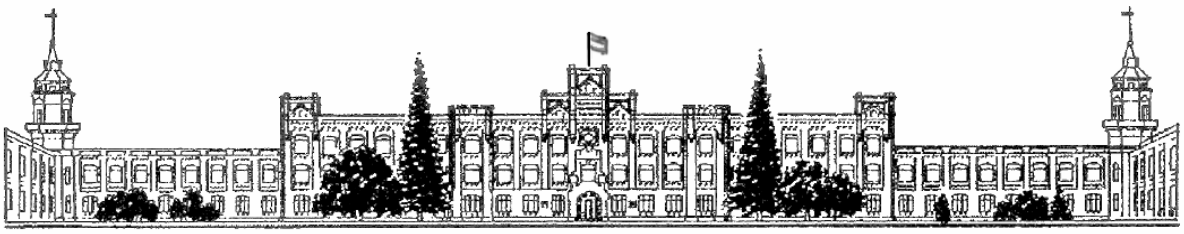
М. Р. Домбругов

ОБЧИСЛЮВАЛЬНА МАТЕМАТИКА.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧННИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ОБЧИСЛЮВАЛЬНА МАТЕМАТИКА.

КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів
спеціальності 153 – «Мікро- та наносистемна техніка»
спеціалізацій «Мікроелектронні інформаційні системи»
та «Мікро- та наноелектронні прилади і пристрої»

Київ
КПІ ім. Ігоря Сікорського
2020

Рецензент: *Бойко Ю.В.*, канд. фіз.-мат. наук, доц.

Відповідальний редактор: *Борисов О. В.*, канд. техн. наук, проф.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 9 від 24.05.2018)
за поданням Вченої ради факультету електроніки
(протокол № 05/2018 від 21.05.2018)

Електронне мережне навчальне видання

Домбругов Михайло Ремович, канд. техн. наук, доц.

ОБЧИСЛЮВАЛЬНА МАТЕМАТИКА. КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Обчислювальна математика. Комп’ютерний практикум. Електронний ресурс: навчальний посібник для студентів спеціальності 153 – «Мікро- та наносистемна техніка» спеціалізацій «Мікроелектронні інформаційні системи» та «Мікро- та наноелектронні прилади і пристрої» / М.Р.Домбругов; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 2 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2018. – 215 с.

Видання п’яте, виправлене і доповнене.

Метою практикуму є опанування методів розв’язання інженерних задач на комп’ютері, вивчення принципів побудови обчислювальних алгоритмів та особливостей їх машинної реалізації. При виконанні лабораторних робіт наголос робиться не тільки на досягненні результату обчислень і осмисленні методики його отримання, але й на набутті навичок коректної постановки задачі та усвідомленні «підводних каменів» на шляху її розв’язання.

© М. Р. Домбругов, 2018
© КПІ ім. Ігоря Сікорського, 2018

Зміст

Передмова

Курс «Обчислювальна математика» призначений для студентів природничо-наукових та інженерних спеціальностей і розрахований на 6 кредитів ECTS (приблизно 100 аудиторних годин – порівну лекційного матеріалу і лабораторних робіт). Його метою є опанування методів розв'язання інженерних задач на комп'ютері, вивчення принципів побудови обчислювальних алгоритмів та особливостей їх машинної реалізації.

Оперування сучасними процедурами для розв'язання стандартних математичних задач як «чорними скриньками» є звичайною практикою. Більшість з таких процедур досить складні, і для того, щоб вивчити їх в усіх деталях, знадобилося б значно більше часу, ніж може бути витрачено для цього більшістю студентів.

Лабораторні роботи даної збірки дозволяють студентам ознайомитися з найважливішими механізмами в алгоритмах обчислювальної математики. При виконанні робіт наголос робиться не тільки на досягненні результату обчислень і осмисленні методики його отримання, але й на набутті навичок коректної постановки задачі та усвідомленні «підводних каменів» на шляху її розв'язання.

Перше видання цього практикуму з'явилося в 1991 р. (*Абрамов И. В., Домбругов М. Р.* Численные методы. Методические указания к выполнению лабораторных работ по курсу «Вычислительная техника и программирование». – К.: КПИ, 1991. – 48 с.). Друге, суттєво перероблене видання (в електронному вигляді) пропонувалося студентам факультету електроніки НТУУ КПІ в 2004-2010 рр. Третє (доповнене, також електронне) видання вийшло українською мовою в 2011 р. Четверте (з незначними виправленнями) видання 2013 р. отримало гриф "Рекомендовано Вченою радою факультета електроніки НТУУ «КПІ»". Це видання є п'ятим, його доповнено лабораторними роботами про сингулярний розклад матриці та про кусково-поліноміальну інтерполяцію кубічними сплайнами, а також невеликим додатком, що містить опис тих елементів синтаксису мови програмування Basic, які використовуються в тексті для ілюстрації алгоритмів.

