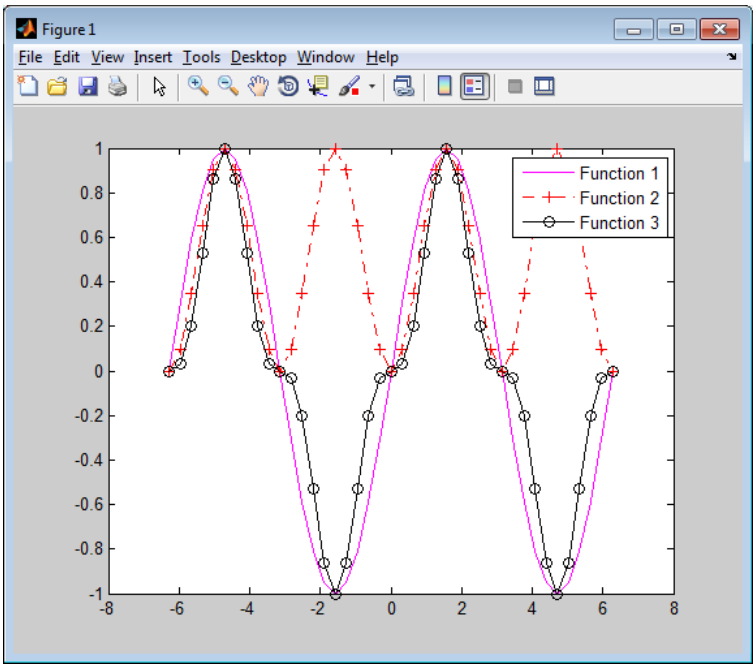


Рис. 12. Легенда на графіку

Зазвичай графіки виводяться в режимі автоматичного масштабування. Наступні команди класу **axis** міняють цю ситуацію:

- **axis([XMIN XMAX YMIN YMAX])** – установка діапазонів координат по осям *x* і *y* для поточного двовимірного графіка;
- **axis([XMIN XMAX YMIN YMAX ZMIN ZMAX])** – установка діапазонів координат по осях *x*, *y* і *z* поточного тривимірного графіка;
- **axis auto** – встановка параметрів осей за замовчуванням;
- **axis manual** «заморожує» масштабування в поточному стані, щоб при використанні команди hold on наступні графіки використовували ті ж параметри осей;
- **axis tight** встановлює діапазони координат по осях відповідно до діапазонів зміни даних;
- **axis ij** задає «матричну» прямокутну систему координат з початком координат у лівому верхньому куті, вісь *i* – вертикальна, що розмічається зверху вниз, вісь *j* – горизонтальна й розмічається ліворуч праворуч;
- **axis xy** встановлює декартову систему координат з горизонтальною віссю *x*, що розмічається ліворуч праворуч, і вертикальною віссю *y*, що розмічається знизу нагору (початок координат розміщається в нижньому лівому куті);
- **axis equal** включає масштаб з однаковою відстанню між мітками по осях *x*, *y* і *z*;
- **axis image** встановлює масштаб, при якому пікселі зображення стають квадратами;
- **axis square** встановлює поточні осі у вигляді квадрата (або куба в тривимірному випадку) з однаковою відстанню між мітками й однаковою довжиною осей;
- **axis normal** відновлює масштаб, скасовуючи установки axis equal і axis square;
- **axis vis3d** «заморожує» пропорції осей для можливості повороту тривимірних об'єктів;

- **axis off** забирає з осей їх позначення й маркери;
- **axis on** відновлює раніше введені позначення осей і маркери;
- **V=axis** повертає вектор-рядок, що містить коефіцієнти масштабування для поточного графіка. Якщо поточний графік двовимірний, то вектор має 4 компонента, якщо тривимірний – 6 компонентів.

Наступний приклад ілюструє застосування команди **axis** при побудові двовимірного графіка функції однієї змінної:

Приклад

```
>> x=-5:0.1:5; plot(x,sin(x));
>> axis([-10 10 -1.5 1.5])
```

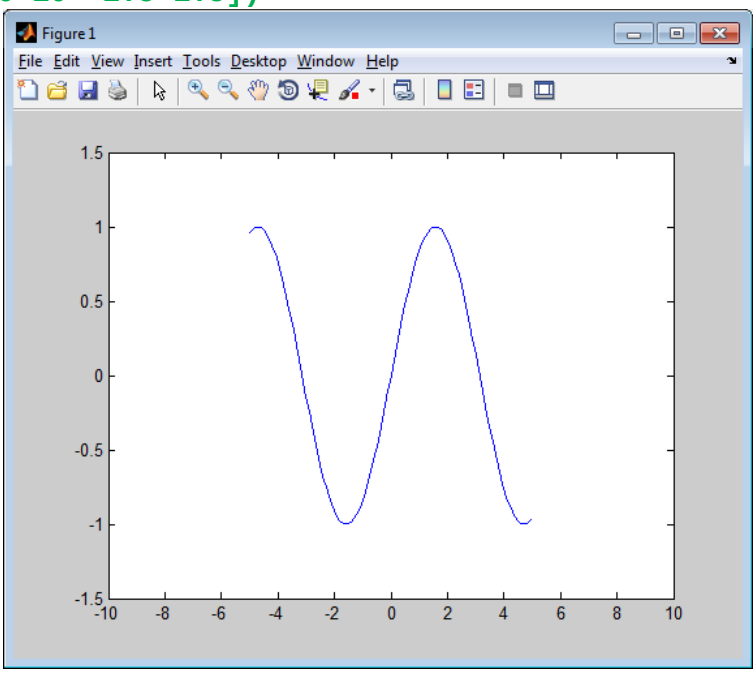


Рис. 13. Зміна діапазонів координат графіку

У математичній, фізичній і іншій літературі при побудові графіків на додаток до розмітки осей часто використовують масштабну сітку. Команди **grid** дозволяють задавати побудова сітки або скасовувати ця побудова:

- **grid on** додає сітку до поточного графіка;
- **grid off** відключає сітку;

У багатьох випадках бажана побудова багатьох накладених один на одного графіків у тому самому вікні. Для цього служить команда продовження графічних побудов **hold**. Вона використовується в наступних формах:

- **hold on** забезпечує продовження виводу графіків у поточне вікно, що дозволяє додавати наступні графіки до вже існуючих;
- **hold off** скасовує режим продовження графічних побудов;
- **hold** працює як перемикач, послідовно включаючи режим продовження графічних побудов і скасовуючи його.

Наведений нижче приклад показує, як за допомогою команди **hold on** на графік синусоїди накладаються ще три графіка параметрично заданих функцій:

Приклад

```
>> x=-5:0.1:5; plot(x,sin(x)); hold on
>> plot(sin(x),cos(x)); plot(2*sin(x),cos(x))
>> plot(4*sin(x),cos(x)); hold off
```

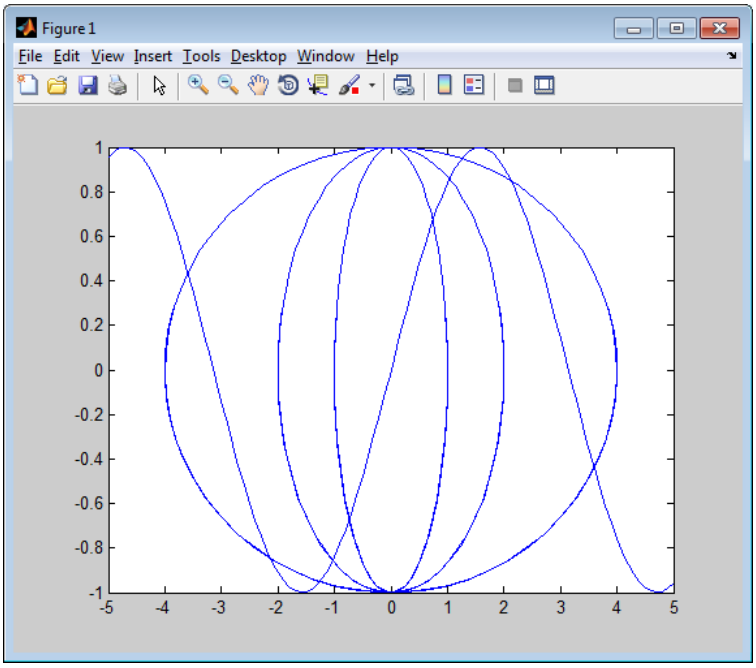


Рис. 14. Накладання графіків за допомогою hold.

Програмування в MATLAB

Система MATLAB зазвичай працює в розглянутому вище режимі інтерпретації команд і операторів: вони вводяться в ході сеансу в командному рядку, а MATLAB виконує їх негайну обробку і видає обчислений результат. Проте в MATLAB є можливість обробки заздалегідь підготовленої послідовності команд і операторів, записаної у вигляді файлу. Коди програм в системі MATLAB пишуться мовою високого рівня, досить зрозумілою для користувачів помірної кваліфікації в області програмування.

Програмні файли, що містять команди і оператори MATLAB, називаються М-файлами. Існує два типи М-файлів: ***m-сценарії*** і ***m-функції*** — з наступними характеристиками:

Таблиця 3

m-сценарій	m-функція
Не допускає вхідних і вихідних аргументів	Допускає вхідні і вихідні аргументи
Оперує з даними з робочої області	За умовчанням внутрішні змінні є локальними по відношенню до функції
Призначений для автоматизації послідовності кроків, які треба виконувати багато разів	Призначена для розширення можливостей мови MATLAB (бібліотеки функцій, пакети прикладних програм)

Визначення власних функцій користувача

В MATLAB користувач може визначити функцію, написавши власний m-файл-функцію (наприкінці файлу ставиться розширення **.m**). М-файли функції створюються, редагуються і налагоджуються в спеціальному редакторі MATLAB (Editor/Debugger). Цей редактор відкривається, наприклад командою **New->Function** у меню **File** (рис. 15) або відкривається в пункті меню **Desktop->Editor**.

Функція в загальному випадку перетворює одні дані в інші. Для багатьох функцій характерне повернення значень у відповідь на звернення до них з вказівкою списку вхідних параметрів — аргументів. Наприклад, говорять, що функція **`sin(x)`** у відповідь на звернення до неї повертає значення синуса аргументу **`x`**. Тому функцію можна використати в арифметичних виразах, наприклад **`2*sin(x+1)`**. Для команд, що не повертають значення, таке застосування звичайно абсурдне.

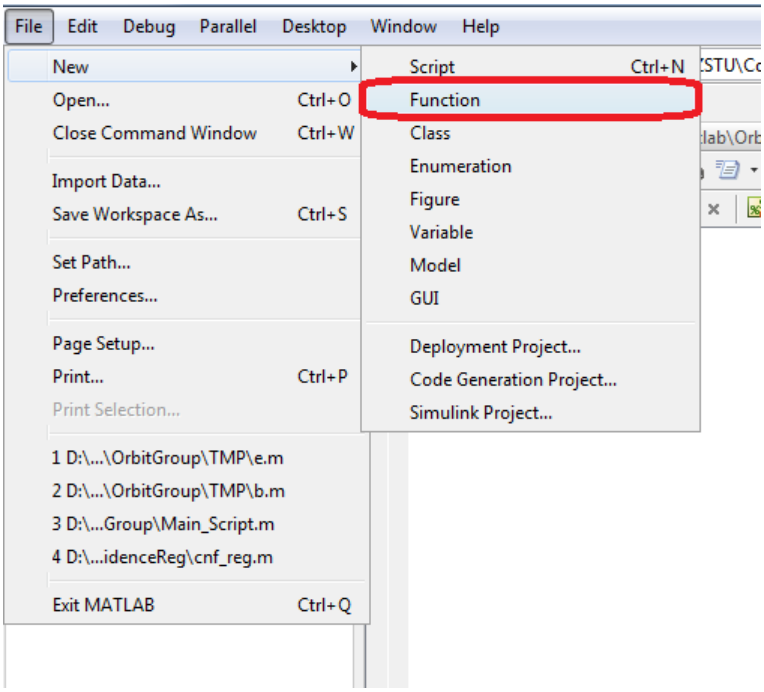


Рис. 15. Створення нового m-файлу функції

Структура m-файлу функції з одним вихідним параметром виглядає таким чином:

```
function var=f_name(Список_параметрів)
%Основний коментар
%Додатковий коментар
Тіло файлу з будь-якими виразами
var=вираз
```

m -файл функція має наступні властивості:

- він розпочинається з оголошення **function**, після якого вказуються ім'я змінної **var** — вихідного параметра, ім'я самої функції і список її вхідних параметрів;
- функція повертає своє значення і може використовуватися в математичних виразах у виді:

f_name(Список_параметрів)

- усі змінні, наявні в тілі файлу-функції, є локальними, тобто діють тільки в межах тіла функції;
- файл-функція є самостійним програмним модулем, який спілкується з іншими модулями через свої вхідні і вихідні параметри;
- файл-функція служить засобом розширення системи MATLAB;
- при виявленні файлу-функції він компілюється і потім виконується, а створені машинні коди зберігаються в робочій області системи MATLAB.

Остання конструкція **var=вираз** вводиться, якщо необхідно, щоб функція повертала результат обчислень.

Наведена форма файлу-функції характерна для функції з одним вихідним параметром. Якщо вихідних параметрів більше, то вони вказуються в квадратних дужках після слова **function**. При цьому структура модуля має наступний вид:

```
function [var1, var2]=f_name(Список_параметров)
%Основний коментар
%Додатковий коментар
Тіло файлу з будь-якими виразами
var1=вираз
var2=вираз
```

Така функція багато в чому нагадує процедуру. Її не можна сліпо використати безпосередньо в математичних виразах, оскільки вона повертає не єдиний результат, а безліч результатів — по числу вихідних параметрів. Якщо функція використовується як що має єдиний вихідний параметр, але має ряд вихідних параметрів, то для повернення значення використовуватиметься перший з них. Це частенько веде до помилок в математичних обчисленнях. Тому, як відзначалося, ця функція використовується як окремий елемент програм виду:

[var1, var2...]=f_name(Список_параметрів)

Після його застосування змінні виходу **var1**, **var2 ...** стають визначеними, і їх можна використати в подальших математичних виразах і інших сегментах програми. Якщо функція використовується у вигляді **f_name(Список_параметрів)**, то повертається значення тільки першого вихідного параметра — змінної **var1**.

Визначимо функцію **fun(x) = 1 + x – x²/4** у m-файлі з назвою **fun.m**. Для створення файлу дамо команду меню **File->New->Function** (рис. 15).

В відкритому вікні Editor записується власна функція:

```
function y=fun(x)
%y(x)=1+x–x^2/4
%Приклад власної функції користувача
y=1+x-x.^2/4;
```

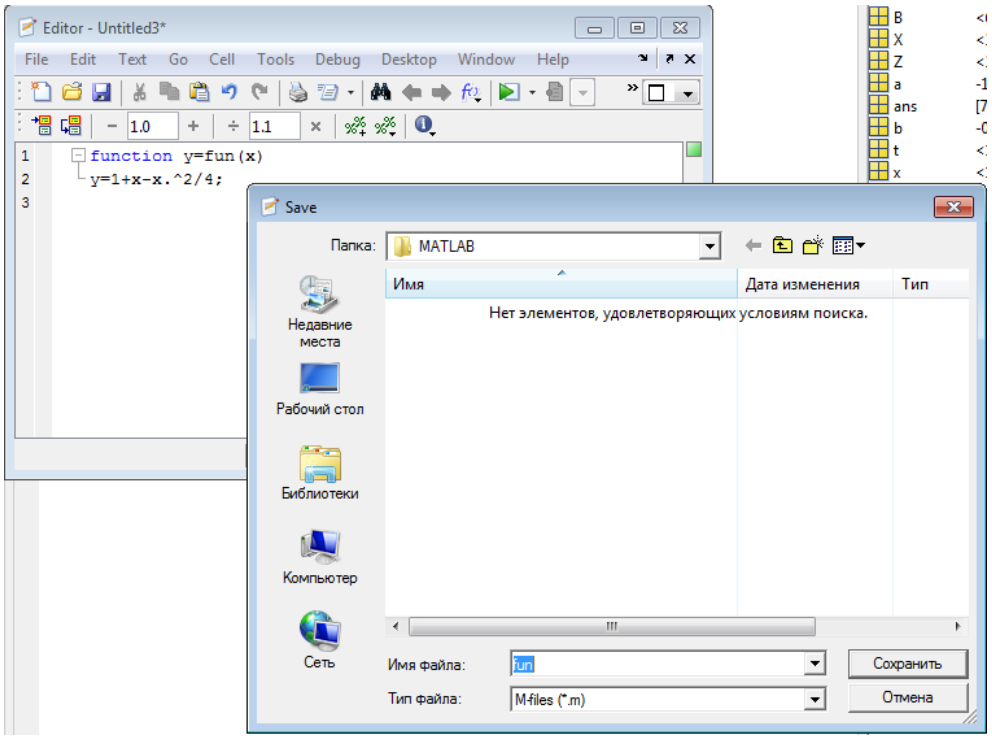


Рис. 16. Збереження m-файлу

Для позначення змінних можна вживати різні букви і для назви функцій — різноманітні імена. Спочатку m-Файл має ім'я Untitled. Його требі зберегти на диск під тою самою назвою, як названа функція (в даному прикладі **fun.m**).

Функцію, записану як m-Файл під іменем **fun.m**, можна викликати в MATLAB Command Window так само, як будь-яку іншу функцію.

Приклад

```
>> cos(fun(3))
ans = –0.1782
```

Корисним і ефективним способом обчислення функцій є використання команди **feval**. Першим аргументом **feval** передається ім'я функції (як текстовий рядок в лапках), далі передаються всі аргументи функції.

Приклад

```
>> feval('fun',4)
ans = 1
```

Ще одна з можливостей створення користувацьких функцій полягає в застосуванні функції **inline**, аргументом якої треба в лапках задати текстовий рядок — вираз, що задає функцію однієї або декількох змінних. У наведеному нижче прикладі задана та сама функція **1 + x – x²/4** , але як **inline**-функція **fun2**:

Приклад

```
>> fun2=inline('1+x-x.^2/4')
fun2 =

     Inline function:
     fun2(x) = 1+x-x.^2/4

%Викличемо створену функцію
>> cos(fun2(3))
ans =
    -0.1782
```

Приклад визначення функції двох змінних – суми квадратів **sin(x)** і **cos(y)**:

Приклад

```
>> sc2=inline('sin(x).^2+cos(y).^2')
sc2 =
     Inline function:
     sc2(x,y) = sin(x).2+cos(y).^2

%Викличемо створену функцію
>> sc2(0.5,0.7)
ans =
     0.8148
```

Визначення сценаріїв (script) MATLAB

Файл-сценарий, що іменується також Script-файлом, є просто записом серії команд без вхідних і вихідних параметрів. Він має таку структуру:

```
%Основний коментар
%Додатковий коментар
Тіло файлу з будь-якими виразами
```

Важливі наступні властивості файлів-сценаріїв :

- вони не мають вхідних і вихідних аргументів;
- працюють з даними з робочої області;
- в процесі виконання не компілюються;
- скрипт є зафіксованою у вигляді файлу послідовністю операцій, повністю аналогічною тій, що використовується в сесії.

Створити новий m-файл сценарію можна давши команду меню **File->New->Blank M-File** (рис. 17). Відкриється вікно редактора для написання нового сценарію.

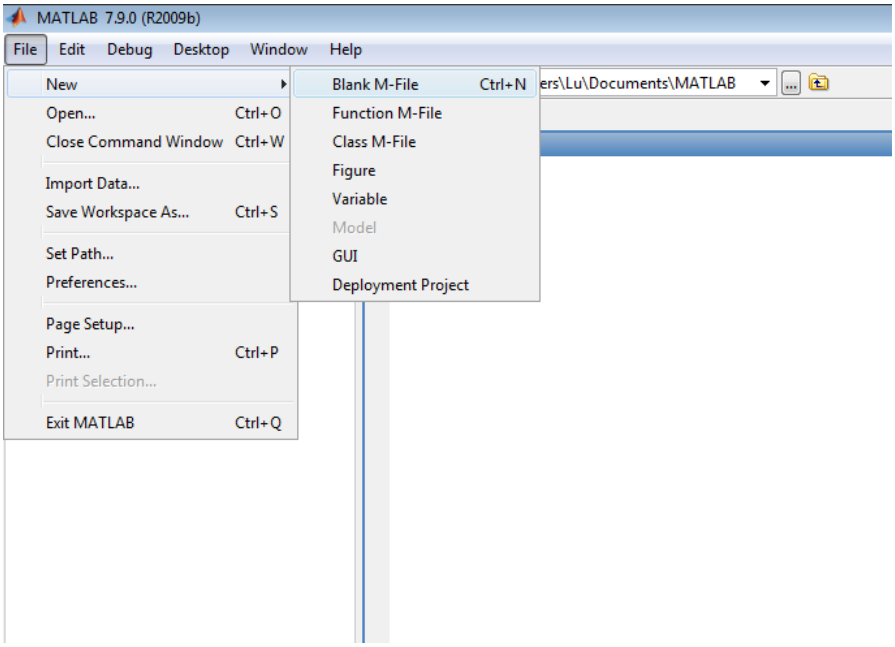


Рис. 17. Створення нового m-файлу скрипта

Згідно з правильним стилем програмування файли сценаріїв необхідно починати з коментарів. Основним коментарем є перший рядок текстових коментарів, а додатковим – наступні рядки. Далі йдуть команди сценарію.

Розглянемо наступний приклад файлу-сценарію, в якому будується графік функції **`sin(x)`**, якщо в Робочій області попередньо задані змінні `xmin` і `xmax`:

```
%Plot with color red
%Будує графік синусоїди лінією червоного кольору
%з виведеною масштабною сіткою в інтервалі [xmin, xmax]
x=xmin: 0.1: xmax;
plot(x, sin(x),'r')
grid on
```

Після редагування файл необхідно зберегти, наприклад з ім'ям **`plsin.m`**. Запустити цей скрипт можна за його назвою **`plsin`** (без будь-яких дужок і параметрів) або, якщо він відкритий в Редакторі кнопкою на панелі інструментів Редактора ( ) чи **F5**.

Нижче розглянуті деякі оператори мови програмування MATLAB.

Оператори відношення

== Дорівнює
~= Не дорівнює
< Менше
> Більше
<= Менше або дорівнює
>= Більше або дорівнює

Логічні оператори

~ Not (доповнення)
& And (справедливо, якщо істині обоє операнди)
| Or (справеливо, якщо один із двох або обоє операнди істині)

Булеві величини

`true` логічна одиниця **`logical(1)`**

false логічний нуль **logical(0)**

Умовне виконання інструкцій

Оператор умовного виконання може бути представлений наступними варіантами:

```
if Логічний вираз
   Виконувані оператори
end
```

```
if Логічний вираз
  Виконувані оператори
else
  Виконувані оператори
end
```

```
if Логічний вираз
  Виконувані оператори
elseif Логічний вираз
  Виконувані оператори
else
  Виконувані оператори
end
```

Оператор умови **if ... end** обчислює деякий логічний вираз і виконує відповідну групу інструкцій в залежності від значення цього виразу. Якщо логічний вираз істинний, то MATLAB виконає усі інструкції між **if** і **end**, а потім продовжить виконання програми в рядку після **end**. Якщо умова неправдива, то MATLAB пропускає усі оператори між **if** та **end** і продовжує виконання в рядку після **end**.

