Міністерство освіти і науки України
Національний університет водного господарства
та природокористування
Навчально-науковий інститут енергетики, автоматики
та водного господарства
Кафедра автоматизації, електротехнічних та
комп'ютерно-інтегрованих технологій

04-03-423М

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з навчальної дисципліни
«**Чисельні методи та моделювання**»
(Частина 1. Чисельні методи) для здобувачів вищої освіти
першого (бакалаврського) ступеня за освітньо-професійною
програмою «Автоматизація, комп'ютерно-інтегровані
технології та робототехніка» спеціальності
174 «Автоматизація, комп'ютерно-інтегровані технології та
робототехніка» денної та заочної форм навчання

Рекомендовано науково-методичною
радою з якості ННІ ЕАВГ
Протокол № 06 від 28.01.2025 р.

Рівне – 2025

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Чисельні методи та моделювання» для здобувачів вищої освіти першого (бакалаврського) ступеня за освітньо-професійною програмою «Автоматизація, комп’ютерно-інтегровані технології та робототехніка» спеціальності 174 «Автоматизація, комп’ютерно-інтегровані технології та робототехнік»” денної та заочної форм навчання. Частина 1. [Електронне видання] / Сафоник А. П., Мащенко В. А., Подвишенний В. С. – Рівне : НУВГП, 2025. – 129 с.

Укладачі: Сафоник А. П., доктор технічних наук, професор, професор кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій; Мащенко В. А., кандидат фізико-математичних наук, доцент, доцент кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій; Подвишенний В. С., викладач кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій.

Відповідальний за випуск: Древецький В. В., доктор технічних наук, професор, завідувач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Керівник групи забезпечення: Христюк А. О., кандидат технічних наук, доцент, доцент кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій.

Попередній варіант методичної вказівки 04-03-199М.

Зміст

ВСТУП

Досвід розв'язування науково-дослідних і прикладних задач показує, що незалежно від їхньої складності кінцевої мети можна досягти або постановкою експерименту, або методом математичного моделювання. Кожен з цих методів має свої переваги і недоліки.

За допомогою експерименту можна розв'язувати навіть дуже складні задачі, при цьому достовірність результатів тим вища, чим ретельніше відпрацьована методика експерименту. Водночас здобуті результати будуть стосуватися тільки тих умов, за яких проводився експеримент, внаслідок чого узагальнення результатів на інші умови не коректне. Крім того, треба враховувати економічний бік постановки складного експерименту. Щодо цього, то більші можливості має метод математичного моделювання за допомогою ЕОМ, коли аналізують не реальну задачу, а її модельне представлення.

Вивчення математичних методів є однією з важливих частин у підготовці фахівців з автоматизації та комп’ютерно-інтегрованих технологій. Дисципліна «Чисельні методи та моделювання» покликана допомогти у підготовці фахівців з автоматизації для різних галузей сучасної промисловості. Під час вивчення даної дисципліни студенти здобудуть знання, які допоможуть застосовувати сучасні розробки в напрямку моделювання процесів та систем, що застосовуються в різних сферах діяльності.

Курс «Чисельні методи та моделювання» носить важливий характер при здобутті студентами знань та навиків моделювання. Вміння використовувати моделі та методи допоможе у формуванні повноцінних фахівців.

Лабораторна робота №1
Чисельне розв'язування нелінійних рівнянь

1.1. Мета роботи

Ознайомитися з чисельними методами розв’язування нелінійних рівнянь. Навчитися визначати інтервали ізоляції коренів нелінійних рівнянь з однією змінною та використовувати різні методи обчислення наближених значень розв’язків рівнянь із заданою точністю.

1.2. Теоретичні відомості

У загальному випадку аналітичний розв'язок рівняння:

$$f(x) = 0 \qquad (1.1)$$

можна знайти лише для певного класу функцій. Нелінійні рівняння вигляду (1.1) часто доводиться розв'язувати числовими методами. Чисельний розв'язок таких рівнянь зазичай знаходять у два етапи:

1) відокремлюють корені рівняння, тобто знаходять досить вузькі проміжки, в яких міститься тільки один корінь. Ці проміжки називають ***інтервалами ізоляції кореня***, і визначити їх можна, графічним (побудувати графік функції та знайти інтервал, у якому графік перетинається з віссю абсцис) або аналітичним методом. Застосування аналітичного методу ґрунтується на тому, що неперервна функція $f(x)$ має на інтервалі $[a, b]$ хоча б один корінь, якщо вона змінює знак на протилежний в точках a і b *(теорема Больцано – Коші)*, тобто:

$$f(a) \cdot f(b) < 0, \qquad (1.2)$$

при цьому a і b називають ***межами інтервалу ізоляції***.

При цьому умовою існування єдиного розв'язку на відрізку $[a, b]$ є монотонність функції. Тоді, якщо $f'(x)$ не змінює знак на інтервалі (a, b), то умова (1.2) є необхідною і достатньою для того, щоб рівняння $f(x) = 0$ мало єдиний корінь на відрізку $[a, b]$.

2) на другому етапі проводять уточнення відокремлених коренів, тобто знаходять корені рівняння із заданою точністю.

Розглянемо методи уточнення наближених значень коренів.

Нехай для рівняння (1.1) виділено відрізок *[a, b]*, на якому функція *f(x)* має єдиний корінь та виконується умова $f(a)\cdot f(b)<0$.

Метод поділу навпіл (метод бісекцій, або дихотомії)

Поділимо відрізок *[a, b]* навпіл точкою $c=(a+b)/2$. Якщо $f(c)\neq 0$, то можливі два випадки:

1) функція *f(x)* змінює знак на відрізку *[a, c];*

2) функція *f(x)* змінює знак на відрізку *[c, b]*.

Вибираючи в кожному випадку той відрізок, на якому функція змінює знак, продовжуємо процес половинного поділу до виконання умови $(b-a)/2^n<\varepsilon$, де n – число проведених поділів початкового відрізка навпіл, ε – задана точність. Тоді середина отриманого відрізка буде значенням кореня $x^*=c$ із заданою точністю ε.

Метод є збіжним для будь-яких неперервних функцій, хоча швидкість збіжності невелика.

Метод простих ітерацій

Замінимо рівняння $f(x) = 0$ рівносильним:

$$x=\varphi(x). \qquad (1.3)$$

Виберемо початкове наближення x_0 і підставимо в праву частину рівняння (1.3). Отримаємо:

$$x_1=\varphi(x_0). \qquad (1.4)$$

Підставляючи в праву частину рівняння (1.4) x_1 замість x_0, отримаємо $x_2=\varphi(x_1)$.. Повторюючи цей процес, матимемо послідовність чисел:

$$x_n=\varphi(x_{n-1}),\ \ n=1, 2, 3... \qquad (1.5)$$

Процес повторюється доти, допоки не виконуватиметься умова: $|x_n - x_{n-1}|<\varepsilon$, де ε – задана похибка.

Якщо послідовність $x_n=\varphi(x_{n-1})$ збіжна, тобто існує границя $\lim\limits_{x\to\infty} x_n = x^*$, то x^* є коренем рівняння (1.3).

Теорема. Нехай функція $\varphi(x)$ визначена та диференційована на проміжку *[a, b]* та існує таке число q, що $|\phi'(x)|\le q<1$, тоді ітераційний процес збігається до точного розв'язку x^* незалежно від початкового наближення $x_0\in[a, b]$.

Похибка методу. Метод простих ітерацій забезпечує на *n*-му кроці абсолютну похибку наближення до кореня рівняння (1.1), яка не перевищує довжини *n*-го відрізка, помноженої на дріб *q/(1-q):*

$$\left|x^* - x_n\right| \le \frac{q}{1-q} \cdot \left|x_n - x_{n-1}\right|,$$

де $q = \max\limits_{x\in[a,b]} |\phi'(x)|$.

Для того, щоб функція *φ(x)* забезпечувала збіжність послідовності (1.5), вона повинна мати вигляд: $\phi(x) = x - \frac{f(x)}{k}$,

де $|k| \ge \frac{Q}{2}$, $Q = \max\limits_{x\in[a,b]} |f'(x)|$, знак *k* збігається зі знаком *f'(x)* на *[a, b]*.

Метод Ньютона (дотичних)

Нехай функція *f(x)* є двічі диференційованою на проміжку [*a, b*], причому *f'(x)* та *f''(x)* не перетворюються в нуль на цьому проміжку.

За початкове наближення x_0 обираємо одну з меж проміжку ізоляції [*a, b*] за формулою:

$$x_0 = \begin{cases} a, \text{якщо } f(a)f''(a) > 0, \\ b, \text{якщо } f(b)f''(b) > 0. \end{cases}$$

Проведемо дотичну до кривої *y=f(x)* в точці $(x_0, f(x_0))$. Запишемо рівняння дотичної:

$$y=f(x_0)+f'(x_0)(x-x_0).$$

Знайдемо точку перетину дотичної з віссю абсцис, яка буде наближенням кореня:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Проведемо дотичну до кривої *y=f(x)* в точці x_1 та отримаємо наступне наближення кореня.

Таким чином побудуємо ітераційний процес:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}, \quad n = 1,\ 2,... \tag{1.6}$$

Процес повторюємо доти, поки не виконуватиметься умова: $|x_n - x_{n-1}| < \varepsilon$, де ($\varepsilon$ – задана похибка).

Необхідні умови збіжності методу Ньютона:

1) функція $f(x)$ – неперервна і диференційована на проміжку $[a, b]$, що містить корінь $x^* \in [a, b]$;

2) перша похідна $f'(x)$ є неперервна, відмінна від нуля і зберігає знак на проміжку ізоляції;

3) друга похідна $f''(x)$ є неперервна, відмінна від нуля і зберігає знак на проміжку ізоляції;

4) початкове наближення x_0: $f(x_0)f''(x_0)>0$.

Теорема. Якщо виконуються необхідні умови (1 – 4), то ітераційний процес Ньютона (1.6) збігається до розв'язку x^* рівняння (1.1) з квадратичною швидкістю.

Метод хорд

Метод ґрунтується на заміні функції $f(x)$ на хорду на кожному проміжку пошуку, перетин якої з віссю OX дає наближення розв'язку. При цьому в процесі пошуку сімейство хорд може будуватися двома способами:

а) за фіксованого лівого кінця хорд ($z = a$), тоді початкова точка буде $x_0 = b$;

б) за фіксованого правого кінця хорд, тобто ($z = b$), тоді початкова точка буде $x_0 = a$.

Наступне наближення розв'язку рівняння за методом хорд обчислюється за рекурентними формулами:

- для випадку **а**:

$$x_0 = b,\ x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(a)}(x_n - a);$$

- для випадку **б**:

$$x_0 = a,\ x_{n+1} = x_n - \frac{f(x_n)}{f(b) - f(x_n)}(b - x_n)\ .$$

Процес пошуку триває до виконання умови:

$$|x_n - x_{n-1}| < \varepsilon.$$

Умови використання методу хорд:

1) функція $f(x)$ є неперервна на проміжку $[a, b]$;

2) перша похідна $f'(x)$ – неперервна і зберігає знак на проміжку ізоляції;

3) друга похідна $f''(x)$ – неперервна і зберігає знак на проміжку ізоляції.

Метод забезпечує швидку збіжність, якщо $f(z) \cdot f''(z) > 0$, тобто хорди фіксуються в тому кінці інтервалу $[a, b]$, де знаки функції $f(z)$ та її кривизни $f''(z)$ співпадають.

Приклад.

Нехай необхідно знайти наближений розв'язок нелінійного рівняння: $x^3-4x-e^x=0$.

1. Знайдемо інтервали ізоляції коренів графічним способом. Для цього побудуємо наближений графік функцій $f(x)=x^3-4x-e^x$ в середовищі Matlab. Виберемо початковий проміжок $x \in [-5, 5]$.

```
>> fplot('x^3-4*x-exp(x)',[-5 5])
>> grid on
```

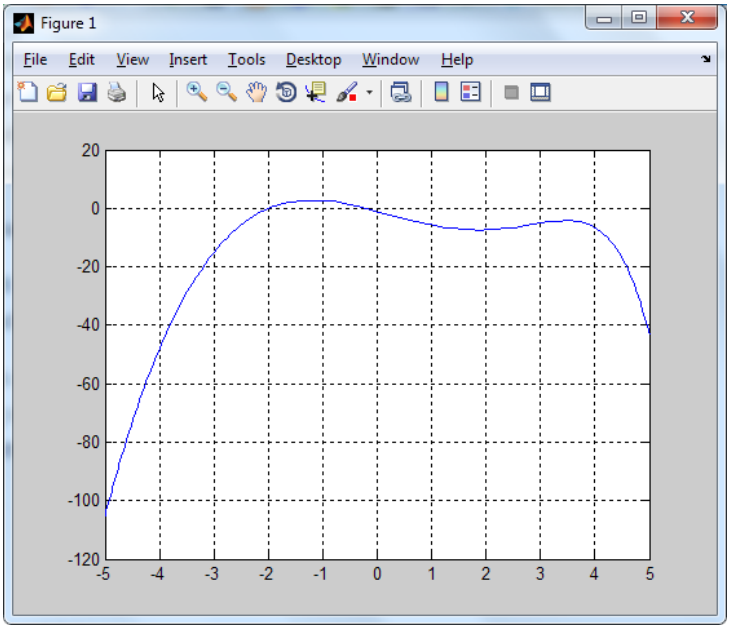


Рис. 1.1. Графік функції $f(x)=x^3-4x-e^x$ на проміжку $x \in [-5, 5]$

Як бачимо, графік функції перетинає вісь абсцис принаймні в двох точках. Отже, маємо інтервали ізоляції [-2.5, -1.5] та [-1, 0].

Виберемо проміжок $x \in [-1, 0]$.

```
>> fplot('x^3-4*x-exp(x)',[-1 0])
>> grid on
```

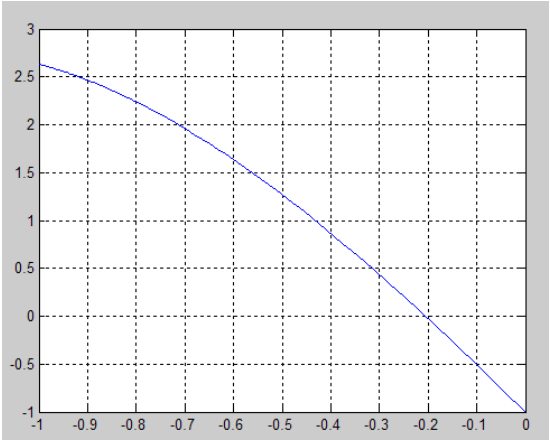


Рис. 1.2. Графік функції $f(x)=x^3-4x-e^x$ на проміжку $x\in[-1, 0]$

З рис. 1.2 бачимо, що на проміжку $x\in[-1, 0]$ функція $f(x)$ має єдиний корінь та виконується умова $f(-1)\cdot f(0)<0$, отже, будемо вважати цей проміжок інтервалом ізоляції кореня заданого рівняння.

2. Вибравши як перше наближення один із кінців початкового відрізка, уточнимо корінь методом простих ітерацій з точністю $\varepsilon=0{,}001$.

Для цього знайдемо похідну $f'(x)=3x^2-4-e^x$ та побудуємо її графік на проміжку $x\in[-1, 0]$:

```
>> fplot('3*x^2-4-exp(x)',[-1 0])
>> grid on
```

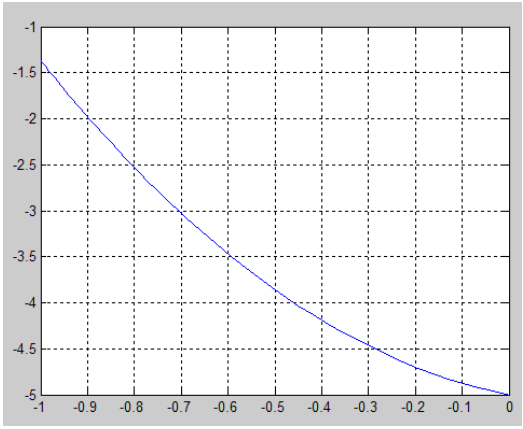


Рис. 1.3. Графік функції $f'(x)=3x^2-4-e^x$ на проміжку $x\in[-1, 0]$

Знаходимо $Q = \max\limits_{x\in[-1,0]} |f'(x)| = |f'(0)| = 5$.

Виберемо *k*, що задовільняє умову $|k| \ge \frac{Q}{2}$.

Оскільки $f'(x)<0$ на проміжку $x\in[-1, 0]$, то виберемо $k=-3$.

Тоді функція $\varphi(x)$ буде мати вигляд:

$$\phi(x) = x - \frac{f(x)}{k} = x + \frac{x^3 - 4x - e^x}{3} = \frac{x^3 - x - e^x}{3}.$$

Знайдемо похідну функції $\varphi(x)$ і побудуємо графік цієї функції на відрізку $x\in[-1, 0]$.

$$\phi'(x) = \frac{3x^2 - 1 - e^x}{3}.$$

```
>> fplot('1/3*(3*x^2-1-exp(x))',[-1 0])
>> grid on
```

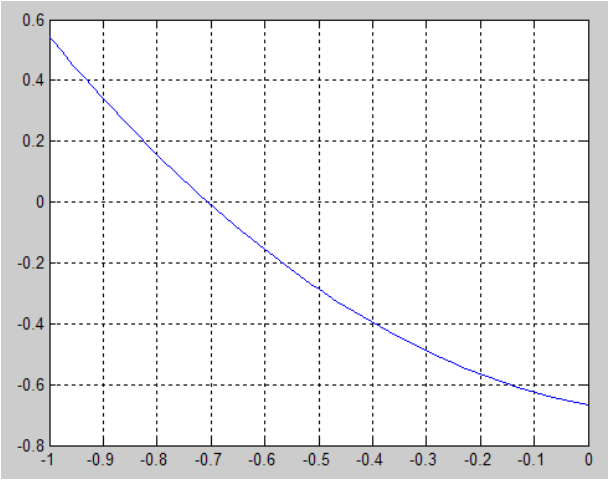


Рис. 1.4. Графік функції $\phi'(x) = \frac{3x^2 - 1 - e^x}{3}$

Тоді $q = \max\limits_{x\in[-1,0]} |\phi'(x)| = |\phi'(0)| \approx 0,67 < 1$. Приймемо за початкове наближення правий кінець відрізка $x_0=0$. Обчислення будемо виконувати до виконання умови:

$$|x_n - x_{n-1}| \le \frac{q}{1-q} \cdot \varepsilon = \frac{0,67}{1-0,67} \cdot 0,001 \approx 0,002.$$

Запишемо ітераційний процес $x_n=\varphi(x_{n-1}),\ \ n=1, 2, 3...$ в Matlab до виконання умови $|x_n - x_{n-1}| < \varepsilon = 0,001$:

```
>> fi = inline('1/3*(x^3-x-exp(x))');
eps=0.001;x0=0;
iter=0; x=x0;
r=100;
while (abs(r)>eps)
    xn=fi(x);
    r=xn-x;
     x=xn;
    iter=iter+1;
end
x
iter
Отримаємо відповідь:
x =
   -0.2055
iter =
    12
```

Отже, наближений корінь рівняння $x^3-4x-e^x=0$ на проміжку [-1 0] буде $x^*=-0,206$ з точністю $\varepsilon=0,001$.

Проведемо перевірку:

```
>> f=inline('x^3-4*x-exp(x)')
f =
     Inline function:
     f(x) = x^3-4*x-exp(x)

>> f(-0.206)
ans =
    0.0014
```

Отже, розв'язок знайдений правильно.

3. Знайдемо наближений корінь рівняння $x^3-4x-e^x=0$ методом Ньютона з точністю $\varepsilon=10^{-6}$.

Знайдемо другу похідну функції та побудуємо її графік:

$f''(x)=6x-e^x$.

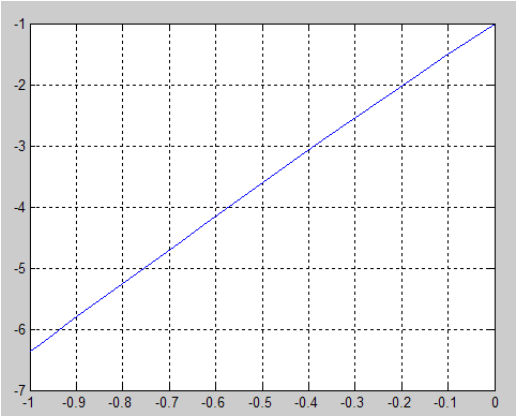


Рис. 1.5. Графік функції $f''(x)=6x-e^x$

За початкове наближення візьмемо $x_0=0$, оскільки $f(0)f''(0)>0$.
Ітераційний процес методу Ньютона:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}, \quad n = 1,\ 2,...$$

Складемо програму в Matlab.

```
>> f=inline('x^3-4*x-exp(x)');
df=inline('3*x^2-4-exp(x)');
eps=10e-6;
x0=0;
iter=0; x=x0;
r=100;
while (abs(r)>eps)
    xn=x-f(x)/df(x);
    r=xn-x;
  x=xn;
    iter=iter+1;
end
x
iter
```

Отримаємо відповідь:

```
x =
   -0.2057
iter = 3
```

Для відображення більшої кількості значущих цифр скористаємося командою `vpa()`:

```
>> vpa(x,7)
 ans =
 -0.2056959
```

Функція `vpa` (абревіатура від `variable-precision arithmetic`) використовується для задання кількості значущих цифр змінної.

Отже, наближений корінь рівняння $x^3-4x-e^x=0$ на проміжку [-1 0], обчислений методом Ньютона буде $x^*=-0,205696$ з точністю $\varepsilon=0,000001$.

Перевірка:

```
>> f(-0.205696)
ans =
 3.0940e-007
```

1.3. Програма роботи

1. Ознайомитися з деякими методами наближеного розв'язування нелінійних рівнянь.
2. Розв'язати задане нелінійне рівняння вказаними чисельними методами.
3. Зробити висновки про швидкість збіжності методів. Оформити звіт про виконання роботи.

1.4. Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями.
2. Вибрати завдання згідно варіанту (за номером у журналі групи).
3. Завантажити середовище MATLAB. Локалізуйте корінь (або один із коренів, якщо рівняння має кілька розв'язків) рівняння $f(x)=0$ графічним методом на проміжку довжиною не менше 1.
4. Вибравши один із кінців проміжку ізоляції як початкове наближення, уточніть корінь методом простих ітерацій в середовищі MATLAB з точністю до 0,0001.
5. Знайдіть корінь заданого рівняння методом Ньютона з точністю 10^{-6}. Вкажіть кількість ітерацій.
6. Знайдіть корені рівняння методом бісекцій та методом хорд з точністю 0,0001. Вкажіть кількість ітерацій для кожного з методів.

7. Виконайте перевірку розв'язків.

8. Зробіть висновки про швидкість збіжності кожного з методів. Оформіть звіт про виконання роботи.

Звіт повинен містити: титульний лист; тему, мету роботи; варіант завдання; послідовність дій для виконання завдання; результати обчислень; висновки.

1.5. Варіанти завдань

№	Рівняння	№	Рівняння
1	$3x^2 - 0,5^x - 1 = 0$	16	$0,5x + \lg(x) = 1$
2	$2^x - x - 4 = 0$	17	$x^3 + x - 4 = 0$
3	$ctg(x + 0,5) - \frac{x}{3} = 0$	18	$x^3 - 0,5x^2 + x + 3 = 0$
4	$x^3 + x^2 - 2x + 3 = 0$	19	$x^3 - x^2 + 2x + 3 = 0$
5	$x^2 - 1 + 2\sin(x) = 0$	20	$x^2 - 4\cos(x) = 0$
6	$x^3 - 2x - 7 = 0$	21	$x^3 - 3x - 4 = 0$
7	$4x - 2\cos(x) - 1 = 0$	22	$4x - \cos(x) - 2 = 0$
8	$x + 2 \cdot \lg(x) = 0,5$	23	$x + 2 \cdot \lg(x) = 1,45$
9	$tg(0,2x + 0,3) = x^2 - 1$	24	$tg(0,5x - 0,3) = x^2 - 1$
10	$x^3 - 1,3x^2 + x - 1 = 0$	25	$x^3 - 3x^2 + 4x - 5 = 0$
11	$x^4 - x - 1 = 0$	26	$\lg(x) - \frac{4}{2x + 1} = 0$
12	$x^2 - 20\sin(x) = 0$	27	$5x + \lg(x) = 3$
13	$2 \cdot \lg(x) - \frac{x}{2} + 1 = 0$	28	$x^3 - 3x^2 + x - 2 = 0$
14	$2x^2 - 0,5^x - 3 = 0$	29	$x^3 - 2x^2 + 2x - 3 = 0$
15	$2^x - 3x - 2 = 0$	30	$2e^x + 5x + 1 = 0$

1.6. Контрольні запитання

1. Що таке інтервал ізоляції кореня рівняння?

2. Яка необхідна умова існування кореня функції на проміжку [*a*, *b*]?

3. Яка необхідна і достатня умова існування кореня функції на проміжку [*a*, *b*]?

4. Який порядок чисельного розв'язку нелінійних алгебраїчних рівнянь?

5. У чому полягає метод бісекцій (дихотомії)?

6. У чому полягає метод простих ітерацій?

7. Які умови збіжності методу Ньютона?

8. Яка швидкість збіжності методу Ньютона?

9. У чому полягає метод хорд?

10. Для чого використовується функція `vpa()` у Matlab?

11. Для чого використовується функція `inline()` у Matlab?

12. Як побудувати графік функції на заданому проміжку у середовищі у Matlab?

Лабораторна робота №2
Чисельне розв'язування систем лінійних алгебраїчних рівнянь

2.1. Мета роботи

Ознайомитися з чисельними методами розв’язування систем лінійних алгебраїчних рівнянь. Навчитися знаходити розв'язки СЛАР точними та наближеними методами.

2.2. Теоретичні відомості

2.2.1. Загальні відомості

Розв’язування систем лінійних алгебраїчних рівнянь (СЛАР) є важливою обчислювальною задачею, до якої зводиться велика кількість прикладних розрахункових задач (наприклад, розрахунок параметрів електричних кіл, аналіз рівноваги сил у жорсткій системі балок, або дослідження умов та параметрів рівноваги хімічної реакції тощо). Практична цінність чисельного методу визначається швидкістю і ефективністю отримання розв'язку. Розглянемо деякі чисельні методи і ефективні алгоритми розв'язування СЛАР.

Системою лінійних алгебраїчних рівнянь (СЛАР) називають систему вигляду:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m; \end{cases} \qquad (2.1)$$

де a_{ij} – коефіцієнти системи; x_j – невідомі; b_i – вільні члени системи; $i = \overline{1,m},\ j = \overline{1,n}$.

Запишемо систему рівнянь (2.1) у матричному вигляді:

$$AX=B, \qquad (2.2)$$

де $A = \begin{pmatrix} a_{11}\ a_{12}\ \dots a_{1n} \\ a_{21}\ a_{22}\ \dots a_{2n} \\ \vdots \\ a_{m1}\ a_{m2} \dots a_{mn} \end{pmatrix}$; $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$; $B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$.

Розв'язати СЛАР – означає знайти такі x_j, що перетворюють кожне рівняння системи (2.1) в тотожність.

Кількість невідомих m у СЛАР називають *порядком* системи. Якщо СЛАР має хоча б один ненульовий розв'язок, то її називають *сумісною*, у протилежному випадку – *несумісною.* Систему рівнянь називають *визначеною,* якщо вона має тільки один розв'язок (при $m = n$). Систему називають *невизначеною,* якщо вона має безліч розв'язків ($m \neq n$). СЛАР називають *виродженою,* якщо її головний визначник дорівнює нулю, і *невиродженою* – у протилежному випадку (det $A \neq 0$). Системи називають *еквівалентними,* якщо вони сумісні, визначені і мають однаковий розв'язок.

СЛАР можна розв'язати чисельними методами, якщо вона сумісна, визначена і невироджена.

Отже, необхідною і достатньою умовою існування єдиного розв'язку СЛАР є: det $A \neq 0$, тобто визначник головної матриці системи повинен бути відмінним від нуля. Ця умова поширюється на СЛАР будь-якого порядку.

2.2.2. Чисельні методи розв'язування СЛАР

Для знаходження розв'язків СЛАР на ЕОМ використовують дві групи чисельних методів:

1) точні (метод Гауса, метод Гауса з вибором головного елемента, метод Гауса з одиничною матрицею, метод Гауса з перетвореною матрицею, метод Гауса-Халецького, метод Гауса-Жордана, метод Крамера);

2) наближені (метод послідовних ітерацій, метод Зейделя, метод векторів зміщень тощо).

Точними є методи, які дозволяють отримати точний розв'язок системи (2.1) за відповідну кількість операцій перетворення без урахування похибок заокруглення.

До **наближених (ітераційних)** належать методи, які дозволяють отримати розв'язок системи (2.1) у вигляді границі послідовності

векторів $\lim\limits_{k\to\infty}\left\{\overline{X}^0, \overline{X}^1, \overline{X}^2, ...\overline{X}^n\right\}$, яка збігається до точного розв'язку системи, де:

$$\overline{X}^0 = \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \\ . \\ . \\ x_n^{(0)} \end{bmatrix}, \quad \overline{X}^1 = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ . \\ . \\ x_n^{(1)} \end{bmatrix}, \quad ... \quad \overline{X}^n = \begin{bmatrix} x_1^{(n)} \\ x_2^{(n)} \\ . \\ . \\ x_n^{(n)} \end{bmatrix}$$

Методом Крамера можна розв'язувати СЛАР будь-якого порядку n. Але зі зростанням n метод стає дуже громіздким, оскільки число арифметичних операцій для обчислення визначника приблизно рівна $(n+1)!$, а визначників є $(n+1)$.

Можна знаходити розв'язок СЛАР (2.2) при $m=n$ з використанням **оберненої матриці** A^{-1}. Помноживши зліва рівняння (2.2) на A^{-1}, одержуємо $X=A^{-1}B$. Однак за кількістю арифметичних операцій знаходження оберненої матриці не простіше за формули Крамера.