Короткий огляд практикуму

Лабораторний практикум складається з 22 робіт. Для їх виконання підходить будь-який транслятор або система програмування з мінімальною графічною бібліотекою.

Наведені схеми алгоритмів зазвичай подаються згідно зі стандартом ISO 5807:1985, за винятком позначення циклу з постійним приростом:

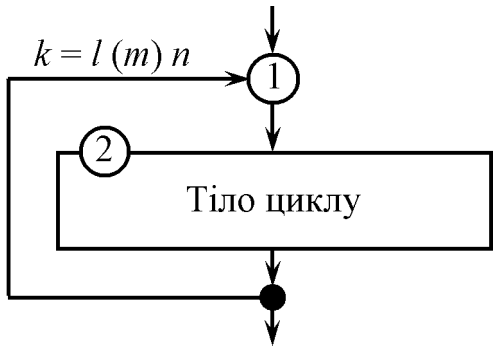


Рис. 0.1. Позначення циклу з постійним приростом
по *k* від *l* з шагом *m* до *n*

Для ілюстрації алгоритмів подекуди використовуються фрагменти програм на Basic, підмножині версії Borland Turbo Basic 1.1, які можуть бути легко адаптовані до будь-якої мови програмування. Короткий опис елементів синтаксису Basic, що використовуються в тексті, наведено в додатку в кінці книжки.

Деякі з робіт вимагають результатів попередніх робіт, або при їх виконанні можуть бути використані значні фрагменти попередньо розробленого програмного коду.

	Тематика лабораторних робіт	з попередніх робіт використовуються
№ 1.	Машинні константи	
№ 2.	Обчислення рядів	
№ 3.	Дослідження функцій	
№ 4.	Розв’язання нелінійних рівнянь з одним невідомим. Метод поділу навпіл (бісекції)	результати № 3
№ 5.	Розв’язання нелінійних рівнянь з одним невідомим. Метод простих ітерацій	результати № 3

№ 6.	Розв’язання нелінійних рівнянь з одним невідомим. Метод Ньютона-Рафсона (дотичних)	результати № 3
№ 7.	Дослідження неявно заданих функцій	фрагмент коду з № 4, № 5 або № 6 в залежності від варіанту
№ 8.	Розв’язання систем лінійних алгебраїчних рівнянь. Метод Гауса	
№ 9.	Розв’язання систем лінійних алгебраїчних рівнянь. Ітераційні методи Якобі та Гауса-Зейделя	
№ 10.	Власні вектори і власні числа матриць з елементами-дійсними числами	
№ 11.	Знаходження власних векторів і власних чисел симетричних матриць. Метод обертань Якобі	
№ 12.	Сингулярний розклад матриці	фрагмент коду з № 10
№ 13.	Оптимізація функцій однієї змінної методом Золотого перетину	результати № 3
№ 14.	Оптимізація функцій кількох змінних методом Хука-Дживса	
№ 15.	Інтерполяція даних. Рівномірне (чебишовське) наближення функцій	результати і фрагмент коду з № 3
№ 16	Кусково-поліноміальна інтерполяція. Кубічні сплайни	результати № 3 та фрагменти коду з № 3 та № 15
№ 17.	Апроксимація функціональних залежностей методом найменших квадратів	результати № 3 та фрагменти коду з № 3 та № 8
№ 18.	Чисельне інтегрування. Формули прямокутників, трапецій, Сімпсона	результати № 3
№ 19.	Кратні інтеграли. Чисельне інтегрування методом Монте-Карло	
№ 20.	Звичайні диференціальні рівняння. Задача Коші	
№ 21.	Системи звичайних диференціальних рівнянь	
№ 22.	1-вимірне рівняння Шредінгера. Розв’язання методом скінченних різниць	фрагмент коду з № 11

Загальні рекомендації до виконання лабораторних робіт

При підготовці до виконання роботи

- Вивчіть теоретичні відомості;
- Ознайомтеся зі змістом завдання, що пропонується;
- Уясніть зміст основного алгоритму і модифікацій, що пропонуються;
- Складіть програму, що реалізує запропонований алгоритм. Забезпечте її в достатній мірі коментарями. Продумайте і передбачте виведення програмою проміжних результатів обчислень, щоб по них можна було відслідкувати їх хід;
- Спробуйте відповісти на всі контрольні запитання.