Оператори **if... else ... end** і **if... elseif... end** створюють додаткові розгалуження усередині тіла оператора **if**:

- оператор **else** не містить логічної умови. Інструкції, пов'язані з ним, виконуються, якщо попередній оператор **if** (і можливо, **elseif**) неправдиві;
- оператор **elseif** містить логічну умову, яка обчислюється, якщо попередній оператор **if** (і можливо, **elseif**) неправдиві. Інструкції, пов'язані з оператором **elseif**, виконуються, якщо відповідна логічна умова істинна. Оператор **elseif** може багаторазово використовуватися усередині оператора умови **if**.

Приклад:

Приклад

```
if rem(a, 2) == 0
   disp('a парне')
   b = a/2;
end
```

Якщо логічна умова включає змінну, що не являється скаляром, то твердження буде істинним, якщо усі елементи відмінні від нуля. Нехай задана матриця **X**; запишемо наступний оператор умови:

```
if X
   Інструкції
end
```

Інструкції виконаються тільки, якщо усі елементи матриці **X** відмінні від нуля.

Якщо в операторові **if** умовний вираз є порожнім масивом, то така умова неправдива.

Цикли

Оператори **for** і **while** установлені в MATLAB аналогічно їх прототипам в інших мовах програмування. Цикли **for** мають наступну основну форму:

```
for <змінна циклу> = <початкове значення>:<приріст>:<кінцеве значення>
   Виконувані оператори
end
```

Оператор циклу **for** виконує інструкцію або групу інструкцій задане число разів. За умовчанням приріст дорівнює 1. Можна задавати будь-який приріст, у тому числі негативний. Для позитивних індексів виконання завершується, коли значення індексу перевищує **<кінцеве значення>**; для негативних приростів виконання завершується, коли значення індексу стає менше ніж **<кінцеве значення>**.

Цей цикл виконується 5 разів:

```
for i = 2:6
   x(i) = 2*x(i - 1);
end
```

Допустимі вкладені цикли типу

```
for i = 1:m
```

```
    for j = 1:n
      A(i, j) = 1/(i + j - 1);
    end
  end
```

Допустимі цикли з векторною керуючою змінною.

```
x =[];
for v =[0 2 3 1]
   x =[x  2^v];
end
```

Цикл формує вектор, елементи якого є степенями $2^{v(i)}$.

Оператор циклу з невизначеним числом операцій **while ... end** багаторазово виконує інструкцію або групу інструкцій, поки логічний вираз істинний.

```
while <логічний вираз>
   Виконувані оператори
end
```

Наведемо простий приклад діалогової програми, яка служить для багатократного обчислення довжини кола по значенню радіусу r, що вводиться користувачем.

```
% Обчислення довжини кола з діалоговим введенням радіусу r=0;
while r >= 0,
   r=input('Введіть радіус кола r=');
   if r>= 0
     disp('Довжина кола l=');
     disp(2*pi*r);
   end
end
```

Приведені рядки включені в оператор **while...end**. Це необхідно для циклічного повторення обчислень з введенням значень r. Поки r >= 0, цикл повторюється. Але варто задати r<0, обчислення довжини кола перестає виконуватися, а цикл завершується.

Тут також продемонстрований приклад організації простого діалогу. Він реалізований за допомогою команди **input**:

```
r=input('Введіть радіус кола r=');
```

При виконанні цієї команди спочатку виводиться запит у вигляді рядка, потім відбувається зупинка роботи програми і очікується введення значення радіусу r (у загальному випадку числа). Введення, як завжди, підтверджується натисненням клавіші **Enter**, після чого введене число привласнюється змінній r. Наступні рядки

```
disp('Довжина кола l=');
disp(2*pi*r);
```

за допомогою команди **disp** при r >= 0 виводять напис **"Довжина кола l="** і обчислене значення довжини кола (**2*pi*r**).

Функція **break** перериває виконання циклів **for** і **while**. У разі вкладених циклів переривання можливо тільки з самого внутрішнього циклу.

Приклад:

```
while 1
   n = input('Введіть n. Припинення введення n <= 0. n =')
   if n <= 0
     break
   end
   r = rank(magic(n))
end
```

Цикл **while ... end** виконуватиметься до тих пір, поки не буде введено нульове або негативне значення змінної **n**.

Оператор **continue** передає управління в наступну ітерацію циклу, пропускаючи оператори, які записані за ним, причому у вкладеному циклі він передає управління на наступну ітерацію основного циклу.

Конструкція переключення switch…case…end

Для здійснення множинного вибору (або розгалуження) використовується конструкція з перемикачем типу **switch**:

```
switch <Вираз>
   case <Значення Виразу 1>
     Виконувані оператори
   case {<Значення Виразу 2>, <Значення Виразу 3>, ... ,<Значення Виразу k>}
     Виконувані оператори
   ...
   otherwise,
     Виконувані оператори
end
```

Якщо обчислене значення виразу **<Вираз>** після заголовка **switch** співпадає з одним із значень якогось з виразів **<Значення_Виразу_i>** в наступних **case**, то виконується блок операторів **case**, а якщо ні, то – список інструкцій після оператора **otherwise**. При виконанні блоку **case** виконуються ті списки інструкцій, для яких **case <Значення_Виразу_i>** збігається з результатом обчислення **<Вираз>**. Зверніть увагу на те, що **<Значення_Виразу_i>** в **case** може бути числом, константою, змінною, вектором або навіть рядком символів. В останньому випадку оператор **case** дійсний, якщо функція **strcmp** повертає логічне значення «істина».

Пояснимо застосування оператора **switch** на прикладі m-файлу **sw1.m**:

```
switch var
  case {1,2,3}
    disp('Перший квартал')
  case {4,5,6}
    disp('Другий квартал')
  case {7,8,9}
    disp('Третій квартал')
  case {10,11,12}
    disp('Четвертий квартал')
  otherwise
    disp('Помилка в завданні')
end
```

Ця програма у відповідь на значення змінної **var** – номера місяця – обчислює, до якого кварталу ставиться заданий місяць, і виводить відповідне повідомлення:

```
>> var=2;
>> var=2;
Перший квартал
>> var=4;sw1
Другий квартал
>> var=7;sw1
Третій квартал
>> var=12;sw1
Четвертий квартал
>> var=-1;sw1
Помилка в завданні
```

Порядок виконання роботи

1. Ознайомитись з основними інструментами середовища MATLAB. Вивчити операції редагування команд текстового редактора MATLAB.
2. Ознайомитись з способами представлення чисел, системними змінними та константами MATLAB. Навчитись виконувати прості обчислення в командному вікні MATLAB. Навчитись керувати змінними в робочій області (Workspace) MATLAB.
3. Коротко ознайомитись з вбудованими функціями MATLAB та інструментами для створення власних функцій (Editor/Debugger). Написати свою власну функцію для обчислення довільного виразу. В процесі виконання даного пункту звернути увагу на вбудовану систему діагностики помилок та навчитися нею користуватись.
4. Вивчити способи введення масивів та матриць та основні операції над ними. Навчитись працювати з масивами та матрицями в MATLAB.
5. Вивчити засоби побудови графіків в MATLAB. Побудувати графіки декількох функцій, додати до графіків пояснювальні написи, застосувати різні стилі оформлення.
6. Ознайомитись з елементи програмування в MATLAB. Вивчити основні оператори мови програмування MATLAB. Написати функції MATLAB:
    a. для обчислення значень функції заданої за варіантом в табл.4;
    b. для обчислення заданих в табл. 5 суми або добутку.
7. Зробити висновки по роботі.

Таблиця 4. Обчислити значення функції

Вар №	Функція	Вар №	Функція
1.	$F(x)=\begin{cases}x^2-3x+9, & \text{если } x\le 3;\\ \dfrac{1}{x^3+6}, & \text{если } x>3.\end{cases}$	15.	$F(x)=\begin{cases}-x^3+9, & \text{если } x\le 13;\\ -\dfrac{3}{x+1}, & \text{если } x>13.\end{cases}$
2.	$F(x)=\begin{cases}-x^2+3x+9, & \text{если } x\ge 3;\\ \dfrac{1}{x^3-6}, & \text{если } x<3.\end{cases}$	16.	$F(x)=\begin{cases}45x^2+5, & \text{если } x>3,6;\\ \dfrac{5x}{10x^2+1}, & \text{если } x\le 3,6.\end{cases}$
3.	$F(x)=\begin{cases}9, & \text{если } x\le -3;\\ \dfrac{1}{x^2+1}, & \text{если } x>-3.\end{cases}$	17.	$F(x)=\begin{cases}x^4+9, & \text{если } x<3,2;\\ \dfrac{54x^4}{-5x^2+7}, & \text{если } x\ge 3,2.\end{cases}$
4.	$F(x)=\begin{cases}0, & \text{если } x\le 1;\\ \dfrac{1}{x+6}, & \text{если } x>1.\end{cases}$	18.	$F(x)=\begin{cases}1,2x^2-3x-9, & \text{если } x>3;\\ \dfrac{12,1}{2x^2+1}, & \text{если } x\le 3.\end{cases}$
5.	$F(x)=\begin{cases}-3x+9, & \text{если } x\le 7;\\ \dfrac{1}{x-7}, & \text{если } x>7.\end{cases}$	19.	$F(x)=\begin{cases}x^2+3x+9, & \text{если } x\le 3;\\ \dfrac{\sin x}{x^2-9}, & \text{если } x>3.\end{cases}$
6.	$F(x)=\begin{cases}3x-9, & \text{если } x\le 7;\\ \dfrac{1}{x^2-4}, & \text{если } x>7.\end{cases}$	20.	$F(x)=\begin{cases}\cos 2x+9, & \text{если } >-4;\\ -\dfrac{\cos x}{x-9}, & \text{если } x\le -4.\end{cases}$
7.	$F(x)=\begin{cases}x^2, & \text{если } 0\le x\le 3;\\ 4, & \text{если } x>3 \text{ или } x<0.\end{cases}$	21.	$F(x)=\begin{cases}\ln x+9, & \text{если } x>0;\\ -\dfrac{x}{x^2-7}, & \text{если } x\le 0.\end{cases}$
8.	$F(x)=\begin{cases}x^2+4x+5, & \text{если } x\le 2;\\ \dfrac{1}{x^2+4x+5}, & \text{если } x>2.\end{cases}$	22.	$F(x)=\begin{cases}-x^2-1,1x+9, & \text{если } x\le -3;\\ \dfrac{\ln(x+3)}{x^2+9}, & \text{если } x>-3.\end{cases}$
9.	$F(x)=\begin{cases}x^2-x, & \text{если } 0\le x\le 1;\\ x^2-\sin \pi x^2, & \text{если } x>1 \text{ или } x<0.\end{cases}$	23.	$F(x)=\begin{cases}9-x, & \text{если } x>1,1;\\ \dfrac{\sin 3x}{x^4+1}, & \text{если } x<-1,1.\end{cases}$
10.	$F(x)=\begin{cases}-x^2+x-9, & \text{если } x\ge 8;\\ \dfrac{1}{x^4-6}, & \text{если } x<8.\end{cases}$	24.	$F(x)=\begin{cases}-x^2, & \text{если } x\ge 7;\\ \dfrac{2^{-x}}{x^2-9}, & \text{если } x\le 7.\end{cases}$
11.	$F(x)=\begin{cases}4x^2+2x-19, & \text{если } x\ge -3,5;\\ -\dfrac{2x}{-4x+1}, & \text{если } x<3,5.\end{cases}$	25.	$F(x)=\begin{cases}-x^2-9, & \text{если } x\ge 13;\\ -\dfrac{1}{x^2+9}, & \text{если } x\le 13.\end{cases}$

Продовження таблиці 4

12.	$F(x)=\begin{cases}-x^2+3x+9, & \text{если } x\le 3;\\ \dfrac{x}{x^2+1}, & \text{если } x>3.\end{cases}$	26.	$F(x)=\begin{cases}-3x+9, & \text{если } x>3;\\ \dfrac{x^3}{x^2+8}, & \text{если } x\le 3.\end{cases}$
13.	$F(x)=\begin{cases}0, & \text{если } x\le 0;\\ x, & \text{если } 0<x\le 1;\\ x^4, & \text{если } x\ge 1.\end{cases}$	27.	$F(x)=\begin{cases}x+5, & \text{если } x\le 1\\ \left\|2(x-3)^2-2\right\|, & \text{если } 1<x\le 4\\ -\sqrt{x-3}+1, & \text{если } 4<x\le 7\end{cases}$
14	$f(x)=\begin{cases}-\dfrac{1}{x+1}, & \text{если } x<-1\\ \sqrt{1-x^2}, & \text{если } -1\le x\le 0\\ \dfrac{\lvert x\rvert}{x}, & \text{если } x>0\end{cases}$	28.	$f(x)=\begin{cases}-\lvert x+2\rvert, & \text{если } x<-1\\ x^2+1, & \text{если } -1\le x<1\\ -x+3, & \text{если } x\ge 1\end{cases}$

Таблиця 5. Ряди

Вар №	Завдання
1.	Обчислити суму *n* членів ряду $S=1-\frac{1}{2}+\frac{1}{4}-\frac{1}{8}+\cdots+(-1)^n\cdot\frac{1}{2^n}.$
2.	Обчислити суму *N* членів ряду $S=\frac{1}{\sin 1}+\frac{1}{\sin 1+\sin 2}+\cdots+\frac{1}{\sin 1+\sin 2+\cdots+\sin N}.$
3.	Обчислити суму *n* членів ряду $S=\frac{1}{a}+\frac{1}{a^2}+\frac{1}{a^4}+\ldots+\frac{1}{a^{2n-2}}.$ *a* — дійсне число
4.	Обчислити суму *n* членів ряду $S=1\cdot 2+2\cdot 3\cdot 4+\ldots+n\cdot(n+1)\ldots 2n.$
5.	Обчислити добуток *n* членів ряду $P=\left(1-\frac{1}{2^2}\right)\left(1-\frac{1}{3^2}\right)\ldots\left(1-\frac{1}{n^2}\right),$ де $n>2$
6.	Обчислити добуток *n* членів ряду $P=\left(1-\frac{1}{2}\right)\left(1-\frac{1}{4}\right)\left(1-\frac{1}{6}\right)\ldots\left(1-\frac{1}{2n}\right).$
7.	Обчислити суму *n* членів ряду $S=1!+2!+3!+\ldots+n!\ \ (n>1).$

8.	Обчислити суму *n* членів ряду $$S = \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \ldots + \frac{1}{(2n+1)^2}.$$
9.	Обчислити добуток *n* членів ряду $$P = a(a-n)(a-2n)\ldots(a-n^2).$$ *a* — дійсне число
10.	Обчислити добуток *n* членів ряду $$P = a(a+1)\ldots(a+n-1).$$ *a* — дійсне число
11.	Обчислити суму *n* членів ряду $$\ln\frac{x+1}{x-1} = 2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}} = 2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots\right) \quad |x|>1$$
12.	Обчислити суму *n* членів ряду $$e^{-x} = \sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!} = 1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots$$
13.	Обчислити суму *n* членів ряду $$\ln(x+1) = \sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1} = x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}-\ldots \quad -1<x\le 1$$
14.	Обчислити суму *n* членів ряду $$\ln\frac{1+x}{1-x} = 2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1} = 2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots\right) \quad |x|<1$$
15.	Обчислити суму *n* членів ряду $$\ln(1-x) = -\sum_{n=1}^{\infty}\frac{x^n}{n} = -\left(x+\frac{x^2}{2}+\frac{x^4}{4}+\ldots\right) \quad -1\le x<1$$
16.	Обчислити суму *n* членів ряду $$\operatorname{arcctg} x = \frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1} = \frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}-\ldots \quad |x|\le 1$$
17.	Обчислити суму *n* членів ряду $$\operatorname{arctg} x = \frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}\ldots \quad x>1$$
18.	Обчислити суму *n* членів ряду $$\operatorname{arctg} x = \sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)} = x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ldots \quad |x|\le 1$$
19.	Обчислити суму *n* членів ряду

	$\operatorname{Arth} x = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \ldots \quad \lvert x \rvert < 1$

Продовження таблиці 5

20.	Обчислити суму *n* членів ряду $\operatorname{Arth} x = \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = \frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \ldots \quad \lvert x \rvert > 1$
21.	Обчислити суму *n* членів ряду $\operatorname{arctg} x = -\frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \ldots \quad x < -1$
22.	Обчислити суму *n* членів ряду $e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \ldots$
23.	Обчислити суму *n* членів ряду $\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$
24.	Обчислити суму *n* членів ряду $\frac{\sin x}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n+1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} - \ldots$
25.	Обчислити суму *n* членів ряду $\ln x = 2\sum_{n=0}^{\infty} \frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}} = 2\left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{5(x+1)^5} + \ldots\right) \quad x > 0$
26.	Обчислити суму *n* членів ряду $\ln x = \sum_{n=0}^{\infty} \frac{(-1)^n (x-1)^{n+1}}{(n+1)} = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} + \ldots) \quad 0 < x \le 2$
27.	Обчислити суму *n* членів ряду $\arcsin x = x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \ldots \cdot 2n \cdot (2n+1)} = x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} \ldots \quad \lvert x \rvert < 1$
28.	Обчислити суму *n* членів ряду $\arccos x = \frac{\pi}{2} - \left(x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \ldots \cdot (2n-1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \ldots \cdot 2n \cdot (2n+1)}\right) =$ $= \frac{\pi}{2} - \left(x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 8} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} \ldots\right) \quad \lvert x \rvert < 1$

Лабораторна робота № 2

Розв'язання нелінійних рівнянь та їх систем

Мета роботи

Отримати практичні навики використання чисельних методів розв'язання нелінійних рівнянь та їх систем.

Короткі теоретичні відомості до роботи

Будь-яке рівняння з одним невідомим можна записати у вигляді $f(x)=0$. Його розв'язком називається таке значення x^* (корінь рівняння), для якого $f(x^*)=0$. Алгоритми знаходження точного значення коренів відомі тільки для вузького класу рівнянь. Тому більшість їх можливо розв'язати лише наближеними чисельними методами.

Задача знаходження наближеного значення кореня передбачає два етапи:

1) відділення коренів – визначення відрізка $[a,b]$ з області визначення функції $y=f(x)$, де знаходиться тільки один корінь;

2) уточнення наближених коренів, тобто обчислення їх із заданою точністю.

Для кожного з цих етапів розроблені свої чисельні методи.

Метод половинного поділу

Нехай функція $f(x)$ – неперервна на відрізку $[a,b]$, на кінцях його набуває значень різних знаків, тобто $f(a)f(b)<0$, похідна $f'(x)$ зберігає на цьому відрізку знак (рис. 5.1). Отже, усередині цього відрізка міститься один корінь. Ділимо відрізок $[a,b]$ навпіл, знаходимо його середину $c=(a+b)/2$. Якщо $f(c)=0$, то корінь $x^*=c$. Інакше, позначимо через $[a_1,b_1]$ ту половину відрізка $[a,b]$, на кінцях якої функція набуває значень різних знаків. Процес послідовного поділу продовжуємо до того часу, поки на n – кроці не буде виконуватися одна з умов:

1) $f((a_n+b_n)/2)=0$, тоді $x^*=(a_n+b_n)/2$ - шуканий корінь;

2) довжина відрізка, що містить корінь, стане менше ε, де ε -задана точність обчислень, тобто $|b_n-a_n|<2\varepsilon$, а $x^*=(a_n+b_n)/2$.

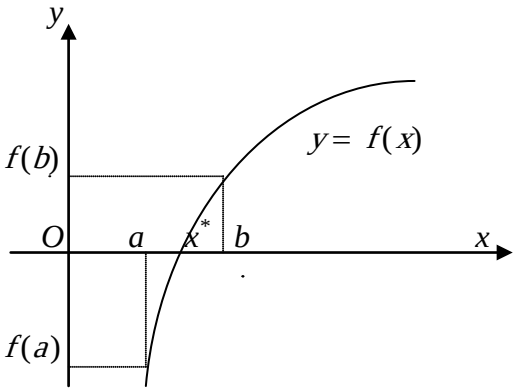


Рис. 5.1. Ілюстрація методу половинного поділу

Метод простої ітерації

Для використання методу простої ітерації (послідовних наближень) замінимо рівняння $f(x)=0$ еквівалентним йому рівнянням

$$x=\phi(x) \tag{5.1}$$

Виберемо деяке наближення $x_0 \in [a,b]$ кореня і підставимо його у праву частину рівняння (5.1). Одержимо $x_1=\varphi(x_0)$. Далі обчислюємо за формулою

$$x_n=\phi(x_{n-1}), n=2,3,... \tag{5.2}$$

Отримуємо послідовність наближень $\{x_n\}$ до кореня, що у випадку її збіжності до кореня x^* може дати наближене його значення із заданою точністю ε. Необхідною і достатньою умовою існування границі послідовності є вимога: $\forall \varepsilon>0,\ N$ такий, що $\forall n>N \Rightarrow |x_n-x_{n-1}|<\varepsilon$. З цієї причини шукаємо наближення (ітерації), які б задовольняли вищезазначену умову.

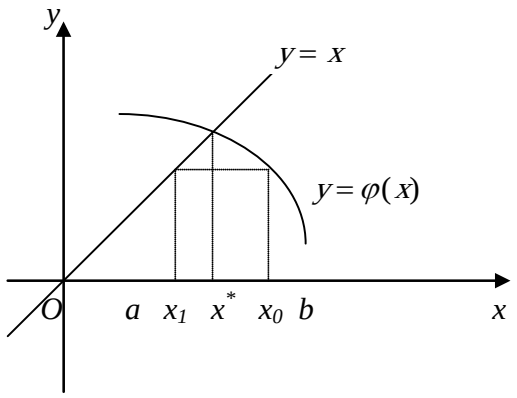


Рис 5.2. Ілюстрація методу простих ітерацій

Перейти від рівняння $f(x)=0$ до еквівалентного йому $x=\phi(x)$ можна багатьма способами. Але оптимальним є той, що задовольнить достатню умову збіжності методу простої ітерації $\forall x \in [a,b]$ $|\phi'(x)|<1$.

При виконанні умови збіжності за початкове наближення x_0 можна взяти довільне значення з інтервалу $[a,b]$.

Метод Хорд

Метод хорд — один з поширених ітераційних методів. Його ще називають методом лінійного інтерполювання, методом пропорційних частин, або методом хибного положення.

Нехай задано рівняння $f(x)=0$, де $f(x)$ на відрізку $[a;b]$ має неперервні похідні першого й другого порядків, які зберігають сталі знаки на цьому відрізку, і $f(a)f(b)<0$, тобто корінь x^* рівняння відокремлений на $[a;b]$.

Ідея методу хорд в тому, що на досить малому відрізку дуга кривої $y=f(x)$ замінюється хордою і абсциса точки перетину хорди з віссю *Ox* є наближеним значенням кореня.

Нехай для визначеності $f'(x)>0$, $f''(x)>0$, $f(a)<0$, $f(b)>0$ (рис. 5,3, а). Візьмемо за початкове наближення шуканого кореня x^* значення $x_0=a$. Через точки A_0 і B_0 проведемо хорду і за перше наближення кореня x^* візьмемо абсцису x_1 точки перетину хорди з віссю *Ox*. Тепер наближене значення x_1 кореня можна уточнити, якщо застосувати метод хорд до відрізка $[x_1;b]$. Абсциса x_2 точки перетину хорди A_1B буде другим наближенням кореня. Продовжуючи цей процес необмежено, дістанемо послідовність $x_0, x_1, x_2, ..., x_k, ...$ наближених значень кореня x^* даного рівняння.