Найпоширенішим обчислювальним методом розв'язування СЛАР є метод, запропонований німецьким математиком Карлом Фрідріхом Гаусом.

Особливості методів Гауса

Найбільш відомим з точних методів розв'язання СЛАР (2.1) є методи Гауса, суть яких полягає в тому, що система рівнянь, яка розв'язується, зводиться до еквівалентної системи з верхньою трикутною матрицею. Невідомі знаходяться послідовними підстановками, починаючи з останнього рівняння перетвореної системи. Алгоритми Гауса складаються із виконання однотипних операцій, які легко формалізуються. Однак, точність результату й витрачений на його отримання час у більшості випадків залежить від алгоритму формування трикутної матриці системи. У загальному випадку алгоритми Гауса складаються з двох етапів:

Прямий хід, в результаті якого СЛАР (2.1), що розв'язується, перетворюється в еквівалентну систему з верхньою трикутною матрицею коефіцієнтів виду:

$$\begin{cases} a_{11}\cdot x_1 & +a_{12}\cdot x_2 & +...+ & a_{1n}\cdot x_n & = b_1 \\ 0\cdot x_1 & +a_{22}\cdot x_2 & +...+ & a_{2n}\cdot x_n & = b_2 \\ 0\cdot x_1 & +0\cdot x_2 & +...+ & a_{nn}\cdot x_n & = b_n \end{cases} \qquad (2.3)$$

Зворотний хід дозволяє визначити вектор розв‘язку починаючи з останнього рівняння системи (2.3) шляхом підстановки координат вектора невідомих, отриманих на попередньому кроці.

Відомо декілька різних алгоритмів отримання еквівалентної системи з верхньою трикутною матрицею.

2.2.3. Метод Гауса з послідовним виключенням невідомих

Розглянемо базовий метод Гауса для розв'язування системи (2.1). Цей метод полягає в послідовному виключенні невідомих із системи. Припустимо, що $a_{11} \neq 0$. Послідовно помножимо перше рівняння на $-\frac{a_{i1}}{a_{11}}$ і додамо з *i*-м рівнянням. Таким чином виключимо x_1 зі всіх рівнянь системи. Отримаємо:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1, \\ \quad a_{22}^{(1)}x_2 + ... + a_{2n}^{(1)}x_n = b_2^{(1)}, \\ \quad ..... \\ \quad a_{n2}^{(1)}x_2 + ... + a_{nn}^{(1)}x_n = b_n^{(1)}, \end{cases}$$

де $a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}$, $b_i^{(1)} = b_i - \frac{a_{i1}b_1}{a_{11}}$, $i, j = 2,3...n.$

Аналогічно виключимо x_2 з отриманої системи. Послідовно виключивши всі невідомі до x_n, отримаємо систему трикутного вигляду:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1k}x_k + ... + a_{1n}x_n = b_1, \\ \quad a_{22}^{(1)}x_2 + ... + a_{2k}^{(1)}x_k + ... + a_{2n}^{(1)}x_n = b_2^{(1)}, \\ \quad ..... \\ \quad a_{n-1,n-1}^{(n-1)}x_{n-1} + a_{n-1,n}^{(n-1)}x_n = b_{n-1}^{(n-1)}, \\ \quad a_{n,n}^{(n-1)}x_n = b_n^{(n-1)}. \end{cases} \qquad (2.4)$$

Виконання описаної вище процедури прямого ходу можливе при умові, що всі $a_{i,i}^{(l)}$, $l = 1,2...n-1$ не дорівнюють нулю.

У результаті зворотного ходу методу Гауса, виконуючи послідовні підстановки (починаючи з останнього рівняння) в системі (2.4), отримаємо всі значення невідомих:

$$x_n = \frac{b_n^{(n-1)}}{a_{n,n}^{(n-1)}}, \qquad x_i = \frac{1}{a_{i,i}^{(i-1)}}(b_i^{(i-1)} - \sum_{j=i-1}^{n} a_{ij}^{(i-1)} x_j).$$

Метод Гауса може бути легко реалізований на комп'ютері. Цей метод та його модифікації вирізняються меншою кількістю арифметичних дій, приблизно рівною n^3. Однак, один з основних недоліків методу Гауса пов'язаний з тим, що при його реалізації накопичуються обчислювальні похибки, які спотворюють розв'язок великих погано обумовлених СЛАР.

Для вирішення цієї проблеми був створений метод *QR*-розкладу, що майже усунув ці похибки.

Однією з найбільш поширених модифікацій методу Гауса є метод ***LU*-розкладу**, що полягає у розкладі матриці СЛАР на добуток двох матриць, нижньої трикутної *L* та верхньої трикутної *U*: $A=LU$. Тоді систему $AX=B$ розв'язують у два етапи: спочатку розкладають матрицю A на добуток LU (прямий хід методу Гауса), а потім послідовно розв'язують системи $LY=B$ та $UX=Y$ (зворотний хід).

***QR*-розклад** полягає у розкладі матриці A на добуток дійсної ортогональної матриці Q і верхньої трикутної R. Поетапний розв'язок системи $AX=B$ виконують так: спочатку обчислюють $Y=Q^{\mathrm{T}}\cdot B$, а потім розв'язують систему $RX=Y$. *QR*-розклад є складнішим, ніж *LU*-розклад, а погано обумовлені системи не так часто зустрічаються.

Для того, щоб зменшити зростання обчислювальної похибки застосовуються різні модифікації методу Гауса. Наприклад, метод Гауса з вибором головного елемента по стовпцях, в цьому випадку на кожному етапі прямого ходу рядка матриці переставляються таким чином, щоб діагональний кутовий елемент був максимальним. При виключення відповідного невідомого з інших рядків поділ буде вироблятися на найбільший з можливих коефіцієнтів і отже відносна похибка буде найменшою.

2.2.4. Метод простих ітерацій

При великій кількості невідомих у СЛАР реалізація методу Гауса є досить складною. Тому для знаходження коренів системи інколи зручніше користуватись наближеними числовими методами. Ітераційні методи забезпечують отримання коренів системи із заданою точністю шляхом збіжних нескінченних процесів (наприклад, метод ітерації, метод Зейделя, метод релаксації та інші).

При використанні ітераційних процесів до похибок округлень додається похибка методу.

Крім того, ефективне застосування ітераційних методів істотно залежить від вдалого вибору початкового наближення та швидкості збіжності процесу.

Розглянемо метод простих ітерацій для розв'язування СЛАР.

Спочатку необхідно СЛАР (2.1) привести до **нормального вигляду** (зручного для ітерації):

$$x_i = \sum_{j=1}^{n} \alpha_{ij} x_j + \beta_i, \qquad (i = 1, 2, \ldots, n)$$

З цією метою розв'яжемо перше рівняння системи (2.1) відносно x_1, друге – відносно x_2 і т. д. Отримаємо систему:

$$\begin{cases} x_1 = \beta_1 + \alpha_{12} x_2 + \alpha_{13} x_3 + \ldots + \alpha_{1n} x_n, \\ x_2 = \beta_2 + \alpha_{21} x_1 + \alpha_{23} x_3 + \ldots + \alpha_{2n} x_n, \\ \vdots \\ x_n = \beta_n + \alpha_{n1} x_1 + \alpha_{n2} x_2 + \ldots + \alpha_{n,n-1} x_{n-1}. \end{cases} \qquad (2.5)$$

де $\beta_i = \dfrac{b_i}{a_{ii}}$; $\alpha_{ij} = -\dfrac{a_{ij}}{a_{ii}}$, при $i \neq j$; $\alpha_{ij} = 0$, при $i = j$;

$x_i = -\sum\limits_{\substack{j=1 \\ j \neq i}}^{n} \dfrac{a_{ij}}{a_{ii}} x_j + \dfrac{b_i}{a_{ii}}$; $a_{ij} \neq 0$; $i = \overline{1,n}$.

Нульове наближення розв'язку системи (2.5) можна вибрати довільним. Виберемо, наприклад, стовпчик вільних членів $\mathbf{x}^{(0)} = \boldsymbol{\beta}$. Далі послідовно побудуємо матриці-стовпці наступних наближень розв'язку системи (2.5):

$x^{(1)} = \beta + \alpha x^{(0)}$ – перше наближення,

$x^{(2)} = \beta + \alpha x^{(1)}$ – друге наближення і т.д.

Будь-яке (k + 1)-е наближення обчислюється за формулою:

$$x^{(k+1)} = \beta + \alpha x^{(k)}, (k = 0, 1, 2, \ldots). \quad (2.6)$$

У розгорнутому вигляді $x_i^{(k+1)} = \beta_i + \sum_{j=1}^{n} \alpha_{ij} x_j^{(k)}$.

Якщо послідовність наближень $x^{(0)}$, $x^{(1)}$, ..., $x^{(k)}$ має границю

$$x = \lim_{k \to \infty} x^{(k)}, \quad (2.7)$$

то ця границя є розв'язком системи (2.5).

Ітераційний процес зупиняють, при умові $\left| x_i^{(k)} - x_i^{(k-1)} \right| < \varepsilon$, де ε – задана абсолютна похибка.

Значно простішою для програмної реалізації є така формула методу простих ітерацій:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{n} a_{ij} \cdot x_j^{(k)} \right). \quad (2.8)$$

Умови збіжності методу простих ітерацій

Нехай задано зведену до нормального вигляду СЛАР

$$x = \beta + \alpha \cdot x.$$

Достатня умова збіжності: якщо сума модулів елементів рядків або модулів елементів стовпців матриці α менша 1, то процес ітерації для даної системи збігається до єдиного розв'язку незалежно від вибору вектора початкових наближень.

Необхідна і достатня умова: ітераційний метод розв'язування системи (2.2) є збіжним, якщо модулі діагональних коефіцієнтів a_{ii} для кожного рівняння системи більші, ніж суми модулів решти коефіцієнтів даного рівняння (не враховуючи вільних членів).

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} \left| a_{ij} \right| < \left| a_{ii} \right|, \ i=1, 2, ..., n.$$

2.2.5. Метод Зейделя

Розглянемо метод Зейделя для розв'язування СЛАР, що зведена до нормального вигляду (2.5).

Виберемо вектор початкових наближень

$$X^{(0)} = \left\{ x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, ..., x_n^{(0)} \right\}$$

(наприклад, стовпець вільних членів $\beta = \{\beta_1, \beta_2, \beta_3, ..., \beta_n\}$). Тоді проводимо уточнення значень невідомих чином:

1) перше наближення:

$$\begin{cases} x_1^{(1)} = \beta_1 + \alpha_{11}x_1^{(0)} + \alpha_{12}x_2^{(0)} + \alpha_{13}x_3^{(0)} + ... + \alpha_{1n}x_n^{(0)}, \\ x_2^{(1)} = \beta_2 + \alpha_{21}x_1^{(1)} + \alpha_{22}x_2^{(0)} + \alpha_{23}x_3^{(0)} + ... + \alpha_{2n}x_n^{(0)}, \\ \vdots \\ x_n^{(1)} = \beta_n + \alpha_{n1}x_1^{(1)} + \alpha_{n2}x_2^{(1)} + \alpha_{n3}x_3^{(1)} + ... + \alpha_{n,n-1}x_n^{(1)} + \alpha_{nn}x_n^{(0)}; \end{cases}$$

2) $k + 1$ наближення:

$$\begin{cases} x_1^{(k+1)} = \beta_1 + \alpha_{11}x_1^{(k)} + \alpha_{12}x_2^{(k)} + \alpha_{13}x_3^{(k)} + ... + \alpha_{1n}x_n^{(k)}, \\ x_2^{(k+1)} = \beta_2 + \alpha_{21}x_1^{(k+1)} + \alpha_{22}x_2^{(k)} + ... + \alpha_{2n}x_n^{(k)}, \\ \vdots \\ x_i^{(k+1)} = \beta_i + \alpha_{i1}x_1^{(k+1)} + \alpha_{i2}x_2^{(k+1)} + ... + \alpha_{i,i-1}x_{i-1}^{(k+1)} + \alpha_{ii}x_i^{(k)} + ... + \alpha_{in}x_n^{(k)}, \\ \vdots \\ x_n^{(k+1)} = \beta_n + \alpha_{n1}x_1^{(k+1)} + \alpha_{n2}x_2^{(k+1)} + ... + \alpha_{n,n-1}x_{n-1}^{(k+1)} + \alpha_{nn}x_n^{(k)}; \end{cases}$$

де $k = 0, 1, 2, \ldots, n$.

Ітераційний процес методу Зейделя подібний до методу простих ітерацій. Відмінним є те, що уточнені значення $x_i^{(k+1)}$ одразу ж підставляються в наступні рівняння. Отже, ітераційна формула методу Зейделя:

$$x_i^{(k+1)} = \beta_i + \sum_{j=1}^{i-1} \alpha_{ij} \cdot x_j^{(k+1)} + \sum_{j=i}^{n} \alpha_{ij} \cdot x_j^{(k)} \ .$$

Тобто, метод Зейделя відрізняється від методу простих ітерацій тим, що при обчисленні $x_i^{(k+1)}$ на $(k+1)$-му кроці враховуються значення $x_1^{(k+1)}$, $x_2^{(k+1)}$, ..., $x_{i-1}^{(k+1)}$, обчислені на цьому самому кроці.

З метою економії пам'яті при програмуванні методу Зейделя недоцільно напряму застосовувати подану формулу методу. На відміну від методу простих ітерацій, у методі Зейделя немає необхідності зберігати в пам'яті повністю вектор попередніх наближень розв'язку. Можна застосовувати один вектор, в якому будуть зберігатися останні наближення розв'язків. При цьому для контролю умови завершення ітераційного процесу по кожному з розв'язків можна застосовувати одну й ту ж допоміжну змінну для тимчасового зберігання попереднього наближення чергового розв'язку.

Умова збіжності: метод Зейделя є збіжним, якщо виконується нерівність

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| < |a_{ii}|, \text{ для всіх } i=1, 2, ..., n.$$

і якщо хоча б для одного i ця нерівність строга

$$|a_{kk}| > \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{kj}|; \ k \in 1, n.$$

2.2.6. Розв'язування СЛАР в MATLAB

Програмний пакет MATLAB має потужний пакет для розв'язування СЛАР різними методами.

При реалізації багатьох завдань, пов'язаних з матричної алгеброю, корисними можуть виявитися функції для оцінок основних характеристик:

`det(A)` – обчислення визначника квадратної матриці;

`rank(A)` – обчислення рангу матриці A;

`inv(A)` – обчислення оберненої матриці A-1, та ж сама операція, що й `A^(-1)`;

`A^n` – піднесення квадратної матриці A до степеня n тощо.

Систему рівнянь типу **Ax=b** можна розв'язати засобами MATLAB різними методами. Розглянемо деякі з них.

1. Операція зворотної скісної риски ****, або функція **`mldivide`**. Розв'язок СЛАР виконується командою:

```
>>x=A\b
```

або

```
>>x=mldivide(A,b)
```

Ці команди є аналогічними. Операція \ реалізується наступним чином.

Якщо матриця **A** діагональна, то розв'язок отримується поділом компонентів вектора **b** на діагональні елементи матриці. Якщо матриця квадратна і стрічкова, то застосовується спеціальний алгоритм на основі методу Гауса. Якщо матриця **A** є трикутною або може бути приведена до трикутної матриці перестановкою рядків і стовпців, то система вирішується методом підстановки. Якщо матриця не є трикутною, але є симетричною і має позитивні діагональні

елементи, то MATLAB намагається застосувати розкладання Холецького і розв'язати дві системи з трикутними матрицями.

Якщо матриця хессенбергова, але не розріджена, то застосовується зведення до системи з трикутною матрицею. Якщо матриця квадратна, але перераховані вище методи застосувати не вдалося, то використовується метод **LU**-розкладу.

Якщо **A** – прямокутна матриця, то використовується **QR**-розкладання, що враховує можливий розріджений характер матриці.

Таким чином, операція **** або функція **mldivide** представляють досить складний *розв'язувач* (solver). Операція **** не дозволяє користувачеві керувати процесом розв'язання. Крім того, на численні перевірки при розв'язуванні великих систем витрачається досить багато часу.

2. Метод оберненої матриці. Розв'язок СЛАР **Ax=b** знаходять у вигляді: $\mathbf{x=A^{-1}b}$.

Наприклад:

```
>> A=[1 4 1 3; 0 -1 3 -1; 3 1 0 2; 1 -2 5 1];
b=[1;2;3;4];
>> x=A^(-1)*b
x =
    1.1364
   -0.0455
    0.5909
   -0.1818
```

3. Функція `linsolve` розв'язує СЛАР і дозволяє користувачеві вибрати метод розв'язування, описавши властивості матриці. У найпростішому випадку звернення до функції має вигляд

```
>>x = linsolve(A, b)
```

При такому виклику у разі квадратної матриці використовується LU-розкладання, а в разі прямокутної матриці – QR-розкладу. Для задання властивостей матриці використовується керуюча структура, яку позначимо `opts` (можна використовувати будь-яке припустиме ім'я): `x = linsolve(A, b, opts)`. Структура може мати поля, що описують властивості матриці: `LT` – нижня трикутна матриця, `UT` – верхня трикутна, `UHESS` – хессенбергова, `SYM` – симетрична, `POSDEF` – позитивно визначена, `RECT` – прямокутна, `TRANSA` – функції передається транспонована матриця. Поля можуть приймати значення `true` або `false`. Можна задавати не всі поля. Допустимі поєднання

значень полів керуючої структури можна знайти в довідковій системі MATLAB. Функція не перевіряє зазначені властивості матриці, що прискорює розв'язок, але може призвести до помилок при неправильному зазначенні її властивостей.

Наприклад:

```
A = [1 2 3;
     0 4 5;
     0 0 6];
b = [6; 9; 6];
% задати true для поля UT (а всі інші false)
opts.UT = true;
opts.TRANSA = false;
opts.LT = false;
opts.UHESS  = false;
opts.SYM  = false;
opts.POSDEF  = false;
opts.RECT  = false;
% функція linsolve буде розглядати матрицю A як
верхню трикутну
x = linsolve(A,b, opts)
```

Отримаємо розв'язок:

```
x =
     1
     1
     1
```

4. Розв'язування СЛАР з використанням функції **`rref`** – приведено-ступінчаста форма матриці (reduced row echelonform). *Ступінчаста матриця* – матриця, що має m рядків і n стовпців, у якої перші r діагональних елементів ненульові, $r \le \min(m,n)$, а елементи, що лежать нижче діагоналі, і елементи останніх m-r рядків дорівнюють нулю:

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1r} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2r} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3r} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{rr} & \dots & a_{rn} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Функція `R=rref()` приводить матрицю до ступінчастої форми з використанням виключення Гауса-Жордана з вибором головного елемента по стовпцю.

Для того, щоб розв'язати СЛАР з квадратною неособливою матрицею **A** і вектором правої частини **b**, необхідно сформувати розширену матрицю, об'єднавши **A** і **b**, і використовуючи функцію `rref`, привести розширену матрицю до ступінчастого вигляду. Останній стовпець отриманої матриці є розв'язком системи.

Наприклад:

```
>> A = [1 -2 1; 2 -5 -1; -7 0 1];
>> B = [2; -1; -2];
>> R = rref([A b])
R =
1.0000       0       0 0.5200
     0 1.0000       0 0.0800
     0       0 1.0000 1.6400
```

5. Розв'язування СЛАР методом ***LU-розкладу*** або ***QR-розкладу.***

Приклад розв'язування СЛАР методом ***LU-розкладу:***

```
>> A=[1 4 1 3; 0 -1 3 -1; 3 1 0 2; 1 -2 5 1];
>> b=[1;2;3;4];
>> [L,U]=lu(A)
L =
    0.3333    1.0000         0         0
         0   -0.2727    0.5806    1.0000
    1.0000         0         0         0
    0.3333   -0.6364    1.0000         0
U =
    3.0000    1.0000         0    2.0000
```

```
         0    3.6667    1.0000    2.3333
         0         0    5.6364    1.8182
         0         0         0   -1.4194
>> y=L\b
y =
    3.0000
         0
    3.0000
    0.2581

>> x=U\y
x =
    1.1364
   -0.0455
    0.5909
   -0.1818
```

Перевіримо правильність отриманого розв'язку:

```
>> b-A*x
ans =
  1.0e-015 *

    0.1110
   -0.4441
         0
    0.4441
```

Приклад розв'язування СЛАР методом ***QR-розкладу:***

```
>> [Q,R]=qr(A);   y=Q'*b;   x3=R\y
x3 =
    1.1364
   -0.0455
    0.5909
   -0.1818
```

6. Програмна реалізація одного з методів розв'язування СЛАР. Розглянемо програмну реалізацію методу Гауса в Matlab. **Створимо m-файл:**

```
function [x,flag]=gauss(A,b)
```

```
% реалізує метод Гауса з вибором головного
елемента по стовпцю,
% використовується функція backsub розв'язку СЛАР
з трикутною матрицею
% A - матриця nxn
% b - вектор правої частини
% x - вектор розв'язку
% flag – ознака виродження матриці A
% (flag=1 - матриця не вироджена, flag=0 - матриця
вироджена)
% Ініціалізація вектора розв'язку x і тимчасового
вектора C
[N, N]=size(A);
x=zeros(N, 1);
C=zeros(1, N+1); % тимчасовий вектор-рядок
B=[A b]; % розширена матриця
flag=1;
epsilon=1e-15; % величина для визначення
виродження матриці A
% Прямий хід
for s=1:N % s - номер кроку
% Вибір головного елемента для неперетвореної
частини стовпця p
% Y - максимальний елемент одновимірного масиву
abs(B(s:N,s))
% j - номер максимального елемента в цьому масиві
[Y,j]=max(abs(B(s:N,s)));
% Міняємо місцями рядки s та j+s-1
C=B(s,:); % рядок s
B(s,:)=B(j+s-1,:);
B(j+s-1,:)=C;
if abs(B(s,s))<=epsilon
disp('Матриця A вироджена');
flag=0; % Ознака виродження матриці
break % Вихід із циклу
end
% Процес виключення для стовпця s
B(s,s:N+1)=B(s,s:N+1)/B(s,s);
for k=s+1:N
m=B(k,s)/B(s,s);
```

```
B(k,s:N+1)=B(k,s:N+1)-m*B(s,s:N+1);
end
end
if flag % Для невиродженої матриці
% Зворотний хід
x=backsub(B(1:N,1:N),B(1:N,N+1));
end
```

Програма викликає функцію **backsub** для розв'язку СЛАР з верхньою трикутною матрицею, яка також має бути записана окремим m-файлом:

```
  function x=backsub(A,b) % Функція розв'язку СЛАР
з верхньою трикутною матрицею:
   % Вхід A - верхня трикутна матриця розміром nxn
  % b - вектор розміром nx1
  % Вихід x – розв'язок системи рівнянь
  n=length(b); % розмір вектора
  x=zeros(n,1); % формування масиву нулів nx1
  % Обчислення розв'язку зворотною підстановкою
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
```

Приклад розв'язування СЛАР методом Гауса ***з вибором головного елемента*** по стовпцях (використовуючи створену) функцію gauss().

```
clear;
clc;
% Приклад матриці і правої частини
disp('Матриця розв''язуваної системи')
A=[4 8 4 0
1 5 4 -3
1 4 7 2
1 3 0 -2]
disp('Вектор правої частини')
b=[ 8
-4
10
4]
```

```
[x,fl]=gauss(A,b);% Виклик методу Гауса
if fl
disp('Розв''язок')
x
disp('Нев''язка');
r=b-A*x
end
```

Приклад розв'язання системи з трьох лінійних рівнянь в MATLAB ***методом простих ітерацій***.

```
  clear, clc
  A = [0 -0.06 0.02;
      -0.03 0 0.05;
      -0.01 -0.02 0];
  B = [2; 3; 5];
  % приводимо до нормального вигляду:
  AB = [A B]; % розширена матриця
  AB = AB([3 1 2],:) % переставляємо рядки
  for i = 1:size(A)
      a(i,1:3) = -AB(i,1:3)/AB(i,i);
      a(i,i) = 0;
      b(i,1) = AB(i,4)/AB(i,i);
  end
  a
  b
  eps = 1e-6; % точність чисельного розв'язку
  X0 = b; % початкове наближення
  X1 = a*X0+b;
  iter=1;
  while norm(X1-X0)>eps
      X0 = X1;
      X1 = a*X0+b;
      iter=iter+1;
  end;
disp('Розв''язок')
X = X1
disp('Кількість ітерацій: ')
iter
```

Отримаємо:

```
  AB =
```

```
   -0.0100   -0.0200         0    5.0000
         0   -0.0600    0.0200    2.0000
   -0.0300         0    0.0500    3.0000
a =
         0   -2.0000         0
         0         0    0.3333
    0.6000         0         0
b =
 -500.0000
  -33.3333
   60.0000
Розв'язок
X =
 -338.0952
  -80.9524
 -142.8571
Кількість ітерацій:
iter =
    65
```

Нев'язка:

```
>> B-A*X
ans =
  1.0e-008 *
    0.3518
    0.5278
    0.8796
```

Отже, розв'язок системи знайдено правильно із заданою точністю.

2.3. Програма роботи

1. Ознайомитися з методами наближеного розв'язування СЛАР.
2. Розв'язати задану СЛАР точними та наближеними чисельними методами.
3. Порівняти розв'язки СЛАР, що отримані різними методами. Оформити звіт про виконання роботи.

2.4. Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями.

2. Вибрати завдання згідно варіанту (за номером у журналі підгрупи або за вказівкою викладача).

3. Завантажити середовище MATLAB. Розв'язати задану систему рівнянь методом зворотної скісної риски (або за допомогою функції `mldivide`().

4. Розв'язати задану систему за допомогою функції `linsolve()`, методом LU-розкладання та QR-розкладання.

5. Знайти розв'язок заданої СЛАР з точністю 10^{-6} методом простих ітерацій та методом Зейделя в Matlab. Для цього спочатку забезпечити виконання умови збіжності ітераційного методу. Розробити блок-схему алгоритму та написати програму, яка забезпечить розв'язок СЛАР в Matlab та виведення на екран результатів роботи. Порівняти кількість ітерацій та нев'язки для кожного з методів.

6. Порівняти отримані розв'язки СЛАР.

7. Зробити висновки про швидкість збіжності кожного з методів. Оформити звіт про виконання роботи.

Звіт повинен містити:

- титульний лист;
- тему, мету роботи, варіант завдання;
- порядок виконання роботи;
- результати всіх етапів обчислень;
- блок-схему алгоритму та текст програми для розв'язання СЛАР методом Зейделя;
- висновки.

2.5. Варіанти завдань

Варіант	Система рівнянь
1	$\begin{cases} -2x_1 + 4x_2 - 5x_3 + 3{,}1x_4 = 0{,}35, \\ x_1 + 5x_2 - 3x_3 + 2{,}3x_4 = 1, \\ -10x_1 + x_2 + 3x_3 - x_4 = -7, \\ 0{,}5x_1 - 2x_2 + 4x_3 - 2{,}3x_4 = -11. \end{cases}$
2	$\begin{cases} 5x_1 + 3x_2 - 2x_3 + x_4 = 14, \\ 7x_1 - 13x_2 + 3x_3 + x_4 = -4, \\ x_1 - 2x_2 + 15x_3 - 7x_4 = 14, \\ 3x_1 - 8x_2 + 2x_3 + 14x_4 = 22. \end{cases}$

3	$\begin{cases} 10x_1 - 5x_2 + 1x_3 + 5x_4 = 21, \\ 2x_1 + 30x_2 - 21x_3 + 5x_4 = 6, \\ 2x_1 - 8x_2 + 23x_3 + 12x_4 = 5, \\ 2x_1 - 8x_2 - 2x_3 - 14x_4 = 6. \end{cases}$
4	$\begin{cases} 34x_1 - 3x_2 + x_3 + 4x_4 = 5, \\ x_1 - 2x_2 + 5x_3 + 4x_4 = 6, \\ 2x_1 - x_2 + 20x_3 - x_4 = -7, \\ 4x_1 + 5x_2 - 2x_3 + 2x_4 = 10. \end{cases}$
5	$\begin{cases} 5x_1 - 5x_2 - 5x_3 - 2x_4 = 8, \\ 2x_1 - 2x_2 + 5x_3 + 4x_4 = -7, \\ 2x_1 - 7x_2 + 10x_3 - x_4 = 7, \\ 4x_1 + 5x_2 - 2x_3 + x_4 = -1. \end{cases}$
6	$\begin{cases} 7x_1 - 13x_2 - 7x_3 - 2x_4 = -20, \\ 3x_1 + 4x_2 - x_3 + x_4 = -11, \\ 6x_1 - 2x_2 + 2x_3 - x_4 = 10, \\ 2x_1 + 4x_2 - 2x_3 + 10x_4 = 4. \end{cases}$
7	$\begin{cases} 6x_1 + 4x_2 + 2x_3 - x_4 = 5, \\ 4x_1 + 12x_2 + 2x_3 + x_4 = 10, \\ x_1 + x_2 + 4x_3 + 2x_4 = 7, \\ 3x_1 + 3x_2 - 2x_3 + 14x_4 = 15. \end{cases}$
8	$\begin{cases} -20x_1 - 4x_2 + 5x_3 - 31x_4 = -35, \\ 5x_1 + 2x_2 + 3x_3 - 23x_4 = 1, \\ x_1 - 10x_2 + 9x_3 + x_4 = 7, \\ x_1 + 3x_2 - 10x_3 + 3x_4 = 11. \end{cases}$
9	$\begin{cases} 3x_1 + x_2 + x_3 - x_4 = 7, \\ x_1 + 5x_2 + 3x_3 - x_4 = 9, \\ x_1 - 2x_2 + 5x_3 - 2x_4 = 3, \\ 3x_1 - x_2 + 2x_3 + 7x_4 = 14. \end{cases}$
10	$\begin{cases} 18x_1 + 5x_2 - 10x_3 + 7x_4 = 14, \\ x_1 + 5x_2 + 3x_3 - 12x_4 = 15, \\ 3x_1 - 2x_2 + 2x_3 - 9x_4 = 19, \\ x_1 - 2x_2 - 15x_3 + 9x_4 = -13. \end{cases}$

11	$\begin{cases} 6x_1+5x_2+3x_3=5, \\ 2x_1-7x_2+3x_3+x_4=20, \\ 5x_1+2x_2+13x_3-x_4=18, \\ 12x_1+18x_2-22x_3+41x_4=7. \end{cases}$
12	$\begin{cases} 8x_1+5x_2-10x_3+8x_4=14, \\ x_1+5x_2+3x_3-12x_4=15, \\ 3x_1-2x_2+5x_3+9x_4=-12, \\ 10x_1-2x_2-15x_3+29x_4=-13. \end{cases}$
13	$\begin{cases} 5x_1+2x_2+3x_3-3x_4=-36, \\ x_1-4x_2+6x_3+24x_4=-27, \\ 3x_1+x_2+7x_3+9x_4=18, \\ 4x_1-x_2-2x_3+6x_4=27. \end{cases}$
14	$\begin{cases} 16x_1+4x_2+2x_3-5x_4=17, \\ 7x_1+13x_2+3x_3+x_4=24, \\ x_1+2x_2+15x_3+2x_4=20, \\ 3x_1-8x_2+2x_3+14x_4=11. \end{cases}$
15	$\begin{cases} 22x_1+5x_2+x_3+x_4=18, \\ 2x_1-30x_2+13x_3+18x_4=-8, \\ 10x_1+2x_2-15x_3-2x_4=-10, \\ 12x_1+18x_2+2x_3-40x_4=-14. \end{cases}$

2.6. Контрольні запитання

1. Сформулюйте необхідну і достатню умову існування єдиного розв’язку СЛАР.