Для виведення формули методу хорд запишемо рівняння прямої, що проходить через точки $A_k(x_k, f(x_k))$ і $B(b, f(b))$:

$$\frac{y - f(x_k)}{f(b) - f(x_k)} = \frac{x - x_k}{b - x_k}.$$

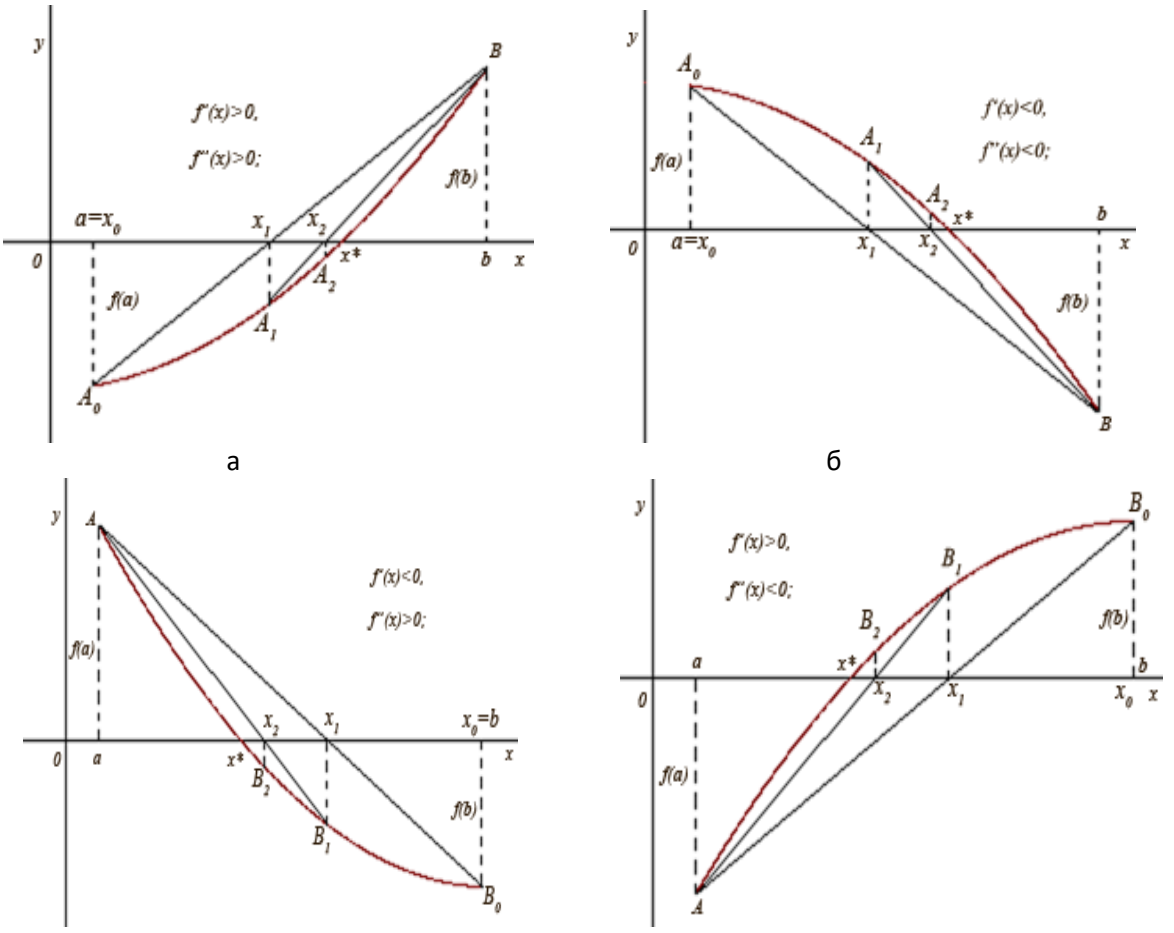


Рис.5.3. Ілюстрація методу хорд

Поклавши $y = 0$, знайдемо абсцису точки перетину хорди $A_k B$ з віссю *Ox*:

$$x = x_k - \frac{f(x_k)}{f(b) - f(x_k)}(b - x_k).$$

Значення x можна взяти за наступне наближення, тобто

$$x_{k+1} = x_k - \frac{f(x_k)}{f(b) - f(x_k)}(b - x_k),$$ тобто *k* = 0,1,2,... .

У цьому разі і тоді, коли $f'(x) < 0$, $f''(x) < 0$, $f(a) > 0$, $f(b) < 0$ (рис. 5.3, б) кінець *b* відрізка $[a;b]$ є нерухомим.

Якщо $f'(x) < 0$, $f''(x) > 0$, $f(a) > 0$, $f(b) < 0$ (рис. 5.3, в), або $f'(x) > 0$, $f''(x) < 0$, $f(a) < 0$, $f(b) > 0$ (рис. 5.3, г), аналогічно можна записати формулу:

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(a)}(x_k - a),$$ тобто *k* = 0,1,2,... .

У цьому випадку точка a є нерухомим кінцем відрізка $[a;b]$.

У загальному випадку нерухомим буде той кінець відрізка ізоляції кореня, в якому знак функції $f(x)$ збігається із знаком другої похідної, а за початкове наближення x_0 можна взяти точку відрізка $[a;b]$, в якій $f(x_0) f''(x_0) < 0$.

Отже, метод хорд можна записати так:

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(c)}(x_k - c)\text{, тобто } k = 0,1,2,.... \tag{5.3}$$

де $c = \begin{cases} a,\ f(a)f''(a) > 0, \\ b,\ f(b)f''(b) > 0. \end{cases}$

З формули (5.3) видно, що метод хорд є методом ітерацій $x_{k+1} = \phi(x_k)$, в якому

$$\phi(x) = x - \frac{f(x)}{f(x) - f(c)(}(x - c) \tag{5.4}$$

Зауважимо, що рівняння $x = x - \frac{f(x)}{f(x) - f(c)(}(x - c)$

на відрізку $[a;b]$ рівносильне рівнянню $f(x) = 0$.

Метод Ньютона

Метод Ньютона (метод дотичних), який використовується для наближеного розв'язку рівняння $f(x) = 0$, полягає в побудові ітераційної послідовності $\{x_n\}, n = 0,1,2,...$, що збігається до кореня рівняння на відрізку $[a,b]$ його локалізації.

На рис. 5.4 зображено спосіб отримання першого наближення за методом дотичних: x_1 є точка перетину дотичної, проведеної до кривої в точці з координатами $(x_0, f(x_0))$. З прямокутного трикутника, гострий кут якого α, маємо

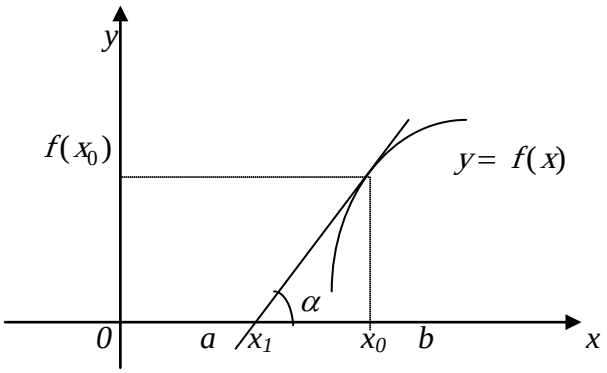


Рис. 5.4. Ілюстрація методу дотичних

$$tg\alpha = f'(x_0) = \frac{f(x_0)}{x_0 - x_1},\ \text{звідки}\ x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Достатні умови збіжності такі. Нехай $f(x)$ - визначена і двічі диференційована на $[a,b]$, причому похідні $f'(x), f''(x)$ зберігають знак на $[a,b]$. Тоді , виходячи з початкового наближення $x_0 \in [a,b]$, що задовольняє нерівність $f(x_0)f''(x_0) > 0,$ ітераційна послідовність

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ \ n = 0,1,2,... \tag{5.5}$$

збігається до єдиного на $[a,b]$ розв'язку ξ рівняння $f(x) = 0.$

Для оцінки похибки *n*-го наближення кореня можна скористатися нерівністю

$$|\xi - x_n| \le \frac{M}{2m}|x_n - x_{n-1}|^2, \tag{5.6}$$

де $M-$ найбільше значення модуля другої похідної $\left|f''(x)\right|$ на $[a,b]$; *m* - найменше значення модуля першої похідної $\left|f'(x)\right|$ на $[a,b]$.

За необхідності обчислити корінь рівняння з точністю ε ітераційну послідовність переривають за умови

$$\left|x_n - x_{n-1}\right| < \sqrt{2m\varepsilon / \mathrm{M}} = \varepsilon_0 \tag{5.7}$$

і беруть x_n за наближене значення кореня ξ.

Метод Ньютона ефективний, якщо вибрано вдале початкове наближення для кореня і навколо кореня графік функції має велику кривину.

Метод простої ітерації і метод Зейделя розв'язання систем рівнянь

Нехай для обчислення невідомих $x_1, x_2, \ldots, x_n$ необхідно розв'язати систему *n* нелінійних рівнянь:

$$\begin{aligned} F_1(x_1, x_2, \ldots, x_n) &= 0, \\ F_2(x_1, x_2, \ldots, x_n) &= 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ F_n(x_1, x_2, \ldots, x_n) &= 0. \end{aligned} \tag{5.8}$$

В векторній формі цю систему можна записати так

$$\mathbf{F}(\mathbf{x}) = 0,$$

де $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}$, $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$.

На відміну від систем лінійних рівнянь не існує прямих методів рішення нелінійних систем загального вигляду. Лише в окремих випадках систему (1) можна розв'язати безпосередньо. Наприклад, для випадку двох рівнянь іноді вдається виразити одне невідоме через інше і таким чином звести завдання до рішення одного нелінійного рівняння відносно одного невідомого.

Для розв'язання систем нелінійних рівнянь зазвичай використовуються ті ж ітераційні методи, що і для розв'язання рівнянь з однією змінною. Нижче будуть розглянуті деякі з цих методів: метод простої ітерації, метод Зейделя і метод Ньютона.

Систему рівнянь (5.8) представимо у вигляді

$$\begin{aligned} x_1 &= f_1(x_1, x_2, \ldots, x_n), \\ x_2 &= f_2(x_1, x_2, \ldots, x_n), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ x_n &= f_n(x_1, x_2, \ldots, x_n). \end{aligned} \tag{5.9}$$

Для розв'язання цієї системи можна використати метод простої ітерації, аналогічний відповідному методу для одного рівняння. Значення невідомих на *k*-й ітерації будуть знайдені з використанням їх значень на попередній ітерації як

$$x_i^{(k)} = f_i\left(x_1^{(k-1)}, x_2^{(k-1)}, \ldots, x_n^{(k-1)}\right), \quad i = 1, 2, \ldots, n. \tag{5.10}$$

Систему (5.9) можна розв'язувати і методом Зейделя, що подібний до методу Гауса-Зейделя розв'язання систем лінійних рівнянь. Значення $x_i^{(k)}$ знаходиться з *i*-го рівняння системи (5.9) з використанням вже вичислених на поточній ітерації значень невідомих. Таким чином, значення невідомих на *k*-й ітерації знаходитимуться не з допомогою (5.10), а з допомогою співвідношення

$$x_i^{(k)} = f_i\left(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i^{(k-1)}, \ldots, x_n^{(k-1)}\right), \quad i = 1, 2, \ldots, n. \tag{5.11}$$

Ітераційний процес в обох методах триває до тих пір, поки зміни усіх невідомих в двох послідовних ітераціях не стануть малими, тобто в якості критерію завершення ітерацій вибирається одна з умов:

$$\left|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right| = \sqrt{\sum_{i=1}^{n}\left(x_i^{(k)} - x_i^{(k-1)}\right)^2} < \varepsilon, \tag{5.12}$$

$$\max_{1 \leqslant i \leqslant n}\left|x_i^{(k)} - x_i^{(k-1)}\right| < \varepsilon, \tag{5.13}$$

$$\max_{1 \leqslant i \leqslant n}\left|\frac{x_i^{(k)} - x_i^{(k-1)}}{x_i^{(k)}}\right| < \varepsilon, \quad \text{при} \quad |x_i| \gg 1. \tag{5.14}$$

де $\varepsilon > 0$ – допустима похибка розв'язку.

Тут в першому випадку відмінність векторів $\mathbf{x}^{(k)}$ і $\mathbf{x}^{(k-1)}$ "на ε" розуміється в сенсі малості модуля їх різниці, в другому – в сенсі малості різниць усіх відповідних компонент векторів, в третьому – в сенсі малості відносних різниць компонент.

При використанні методу простої ітерації і методу Зейделя успіх багато в чому визначається вдалим вибором початкових наближень невідомих: вони мають бути досить близькими до істинного розв'язку. Інакше ітераційний процес може не збігтися.

Порядок виконання роботи

1. Для заданого за варіантом рівняння (табл. 5.1) у виді $F(x) = 0$ побудувати графік функції $y = F(x)$. За графіком відділити інтервали, які містять лише один корінь.
2. Написати функції MATLAB для розв'язання нелінійного рівняння методами простої ітерації, половинного поділу, хорд, Ньютона.
3. Розв'язати задане за варіантом рівняння використовуючи реалізовані функції з точністю $\varepsilon = 10^{-6}$. Порівняти кількість ітерацій, яка була необхідна кожному з методів для розв'язання рівняння з заданою точністю.
4. Розв'язати задане за варіантом рівняння з точністю $\varepsilon = 10^{-6}$ використовуючи стандартну функцію MATLAB `fzero`. Порівняти результат, отриманий `fzero`, з результатами п. 3.
5. Написати функцію MATLAB для розв'язання системи нелінійних рівнянь методом Зейделя.
6. Розв'язати задану за варіантом (табл. 5.2) систему рівнянь використовуючи реалізовану функцію з точністю $\varepsilon = 10^{-6}$. Початкове наближення розв'язку системи можна знайти графічно побудувавши графіки функцій, що входять до системи в одній системі координат (див. приклад нижче).
7. Розв'язати задану за варіантом систему рівнянь з точністю $\varepsilon = 10^{-6}$ використовуючи стандартну функцію MATLAB `fsolve`. Порівняти результат, отриманий `fsolve`, з результатом п. 6.
8. Зробити висновки по роботі.

Варіанти до роботи

Таблиця 5.1

Вар		Вар	
1.	$exp(x)-x-9.5=0$	2.	$10^x+2x-100=0$
3.	$x-1.6\sin(x)+0.25=0$	4.	$x+x^{1/3}-6=0$
5.	$\sin(x)-3x+3.2=0$	6.	$x^2-\ln(1+x)-9=0$
7.	$\sin(x)-x\cos(x)=0$	8.	$x^5-4x^3-8=0$
9.	$\cos(x)-3x=0$	10.	$2exp(-0.5x)-x^2=0$
11.	$exp(x)+2x-25=0$	12.	$12\ln x-x^2-0.6=0$
13.	$sqrt(x)-\cos(sqrt(x))=0$	14.	$(x-1)\sin(x-1)-0.3=0$
15.	$x^3-9x^2+5x+1=0$	16.	$x^{\ln x}-2=0$
17.	$exp(x)-\ln(x)-20=0$	18.	$exp(x\text{-}2)-\ln(x+2)=0$
19.	$x^4+2x^3-x-18=0$	20.	$(exp(x))/(x-1)+2=0$
21.	$\cos(2x)+x-0.8=0$	22.	$arctg(2x)-1/(1+x)=0$
23.	$0.6*1.5^x-2.25x=0$	24.	$sqrt(x)\text{-}tg(1\text{-}x)\text{-}0.4=0$
25.	$2x-x^2+2=0$	26.	$\sin(sqrt(x))-\cos(sqrt(x))+2sgrt(x)=0$
27.	$x^3-sqrt(x)-9.5=0$	28.	$x^3-exp(x)+1=0$
29.	$tg(x^3)-tg(x)+0.5=0$	30.	$x^4+2x^3-x-1=0$

Таблиця 5.2

Вар		Вар	
1.	sin(y + 1)–y = 1,2; 2x+cos y = 2	2.	cos(x – 1) + y = 0,5; x – cos y = 3
3.	sin x+2y=2; cos(y – 1)+x=0,7	4.	cos x + y = 1,5; 2x – sin(y – 0,5) = 1
5.	sin(x + 0,5) – y = 1; cos(y – 2) + x = 0	6.	cos(x + 0,5) + y = 0,8; siny – 2x = 1,6
7.	sin(x – 1) = 1,3–y; x – sin(y + 1) = 0,8	8.	2y – cos(y + 1) = 0; x + siny = –0,4
9.	cos(x + 0,5) – y = 2; sin y – 2x= 1	10.	sin(x + 2) – y = 1,5; x + cos(y – 2) = 0,5
11.	sin(y + 1) – x = 1,2; 2y+cos x = 2	12.	cos(y – 1) + x = 0,5; y – cosy = 3
13.	sin y + 2x = 2; cos(x – 1) + y = 0,7	14.	cos y + x = 1,5: 2y–sin(x – 0,5) = 1
15.	sin(x + 0,5) – x = 1; cos(x – 2) + y = 0	16.	cos(y + 0,5) + x = 0,8; sin x – 2y = 16
17.	sin(y – 1) + x= 1,3; y – sin(x + 1) = 0,8	18.	2x – cos(y + 1) = 0; y + sinx= –0,4
19.	cos(y + 0,5) – x = 2; sinx – 2y = 1	20.	sin(y+2) – x = 1,5; y+cos(x – 2) = 0,5
21.	sin(x + 1) – y = 1; 2x + cosy = 2	22.	cos(x – 1) + y = 0,8; y – cos x = 2
23.	sin x + 2y = 1,6; cos(y– 1) + x = 1	24.	cos x + y = 1,2; 2x – sin(y – 0,5)=2
25.	sin(x + 0,5) – y = 1,2; cos(y – 2) + x = 0	26.	cos(x + 0,5) + y = 1; sin y – 2x = 2
27.	sin(x – 1) + y = 1,5; x – sin(y + 1) = 1	28.	sin(y + 1) – x = 1; 2y + cos x = 2
29.	cos(y – 1) + x = 0,8; y – cos x = 2	30.	cos(x – 1) + y = 1; sin y + 2x = 1,6

Функція розв'язання (знаходження коренів) нелінійного рівняння `fzero`

Функція `fzero` визначає корені нелінійного рівняння виду f(x) = 0

Синтаксис:

`fzero(fun, x)`

повертає уточнене значення `x`, при якому досягається нуль функції `fun`, представленої рядком, при початковому значенні аргументу `x`. Повернене значення близьке до точки, де функція міняє знак, або рівне `NaN`, якщо така точка не знайдена;

`fzero(fun,[x1 x2])`

повертає значення `x`, при якому `fun(x)=0` с завданням інтервалу пошуку за допомогою вектору `x=[x1 x2]`, такого що знак `fun(x(1))` відрізняється від знаку `fun(x(2))`. Якщо це не так, видається повідомлення про помилку. Виклик функції `fzero` з інтервалом гарантує, що `fzero` поверне значення, близьке до точки, де `fun` змінює знак;

`fzero(fun, x, tol)`

повертає результат із заданою похибкою `tol`;

`fzero(fun, x, tol, trace)`

видає на екран інформацію про кожну ітерацію;

`fzero(fun, x, tol, trace, P1, P2,…)`

передбачає додаткові аргументи, що передаються у функцію `fun(x, P1, P2,.)`. При завданні порожньої матриці для `tol` або `trace` використовуються значення за умовчанням, наприклад `fzero(fun, x,[ ],[ ],P1)`.

Для функції `fzero` нуль розглядається як точка, де графік функції `fun` *перетинає* вісь *x*, а не *торкається* її. Залежно від форми завдання функції `fzero` реалізуються наступні добре відомі чисельні методи пошуку нуля функції: ділення відрізку навпіл, січних і зворотної квадратичної інтерполяції.

Приклад:

Знайдемо розв'язок рівняння `0.25*x + sin(x) – 1 = 0;`

Функцію, корені якої необхідно знайти задамо в m-файлі **fun1.m**:

```
%Функція, корені якої шукаються
function f=fun1(x)
  f=0.25*x+sin(x)-1;
```

Будуємо графік функції (рис.5.5) для приблизного визначення інтервалів, що містять корені:

```
>> x=0:0.1:10; plot(x, fun1(x)); grid on;
```

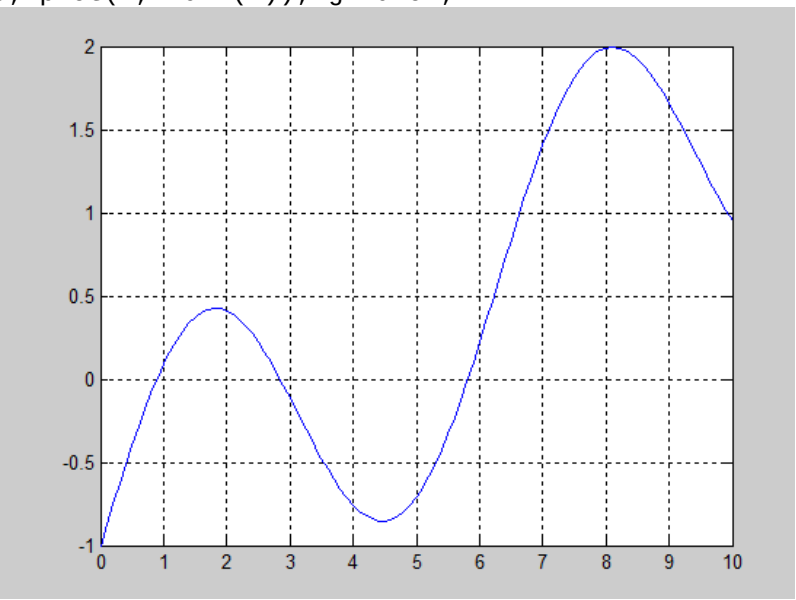


Рис. 5.5. Графік `f=0.25*x+sin(x)-1`

З рисунку видно, що значення коренів необхідно шукати на інтервалах [0.5; 1], [2; 3] и [5; 6]. Знайдемо їх, використовуючи функцію `fzero`:

```
>> x1=fzero(@fun1,[0.5 1])
x1 = 0.8905
>> x2=fzero(@fun1,[2 3])
x2 = 2.8500
>> x3=fzero(@fun1,[5,6])
x3 = 5.8128
>> x3=fzero(@fun1,5,0.001)
x3 = 5.8111
```

Тут корінь `x3` знайдено двома способами. В другому випадку задана похибка обчислень `tol=0.001`.

Функція розв'язання систем нелінійних рівнянь

Для розв'язання систем нелінійних рівнянь може бути використана функція `fsolve`. Функція `fsolve` намагається розв'язати систему нелінійних рівнянь виду **F**(**X**) = 0, де **F** і **X** можуть бути вектором або матрицею.

Синтаксис:

`fsolve(fun, X0)`

Бере за початкове наближення вектор (матрицю) `X0` і намагається розв'язати систему рівнянь **F**(**X**) = 0. Вектор (матрицю) лівих частин рівнянь **F**(**X**) необхідно задати в m-файлі `fun.m`, оформленому у вигляді функції `fun`, що приймає на вхід вектор (матрицю) **X** і повертає **F**(**X**).

`X = fsolve(fun, X0, options)`

Розв'язує систему рівнянь але параметри розв'язання за замовчуванням замінюються параметрами із структури `options`; для створення цієї структури використовується функція `optimset` (див. допомогу MATLAB). Зокрема можна задати точність обчислень і вказати необхідність вивести на екран всі ітерації пошуку коренів: `optimset('Display','iter','TolX',0.001)`.

Приклад:

Знайдемо розв'язок системи рівнянь

$2*x_1 - x_2 - \exp(-x_1) = 0;$

$-x_1 + 2*x_2 - \exp(-x_2) = 0.$

Систему, яку необхідно розв'язати задамо в m-файлі my**fun.m**:

```
%Система, що розв'язується
function F=myfun(x)
  F = [ 2*x(1) - x(2) - exp(-x(1))
       -x(1) + 2*x(2) - exp(-x(2))];
```

Для системи двох рівнянь з двома невідомими розв'язок системи можна знайти графічно. Розв'язком системи буде точка, в якій перетинаються графіки рівнянь системи. Побудуємо графіки функцій, що входять до системи (рис.5.5) для приблизного визначення початкового наближення розв'язку. Використаємо функцію, яка будує графіки функцій, заданих неявно (тобто де функція не виражена через її аргумент):

```
>> ezplot('2*x1 - x2 - exp(-x1)');hold on;ezplot('-x1 + 2*x2 - exp(-x2)');
```

З рисунку видно, що за початкове значення можна взяти x0 = [0; 0]. Знайдемо розв'язок, використовуючи функцію `fsolve`:

```
>> x0 = [0 0];
>> options=optimset('Display','iter','TolX',0.001);
>> [x,fval] = fsolve(@myfun,x0,options)

                                        Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)          step         optimality    radius
     0          3              2                          2             1
     1          6       0.0226976       0.707107          0.171         1
     2          9     3.40349e-006      0.0937779         0.00204      1.77
     3         12     7.72066e-014     0.00117685       3.08e-007      1.77
```

```
Equation solved.


x =

    0.5671
    0.5671


fval =

  1.0e-006 *

   -0.1965
   -0.1965
```

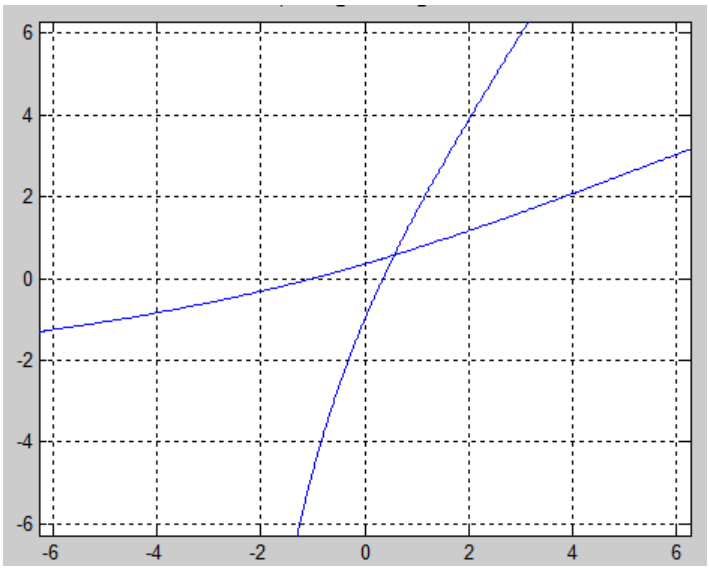


Рис. 5.6. Графік системи рівнянь

Отже розв'язок: $x_1 = 0.5671$, $x_2 = 0.5671$.

Приклади програм

Метод простих ітерацій. Дозволяє отримати наближений розв'язок нелінійного рівняння $F(x) = 0$, представленого у виді $x = g(x)$ ($g(x) = x - \tau F(x)$). Обчислення починаються з початкового наближення p_0 і продовжуються за ітераційним правилом $p_{n+1} = g(p_n)$.

```
function [k, p, err, P]=iter(g, p0, tol, maxI)
% Вхід - g - ітераційна функція, вводиться як рядок 'g'
%        p0 - початкове значення кореня
%        tol - допустиме відхилення
%        maxI - максимальне число ітерацій
%Вихід - k - число виконаних ітерацій
%        p - наближене значення кореня рівняння
%        err - похибка наближення
%        P - послідовність {pi}
P(1)= p0;
for k=2:maxI
    P(k)=feval(g,P(k-1));
    err=abs(P(k)-P(k-1));
    relerr=err/(abs(P(k))+eps);
    p=P(k);
    if ((err<tol) || (relerr<tol))
        break;
    end
end
if (k == maxI)
    disp('максимально допустимое число ітерацій')
end
P=P';
```

Метод половинного поділу. Дозволяє отримати наближений розв'язок нелінійного рівняння $F(x) = 0$ на інтервалі $[a, b]$.

```
function [c,err,yc]=bisect(f,a,b,delta)
%Ввод - f -  функція, вводиться як рядок 'f'
%       a i b - ліва и права крайні точки
%       delta - допустиме відхилення
%Вихід - c - розв'язок
%       yc=f(c)
%       err - похибка c

ya=feval(f,a);
yb=feval(f,b);
if (ya*yb>0)
    return
end
maxI=1+round((log(b-a)-log(delta))/log(2));
for k=1:maxI
    c=(a+b)/2;
    yc=feval(f,c);
    if (yc==0)
        a=c;
        b=c;
    elseif yb*yc>0
        b=c;
        yb=yc;
    else
        a=c;
        ya=yc;
    end
    if (b-a < delta)
        break;
    end
end
```

```
c=(a+b)/2;
err=abs(b-a);
yc=feval(f,c);
```

Метод Хорд. Дозволяє отримати наближений розв'язок нелінійного рівняння $F(x) = 0$ на інтервалі $[a, b]$.

```
function [c,err,yc]=regula(f,a,b,delta,epsilon,maxI)
%Ввод - f - функція, вводиться як рядок 'f'
%       a i b - ліва и права крайні точки
%       delta - допустиме відхилення  для кореня
%       epsilon - допустиме відхилення для значения f в корені
%       maxI - максимальне число ітерацій
%Вихід - c - нуль
%       yc=f(c)
%       err - похибка обчислення c
ya=feval(f,a);
yb=feval(f,b);
if ya*yb>0
    disp('Замечание: f(a)*f(b)>0>'),
    return;
end
for k=1:maxI
    dx=yb*(b-a)/(yb-ya);
    c=b-dx;
    ac=c-a;
    yc=feval(f,c);
    if yc==0
        break;
    elseif yb*yc>0
        b=c;
        yb=yc;
    else
        a=c;
        ya=yc;
    end
    dx=min(abs(dx),ac);
    if abs(dx)<delta
        break;
    end
    if abs(yc)<epsilon
        break;
    end
end
c;
err=abs(b-a)/2;
yc=feval(f,c);
```

Метод Ньютона. Дозволяє отримати наближений розв'язок нелінійного рівняння $F(x) = 0$ з початковим наближенням p_0, використовуючи ітерацію

$$p_k = p_{k-1} - \frac{f(p_{k-1})}{f'(p_{k-1})} \quad \text{для } k = 1, 2, \ldots.$$

```
function [p0, err, k, y]=newton(f, df, p0, delta, epsilon, maxI)
% Вхід - f - функція, вводиться як рядок 'f'
%       df - похідна f, вводиться як рядок 'df'
%       p0 - початкове наближення кореня функции f
%       delta - допустиме відхилення для p0
%       epsilon - допустимое відхилення для значення функції f(p0)
%       maxI - максимальне число ітерацій
%Вихід - p0 - наближення кореня за методом Ньютона
%       err - похибка обчислення для p0
%       k - число ітерацій
```

```
%       У - значення функції f(p0)
for k=1:maxI
    p1=p0-feval(f,p0)/feval(df,p0);
    err=abs(p1-p0);
    relerr=2*err/(abs(p1)+delta);
    p0=p1;
    y=feval(f,p0);
    if (err<delta) || (relerr<delta) || (abs(y)<epsilon)
        break;
    end
end
```

Метод Зейделя. Розв'язання нелінійної системи рівнянь, заданої у виді $X = G(X)$, із заданим початковим наближенням P_0 і генеруванням послідовності $\{P_k\}$, яка сходиться до рішення P.

```
function [P,iter] = seidel(G,P,delta,maxI)
%Вхід - G - нелінійна система, записана в формі М-файлу G.m
%       P - начальне наближення для розв'язку
%       delta - грань похибки (точність)
%       maxI - число ітерацій
%Вихід - P - наближення Зейделя до розв'язку
%       iter - число виконаних ітерацій
N=length(P);
for k=1:maxI
    X=P;
    % X k-e наближення до розв'язку
    for j=1:N
        A=feval(G,X);
        % Вивід членів X по мірі їх обчислення
        X(j)=A(j);
    end
    err=abs(norm(X-P));
    relerr=err/(norm(X)+eps);
    P=X;
    iter=k;
    if (err<delta)||(relerr<delta)
        break
    end
end
```

Лабораторна робота № 3

Розв’язання систем лінійних алгебраїчних рівнянь

Мета роботи

Отримати практичні навики використання чисельних методів розвзання систем лінійних алгебраїчних рівнянь.

Короткі теоретичні відомості до роботи

Розглянемо системи вигляду

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = c_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = c_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = c_n. \end{cases} \qquad (4.1)$$

Її матричний вигляд

$$AX=C. \qquad (4.2)$$

Тут A – $\{[\, a_{ij} \,],(i,j=\overline{1,n})\}$ - матриця коефіцієнтів системи,

$$C=\begin{bmatrix} c_1 \\ c_2 \\ \ldots \\ c_n \end{bmatrix}, \quad X=\begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}$$ - вектори-стовпці.

Методи чисельного розв’язання систем лінійних алгебраїчних рівнянь (СЛАР) поділяються на точні і наближені. Метод вважають точним, якщо, нехтуючи похибками округлення, він дає точний результат після виконання певної кількості обчислювальних операцій. Математичні пакети прикладних програм для ПЕОМ містять стандартні процедури розв’язання СЛАР такими поширеними точними методами, як метод Гаусса. До наближених методів розв’язання СЛАР відносять метод простої ітерації та метод Зейделя. Вони дозволяють отримати послідовність $\left\{X^k\right\}$ наближень до розв’язку X^*, таку, що $\lim\limits_{k\to\infty} X^k = X^*$.

Метод Гаусса

Усі чисельні методи розв'язування СЛАР спираються на деякі перетворення вихідної системи рівнянь, які, з одного боку, не змінюють шуканого розв’язання, а, з іншого - значно спрощують подальше його відшукування. Звичайно, при цьому домагаються, щоб загальна кількість операцій разом із попередніми перетвореннями була якомога меншою у порівнянні з методом Крамера.

Розглянемо, які ж саме перетворення рівнянь (4.1) не змінюють розв'язків цієї системи. Неважко зрозуміти, що до таких перетворень належать:

1) ділення будь-якого з рівнянь на будь-яке число *s*, що не дорівнює нулю; така операція не змінює розв'язків СЛАР, але змінює визначник матриці, який зменшується у стільки ж разів, тобто визначник початкової матриці ***A*** дорівнюватиме значенню визначника нової матриці коефіцієнтів, помноженому на *s*;
2) додавання до будь-якого рівняння системи (4.1) будь-якого іншого рівняння тієї ж системи, помноженого на будь-яке число, що не дорівнює нулю;
3) переставлення місцями двох яких-небудь рівнянь; ця операція призводить до зміни знака визначника системи.

Якщо ввести у розгляд так звану "розширену матрицю" коефіцієнтів рівнянь системи

$$A^* = \begin{bmatrix} A & B \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{bmatrix}, \qquad (4.3)$$

яка матиме розміри $n*(n+1)$, то вищезгадані перетворення можна інтерпретувати як перетворення саме розширеної матриці коефіцієнтів СЛАР. До таких допустимих перетворень цієї матриці можна, як це було зазначено, віднести:

1) множення будь-якого рядка матриці на число, що не дорівнює нулю;
2) підсумовування будь-якого рядка матриці з іншим рядком цієї матриці, помноженим на число, що не дорівнює нулю;
3) переставлення місцями двох будь-яких рядків розширеної матриці; при цьому знак головного визначника матриці ***A*** змінюється на протилежний;
4) переставляння місцями двох будь-яких стовпців матриці ***A***; ця операція є еквівалентною лише змінюванню позначень шуканих змінних; при цьому також змінюється знак визначника матриці ***A***.

Схема єдиного ділення

Найпростіший варіант методу Гаусса називається схемою єдиного ділення. Розглянемо його докладніше.

Схема єдиного ділення складається із двох етапів. На першому з них (його називають прямим ходом) вихідні рівняння (4.1) перетворюються таким чином, що з наступних рівнянь вилучаються усі попередні змінні, тобто із другого і подальших рівнянь вилучається змінна x_1, з третього і подальших - змінна x_2 і так далі. У результаті таких дій випливає, що останнє рівняння міститиме лише одну змінну – x_n, передостаннє - дві змінні - x_n і x_{n-1} і так далі у порядку зростання кількості змінних.

На другому етапі (який називається зворотним ходом) визначаються шукані розв'язки СЛАР. Значення змінної x_n визначається безпосередньо з останнього одержаного рівняння, значення x_{n-1} – з передостаннього (із урахуванням відшуканого значення x_n) і так далі.

Розглянемо докладніше прямий хід.

Припускаючи, що a_{11} не дорівнює нулю, поділимо на нього перше рівняння (4.1). Одержимо перше рівняння у вигляді

$$x_1 + a_{12}^{(1)} \cdot x_2 + a_{13}^{(1)} \cdot x_3 + a_{14}^{(1)} \cdot x_4 + \dots + a_{1n}^{(1)} \cdot x_n = b_1^{(1)},$$

де використане позначення

$$a_{1i}^{(1)} = \frac{a_{1i}}{a_{11}}, \ (i = 2,3,\dots,n); \ b_1^{(1)} = \frac{b_1}{a_{11}},$$

а індекс угорі позначає номер кроку прямого ходу.

Тепер виключимо із другого рівняння (4.1) змінну x_1. Для цього помножимо перетворене перше рівняння (4.1) на коефіцієнт a_{21} і віднімемо його від другого рівняння (4.1). Одержимо друге рівняння у вигляді

$$a_{22}^{(1)} \cdot x_2 + a_{23}^{(1)} \cdot x_3 + a_{24}^{(1)} \cdot x_4 + \dots + a_{2n}^{(1)} \cdot x_n = b_2^{(1)},$$

де позначено

$$a_{2i}^{(1)} = a_{2i} - a_{21} \cdot a_{1i}^{(1)}, \qquad (i = 2,3,...,n);$$
$$b_2^{(1)} = b_2 - a_{21} \cdot b_1^{(1)}.$$

Так само перетворюються усі подальші рівняння. Після цього вони набудуть вигляду (k -номер рівняння):

$$a_{k2}^{(1)} \cdot x_2 + a_{k3}^{(1)} \cdot x_3 + a_{k4}^{(1)} \cdot x_4 + ... + a_{kn}^{(1)} \cdot x_n = b_k^{(1)},$$

причому

$$a_{ki}^{(1)} = a_{ki} - a_{k1} \cdot a_{1i}^{(1)}, (i = 2,3,...,n); \; b_k^{(1)} = b_k - a_{k1} \cdot b_1^{(1)}. \qquad (4.4)$$

У результаті першого кроку прямого ходу система рівнянь (4.1) набуває вигляду (5). При цьому усі рівняння системи, починаючи із другого, матимуть на одну змінну менше за вихідну систему (4.1), тобто у сукупності утворюють СЛАР ($n-1$)-го порядку.

$$\begin{cases} x_1 + a_{12}^{(1)} x_2 + ... + a_{1n}^{(1)} x_n = b_1^{(1)}, \\ a_{22}^{(1)} x_2 + ... + a_{2n}^{(1)} x_n = b_2^{(1)}, \\ ... \quad ... \quad ... \quad ... = ..., \\ a_{n2}^{(1)} x_2 + ... + a_{nn}^{(1)} x_n = b_n^{(1)}. \end{cases} \qquad (4.5)$$

Другий крок прямого ходу методу Гаусса полягає у аналогічному перетворенні СЛАР ($n-1$)-го порядку, яку складають одержані рівняння (5) з другого по останнє. У результаті виходить така система рівнянь:

$$\begin{cases} x_1 + a_{12}^{(1)} x_2 + a_{13}^{(1)} x_3 + ... a_{1n}^{(1)} x_n = b_1^{(1)}, \\ x_2 + a_{23}^{(2)} x_3 + ... a_{2n}^{(2)} x_n = b_2^{(2)}, \\ ... \quad ... \quad ... = ..., \\ a_{n3}^{(2)} x_3 + ... a_{nn}^{(2)} x_n = b_n^{(2)}. \end{cases} \qquad (4.6)$$

Коефіцієнти визначаються аналогічно:

$$a_{2i}^{(2)} = \frac{a_{2i}^{(1)}}{a_{22}^{(1)}}, \quad b_2^{(2)} = \frac{b_2^{(1)}}{a_{22}^{(1)}},$$
$$a_{ki}^{(2)} = a_{ki}^{(1)} - a_{k2}^{(1)} \cdot a_{2i}^{(2)},$$
$$b_k^{(2)} = b_k^{(1)} - a_{k2}^{(1)} \cdot b_2^{(2)}; (k, i = 3,4,...,n). \qquad (4.7)$$

У такий само спосіб здійснюються подальші кроки прямого ходу. Формули перетворення на m -му кроці визначаються співвідношеннями:

$$a_{mi}^{(m)} = \frac{a_{mi}^{(m-1)}}{a_{mm}^{(m-1)}}, \; b_m^{(m)} = \frac{b_m^{(m-1)}}{a_{mm}^{(m-1)}}, \; a_{ki}^{(m)} = a_{ki}^{(m-1)} - a_{k2}^{(m-1)} \cdot a_{mi}^{(m)},$$
$$b_k^{(m)} = b_k^{(m-1)} - a_{km}^{(m-1)} \cdot b_m^{(m)}; (k, i = m+1, m+2,..., n). \quad (4.8)$$

У підсумку за $m-1$-м кроком утворюється така система рівнянь:

$$\begin{cases} x_1 + a_{12}^{(1)} x_2 + a_{13}^{(1)} x_3 + a_{14}^{(1)} x_4 + ... a_{1n}^{(1)} x_n = b_1^{(1)}, \\ x_2 + a_{23}^{(2)} x_3 + a_{24}^{(2)} x_4 + ... a_{2n}^{(2)} x_n = b_2^{(2)}, \\ x_3 + a_{34}^{(3)} x_4 + ... a_{3n}^{(3)} x_n = b_2^{(3)}, \\ ... \quad ... \quad = ..., \\ x_n = b_n^{(n)}. \end{cases} \qquad (4.9)$$

Матриця $A_{(n-1)}$ коефіцієнтів цієї системи є верхньою трикутною матрицею з одиничними елементами вдовж головної діагоналі:

$$A_{(n-1)} = \begin{bmatrix} 1 & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & 1 & a_{23}^{(2)} & a_{24}^{(2)} & \dots & a_{2n}^{(2)} \\ 0 & 0 & 1 & a_{34}^{(3)} & \dots & a_{3n}^{(3)} \\ 0 & 0 & 0 & 1 & \dots & a_{4n}^{(4)} \\ 0 & 0 & 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \quad (4.10)$$

Відповідна до цієї системи розширена матриця коефіцієнтів має вигляд

$$A_{(n-1)}^{*} = \begin{bmatrix} 1 & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} & \dots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & 1 & a_{23}^{(2)} & a_{24}^{(2)} & \dots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & 1 & a_{34}^{(3)} & \dots & a_{3n}^{(3)} & b_3^{(3)} \\ 0 & 0 & 0 & 1 & \dots & a_{4n}^{(4)} & b_4^{(4)} \\ 0 & 0 & 0 & 0 & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 1 & b_n^{(n)} \end{bmatrix}. \quad (4.11)$$

Підсумовуючи, можна сказати, що основною метою прямого ходу методу Гаусса є перетворення розширеної матриці системи до трикутної форми (11), після чого відшукання розв'язків СЛАР легко здійснюється зворотним ходом за співвідношеннями:

$$x_n = b_n^{(n)}; x_m = b_m^{(m)} - \sum_{j=m+1}^{n} a_{mj}^{(m)} \cdot x_j \text{ , } (m = n-1, n-2, \dots, 1). \quad (4.12)$$

Таким чином, прямий хід методу Гаусса зводиться до побудови розширеної матриці системи (1.16) і подальшого її перетворення до верхньої трикутної форми за допомогою таких операцій:

1) ділення елементів першого рядка матриці на перший елемент цього рядка (який міститься на головній діагоналі); цей елемент називається роздільним;

2) віднімання з подальших рядків матриці першого рядка, помноженого на елемент відповідного рядка, що знаходиться у тому зі стовпців, що й роздільний елемент; обнуління елементів, які містяться у стовпці роздільного елемента;

3) повторення цих дій щодо другого рядка, а потім і для усіх подальших рядків нових одержаних матриць.

Метод Гаусса з обранням роздільного елемента

Неважко помітити суттєвий недолік розглянутого методу єдиного ділення, який випливає з того, що однією з необхідних операцій є ділення на роздільний елемент. Якщо на головній діагоналі перед цим виявиться, що роздільний елемент дорівнює нулю, відповідна операція стає неможливою. Якщо ж роздільний елемент виявиться досить малою за абсолютним значенням величиною, то хоч операція ділення при цьому є здійсненною, але вона приводить до значної похибки у визначенні подальших коефіцієнтів - членів розширеної матриці. Тому необхідно здійснювати заходи з уникнення цього явища.

У методі Гаусса з обранням роздільного елемента цього досягають у такий спосіб. Перед діленням рядка на роздільний елемент порівнюють за модулем усі елементи відповідного стовпця матриці, відшукують той рядок (із подальших), де міститься елемент, найбільший за модулем, і змінюють місцями поточний рядок із знайденим рядком із максимальним елементом. Таким чином домагаються, щоб роздільний елемент був якомога більшим за модулем.

При обміні місцями рядків визначник матриці ***A*** коефіцієнтів змінює свій знак на протилежний. Тому якщо завданням є обчислення у подальшому визначника цієї матриці, необхідно лічити кількість *p* таких обмінів рядків місцями. Тоді визначник початкової матриці ***A*** може бути обчислений як значення визначника перетвореної матриці $A_{(n-1)}$, помножене на $(-1)^p$.

Метод простої ітерації

Розглянемо СЛАР у матричному вигляді (4.1) (діагональні коефіцієнти a_{ii} не дорівнюють нулю для всіх *i*). Приведемо її до вигляду *X=BX+D*, де *B*=[b_{ij}] - квадратна матриця порядку *n*:

$$b_{ii} = 0; \quad b_{ij} = \frac{-a_{ij}}{a_{ii}}, \quad d_i = \frac{c_i}{a_{ii}}, \quad i, j = \overline{1,n}.$$

При цьому СЛАР (4.1) набуде вигляду

$$\begin{cases} x_1 = b_{12}x_2 + b_{13}x_3 + ... + b_{1n}x_n + d_1, \\ x_2 = b_{21}x_1 + b_{23}x_3 + ... + b_{2n}x_n + d_2, \\ ....., \\ x_n = b_{n1}x_1 + b_{n2}x_2 + ... + b_{nn-1}x_{n-1} + d_n. \end{cases} \qquad (4.13)$$

Взявши довільне початкове наближення $X^0 = col(x_1^0, ...x_n^0)$, будуємо ітераційний процес за формулою $X^k = BX^{k-1} + D$ (*k=1,2,…*).