2. Опишіть порядок знаходження розв'язку системи лінійних алгебраїчних рівнянь методом простої ітерації.

3. Яка умова збіжності ітераційного процесу?

4. Опишіть порядок знаходження розв'язку СЛАР методом Зейделя.

5. У чому полягає відмінність методів Зейделя та простих ітерацій? Який з цих методів забезпечує більш точні результати і чому?

6. У чому полягає метод *LU*-розкладу?

7. У чому полягає метод оберненої матриці для розрахунку СЛАР?

Лабораторна робота №3
Чисельне розв'язування систем нелінійних алгебраїчних рівнянь

3.1. Мета роботи

Ознайомитися з чисельними методами розв’язування систем лінійних алгебраїчних рівнянь. Навчитися знаходити розв'язки СЛАР точними та наближеними методами.

3.2. Теоретичні відомості

3.2.1. Загальні відомості

Нехай необхідно знайти розв'язок системи нелінійних рівнянь (СНР):

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0, \\ f_2(x_1, x_2, ..., x_n) = 0, \\ \vdots \\ f_n(x_1, x_2, ..., x_n) = 0; \end{cases} \tag{3.1}$$

де f_1, f_2, ..., f_n – задані нелінійні функції від n невідомих; x_1, x_2, ..., x_n – невідомі.

Введемо n-вимірні вектори:

$$\bar{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix}, \quad F\left(\bar{x}\right) = \begin{pmatrix} f_1(\bar{x}) \\ f_2(\bar{x}) \\ \cdots \\ f_n(\bar{x}) \end{pmatrix}, \quad \bar{0} = \begin{pmatrix} 0 \\ 0 \\ \cdots \\ 0 \end{pmatrix}.$$

Тоді систему (3.1) можна записати у вигляді:

$$F\left(\bar{x}\right) = \left(\bar{0}\right). \tag{3.2}$$

3.2.2. Метод Ньютона

Введемо матрицю Якобі (якобіан) для системи рівнянь (3.1):

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}; \frac{\partial f_1}{\partial x_2}; \ldots; \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1}; \frac{\partial f_2}{\partial x_2}; \ldots; \frac{\partial f_2}{\partial x_n} \\ \vdots \\ \frac{\partial f_n}{\partial x_1}; \frac{\partial f_n}{\partial x_2}; \ldots; \frac{\partial f_n}{\partial x_n} \end{pmatrix}.$$

Тоді ітераційний процес методу Ньютона для розв'язання системи (3.1) буде:

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} - \left(J(\bar{x}^{(k)}) \right)^{-1} \cdot F\left(\bar{x}^{(k)} \right), \qquad (3.3)$$

де k=0, 1, 2, ...

У певному невеликому околі розв'язку $\bar{x}^*$ ітераційний процес класичного методу Ньютонає (3.3) є збіжним, якщо існує обернена матриця $\left(J(\bar{x}^{(k)}) \right)^{-1}$, тобто

$$\det J(\bar{x}^{(k)}) \neq 0 .$$

Якщо початкове наближення вибране достатньо близько до розв'язку СНР, то ітераційний процес (3.3) збігається з квадратичною швидкістю.

Недоліком методу Ньютона є досить велика трудомісткість, оскільки на кожному кроці ітераційного процесу потрібно знайти обернену матрицю Якобі.

3.2.3. Метод простих ітерацій

Систему рівнянь (3.1) зведемо до еквівалентного вигляду:

$$\bar{x} = \Phi\left(\bar{x} \right), \qquad (3.4)$$

де $\Phi\left(\bar{x} \right) = \begin{pmatrix} \phi_1(\bar{x}) \\ \phi_2(\bar{x}) \\ \ldots \\ \phi_n(\bar{x}) \end{pmatrix} = \begin{pmatrix} \phi_1(x_1, x_2, ..., x_n) \\ \phi_2(x_1, x_2, ..., x_n) \\ \ldots \\ \phi_n(x_1, x_2, ..., x_n) \end{pmatrix}.$

Тоді запишемо формулу методу простих ітерацій:

$$\bar{x}^{(k+1)} = \Phi\left(\bar{x}^{(k)}\right), \quad k=0, 1, 2, \ldots \tag{3.5}$$

Якщо існує границя $\lim\limits_{k\to\infty} \bar{x}^{(k)} = \bar{x}^{*}$, то $\bar{x}^{*}$ є розв'язком системи (3.4).

Сформулюємо умови збіжності методу без доведень та теоретичного обґрунтування.

Нехай вектор-функція $\Phi\left(\bar{x}\right)$ має неперервні частинні похідні в деякому околі Ω розв'язку $\bar{x}^{*}$.

Введемо матрицю **M** з елементами:

$$m_{ij} = \max_{x\in\Omega}\left|\frac{\partial \phi_i}{\partial x_j}\right|.$$

Для того, щоб формула (3.5) методу простих ітерацій була збіжною необхідно, щоб норма матриці **M** була меншою від одиниці. Тобто повинна виконуватись одна з умов:

$$\sum_{i=1}^{n} m_{ij} < 1, \text{ або } \sum_{j=1}^{n} m_{ij} < 1, \text{ або } \sum_{i,j=1}^{n} m_{ij}^2 < 1.$$

У свою чергу, ці нерівності будуть справедливими при виконанні умови:

$$|m_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |m_{ij}|, \qquad i=1, 2, \ldots, n.$$

3.2.4. Розв'язування систем нелінійних рівнянь в MATLAB

Для розв'язання систем нелінійних рівнянь вигляду **F(x)** = 0 в MATLAB призначена функція `fsolve`. Тут **x** – вектор і **F(x)** – функція, яка повертає значення вектора.

Детальна інформація про функцію `fsolve` наведена в довідковій системі MATLAB. Наведемо лише короткий опис.

Синтаксис:

```
x=fsolve(fun, x0)
x=fsolve(fun, x0, options)
[x, fval, exitflag]=fsolve(...)
[x, fval, exitflag, output]=fsolve(...)
[x,fval,exitflag,output jacobian]=fsolve(...)
```

Опис:

`fsolve` знаходить корені (нулі) системи нелінійних рівнянь.

`x = fsolve(fun, x0)` – починає з точки `x0` і намагається розв'язати описані в fun рівняння.

`x = fsolve(fun, x0, options)` – проводить оптимізацію з певними параметрами у структурній опції.

`[x, fval, exitflag] = fsolve(...)` – повертає значення `exitflag`, яке містить опис вихідних умов.

`[x, fval, exitflag, output] = fsolve(...)` – повертає структурний вихід з інформацією про оптимізацію

`[x,fval,exitflag,output,jacobian] = fsolve(...)` – повертає Якобіана від `fun` як розв'язок від `x`.

Приклад виклику функції:

```
x = fsolve(@myfun,x0)  % myfun – функція користувача, що описана в m-файлі
x = fsolve(inline('sin(x.*x)'),x0);
```

Приклад. Нехай необхідно знайти розв'язок системи двох нелінійних рівнянь:

$$\begin{cases} 2x_1 - x_2 = e^{-x_1}, \\ -x_1 + 2x_2 = e^{-x_2}. \end{cases}$$

Створимо М-файл для розрахунку **F(x).**

```
function F = myfun(x)
 F = [2*x(1) - x(2) - exp(-x(1));
    -x(1) + 2*x(2) - exp(-x(2))];
```

За початкове наближення візьмемо точку x0=[-5; -5].

Викличемо підпрограму оптимізації

```
x0 = [-5; -5]; % початкове наближення
options=optimset('Display','iter');        %Опція відображення процесу обчислень
[x,fval] = fsolve(@myfun,x0,options)  % виклик оптимізатора
```

Після виконання 33 ітерацій тримаємо розв'язок

```
x =
    0.5671
    0.5671

fval =
```

```
1.0e-006 *
 -0.4059
 -0.4059
```

Приклад знаходження розв'язку системи двох нелінійних рівнянь методами простих ітерацій та Ньютона в Matlab.

Нехай необхідно знайти розв'язок системи нелінійних рівнянь вигляду:

$$\begin{cases} f_1(x_1, x_2) = 0, \\ f_2(x_1, x_2) = 0. \end{cases}$$

Створимо М-файл `funF.m` із заданим вхідним вектором функцій F:

```
function[F]=funF(x)
F=[f1(x(1),x(2));  f2(x(1),x(2))]; %тут
необхідно задати вирази функцій f1 і f2
```

Наприклад, для системи

$$\begin{cases} x_1^2 + x_2^2 - 4 = 0, \\ x_1 \cdot x_2 - 1 = 0. \end{cases}$$

файл `funF.m` матиме вигляд:

```
function[F]=funF(x)
F=[x(1)^2+x(2)^2-4;  x(1)*x(2)-1];
```

Також створимо окремий файл `funJ.m` для задання матриці Якобі:

```
function[J]=funJ(x)
% тут потрібно задати елементи матриці Якобі,
які обчислюють аналітично
```

J=[$f'_{1x_1}(x_1, x_2)$, $f'_{1x_2}(x_1, x_2)$;

$f'_{2x_1}(x_1, x_2)$, $f'_{2x_2}(x_1, x_2)$].

У нашому випадку файл `funJ.m`

```
function[J]=funJ(x)
J=[2*x(1), 2*x(2); x(2), x(1)] ; % тут потрібно
задати елементи матриці Якобі
```

Тоді створимо М-файл `mpi.m` для реалізації методу простих ітерацій.

```
function [xout, dxout, mout]=mpi(x0, eps)
F=funF(x0); nf=norm(F);  % знаходимо значення і
норму вектора-функції в точці x0
```

```
  J=funJ(x0);  % знаходимо значення якобіана в
точці x0
  alpha=J(2,2)/(-J(1,1)*J(2,2)+J(1,2)*J(2,1));
  beta=-alpha*J(1,2)/J(2,2);
  gamma=-J(2,1)/(J(1,2)*J(2,1)-J(1,1)*J(2,2));
  delta=-gamma*J(1,1)/J(2,1);
  dx=[alpha*F(1)+beta*F(2);
gamma*F(1)+delta*F(2)];
  ndx=norm(dx);
  m=0;  x=x0;
  xout=x';  dxout=dx';  mout=m;
  while max([nf; ndx])>eps  % перевіряємо
виконання критерію закінчення ітерацій
  x=x+dx;   % знаходимо нове значення вектора
розв'язку системи
  F=funF(x);  nf=norm(F);  знаходимо значення і
норму
  dx=[alpha*F(1)+beta*F(2);
gamma*F(1)+delta*F(2)]; % знаходимо чергову
похибку
  ndx=norm(dx);  % знаходимо норму чергової
похибки
  m=m+1;  % збільшуємо кількість ітерацій
  xout=[xout; x'];  dxout=[dxout; dx'];
mout=[mout;m];
  end;
```

Створимо М-файл `kmn.m` для реалізації класичного методу Ньютона.

```
  function[xout, dxout, mout]=kmn(x0, eps)
  F=funF(x0);  nf=norm(F);  % знаходимо значення
функції і норму вектора-функції в точці x0
  J=funJ(x0);  % знаходимо значення якобіана в
точці x0
  delta=[det([F(:)  J(:,2)]); det([J(:,1) F(:)])];
  dx=-delta/det(J); ndx=norm(dx);
  m=0; x=x0;
  xout=x'; dxout=dx';  mout=m;
  while max([nf; ndx])>eps  % перевіряємо
виконання критерію закінчення ітерацій
```

```
    x=x+dx;   % знаходимо нове значення вектора
розв'язку системи
  J=funJ(x);
  F=funF(x); nf=norm(F);
  delta=[det([F(:)  J(:,2)]); det([J(:,1) F(:)])];
  dx=-delta/det(J); ndx=norm(dx);
  m=m+1;
  xout=[xout; x'];  dxout=[dxout; dx'];
mout=[mout;m];
  end;
```

Тоді у робочому середовищі Matlab задамо початкове наближення та точність обчислень:

```
  x0=[2; 0.1];
  eps=0.001;
```

Знайдемо розв'язок заданої системи рівнянь методом простих ітерацій:

```
  [xout, dxout, mout]=mpi(x0, eps);
  plot(mout, xout(:,1), mout, xout(:,2))
  title('x mpi')
  pause
  plot(mout, dxout(:,1), mout, xout(:,2))
  title('dx mpi')
  pause
  out_mpi=[mout, xout, dxout]
```

Отримаємо:

```
  out_mpi =
           0    2.0000    0.1000   -0.0226
0.4011
      1.0000    1.9774    0.5011   -0.0407
0.0066
      2.0000    1.9368    0.5077   -0.0026
0.0085
      3.0000    1.9341    0.5162   -0.0019
0.0009
      4.0000    1.9323    0.5171   -0.0003
0.0004
      5.0000    1.9320    0.5175   -0.0001
0.0001
```

Обчислимо нев'язки:

```
  >> funF([1.932, 0.518])
```

```
ans =
  1.0e-003 *

    0.9480
    0.7760
```

Як бачимо, розв'язок x=[1.932, 0.518] з точністю до 0,001 знайдено за 5 ітерацій.

Знайдемо наближений розв'язок заданої СНР класичним методом Ньютона:

```
[xout, dxout, mout]=kmn(x0, eps);
plot(mout, xout(:,1), mout, xout(:,2))
title('x kmn')
pause
plot(mout, dxout(:,1), mout, xout(:,2))
title('dx kmn')
pause
out_kmn=[mout, xout, dxout]
```

Отримаємо:

```
out_kmn =
          0    2.0000    0.1000   -0.0226
0.4011
     1.0000    1.9774    0.5011   -0.0449
0.0159
     2.0000    1.9326    0.5171   -0.0007
0.0006
     3.0000    1.9319    0.5176   -0.0000
0.0000
```

Як бачимо, наближений розв'язок за класичним методом Ньютона знайдено за 3 ітерації.

3.3. Програма роботи

1. Ознайомитися з методами Ньютона та простих ітерацій для наближеного розв'язування систем нелінійних рівнянь.
2. Знайти початкове наближення розв'язку СНР графічним способом. Розв'язати задану СНР наближеними чисельними методами.
3. Порівняти розв'язки СНР, що отримані різними методами. Оформити звіт про виконання роботи.

3.4. Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями.

2. Вибрати завдання згідно варіанту (за номером у журналі підгрупи або за вказівкою викладача).

3. Завантажити середовище MATLAB. Знайти початкове наближення розв'язку СНР, побудувавши графіки функцій $f_1(x_1, x_2)$ та $f_2(x_1, x_2)$ і знайшовши точку їх перетину.

Слід спочатку звести систему до вигляду

$$\begin{cases} f_1(x_1, x_2) = 0, \\ f_2(x_1, x_2) = 0. \end{cases}$$

Для побудови графіків неявно заданих функцій можна використати команду `ezplot`.

Наприклад:

```
ezplot('x^2+y^2+16*x',[-5, 5]) %графік першого рівняння
hold on
grid on
ezplot('x^2+y^2+16*x',[-5, 5]) %графік другого рівняння
```

4. Створити **М-файл** із заданим вхідним вектором функцій

```
function[F]=funF(x)
F=[f1(x(1),x(2)); f2(x(1),x(2))].
```

5. Обчислити матрицю Якобі та записати її в окремий файл (`funJ.m`).

6. Задати початкове наближення розв'язку системи, знайдене графічним способом. Задати необхідну точність розв'язку СНР.

7. Знайти розв'язок заданої СНР з точністю 10^{-4} методом простих ітерацій та методом Ньютона в Matlab.

8. Розв'язати задану СНР за допомогою функції `fsolve`. Порівняти кількість ітерацій та нев'язки для кожного з методів.

9. Зробити висновки про швидкість збіжності кожного з методів. Оформити звіт про виконання роботи.

Звіт повинен містити:

- титульний лист;
- тему, мету роботи, варіант завдання;
- порядок виконання роботи;
- результати всіх етапів обчислень;

- висновки.

3.5. Варіанти завдань

Варіант	Система рівнянь	Варіант	Система рівнянь
1	$\begin{cases} x^2 + y^2 = 16 \cdot x \\ \cos(x \cdot y) - x = 1 \end{cases}$	16	$\begin{cases} tg(xy + 0.1) = x^2 \\ x^2 + y^2 = 1 \end{cases}$
2	$\begin{cases} \sin y + 2x = 2 \\ \cos(x - 1) + y = 0.7 \end{cases}$	17	$\begin{cases} y(xy + 0.1) = x^2 \\ 0.9x^2 + 2y^2 = 1 \end{cases}$
3	$\begin{cases} \sin(x + y) - 1.2x = 0.2 \\ x^2 + y^2 = 1 \end{cases}$	18	$\begin{cases} tg(xy + 0.1) = x^2 \\ 0.5x^2 + 2y^2 = 1 \end{cases}$
4	$\begin{cases} tg(xy + 0.3) = x^2 \\ 0.9x^2 + 2y^2 = 1 \end{cases}$	19	$\begin{cases} \sin(x + y) = 1.1x - 0.1 \\ x^2 + 2y^2 = 1 \end{cases}$
5	$\begin{cases} \sin(x + y) - 1.3x = 0 \\ x^2 + y^2 = 1 \end{cases}$	20	$\begin{cases} tg(x - y) - xy = 0 \\ x^2 + 2y^2 = 1 \end{cases}$
6	$\begin{cases} tg(xy) = x^2 \\ 0.8x^2 + 2y^2 = 1 \end{cases}$	21	$\begin{cases} \sin(x + y) - 4y = -1 \\ x^2 + y^2 = 3/4 \end{cases}$
7	$\begin{cases} \sin(x + y) - 1.5x = 0.1 \\ x^2 + y^2 = 1 \end{cases}$	22	$\begin{cases} tg(xy + 0.2) = x^2 \\ x^2 + 2y^2 = 1 \end{cases}$
8	$\begin{cases} tg(xy) = x^2 \\ 0.1x^2 + 2y^2 = 1 \end{cases}$	23	$\begin{cases} \sin(x + y) - 1.5x = 0 \\ x^2 + y^2 = 1 \end{cases}$
9	$\begin{cases} \sin(x + y) - 1.2x = 0.1 \\ x^2 + y^2 = 1 \end{cases}$	24	$\begin{cases} tg(xy) = x^2 \\ 0.5x^2 + 2y^2 = 1 \end{cases}$

10	$\begin{cases} tg(xy+0.2)=x^2 \\ 0.6x^2+2y^2=1 \end{cases}$	25	$\begin{cases} \sin(x+y)=1.2x-0.2 \\ x^2+y^2=1 \end{cases}$
11	$\begin{cases} \sin(x+y)-1.5x=-0.1 \\ x^2+y^2=1 \end{cases}$	26	$\begin{cases} tg(xy+0.1)=x^2 \\ 0.7x^2+2y^2=1 \end{cases}$
12	$\begin{cases} tg(xy+0.4)=x^2 \\ 0.8x^2+2y^2=1 \end{cases}$	27	$\begin{cases} \sin(x+y)-1.5x=0.2 \\ x^2+y^2=1 \end{cases}$
13	$\begin{cases} \sin(x+y)=1.2x-0.1 \\ x^2+y^2=1 \end{cases}$	28	$\begin{cases} \sin(x+y)-1.6x=0 \\ x^2+y^2=1, x>0, y>0 \end{cases}$
14	$\begin{cases} 9\cdot tg(x+y)+15=0 \\ \frac{x^2}{25}-\frac{y^2}{9}=1 \end{cases}$	29	$\begin{cases} tg(x+y)+y=3 \\ y-\sin(y+0,3)=6\cdot x \end{cases}$
15	$\begin{cases} \sin(x+y)-1.4x=0 \\ x^2+y^2=1 \end{cases}$	30	$\begin{cases} \sin(y+0.5)-x=1 \\ \cos(x-2)+y=1 \end{cases}$

3.6. Контрольні запитання

1. Сформулюйте постановку задачі розв'язання системи нелінійних рівнянь.
2. Що таке матриця Якобі?
3. Яка умова збіжності методу Ньютона для розв’язування СНР?
4. Запишіть ітераційну формулу методу Ньютона для розв'язування СНР.
5. У чому полягає метод простих ітерацій для розв'язування систем нелінійних рівнянь?
6. Які Ви знаєте методи розв'язування СНР у середовищі Matlab?

Лабораторна робота №4
Інтерполяція таблично заданих функцій

4.1. Мета роботи

Навчитися знаходити наближені значення функцій за допомогою інтерполяції.

4.2. Теоретичні відомості

4.2.1. Загальні відомості

Задача інтерполяції часто виникає тоді, коли задану дискретну функцію потрібно замінити неперервною.

Нехай відомі значення функції $y=f(x)$ в точках $x_0, x_1, x_2, ..., x_n$, причому $a \le x_0 < x_1 < x_2 < ... < x_n \le b$, тобто на відрізку $[a; b]$ задана таблична (сіткова) функція:

x	x_0	x_1	...	x_n
y	y_0	y_1	...	y_n

Необхідно знайти значення функції $f(x)$ в будь-якій точці $x \in [a; b]$. Для цього можна вибрати наближену функцію $F(x)$ так, щоб у точках $x_0, x_1, x_2, ..., x_n$ її значення співпадали із заданими значеннями функції $f(x)$:

$$f(x_i)=F(x_i)=y_i, \quad i=0, 1, 2, ..., n. \tag{4.1}$$

Тоді функцію $F(x)$ називають ***інтерполяційною*** для $f(x)$, а точки $x_0, x_1, x_2, ..., x_n$ – *вузлами інтерполяції.* Цей спосіб наближення називають ***інтерполяцією***. Якщо ж x не належить проміжку інтерполювання $[x_0, x_n]$, то таке наближення називають *екстраполяцією*.

4.2.2. Інтерполяційний поліном Лагранжа

Знайдемо наближену функцію для таблично заданої функції $f(x)$. Для цього побудуємо інтерполяційний многочлен $L_n(x)$ такого вигляду:

$$L_n(x)=l_0(x)+ l_1(x)+...+ l_n(x),$$

де $l_i(x)$ – многочлен, причому

$$l_i(x_j) = \begin{cases} 0, \text{ якщо } i \neq j \\ f_i, \text{ якщо } i = j, \end{cases} \quad i, j=0, 1, 2, ..., n. \qquad (4.2)$$

Многочлени $l_i(x)$ побудуємо таким чином:

$$l_i(x) = c_i(x - x_0)(x - x_1)\cdot\ldots\cdot(x - x_{i-1})(x - x_{i+1})\cdot\ldots\cdot(x - x_n),$$

де c_i – постійний коефіцієнт, значення якого можна знайти з умови (4.2):

$$c_i = \frac{f_i}{(x_i - x_0)(x_i - x_1)\cdot\ldots\cdot(x_i - x_{i-1})(x_i - x_{i+1})\cdot\ldots\cdot(x_i - x_n)}.$$

Тоді

$$L_n(x) = \sum_{i=1}^{n} f_i \frac{(x - x_0)(x - x_1)\cdot\ldots\cdot(x - x_{i-1})(x - x_{i+1})\cdot\ldots\cdot(x - x_n)}{(x_i - x_0)(x_i - x_1)\cdot\ldots\cdot(x_i - x_{i-1})(x_i - x_{i+1})\cdot\ldots\cdot(x_i - x_n)}. \qquad (4.3)$$

Функцію (4.3) називають *інтерполяційним поліномом (многочленом) Лагранжа.*

Інтерполяційний поліном Лагранжа можна записати через коефіцієнти Лагранжа:

$L_n^{(i)}(x) = \frac{(x - x_0)(x - x_1)\cdot\ldots\cdot(x - x_{i-1})(x - x_{i+1})\cdot\ldots\cdot(x - x_n)}{(x_i - x_0)(x_i - x_1)\cdot\ldots\cdot(x_i - x_{i-1})(x_i - x_{i+1})\cdot\ldots\cdot(x_i - x_n)}$, які зручно обчислювати як добутки n множників:

$$L_n^{(i)}(x) = \prod_{j=0}^{n} \frac{x - x_j}{z}, \text{ де } z = \begin{cases} x - x_j, & i = j \\ x_i - x_j, & i \neq j. \end{cases}$$

Тоді $L_n(x) = \sum_{i=0}^{n} L_n^{(i)}(x) f(x_i).$

4.2.3. Інтерполяційні формули Ньютона

Побудова інтерполяційних формул значно спрощується, коли вузли інтерполяції $x_0, x_1, ..., x_n$ є рівновіддаленими, тоді *крок інтерполяції*

$$h = x_{i+1} - x_i = \text{const } (i = 0, 1, ..., n-1).$$

Нехай функцію $y = f(x)$ задано своїми значеннями $y_0 = f(x_0)$, $y_1 = f(x_1)$, ..., $y_n = f(x_n)$ причому $x_1 = x_0 + h$; $x_2 = x_0 + 2h, ...,$ $x_n = x_0 + nh$, де $h = x_1 - x_0 = x_2 - x_1 = ... = x_n - x_{n-1}$.

Скінченною різницею першого порядку називають різницю між значеннями функції у сусідніх вузлах інтерполяції:

$$\Delta f(x_i) = f(x_{i+1}) - f(x_i).$$

Наприклад:

$$\Delta y_0 = y_1 - y_0\,; \quad \Delta y_1 = y_2 - y_1\,; \quad .\Delta y_{n-1} = y_n - y_{n-1}\cdot .$$

Скінченні різниці другого порядку:

$\Delta^2 f(x_i) = \Delta(\Delta f(x_{i+1})) = \Delta f(x_{i+1}) - \Delta f(x_i) = f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)$. Наприклад $\Delta^2 y_0 = \Delta y_1 - \Delta y_0\,; \quad \Delta^2 y_1 = \Delta y_2 - \Delta y_1\,; \quad ...\Delta^2 y_{n-2} = \Delta y_{n-1} - \Delta y_{n-2}$

Скінченні різниці n-го порядку визначають за формулою:

$$\Delta^n f(x_i) = \Delta(\Delta^{n-1} f(x_{i+1})) = \Delta^{n-1} f(x_{i+1}) - \Delta^{n-1} f(x_i). \qquad (4.4)$$

Перша інтерполяційна формула Ньютона для рівновіддалених вузлів інтерполяції

Знайдемо інтерполяційний поліном $P_n(x)$ степеня *n* у вигляді

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + ... + a_n(x - x_0)(x - x_1)\cdot ...\cdot (x - x_{n-1}), \qquad (4.5)$$

де $x_0, x_1, ..., x_n$ – задані значення аргумента *x*, причому $x_i - x_{i-1} = h = \text{const}$ ($i = 1, 2, ..., n$), коефіцієнти $a_0, ..., a_n$ невідомі. Визначимо їх, виходячи з початкових умов $P_n(x_i) = f(x_i)$.

Приймемо у формулі (4.5) $x = x_0$. Тоді $P_n(x_0) = a_0$. За початкових умов $P_n(x_0) = f(x_0)$. Отже, $a_0 = f(x_0)$. Загальний вираз для обчислення коефіцієнтів a_k буде:

$$a_i = \frac{\Delta^i f(x_0)}{i! h^i}.$$

Підставивши відповідні коефіцієнти у (4.5), одержимо:

$$P_n(x) = f(x_0) + \frac{\Delta f(x_0)}{h}(x - x_0) + \frac{\Delta^2 f(x_0)}{2! h}(x - x_0)(x - x_1) +$$

$$+ ... + \frac{\Delta^n f(x_0)}{n! h^n}(x - x_0)(x - x_1)...(x - x_{n-1}). \qquad (4.6)$$

Вираз (4.6) називають *першою інтерполяційною формулою Ньютона.*

Позначимо

$$\frac{x - x_0}{h} = q.$$

Тоді

$$\frac{(x - x_1)}{h} = \frac{x - (x_0 + h)}{h} = q - 1;$$

$$\frac{x-x_2}{h}=q-2;\quad ...,\frac{x-x_{n-1}}{h}=q-n+1.$$

Формула (4.6) набуває вигляду:

$$P_n(x)=f(x_0)+q\Delta f(x_0)+\frac{q(q-1)}{2!}\Delta^2 f(x_0)+\frac{q(q-1)(q-2)}{3!}\Delta^3 f(x_0)+...+$$

$$+\frac{q(q-1)(q-2)...(q-n+1)}{n!}\Delta^n f(x_0). \qquad (4.7)$$

Формулу (4.13) доцільно використовувати для інтерполювання (екстраполювання) функції в околі початкового значення x_0.