Метод Зейделя

Ітераційний процес Зейделя відрізняється від методу простої ітерації тим, що при розв'язуванні систем вигляду *X=BX+D* обчислення наступного наближеного значення x_i при *1<i<n* використовується при обчисленні раніше невідомих наближених значень $x_1, x_2, ..., x_{i-1}$.

Розглянемо тепер систему *AX=C* (2) з *n* рівнянь із *n* невідомими, як і раніше припускаючи, що діагональні коефіцієнти a_{ii} не дорівнюють нулю для всіх *i.* Перетворимо вихідну систему вигляду *AX=C* до вигляду *X=BX+D*, де *B*=[b_{ij}] - квадратна матриця порядку *n*:

$$b_{ii} = 0; \quad b_{ij} = \frac{-a_{ij}}{a_{ii}}, \quad d_i = \frac{c_i}{a_{ii}}, \quad i, j = \overline{1,n}.$$

У цьому випадку ітераційний процес методу Зейделя має вигляд

$$\begin{cases} x_1^{k+1} = \sum_{j=2}^{n} b_{1j} \cdot x_j^k + d_1, \\ x_2^{k+1} = b_{21} \cdot x_1^{k+1} + \sum_{j=3}^{n} b_{2j} \cdot x_j^k + d_2, \\ ....., \\ x_n^{k+1} = \sum_{j=1}^{n-1} b_{nj} \cdot x_j^{k+1} + d_n. \end{cases} \qquad (4.14)$$

Часто в практичних обчисленнях ітераційний процес припиняють, якщо два послідовних наближення відрізняються менше від наперед заданого числа Δ у змісті обраної норми:

$$\left\| X^{k+1} - X^k \right\| < \Delta,$$

тобто коли наближені розв'язки x_i^{k+1} і x_i^k $(i = \overline{1,n})$ стануть досить близькими і $\max_i \left| x_i^{k+1} - x_i^k \right| < \Delta.$

Величина Δ пов'язана з точністю ε розв'язання системи таким співвідношенням: $\Delta = \frac{1 - \|B\|}{\|B\|} \cdot \varepsilon$, де $\|B\|$ - норма матриці з коефіцієнтів при невідомих у правих частинах рівнянь перетвореної системи*: X=BX+D*. Норма матриці може бути визначена по-різному, наприклад, як

$$\|B\| = \max_i \sum_{j=1}^{n} |b_{ij}|, \text{ або } \|B\| = \max_j \sum_{i=1}^{n} |b_{ij}|.$$

Порядок виконання роботи

1. Написати функцію MATLAB для розв’язання заданої системи лінійних алгебраїчнх рівнянь відповідно до варіанту (табл. 4.1) методом Гаусса.
2. Написати функцію MATLAB для розв’язання заданої системи лінійних алгебраїчнх рівнянь відповідно до варіанту методом простих ітерацій.
3. Написати функцію MATLAB для розв’язання заданої системи лінійних алгебраїчнх рівнянь відповідно до варіанту методом Зейделя.
4. Зробити висновки по роботі.

Варіанти до роботи

Таблиця 4.1

Вар	$(A\|b)$	Вар	$(A\|b)$
1.	0.12 -0.43 0.14 -0.17 -0.07 0.34 -0.72 0.62 1.18 -0.08 -0.25 1.12	2.	1.17 0.53 -0.84 1.15 0.64 -0.72 -0.43 0.15 0.32 0.43 -0.93 -0.48
3.	1.6 0.12 0.57 0.18 0.38 0.25 -0.54 0.64 0.28 0.46 -1.12 0.88	4.	1.16 1.3 -1.14 0.43 0.83 -0.48 -2.44 -0.15 2 -0.16 1.3 1.5
5.	1.2 -0.2 0.3 -0.6 0.2 1.6 -0.1 0.3 -0.3 0.1 -1.5 0.4	6.	0.3 1.2 -0.2 -0.6 -0.1 -0.2 -0.3 0.3 -1.5 -0.3 0.1 40
7.	6.36 11.75 10 -41.7 7.42 19.03 11.75 -49.49 5.77 7.42 6.36 -27.67	8.	0.18 0.25 -0.44 1.15 0.42 -0.35 1.12 0.86 1.14 0.12 -0.83 0.68
9.	0.64 -0.43 0.67 0.43 0.56 0.12 -0.27 0.88 0.63 -0.83 0.43 -0.12	10.	0.75 -0.84 1.11 0.66 1.12 -0.14 0.45 0.15 0.32 0.23 -0.48 0.14
11.	1.06 -0.28 0.84 0.57 0.43 0.62 -0.35 0.66 0.37 -0.75 -0.64 -0.38	12.	1.6 2.18 -0.72 1.15 0.43 -0.16 0.53 0.83 0.34 0.57 -0.83 -0.42
13.	0.4 0.11 0.18 0.47 0.28 -0.59 0.03 0.01 0.02 0.24 0.1 0.22	14.	3.16 -1.72 -1.23 2.15 0.72 0.67 1.18 1.43 2.57 -1.34 -0.68 1.03
15.	0.4 0.11 0.18 0.47 0.28 -0.59 0.03 0.01 0.02 0.24 0.1 0.22	16.	1.53 -1.63 -0.76 2.18 0.86 0.17 1.84 1.95 0.32 -0.66 1.11 -0.47
17.	1.73 -0.83 1.82 0.36 0.27 0.53 -0.64 1.23 0.56 -0.48 1.95 -0.76	18.	0.62 0.56 -0.43 1.16 1.32 -0.88 1.76 2.07 0.73 1.42 -0.34 2.18
19.	1.06 0.34 1.26 1.17 2.54 -1.16 0.55 2.23 1.34 -0.47 -0.83 3.26	20.	2.48 1.75 -3.24 1.23 1.75 -1.16 2.18 3.43 -3.24 2.18 -1.85 -0.16
21.	3.23 1.62 0.65 1.28 1.62 -2.33 -1.43 0.87 0.65 -1.43 2.18 -2.67	22.	1.42 -2.15 1.07 2.48 -2.16 0.76 -2.18 1.15 1.07 -2.18 1.23 0.88
23.	0.53 -0.75 1.83 0.68 -0.75 0.68 -1.19 0.95 1.83 -1.19 2.15 1.27	24.	1.65 -1.76 0.77 2.15 -1.76 1.04 -2.61 0.82 0.77 -2.61 3.18 -0.73
25.	0.64 1.05 -2.03 1.18 1.05 -1.41 0.16 0.27 -2.93 0.16 -1.51 0.72	26.	1.54 -0.75 1.36 2.45 -0.75 0.87 -0.79 1.07 1.36 -0.79 0.64 0.54
27.	0.77 -0.38 0.68 1.22 -0.38 0.43 -0.39 0.33 0.68 -0.39 0.32 0.27	28.	3.08 -1.5 2.72 4.9 -1.5 1.72 -1.58 2.14 2.72 -1.58 1.28 1.08

Метод Гаусса (прямий хід). Щоб знайти розв'язок системи $AX = B$, спочатку приводимо розширену матрицю $[A|B]$ до верхньої трикутної форми, а потім виконуємо обернену підстановку.

```
function X = gauss(A, B)
%Вхід  - A - невирождена матриця разміру N x N
%      - B - матриця разміру N x 1
%Вихід - X - матриця разміру N x 1, що містить розв'язок AX=B
%Ініціалізація X і тимчасове збереження матриці C

[N N]=size(A);
X=zeros(N,1);
C=zeros(1,N+1);

%Вид расширеної матриці: Aug=[A|B]
Aug=[A B];
for p=1:N-i
    %Частиний вибір головного елементу для стовпця p
    [Y,j]=max(abs(Aug(p:N,p)));
    %Меняємо місцями рядки p і j
    C=Aug(p,:);
    Aug(p,:)=Aug(j+p-1,:);
    Aug(j+p-1,:)=C;
    if Aug(p,p)==0
        'A вирождена. Немає єдиного рішения'
        break
    end

    %Процес виключення для стовбця p
    for k=p+1:N
        m=Aug(k,p)/Aug(p,p);
        Aug(k,p:N+1)=Aug(k,p:N+1)-m*Aug(p,p:N+1);
    end
end

%Обернена підстановка у [U|Y] з використанням підпрограми backsub
X=backsub(Aug(1:N,1:N),Aug(1:N,N+1));
```

Метод Гаусса (зворотній хід). Розв'язання верхньої трикутної системи лінійних рівнянь $AX = B$ методом зворотної підстановки. Діагональні елементи не рівні нулю. Спочатку обчислюється $x_N = b_N/a_{NN}$, а потім використовується правило

$$x_k = \frac{b_k - \sum_{j=k+1}^{N} a_{kj} x_j}{a_{kk}} \quad \text{для } k = N-1,\ N-2,\ \ldots,\ 1.$$

```
function X=backsub(A,B)
%Вхід  - A - верхня трекутна матриця розмеру n x n
%      - B - матриця розміру n x 1
%Вихід - X - розв'язок системи лінійных рівнянь AX = B

%Знаходимо розмір матриці B і ініціалізуємо X
n=length(B);
X=zeros(n,1);
X(n)=B(n)/A(n,n);
for k=n-1:-1:1
    X(k)=(B(k)-A(k,k+1:n)*X(k+1:n))/A(k,k);
end
```

Метод простих ітерацій. (ітерації Якобі). Розв'язання лінійної системи *AX* = *B*, починаючи з вихідного значення X = P_0, і генерування послідовности {$P_к$}, яка збігається до розв'язку. Ефективною умовою для застосування методу є те, що *A* — строго діагонально домінуюча матриця.

```
function X=jacobi(A,B,P,delta,maxI)
%Вхід - A - невироджена матриця розміру NxN
% - B - матриця розміру Nx1
% - P - матриця розміру Nx1, початкове припущення
% - delta - допустиме відхилення для P
% - maxI - максимальне число ітерацій
%Вихід - X - матриця розміру Nx1: наближення Якобі до розв'язку
% AX - B
N = length(B);
for k=1:maxI
    for j=1:N
        X(j)=(B(j)-A(j,[1:j-1,j+1:N])*P([1:j-1,j+1:N]))/A(j,j);
    end
    err=abs(norm(X'-P));
    relerr=err/(norm(X)+eps);
    P=X';
    if(err<delta)|(relerr<delta)
        break
    end
end
X=X';
```

Метод Гаусса-Зейделя. Розв'язання лінійної системи *AX* = *B*, починаючи з висхідного значення *X* = P_0 методом генерування послідовности {P_k}, яка сходиться до розв'язку. Ефективною умовою для застосування методу є те, що *A* — строго діагонально домінуюча матриця.

```
function X=gseid(A,B,P,delta,maxI)
%Вхід - A - невироджена матриця розміру NxN
% - B - матриця розміру Nx1
% - P - матриця розміру Nx1, початкове припущення
% - delta - допустиме відхилення для P
% - maxI - максимльне число ітерацій
%Вихід - X матриця розмеру Nx1: наближення Гаусса-Зейделя до розв'язку AX =
B
N = length(B);
for k=1:maxI
    for j=1:N
        if j==1
            X(1)=(B(1)-A(1,2:N)*P(2:N))/A(1,1);
        elseif j==N
            X(N)=(B(N)-A(N,1:N-1)*(X(1:N-1))')/A(N,N);
        else
            %X містить k-e наближення і Pk-1
            X(j)=(B(j)-A(j,1:j-1)*X(1:j-1)-A(j,j+1:N)*P(j+1:N))/A(j,j);
        end
    end
    err=abs(norm(X'-P));
    relerr=err/(norm(X)+eps);
    P=X';
    if(err<delta)|(relerr<delta)
        break
    end
end
X=X';
```

Лабораторна робота № 4

Інтерполяція функцій

Мета роботи

Отримати практичні навики використання методів інтерполяції. Засвоїти властивості різних інтерполяційних поліномів, реалізованих власноруч.

Короткі теоретичні відомості до роботи

Задача інтерполяції

Задача інтерполяції – частинна, але досить поширена задача наближення функцій. Нехай в $n+1$ точках $x_0, x_1, ..., x_n$ відомі значення деякої функції $f(x_0), f(x_1), ..., f(x_n)$. Нехай значення x відмінне від x_j, $j = 0,1,...,n$, треба знайти значення $f(x)$.

Розглянемо сукупність функцій, досить простих і таких, що легко обчислюються, наприклад функцій, які лінійно залежать від параметрів $a_0, a_1, ..., a_n$:

$$\varphi(x, a_0, a_1, ..., a_n) = \sum_{i=0}^{n} a_i \varphi_i(x), \qquad (1)$$

де $\varphi_i(x)$, $i = 0,1,...,n$ – фіксовані функції. З усіх функцій сукупності (1) виберемо ту, для якої виконуються рівності

$$\varphi(x_j, a_0, a_1, ..., a_n) = f(x_j),\ j = 0,1,...,n. \qquad (2)$$

Рівності (2) складають систему лінійних алгебраїчних рівнянь відносно невизначених коефіцієнтів a_i:

$$\sum_{i=0}^{n} a_i \varphi_i(x_j) = f(x_j),\ j = 0,1,...,n. \qquad (3)$$

Покладемо

$$f(x) \approx \varphi(x, a_0, a_1, ..., a_n),$$

де $(a_0, a_1, ..., a_n)$ – розв'язок системи (3).

Такий спосіб наближення функцій зветься інтерполяцією, а точки x_j, $j = 0,1,...,n$ – вузлами інтерполяції.

Найбільш вивченим є випадок інтерполяції многочленами, коли

$$\varphi_i(x) = x^i,\ i = 0,1,...,n;$$

$$\varphi(x, a_0, a_1, ..., a_n) = P_n(x) = \sum_{i=0}^{n} a_i x^i, \qquad (4)$$

і система (3) має вигляд

$$\sum_{i=0}^{n} a_i x_j^i = f\left(x_j\right),\ j = 0,1,...,n. \qquad (5)$$

Якщо припустити, що $x_i \neq x_j$ при $i \neq j$ $\left(i = 0,1,...,n;\ j = 0,1,...,n\right)$, то система (4) має єдиний розв'язок, тому що її визначник відмінний від нуля (визначник Вандермонда). Звідси випливає, що серед усіх многочленів вигляду (4) існує єдиний многочлен $P_n(x)$ такий, що задовольняє умови (5).

Спосіб побудови інтерполяційного многочлена, за яким коефіцієнти a_i визначаються безпосереднім розв'язанням системи (5), зветься способом невизначених коефіцієнтів, а зображення інтерполяційного многочлена у формі (4) – степеневим зображенням.

Оптимальна схема обчислення форми (4) – схема Горнера:

$$P_n(x) = a_0 + x\left(a_1 + x\left(a_2 + ... + x\,a_n\right)\right), \qquad (6)$$

яка потребує n операцій множення і n операцій додавання.

Інтерполяційний многочлен Лагранжа

Так називають наступну форму запису інтерполяційного многочлена:

$$L_n(x) = \sum_{i=0}^{n} f\left(x_i\right) \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \qquad (7)$$

Інтерполяційний многочлен Лагранжа зображають також у вигляді

$$L_n(x) = \sum_{i=0}^{n} f\left(x_i\right) \frac{\omega_{n+1}(x)}{\omega'_{n+1}(x_i)(x - x_i)},$$

де

$$\omega_{n+1}(x) = \prod_{j=0}^{n}\left(x - x_j\right) = (x - x_0)(x - x_1)...(x - x_n).$$

Многочлени

$$l_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{\omega_{n+1}(x)}{\omega'_{n+1}(x_i)(x - x_i)},\ i = 0,1,...,n, \qquad (8)$$

називають множниками Лагранжа. Очевидно,

$$l_i\left(x_j\right) = \begin{cases} 0, & \text{коли } i \neq j, \\ 1, & \text{коли } i = j. \end{cases}$$

Обчислення можна організувати економічно, якщо записати (7) у вигляді (8)

$$L_n(x) = \omega_{n+1}(x) \sum_{i=0}^{n} \frac{Y_i}{x - x_i}, \qquad (9)$$

де

$$Y_i = \frac{f\left(x_i\right)}{\prod_{j=0,\, j\neq i}^{n} (x_i - x_j)},\ i = 0,1,...,n. \qquad (10)$$

Залишковий член інтерполяційної формули

Припускаючи вузли інтерполяції відмінними один від другого, а функцію $f(x)$ такою, що має неперервну похідну порядку $n+1$ на проміжку $[a,b]$, де розміщені вузли інтерполяції, можна записати залишковий член інтерполяційної формули

$$R_n(x) = f(x) - P_n(x)$$

на цьому проміжку у вигляді

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x), \qquad (11)$$

де

$$\xi \in [\alpha_1, \alpha_2],\ \alpha_1 = \min(x, x_0, x_1, ..., x_n),\ \alpha_2 = \max(x, x_0, x_1, ..., x_n).$$

Тоді

$$|R_n(x)| \le \frac{M_{n+1}}{(n+1)!}|\omega_{n+1}(x)|, \qquad (12)$$

де

$$M_{n+1} = \max_{x\in[\alpha_1,\alpha_2]} |f^{(n+1)}(x)|.$$

Приклад 1. Розглянемо фрагмент таблиці функції $y = x + \sin x$

x_i	1,4	1,5	1,7	1,8
y_i	2,38545	2,49749	2,69166	2,77385

Запишемо многочлен Лагранжа, використовуючи всю наявну інформацію, тобто покладаючи $n=3$, у вигляді

$$L_3(x) = 2,38548\cdot\frac{(x-1,5)\cdot(x-1,7)\cdot(x-1,8)}{(1,4-1,5)\cdot(1,4-1,7)\cdot(1,4-1,8)} +$$
$$+2,49749\cdot\frac{(x-1,4)\cdot(x-1,7)\cdot(x-1,8)}{(1,5-1,4)\cdot(1,5-1,7)\cdot(1,5-1,8)} +$$
$$+2,69166\cdot\frac{(x-1,4)\cdot(x-1,5)\cdot(x-1,8)}{(1,7-1,4)\cdot(1,7-1,5)\cdot(1,7-1,8)} +$$
$$+2,77385\cdot\frac{(x-1,4)\cdot(x-1,5)\cdot(x-1,7)}{(1,8-1,4)\cdot(1,8-1,5)\cdot(1,8-1,7)}.$$

Обчислимо значення $L_3(x)$ в точці $x=1,6$, оцінивши спочатку $R_n(x)$ відповідно до формули (12).

$$\omega_4(1,6) = 0,2\cdot 0,1\cdot(-0,1)\cdot(-0,2) = 0,0004,$$
$$y^{(4)}(x) = \sin x;\ |\sin x| \le 1.$$

Отже

$$|R_3(1,6)| \le \frac{1}{4!}\cdot 4\cdot 10^{-4} \le 0,2\cdot 10^{-4}.$$

Скористаємось формою запису інтерполяційного многочлена (9). Усі обчислення розташуємо в таблиці 1.

Таблиця 1

i	$x - x_i$	$x_i - x_j,\ i \neq j$			$\prod_{i \neq j}(x_i - x_j)$,	$f(x_i)$	$\frac{Y_i}{(x - x_i)}$
0	0,2	– 0,1	– 0,3	– 0,4	– 0,012	2,38545	– 993,938
1	0,1	0,1	– 0,2	– 0,3	0,006	2,49749	4162,48
2	– 0,1	0,3	0,2	– 0,1	– 0,006	2,69166	4486,10
3	– 0,2	0,4	0,3	0,1	0,012	2,77385	– 1155,77

$$L_3(1,6) = \omega_4(1,6) \cdot \sum_{i=0}^{3} \frac{Y_i}{(1,6 - x_i)},$$

$$L_3(1,6) = 2,59955.$$

Для порівняння наведемо значення функції $y = x + \sin x$ для $x = 1,6$ з п'ятьма точними десятковими знаками: $y(1,6) = 2,59957$.

Інтерполяційний многочлен Ньютона з поділеними різницями

Розглянемо форму запису інтерполяційного многочлена, яка використовує поняття поділеної різниці.

За означенням, значення функції $f(x_s)$, $s = 0,1,...,n$ називають поділеними різницями нульового порядку. Поділені різниці першого порядку визначаються співвідношеннями

$$f(x_s; x_{s+1}) = \frac{f(x_{s+1}) - f(x_s)}{x_{s+1} - x_s},\ s = 0,1,...,n-1.$$

Різниці *k*-го порядку $(k \leq n)$ визначаються через різниці (*k* – 1)-го порядку співвідношеннями

$$f(x_s; x_{s+1}; ...; x_{s+k}) = \frac{f(x_{s+1}; x_{s+2}; ...; x_{s+k}) - f(x_s; x_{s+1}; ...; x_{s+k-1})}{x_{s+k} - x_s}, \qquad (13)$$

$$s = 0,1,...,n-k.$$

Методом математичної індукції можна показати, що для поділених різниць *k*-го порядку справедлива формула

$$f(x_s; x_{s+1}; ...; x_{s+k}) = \sum_{i=0}^{s+k} \frac{f(x_i)}{\prod_{j=0,\ j \neq i}^{s+k} (x_i - x_j)}, \qquad (14)$$

з якої випливає, що поділена різниця є лінійна функція відносно функції f:

$$(\alpha_1 f_1 + \alpha_2 f_2)(x_s; x_{s+1}; ...; x_{s+k}) = \alpha_1 f_1(x_s; x_{s+1}; ...; x_{s+k}) + \alpha_2 f_2(x_s; x_{s+1}; ...; x_{s+k}),$$

де α_1, α_2 – деякі числа, і що поділена різниця є симетрична функція своїх аргументів (тобто не змінюється при довільній їх перестановці).

Для практичного застосування важливою є властивість поділеної різниці, яка показує її зв'язок з похідною

$$f\left(x;x_0;x_1;...;x_n\right)=\frac{f^{(n+1)}(\xi)}{(n+1)!}, \qquad (15)$$

де $x\in[a,b]$ і не співпадає з жодним вузлом інтерполяції x_i $\left(x_i\in[a,b]\right)$, $i=0,1,...,n$, $\xi\in\left[\alpha_1,\alpha_2\right]$, $\alpha_1=\min\left(x,x_0,x_1,...,x_n\right)$, $\alpha_2=\max\left(x,x_0,x_1,...,x_n\right)$.

Зауваження 1. З (15) випливає, що поділені різниці *k*-го порядку від многочлена *k*-го степеня постійні, а різниці (*k* + 1)-го і вищого порядку дорівнюють нулю. Отже, спостерігаючи за поведінкою поділених різниць великої таблиці функцій, можна зробити висновок про те, який степінь інтерполяційного многочлена доцільно використовувати для наближення цієї функції.

Інтерполяційним многочленом Ньютона з поділеними різницями називають таку форму запису інтерполяційного многочлена

$$\begin{aligned}H_n(x)=\ & f\left(x_0\right)+\left(x-x_0\right)\cdot f\left(x_0;x_1\right)+\left(x-x_0\right)\cdot\left(x-x_1\right)\cdot f\left(x_0;x_1;x_2\right)+\\ &+...+\left(x-x_0\right)\cdot\left(x-x_1\right)\cdot...\cdot\left(x-x_{n-1}\right)\cdot f\left(x_0;x_1;...;x_n\right).\end{aligned} \qquad (16)$$

Будемо називати її у подальшому формою Ньютона.

З (16) бачимо, що додавання одного або декількох вузлів з відповідним підвищенням степеня інтерполяційного многочлена потребує додавання до правої частини цієї формули відповідного числа доданків. Зазначена особливість надає формулі Ньютона переваги у порівнянні з формулою Лагранжа.