Друга інтерполяційна формула Ньютона для рівновіддалених вузлів інтерполяції

Коли значення аргумента знаходиться ближче до кінця відрізка інтерполяції x_n, то використовують другу інтерполяційну формулу Ньютона:

$$P_n(x)=f(x_n)+m\Delta f(x_{n-1})+\frac{m(m+1)}{2!}\Delta^2 f(x_{n-2})+$$

$$+\frac{m(m+1)(m+2)}{3!}\Delta^3 f(x_{n-3})+...+\frac{m(m+1)...(m+n-1)}{n!}\Delta^n f(x_0), \qquad (4.8)$$

де $m=\frac{x-x_n}{h}$.

4.2.4. Сплайн-інтерполяція

Сплайн-інтерполяцію застосовують для представления даних відрізками поліномів невисокого порядку (найчастіше – не вище третього). ***Сплайном*** називають ламану лінію, ланками якої є відрізки кривих, заданих многочленами.

Нехай функція $y=f(x)$ задана таблично або графічно на відрізку $[a, b]$. Розділимо відрізок на n частин $a=x_0<x_1<x_2<...<x_n=b$, значення функції в цих точках: $y_0, y_1, y_2, ..., y_n$. Замінимо функцію $y=f(x)$ на кожному з елементарних відрізків $[x_i, x_{i+1}]$ (i=0, 1,2, ..., n) відрізком прямої:

$$L_i(x)=\frac{x-x_i}{x_{i+1}-x_i}(f(x_{i+1})-f(x_i))+f(x_i). \qquad (4.9)$$

У цьому випадку функція *y=f(x)* на відрізку [*a*, *b*] наближена *лінійним сплайном* $S(x)=\sum_{i=0}^{n} L_i(x)$.

Якщо на елементарних відрізках $[x_i, x_{i+1}]$ функцію *y=f(x)* замінити кубічним поліномом

$$P_i(x)=\sum_{k=0}^{3} a_k(x_i-x_k),\quad i=0,1,2,...,n,$$

тоді функція *y=f(x)* наближена *кубічним сплайном*

$$S_3(x)=\sum_{i=0}^{n} P_i(x_i). \qquad (4.10)$$

Кубічна інтерполяція забезпечує безперервність першої та другої похідних результату інтерполяції у вузлових точках. З цього випливають такі властивості кубічної сплайн-інтерполяції: графік кусково-полиноміальної інтерполюючої функції проходить точно через вузлові точки; у вузлових точках немає розривів і різких перегинів функції; завдяки низькому степені поліномів похибка між вузловими точками зазвичай досить мала; зв'язок між числом вузлових точок і ступенем полінома відсутній.

4.2.5. Інтерполяція функцій засобами MATLAB

MATLAB має широкий спектр засобів для інтерполювання та апроксимації функцій.

Для ***одновимірної табличної інтерполяції*** використовується функція `interp1`:

- `yi=interp1(x,Y,xi)` – повертає вектор `yi`, що містить елементи, які відповідають елементам `xi` й отримані інтерполяцією векторів `x` і `Y`. Вектор `x` визначає точки, в яких задано значення `Y`.
- `yi=interp1(x,Y,xi method)` – дозволяє за допомогою параметра `method` задати метод інтерполяції:

  `'nearest'` – ступінчаста інтерполяція;

  `'linear'` – лінійна інтерполяція (прийнята за замовчуванням);

  `'spline'` – кубічна сплайн-інтерполяція;

  `'cubic'` або `'pchip'` – інтерполяція многочленами Ерміта;

  `'v5cubic'` – кубічна інтерполяція MATLAB 5.

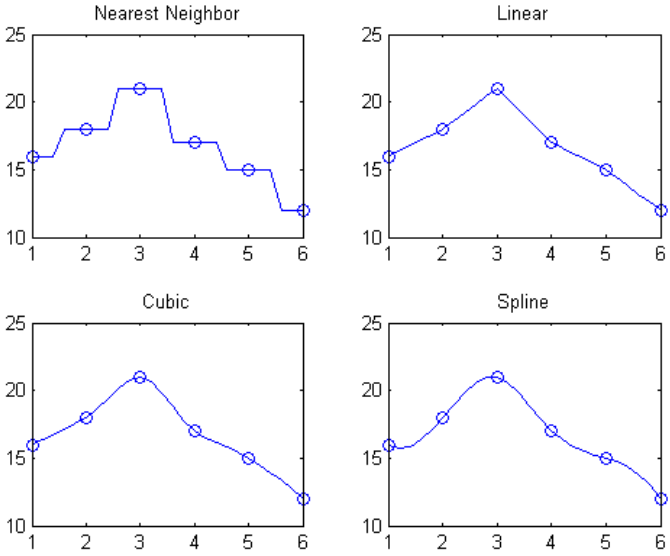
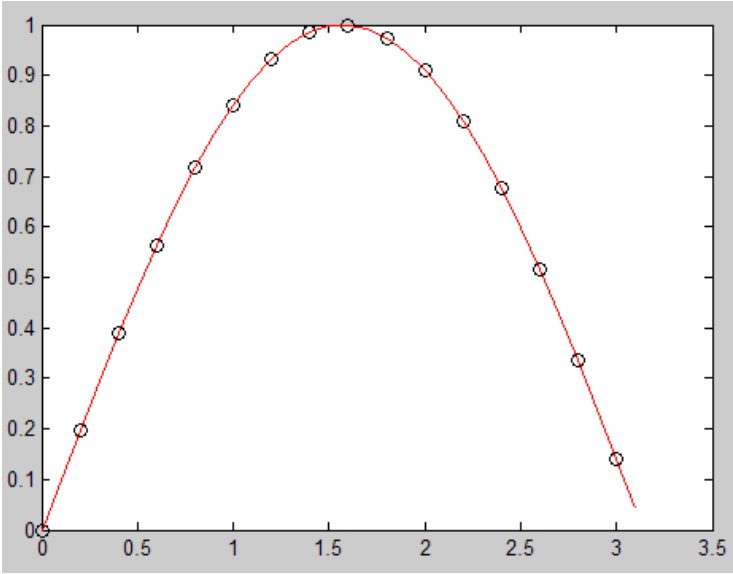


Рис. 4.1. Порівняння різних методів інтерполяції

- `yi=interp1(x,Y,xi,method,'extrap')` – використовує вказаний алгоритм інтерполювання для виконання екстраполяції, тобто наближення значень функції, що лежать за межами вказаного проміжку.
- `pp=interp1(x,Y,method,'pp')` – використовує алгоритм інтерполювання, вказаний у параметрі `method` для створення кусково-поліноміальної форми (ppform) `Y`. Можна використовувати один із вказаних вище методів, крім `'v5cubic'`. `ppval(pp,xi)` – те ж саме, що й `interp1(x,Y,xi,method,'extrap')`.

Наприклад:

```
x = 0:.2:pi; y = sin(x);
pp = interp1(x,y,'spline','pp');
xi = 0:.1:pi;
yi = ppval(pp,xi);
plot(x,y,'ko'),hold on,plot(xi,yi,'r-'),hold off
```



`yi = interp1q(x,Y,xi)` – повертає значення наближеної функції в точках `xi`, використовуючи ***лінійну інтерполяцію***.

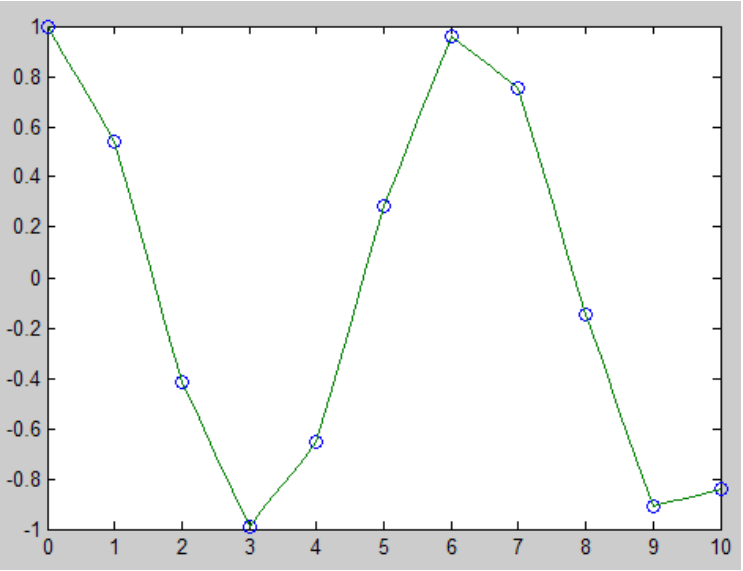
Функція `interp1q` є швидшою, ніж `interp1` при нерівномірно розташованих вузлах інтерполяції, оскільки не проводить перевірки вхідних даних.

Для того, щоб функція `interp1q` працювала коректно, повинні виконуватися умови:

- `x` має бути монотонно зростаючим вектором-стовпцем;
- `Y` має бути вектором стовпцем або матрицею з `length(x)` рядків;
- `xi` має бути вектором-стовпцем.

Наприклад:

```
x = (0:10)';
y = cos(x);
xi = (0:.25:10)';
yi = interp1q(x,y,xi);
plot(x,y,'o',xi,yi)
```



`yy = spline(x,Y,xx)` – виконує інтерполяцію кубічними сплайнами для знаходження значень наближеної функції `yy` в точках `xx`.

`pp = spline(x,Y)` – повертає кусково-поліноміальну форму функції, інтерпольованої кубічними сплайнами, для подальшого використання з функцією `ppval`.

З огляду на важливість сплайн-інтерполяції та апроксимації в обробці і представленні складних даних, до складу системи MATLAB входить пакет розширення Spline Toolbox (викликається командою `splinetool`), що містить близько 70 додаткових функцій, котрі відносяться до реалізації сплайн-інтерполяції та апроксимації, а також графічного представлення сплайнами їх результатів. Для виклику даних про цей використовуйте команду `help splines`.

Для періодичних (особливо для гладких періодичних) функцій хороші результати може дати їхня інтерполяція тригонометричним рядом Фур'є. Для цього використовується наступна функція:
`interpft(x,n)` – повертає вектор `y`, що містить значення періодичної функції, визначені в `n` рівномірно розташованих точках. Якщо `length(x)=m` та `x` має інтервал дискретизації `dx`, то інтервал дискретизації для `y` складає `dy = dx*m/n`, причому `n` не може бути

менше, ніж m. Якщо X – матриця, interpft оперує стовпцями X, повертаючи матрицю Y з таким же числом стовпців, як і в X, але з n рядками. Функція y = interpft(x,n,dim) працює або з рядками, або зі стовпцями залежно від значення параметра dim.

Приклад 1. Проінтерполювати функцію *f(x)=sin(x)*, задану таблично на інтервалі [0, 2π] за допомогою многочлена Лагранжа при n = 8.

1. Створимо М-файл для розрахунку многочлена $l_i(x)$.

```
function z=lagrange(x,i,X,Y)
% x – абсциса точки інтерполяції
% i – номер полінома Лагранжа
% X – вектор вузлів інтерполяції
% Y – вектор значень функції у вузлах інтерполяції
N=length(X);
L=1;
for j=1:N
    if not(j==i)
       L=L*(x-X(j))/(X(i)-X(j));
    end;
end;
z=L*Y(i);
```

2. Створимо файл lagr_pol.m, що повертає значення полінома Лагранжа.

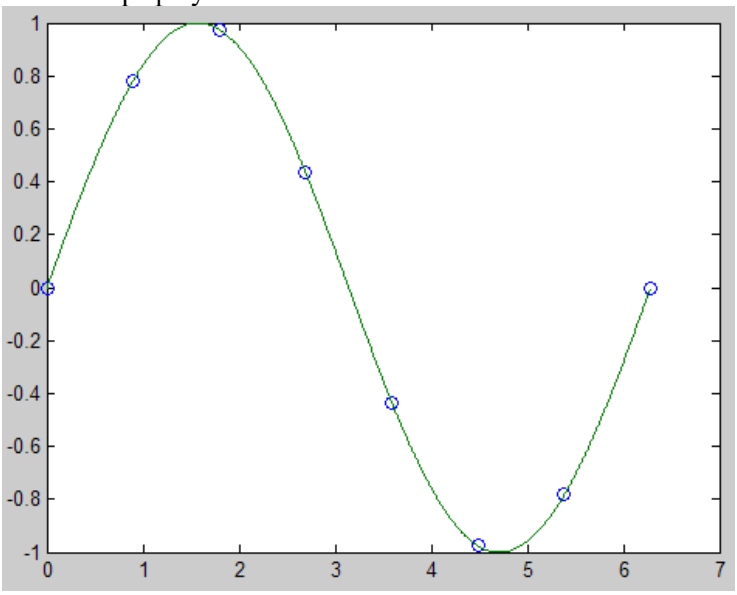
```
function z=lagr_pol(x,X,Y)
N=length(X);
s=0;
for i=1:N
    s=s+lagrange(x,i,X,Y);
end;
z=s;
```

3. Задаємо табличну функцію та обчислюємо значення полінома Лагранжа.

```
>> clear all
>>
>> % Задаємо табличні значення інтерпольованої функції
```

```
N=8;
i=1:N;
x(i)=2*pi/(N-1)*(i-1);
y=sin(x);
% Задаємо число проміжних точок, обчислюємо їхні координати і значення інтерпольованої функції
M=1000;
j=1:M;
X(j)=2*pi/(M-1)*(j-1);
Y=sin(X);
% Обчислюємо значення полінома Лагранжа в проміжних точках
for j=1:M
    Y2(j)=lagr_pol(X(j),x,y);
end;
plot(x,y,'o',X,Y2);
pause
plot(X,Y2-Y);
```

Значення табличної функції та інтерпольовані значення функції представлені на графіку.

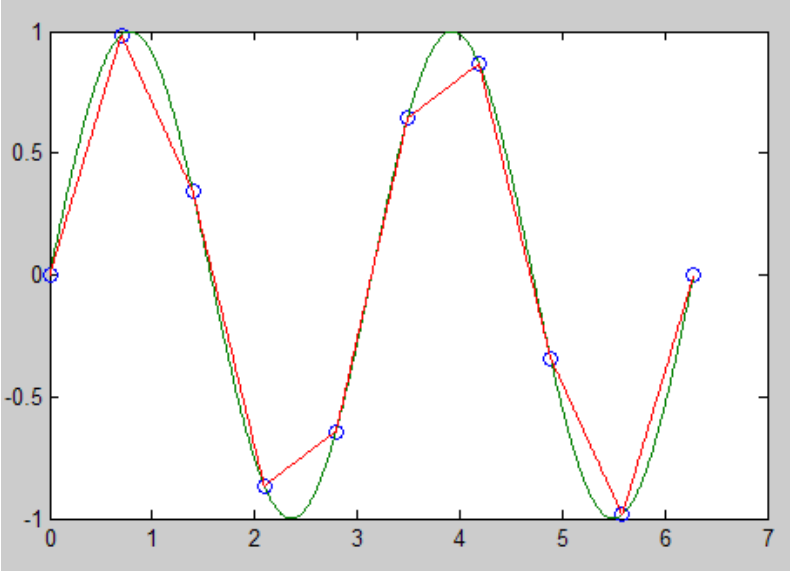


Приклад 2. Проінтерполювати функцію *f(x)=sin(2*x)*, задану таблично на інтервалі [0, 2π] засобами MATLAB при n = 10.

Для проведення лінійної інтерполяції у командному вікні MATLAB введемо:

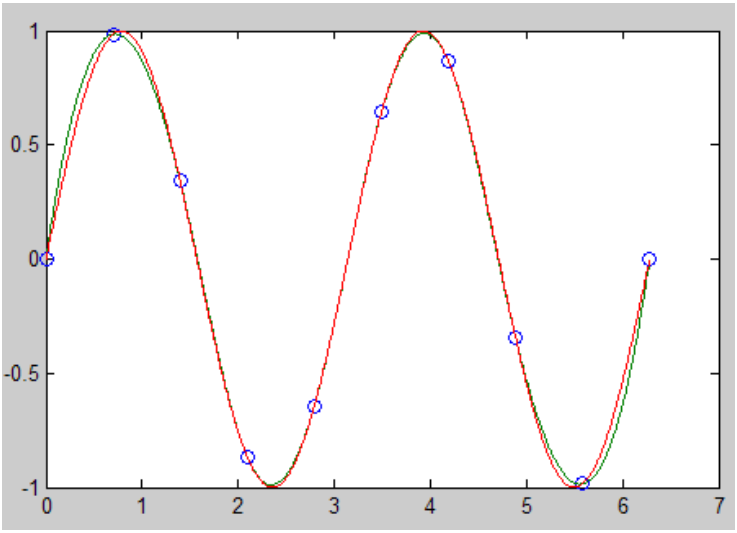
```
clear all
N=10;
i=1:N;
x(i)=2*pi/(N-1)*(i-1);
y=sin(2*x);
% Задаємо число проміжних точок, обчислюємо їхні координати і значення інтерпольованої функції
M=1000;
j=1:M;
X(j)=2*pi/(M-1)*(j-1);
Y=sin(2*X);
Yi=interp1(x,y,X);
plot(x,y,'o',X,Y,X,Yi);
```

Вихідні дані та результати інтерполяції зображені нижче.



Проінтерполюємо задану функцію за допомогою кубічних сплайнів.

```
clear all
N=10;
i=1:N;
x(i)=2*pi/(N-1)*(i-1);
y=sin(2*x);
M=1000;
j=1:M;
X(j)=2*pi/(M-1)*(j-1);
Y=sin(2*X);
yy=spline(x,y,X);
plot(x,y,'o',X,yy,X,Y);
```



4.3. Програма роботи

1. Ознайомитися з методами інтерполяції функцій.
2. Обчислити значення табличної функції в заданій точці за допомогою інтерполяційних формул Ньютона.
3. Розв'язати задачу інтерполяції засобами Matlab за допомогою лінійних та кубічних сплайнів.

4.4. Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями.

2. Вибрати завдання згідно варіанту (за номером у журналі підгрупи або за вказівкою викладача).

3. Завантажити середовище MATLAB. Обчислити значення таблично заданої функції в точках x^*_1 та x^*_2 за допомогою інтерполяційних формул Ньютона.

4. Проінтерполювати таблично задану функцію за допомогою многочлена Лагранжа.

5. Проінтерполювати таблично задану функцію лінійними та кубічними сплайнами, використовуючи функції MATLAB. Побудувати графіки.

6. Провести інтерполяцію функції Рунге $y=1/(1+x^2)$ степеневим поліномом. Побудувати графіки. Визначити похибки інтерполяції.

7. Зробити висновки. Оформити звіт про виконання роботи.

4.5. Варіанти завдань

Варіант	***x***	***y***	**Значення аргумента**	**Варіант**	***x***	***y***	**Значення аргумента**
1	***2***	***3***	***4***	***5***	***6***	***7***	***8***
1	1.50	0.51183	x^*_1 =1.50911 x^*_2 =1.59513	**6**	0.50	1.6487	x^*_1 = 0.50721 x^*_2 =0.56894
	1.51	0.50624			0.51	1.6653	
	1.52	0.50064			0.52	1.6820	
	1.53	0.49503			0.53	1.6989	
	1.54	0.48940			0.54	1.7160	
	1.55	0.48376			0.55	1.7333	
	1.56	0.47811			0.56	1.7507	
	1.57	0.47245			0.57	1.7683	
	1.58	0.46678			0.58	1.7860	
	1.59	0.46110			0.59	1.8040	
	1.60	0.45540			0.60	1.8221	

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
2	0.00	0.28081	x^*_1 = 0.01928 x^*_2 =0.47113	**7**	1.1	0.89121	x^*_1 = 1.1511 x^*_2 =2.0316
	0.05	0.31270			1.2	0.93204	
	0.10	0.34549			1.3	0.96356	
	0.15	0.37904			1.4	0.98545	
	0.20	0.41318			1.5	0.99749	
	0.25	0.44774			1.6	0.99957	
	0.30	0.48255			1.7	0.99166	
	0.35	0.51745			1.8	0.97385	
	0.40	0.55226			1.9	0.94630	
	0.45	0.58682			2.0	0.90930	
	0.50	0.62096			2.1	0.86321	
3	1.0	0.5652	x^*_1 = 1.0113 x^*_2 =1.9592	**8**	0.10	3.63004	x^*_1 =0.1006 x^*_2 = 2.5304
	1.1	0.6375			0.35	3.75680	
	1.2	0.7147			0.60	3.88933	
	1.3	0.7973			0.85	4.03258	
	1.4	0.8861			1.10	4.19310	
	1.5	0.9817			1.35	4.38042	
	1.6	1.0848			1.60	4.60963	
	1.7	1.1964			1.85	4.90697	
	1.8	1.3172			2.10	5.32331	
	1.9	1.4482			2.35	5.97322	
	2.0	1.5906			2.60	7.18210	
4	0.50	1.6487	x^*_1 =0.52301 x^*_2 =0.58967	**9**	1.50	0.51183	x^*_1 = 1.50253 x^*_2 =1.59614
	0.51	1.6653			1.51	0.50624	
	0.52	1.6820			1.52	0.50064	
	0.53	1.6989			1.53	0.49503	
	0.54	1.7160			1.54	0.48940	
	0.55	1.7333			1.55	0.48376	
	0.56	1.7507			1.56	0.47811	
	0.57	1.7683			1.57	0.47245	
	0.58	1.7860			1.58	0.46678	
	0.59	1.8040			1.59	0.46110	
	0.60	1.8221			1.60	0.45540	

5	0.10	3.63004	$x^*_1 = 0.10056$ $x^*_2 = 2.57321$	**10**	0.00	0.28081	$x^*_1 = 0.02475$ $x^*_2 = 0.48675$
	0.35	3.75680			0.05	0.31270	
	0.60	3.88933			0.10	0.34549	
	0.85	4.03258			0.15	0.37904	
	1.10	4.19310			0.20	0.41318	
	1.35	4.38042			0.25	0.44774	
	1.60	4.60963			0.30	0.48255	
	1.85	4.90697			0.35	0.51745	
	2.10	5.32331			0.40	0.55226	
	2.35	5.97322			0.45	0.58682	
	2.60	7.18210			0.50	0.62096	

4.6. Контрольні запитання

1. Що таке інтерполяція?
2. Що називають вузлами інтерполяції?
3. Що таке екстраполяція?
4. Запишіть формулу інтерполяційного полінома Лагранжа.
5. Що таке скінченна різниця?
6. Яка відмінність між інтерполяцією та екстраполяцією функції ?
7. Що таке сплайн-інтерполяція?
8. Яка функція використовується для інтерполяції кубічними сплайнами у MATLAB?

Лабораторна робота №5
Апроксимація експериментальних даних методом найменших квадратів

5.1. Мета роботи

Вивчити основи методу найменших квадратів для побудови апроксимаційних залежностей. Навчитися розв'язувати задачу апроксимації дискретної залежності неперервною функцією. Ознайомитися з методикою застосування програмних продуктів для побудови апроксимаційних функцій на основі експериментальних даних.

5.2. Теоретичні відомості

5.2.1. Загальні відомості

Обробка експериментальних даних відіграє важливу роль в інженерній та науковій діяльності. Результати дослідів та експериментів *(емпіричну залежність між певними величинами)* зазвичай представляють табличним способом.

Перевагою табличного способу завдання експериментальної функції є те, що для кожного табличного значення незалежної змінної можна безпосередньо знайти відповідне значення функції. Недоліком такого способу є те, що таблична функція є дискретною, тобто визначена лише для окремого набору значень незалежної змінної. Тому, для аналізу результатів експериментів зручним у використанні є аналітичний спосіб представлення емпіричних залежностей.

Для того, щоб отримати аналітичні залежності для опису експериментальних даних, використовують ***методи апроксимації***, які основані на заміні табличної функції наближеною простою неперервною функцією на заданому проміжку, яка не обов’язково проходить через всі експериментальні точки, але описує тенденції зміни цих даних.

Нехай в результаті експерименту отримали таблицю:

x	x_0	x_1	...	x_n
y	y_0	y_1	...	y_n

Вважаємо, що між параметрами x та y існує певна функціональна залежність $y=f(x)$. Потрібно знайти наближений аналітичний вираз цієї залежності, тобто підібрати таку наближену функцію $F(x)$, яка б апроксимувала на відрізку $[x_0, x_n]$ задану функцію окремими наближеними значеннями $y_i=f(x_i)$, $i=0, 1, 2, ..., n$.

Отриману залежність $y=F(x)$ називають *емпіричною*, або *рівнянням регресії*. Під *регресією* розуміють підгонку параметрів простої функції для найкращої апроксимації експериментальних даних. Використання емпіричних залежностей має велике практичне значення, оскільки дає змогу не тільки апроксимувати сукупність експериментальних даних, а й екстраполювати знайдену залежність на інші проміжки значень x.

Для того, щоб встановити загальний вигляд емпіричної формули *графічним способом*, будують наближений графік залежності $y=f(x)$. При цьому на основі експериментальних даних на площині зображують точки (x_i, y_i) і будують плавну криву, яку проводять якомога ближче до всіх даних точок. Після цього за формою кривої візуально вибирають вигляд наближеної функції (зазвичай підбирають найпростіші функції: лінійну, квадратичну, дробово-раціональну, степеневу, показникову, логарифмічну).

Застосуємо **метод найменших квадратів** (МНК) для знаходження емпіричної залежності $y=F(x)$. За цим методом функція $F(x)$ вибирається з певного m-параметричного сімейства функцій, а її параметри підбираються так, щоб сума квадратів відхилень обчислених значень $F(x_i)$ від заданих наближених значень y_i була мінімальною.

Задамо сімейство m-параметричних функцій $y=F(x, a_1, a_2,..., a_m)$. Потрібно знайти значення параметрів $a_1, a_2,..., a_m$, розв'язуючи екстремальну задачу:

$$S = \sum_{i=0}^{m} (F(x_i, a_1, a_2, ..., a_m) - y_i)^2 \to \min. \qquad (5.1)$$

Для цього знайдемо частинні похідні функціонала S і прирівняємо їх до нуля.

$$\frac{\partial S}{\partial a_1} = 0, \ \frac{\partial S}{\partial a_2} = 0, \ \ldots, \ \frac{\partial S}{\partial a_m} = 0.$$

Отримаємо систему рівнянь:

$$\begin{cases} \sum_{i=1}^{n}\left(y_i - F\left(x_i; a_1, a_2, \ldots, a_m\right)\right)\cdot\dfrac{\partial F\left(x_i; a_1, \ldots, a_m\right)}{\partial a_1} = 0, \\ \sum_{i=1}^{n}\left(y_i - F\left(x_i; a_1, \ldots, a_m\right)\right)\cdot\dfrac{\partial F\left(x_i; a_1, \ldots, a_m\right)}{\partial a_2} = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sum_{i=1}^{n}\left(y_i - F\left(x_i; a_1, \ldots, a_m\right)\right)\cdot\dfrac{\partial F\left(x_i; a_1, \ldots, a_m\right)}{\partial a_m} = 0. \end{cases} \quad (5.2)$$

Систему (5.2) називають *нормальною*. Якщо емпірична функція *F(x)* лінійна відносно параметрів $a_1, a_2, \ldots, a_m$, то нормальна система (5.2) буде системою з m лінійних рівнянь відносно цих параметрів.

Похибка апроксимації:

$$\delta = \sqrt{\sum_{i=0}^{m}(F(x_i, a_1, a_2, ..., a_m) - y_i)^2}\ .$$

Апроксимація експериментальних даних лінійною залежністю. Нехай між даними (x_i, y_i) $(i = 1, 2, \ldots, n)$ існує лінійна залежність. Знайдемо емпіричну формулу у вигляді

$$y = ax + b\ , \quad (5.3)$$

де коефіцієнти a і b невідомі.

Необхідно знайти параметри a і b, за яких буде забезпечено мінімум функції

$$S(a, b) = \sum_{i=1}^{n}\left(y_i - ax_i - b\right)^2\ .$$

Для цього знайдемо частинні похідні функції $S(a, b)$ і прирівняємо їх до нуля:

$$\begin{cases} \dfrac{\partial S}{\partial a} = 2\sum_{i=1}^{n}\left(y_i - ax_i - b\right)\left(-x_i\right) = 0, \\ \dfrac{\partial S}{\partial b} = 2\sum_{i=1}^{n}\left(y_i - ax_i - b\right)\left(-1\right) = 0. \end{cases}$$

Оскільки $\sum_{i=1}^{n} b = nb$, маємо

$$\begin{cases} a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i, \\ a\sum_{i=1}^{n} x_i + nb = \sum_{i=1}^{n} y_i. \end{cases} \tag{5.4}$$

Розв'язавши відносно a і b останню систему, знайдемо параметри апроксимаційної функції

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}, \tag{5.5}$$

$$b = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} x_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}. \tag{5.6}$$

Поліноміальна апроксимація. Розглянемо випадок, коли за емпіричну функцію вибирають многочлен

$$F(x)= a_0 + a_1x + a_2x^2 + \ldots + a_mx^m. \tag{5.7}$$

Тоді з (5.1) формула визначення суми квадратів відхилень:

$$S = \sum_{i=1}^{n} (a_0 + a_1x_i + a_2x_i^{\,2} + \ldots + a_m x_i^{\,m} - y_i)^2 . \tag{5.8}$$

Запишемо систему рівнянь для визначення параметрів a_0, a_1, … , a_m:

$$\begin{cases} \dfrac{\partial \mathrm{S}}{\partial a_0} = 2\sum_{i=1}^{n} \left(a_0 + a_1x_i + a_2x_i^{\,2} + \ldots + a_m x_i^{\,m} - y_i\right) = 0, \\ \dfrac{\partial \mathrm{S}}{\partial a_1} = 2\sum_{i=1}^{n} \left(a_0 + a_1x_i + a_2x_i^{\,2} + \ldots + a_m x_i^{\,m} - y_i\right) x_i = 0, \\ \qquad \ldots \qquad\qquad \ldots \\ \dfrac{\partial \mathrm{S}}{\partial a_m} = 2\sum_{i=1}^{n} \left(a_0 + a_1x_i + a_2x_i^{\,2} + \ldots + a_m x_i^{\,m} - y_i\right) x_i^{\,m} = 0. \end{cases} \tag{5.9}$$

Згрупувавши коефіцієнти при невідомих параметрах a_0, a_1, … , a_m, отримаємо наступну систему рівнянь (попередньо поділимо кожне рівняння системи на 2):

$$\begin{cases} na_0 + a_1\sum_{i=1}^{n} x_i + a_2\sum_{i=1}^{n} x_i^2 + \ldots + a_m\sum_{i=1}^{n} x_i^m = \sum_{i=1}^{n} y_i, \\ a_0\sum_{i=1}^{n} x_i + a_1\sum_{i=1}^{n} x_i{}^2 + a_2\sum_{i=1}^{n} x_i^3 + \ldots + a_m\sum_{i=1}^{n} x_i^{m+1} = \sum_{i=1}^{n} y_i x_i, \\ \quad\ldots \qquad\qquad \ldots \\ a_0\sum_{i=1}^{n} x_i{}^m + a_1\sum_{i=1}^{n} x_i^{m+1} + a_2\sum_{i=1}^{n} x_i^{m+2} + \ldots + a_m\sum_{i=1}^{n} x_i^{2m} = \sum_{i=1}^{n} x_i{}^m y_i. \end{cases} \quad (5.10)$$

Многочлен (5.7) степеня $m < n$, де n – число вузлів апроксимації забезпечує наближення таблично заданої функції $y_i(x_i)$ з мінімальною середньоквадратичною похибкою:

$$e = \sqrt{\frac{\left(\sum_{i=1}^{n} \varepsilon_i{}^2\right)}{(n+1)}}\,. \quad (5.11)$$

Якщо $m = n$, тоді многочлен (5.7) є інтерполяційним, тобто

$$F(x_i) = y_i\,.$$

Часто застосовують поліноміальну апроксимацію за МНК з автоматичним вибором степеня полінома за таким алгоритмом:

1) задається початкове значення степеня полінома m;

2) обчислюються коефіцієнти полінома $a_0, a_1, \ldots, a_m$;

3) за формулою (5.11) обчислюється середньоквадратична похибка і порівнюється із заданою e_0. Якщо $e>e_0$, то степінь m збільшується на 1 і цикл повторюється. Обчислення завершують при $e < e_0$.

Подібним чином знаходять параметри інших апроксимаційних залежностей. Зазвичай використовують одну з відомих функцій:

$y = ax + b$ – лінійна;

$y = ax^b$ – степенева;

$y = ab^x$ – показникова;

$y = a\ln x + b$ – логарифмічна;

$y = \frac{a}{x} + b$ – гіперболічна тощо.

5.2.2. Апроксимація функцій засобами MATLAB

У MATLAB є ряд додатків, які надають широкі можливості для наближення (апроксимації) одновимірних і багатовимірних даних. Ці

можливості реалізовані на різних рівнях: від досить простих засобів графічного вікна для наближення вже візуалізованих даних, до спеціальних функцій MATLAB і його пакетів, включаючи середовища з графічним інтерфейсом користувача, що дозволяють імпортувати дані, проводити їх попередню обробку і згладжування, апроксимувати дані різними методами. До складу MATLAB входять функції для вирішення деяких задач обчислювальної геометрії. Функції MATLAB дозволяють вирішувати такі завдання: наближення поліномами за методом найменших квадратів; інтерполяція одновимірних даних сплайнами; наближення згладжуючими сплайнами; наближення сплайнами методом найменших квадратів; інтерполяція і наближення двовимірних і багатовимірних даних сплайнами; наближення раціональними сплайнами; побудова опуклої оболонки двовимірних і багатовимірних даних; знаходження найближчої точки; наближення розкиданих даних; побудова лінійних і нелінійних параметричних моделей для наближення даних (розв'язок задачі про підбір параметрів в одній зі стандартних моделей або заданої користувачем і оцінка якості наближення).

Для ***поліноміальної апроксимації*** використовується функція `polyfit`:

`p=polyfit(x,y,n)` – функція обчислює коефіцієнти полінома `p(x)` степеня *n* для найкращого наближення функції `y` методом найменших квадратів. Вектор-рядок `p` містить `n+1` коефіцієнтів апроксимаційного полінома у порядку спадання степенів: `p(1)*x^n + p(2)*x^(n-1) +...+ p(n)*x + p(n+1)`.

`y=polyval(p, x)` – обчислення значення полінома `p` в т. `x`.

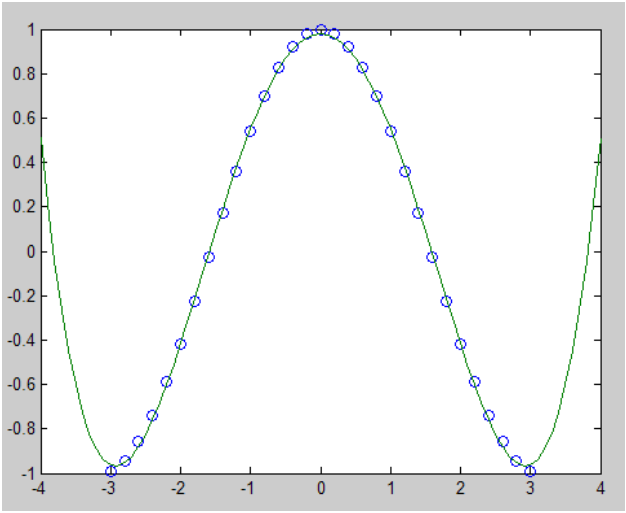
Наприклад:

```
x=(-3:0.2:3); % задаємо вузли апроксимації
y=cos(x);
% проводимо апроксимацію табличної функції поліномом 4-го степеня
p=polyfit(x,y,4)
```

Отримаємо коефіцієнти полінома:

```
p =
0.0266   -0.0000   -0.4554    0.0000    0.9805
>> x1=(-4:0.1:4);
% обчислюємо значення полінома на ширшому інтервалі
```

```
  f=polyval(p,x1);
  plot(x,y,'o',x1,f)  % будуємо графік табличної функції та апроксимаційного полінома
```



При степені полінома вище 5 похибка поліноміальної регресії (і апроксимації) сильно зростає і її застосування без центрування і масштабування стає ризикованим.

Для розв'язання задачі узагальненої нелінійної регресії методом найменших квадратів у пакеті MATLAB є функція `lsqnonlin()`.

Синтаксис функції:

```
x = lsqnonlin(fun,x0)
x = lsqnonlin(fun,x0,lb,ub)
x = lsqnonlin(fun,x0,lb,ub,options)
```

Розглянемо приклад застосування цієї функції для визначення параметрів апроксимаційної функції вигляду:

$$F(x,a,b,c) = e^{a+bx+cx^2}.$$

Для цього виконаємо наступні дії.

У MATLAB створимо m-файл **Apr.m**

```
function z=Apr(c,x,y)
  z=y-exp(c(1)+c(2)*x+c(3)*x.^2);
end
```

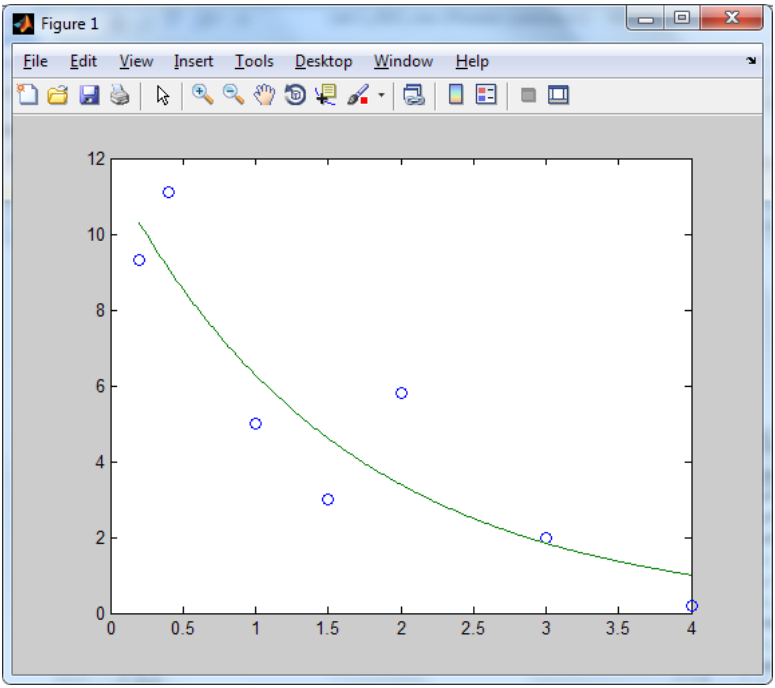
У командному вікні Matlab запишемо:

```
% задаємо початкові дані
x=[0.2;0.4;1;1.5;2;3;4]
y=[9.3;11.1;5;3;5.8;2;0.2]
z=[1 0 -1] % початкове наближення
%обчислення коефіцієнтів апроксимаційної функції
c = lsqnonlin('Apr',z',[],[],[],x,y)

% задаємо апроксимаційну функцію
f=inline('exp(a+b*x+c*x.^2)','x','a','b','c');

% формування вектора координат, в яких будуть
обчислюватися значення апроксимаційної функції
X=x(1):0.01:x(end);

% обчислення значень апроксимаційної функції
Y=f(X,c(1),c(2),c(3));
% побудова графіка графіків
plot(x,y,'o',X,Y)
```

Отримаємо:

```
c =
    2.4578
   -0.6242
    0.0030
```

Отже, задані експериментальні дані можна апроксимувати емпіричною залежністю: $y = e^{2.45-0.62x+0.003x^2}$.



Додаток Basic Fitting

Цей додаток надає простий графічний інтерфейс до деяких функцій MATLAB, призначений для наближення даних і дозволяє інтерполювати одновимірні дані кубічними сплайнами, кубічними поліномами Ерміта та апроксимувати дані поліномами в сенсі найменших квадратів. Крім того, є можливість обчислити норму похибки та обчислити значення апроксимаційної функції в заданих точках. Одночасна робота може вестися з декількома наборами одновимірних даних. Засоби додатку Basic Fitting дозволяють експортувати інформацію про побудований сплайн або поліном у робоче середовище MATLAB. Обчислені значення сплайна або полінома можуть бути також експортовані в робоче середовище.

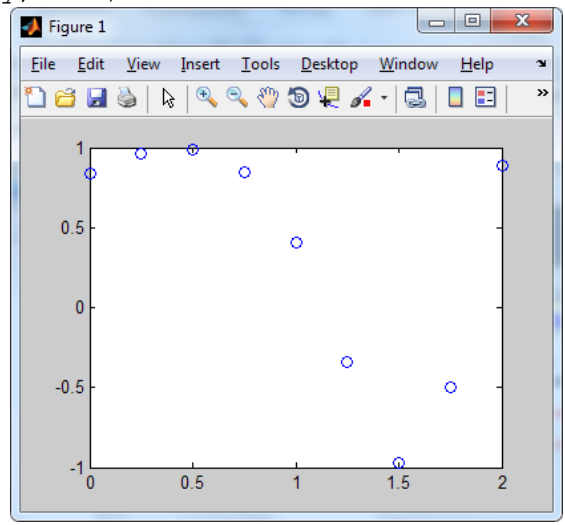
Запуск додатка Basic Fitting виконується з меню **Tools** графічного вікна в пункті Basic Fitting. Тому для запуску програми спочатку потрібно візуалізувати дані.

Приклад. Дані експерименту задані двома масивами `x` і `y` наступним чином:

```
x = 0: 0.25: 2;
y =[0.84  0.96  0.99  0.85  0.41  -0.34  -0.97
-0.50  0.89];
```

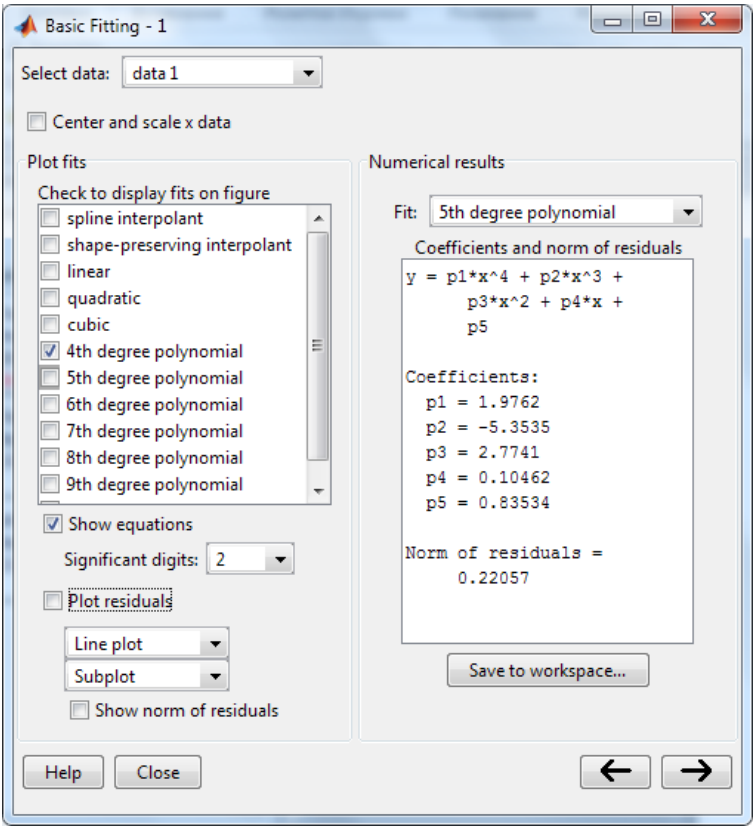
Побудуємо їх графік круглими маркерами, використовуючи функцію `plot`:

```
plot(x, y, 'o')
```

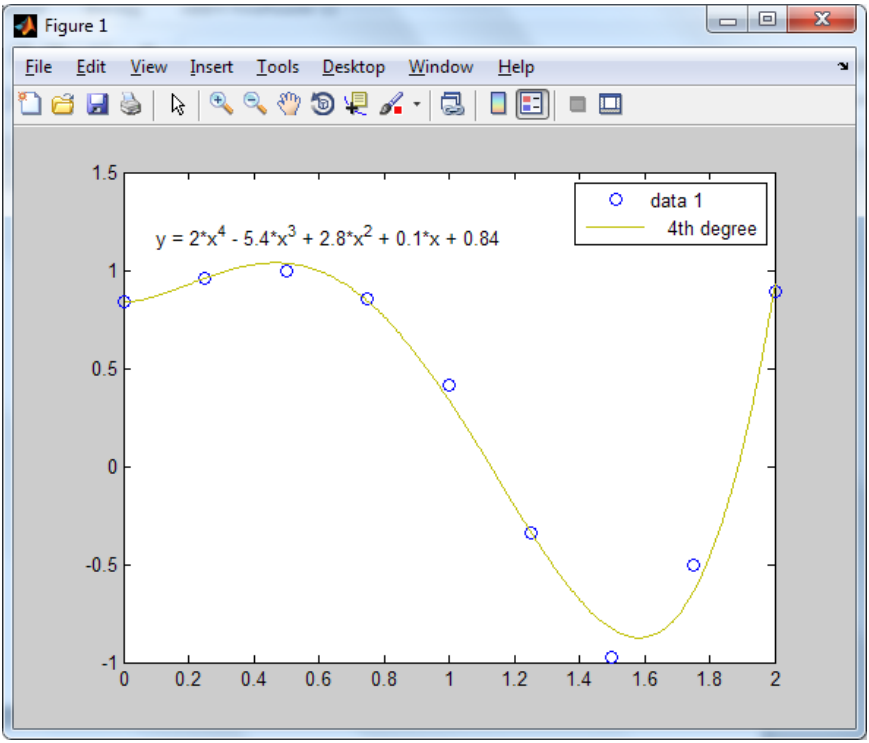


і запустимо додаток **Basic Fitting** з меню **Tools** графічного вікна Figure 1.

Апроксимуємо вихідні дані поліномом 4-го степеня, поставивши позначку навпроти 4th degree polynomial.



Отримаємо графік та рівняння апроксимаційного полінома.



Curve Fitting Toolbox

Curve Fitting Toolbox – це набір графічних інтерфейсів та M-функцій, створених в обчислювальному середовищі MATLAB. Пакет дозволяє використовувати:

- первинну обробку даних, таких як розбиття на області і згладжування;

- параметричну і непараметричну апроксимацію. У бібліотеку включені поліноми, експоненти, дробово-раціональні функції тощо. Для непараметричної апроксимації використовуються сплайни і різні методи інтерполяції;

- стандартний лінійний метод найменших квадратів, нелінійний метод найменших квадратів, зважений метод найменших квадратів, метод найменших квадратів з обмеженнями і стійкі методи апроксимації;

- можливість оцінки якості проведеної апроксимації;

- можливості аналізу, такі як екстраполяція, диференціювання та інтегрування.

Curve Fitting Toolbox запускається командою `cftool` з командного вікна MATLAB або з меню **Start->Toolboxes->Curve Fitting**.

Curve Fitting Toolbox містить ряд стандартних параметричних і непараметричних моделей апроксимаційних функцій, які вибираються в діалоговому вікні **Fitting**. Для переходу в це вікно слід натиснути кнопку **Fitting** в основному вікні програми. У Curve Fitting Toolbox є можливіть вибору стандартних типів функцій для параметричного та непараметричного наближення (параметри позначаються `a, b, c, d, a1, b1, p1...`).

Параметричні моделі

1. Експоненціальні моделі (Exponential):

ae^{bx}; $ae^{bx}+ce^{dx}$.

2. Відрізки ряду Фур'є (Fourier):

$$a_0 + a_1\cos xw + b_1\sin xw$$

$$a_0 + a_1\cos xw + b_1\sin xw + a_2\cos 2xw + b_2\sin 2xw$$

$$\vdots$$

$$a_0 + a_1\cos xw + b_1\sin xw + \ldots + a_8\cos 8xw + b_8\sin 8xw$$

3. Сума синусів (Sum of Sin Functions):

$$a_1\sin(b_1x + c_1)$$

$$a_1\sin(b_1x + c_1) + a_2\sin(b_2x + c_2)$$

$$\vdots$$

$$a_1\sin(b_1x + c_1) + a_2\sin(b_2x + c_2) + a_8\sin(b_8x + c_8)$$

4. Гаусові моделі (Gaussian)

$$a_1e^{-(x-b_1)^2/c_1^2}$$

$$a_1e^{-(x-b_1)^2/c_1^2} + a_2e^{-(x-b_2)^2/c_2^2}$$

$$\vdots$$

$$a_1e^{-(x-b_1)^2/c_1^2} + a_2e^{-(x-b_2)^2/c_2^2} + \ldots + a_8e^{-(x-b_8)^2/c_8^2}$$

5. Модель Вейбула (Weibull)

$abx^{b-1}e^{-ax^b}$.

6. Степеневі моделі (Power)

ax^b ; $ax^b + c$.

7. Поліноміальні моделі (Polynomials)

p_1

$p_1x + p_2$

$\vdots$

$p_1x^9 + p_2x^8 + \ldots + p_{10}$

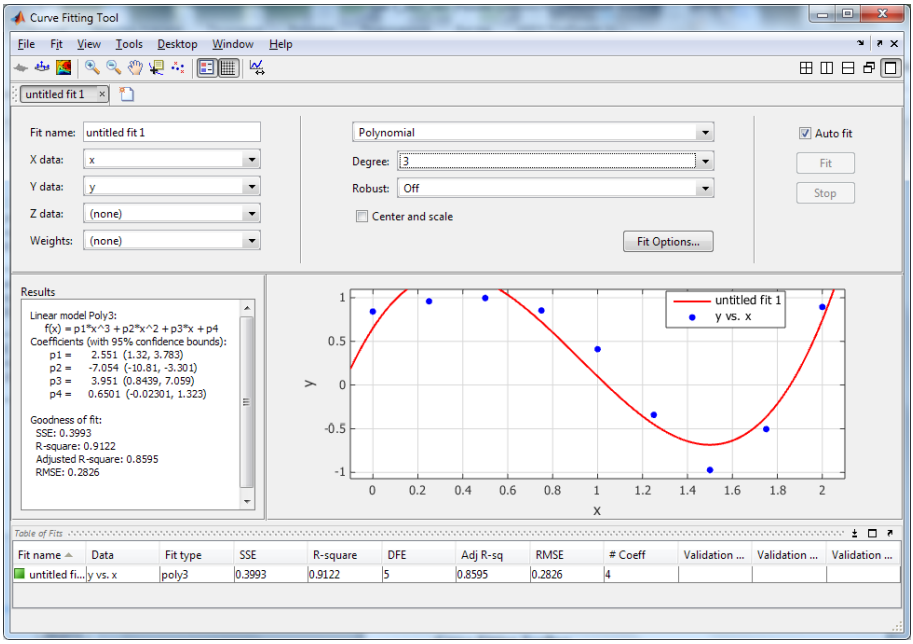
8. Дробово-раціональні моделі (Rational). Ці моделі представляються дробом, в чисельнику і знаменнику якого стоять поліноми до п'ятого степеня включно. Коефіцієнт при старшому степені в знаменнику дорівнює одиниці для однозначного визначення дробово-раціонального виразу.

$$\frac{\sum_{k=1}^{n+1} p_k x^{n+1-k}}{x^m + \sum_{k=1}^{m} q_k x^{n-k}}$$

Невідомими є коефіцієнти поліномів, що стоять в чисельнику і знаменнику дробу. При виборі моделі цього типу в списку Type of fit з'являються два списки для вибору степеня чисельника і знаменника.

Також є можливість задати довільний вигляд апроксимаційної залежності (Custom Equation).

Приклад апроксимації поліномом 3-го степеня за допомогою Curve Fitting Toolbox:



5.3. Програма роботи

1. Ознайомитися з методом найменших квадратів для апроксимації.
2. Розв'язати задачу апроксимації засобами Matlab згідно заданого варіанту.
3. Зробити висновки та оформити звіт.

5.4. Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями.
2. Вибрати завдання згідно варіанту (за номером у журналі підгрупи або за вказівкою викладача).
3. Завантажити середовище MATLAB. Розв'язати завдання згідно варіанту.
4. Зробити висновки. Оформити звіт про виконання роботи.

5.5. Завдання

Завдання 1.

У табл. 5.1 наведено результати спостережень за переміщенням x матеріальної точки по осі Ox в моменти часу $t \in [t_0, T]$. Відомо, що рух є рівномірним і описується лінійною залежністю $x(t)=vt+b$. Використовуючи метод найменших квадратів, визначити швидкість v і спрогнозувати положення точки в момент часу $t=2T$. В одних координатних осях побудувати графік руху точки і точковий графік вихідних спостережень.

Завдання розв'язати двома способами:

1) безпосереднє знаходження лінійної апроксимаційної функції в MATLAB (у звіті привести результати, тексти програм і графіки).

2) використання програми `cftool`. (У звіті навести результати та необхідні скріншоти і графіки).

Таблиця 5.1

Варіанти

1	***t***	1	1.4	1.8	2.6	3	3.4	3.8	4.2	4.6	5
	x	10.60	18.01	25.85	44	50.64	60.2	68.27	77.77	84.50	93.4
2	***t***	1	1.63	2.25	2.88	3.5	4.13	4.75	5.375	6	
	x	14.86	27.15	41.19	54	69.03	81.6	96.11	109.4	124.03	
3	***t***	0	0.5	1	1.5	2	2.5	3	3.5	4	
	x	3.72	9.38	15.53	22	29.52	35.2	42.35	48.61	55.51	
4	***t***	0	0.6	1.2	l.8	2.4	3	4.2	4.8	5.4	6
	x	6.45	19.97	33.91	48.2	64.15	76.9	106.2	122.2	135.6	149
5	***t***	2	3.2	4.4	5	5.6	6.8	7.4	8		
	x	18.50	35.73	54.65	62.4	71.74	90.5	98.10	107.6		
6	***t***	5	5.5	6	6.5	7	7.5	8	8.5	9	
	x	13.85	14.30	15.84	16.9	18.89	19.7	21.03	22.08	23.95	

Завдання 2.

У результаті експерименту отримано таблицю чисел x_i, y_i (табл.5.2). Знайти наближену функцію, що найкраще описує залежність $y=f(x)$.

Завдання розв'язати трьома способами:

1) безпосереднє знаходження лінійної апроксимаційної функції в MATLAB (у звіті привести результати, тексти програм і графіки);

2) за допомогою додатка Basic Ftitting;

3) за допомогою пакету Curve Fitting Toolbox.

Таблиця 5.2

Варіант 1		Варіант 2		Варіант 3		Варіант 4	
x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
0,05	0,0500	0,210	4,831	1,415	0,8885	0,101	1,261
0,10	0,1003	0,215	4,722	1,420	0,8895	0,106	1,276
0,17	0,1716	0,220	4,618	1,425	0,8906	0,111	1,291
0,25	0,2553	0,225	4,519	1,430	0,8916	0,116	1,306
0,30	0,3093	0,230	4,424	1,435	0,8926	0,121	1,321
0,36	0,3764	0,235	4,333	1,440	0,8936	0,126	1,326

Варіант 5		Варіант 6		Варіант 7		Варіант 8	
x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
0,43	1,635	0,02	1,023	0,35	2,739	0,41	2,574
0,48	1,732	0,08	1,095	0,41	2,300	0,46	2,325
0,55	1,876	0,12	1,147	0,47	1,968	0,52	2,093
0,62	2,033	0,17	1,214	0,51	1,787	0,60	1,862
0,70	2,228	0,23	1,301	0,56	1,595	0,65	1,749
0,75	2,359	0,30	1,409	0,64	1,343	0,72	1,620

Варіант 9		Варіант 10		Варіант 11		Варіант 12	
x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
0,68	0,808	0,11	9,054	1,375	5,041	0,115	8,657
0,73	0,894	0,15	6,616	1,380	5,177	0,120	8,293
0,80	1,029	0,21	4,691	1,385	5,320	0,125	7,958
0,88	1,209	0,29	3,351	1,390	5,470	0,130	7,648
0,93	1,340	0,35	2,739	1,395	5,629	0,135	7,362
0,99	1,523	0,40	2,365	1,400	5,797	0,140	7,096

Варіант 13		Варіант 14		Варіант 15		Варіант 16	
x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
0,150	6,616	0,180	5,615	0,210	4,831	1,415	0,8885
0,155	6,399	0,185	5,466	0,215	4,722	1,420	0,8895
0,160	6,196	0,190	5,326	0,220	4,618	1,425	0,8906
0,165	6,005	0,195	5,193	0,225	4,519	1,430	0,8916
0,170	5,825	0,200	5,066	0,230	4,424	1,435	0,8926
0,175	5,655	0,205	4,946	0,235	4,333	1,440	0,8936

Варіант 17		Варіант 18		Варіант 19		Варіант 20	
x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
0,115	0,808	0,11	9,054	1,375	5,041	0,68	8,657
0,120	0,894	0,15	6,616	1,380	5,177	0,73	8,293
0,125	1,029	0,21	4,691	1,385	5,320	0,80	7,958
0,130	1,209	0,29	3,351	1,390	5,470	0,88	7,648
0,135	1,340	0,35	2,739	1,395	5,629	0,93	7,362
0,140	1,523	0,40	2,365	1,400	5,797	0,99	7,096

5.6. Контрольні запитання

1. Що таке апроксимація?
2. У чому полягає відмінність апроксимації від інтерполяції?
3. У чому полягає практичне значення апроксимації експериментальних даних?
4. У чому полягає метод найменших квадратів?
5. Як визначають параметри апроксимаційної функції при застосуванні МНК?
6. Що таке лінійна апроксимація?
7. Що таке поліноміальна апроксимація?
8. Які функції для апроксимації дослідних даних застосовують найчастіше?
9. Які існують засоби апроксимації даних в Matlab?

Лабораторна робота №6
Обчислення інтегралів у MATLAB

6.1. Мета роботи

Вивчити основні методи обчислення визначених і невизначених інтегралів. Навчитися розв'язувати задачі чисельного та аналітичного інтегрування. Ознайомитися з методикою застосування програмних продуктів для обчислення інтегралів.

6.2. Теоретичні відомості

Аналітичне інтегрування.

Для обчислення визначених і невизначених інтегралів аналітичним способом в MATLAB використовується функція `int`.

`int(S)` – обчислює невизначений інтеграл від функції S за її символьною змінною, оголошеною в `syms`;

`int(S,v)` – обчислює невизначений інтеграл від функції S за її символьною змінною v, оголошеною в `syms`.

`int(S,a,b)` – обчислює визначений інтеграл від функції S за її символьною змінною від a до b. a і b можуть бути дійсними або символьними числами;

`int(S,v,a,b)` – обчислює визначений інтеграл від функції S за її символьною змінною v, оголошеною в syms.

Чисельне інтегрування.

Для наближеного обчислення значень визначених інтегралів у Matlab використовуються функції `quad(), quad1(), trapz()`.

`quad(fun, a, b)` – повертає значення інтеграла від функції `fun` на відрізку [*a, b*], при обчисленні використовується метод Сімпсона.

Приклад:

```
Q = quad(@myfun,0,2);
```

Файл myfun.m має містити підінтегральну функцію, наприклад:

```
function y = myfun(x)
y = 1./(x.^3-2*x-5);
```

Функція `quad('fun', a, b, tol)` – обчислює значення інтеграла від функції `fun` на відрізку [*a, b*] метод Сімпсона із заданою абсолютною похибкою.

`quadl (fun, a, b)` – повертає значення інтеграла від функції `fun` на відрізку [*a, b*], використовуючи для обчислення метод Лоббато.

Функція `trapz(y)` – повертає значення визначеного інтеграла в припущенні, що `x = 1:length(y)`.

Функція `trapz(x, y)` обчислює інтеграл від функції `y(x)` по `x` методом трапецій. Аргумент і функція задаються у вигляді векторів або *x* – у вигляді вектора, а *y* - у вигляді матриці. Якщо `y(x)` – матриця, то функція повертає вектор значень інтеграла для кожного стовпця матриці.

Приклад.

Нехай підінтегральна функція має вигляд $y(x) = x \cdot e^x + ln(x) + 1$. Необхідно обчислити визначений інтеграл в діапазоні від 1 до 10 з кроком 0.5.

Розв'язок:

```
x = 1:0.5:10;
y = x.* exp(x)+log(x)+1;
trapz(x,y)
```

Відповідь: `ans = 2.032841320958599e + 005`

Також можна обчислювати значення визначених інтегралів за допомогою програмної реалізації будь-якого з чисельних методів інтегрування.