Зупинимось на питанні обчислення форми Ньютона. Коефіцієнти форми Ньютона (16) розміщені на “головній діагоналі” таблиці 2 поділених різниць.

Таблиця 2

x_i	0-го пор.	1-го пор.	2-го пор.	⋮	($n-1$)-го пор.	*n*-го пор.
x_0	$f\left(x_0\right)$					
x_1	$f\left(x_1\right)$	$f\left(x_0;x_1\right)$				
x_2	$f\left(x_2\right)$	$f\left(x_1;x_2\right)$	$f\left(x_0;x_1;x_2\right)$			
x_3	$f\left(x_3\right)$	$f\left(x_2;x_3\right)$	$f\left(x_1;x_2;x_3\right)$			
⋮	⋮	⋮	⋮			
x_{n-1}	$f\left(x_{n-1}\right)$	$f\left(x_{n-2};x_{n-1}\right)$	$f\left(x_{n-3};x_{n-2};x_{n-1}\right)$		$f\left(x_0;...;x_{n-1}\right)$	
x_n	$f\left(x_n\right)$	$f\left(x_{n-1};x_n\right)$	$f\left(x_{n-2};x_{n-1};x_n\right)$		$f\left(x_1;...;x_n\right)$	$f\left(x_0;x_1;...;x_n\right)$

Для обчислення форми Ньютона (16) скористаємося позначенням $f\left(x_0;x_1;...;x_n\right)=A_i$ для усіх i та зобразимо її у вигляді

$$H_n(x)=A_0+\left(x-x_0\right)\cdot\left(A_1+\left(x-x_1\right)\cdot\left(A_2+...+\left(x-x_{n-1}\right)\cdot A_n\right)...\right), \qquad (17)$$

який є узагальненням схеми Горнера (16).

Наведемо ще одну форму запису інтерполяційного многочлена Ньютона з поділеними різницями

$$H_n(x) = f(x_n) + (x - x_n) \cdot f(x_{n-1}; x_n) + (x - x_{n-1}) \cdot (x - x_n) \cdot f(x_{n-2}; x_{n-1}; x_n) + \\ + \ldots + (x - x_1) \cdot (x - x_2) \cdot \ldots \cdot (x - x_n) \cdot f(x_0; x_1; \ldots; x_n). \quad (18)$$

Припустимо тепер, що степінь інтерполяційного многочлена зафіксована, дорівнює n і задана таблиця $\{x_i, f(x_i)\}, i = 0,1,\ldots,m$, де $m = n$. Якщо вузли інтерполяції вибрати поблизу точки x, для якої наближено обчислюються значення функції $f(x)$, то проміжна точка ξ в виразі для похибки інтерполяції (11) також буде поблизу точки x, при цьому величина $f^{(n+1)}(\xi)$ змінюється не дуже суттєво. На величину похибки суттєво впливає співмножник $\omega_{n+1}(x)$.

Зауваження 2. Розглянемо $|\omega_{n+1}(x)| = \prod_{j=0}^{n} |x - x_j|$, тобто модуль добутку відстаней від точки x до вузлів інтерполяції. Ця величина буде мінімальною, якщо використати $(n+1)$ вузлів, найближчих до x. Тому формулу Ньютона (16) використовують в околі точки x_0, а формулу (18) в околі точки x_n.

Звернемося до попереднього прикладу і розглянемо таблицю поділених різниць для функції $y = x + \sin x$

Таблиця 3

i	x_i	$f(x_i)$	$f(x_i; x_{i+1})$	$f(x_i; x_{i+1}; x_{i+2})$	$f(x_i; \ldots; x_{i+3})$
0	1,4	2,38545			
1	1,5	2,49749	1,1204		
2	1,7	2,69166	0,97085	– 0,4985	
3	1,8	2,77385	0,8219	– 0,4965	0,005

Обчислимо $H_3(1,6)$ за формулою Ньютона (16)

$$H_3(1,6) = 2,38545 + 0,2 \cdot 1,1204 + 0,2 \cdot 0,1 \cdot (-0,4985) + \\ + 0,2 \cdot 0,1 \cdot (-0,1) \cdot 0,005 = 2,59955.$$

Обчислимо тепер $H_3(x)$ для $x = 1,45$

$$H_3(1,45) = 2,38545 + 0,05 \cdot 1,1204 + 0,05 \cdot (-0,05) \cdot (-0,4985) + \\ + 0,05 \cdot (-0,05) \cdot (-0,25) \cdot 0,005 = 2,44272.$$

Оцінимо $R_3(x)$ для $x = 1,45$. Маємо

$$|R_3(1,45)| \le \frac{1}{4!} |\omega_4(1,45)| < 0,1 \cdot 10^{-4}.$$

Обчислимо $H_3(x)$ для $x = 1,75$, використовуючи формулу (18)

$$H_3(1,75) = 2,77385 + (-0,05) \cdot 0,8219 + (-0,05) \cdot 0,05 \cdot (-0,4985) + \\ + (-0,05) \cdot 0,05 \cdot 0,25 \cdot 0,005 = 2,73400.$$

Оцінюючи похибку наближення $f(1,75) \approx H_3(1,75)$, отримуємо

$$|R_3(1,75)| \le \frac{1}{4!} |\omega_4(1,75)| < 0,1 \cdot 10^{-4}.$$

Інтерполяційний многочлен Ньютона з скінченними різницями

Частинним, але особливо поширеним, випадком задання таблиці $\{x_i, f(x_i)\}, i = 0,1,...,n$, є випадок, коли вузли розміщені на рівних відстанях: $x_i = x_0 + ih$, де h – крок таблиці. При цьому замість поділених різниць є сенс розглядати скінченні різниці.

Розглянемо скінченні різниці першого порядку вперед

$$\Delta f_i = f_{i+1} - f_i, i = 0,1,2,...,n-1.$$

Різниці вищого порядку визначаються рекурентними співвідношеннями

$$\Delta^m f_i = \Delta\left(\Delta^{m-1} f_i\right) = \Delta^{m-1} f_{i+1} - \Delta^{m-1} f_i, m \le n.$$

Для $x_i = x_0 + ih$ справедлива рівність

$$f(x_i, x_{i+1},..., x_{i+k}) = \frac{\Delta^k f_i}{k! h^k},$$

з якої випливає, що скінченні різниці *k*-го порядку від многочлена степеня *k* постійні, а різниці вищого порядку дорівнюють нулю і зберігає сенс зауваження 1.

Інтерполюючи по таблицям з постійним кроком, використовують многочлени з скінченними різницями. Переходячи від поділених різниць до скінченних і виконавши заміну змінної $x = x_0 + th$, з (16) одержимо

$$H_n(x_0 + ih) = f_0 + \Delta f_0\, t + \frac{\Delta^2 f_0}{2!} t(t-1) + ... + \frac{\Delta^n f_0}{n!} t(t-1)(t-2)...(t-n+1). \quad (19)$$

Похибка інтерполяції при цьому має вигляд

$$f(x) - H_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} t(t-1)(t-2)...(t-n) \cdot h^{n+1}. \quad (20)$$

Многочлен (19) називають інтерполяційним многочленом Ньютона інтерполяції вперед.

Виходячи з форми Ньютона (18), де виконані аналогічні перетворення і заміна змінної $x = x_n + th$, одержимо інтерполяційний многочлен Ньютона інтерполяції назад

$$H_n(x_n + ih) = f_n + \Delta f_{n-1}\, t + \frac{\Delta^2 f_{n-2}}{2!} t(t+1) + ... + \frac{\Delta^n f_0}{n!} t(t+1)(t+2)...(t+n-1) \quad (21)$$

і відповідно

$$f(x) - H_n(x_n + ih) = \frac{f^{(n+1)}(\xi)}{(n+1)!} t(t+1)(t+2)...(t+n) \cdot h^{n+1}. \quad (22)$$

Формули (19) і (21) зручно використовувати при інтерполяції відповідно на початку і в кінці таблиці.

Приклад 2. Розглянемо таблицю функції $y = x^2 e^{-x}$ на проміжку $[2, 3]$ за умов $x_0 = 2$, $x_i = x_0 + ih$, $h = 0,2$, $i = \overline{0, 5}$ і поставимо задачу: обчислити за даною таблицею з використанням інтерполяційного многочлена третього степеня значення функції $y = x^2 e^{-x}$ в точках $x = 2,1;\ 2,5;\ 2,9$ і оцінити відповідні похибки інтерполяції.

Складемо таблицю скінченних різниць. Оскільки степінь інтерполяційного многочлена дорівнює трьом, то таблицю складемо до різниць третього порядку включно.

Таблиця 4

i	x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$
0	2,0	0,541341			
1	2,2	0,536287	– 0,005054		
2	2,4	0,522535	– 0,013752	– 0,008698	
3	2,6	0,502089	– 0,020446	– 0,006694	0,002004
4	2,8	0,476751	– 0,025338	– 0,004892	0,001802
5	3,0	0,448084	– 0,028667	– 0,003329	0,001563

Для розв'язання задачі скористаємося формулами (19) – (20) і (21) – (22).

$$1.\ x = 2,1;\ x_0 = 2;\ t = \frac{x - x_0}{h} = 0,5.$$

$$H_3(2,1) = 0,541341 + (-0,005054)\cdot 0,5 + \frac{1}{2!}(-0,008698)\cdot 0,5\cdot(-0,5) +$$

$$+\frac{1}{3!}0,002004\cdot 0,5\cdot(-0,5)\cdot(-1,5) = 0,540056.$$

$$y^{(\mathrm{IV})}(x) = e^{-x}\left(x^2 - 8x + 12\right),$$

$$\max_{x\in[2,3]}\left|y^{(\mathrm{IV})}(x)\right| < 0,2,$$

$$\left|R_3(2,1)\right| = \left|y(2,1) - H_3(2,1)\right| \le$$

$$\le \frac{\max\limits_{x\in[2,3]}\left|y^{(\mathrm{IV})}(x)\right|}{4!}\cdot(0,2)^4\cdot\left|0,5\cdot(-0,5)\cdot(-1,5)\cdot(-2,5)\right| < 0,2\cdot 10^{-4}.$$

$$2.\ x = 2,9;\ x_0 = 3;\ t = \frac{x - x_n}{h} = -0,5.$$

$$H_3(2,9) = 0,448084 + (-0,028667)\cdot(-0,5) + \frac{1}{2!}(-0,003329)\cdot(-0,5)\cdot 0,5 +$$

$$+\frac{1}{3!}0,001563\cdot(-0,5)\cdot 0,5\cdot 1,5 = 0,462736.$$

$$\left|R_3(2,9)\right| \le \frac{0,2}{4!}(0,2)^4\cdot\left|(-0,5)\cdot 0,5\cdot 1,5\cdot 2,5\right| < 0,2\cdot 10^{-4}.$$

3. $x = 2,5$. Враховуючи зауваження 2, слід вибрати вузли інтерполяції $x_0 = 2,2$; $x_1 = 2,4$; $x_2 = 2,6$; $x_3 = 2,8$. При цьому $t = \frac{x - x_0}{h} = 1,5$. Отже, якщо використати знову формулу (19), будемо мати

$$H_3(2,5) = 0,536287 + (-0,013752)\cdot 1,5 + \frac{1}{2!}(-0,006694)\cdot 1,5\cdot 0,5 +$$

$$+\frac{1}{3!}0,001802\cdot 1,5\cdot 0,5\cdot(-0,5) = 0,513036.$$

$$\left|R_3(2,9)\right| \le \frac{0,2}{4!}(0,2)^4\cdot\left|1,5\cdot 0,5\cdot(-0,5)\cdot(-1,5)\right| < 0,8\cdot 10^{-5}.$$

Наближення сплайнами. Основні відомості

Відомо, що збільшення кількості вузлів інтерполяції не завжди покращує результат інтерполяції, тому що в цьому випадку виникає осциляція многочлена між вузлами. Цього можна уникнути зменшенням степеня інтерполяційного многочлена (кусково-поліноміальна інтерполяція). При цьому отримуємо функцію , яка є неперервною, але в вузлах склеювання її похідна розривна. Така інтерполяція є неприйнятною, наприклад, в тому випадку коли потрібна деяка гладкість апроксимуючої функції.

Для отримання досить гладкої апроксимації використовують так звані сплайн-функції, або сплайни.

Розглянемо далі задачу побудови інтерполяційного кубічного сплайну.

1. Через $C^{(k)}[a,b]$ позначимо множину k раз неперервно диференційованих на $[a,b]$ функцій.

Розглянемо сітку

$$\Delta : a = x_0 < x_1 < \ldots < x_N = b,\ h_i = x_{i+1} - x_i,\ i = \overline{0, N-1}.$$

Визначення. Функцію $S_{n,v}(x)$ називаємо інтерполяційним сплайном степеня n дефекту v (v – ціле число) якщо:

а) на кожному відрізку $\left[x_i, x_{i+1}\right]$ функція $S_{n,v}(x)$ є многочлен степеня n;

б) $S_{n,v}(x) \in C^{(n-v)}[a,b]$;

в) $S_{n,v}\left(x_i\right) = y_i,\ y_i = f\left(x_i\right),\ i = \overline{1,N}$.

Далі будемо вивчати сплайни $g(x) = S_{3,1}(x)$.

Згідно з означенням $g(x)$ на кожному відрізку $\left[x_i, x_{i+1}\right]$, $i = \overline{0, N-1}$ визначається чотирма коефіцієнтами. Тобто необхідно визначити $4N$ коефіцієнтів.

Умова $g(x) \in C^{(2)}[a,b]$ дає $3\cdot(N-1)$ рівностей. Крім того, умови $g\left(x_i\right) = y_i, i = \overline{1,N}$ дають ще $N+1$ співвідношення на коефіцієнти (всього $4N-2$ співвідношень). Ще дві умови отримаємо, якщо задано обмеження на значення сплайна, або його похідних на кінцях відрізку $[a,b]$.

Найчастіше використовують такі умови

$$g'(a) = f'(a),\ g'(b) = f'(b); \qquad \text{(A)}$$

$$g''(a) = f''(a),\ g''(b) = f''(b);\qquad (B)$$

$$g^{(r)}(a) = g^{(r)}(b),\ r = 1,2.\qquad (C)$$

Умови (С) використовуються при побудові сплайна для періодичної функції $f(x)$.

Нехай для $x_i \le x \le x_{i+1}$ $g''(x) = a_i\left(x - x_i\right) + s_i$. Тоді

$$g'(x) = a_i\frac{\left(x - x_i\right)^2}{2} + s_i\left(x - x_i\right) + p_i,\ p_i = g'\left(x_i\right).\qquad (23)$$

Із (23) при $x = x_{i+1}$ маємо

$$p_{i+1} = a_i\frac{h_i^2}{2} + s_i h_i + p_i,$$

$$s_i = \frac{\Delta\, p_i}{h_i} - \frac{1}{2}a_i h_i,\ \Delta\, p_i = p_{i+1} - p_i.\qquad (24)$$

Інтегруючи (23) в межах від x_i до x, отримаємо

$$g(x) = \frac{a_i\left(x - x_i\right)^3}{6} + s_i\frac{\left(x - x_i\right)^2}{2} + p_i\left(x - x_i\right) + y_i.\qquad (25)$$

Із (25) при $x = x_{i+1}$ та беручи до уваги (2), отримаємо, що

$$a_i = \frac{6}{h_i^2}\left(p_i + p_{i+1} - 2\frac{\Delta\, y_i}{h_i}\right),\ \Delta\, y_i = y_{i+1} - y_i.\qquad (26)$$

Умова $g(x) \in C^{(2)}[a,b]$ має місце коли

$$g''\left(x_i - 0\right) = g''\left(x_i + 0\right) = g''\left(x_i\right),\ i = \overline{1, N-1}.$$

Приймаючи до уваги, що

$$g''\left(x_i - 0\right) = a_{i-1}h_{i-1} + s_{i-1},\ g''\left(x_i + 0\right) = g''\left(x_i\right) = s_i,$$

отримаємо

$$h_i p_{i-1} + 2\left(h_{i-1} + h_i\right)p_i + h_{i-1}p_{i+1} = 3\left(\frac{h_i}{h_{i-1}}\Delta\, y_{i-1} + \frac{h_{i-1}}{h_i}\Delta\, y_i\right),\ i = \overline{1, N-1}.\qquad (27)$$

Система (27) має $N-1$ рівняння і $N+1$ невідомих. Ще два рівняння отримаємо, якщо скористаємося умовами (А) або (В).

Умова (А) дає наступні два рівняння:

$$p_0 = g'(a) = f'(a),\ p_N = g'(b) = f'(b).\qquad (28)$$

Якщо виконуються умови (В), то

$$g''(x_0) = s_0 = \frac{\Delta\, p_0}{h_0} - \frac{1}{2}a_0 h_0 = \frac{1}{h_0}\left(-4p_0 - 2p_1\right) + \frac{6\Delta\, y_0}{h_0^2} = f''(x_0),$$

$$p_0 + \frac{1}{2}p_1 = \frac{3}{2h_0}\Delta\, y_0 - \frac{h_0}{4}f''(x_0).\qquad (29)$$

Аналогічно отримуємо друге рівняння

$$\frac{1}{2}p_{N-1} + p_N = \frac{h_{N-1}}{4}f''(x_N) + \frac{3\Delta\, y_{N-1}}{2h_{N-1}}.\qquad (30)$$

Таким чином, в цьому випадку система (27) доповнюється рівняннями (29) і (30).

Обчислювальна схема

1) Розв’язати систему

$$\begin{cases} p_0 + \gamma_0 p_1 = F_0, \\ h_i p_{i-1} + 2\left(h_{i-1} + h_i\right) p_i + h_{i-1} p_{i+1} = 3\left(\dfrac{h_i}{h_{i-1}} \Delta\, y_{i-1} + \dfrac{h_{i-1}}{h_i} \Delta\, y_i \right), \; i = \overline{1, N-1}, \\ \gamma_N p_{N-1} + p_N = F_N, \end{cases} \quad (31)$$

де

$$F_0 = \begin{cases} f'(x_0) & \text{у випадку (А)}, \\ \dfrac{3\Delta\, y_0}{2h_0} - \dfrac{h_0}{4} f''(x_0) & \text{у випадку (В)}, \end{cases}$$

$$F_N = \begin{cases} f'(x_N) & \text{у випадку (А)}, \\ \dfrac{3\Delta\, y_{N-1}}{2h_{N-1}} + \dfrac{h_{N-1}}{4} f''(x_N) & \text{у випадку (В)}, \end{cases}$$

$$\gamma_0 = \gamma_N = \begin{cases} 0 & \text{у випадку (А)}, \\ \dfrac{1}{2} & \text{у випадку (В)}. \end{cases}$$

2) $a_i = \dfrac{6}{h_i^2}\left(p_i + p_{i+1} - 2\dfrac{\Delta\, y_i}{h_i} \right), \; s_i = \dfrac{\Delta\, p_i}{h_i} - \dfrac{1}{2} a_i h_i, \; i = \overline{0, N-1}.$

3) $g(x) = \dfrac{a_i \left(x - x_i\right)^3}{6} + s_i \dfrac{\left(x - x_i\right)^2}{2} + p_i \left(x - x_i\right) + y_i, \; x_i \le x \le x_{i+1}, \; i = \overline{0, N-1}.$

Матриця коефіцієнтів системи (31) являється тридіагональною. Наведемо обчислювальну схему метода прогонки розв’язання системи

$$\begin{cases} B_0 z_0 + C_0 z_1 = F_0, \\ A_i z_{i-1} + B_i z_i + C_i z_{i+1} = F_i, \;\; i = \overline{1, N-1} \\ A_N z_{N-1} + B_N z_N = F_N. \end{cases} \quad (32)$$

Обчислювальна схема.

1) $$\alpha_0 = -\frac{C_0}{B_0}, \; \beta_0 = \frac{F_0}{B_0};$$

2) $$\alpha_i = -\frac{C_i}{\alpha_{i-1} A_i + B_i}, \; \beta_i = \frac{F_i - \beta_{i-1} A_i}{\alpha_{i-1} A_i + B_i}, \; i = \overline{1, N-1}; \quad (33)$$

3) $$z_N = \frac{F_N - \beta_{N-1} A_N}{B_N + \alpha_{N-1} A_N};$$

4) $$z_i = \alpha_i z_{i+1} + \beta_i, \; i = N-1, \, N-2, \ldots, 1, \, 0.$$

Називаємо прогонку коректною, якщо знаменники коефіцієнтів α_i, β_i не перетворюються в нуль і стійкою, якщо всі $|\alpha_i| < 1,\ i = \overline{0, N-1}$. Наведемо достатні умови коректності і стійкості прогонки:

$$A_i \neq 0, C_i \neq 0, i = \overline{1, N-1}; |B_i| > |A_i| + |C_i|, i = \overline{1, N}, \left|\frac{C_0}{B_0}\right| \leq 1, \left|\frac{A_N}{B_N}\right| \leq 1.$$

Приклад. Знайдемо методом прогонки розв'язок наступної системи

$$\begin{cases} 2z_0 - z_1 = 3, \\ -z_0 + 2z_1 + z_2 = -1, \\ 3z_1 - z_2 + 2z_3 = -1, \\ z_2 + z_2 + z_4 = 5, \\ z_3 + 2z_4 = 4. \end{cases}$$

Скористаємось обчислювальною схемою (11), а всі результати обчислень занесемо в таблицю 5

Таблиця 5

i	A_i	B_i	C_i	F_i	α_i	β_i	z_i
0		2	– 1	3	$\frac{1}{2}$	$\frac{3}{2}$	1
1	– 1	2	1	– 1	$-\frac{2}{3}$	$\frac{1}{3}$	– 1
2	3	– 1	2	– 1	$\frac{2}{3}$	$\frac{2}{3}$	2
3	1	1	1	5	$-\frac{3}{5}$	$\frac{13}{5}$	2
4	1	2		4			1

1) $$\alpha_0 = -\frac{C_0}{B_0} = -\frac{-1}{2};\ \beta_0 = \frac{F_0}{B_0} = \frac{3}{2};$$

2) $$\alpha_1 = -\frac{C_1}{\alpha_0 A_1 + B_1} = -\frac{1}{\frac{1}{2}\cdot(-1) + 2} = -\frac{2}{3};\ \beta_1 = \frac{F_1 - \beta_0 A_1}{\alpha_0 A_1 + B_1} = \frac{-1 - \frac{3}{2}\cdot(-1)}{\frac{1}{2}\cdot(-1) + 2} = \frac{1}{3}.$$

Аналогічно обчислюємо $\alpha_2, \beta_2, \alpha_3, \beta_3$.

3) $$z_4 = \frac{F_4 - \beta_3 A_4}{\alpha_3 A_4 + B_4} = \frac{4 - \frac{13}{5} \cdot 1}{-\frac{3}{5} \cdot 1 + 2} = 1;$$

4) $z_3 = \alpha_3 z_4 + \beta_3 = -\frac{3}{5} \cdot 1 + \frac{13}{5} = 2$, $z_2 = \alpha_2 z_3 + \beta_2 = \frac{2}{3} \cdot 2 + \frac{2}{3} = 2$; $z_1 = -1$; $z_0 = 1$.