Приклад.

Обчислити значення визначеного інтеграла:

$$\int_0^{\sqrt{\frac{\pi}{2}}} x \cdot \sin(x^2) dx$$

методом правих прямокутників.

Розв'яжемо цю задачу в MATLAB. Для цього складемо наступну програму:

```
f=inline('x.*sin(x.^2)');     % підінтегральна функція
Xmin=0; % нижня межа інтегрування
Xmax=sqrt(pi/2); % верхня межа інтегрування
N=1001; % кількість вузлів
```

```
  i=1:N;
  dx=(Xmax-Xmin)/(N-1);    % обчислення кроку
інтегрування
  x=Xmin:dx:Xmax;  % обчислення координат вузлів
сітки
  y=feval(f,x);      %      обчислення      значень
підінтегральної функції у вузлах
  % обчислення інтеграла за формулою правих
прямокутників
  m=2:N;
  y1(m-1)=y(m);
  I=sum(y1)*dx
  % обчислюємо похибку інтегрування
  err=I-0.5 % 0.5 – точне значення інтеграла
```

В результаті виконання програми отримаємо значення інтеграла I та значення похибки err між точним значенням інтеграла і значенням, отриманим методом правих прямокутників.

```
  I =
      0.5008
  err =
    7.8553e-004
```

Складемо програму для обчислення інтеграл $\int_0^{\sqrt{\frac{\pi}{2}}} x\cdot\sin(x^2)dx$ методом Сімпсона.

```
  f=inline('x.*sin(x.^2)');    %    підінтегральна
функція
  Xmin=0; % нижня межа інтегрування
  Xmax=sqrt(pi/2); % верхня межа інтегрування
  N=1001; % кількість вузлів
  i=1:N;
  dx=(Xmax-Xmin)/(N-1);    %    обчислення    кроку
інтегрування
  x=Xmin:dx:Xmax;  % обчислення координат вузлів
сітки
  y=feval(f,x);      %      обчислення      значень
підінтегральної функції у вузлах
  % обчислення інтеграла за формулою правих
прямокутників
```

```
s=0;
for i=2:N-1;
    if i-2*ceil(i/2)==0
        k=4;
    else
        k=2;
    end;
    s=s+k*y(i);
end;
Is=(y(1)+s+y(N))*dx/3
% обчислюємо похибку інтегрування
err2=Is-0.5
```

Отримаємо:

```
Is =    0.5000
err2 = -5.9908e-013
```

6.3. Програма роботи

1. Ознайомитися з методами обчислення інтегралів.
2. Розв'язати задачу засобами Matlab згідно заданого варіанту.
3. Зробити висновки та оформити звіт.

6.4. Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями.
2. Вибрати завдання згідно варіанту (за номером у журналі підгрупи або за вказівкою викладача).
3. Завантажити середовище MATLAB. Розв'язати завдання згідно варіанту.
4. Зробити висновки. Оформити звіт про виконання роботи.

6.5. Завдання

1. Реалізуйте програму обчислення інтегралів по формулах прямокутників (непарні варіанти) і Сімпсона (парні варіанти) з виведенням проміжних значень у вигляді таблиці:

№	x_i	y_i

2. Реалізуйте програму обчислення інтегралів із заданою точністю для метода Симпсона.

3. Виберіть інтеграли відповідно до вашого варіанту і обчисліть їх за допомогою програм.

4. Для заданих інтегралів знайдіть оцінку із заданим числом вірних знаків (не менше 6) за допомогою функцій MatLab.

Таблиця 6.1

1	$\int_{0.8}^{1.6} \frac{dx}{\sqrt{2x^2+1}}$	$\int_{0.5}^{1} (x+1)\frac{\sin(x+2)}{x}\,dx$
2	$\int_{0}^{\sqrt{\pi}} \frac{dx}{\sqrt{x^2+\pi}}$	$\int_{1}^{\pi} e^{-x} / x(1+\ln(x))dx$
3	$\int_{0}^{\pi} \frac{dx}{\sqrt{x^2+\pi^2}}$	$\int_{0.5}^{1} \frac{\cos(x)}{x^2+1} e^{-x}\,dx$
4	$\int_{0.5}^{1} x^2 e^{-x}\,dx$	$\int_{0.5}^{1} \frac{\cos(x^2+2)}{x}\,dx$
5	$\int_{0.}^{1} x^3 e^{-x^2}\,dx$	$\int_{0}^{1} \frac{\ln(x+1)}{x} e^{-x}\,dx$
6	$\int_{0}^{0.6} x \cdot tg(x^2+1)dx$	$\int_{0}^{1} \frac{\ln^2(x+1)}{x} e^{-x}\,dx$
7	$\int_{0}^{1} \frac{x+1}{\sqrt{x^2+1}} e^{-x}\,dx$	$\int_{0}^{1} \frac{\ln(x+1)}{x} e^{-x^2}\,dx$

8	$\int_0^1 \frac{3^x \cdot \ln(1+3^x)}{1+3^x}\,dx$	$\int_0^{0.5} x \cdot tg(x^2+1)\ln(x+1)\,dx$
9	$\int_0^{\sqrt{3}/2} \frac{\arcsin(x)}{\sqrt{1-x^2}}\,dx$	$\int_0^1 \frac{x+1}{\sqrt{x^2+4}}\,arctg(x)\,dx$
10	$\int_0^1 \frac{arctg(x)}{x^2+1}\,dx$	$\int_0^1 \frac{x^2+1}{\sqrt{x^2+4}}\ln\frac{2-x}{2+x}\,dx$
11	$\int_0^{0.9} \frac{dx}{1-x^2}$	$\int_0^1 \frac{arctg(x)}{x}e^{-x}\,dx$
12	$\int_0^{0.5} x \cdot tg(x^2+1)\,dx$	$\int_1^{\pi} dx/(1+\ln(x))$
13	$\int_1^{\pi} dx/(x(1+\ln(x)))$	$\int_1^{\pi} e^{-x}/(x(1+\ln^2(x)))\,dx$
14	$\int_0^1 \frac{x+1}{\sqrt{x^2+4}}\,dx$	$\int_1^{\pi} e^{-x}/(x(1+\ln(x)))\,dx$
15	$\int_0^{0.5} \frac{\arcsin(x)}{\sqrt{1-x^2}}\,dx$	$\int_0^{2\pi} \sin(x)\cdot e^{-x}/(1+e^{-x})\,dx$
16	$\int_0^{0.5} x \cdot tg(x^2+1)\,dx$	$\int_1^{\pi} dx/(1+\ln(x))$

17	$\int_0^{\pi} e^{-x} / {1+e^{-x}} dx$	$\int_1^{\pi} e^{-x} / {x(1+\ln(x))} dx$
18	$\int_0^{0.5} \frac{\arcsin(x)+1}{\sqrt{1-x^2}} dx$	$\int_{-0.5}^{0.5} \frac{\arcsin(x)}{\sqrt{1-x^2}} e^{-x} dx$
19	$\int_0^{0.5} \frac{\arcsin(x)}{\sqrt{1-x^2}} dx$	$\int_0^{2\pi} \sin(x) \cdot e^{-x} / {1+e^{-x}} dx$

6.6. Контрольні запитання

1. Як в чисельних методах можна зменшити похибку обчислення інтеграла?
2. Якою апроксимуючою замінюється підінтегральна функція в методах прямокутників, трапецій і Симпсона?
3. Алгоритми чисельних методів обчислення визначених інтегралів.
4. Функції системи MatLab для обчислення інтегралів.

Лабораторна робота №7
Чисельне розв'язування диференціальних рівнянь першого порядку

7.1. Мета роботи

Ознайомитися з методами чисельного розв'язування диференціальних рівнянь першого порядку. Навчитися розв'язувати задачу Коші методами Ейлера та Рунге – Кутта.

7.2. Теоретичні відомості

7.2.1. Загальні відомості

Велика кількість задач у різних областях науки і техніки при їх математичному моделюванні зводяться до диференціальних рівнянь. ***Диференціальними рівняннями*** називають такі рівняння, які, крім невідомих функцій однієї або кількох незалежних змінних, містять також і їх похідні. Диференціальні рівняння (ДР) називають звичайними (ЗДР), якщо невідомі функції є функціями однієї змінної, в іншому випадку – рівняннями в частинних похідних.

Рівняння

$$F(x, y, y', y'', ...y^{(n)}) = 0\,, \tag{7.1}$$

що зв'язує змінну x, невідому функцію $y=y(x)$ та її похідні до порядку n включно, називають ЗДР n-го порядку. Розв'язати диференціальне рівняння (7.1) – означає знайти функціональну залежність $y=y(x)$, що перетворює це рівняння в тотожність.

Загальний вигляд ДР першого порядку:

$$\frac{dy}{dx} = f(x, y)\,, \tag{7.2}$$

де x – незалежна змінна; dy/dx – похідна функції $y(x)$; f – деяка задана функція.

Загальний розв’язок диференціального рівняння (7.2) $y=y(x)$ містить довільну константу. Отже, ЗДР має нескінченну множину розв’язків. Для знаходження єдиного розв’язку потрібно задати початкові умови. Залежно від способу задання цих умов розрізняють дві задачі для ЗДР:

1) задача Коші;

2) крайова задача.

Додатковими умовами можуть бути значення шуканої функції або її похідних. Якщо умови задаються в одній точці відрізка $x \in [a, b]$ і, як правило, на його початку $x = x_0 = a$, тоді це задача Коші з початковою точкою. Якщо додаткові умови задаються в точках $x = a$ та $x = b$ – це крайова задача з граничними умовами. Загальним розв'язком для ДР першого порядку буде $y = f(x, c)$, частковим розв'язком буде $y = f(x, c_0)$.

Методи розв'язку ЗДР можна поділити на такі групи:

- графічні;
- аналітичні;
- наближені аналітичні;
- чисельні.

Перші три методи розглядаються в курсі диференціальних рівнянь.

Чисельні методи є основним інструментом при розв'язанні задач за допомогою ЕОМ.

Найпоширенішими чисельними методами розв'язку ЗДР є *методи скінченних різниць*, суть яких полягає в заміні області неперервної зміни аргументу (наприклад, відрізок) дискретною множиною точок, які називають вузлами. Ці вузли складають різницеву сітку. На ній шукана функція неперервного аргументу замінюється наближеною функцією дискретного аргументу, тобто розв'язок ДР зводитися до відшукання значень сіткової функції у вузлах сітки.

З їх допомогою можна визначити лише частковий розв'язок, але вони застосовуються до широких класів рівнянь і до всіх типів задач. Слід зауважити, що як у всіх наближених задачах тут також потрібні дослідження на коректність і точність розв'язків. Це докладно розглядається в спеціальній літературі.

7.2.2. Задача Коші для ЗДР

Залежно від виду ДР (7.1) задача Коші формулюється наступним чином. Якщо $n = 1$ (тобто похідна не вище першого порядку), то потрібно знайти $y = y(x)$, що задовольняє рівняння:

$$\frac{dy}{dx} = f(x, y) \tag{7.3}$$

і приймає при $x = x_0$ задане значення y_0:

$$y(x_0) = y_0. \tag{7.4}$$

Будемо вважати, що розв'язок потрібно отримати для значень $x > x_0$. Як початкового значення може бути задане довільне x, але найчастіше приймають $x_0 = 0$. Зауважимо, що всі нижченаведені чисельні методи розроблені для розв'язку ЗДР саме першого порядку.

7.2.3. Чисельні методи розв'язання задачі Коші

Для розв'язання задачі Коші (7.3) – (7.4) за різницевими методами введемо послідовність точок $x_0, x_1, ..., x_n$ і крок $h_i = x_{i+1} - x_i$ ($i = 0, 1, ..., n-1$). У кожному вузлі x_i замість значень функції $y(x_i)$ вводяться числа y_i, як результат апроксимації точного розв'язку $y(x)$ на даній множині точок. Функцію y, задану у вигляді таблиці $\{x_i, y_i\}$ називають ***сітковою функцією***. Замінюючи значення похідної у рівнянні (7.3) відношенням скінченних різниць здійснюємо перехід від диференціальної задачі (7.3) – (7.4) відносно функції $y(x)$ до різницевої задачі відносно сіткової функції

$$y_{i+1} = F(x_i, h_i, y_{i+1}, y_i, ...y_{i-k+1}), \quad i = 1, 2, ...; \tag{7.5}$$

$$y_0 = y_0. \tag{7.6}$$

Рівняння (7.5) є різницевим рівнянням у загальному вигляді, а конкретний вираз правої частини тут залежить від способу апроксимації похідної. Для кожного чисельного методу отримуємо свій вигляд рівняння (7.5).

Якщо в правій частині рівняння (7.5) відсутнє y_{i+1}, тобто значення y_{i+1} обчислюється по k за допомогою попередніх значень $y_i, y_{i-1}, ...,$ y_{i-k+1}, то різницева схема називається явною. При цьому має місце k-кроковий метод: $k = 1$ – однокроковий, $k=2$ – двокроковий і т.д., тобто в однокрокових методах для обчислення y_{i+1} використовується лише одне знайдене значення на попередньому кроці y_i, в багатокроковому – багато з них.

Якщо y_{i+1} входить в праву частину (7.5), то це будуть неявні методи, реалізація яких носить тільки ітераційний характер.

7.2.4. Однокрокові методи розв'язання задачі Коші

Найпростішими чисельними методами розв'язування задачі Коші для ЗДР є такі методи.

1. Метод Ейлера

Цей метод базується на розкладанні шуканої функції $y(x)$ в ряд Тейлора в околах вузлів системи $x = x_i$ ($i = 0, 1, 2, \ldots, n$), в якому відкидаються всі члени, що містять похідні другого і більш високих

порядків. Як правило, використовується рівномірна сітка $\Delta x = x_{i+1} - x_i = h = \text{const}$ ($i = \overline{0,n}$). Розклад запишемо у вигляді

$$y(x_i + \Delta x) = y(x_i) + y'(x_i) \cdot \Delta x_i + O(\Delta x_i^2) \ . \qquad (7.7)$$

Замінюючи значення функції $y(x)$ у вузлах сітки x_i значеннями сіткової функції і використовуючи рівняння (7.3), отримаємо

$$y'(x_i) = f(x_i, y(x_i)) = f(x_i, y_i) \ .$$

Тоді з (7.7) отримаємо

$$y_{i+1} = y_i + h \cdot f(x_i, y_i); \quad i = 0,1,2,...,n-1 \ . \qquad (7.8)$$

При $i = 0$, для вузла $x = x_1$ маємо $y_1 = y_0 + h \cdot f(x_0, y_0)$.

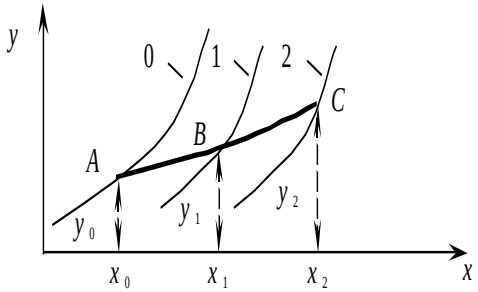
Далі за алгоритмом (7.8)

$y_2 = y_1 + h \cdot f(x_1, y_1)$;

. . .

$y_n = y_{n-1} + h \cdot f(x_{n-1}, y_{n-1})$.

Геометрична інтерпретація має вигляд:



На рис. лінія «0» – точний розв'язок, лінії «1» і «2» – наближені розв'язки.

Шукана інтегральна крива $y(x)$, що проходить через точку (x_0, y_0), замінюється ламаною з вершинами в точках (x_i, y_i). Кожна ланка ламаної має напрямок, що співпадає з напрямком інтегральної кривої (7.3), яка проходить через точку (x_i, y_i).

Локальна похибка методу Ейлера, як видно з (7.7), оцінюється, як $O(h^2)$. Весь інтервал $[a, b]$ розбивається на n частин, тоді загальна похибка

$n\ O(h^2) = \dfrac{1}{h} O(h^2) = O(h)$ – 1-й порядок.

Для оцінки похибки при машинному розрахунку користуються подвійним прорахунком, тобто на відрізку $[x_i, x_{i+1}]$ розрахунок

повторюють з кроком $h/2$ і похибка більш точного розв'язку y^*_{i+1} (при кроці $h_i/2$) оцінюється як різниця $\left|y^*_{i+1} - y_{i+1}\right|$.

2. Метод Ейлера з перерахунком

При цьому підході рекурентне співвідношення (7.8) видозмінюється, а саме, замість $f(x_i, y_i)$ беруть середнє арифметичне між $f(x_i, y_i)$ і $f(x_{i+1}, y_{i+1})$.

Тоді

$$y_{i+1} = y_i + \frac{h}{2}\left[f\left(x_i, y_i\right) + f\left(x_{i+1},\, y_{i+1}\right)\right], \quad i = 0,1,... \tag{7.9}$$

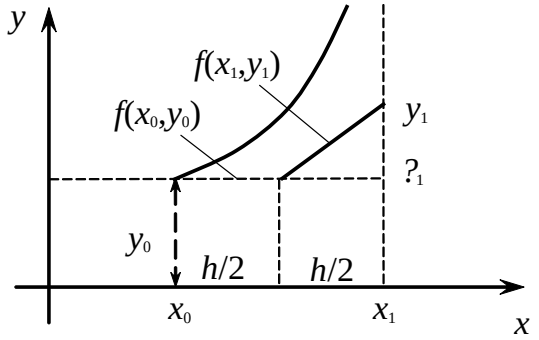
Це неявна схема. Вона реалізується у дві ітерації: спочатку знаходиться перше наближення по (7.8), вважаючи y_i початковою

$$\tilde{y}_{i+1} = y_i + hf\left(x_i, y_i\right), \tag{7.10}$$

потім (7.10) підставляється в праву частину (7.9) замість y_{i+1}

$$y_{i+1} = y_i + \frac{h}{2}\left[f\left(x_i, y_i\right) + f\left(x_{i+1}, \tilde{y}_{1+i}\right)\right], \qquad i = 0,1,... \tag{7.11}$$

Геометрична інтерпретація методу:



За допомогою методу Ейлера з перерахунком можна виконувати контроль точності, порівнюючи y_{i+1} та $\tilde{y}_{i+1}$.

На основі цього можна вибирати крок. Якщо величина $|\tilde{y}_{i+1} - y_{i+1}|$ співрозмірна із заданою точністю ε, то крок можна збільшувати, якщо більша, то зменшувати, тобто має місце схема подвійного перерахунку з оцінкою похибки за величиною

$$\frac{1}{3}\left|y^*_i - y_i\right| \approx \left|y^*_i - y(x_i)\right|,$$

де $y(x_i)$ – точний розв'язок у точці $x = x_i$, а y_i та y^*_i – наближені значення, отримані з кроком h і $h/2$ відповідно.

2. *Метод Рунге–Кутта*

На його основі можуть бути побудовані різницеві схеми різного порядку точності. Ідея його реалізації полягає в підгонці ряду Тейлора при розкладанні шуканої функції $y = y(x)$ в околі вузлів сітки в плані підвищення точності цього розкладання, а саме, збільшення числа похідних вищого порядку без їх безпосереднього визначення через складність аналітичних виразів повних похідних по x від функції $f(x,y)$.

Розглянемо різницеву схему четвертого порядку, що найбільш широко застосовується на практиці.

Її алгоритм полягає в наступному:

$$\left.\begin{aligned} y_{i+1} &= y_i + \Delta y_i; \\ \Delta y_i &= \frac{1}{6}(k_1^{(i)} + k_2^{(i)} + k_3^{(i)} + k_4^{(i)}); \end{aligned}\right\} \quad i = 0,1,2,... \quad (7.12)$$

де $k_1^{(i)} = hf(x_i, y_i)$;

$k_2^{(i)} = hf(x_i + h/2, y_i + k_1^{(i)}/2)$;

$k_3^{(i)} = hf(x_i + h/2, y_i + k_2^{(i)}/2)$;

$k_4^{(i)} = hf(x_i + h,\ y_i + k_3^{(i)})$.

У цій розрахунковій схемі Рунге–Кутта на кожному кроці обчислення y_i потрібно 4 рази звернутися до правої частини рівняння $f(x,\ y)$, тобто метод Рунге–Кутта (7.12) вимагає більшого обсягу обчислень, однак він має підвищену точність, що дозволяє проводити розрахунок з великим кроком.

Можна показати, що метод Ейлера і його модифікований варіант є аналогом методу Рунге–Кутта першого і другого порядку, проте для досягнення однакової точності у них крок розрахунку буде значно меншим.

Для даного методу крок розрахунку можна змінювати при переході від однієї точки до іншої. Для контролю правильності вибору кроку h рекомендується обчислити дріб

$$Q = \left| \frac{k_2^{(i)} - k_3^{(i)}}{k_1^{(i)} - k_2^{(i)}} \right| .$$

Величина Q не повинна перевищувати декількох сотих. В іншому випадку h слід зменшувати.

Найчастіше використовується груба оцінка похибки за формулою $\left|y_n^* - y(x_n)\right| \approx \frac{\left|y_n^* - y_n\right|}{15}$, де $y(x_n)$ – значення точного розв'язку рівняння (7.3) в точці x_n, а y_n^*, y_n – наближений розв'язок, отриманий з кроком $h/2$ і h.

При реалізації на ЕОМ методу Рунге–Кутта з автоматичним вибором кроку, зазвичай у кожній точці x_i роблять подвійний перерахунок спочатку з кроком h, потім з $h/2$. Якщо отримане y_i при цьому знаходиться в межах допустимої точності, то крок h для наступної точки x_{i+1} подвоюють, в іншому випадку беруть половинний крок.

7.2.2. Чисельне розв'язування диференціальних рівнянь у MATLAB

Для чисельного розв'язування ЗДР в MATLAB існують різні методи. Їх реалізації названі розв'язувачами (solvers) ЗДР. Всі засоби розв'язування ЗДР (`ode45`, `ode23`, `ode133`, `ode15s`, `ode23s`, `ode23t`, `ode23tb`) можуть розв'язувати системи рівнянь явного виду *y '= f (t, y):*

$$\left.\begin{aligned} \bar{y}'(x) &= \bar{f}(x, \bar{y}(x)), \\ \bar{y}(x_0) &= \overline{y_0}. \end{aligned}\right\}$$

Розв'язувачі `ode15s`, `ode23s`, `ode23t`, `ode23tb` можуть розв'язувати рівняння неявного виду *F(t, y, y ') = 0.*

Розв'язувачі ДР в MatLab:

`ode45` – однокрокові явні методи Рунге-Кутта 4-го і 5-го порядку. У багатьох випадках він дає хороші результати;

`ode23` – однокрокові явні методи Рунге-Кутта 2-го і 3-го порядку. При помірної жорсткості системи ЗДР і низьких вимогах до точності цей метод може дати виграш у швидкості розв'язання;

`ode133` – багатокроковий метод Адамса–Башворта–Мултона змінного порядку. Це адаптивний метод, який може забезпечити високу точність розв'язку;

`ode15s` – багатокроковий метод змінного порядку (від 1-го до 5-го, за замовчуванням 5), що використовує формули чисельного диференціювання. Це адаптивний метод, його варто застосовувати,

якщо розв'язувач ode45 не забезпечує розв'язку (для жорстких систем);

ode23s – однокроковий метод, що використовує модифіковану формулу Розенброка 2-го порядку. Може забезпечити високу швидкість обчислень при низькій точності (для жорстких систем);

ode23t – метод трапецій з інтерполяцією. Цей метод дає хороші результати при розв'язанні завдань, що описують осцилятори з майже гармонійним вихідним сигналом;

ode23tb – неявний метод Рунге-Кутта на початку розв'язування і метод, що використовує формули зворотного диференціювання 2-го порядку в подальшому. При низькій точності цей метод може виявитися більш ефективним, ніж ode15s.

У найпростішому варіанті для розв'язання задачі Коші для систем звичайних диференціальних рівнянь досить скористатися командою

```
[T, X] = solver ('F', [DT], X0),
```

або `[T, X] = solver ('F', [DT], X0, opt),`

де DT – діапазон інтегрування, X0 – вектор початкових значень, F – ім'я функції обчислення правих частин системи (і відповідно ім'я m-файла), *solver* – назва функції (розв'язувача). Версії розв'язувачів відрізняються використовуваними методами (за замовчуванням відносна похибка 10^{-3} і абсолютна 10^{-6}) і відповідно часом та успішністю розв'язання. Під жорсткістю тут розуміють підвищену вимогу до точності – використання мінімального кроку у всій області інтегрування. При відсутності інформації про жорсткість рекомендується спробувати отримати рішення за допомогою ode45 і потім ode15s.

Якщо діапазон DT заданий початковим і кінцевим значенням [to, tn], то кількість елементів у масиві T (і в масиві розв'язків X) визначається необхідним для забезпечення точності кроком; при заданні DT у вигляді [to, t1, t2, ..., tn] або [to: Δt: tn] – зазначеними значеннями.

opt – аргумент, створюваний функцією odeset. Наприклад, команда

```
opt=odeset('namel',valuel,'name2',value2,...)
```

– створює структуру параметрів, в якій зазначені властивості 'name ...' приймають наступні за ними значення. Зокрема, за допомогою odeset можна змінити значення абсолютної і відносної похибки розв'язку. Наприклад

```
opt=odeset('AbsTol', valuel) або
opt=odeset('RelTol', valuel)
```

Приклад 1. Розв'язати задачу Коші

$$\frac{dy}{dx} = 2\left(x^2 + y\right) \text{ на } [0;\ 1] \quad y(0) = 1.$$

Точний розв'язок має вигляд

$$y(x) = 1{,}5e^{2x} - x^2 - x - 0{,}5.$$

Розв'яжемо цю задачу за допомогою методу `ode45`**.** Спочатку в М-Файл записуємо праву частину заданого диференціального рівняння. Створимо М-Файл з ім'ям `F`:

```
function dydx = F(x, y)
dydx = zeros(1,1);
dydx(1) = 2*(x^2+y(1));
```

Для чисельного розв'язку задачі Коші у вікні команд набираються наступні оператори.

Протокол програми.

```
>>[X Y] = ode45(@F,[0 1],[1]);
```

% Дескриптор @ забезпечує зв'язок з файлом-функцією правої частини

% [0 1] – інтервал на якому необхідно одержати розв'язок

% [1] – початкове значення розв'язку

```
>> plot(X,Y);
```

% Побудова графіка чисельного розв'язку задачі Коші

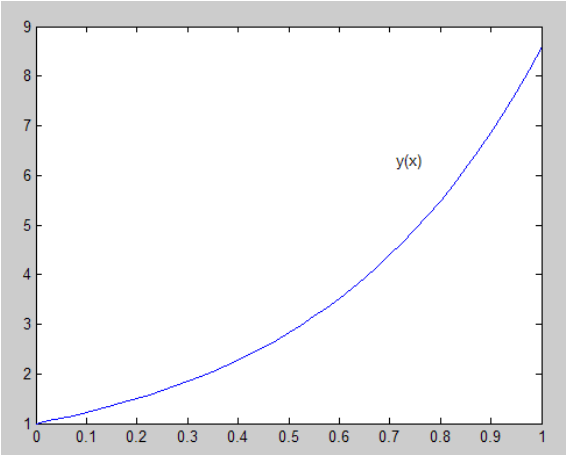
```
>> hold on; gtext('y(x)')
```

% Команда дозволяє за допомогою мишки нанести на графік напис y(x)

>>[X Y]

% Остання команда виводить таблицю чисельного розв'язку задачі.

Результати розв'язку. Графік розв'язку задачі Коші показаний на рисунку. Чисельний розв'язок представлений в таблиці 7.1, де наведені тільки окремі вузлові точки. У методі `ode45` за замовчуванням інтервал розбивається на 40 точок із кроком h = 1/40 = 0.025.



Таблиця 7.1

X_i	Метод Рунге–Кутта	Точний розв'язок
0.0	1.0	1.0
0.1	1.2221	1.2221
0.2	1.4977	1.4977
0.3	1.8432	1.8432
0.4	2.2783	2.2783
0.5	2.8274	2.8274
0.6	3.5202	3.5202
0.7	4.3928	4.3928
0.8	5.4895	5.4895
0.9	6.8645	6.8645
1.0	8.5836	8.5836

Як видно з табл. 7.1, чисельний розв'язок методом `ode45` є точним.

Приклад 2. Знайти розв'язок задачі Коші $y' = \frac{2}{x} y + x$, $y(1) = 0$, $x \in [1, 1.5]$. Для задання правої частини диференціального рівняння використовуємо inline-функцію:

```
>>fun2=inline('(2/t)*y+t')
```

```
fun2 =
    Inline function:
    fun2(t, y)=(2/t)*y+t
>>[T, Y]=ode45(fun2,[1 1.5],[0],[])
```

7.3. Програма роботи

1. Ознайомитися з методами чисельного розв'язування диференціальних рівнянь.

2. Розв'язати задачу Коші різними методами із заданою точністю. Оцінити похибку. Побудувати графік шуканої функції.

3. Зробити висновки та оформити звіт.

7.4. Порядок виконання роботи

1. Прочитати теоретичні відомості.

2. Вибрати завдання згідно варіанту (за номером у журналі групи або за вказівкою викладача).

3. Завантажити середовище MATLAB.

4. ***Завдання 1.*** Розв'язати звичайне диференціальне рівняння $\frac{dy}{dx} = f(x, y)$ методами Рунге – Кутта другого і четвертого порядку.

Знайти точний розв'язок рівняння аналітично, або використовуючи MATLAB.

Знайти наближене рішення задачі Коші на інтервалі [0; 1] з кроком h = 0.1 методами Рунге-Кутта 2-го і 4-го порядків. З цією метою створити необхідні m- файли MATLAB.

Побудувати таблиці значень наближених і точного розв'язків. В одній системі координат побудувати графіки наближеного й точного розв'язків.

Оцінити похибку наближених розв'язків за правилом Рунге і безпосередньо. Результати звести в таблицю і проаналізувати.