Порядок виконання роботи

1. Написати власну функцію MATLAB для обчислення значень функції $f(x)$, заданої аналітично в табл.6 відповідно до варіанту.
2. Вивести графік заданої функції. Побудувати таблицю значень заданої функції $y_i = f(x_i)$ в $N + 1$ ($N = 5, 10, 50$) рівновіддалених точках на відрізку $[a, b]$ відповідно до варіанту (табл. 6).
3. Написати функцію MATLAB, яка будує інтерполяційний поліном $P_N(x)$ оснований на $N + 1$ точках (x_i, y_i) $i = 0, 1, \ldots, N$ методом Лагранжа. Функція повинна приймати на вхід вектор вузлів інтерполяції x_i та вектор відповідних значень функції y_i. На виході функція повинна повертати вектор коефіцієнтів інтерполяційного поліному і матрицю коефіцієнтів Лагранжа ((i+1)-ий рядок матриці містить коефіцієнти відповідного i-го поліному – коефіцієнту Лагранжа, $i = 0, 1, \ldots, N$). Отримати за допомогою написаної функції інтерполяційні поліноми $P_N(x)$ основані на $N + 1$ ($N = 3, 9$) вузлах (розподілених зі сталим кроком та довільно) на заданому відрізку $[a, b]$. Побудувати графіки отриманих інтерполяційних поліномів суміщені з графіками вихідної функції.
4. Написати функцію MATLAB, яка будує інтерполяційний поліном $P_N(x)$ оснований на $N + 1$ точках (x_i, y_i) $i = 0, 1, \ldots, N$ методом Ньютона для нерівних проміжків. Функція повинна приймати на вхід вектор вузлів інтерполяції x_i та вектор відповідних значень функції y_i. На виході функція повинна повертати вектор коефіцієнтів інтерполяційного поліному і матрицю розділених різниць, отриманих при побудові інтерполяційного поліному Ньютона. Отримати за допомогою написаної функції інтерполяційні поліноми $P_N(x)$ з тими ж умовами, що і в п. 3. Вивести відповідні графіки.
5. Порівняти коефіцієнти поліномів отримані в п. 3 і 4 на однакових вузлах. В одному графічному вікні вивести графік заданої функції $f(x)$, інтерполяційного поліному $P_N(x)$, отриманого в п. 3 або 4 та позначити точки (x_i, y_i), на основі яких проводилась інтерполяція. Окремо вивести графік похибки інтерполювання $e(x) = |f(x) - P_N(x)|$ на відрізку [a; b]. Оцінити похибку інтерполяції для заданої функції теоретично (за формулою (12)). Порівняти теоретичний результат з графіком $e(x)$.
6. Написати функцію для побудувати інтерполяційного сплайну третього порядку $S(x)$. Функція повинна приймати на вхід вектор вузлів інтерполяції x_i та вектор відповідних значень функції y_i, граничні умови в крайніх точках відрізку інтегрування. На виході функція повинна повертати матрицю кубічних поліномів сплайна (i-ий рядок мстриці – це коефіцієнти в порядку зменшення i-го кубічного полінома). Отримати за допомогою написаної функції сплайни основані на $N + 1$ ($N = 3, 9, 24$) вузлах (розподілених зі сталим кроком) на заданому відрізку $[a, b]$.
7. Побудувати графіки отриманих сплайнів суміщені з графіками вихідної функції та позначити точки (x_i, y_i), на основі яких проводилась інтерполяція. Окремо побудувати графік похибки інтерполяції $e(x) = |f(x) - S(x)|$.
8. Виконати інтерполяцію заданої функції на відрізку $[a, b]$ використавши стандартну функцію MATLAB `interp1(x,Y,xi,method)` задавши різні методи інтерполяції

('nearest', 'linear', 'spline'). Вивести відповідні графіки. Порівняти отримані результати з результатами п. 3, 4, 6 при тих же умовах.

9. Зробити висновки по роботі порівнявши методи з п. 3, 4 та 6, 8.

Варіанти до роботи

Таблиця 6

Вар	$f(x)$	$[a, b]$
1.	$\sin(x^3)$	[0, 2]
2.	$\cos(x^3)$	[0, 5]
3.	$1/(0.5 + x^2)$	[0, 2]
4.	$e^{-(x+\sin(x))}$	[2, 5]
5.	$1/(1 + e^{-x})$	[-2, 3]
6.	$\sin(x + e^{\sin(x)})$	[0, 3]
7.	$e^{-(x + 1/x)}$	[1, 3]
8.	$x \cdot \cos(x + \ln(1+x))$	[1, 5]
9.	$10 \cdot \ln(2x)/(1+x)$	[1, 5]
10.	$\sin^2(x) \cdot e^{-(x/2)^2}$	[0, 3]
11.	$\cos(x + \cos^3(x))$	[0, 2]
12.	$\cos(x + e^{\cos(x)})$	[3, 6]
13.	$\text{atan}(x)$	[-2, 3]
14.	$\ln(x^2 + 1)$	[0, 2]
15.	$\tan(x)$	[-1, 1]
16.	$x / (x^2 + 1)$	[0, 3]
17.	$x^{1/2}$	[1, 9]
18.	$e^{-(\cos(x))}$	[3, 6]
19.	$\cos(x) + e^{-x}$	[0, 1]
20.	$\sin(x)$	[0, 2]
21.	$x^{-\sin(x)}$	[-2, 1]
22.	$x \cdot \ln(1+x))$	[0, 3]
23.	$(1+x) / \ln(5x)$	[0.5, 2.5]
24.	$\sin^2((x/2)^2)$	[0, 3.5]
25.	$\cos(x) + \cos^3(x)$	[1, 5]
26.	$e^{\cos(x)}$	[-2, 1]
27.	$x^{1/2} + \sin(x)$	[1, 15]
28.	$(x+5)^2/e^x$	[-5, 0]
29.	$x \cdot \cos(x)$	[-1, 3]
30.	$x \cdot \sin(x)$	[1, 5]

Функції, що задаються користувачем

Є кілька способів визначення власних функцій користувача. Перший спосіб використовує команду `inline`. Аргументом цієї команди є вираз, що задає функцію однієї або декількох змінних. Вираз функції задається в одинарних лапках (апострофах). В наступному прикладі задається функція **f(x) = x^2**.

```
>> f=inline('x^2')
f =
   Inline function:
   f(x) = x^2
```

Приклад визначення функції двох змінних – суми квадратів sin(x) і cos(y):

```
>> sc2=inline('sin(x).^2+cos(y).^2', 'x', 'y')
sc2 =
   Inline function:
   sc2(x,y) = sin(x).^2+cos(y).^2
```

В останньому прикладі крім функції явно задаються і її аргументи 'x' і 'y'. Це зручно, коли MATLAB не може вірно визначити назви або прядок аргументів функції, що задається користувачем. Необхідно також відмітити, що більшість функцій MATLAB можуть оперувати як із скалярами, так і з векторами. Щоб бути впевненим, що задана вами функція може оперувати з векторами, вставляйте крапки перед математичними операторами *, / і ^. З наведених вище прикладів, перша функція **f** буде працювати лише зі скалярними аргументами, а друга **sc2** – як із скалярами, так і з векторами.

Викликати функцію можна задавши її ім’я та значення аргументів: sc2(x,y), де замість x і y підставляються значення аргументів. Наприклад:

```
>> sc2(1,2)
ans = 0.8813
>> sc2(2,1)
y = 1.1187
ans = 1.1187
>> x=1:10
x =
     1     2     3     4     5     6     7     8     9    10
>> y=-10:-1
y =
   -10    -9    -8    -7    -6    -5    -4    -3    -2    -1
>> sc2(x,y)
ans = 1.4121    1.6570    0.0411    1.1411    1.8415    0.1585    0.8589
1.9589    0.3430    0.5879
```

Функції можна також задавати в окремих файлах, які називаються M-файлами. Для створення такого файлу можна, наприклад, командою New з меню File викликати редактор m-файлів. В редакторі набрати:

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

Записати новий файл з ім’ям ‘sc2.m’ на диск. Ім’я файлу (за виключенням розширення .m) повинне співпадати з назвою функції (**sc2** в даному випадку). Перший рядок файлу розпочинається із слова function, яке ідентифікує файл як M-файл-функцію. У модулі Editor (редактор m-файлів) це зарезервоване слово виділяється синім кольором. Перший рядок M-файлу задає ім'я функції і описує як вхідні аргументи(чи параметри), так і вихідні значення. В даному прикладі функція називається **sc2**. Коли ви створюєте цей новий M-файл-функцію у безіменному вікні редактора і вибираєте команду Save (Зберегти), модуль Editor (редактор) сам привласнює файлу ім'я sc2.m. Функція в нашому прикладі має для введення два елементи, які усередині M-файлу позначені і використовуються як змінні **x** та **y**. В якості результату повертається один елемент значення **f**, що з'являється у кінці виконання функції.

Використовуючи М-файли можна задавати більш складні функції, ніж `inline`.

Поліноми і деякі операції над ними

MATLAB має декілька стандартних функцій для роботи з поліномами, які можуть бути корисними при виконанні інтерполяції функцій.

Обчислення полінома. Функція **`polyval`**

Синтаксис:

```
Y=polyval(p,X)
```

Функція обчислює значення полінома **p** в точках, заданих масивом **X**. Поліном $p(x) = p_1x^n + p_2x^{n-1} + \ldots + p_nx + p_{n+1}$ задається вектором коефіцієнтів при степенях x починаючи з старшого степеня **p**=[p_1 p_2 … p_n p_{n+1}]. Це стандартна форма для визначення поліному степеня n. Вона використовуватиметься і в наступних функціях.

Приклад:

Обчислимо значення полінома $p(x) = 3x^2 + 2x + 1$ в точці $x = 5$.

```
>> p=[3 2 1]
p =  3   2   1
>> y=polyval(p,5)
y =  86
```

Добуток поліномів. Функція **`conv`**

Синтаксис:

```
c=conv(a,b)
```

Функція обчислює добуток двох поліномів **a** і **b**.

Приклад:

Обчислимо добуток поліномів $x^3 + 2x^2 + 3x + 4$ і $10x^2 + 20x + 30$. Для цього сформуємо вектори a=[1 2 3 4] і b=[10 20 30] і обчислимо

```
>> c=conv(a,b)
c =   10    40   100   160   170   120
```

Обчислення коренів полінома. Функція **`roots`**

Синтаксис:

```
r=roots(p)
```

Функція обчислює вектор коренів **r** полінома **p**.

Приклад:

Обчислимо корені полінома $x^3 + 3x^2 + 5x + 7$.

```
>> p=[1 3 5 7]
p =  1     3     5     7
>> r=roots(p)
r =
  -2.1795
  -0.4102 + 1.7445i
  -0.4102 - 1.7445i
```

Обчислення полінома з заданими коренями. Функція **`poly`**

Синтаксис:

```
p=poly(r)
```

Функція обчислює вектор коефіцієнтів полінома **p** за заданим вектором коренів полінома **r**.

Приклад:

Обчислимо поліном, який має два корені $r_1 = 1$, $r_2 = -1$.

```
>> r=[1 -1]
r =   1    -1
>> poly(r)
ans =   1     0    -1
```

Деякі функції роботи з масивами

Створення масиву заповненого нулями. Функція **zeros**

Синтаксис:

```
B=zeros(n)
B=zeros(n, m)
```

Функція zeros(n) повертає матрицю нулів розміром n-на-n. Функція zeros(n, m) повертає матрицю нулів розміром n-на-m.

Приклад:

```
>> A=zeros(1,3)
A =  0     0     0
>> B=zeros(3)
B =
     0     0     0
     0     0     0
     0     0     0
```

Створення масиву заповненого одиницями. Функція **ones**

Синтаксис:

```
B=ones(n)
B=ones(n, m)
```

Функція ones(n) повертає матрицю одиниць розміром n-на-n. Функція ones(n, m) повертає матрицю одиниць розміром n-на-m.

Приклад:

```
>> A=ones(1,3)
A =  1     1     1
>> B=ones(3)
B =
     1     1     1
     1     1     1
     1     1     1
```

Створення одиничної матриці. Функція **eye**

Синтаксис:

```
B=eye(n)
B=eye(n, m)
```

Функція eye(n) повертає одиничну матрицю розміром n-на-n. Функція eye(n, m) повертає одиничну матрицю розміром n-на-m.

Приклад:

```
>> eye(2)
ans =
     1     0
     0     1
>> eye(2,3)
ans =
     1     0     0
     0     1     0
```

Визначення довжини вектора. Функція **length**

Синтаксис:

```
numberOfElements = length(array)
```

Функція повертає кількість елементів масиву array (вздовж найбільшої розмірності, якщо масив багатомірний).

Приклад:

```
>> X = [5, 3.4, 72, 28/4, 3.61, 17 94 89]
```

```
X = 5.0000    3.4000   72.0000    7.0000    3.6100   17.0000   94.0000
89.0000
>> length(X)
ans =  8
```

Визначення розмірів масиву. Функція **size**

Синтаксис:

```
d = size(X)
[m,n] = size(X)
```

Функція d=size(X) для масиву X розміром повертає вектор d=[m,n], який відповідно вказує кількість рядків і стовпців.

Функція [m,n]=size(X) повертає кількість рядків і стовпців у вигляді двох змінних m і n.

Приклад:

```
>> X = [5, 3.4, 72, 28/4, 3.61, 17 94 89]
X = 5.0000    3.4000   72.0000    7.0000    3.6100   17.0000   94.0000
89.0000
>> d=size(X)
d =  1     8
>> [m, n]=size(X)
m = 1
n = 8
```

Деякі функції обробки даних

Обчислення скінченних різниць. Функція **diff**

Синтаксис:

```
Y=diff(X)
Y=diff(X, n)
```

Функція diff(X) обчислює скінченні різниці. Якщо X – одномірний масив виду X = [x(1) x(2) … x(n)], то diff(X) поверне масив різниць сусідніх елементів Y=[x(2) – x(1) x(3) – x(2) … x(n)–x(n–1)] Кількість елементів Y на одиницю менша ніж кількість елементів X.

Функція diff(X, n) обчислює скінченні різниці порядку n.

Приклад:

```
>> t=1:5
t = 1     2     3     4     5
>> x=t.^2
x = 1     4     9    16    25
>> y=diff(x)
y = 3     5     7     9
```

При наявності спеціалізованого пакету Symbolic Math Toolbox можна виконати диференціювання в символьному виді використовуючи однойменну функцію з цього пакету:

`Y=diff(S)` диференціює символьний вираз S за незалежною змінною.

`Y=diff(S, n)` диференціює n раз символьний вираз S.

`Y=diff(S, 'v')` диференціює символьний вираз S за v.

Приклад:

```
>> syms t
>> diff(sin(t^2))
ans = 2*t*cos(t^2)
```

Одномірна таблична інтерполяція. Функція **interp1**

Синтаксис:

```
yi = interp1(x,Y,xi)
yi = interp1(x,Y,xi,method)
```

Функція yi=interp1(x,Y,xi) будує кусково-лінійну інтерполюючи криву для одномірного масиву y, заданого на сітці x; вихідний масив yi може бути визначений на дрібнішій сітці xi.

Функція yi = interp1(x,Y,xi,method) дозволяє задати метод інтерполяції method:

‘nearest’ – ступінчата інтерполяція;

‘linear’ – лінійна інтерполяція;

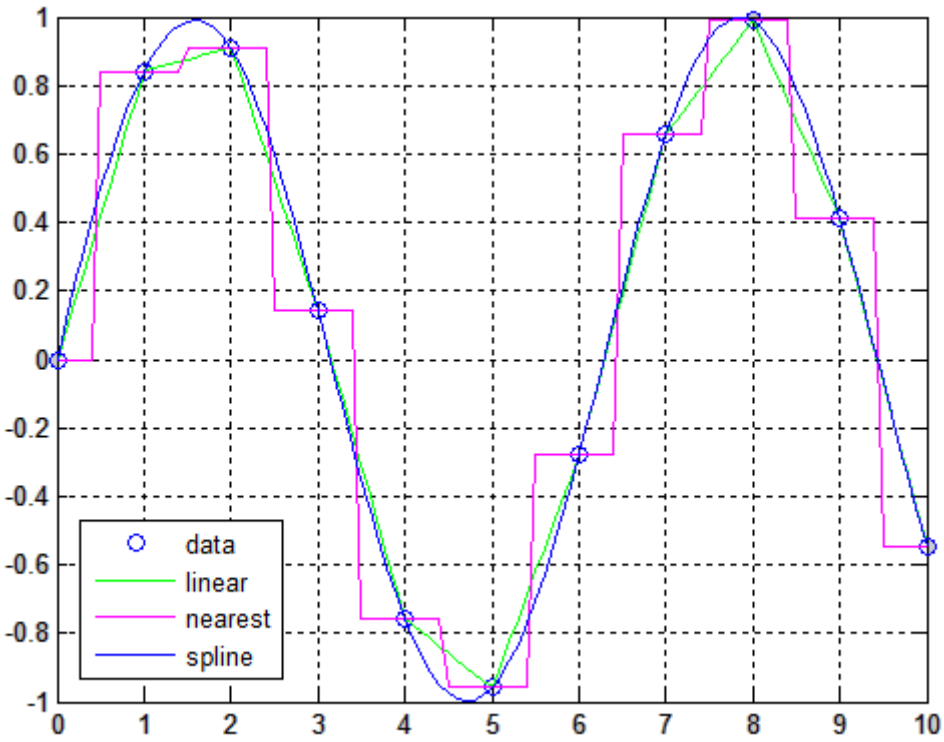
‘cubic’ – кубічна інтерполяція;

‘spline’ – кубічний сплайн.

Приклад:

Задамо синусоїду всього 10 точками і проведемо інтерполяцію, використовуючи дрібну сітку.

```
x=0:10; y=sin(x);
xi=0:0.1:10;
yi=interp1(x, y, xi);
plot(x, y, 'o', xi, yi, 'g'), hold on
yi=interp1(x, y, xi, 'nearest');
plot(xi, yi, 'm')
yi=interp1(x, y, xi, 'spline');
plot(xi, yi, 'b'), grid, hold off
legend('data', 'linear', 'nearest', 'spline')
```



Приклади програм

Інтерполяція за Лагранжем. Програма обчислює коефіцієнти полінома Лагранжа

$$P(x) = \sum_{k=0}^{N} y_k \, L_{N,k}(x),$$

основаного на $N + 1$ точках (x_k, y_k) при $k = 0, 1, \ldots, N$.

```
function [C,L] = lagran( X, Y )

%Вхід  - X - вектор абсцис (вузлів інтерполяції)
%      - Y - вектор ординат
%Вихід - C - матриця коефіцієнтів інтерполяційного поліному Лагранжа
%      - L - матриця коефіцієнтів Лагранжа
w=length(X);
n=w-1;
L=zeros(w,w);
%Формування коефіцієнтів поліному Лагранжа
    for k=1:n+1
        V=1;
        for j=1:n+1
            if k~=j
                V=conv(V,poly(X(j)))/(X(k)-X(j));
            end
        end
        L(k,:)=V;
    end

%Визначення коефіцієнтів інтерполяційного поліному Лагранжа
C = Y*L;
end
```

Інтерполяція за Ньютоном. Програма для побудови та обчислення полінома Ньютона степеня ≤ N, що проходить через $N + 1$ точок (x_k, y_k) для $k = 0, 1, \ldots, N$.

$$P(x) = d_{0,0} + d_{1,1}(x - x_0) + d_{2,2}(x - x_0)(x - x_1) + \cdots + d_{N,N}(x - x_0)(x - x_1) \ldots (x - x_{N-1}),$$

де

$$d_{k,0} = y_k \text{ і } d_{k,j} = \frac{d_{k,j-1} - d_{k-1,j-1}}{x_k - x_{k-j}}$$

```
function [C,D]=newpoly(X,Y)
%Вхід  - X вектор абсцис
%      - Y вектор ординат
%Вихід - C вектор коефіцієнтів інтерполяційного поліному Ньютона
%      - D таблиця розділених різниць
n=length(X);
D=zeros(n,n);
D(:,1)=Y';
%Формування таблиці розділених різниць
for j=2:n
    for k=j:n
        D(k,j) = (D(k,j-1)-D(k-1,j-1))/(X(k)-X(k-j+1));
    end
end
%Визначення коефіцієнтів інтерполяційного поліному Ньютона
C=D(n,n);
for k=(n-1):-1:1
    C=conv(C,poly(X(k)));
    m=length(C);
    C(m)=C(m)+D(k,k);
end
```

Інтерполяція сплайнами. Програма будує і обчислює інтерполяційний змикаючий кубічний сплайн $S(x)$ по $N + 1$ точках (x_k, y_k) при $k = 0, 1, \ldots, N$.

```
function S=csfit(X,Y,dx0,dxn)
%Вхід  - X - вектор абсцис розміру 1xn
%      - Y - вектор ординат розміру 1xn
%      - dx0 = S'(x0) - гранична умова на першу похідну в першій точці
%      - dxn = S'(xn) - гранична умова на першу похідну в останній точці
%Вихід - S: рядки S - це коефіцієнти в порядку зменшення
%        для інтерполяційного кубічного сплайна
N=length(X)-1;
H=diff(X);
D=diff(Y)./H;
A=H(2:N-1);
B=2*(H(1:N-1)+H(2:N));
C=H(2:N);
U=6*diff(D);
%Змикаючий сплайн с обмеженнями в крайніх точках
B(1)=B(1)-H(1)/2;
U(1)=U(1)-3*(D(1)-dx0);
B(N-1)=B(N-1)-H(N)/2;
U(N-1)=U(N-1)-3*(dxn-D(N));
for k=2:N-1
    temp=A(k-1)/B(k-1);
    B(k)=B(k)-temp*C(k-1);
    U(k)=U(k)-temp*U(k-1);
end
M(N)=U(N-1)/B(N-1);
for k=N-2:-1:1
    M(k+1)=(U(k)-C(k)*M(k+2))/B(k);
end
M(1)=3*(D(1)-dx0)/H(1)-M(2)/2;
M(N+1)=3*(dxn-D(N))/H(N)-M(N)/2;
for k=0:N-1
    S(k+1,1)=(M(k+2)-M(k+1))/(6*H(k+1));
    S(k+1,2)=M(k+1)/2;
    S(k+1,3)=D(k+1)-H(k+1)*(2*M(k+1)+M(k+2))/6;
    S(k+1,4)=Y(k+1);
end
```

Генератор випадково розподілених вузлів. Рядок, що дозволяє згенерувати N+1 випадково розподілених вузлів на інтервалі [a, b]. Перший вузол x_0 = a, останній – x_N = b.

```
x=sort([a a + (b-a).*rand(1,N-1) b])
```

Обраховує значення кубічного сплайна S для вектора x.

```
function y = cscalc(S, xi, x)
%calculates valuse of cubic spline S on vector x
%S  - spline matrix built on [a,b]
%x  - vector of points on which spline was built
%xi - vector of points on [a,b] to calculate spline values
%yi - vector of spline values in x

    N = length(x);
    y=zeros(1,N);
    j=1;
    for i=1:N
        if (x(i)>xi(j+1))
            j=j+1;
        end
        y(i)=polyval(S(j,:),x(i)-xi(j));
    end
end
```

Лабораторна робота № 5

Чисельне диференціювання та інтегрування функцій

Мета роботи

Отримати практичні навики використання методів диференціювання та інтегрування функцій.

Короткі теоретичні відомості до роботи

Чисельне інтегрування

Для наближеного обчислення інтеграла від обмеженої на проміжку $[a,b]$ функції

$$I(f)=\int_a^b f(x)dx$$

застосовують квадратурні формули

Квадратурна формула – це наближена рівність

$$I(f)=\int_a^b f(x)dx\approx\sum_{i=1}^{n}q_i f(x_i)=S_n(f) \qquad (1)$$

де q_i- деякі числа, які називають коефіцієнтами, а x_i- точки відрізку [a,b]

$$a\le x_1<x_2<\ldots<x_n\le b.$$

Їх називають вузлами квадратурної формули. Кожна квардратурна формула визначається набором вагових коефіцієнтів q_i та вузлів $x_i, i=1,2,\ldots,n.$ Для обчислення необхідна також інформація про підінтегральну функцію. Такою інформацією є обчислені значення функції $f(x)$ у вузлах $x_i, i=1,2,\ldots,n.$

Різниця

$$R(f)=\int_a^b f(x)dx-\sum_{i=1}^{n}q_i f(x_i)$$

називається похибкою квадратурної формули.

На практиці широко використовуються формули Ньютона-Котеса з рівновіддаленими вузлами. Наведемо найпростіші з них.

Нехай відрізок інтегрування [*a*, *b*] поділено на *N* рівних частин точками ділення $x_i=x_0+ih, i=1,2,\ldots N, x_0=a, x_N=b, h=\frac{b-a}{N}$

Формула лівих прямокутників

$$I(f)\approx h[f(x_0)+f(x_1)+\ldots+f(x_{N-1})]. \qquad (2)$$

Формула правих прямокутників

$$I(f)\approx h[f(x_1)+f(x_2)+\ldots+f(x_N)]. \qquad (3)$$

Формула середніх прямокутників

$$I(f)\approx h\left[f(x_0+\frac{h}{2})+f(x_1+\frac{h}{2})+\ldots+f(x_{N-1}+\frac{h}{2})\right]. \qquad (4)$$

Формула трапецій

$$I(f) \approx h\left[\frac{1}{2} f(x_0) + f(x_1) + \ldots + f(x_{N-1}) + \frac{1}{2} f(x_N)\right]. \quad (5)$$

Формула Симпсона $\left(N = 2m, h = \frac{b-a}{2m}\right)$

$$I(f) \approx \frac{h}{3}\{f(x_0) + 4\left[f(x_1) + f(x_3) + \ldots + f(x_{2m-1})\right] + 2\left[f(x_2) + f(x_4) + \ldots + f(x_{2m-2})\right] + f(x_{2m})\} \quad (6)$$

Похибки наведених формул мають вигляд. Для формул лівих та правих прямокутників

$$|R_{\text{ïð.}}(f)| \le h\frac{b-a}{2}\max_{a\le x\le b}|f'(x)|. \quad (7)$$

Для формули середніх прямокутників

$$|R_{\text{ñð.}}(f)| \le h^2\frac{b-a}{24}\max_{a\le x\le b}|f''(x)|. \quad (8)$$

Для формули трапецій

$$|R_{\text{òð.}}(f)| \le h^2\frac{b-a}{12}\max_{a\le x\le b}|f''(x)|. \quad (9)$$

Для формули Симпсона

$$|R_{\text{c}}(f)| \le h^4\frac{b-a}{180}\max_{a\le x\le b}|f^{(IV)}(x)|. \quad (10)$$

Наведені формули дозволяють оцінити порядок точності *p* квадратур по відношенню до кроку h. Для формул середніх прямокутників і трапецій *p* = 2, для формули Симпсона *p* = 4. Це означає, що зменшуючи крок інтегрування у λ разів, ми можемо розраховувати на зменьшення похибки у λ^p разів.

Якщо розглядати похибку $R(f)$ як функцію h, то, враховуючи сказане вище, $R(f)$ можна записати у вигляді

$$R(f) = R_h = Mh^p + o(h^p),$$

де p – порядок точності квадратури, M – деяка константа, а $o(h^p)$ – величина більш високого порядку малості ніж h^p

$$\lim_{h\to 0}\frac{o(h^p)}{h^p} = 0.$$

Принцип Рунге оцінки похибки

Центральною проблемою кожного алгоритму є оцінка його похибки. Формули (7)-(10) дозволяють за певних умов розв'язати цю проблему. Проте з практичної точки зору вони не є ефективними, тому що їх застосування пов'язане, як правило, з складними перетвореннями та дослідженнями похідних. Існують інші методи оцінки похибки, які засновані на інформації, яка використовується для розв'язання поставленої проблеми. Такою інформацією є обчислені значення функції. Для розв'язання такої задачі існують спеціальни методи. Одним з них є принцип Рунге, який базується на подвійному перерахунку. Він полягає в наступному.

Нехай для обчислення інтеграла

$$I(f) = \int_a^b f(x)dx$$

застосовується деяка квадратурна формула S з порядком точності p відносно h. Обчислимо двічи наближене значення інтегралів S_{h_1} та S_{h_2}, з кроком h_1 та h_2, $h_1 = \lambda h_2$ $(\lambda > 1)$. Тоді похибка $R(f)$

$$|R(f)| = |I(f) - S_{h_2}| \approx \frac{|S_{h_2} - S_{h_1}|}{\lambda^p - 1}. \quad (11)$$

Формула дозволяє наближено оцінити похибку обчислень.

Якщо точність ε задана, то для досягнення необхідної точності послідовно проводять обчислення, зменшуючи кожний раз крок у λ разів (λ – фіксоване число) поки не буде досягнена необхідна точність

$$\frac{|S_{h_2} - S_{h_1}|}{\lambda^p - 1} < \varepsilon. \qquad (12)$$

Після цього покладають

$$I(f) \approx S_{h_2}.$$

Як правило, таку процедуру застосовують, зменшуючи крок у 2 рази (λ =2).

Зауваження.

Як говорилось вище, формула (11) надає наближене значення оцінки похибки.

Формула виводиться за припущенням, що похибка може бути представлена у виді

$$R(f) = Mh^p,$$

де $h-$ шаг інтегруванння, $p-$ порядок точності, $M-$ константа.

Насправді похибка має іншу структуру

$$R(f) = Mh^p + o(h^p).$$

Тому умова (11) не завжди виконується. Для її підтведження необхідно зберегти три обчислених значення інтегралів $S_{h_1}, S_{h_2}, S_{h_3}$ з кроком $h_1, h_2 = \frac{h_1}{\lambda}, h_3 = \frac{h_2}{\lambda}$ відповідно та перевірити умову

$$\left| \frac{S_{h_2} - S_{h_1}}{S_{h_3} - S_{h_2}} - \lambda^p \right| < (\lambda^{p+1} - 1)\varepsilon, \qquad (13)$$

де $p-$порядок точності формули.

Чисельне диференціювання

Задача чисельного диференціювання полягає у знаходженні значень похідних функції *y* = *f*(*x*) у заданих точках у випадку, коли аналітичний запис функції *f*(*x*) невідомий або дуже складний чи функція задана таблично. Привабливість чисельного підходу здебільшого пояснюється наявністю простих залежностей, за допомогою яких похідні в заданих точках можна апроксимувати декількома значеннями функції в цих і близьких до них точках.

Конструювання формул наближеного диференціювання полягає в тому, що функцію *f*(*x*) на заданому відрізку [*a*, *b*] замінюють відповідною апроксимуючою функцією *φ*(x), а потім вважають, що похідні від функцій *f*(*x*) і *φ*(*x*) збігаються, наприклад:

$$f'(x) \approx \varphi'(x).$$

де

$$a \le x \le b.$$

Аналогічно знаходять похідні вищих порядків від функції *f(x).* При цьому апроксимуюча функція *φ*(*x*) найчастіше задається у вигляді полінома.

Формули чисельного диференціювання на основі першої інтерполяційної формули Ньютона. Нехай функція *f*(*x*) задана в рівноміддалених точках $x_i = a + ih$, $i = 0,1,\ldots, n$ відрізка [*a, b*] значеннями $f_i = f(x_i)$.

Щоб обчислити похідні *f*′(*x*), *f*′′(*x*) і т. д., замінимо *f*(*x*) інтерполяційним поліномом Ньютона для інтерполяції вперед, тобто

$$f(x) \approx f_0 + t\Delta f_0 + \frac{t(t-1)}{2!}\Delta^2 f_0 + \cdots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n f_0 \qquad (14)$$

де $t = (x - x_0)/h$.

Для отримання скінченно-різницевої формули чисельного диференціювання візьмемо похідні від обох частин тотожності, скориставшись співвідношеннями

$$f'(x) = \frac{df}{dx} = \frac{df}{dt}\frac{dt}{dx} = \frac{1}{h}\frac{df}{dt}.$$

Тоді

$$f'(x) \approx \frac{1}{h}\left[\Delta f_0 + \frac{2t-1}{2!}\Delta^2 f_0 + \frac{3t^2-6t+2}{3!}\Delta^3 f_0 + \frac{4t^3-18t^2+22t-6}{4!}\Delta^4 f_0 + \cdots\right] \quad (15)$$

Для обчислення другої похідної використаємо співвідношення

$$f''(x) = \frac{d(f'(x))}{dx} = \frac{d(f'(x))}{dt}\frac{dt}{dx} = \frac{1}{h}\frac{d(f'(x))}{dt}.$$

Тоді формула чисельного диференціювання матиме такий вигляд:

$$f''(x) \approx \frac{1}{h^2}\left[\Delta^2 f_0 + \frac{6t-6}{3!}\Delta^3 f_0 + \frac{12t^2-36t+22}{4!}\Delta^4 f_0 + \frac{20t^3-120t^2+210t-100}{5!}\Delta^5 f_0 + \cdots\right] \quad (16)$$

У разі потреби на основі інших інтерполяційних формул можна обчислити похідні функції будь-якого порядку.

Користуючись виразом (15), можна отримати формули чисельного диференціювання для інтерполяційних поліномів різних степенів. Для n = 1 маємо формулу чисельного диференціювання на основі лінійного інтерполяційного полінома:

$$f'(x) \approx \frac{\Delta f_0}{h}, x \in [x_0; x_1].$$

Для n = 2 маємо формулу чисельного диференціювання на основі квадратичного полінома

$$f'(x) \approx \frac{1}{h}\left[\Delta f_0 + \frac{2t-1}{2!}\Delta^2 f_0\right], x \in [x_0; x_2].$$

Формулу на основі кубічної інтерполяції отримуємо, приймаючи, що n = 3:

$$f'(x) \approx \frac{1}{h}\left[\Delta f_0 + \frac{2t-1}{2!}\Delta^2 f_0 + \frac{3t^2-6t+2}{3!}\Delta^3 f_0\right], x \in [x_0; x_3].$$

Слід відмітити, що в разі обчислення похідних за формулами наведеними вище у фіксованій точці x_i як x_0 варто вибирати найближче табличне значення аргументу.

Інколи виникає потреба обчислити значення похідної від функції $f(x)$ безпосередньо у вузлах інтерполяції, тобто для t = 0. У цьому випадку формули чисельного диференціювання спрощуються, наприклад з виразу (15) маємо:

$$f'(x) \approx \frac{1}{h}\left[\Delta f_0 - \frac{1}{2}\Delta^2 f_0 + \frac{1}{3}\Delta^3 f_0 - \frac{1}{4}\Delta^4 f_0 + \cdots\right] \quad (17)$$

На практиці похідні перших порядків від функцій, заданих таблично зручно знаходити за формулами з табл. 1.

Таблиця 1.

Формули для оцінок перших похідних

Тип формули	**Формула**	**Порядок точності**
Несиметричні обернені, чи формули диференціювання назад	$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h}$	$0(h)$
	$f'(x_i) = \frac{3f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{2h}$	$0(h^2)$
Несиметричні прямі, чи формули диференціювання уперед	$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$	$0(h)$
	$f'(x_i) = \frac{-f(x_{i+2}) + 4f(x_{i+1}) - 3f(x_i)}{2h}$	$0(h^2)$
Симетричні	$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$	$0(h^2)$
	$f'(x_i) = \frac{-f(x_{i+2}) + 8f(x_{i+1}) - 8f(x_{i-1}) + f(x_{i-2})}{12h}$	$0(h^4)$

Порядок виконання роботи

1. Написати власну функцію MATLAB для обчислення значень заданої підінтегральної функції $f(x)$ відповідно до варіанту (табл. 2).
2. Написати функції MATLAB, які реалізують обчислення визначеного інтегралу $I = \int_a^b f(x)dx$ методами прямокутників (2) та Симпсона (6) з заданою кількістю підінтервалів інтегрування (кроком інтегрування). Написані функції повинні приймати на вхід підінтегральну функцію у вигляді текстового рядка, верхню і нижню границі інтегрування кількість під інтервалів інтегрування. На виході функція повинна повертати наближене значення інтегралу.
3. Ознайомитись і навчитись користуватись стандартними функціями MATLAB для чисельного інтегрування: `trapz`, `quad`, `quad8` (в останніх версіях MATLAB `quad8` замінена на `quadl`). Також вивчити функцію MATLAB `int` для символьного інтегрування.
4. Скориставшись функцією MATLAB `int` для символьного інтегрування визначити точне значення визначеного інтегралу заданої згідно варіанту (табл. 2) функції $f(x)$ на заданому проміжку $[a, b]$. Для того щоб в результатах, які дає MATLAB побачити більше знаків після коми необхідно перед виконанням даного пункту змінити формат виводу командою `format long` (див. л.р.1). Знайти той самий інтеграл за допомогою власних функцій (написаних п.2) та функцій чисельного інтегрування (`trapz`, `quad`, `quad8/quadl`). Порівняти точність чисельного диференціювання для всіх використаних функцій знайшовши абсолютну та відносну похибки інтегрування відносно точного значення. Виконати власні функції збільшивши кількість підінтервалів інтегрування і знову перевірити точність. Порівняти кількість кроків (обчислень підінтегральної функції), необхідних кожній з функцій чисельного інтегрування для досягнення однієї і тієї ж точності (наприклад, 10^{-6}).
5. Написати власну функцію MATLAB для наближення першої похідної функції $f(x)$, яка задається таблично у рівновіддалених точках, на основі інтерполяційної формули Ньютона. Написана функція повинна приймати на вхід два вектори: вектор x_i, та вектор значень функції $y_i = f(x_i)$. На виході функція повинна повертати коефіцієнти полінома, який наближує похідну $f'(x)$ на заданому наборі точок.
6. Знайти символьний (аналітичний) вираз для першої похідної заданої за варіантом (табл. 2) функції $f(x)$ за допомогою стандартної функції MATLAB `diff`. Знайти точні значення похідної $f'(x)$ за її аналітичним виразом в точках $x_k = a + k\cdot(b - a)$, $k = 0, 0.25, 0.5, 0.75, 1$. Знайти наближені значення похідної в даних точках використовуючи поліном, отриманий в п.5. Порівняти точні та наближені значення, оцінити точність чисельного диференціювання.
7. Зробити висновки по роботі.

Варіанти до роботи

Таблиця 2

Вар	$f(x)$	$[a, b]$
1.	$1/(1+x^2)$	[0;1]
2.	$(1+x^2)/\exp(x)$	[0;1]
3.	$\ln(1+x)/(1+x)$	[0;1]
4.	$1+x^2+\ln(x)$	[0;1]
5.	$\cos(x)/(1+\sin(x))$	[0;1]
6.	$\sin(x)/x$	[1;2]
7.	$1/(1+2x-2x^2)$	[0;1]
8.	$\sin(-x)*(1+x^3)$	[0;1]
9.	$(1+x^3)/\exp(x)$	[0;1]
10.	$(x-1)/(1+x^3)$	[1;2]
11.	$\exp(x)$	[0;1]
12.	$1+x^2$	[0;1]
13.	$\ln(x)$	[1;2]
14.	$1/(1+x)$	[1;2]
15.	$\cos(x)$	[0;π]
16.	$\sin(x)$	[0;π]
17.	$1+2x^2-x$	[0;1]
18.	$1+x^3+x$	[0;1]
19.	$(1+x^3)/x$	[1;2]
20.	$x*\ln(x)+x^2$	[1;2]
21.	$\exp(x)/(1+\exp(x))$	[0;1]
22.	$(1+x^2)*\exp(x)$	[0;1]
23.	$\ln(x)*(1+x^2)$	[1;2]
24.	$(1+x)*(1+\ln(x))$	[1;2]
25.	$\cos(x)*(1+x)$	[0;1]
26.	$x*\sin(x)$	[0;2π]
27.	$\cos(x)*(1+2x)$	[0;1]
28.	$\exp(-x)*(1+x^3)$	[0;1]
29.	$(1+x^3)*\exp(x)$	[0;1]
30.	$\ln(x)*(1+x^3)$	[1;2]

MATLAB

Інтегрування функцій

Чисельне інтегрування методом трапецій. Функція **`trapz`**

Синтаксис:

```
I = trapz(x, Y)
```

Функція I = trapz(x, Y) обчислює визначений інтеграл від функції *y* по змінній *x* в заданих межах, використовуючи метод трапецій. Аргументи ***x*** і ***y*** можуть бути одновимірними масивами однакового розміру, або масив ***Y*** може бути двовимірним, але тоді повинна виконуватися умова size(Y, 1) = length(x). У останньому випадку обчислюється інтеграл для кожного стовпця.

Приклад:

```
>> X=0:pi/70:pi/2;
>> Y=cos(X);
>> Z = trapz(Y)
   Z = 22.2780
```

Чисельне інтегрування методом квадратур. Функції **`quad,quad8,quadl`**

Синтаксис:

```
I = quad(fun, a, b)
I = quad(fun, a, b, tol)
```

Квадратура – чисельний метод знаходження площі під графіком функції *f*(*x*), тобто обчислення певного інтеграла виду

$$I = \int_a^b f(x)dx$$

.

Всі ці функції використовують різні квадратурні формули для обчислення інтегралу. Функція quad виконує інтегрування по методу низького порядку, використовуючи рекурсивне правило Симпсона. Але вона може бути ефективнішою при нерівних підінтегральних функціях чи при низькій необхідній точності обчислень. Нова функція quadl (квадратура Лобатто) використовує адаптивне правило квадратури Гаусса-Лобатто дуже високого порядку. Застаріла функція quad8 виконувала інтеграцію, використовуючи квадратурні формули Ньютона-Котеса 8-го порядку.

У приведених формулах підінтегральний вираз **fun** зазвичай задається в прямих апострофах (текстовий вираз). Підінтегральний вираз можна також задавати у формі handle -функції. **a**, **b** – границі інтегрування, **tol** – відносна точність.

Приклад:

```
>> quad('exp(x)-1',0,1,1e-5)
  ans = 0.7183
>> q = quad('exp',0,2,1e-4)
  q = 6.3891
>> q = quad(@exp,0,2,1e-4)
  q = 6.3891
>> [q n] = quadl(@sin,0,pi,1e-4)
  q = 1.999999977471133
  n = 18
```

В останньому прикладі виводиться також кількість обчислень підінтегральної функції.

Символьне інтегрування. Функція **`int`**

Синтаксис:

```
I = int(fun)
I = int(fun, a, b)
```

У приведених формулах підінтегральний вираз **fun** може задаватися в прямих апострофах (текстовий вираз). Або це може бути вираз, заданий з використанням символьних змінних, визначених за допомогою функції syms. **a**, **b** – границі інтегрування, якщо необхідно обчислити визначений інтеграл.

Приклади:

```
>> syms x;
>> y = x/(1+x^2);
>> int(y)
  ans = 1/2*log(1+x^2)

>> I = int('x/(1+x^2)',1,5)
  I = 1/2*log(13)
>> eval(I)
  ans = 1.2825
```

В останньому прикладі використовується функція `eval`, яка обчислює виразна мові MATLAB.

Диференціювання функцій

Символьне диференціювання. Функція **`diff`**

Синтаксис:

```
Y=diff(S)
Y=diff(S, n)
```

При наявності спеціалізованого пакету Symbolic Math Toolbox можна виконати диференціювання в символьному виді використовуючи дану функцію.

Y=diff(S) диференціює символьний вираз S за незалежною змінною.

Y=diff(S, n) диференціює n раз символьний вираз S за незалежною змінною.

Y=diff(S, ‘v’) диференціює символьний вираз S за v.

Приклад:

```
>> F1=diff('sin(t^2)')
   F1 = 2*t*cos(t^2)
>> subs(F1,0:5)
   ans = 0    1.0806   -2.6146   -5.4668   -7.6613    9.9120
```

В останньому прикладі використовується функція `subs`, яка дозволяє виконувати підстановку в символьних виразах. Зокрема замість змінної t ми підставили вектор її значень і отримали значення функції при заданих значеннях t.

Інтегрування за формулою лівих прямокутників. Програма обчислює інтеграл за формулою трапецій

$$\int_a^b f(x)dx \approx h[f(x_0)+f(x_1)+\ldots+f(x_{M-1})].$$

```
function s=rectrl(f,a,b,M)
%Вхід   f - підінтегральна функція, вводиться як текстовий рядок
%       a і b - верхня і нижня границі інтегрування
%       M - число підінтервалів
%Вихід  s - сума формули прямокутників
h=(b-a)/M;
s=0;
for k=0:(M-1)
    x=a+h*k;
    s=s+eval(f);
end
s=h*s;
```

Інтегрування за формулою Симпсона. Програма обчислює інтеграл за формулою Симпсона

$$\int_a^b f(x)\,dx \approx \frac{h}{3}(f(a)+f(b)) + \frac{2h}{3}\sum_{k=1}^{M-1} f(x_{2k}) + \frac{4h}{3}\sum_{k=1}^{M} f(x_{2k-1})$$

```
function s=simprl(f,a,b,M)
%Вхід  f - підінтегральна функція, вводиться як текстовий рядок
%      a і b - верхня і нижня границі інтегрування
%      M - число підінтервалів
%Вихід s - сума формули Симпсона
h=(b-a)/(2*M);
s1=0;
s2=0;
for k=1:M
    x=a+h*(2*k-1);
    s1=s1+eval(f);
end
for k=1:(M-1)
    x=a+h*2*k;
    s2=s2+eval(f);
end
x=a;
f_a=eval(f);
x=b;
f_b=eval(f);
s=h*(f_a+f_b+4*s1+2*s2)/3;
```

Диференціювання з використанням наближня $f'(x)$ поліномом Ньютона. Програма для побудови та обчислення полінома Ньютона степеня ≤ N, що проходить через $N + 1$ точок (x_k, y_k) для $k = 0, 1, \ldots, N$.

$$P(x) = d_{0,0} + d_{1,1}(x - x_0) + d_{2,2}(x - x_0)(x - x_1) + \cdots + d_{N,N}(x - x_0)(x - x_1)\ldots(x - x_{N-1}),$$

де

$$d_{k,0} = y_k \text{ і } d_{k,j} = \frac{d_{k,j-1} - d_{k-1,j-1}}{x_k - x_{k-j}}$$

За знайденим поліномом Ньютона знаходиться наближення першої похідної заданої функції.

```
function [DF,C]=diffnew(X,Y)
%Вхід  X - вектор абсцис розміру 1xn
%      Y - вектор ординат розміру 1xn
%Вихід C - вектор коефіцієнтів інтерполяційного поліному Ньютона
%          поліному Ньютона N-го степеня
%     DF - вектор коефіцієнтів поліному апроксимуючого першу похідну
n=length(X);
D=zeros(n,n);
D(:,1)=Y';
%Формування таблиці розділених різниць
for j=2:n
    for k=j:n
        D(k,j) = (D(k,j-1)-D(k-1,j-1))/(X(k)-X(k-j+1));
    end
end
%Визначення коефіцієнтів інтерполяційного поліному Ньютона
C=D(n,n);
for k=(n-1):-1:1
    C=conv(C,poly(X(k)));
    m=length(C);
    C(m)=C(m)+D(k,k);
end
DF=polyder(C);
```