5. ***Завдання 2.*** Розв'язати рівняння для свого варіанту будь-яким методом, при цьому задати абсолютну похибку розв'язку 10^{-7}. Побудувати таблиці та графіки шуканих функцій. Переконатися, що розв'язок отримано із заданою точністю. Проаналізувати результати.

6. Зробити висновки. Оформити звіт про виконання роботи.

7.5. Варіанти завдань

Таблиця 7.2

Варіант	$f(x, y)$	y_0
1.	$x^3 \sin y + 1$	0.0
2.	$x^2 \sin y + 1$	0.1
3.	$e^x + 3y$	2.0
4.	$\sqrt{y^2 + x^3}$	0.3
5.	$\sqrt{y^3 + x^2}$	0.4
6.	$\frac{1}{1+y^2} + x^2$	0.0
7.	$\frac{1}{1+y^2} + xy$	0.1
8.	$\cos y + xy$	0.2
9.	$x^2 \cos y + 0.1$	0.3
10.	$x^3 \cos y + 0.1$	0.4
11.	$\cos(xy) - 0.5$	0.5
12.	$e^{-y} + e^x - 2$	0.0
13.	$e^{-y} - e^x - 0.1$	0.5
14.	$e^{-xy} + 1$	0.4
15.	$\sqrt{y^2 + x^4}$	0.3
16.	$(xy)^2 + \sqrt{x}$	0.2
17.	$\cos(x+y) + x^2$	0.1
18.	$\sin(x+y) - x^2$	0.0
19.	$\sin(xy) + 1$	0.1
20.	$\sin y + xy$	0.2
21.	$(x^2 + y^2)x$	0.3
22.	$e^x(1+xy)$	0.4
23.	$e^y(1+xy)$	0.5
24.	$\ln(1+x^2) + y$	0.6

25.	$1+ye^{x^2}$	0.7
26.	$\sin x^2 + y^2$	0.0
27.	$\cos x^2 + xy$	0.1
28.	$\sin\left(\frac{y}{1+x^2}\right)+1$	0.2
29.	$\ln(1+y^2)+1$	0.3
30.	$1+xe^{y^2}$	0.4

7.6. Контрольні запитання

1. Які рівняння називають диференціальними?
2. У чому полягає різниця між звичайним ДР та ДР в частинних похідних?
3. Що таке порядок ЗДР ?
4. Запишіть загальний вигляд ЗДР першого порядку.
5. Чому єдиний розв’язок звичайного диференціального рівняння існує лише за початкових умов?
6. У чому полягає суть задачі Коші?
7. Що є загальним розв’язком ЗДР?
8. У чому суть методу Ейлера для розв’язання ЗДР?
9. Поясніть суть методу Рунге–Кутта 4-го порядку для розв’язання ЗДР.
10. Які особливості методу Рунге–Кутта з автоматичним вибором кроку?

Лабораторна робота №8
Розв'язування крайової задачі для лінійного диференціального рівняння другого порядку методом прогону

8.1. Мета роботи

Навчитися розв'язувати крайову задачу для лінійного диференціального рівняння другого порядку методом прогону.

8.2. Теоретичні відомості

8.2.1. Загальні відомості

Розглянемо лінійне ДР другого порядку

$$y'' + p(x)y' + q(x)y = f(x) \qquad (8.1)$$

де $p(x), q(x), f(x)$ – відомі функції, неперервні на відрізку $[a, b]$.

Лінійна крайова задача для рівняння (8.1) полягає у знаходженні функції $y = y(x)$, що задовольняє рівняння (8.1) на проміжку $[a, b]$, якщо на кінцях проміжку задані лінійні крайові умови:

$$\left.\begin{aligned} \alpha_0 y(a) + \alpha_1 y'(a) = A, \\ \beta_0 y(b) + \beta_1 y'(b) = B, \end{aligned}\right\} \qquad (8.2)$$

де $\alpha_0, \alpha_1, \beta_0, \beta_1, A, B$ – відомі сталі, причому $\alpha_0, \alpha_1, \beta_0, \beta_1$ не дорівнюють одночасно нулю $\left(|\alpha_0| + |\alpha_1| \neq 0; |\beta_0| + |\beta_1| \neq 0\right)$.

Якщо $A=B=0$, то крайові умови (8.2) називають *однорідними.*

Лінійну крайову задачу називають *однорідною,* якщо однорідним є диференціальне рівняння (8.1) $(f(x) = 0)$ і крайові умови (8.2). У протилежному випадку крайову задачу (8.1), (8.2) називають *неоднорідною.* Умови (8.2) повинні виконуватися у двох точках – на кінцях відрізка $[a, b]$, тому їх називають *двоточковими крайовими умовами*, а крайову задачу – *двоточковою крайовою задачею.*

Наближені методи розв'язування лінійної крайової задачі поділяють на різницеві й аналітичні.

8.2.2. Метод скінченних різниць

Одним із найпростіших методів розв'язування лінійної крайової задачі (8.1), (8.2) є зведення її до системи скінченнорізницевих рівнянь.

Поділимо відрізок [*a, b*] на n рівних частин довжиною

$$h = \frac{b-a}{n}.$$

Позначимо точки поділу відрізка [*a,b*] $x_0 = a, x_n = b, x_i = x_0 + ih$, (*i=1, 2,…, n-1*);

$p_i = p(x_i), q_i = q(x_i), f_i = f(x_i), y(x_i) = y_i, y'(x_i) = y'_i, y''(x_i) = y''_i.$

Замінимо наближено в кожній внутрішній точці x_i відрізка *[a,b]* похідні $y'(x_i)$ та $y''(x_i)$ скінченнорізницевими відношеннями:

$$(i=1, 2, \ldots, n-1) \quad y'_i = \frac{y_{i+1} - y_i}{h}; \qquad (8.3)$$

$$y''_i = \frac{y_{i+2} - 2y_{i+1} + y_i}{h^2}$$

Для кінців відрізка $x_0 = a$ та $x_n = b$ візьмемо

$$y'_0 = \frac{y_1 - y_0}{h}; y'_n = \frac{y_n - y_{n-1}}{h}. \qquad (8.4)$$

З використанням (8.3) і (8.4), наближено замінимо рівняння (8.1) і крайові умови (8.2) системою *n+1* лінійних алгебраїчних рівнянь з *n+1* невідомими $y_0, y_2, \ldots, y_n$, що є значеннями шуканої функції $y = y(x)$ у точках $x_0, x_1, \ldots, x_n$:

$$\left.\begin{array}{l} \dfrac{y_{i+2} - 2y_{i+1} + y_i}{h^2} + p_i \dfrac{y_{i+1} - y_i}{h} + q_i y_i = f_i \\ (i = 0, 1, 2, \ldots, n-2), \\ \alpha_0 y_0 + a_1 \dfrac{y_1 - y_0}{h} = A, \\ \beta_0 y_n + \beta_1 \dfrac{y_n - y_{n-1}}{h} = B. \end{array}\right\} \qquad (8.5)$$

Розв’язавши цю систему, одержимо таблицю наближених значень шуканої функції $y = y(x)$.

На практиці часто похідні $y'(x_i)$ і $y''(x_i)$ у внутрішніх точках x_i відрізка $[a,b]$ замінюють центрально-різницевими відношеннями

$$y_1' = \frac{y_{i+1} - y_{i-1}}{2h}; \qquad y_1'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}; \tag{8.6}$$

а для кінців відрізка $x_0 = a$ та $x_n = b$ справедливими є формули (8.4) .

Тоді система рівнянь для знаходження $y_0, y_2, ..., y_n$ набуває вигляду

$$\left.\begin{aligned} &\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p_i \frac{y_{i+1} - y_{i-1}}{h} + q_i y_i = f_i \\ &(i = 0, 1, 2, ..., n-2), \\ &\alpha_0 y_0 + a_1 \frac{y_1 - y_0}{h} = A, \\ &\beta_0 y_n + \beta_1 \frac{y_n - y_{n-1}}{h} = B. \end{aligned}\right\} \tag{8.7}$$

Оцінку похибки методу скінченних різниць проводять із застосуванням такого наближеного співвідношення:

$$\left|y_i^* - y(x_i)\right| \approx \frac{1}{3}\left|y_i^* - y_i\right|,$$

де $y(x_i)$ – значення точного розв'язку крайової задачі в точці $x = x_i$, y_i – значення наближеного розв'язку, обчислене в точці $x = x_i$, з кроком h; y_i^* – значення наближеного розв'язку, обчислене в точці $x = x_i$, з кроком $h/2$.

Для знаходження наближеного розв'язку крайової задачі із заданою точністю ε, необхідно обчислити з кроком h чи $h/2$ і порівняти отримані результати.

Якщо $\left|y_i^* - y_i\right| < 3\varepsilon$, то $\left|y_i^* - y(x_i)\right| < \varepsilon$ і значення $y_i^* (i = 0,1,2,3,...,n-2)$ можна взяти за шуканий розв'язок крайової задачі.

8.2.3. Метод прогону

Якщо число n є великим, то розв'язок систем (8.5), (8.7) стає досить громіздким. Розглянемо *методу прогону (прогонки)*, розроблений спеціально для розв'язування таких систем.

Нехай маємо систему (8.5). Розглянемо перші $n-1$ рівнянь:

$$\frac{y_{i+2} - 2y_{i+1} + y_i}{h^2} + p_i \frac{y_{i+1} - y_i}{h} + q_i y_i = f_i \qquad (i = 0,1,2,...,n-2).$$

Після найпростіших перетворень одержимо:

$$y_{i+2}+(-2+hp_i)y_{i+1}+(1-hp_i+h^2q_i)y_i=h^2f_i. \tag{8.8}$$

Позначимо

$$\begin{aligned} &m_i=-2+hp_i;k_i=1-hp_i+h^2q_i \\ &(i=0,1,2,3,...,n-2), \end{aligned} \tag{8.9}$$

тоді запишемо (8.8) у вигляді

$$y_{i+2}+m_iy_{i+1}+k_iy_i=h^2f_i. \tag{8.10}$$

З рівняння (8.10), отримаємо:

$$y_{i+1}=\frac{h^2}{m_i}f_i-\frac{1}{m_i}y_{i+2}-\frac{k_i}{m_i}y_i. \tag{8.11}$$

Виключивши y_i з рівняння (8.11) за допомогою крайових умов системи (8.5), одержимо:

$$y_{i+1}=c_i(d_i-y_{i+2}),\ (i=0,1,2,...,n-2), \tag{8.12}$$

де c_i,d_i – коефіцієнти.

Нехай, наприклад, i=0, тоді рівняння (8.11) матиме вигляд

$$y_1=\frac{h^2}{m_0}f_0-\frac{1}{m_0}y_2-\frac{k_0}{m_0}y_0. \tag{8.13}$$

Знайдемо з крайової умови $\alpha_0y_0+\alpha_1\dfrac{y_1-y_0}{h}=A$

$y_0=\dfrac{Ah}{\alpha_0h-\alpha_1}-\dfrac{\alpha_1}{\alpha_0h-\alpha_1}-y_1$ і підставимо цей вираз у (8.13).

Отримаємо:

$$y_1=\frac{\alpha_1-\alpha_0h}{m_0(\alpha_1-\alpha_0h)+k_0\alpha_1}\left[\left(\frac{k_0Ah}{\alpha_1-\alpha_0h}+h^2f_0\right)-y_2\right].$$

Позначимо $c_0=\dfrac{\alpha_1-\alpha_0h}{m_0(\alpha_1-\alpha_0h)+k_0\alpha_1}$.

$$d_0=\frac{k_0Ah}{\alpha_1-\alpha_0h}+h^2f_0. \tag{8.14}$$

На основі (8.12) запишемо:

$$y_i=c_{i-1}(d_{i-1}-y_{i+1}).$$

Підставивши отриманий вираз у (8.10), одержимо $y_{i+2}+m_iy_{i+1}+k_ic_{i-1}(d_{i-1}-y_{i+1})=h^2f_i$, звідки

$$y_{i+1}=\frac{(h^2f_i-k_ic_{i-1}d_{i-1})-y_{i+2}}{m_i-k_ic_{i-1}}. \tag{8.15}$$

З (8.12) і (8.15), одержимо рекурентні формули для визначення коефіцієнтів c_i і d_i:

$$c_i = \frac{1}{m_i - k_i c_{i-1}}; \; d_i = h^2 f_i - k_i c_{i-1} d_{i-1}, \; (i = 0,1,2,..,n-2) \qquad (8.16)$$

За формулами (8.14) визначаємо коефіцієнти c_0 і d_0, потім послідовно застосовуючи рекурентні формули (8.16), знайдемо значення c_i, d_i (*i=1, 2, ..., n-2*). Це *прямих хід* методу прогону.

Зворотній хід починається з визначення y_n. Підставивши другу крайову умову системи (8.5) у формулу (8.12) при *i=n-2*, складемо систему рівнянь:

$$\begin{cases} \beta_0 y_n + \beta_1 \dfrac{y_n - y_{n-1}}{h} = B; \\ y_{n-1} = c_{n-2}(d_{n-2} - y_n). \end{cases} \qquad (8.17)$$

Розв'язавши цю систему, отримаємо

$$y_n = \frac{\beta_1 c_{n-2} d_{n-2} + Bh}{\beta_1 (1 + c_{n-2}) + B_0 h}. \qquad (8.18)$$

Підставивши у (8.18) вже знайдені прямим ходом значення c_{n-2}, d_{n-2}, визначимо y_n. Потім обчислимо y_{n-1}, y_{n-2}, y_{n-3}, … , y_1, послідовно застосовуючи рекурентну формулу (8.12):

$$\begin{gathered} y_{n-1} = c_{n-2}(d_{n-2} - y_n); \\ y_{n-2} = c_{n-3}(d_{n-3} - y_{n-1}); \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ y_1 = c_0(d_0 - y_2). \end{gathered} \qquad (8.19)$$

Таблиця 8.1

Схема методу прогону

i	x_i	m_i	k_i	f_i	Прямий хід	Зворотний хід	
					c_i	d_i	y_i
0	x_0	m_0	k_0	f_0	c_0	d_0	y_0
1	x_1	m_1	k_1	f_1	c_1	d_1	y_1
2	x_2	m_2	k_2	f_2	c_2	d_2	Y_2
...	...	...	...	...	...	...	...
n-2	x_{n-2}	m_{n-2}	k_{n-2}	f_{n-2}	c_{n-2}	d_{n-2}	y_{n-2}
n-1	x_{n-1}						y_{n-1}
n	x_n						y_n

Значення y_0 знайдемо за формулою:

$$y_0 = \frac{\alpha_1 y_1 + Ah}{\alpha_1 - \alpha_0 h}, \tag{8.20}$$

одержаною з першої крайової умови системи (8.5).

Обчислення зручно записувати у вигляді таблиці (табл. 8.1).

Перевагою методу прогону є те, що похибки округлення не призводять до необмеженого зростання похибки розв'язку.

8.2.4. Приклад розв'язування крайової задачі в MATLAB

Розв'яжемо крайову задачу на відрізку [0,1: 1,1], розділивши його на 10 рівних частин

$$\begin{cases} y'' + e^x y' + \dfrac{x}{2} y = x^2, \\ y(0,1) + 1,2\, y'(0,1) = 0, \\ 2y(1,1) - 2,5\, y'(1,1) = -4. \end{cases}$$

У цій крайовій задачі $c1 = \alpha_0 = 1$, $c2 = \alpha_2 = 1,2$, c=A=0, $d1 = \beta_0 = 2$, $d2 = \beta_1 = -2,5$, d=B=2,15; вузлові точки мають абсциси $x_i = 0.1 + h*i$; коефіцієнти $p_i = e^x$, $q_i = x_i / 2$; $f_i = x^2$, (i = 0, 1, 2,…, 10).

Тоді складемо програму для розв'язання цієї крайової задачі в Matlab за алгоритмом, що наведений вище:

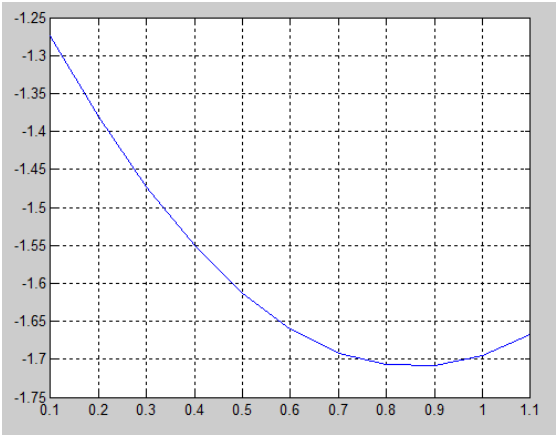
```
N=10;
p=inline('exp(x)');
q=inline('x/2');
f=inline('x^2');
 a=0.1;
 b=1.1;
 c1=1;
 c2=-1.2;
 c=0;
 d1=2;
 d2=-2.5;
 d=-4;
 h=(b-a)/N;
 r1=h^2;
 r2=h/2;
```

```
 p1(1)=-c2/(c1*h-c2);
 for i=1:N
   i1=2*i;
   q1(1)=-c*h*p1(1)/c2;
   x=a;
   for i=2:N
        i1=i-1;
        x=x+h;
        t=1-p(x)*r2;
        p1(i)=(t-2)/(q(x)*r1+t*p1(i1)-2);
        q1(i)=(f(x)*r1-t*q1(i1))*p1(i)/(t-2);
   end;
end;
 p1(N+1)=0;
 q1(N+1)=(d*h+d2*q1(N))/(d1*h+d2-d2*p1(N));
 y(N+1)=q1(N+1);
for j=N:-1:1
   y(j)=p1(j)*y(j+1)+q1(j);
   for j=1:(N+1)
   x(j)=a+(j-1)*h;
   end;
end;
x=x';
y=y';
[x,y]
plot(x,y)
grid on
```

У результаті виконання програми отримаємо:

```
ans =
    0.1000   -1.2746
    0.2000   -1.3809
    0.3000   -1.4732
    0.4000   -1.5509
    0.5000   -1.6135
    0.6000   -1.6604
    0.7000   -1.6917
    0.8000   -1.7075
    0.9000   -1.7081
    1.0000   -1.6946
```

```
1.1000   -1.6680
```



8.3. Програма роботи

1. Ознайомитися з методом скінченних різниць та методом прогону для чисельного розв'язування крайової задачі для ЛДР.
2. Розв'язати крайову задачу для ЛДР методом прогону із заданою точністю. Оцінити похибку.
3. Зробити висновки та оформити звіт.

8.4. Порядок виконання роботи

1. Прочитати теоретичні відомості.
2. Вибрати завдання згідно варіанту (за номером у журналі групи або за вказівкою викладача).
3. Завантажити середовище MATLAB.
4. ***Завдання***. Використовуючи метод прогону, знайти розв’язок крайової задачі для звичайного диференціального рівняння з точністю $\varepsilon = 10^{-4}$; крок $h = 0,05$.
6. Зробити висновки. Оформити звіт про виконання роботи.

8.5. Варіанти завдань

Таблиця 8.1

Варіант	Крайова задача
1.	. $y'' + xy' + 2y = x + 1,$ $\begin{cases} y(0,9) - 0,5y'(0,9) = 2, \\ y(1,2) = 1. \end{cases}$
2.	$y'' + xy' + y = x + 1,$ $\begin{cases} y(0,5) + 2y'(0,5) = 1, \\ y'(0,8) = 1,2. \end{cases}$
3.	$y'' + 2y' - \frac{y}{x} = 3,$ $\begin{cases} y(0,2) = 2, \\ 0,5y(0,5) - y(0,5) = 1. \end{cases}$
4.	$y'' + 2y' - \frac{y}{x} = \frac{1}{x},$ $\begin{cases} 0,5y(0,9) + y'(0,9) = 1,5, \\ y(1,2) = 0,8. \end{cases}$
5.	$y'' - y' + \frac{2y}{x} = x + 0,4,$ $\begin{cases} y(1,1) - 0,5y'(1,1) = 2, \\ y'(1,4) = 4. \end{cases}$
6.	$y'' - \frac{y'}{2} + 3y = 2x^2,$ $\begin{cases} y(1) + 2y'(1) = 0,6, \\ y(1,3) = 1. \end{cases}$
7.	$y'' - 3xy' + 2y = 1,5,$ $\begin{cases} y'(0,7) = 0,5, \\ 0,5y(1) + y'(1) = 2. \end{cases}$
8.	$y'' - 3y' + y / x = 1,$ $\begin{cases} y(0,4) = 2, \\ y(0,7) + 2y'(0,7) = 0,7. \end{cases}$

9.	$y'' + 2xy' - 2y = 0,6$, $\begin{cases} y(2) = 1, \\ 0,4y(2,3) - y'(2,3) = 1. \end{cases}$
10.	$y'' + \frac{y'}{2x} + 0,8y = x$, $\begin{cases} y(1,7) + 1,2y'(1,7) = 2, \\ y'(2) = 1. \end{cases}$
11.	$y'' + 2xy' - y = 0,4$, $\begin{cases} 2y(0,3) + y'(0,3) = 1, \\ y'(0,6) = 2. \end{cases}$
12.	$y'' - 0,8y' - xy = 1,4$, $\begin{cases} y(1,8) = 0,5, \\ 2y(2,1) + y'(2,1) = 1,7. \end{cases}$
13.	$y'' - 0,5xy' + y = 2$, $\begin{cases} y(0,4) = 1,2, \\ y(0,7) + 2y'(0,7) = 1,4. \end{cases}$
14.	$y'' - \frac{y'}{4} + \frac{2y}{x} = \frac{x}{2}$, $\begin{cases} 1,5y(1,3) - y'(1,3) = 0,6, \\ 2y(1,6) = 0,3. \end{cases}$
15.	$y'' + 0,5y' + 0,5xy = 2x$, $\begin{cases} y'(1) = 0,5, \\ 2y(1,3) - y'(1,3) = 2. \end{cases}$

8.5. Контрольні запитання

1. Що таке крайова задача?
2. Які крайові умови називають однорідними?
3. Запишіть загальний вигляд крайової задачі для ЛДР другого порядку.
4. Яку крайову задачу називають двоточковою?
5. У чому полягає метод прогону?

Лабораторна робота №9
Розв'язання крайової задачі для одновимірного рівняння теплопровідності різницевим методом

9.1. Мета роботи

Практичне оволодіння різницевим методом Розв'язання крайової задачі для одновимірного рівняння теплопровідності. Навчитися розв'язувати поставлені задачі. Ознайомитися з методикою застосування програмних продуктів для побудови функцій на основі експериментальних даних.

9.2. Теоретичні відомості. Реалізація методу в пакеті MATLAB

Розглянемо наступну модельну задачу. При значеннях:

$$\begin{aligned} &f(x,t) = (2 - x(1-x))\exp(-t), \\ &U0(x) = x(1-x), \\ &l = 1, c_1 = 1, d_1 = 0, \phi_1(t) = 0, \\ &c_2 = 0, d_2 = 1, \phi_2(t) = -\exp(t), \end{aligned} \tag{9.1}$$

задача має наступний розв'язок $U(x,t) = x(1-x)\exp(-t)$.

Введемо рівномірну сітку за змінними *x, t* :

$$\begin{aligned} &\overline{\omega} = \overline{\omega}_h \times \overline{\omega}_\tau, \\ &\overline{\omega}_h = \{x_i = (i-1)N, i = \overline{1,N}, h = 1/(N-1)\}, \\ &\overline{\omega}_\tau = \{t_j = j\tau, j = \overline{0,M}, \tau > 0\}. \end{aligned}$$

Виконаємо на шаблоні $S\{(x_{i-1},t_j), (x_i,t_j), (x_{i+1},t_j);(x_i,t_{j+1})\}$ безпосередню заміну диференціальних операторів на різницеві, застосуємо метод *"законтурних точок"* для апроксимації з порядком $O(h^2)$ крайової умови в точці $x=1$.

Отримаємо явну різницеву схему з порядком $O(\tau+h^2)$:

$$\begin{aligned} &y_i^0 = U0(x_i), i = \overline{1,N}; \\ &y_1^{j+1} = \phi_1(t_{j+1}); \\ &y_i^{j+1} = y_i^j + \gamma(y_{i-1}^j - 2y_i^j + y_{i+1}^j) + \tau f_{i,j+0.5}, \quad i = 2,3,\ldots,N-1; \quad \gamma = \tau h^{-2}; \\ &y_N^{j+1} = y_N^j + 2\gamma(y_{N-1}^j - y_N^j) + D\phi_2(t_j) + \tau f_{N,j+0.5}, \quad D = 2\gamma h; \\ &j = \overline{0,M-1}. \end{aligned} \tag{9.2}$$

При використанні схеми (9.2) необхідно враховувати умову стійкості $\tau \le 0.5h^2$.

Використовуючи шаблон $S\{(x_{i-1}, t_j), (x_i, t_j), (x_{i+1}, t_j); (x_{i-1}, t_{j+1}), (x_i, t_{j+1}), (x_{i+1}, t_{j+1})\}$, і числовий параметр $0 < \sigma \le 1$ можна отримати неявну різницеву схему (порядок апроксимації залежить від σ):

$$y_i^0 = U0(x_i), i = 1,2,\ldots,N,$$
$$\begin{cases} A_i y_{i-1}^{j+1} - C_i y_i^{j+1} + B_i y_{i+1}^{j+1} = -F_i, \\ i = 1,2,\ldots,N;\ j = 0,1,\ldots,M-1. \end{cases} \qquad (9.3)$$

Тут введено наступні позначення:

$$\begin{aligned} i=1: \quad & A_1 = 0, \quad C_1 = 1, \quad B_1 = 0, \quad F_1 = \phi(t_{j+1}); \\ i = \overline{2, N-1}: \quad & A_i = \sigma\gamma, \quad C_i = 1 + 2\sigma\gamma, \quad B_i = A_i, \\ & F_i = y_i^j + (1-\sigma)\gamma\left(y_{i-1}^j - 2y_i^j + y_{i+1}^j\right) + \tau f_{i,j+0.5}; \\ i = N: \quad & A_N = 2\sigma\gamma,\ C_N = 1 + 2\sigma\gamma,\ B_N = 0, \\ & F_N = y_N^j + 2(1-\sigma)\gamma\left(y_{N-1}^j - y_N^j\right) + D_1\phi_2(t_j) + \tau f_{N,j+0.5} + D_2\phi_2(t_{j+1}), \\ & D_1 = 2(1-\sigma)\gamma h, \quad D_2 = 2\sigma\gamma h. \end{aligned}$$

Для кожного значення j в (9.2) отримали СЛАР з тридіагональною матрицею, яку ефективно розв'язуємо методом монотонної прогонки m_progm.

Для модельної задачі (9.4), (9.1) нижче наводиться повний алгоритм розв'язання (lr60main – явна схема, lr6smain – неявна схема), а також порівняння наближеного і точного розв'язку.

```
% lr60main.m
% Лабораторна робота №9 з ЧМ
% Розв'язування крайової задачі для одновимірного
% рівняння параболічного типу
% різницевим методом (явна схема)
clc
clear
% параметри модельної задачі :
l=1; c1=1; d1=0; c2=0; d2=1;
% параметри різницевої схеми :
N=51; N1=N-1; h=l/N1; x=[0:h:l]; x(N)=l;
tau=2e-4; M=25000;
% обчислення додаткових констант (з урахуванням
крайових умов) :
gam=tau./(h.*h); D2=2.*h./d2;
% формування початкових умов при t=0 :
for k=1:N
```

```
Y(k)=lr6_u0(x(k));
end
Y0=Y;  % запам'ятовуємо для побудови графіку
розв'язку в t=0
for j=1:M
% знаходження наближеного розв'язку Y(:) на
черговому шарі t=t(j) :
t=(j-1).*tau;
t1=t+0.5.*tau;
yc=Y(1);
Y(1)=lr6_fi1(t+tau);
for k=2:N1
yl=yc; yc=Y(k);
Y(k)=yc+gam.*(yl-
2.*yc+Y(k+1))+tau.*lr6_f(x(k),t1);
end
yl=yc; yc=Y(N); yr=yl+D2.*lr6_fi2(t);
Y(N)=yc+gam.*(yl-2.*yc+yr)+tau.*lr6_f(l,t1);
if j*2==M
Y1=Y;  % запам'ятовуємо для побудови графіку
розв'язку в t=M*tau/2
end
end
% побудова графіків розв'язку при t=0, M*tau/2,
M*tau :
t=M.*tau; t1=t./2; T1=num2str(t1); T=num2str(t);
plot(x,Y0,'b.-',x,Y1,'k.--',x,Y,'r*-');
xlabel('x');
ylabel('u');
legend('u(x,0)',strcat('u(x,',T1,')'),strcat('u(x,
',T,')'),'Location','Best');
pause
% порівняння точного розв'язку U(x,t)=x(1-x)exp(-
t) і наближеного Y(:)
% в точці t=M*tau :
u=@(x,t)(x.*(1-x).*exp(-t));
U=u(x,t);
plot(x,Y,'b.-',x,U,'r*-');
xlabel('x');
ylabel('u');
```

```
legend(strcat('u(x,',T,')'),strcat('U(x,',T,')'),'
Location','Best');
norm(Y-U,inf)
```
% lr6smain.m
```
% Лабораторна робота №9 з ЧМ
% Розв'язування крайової задачі для одновимірного
% рівняння параболічного типу
% різницевим методом (неявна схема)
clc
clear
% параметри задачі :
l=1; c1=1; d1=0; c2=0; d2=1;
% параметри різницевої схеми :
N=51; N1=N-1; h=l/N1; x=[0:h:l]; x(N)=l;
tau=1e-2; M=500;
sigma=0.5;
% обчислення додаткових констант (з урахуванням
крайових умов) :
gam=tau./(h.*h); gs=gam.*sigma; gs2=2.*gs; s1=1-
sigma; gs1=gam.*s1;
ck=1+gs2; ckj=1-2.*gs1; D2=2.*h./d2; D2gs=gs.*D2;
% формування початкових умов при t=0 і матриці
СЛАР :
for k=1:N
Y(k)=lr6_u0(x(k));
A(k)=gs; C(k)=ck; B(k)=gs;
end
A(1)=0; C(1)=1; B(1)=0; % враховуємо крайові умови
при x=0
A(N)=gs2; B(N)=0; % враховуємо крайові умови при
x=l
Y0=Y; % запам'ятовуємо для побудови графіку
розв'язку в t=0
for j=1:M
% знаходження наближеного розв'язку Y(:) на
черговому шарі t=t(j) :
t=j.*tau;
t1=t-0.5.*tau;
% формування правої частини СЛАР :
F(1)=lr6_fi1(t);
```

```
for k=2:N1
F(k)=gs1.*(Y(k-
1)+Y(k+1))+ckj.*Y(k)+tau.*lr6_f(x(k),t1);
end
F(N)=gs1.*(2.*Y(N-1)+D2.*lr6_fi2(ttau))+
ckj.*Y(N)+tau.*lr6_f(l,t1)+D2gs.*lr6_fi2(t);
% розв'язок СЛАР, знаходження Y(:) :
[Y,alfa,err]=m_progm(A,C,B,F);
if j*2==M
Y1=Y;  % запам'ятовуємо для побудови графіку
розв'язку в t=M*tau/2
end
end
% побудова графіків розв'язку при t=0, M*tau/2,
M*tau :
t=M.*tau; t1=t./2; T1=num2str(t1); T=num2str(t);
plot(x,Y0,'b.-',x,Y1,'k.--',x,Y,'r*-');
xlabel('x');
ylabel('u');
legend('u(x,0)',strcat('u(x,',T1,')'),strcat('u(x,
',T,')'),'Location','Best');
pause
% порівняння точного розв'язку U(x,t)=x(1-x)exp(-
t) і наближеного Y(:)
% в точці t=M*tau :
u=@(x,t)(x.*(1-x).*exp(-t));
U=u(x,t);
plot(x,Y,'b.-',x,U,'r*-');
xlabel('x');
ylabel('u');
legend(strcat('u(x,',T,')'),strcat('U(x,',T,')'),'
Location','Best');
norm(Y-U,inf)
function f=lr6_f(x,t)
% Функція f(x,t)
f=(2-x.*(1-x)).*exp(-t);
return
function u=lr6_u0(x)
% Функція u0(x)
u=x.*(1-x);
```

```
return
function fi1=lr6_fi1(t)
% Функція fi1(t)
fi1=0;
return
function fi2=lr6_fi2(t)
% Функція fi2(t)
fi2=-exp(-t);
  return
```

На рис. 9.1 – 9.2 наведено графіки отриманих розв'язків для різних моментів часу.

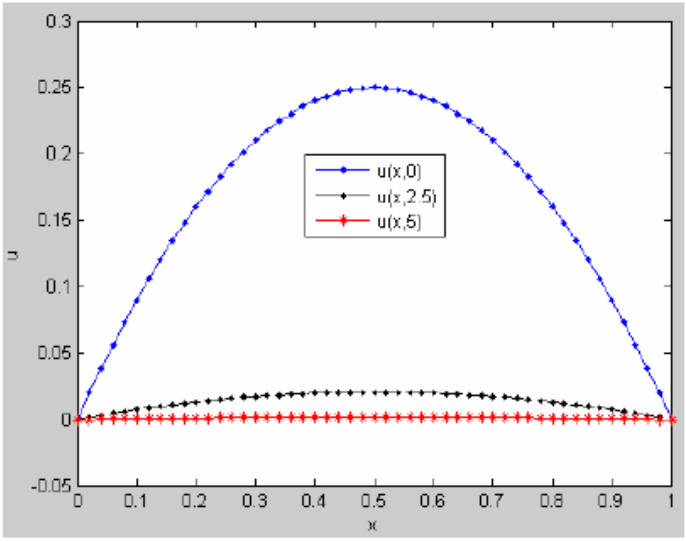


Рис. 9.1. Графік наближеного розв'язку *U(x,t)* для різних значень *t*.

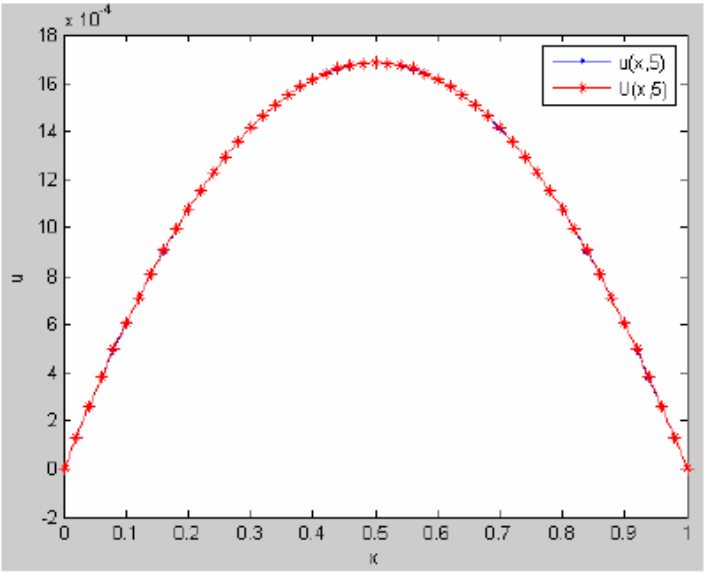


Рис. 9.2. Графік наближеного розв'язку *u(x,t)* і точного *U(x,t)* для *t=5*.

9.3. Програма роботи

1. Ознайомитися з методом рішення крайової задачі для одновимірного рівняння теплопровідності різницевим методом.
2. Розв'язати СЛАР засобами Matlab згідно заданого варіанту.
3. Зробити висновки та оформити звіт.

9.4. Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями.
2. Вибрати завдання згідно варіанту (за номером у журналі підгрупи або за вказівкою викладача).
3. Завантажити середовище MATLAB. Розв'язати завдання згідно варіанту.
4. Зробити висновки. Оформити звіт про виконання роботи.

9.5. Завдання

Використовуючи різницевий сітковий метод розв'язати крайову задачу.

$$\begin{cases} \dfrac{\partial U(x,t)}{\partial t} = \dfrac{\partial^2 U(x,t)}{\partial x^2} + f(x,t), \quad 0 < x < l, \quad 0 < t < T; \\ U(x,0) = U0(x), \quad 0 \le x \le l; \\ c_1 U(0,t) + d_1 \dfrac{\partial U(0,t)}{\partial x} = \varphi_1(t), \quad c_2 U(l,t) + d_2 \dfrac{\partial U(l,t)}{\partial x} = \varphi_2(t), \quad 0 \le t < T. \end{cases} \tag{9.4}$$

Таблиця 9.1

k	$f(x,t)$	l	$U0(x)$	c_1	d_1	$\varphi_1(t)$	c_2	d_2	$\varphi_2(t)$
1	$(1-x^g)/(1+t)$	1	$0.5x^2 d$	0	1	0	0	1	d
2	$(1-x)/(1+t^{1+g})$	2	$2x$	1	0	0	2	1	10
3	$-2xt(1-x)$	1	$3\sin(7\pi x)$	1	0	0	1	0	0
4	$-U+x+tx^2$	1	0	1	0	0	1	0	t
5	$-U+(4x+6)x^2$	1	3	0	1	0	0	1	0
6	-6	3	$3x^2(3-x)^2$	0	1	0	0	1	0
7	$-U-\cos(2.5\pi x)$	1	x^2-1	0	1	0	1	0	0
8	$(1-x/2)(\cos t - x\sin t)$,	2	$\sin(\pi x)$	1	0	$\sin t$	1	0	0
9	$-U$	1	$0.5x^2 d$	0	1	0	0	1	d
10	$0.1\pi\cos 0.25\pi x$	2	0	0	1	0	1	0	0
11	$-Ud$	1	$-36.6\cos^2(\pi x)$	0	1	0	0	1	0
12	$-U+d\exp(-t)$	1	$x(1-x)$	1	1	0	1	0	0
13	$-x(2-x)\sin t$	2	$3\sin(3\pi x)$	1	0	0	1	0	0
14	$-Ud$	1	$2x(1-x)$	1	0	0	1	0	0
15	$\left(x^2-4\right)\sin t$,	2	$4\cos(0.75\pi x)$	0	1	0	1	0	0
16	$-4U$	3	x^2	0	1	0	0	1	6
17	$(1-x^g)/(1+t)$	1	0	-1	1	$5t$	1	0	$5t$
18	$(1-x)/(1+t^{1+g})$	1	$x^3 d/3$	0	1	0	0	1	d
19	$0.5x^2$	1	0	0	1	0	0	1	t
20	$-2(U-t)$	2	0	0	1	0	0	1	0
21	$-Ud$	1	$x-1$	-2	1	3	1	0	0
22	$-6U$	1	$1-\cos(2\pi x)$	1	0	0	d	1	0
23	$-U-xt$	1	0	1	1	0	1	1	0
24	$-25U$	1	$2\cos x+\sin x$	-0.5	1	0	1	0	0
25	$-U+\mathrm{sh}\,t$	1	1	0	1	0	1	0	0

9.6. Контрольні запитання

1. Сформулюйте задачу Коші і крайову задачу. В чому відмінність у постановці цих задач?
2. Наведіть приклади звичайних диференціальних рівнянь, які розв’язуються лише чисельними методами.
3. Дайте порівняльну оцінку і визначте області застосування методів розв’язання звичайних диференціальних рівнянь.
4. Чи можлива крайова задача для рівнянь першого порядку ?
5. Що таке похибка зрізання (обмеження) ? Чим визначається порядок похибки методу ?

Лабораторна робота №10
Методи безумовної оптимізації

5.1. Мета роботи

Вивчити основи методу безумовної оптимізації. Навчитися розв'язувати задачу використовуючи набуті знання. Ознайомитися з методикою застосування програмних продуктів для побудови функцій на основі експериментальних даних.

10.2. Теоретичні відомості

10.2.1. Загальні відомості

Оптимізацією називають процес вибору найкращого варіанту з усіх можливих. При інженерних розрахунках оптимізація дозволяє вибрати найкращий варіант конструкції, найкращий розподіл ресурсів тощо.

Оптимізація здійснюється завдяки зміні деяких параметрів задачі, які називають параметрами оптимізації. Число цих параметрів є розмірністю задачі оптимізації. Оптимальний розв'язок знаходять за допомогою спеціальної функції від параметрів оптимізації, яку називають функцією мети, або критерієм якості. В процесі оптимізації функція мети досягає екстремального значення, тобто максимального чи мінімального. Відповідні значення параметрів оптимізації є розв'язком задачі. До речі, максимум функції мети співпадає з мінімумом функції мети з оберненим знаком.

Отже, **математична задача оптимізації** полягає в мінімізації функції мети $f=f(x_1, x_2,..., x_n)$ за n-вимірним вектором параметрів оптимізації
$x=(x_1, x_2,..., x_n)$ у межах області Ω:

$$x_{opt} = (x_1, x_2,..., x_n)_{opt} = \arg \min_{(x_1, x_2,..., x_n)\in\Omega} f(x_1, x_2,..., x_n).$$

Є два типи задач оптимізації – **безумовні та умовні**.

Задача (7.1) є безумовною. Відповідний розв'язок x_{opt} називають **глобальним мінімумом**. Прикладом безумовної оптимізації є розв'язування СЛАР за методом найменших квадратів (2.5). Тут параметрами оптимізації є складові вектора $x=(x_1, x_2)$, функція мети

$$f(x_1, x_2)= \sum_{i=1}^{3} (a_{i1}x_1 + a_{i2}x_2 - b_i)^2 ,$$

а область Ω – це вся область дійсних чисел.

Складнішими є умовні задачі, де на параметри накладено **обмеження**
у вигляді рівнянь

$g_i(x_1, x_2,..., x_n)=0; \quad i=\overline{1,m}$

і (або) нерівностей

$a_i \le \varphi_i(x_1, x_2,..., x_n) \le b_i; \quad i=\overline{1,k}$.

Обмеження загалом звужують область оптимізації. Задачу (7.1)-(7.3) називають умовною задачею оптимізації, а її розв'язок – **локальним мінімумом**.

Наведемо просту задачу умовної оптимізації.

Треба спроектувати контейнер у формі прямокутного паралепіпеда об'ємом 1м^3, причому з мінімальними витратами матеріалу. За незмінної товщини стінок остання умова означає мінімальну поверхню контейнера. Якщо позначити x_1, x_2, x_3 довжини ребер контейнера, то задача полягає у мінімізації функції мети $f=2(x_1x_2+x_1x_3+x_2x_3)$ за трьома параметрами x_1, x_2, x_3 з обмеженням $x_1x_2x_3=1$.

Рівняння-обмеження можна використати для виключення одного параметра: $x_3=1/x_1x_2$; $f=2(x_1x_2+1/x_1+1/x_2)$. Таким чином, задача стала безумовною з двома параметрами. Її розв'язок є сумісним розв'язком двох рівнянь з двома невідомими: $\partial f/\partial x_1=2(x_2-1/x_1^2)=0$; $\partial f/\partial x_2=2(x_1-1/x_2^2)=0$. Майже очевидний їх розв'язок: $x_1=x_2=1$. З рівняння $x_3=1/x_1x_2$ маємо $x_3=1$. Отже, шуканий контейнер має форму куба з ребрами 1м.

Вивчає та розв'язує задачі умовної оптимізації важливий розділ прикладної математики – **математичне програмування**.

Розгляд методів оптимізації почнемо з найпростіших: безумовних одновимірних задач, тобто задач **мінімізації функції одного аргументу** $f(x)$, якщо x лежить на відрізку $[a,b]$ ($a \le x \le b$).

Шукати екстремальні точки функції $f(x)$ легко, якщо на відрізку $[a,b]$ існує похідна $df(x)/dx$. Тоді екстремум знаходиться або на границі відрізка $[a,b]$, або у точках, що є розв'язками рівняння $df(x)/dx=0$.

Розглянемо приклад. Нехай треба знайти екстремуми функції $f(x)=x^3/3-x^2$ на відрізку [1,3]. Похідна $df(x)/dx=x^2-2x$. Розв'язки рівняння $x^2-2x=0$ такі: $x_1=0$; $x_2=2$. Але точка $x_1=0$ є поза відрізком [1,3]. Отже, екстремальними можуть бути точки: $a=1$; $x_2=2$; $b=3$. Значення функції $f(x)$ у цих точках такі: $f(1)=-2/3$; $f(2)=-4/3$; $f(3)=0$. Простим порівнянням значень функції знаходимо, що $f_{min}=f(2)=-4/3$, а

$f_{max}=f(3)=0$. На рис. 10.1 показано графік функції та екстремальні точки.

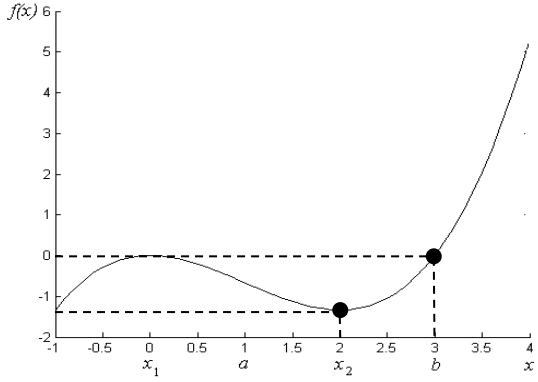


Рис. 10.1. Графік функції $f(x)=x^3/3-x^2$ з точками екстремумів на відрізку $[a,b]$

Однак обчислити похідну можна не завжди. Часто функція мети задана лише сукупністю значень в певних точках. Можливий вихід – інтерполювати функцію мети степеневим поліномом або сплайном, для яких легко обчислити похідну. Недолік відповідних **інтерполяційних методів** – у появі похибок інтерполяції.

Загальна ідея **числових пошукових методів** полягає у послідовному звуженні інтервалу з екстремумом, поки розмір інтервалу не відповідатиме заданій точності ε.

Прямий перебір полягає у поділі відрізка $[a,b]$ на $n=2(b-a)/\varepsilon$ частин, обчисленні значень функції $f(x)$ у кожній точці поділу та порівнянням значень виділити точки з найменшим значенням. Метод гранично простий, але вимагає обчислень значень функції $(n+1)$ раз. Нехай довжина відрізка $[a,b]$ рівна 1, а $\varepsilon=0.001$. Тоді значення функції треба обчислити 2001 раз.

Деяке ускладнення методу прямого перебору значно зменшує необхідні обчислення. Поділимо відрізок $[a,b]$ не на 2000, а на 20 частин і знайдемо точку з найменшим значенням функції (21 обчислення функції). Далі відрізок, утворений сусідніми до екстремальної точками, поділимо ще на 20 частин і знайдемо точку з мінімумом функції вже у межах цього відрізка (ще 19 обчислень функції). Наступне наближення забезпечує задану точність

знаходження екстремуму, а загальна кількість обчислень функції всього 59 проти 2001 при прямому переборі.

Метод „золотого січення” є найощадливішим методом глобальної оптимізації функцій однієї змінної. Тут вповні реалізовано основну ідею пошукових методів – побудова послідовності вкладених відрізків, що стягуються до точки мінімума, причому для кожного наступного відрізка значення функції обчислюється лише один раз. Це можливо завдяки простому правилу поділу відрізка на більшу і меншу частини за принципом „золотого січення”: відношення довжини відрізка до його більшої частини дорівнює відношенню більшої частини до меншої (дивись рис. 10.2).

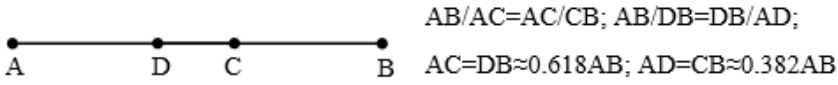


Рис. 10.2. Правило „золотого січення”

Це правило вперше зустрічається в працях давньогрецького математика і філософа Евкліда (Ενκλειδηζ, Euclid, 3-тє ст. до н. е.), а відоме було ще Піфагору (Πυθαγορασ, Pythagoras, 6-те ст. до н. е.). Назву „золоте січення” дав знаменитий італійський митець, інженер і філософ кінця 15-го – початку 16-го сторіч Леонардо да Вінчі (Leonardo da Vinci). З його легкої руки багато наступних митців та дослідників вбачали в „золотому січенні” фундаментальне співвідношення живої та неживої природи, хоча об’єктивних підстав для цього вкрай мало.

Пошук мінімума функції починається „золотим січенням” початкового відрізка на частини, як показано на рис. 21. Серед значень функції у чотирьох точках знаходимо мінімальне і наступний відрізок утворюють точка мінімуму та ще дві найближчі до неї точки. Для „золотого січення” цього відрізка треба знайти лише одну нову точку. Саме тому кількість обчислень є мінімальною. Пошук триває, поки довжина чергового відрізка не стане меншою заданої точності.

Практичні методи одновимірної мінімізації є, як правило, **комбінованими**, тобто поєднують, наприклад, **інтерполяційний та метод „золотого січення”**.

Значно складнішими є **методи мінімізації функції багатьох аргументів**. Якщо обмежень немає, то мінімум функції $f=f(x_1, x_2,..., x_n)$ знаходять серед точок, що є розв’язками системи рівнянь

$\partial f(x_1, x_2,..., x_n)/\partial x_i=0$, $(i=1,...,n)$. Приклад такої мінімізації показано трьома сторінками раніше при пошуку паралелепіпеда з найменшою поверхнею. Але далеко не завжди можна легко обчислити похідні функції мети. Тоді застосовують числові пошукові методи.

Одразу можна відкинути методи перебору, які є неприпустимо громіздкими для багатовимірних задач мінімізації. Необхідні спеціальні числові методи цілеспрямованого пошуку.

Порівняно простим є **метод покоординатного спуску**. Пошук мінімума функції мети $f=f(x_1, x_2,..., x_n)$ починаємо з початкової точки з координатами $x_1^{(0)}, x_2^{(0)},..., x_n^{(0)}$. Зафіксуємо у цій точці всі координати, крім першої. Тоді $f=f(x_1,x_2^{(0)},..., x_n^{(0)})$ є функцією лише однієї змінної x_1, і задача мінімізації розв'язується методами, що розглянуті вище. Знайдена точка мінімума з координатами $x_1^{(1)}, x_2^{(0)},..., x_n^{(0)}$ є початковою для наступного кроку, де мінімізується функція мети $f=f(x_1^{(1)}, x_2, x_3^{(0)},..., x_n^{(0)})$ за другою змінною. Послідовність кроків продовжується, поки задана точність не досягнена. Метод є послідовністю одновимірних задач оптимізації, але існує клас задач, які він принципово не може розв'язати. Часто графічне зображення функції мети має яроподібний вигляд, що у двовимірному випадку нагадує вузький яр зі стрімкими стінками і пологим дном. Метод покоординатного спуску швидко досягає дна яру, але подальший рух до мінімума за його дном різко сповільнюється або й зовсім зупиняється. Математично це означає, що функція мети досягла мінімума за кожною змінною зокрема, але це ще не є мінімум за всіма змінними одночасно.

Успішно справляється з яроподібними задачами **метод градієнтного спуску**. В природі є чимало оптимізаційних задач. Наприклад, вода стікає на дно ями за найкоротшими траєкторіями, що відповідають найбільшій крутизні стінок ями. Ця крутизна математично виражається поняттям градієнта. Градієнтом функції $f=f(x_1, x_2,..., x_n)$ у точці $(x_1^{(0)}, x_2^{(0)},..., x_n^{(0)})$ є вектор, складові якого є частинними похідними функції у заданій точці:

$$\operatorname{grad}_{x_1^{(0)}, x_2^{(0)},..., x_n^{(0)}} f(x_1, x_2,..., x_n)=\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2},..., \frac{\partial f}{\partial x_n}\right)_{x_1^{(0)}, x_2^{(0)},..., x_n^{(0)}}.$$

Вектор градієнту показує напрям найбільшого зростання функції. Отже, якщо зробити крок у напрямі антиградієнта, то функція загалом зменшується. Величину кроку визначають спеціальні адаптивні алгоритми. Нова точка $(x_1^{(1)}, x_2^{(1)},..., x_n^{(1)})$, у якій значення функції мети

зменшилось, є початковою для наступного кроку. У цій точці знову обчислюють градієнт

$$\underset{x_1^{(1)}, x_2^{(1)}, ..., x_n^{(1)}}{\text{grad}} f(x_1, x_2, ..., x_n) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\right)_{x_1^{(1)}, x_2^{(1)}, ..., x_n^{(1)}} \quad (7.5)$$

та здійснюють крок у протилежному напрямку. Процес пошуку неухильно наближується до точки мінімума, де всі складові градієнта рівні нулю. Недоліком градієнтного спуску є обчислення градієнта (7.5) на кожному кроці. Аналітичне диференціювання функції мети можливе у виключних випадках. Як правило, складові градієнта знаходять наближено:

$$\frac{\partial f}{\partial x_i} \approx \frac{1}{\Delta x_i}(f(x_1, ..., x_i + \Delta x_i, ..., x_n) - f(x_1, ..., x_i, ..., x_n)); \quad i = \overline{1, n}. \quad (7.6)$$

Обчислення градієнта на кожному кроці обтяжливе. **Метод найшвидшого спуску**, вперше запропонований знаменитим радянським математиком і економістом Л. В. Канторовичем, є поширеною модифікацією градієнтного метода. За методом найшвидшого спуску рух у напрямі антиградієнта триває не один крок, а поки функція мети зменшується. Коли мінімум досягнено, рух продовжується за новим напрямом, що вказує обчислений антиградієнт у цій точці. Таким чином, точок обчислення градієнта значно менше. По суті, метод найшвидшого спуску поєднує методи покоординатного і градієнтного спуску. Мінімізація функції мети за антиградієнтом є одновимірною задачею мінімізації вздовж напряма антиградієта, а в методі покоординатного спуску напрям співпадає по черзі з осями координат.

10.2.2. Оптимізація функції однієї змінної у системі MATLAB

У системі MATLAB є чимало функцій, що реалізують різні методи оптимізації. Вивчимо одну з них, що шукає мінімум дійсної функції однієї дійсної змінної. Відповідна коротка MATLAB-програма наведена нижче.

```
%************************************************
**********************
% Optimization (golden section and parabolic
interpolation)
%************************************************
**********************
```

```
  clear;
  x=[-pi:0.01:4*pi];
  f=sin(x)+x/10;
  plot(x,f);  grid;
  options=optimset('display', 'iter');
  [xmin,fmin,exitflag,output]=fminbnd('sin(x)+x/10
', 0, 4*pi, options),
```

MATLAB-програма після очищення робочої області пам'яті будує графік функції мети f за значеннями аргумента в масиві x від –π до 4π з кроком 0.01 (рис. 10.3).

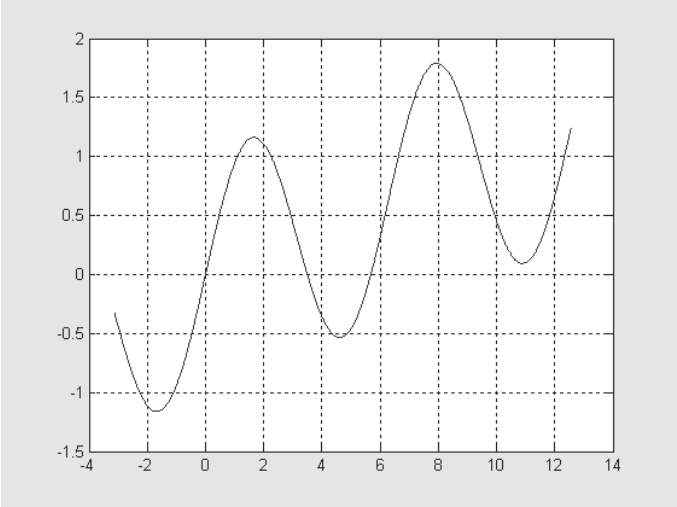


Рис. 10.3. Графік функції, мінімум якої шукає fminbnd

Далі йде звертання до функції MATLAB fminbnd. Ця функція надійно знаходить мінімум дійсної функції дійсного аргумента за комбінованим методом, що спочатку звужує інтервал пошуку методом „золотого січення", а потім уточнює рішення інтерполяційним параболічним методом.

Чотирма аргументами функції fminbnd є: стрічковий вираз, що задає функцію мети; нижня та верхня початкові межі інтервалу пошуку; масив параметрів оптимізації, заданий функцією optimset у попередньому операторі програми. Функція optimset задає вивід у

командне вікно MATLAB результатів усіх ітерацій пошуку мінімума з методом пошуку на кожному кроці (дивись таблицю 1).

Кінцеві результати пошуку виводить в командне вікно функція fminbnd в наступному порядку: значення аргумента у точці мінімума xmin; значення функції мети у точці мінімума fmin; логічна змінна результатів пошуку exitflag; підсумкові параметри пошуку output (кількість ітерацій, кількість обчислень функції мети, назва методу оптимізації).

10.3. Програма роботи

1. Ознайомитися з методом безумовної оптимізації.
2. Розв'язати задачу засобами Matlab згідно заданого варіанту.
3. Зробити висновки та оформити звіт.

10.4. Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями.
2. Вибрати завдання згідно варіанту (за номером у журналі підгрупи або за вказівкою викладача).
3. Завантажити середовище MATLAB. Розв'язати завдання згідно варіанту.
4. Зробити висновки. Оформити звіт про виконання роботи.

10.5. Завдання

1. Набрати в системі MATLAB та відлагодити текст програми оптимізації функції однієї змінної.
2. Підібрати такі значення нижньої межі початкового відрізка, за якого ітераційний процес пошуку мінімума збігається до локального чи глобального мінімума (рисунок 10.3). Відзначити кількість ітерацій в кожному випадку та перебіг ітераційного процесу (таблиця 1).
3. Пояснити всі оператори MATLAB-програми.

Нижня межа початкового відрізка пошуку мінімума є другим аргументом функції fminbnd. Залежно від значення цієї нижньої межі функція fminbnd знаходить або локальні мінімуми з координатами (10.8954, 0.0946) та (4.6122, -0.5338), або глобальний мінімум з координатами (-1.6710, -1.1621) (дивись таблицю 2).

Треба задавати різні значення другого аргумента і слідкувати за кординатами мінімума (xmin, fmin), який знаходить функція fminbnd. Достатньо зафіксувати які-небудь три значення нижньої межі, що відповідають трьом мінімумам.

Таблиця 1. Вивід результатів пошуку мінімума функції sin(x)+x/10, якщо початковий інтервал пошуку [0, 4*pi].

Func-count	x	f(x)	Procedure
1	4.79993	-0.516178	initial
2	7.76644	1.77282	golden
3	2.96652	0.470834	golden
4	4.86953	-0.500727	parabolic
5	4.55691	-0.532246	parabolic
6	3.94943	-0.327853	golden
7	4.61212	-0.533765	parabolic
8	4.61209	-0.533765	parabolic
9	4.61222	-0.533765	parabolic
10	4.68392	-0.531203	golden
11	4.61219	-0.533765	parabolic
12	4.61226	-0.533765	parabolic

```
  Optimization terminated successfully:  the
current x satisfies the termination criteria using
OPTIONS. TolX of 1.000000e-004
  xmin = 4.6122
  fmin = -0.5338
```

```
  exitflag = 1
  output =      iterations: 9       funcCount: 9
algorithm: 'golden section search, parabolic
interpolation'
```

Варіант 1

Вивід результатів пошуку мінімума функції $\cos(x)+x^2$, якщо початковий інтервал пошуку [0, 10].

Варіант 2

Вивід результатів пошуку мінімума функції $x^2-\mathrm{tg}(x+5)$, якщо початковий інтервал пошуку [0, 8].

Варіант 3

Вивід результатів пошуку мінімума функції $\sqrt{x^2}-2$, якщо початковий інтервал пошуку [2, 5].

Варіант 4

Вивід результатів пошуку мінімума функції $\cos(x+y)+x^2$, якщо початковий інтервал пошуку [1, 8].

Варіант 5

Вивід результатів пошуку мінімума функції $0,5x+\lg(x)$, якщо початковий інтервал пошуку [1, 6].

10.6. Контрольні запитання

1. В чому відмінність безумовних та умовних задач оптимізації?
2. Чому поєднують методи „золотого січення” та інтерполяційний?
3. Поясніть градієнтний метод та метод найшвидшого спуску.