Під час виконання роботи

- Виконуйте завдання послідовно. Як правило, результат попереднього завдання потрібен для виконання наступного;
- При внесенні змін в програму забезпечуйте нові фрагменти детальними коментарями;
- Знайдіть відповіді на контрольні запитання, що залишилися нез'ясованими при підготовці до роботи;
- Зробіть висновки по роботі.

Оформлений звіт по роботі повинен містити

- Робоче завдання та варіант для самостійної роботи;
- Коротку постановку задачі, розрахункові формули, проміжні і остаточні результати обчислень з необхідними поясненнями.

  Рисунки і таблиці повинні мати підписи, а всі скорочення і умовні позначення – їх роз'яснення в тексті. Потурбуйтеся про охайність виведення результатів і легкість їх читання. Зокрема, стовпчики таблиць повинні бути незсунутими, а друковані скріншоти мати білий фон (за потреби інвертуйте кольори зображення в негатив);
- Фрагмент програмного коду, що забезпечує основні обчислення, з коментарями;

- Обговорення отриманих результатів і допущених помилок у вигляді коротких, але принципово необхідних доведень, аналізів, обгрунтувань, пояснень, узагальнень тощо;
- Висновки по роботі.

Максимальна оцінка виставляється за умови

- Повного виконання завдання;
- Наявності оформленного належним чином звіту;
- Вчасного захисту лабораторної роботи;
- Відповіді на контрольні запитання під час захисту.

Лабораторна робота № 1.

Машинні константи

Мета роботи: знайомство з множиною чисел машинної арифметики з плаваючою точкою.

Що зробити: отримати значення машинних констант (машинного епсілон, найменьшого та найбільшого чисел машинної арифметики).

Стислі теоретичні відомості

Дійсні числа в пам'яті комп'ютера представляють як числа з плаваючою точкою у вигляді $a \times N^b$; число a називають *мантисою*, b – *показником*, а N – *основою системи обчислення*. Більшість сучасних комп'ютерів використовують основу системи обчислень N=16, хоча відомі моделі з N = 2, 8 або 10.

Щоб уникнути неоднозначності, зазвичай вимагають, щоб у мантиси ціла частина була нульова, а перша цифра справа від точки – навпаки, ненульовою. Таке представлення числа називають *нормалізованим*. Виключенням з правила є представлення нуля: $0.00...0 \times N^0$.

Кількість розрядів в мантисі a і порядку b визначає множину чисел, які мають представлення в пам'яті комп'ютера, зокрема межі його числової області. Наприклад, для десяткової (N=10) машини, у якої a має чотири, а b – два десяткових розряди, найменше та найбільше числа є -0.9999×10^{99} та 0.9999×10^{99}. Найменшим додатнім числом є 0.1000×10^{-99}, і між нулем та цим числом інших чисел немає.

Якщо абсолютна величина результату операції перевищує найбільше з машинних чисел, то відбувається *переповнення*, і обчислення зазвичай закінчуються. Якщо результатом операції є число, занадто близьке до нуля (в нашому прикладі менше ніж 10^{-100}), то відбувається *зникнення порядку*, або, іншими словами, *поява машинного нуля*.

Відносна точність машинної арифметики характеризується величиною *машинного епсілон* $\varepsilon_{маш}$, тобто найменшого числа з плаваючою точкою, що при складанні з числом 1.0 дає результат більший, ніж 1.0. (В нашому прикладі $\varepsilon_{маш}$ становить близько 0.0005.) Величина

$\varepsilon_{маш}$ не перевищує відстані між одиницею і наступним числом множини машинних чисел.

Завдання

1. Складіть програми, що дозволять вам визначити машинне епсілон, а також найменше та найбільше з чисел машинної арифметики. Ідеї алгоритмів запозичте з наступних фрагментів програм.

```
' Визначення машинного епсілон

eps=1
DO
   eps = eps/2
   epsp1 = eps+1
   PRINT eps, epsp1
LOOP WHILE epsp1 > 1
eps = eps*2
PRINT "eps mach ="; eps
END
```

```
' Найменше машинне число (underflow level)

ufl=1
DO
   ufl = ufl/2
   PRINT ufl
LOOP WHILE ufl > 0
END
```

```
' Найбільше машинне число (overflow level)

ofl=1
DO
   ofl = ofl*2
   PRINT ofl
LOOP             ' дочекатися аварійного завершення
END
```

2. Якщо версія транслятора, якою ви користуєтесь, допускає існування змінних з подвоєною точністю, визначте машинне епсілон, найменше та найбільше з машинних чисел для цього випадку.

Контрольні запитання

1. Поясніть, чому множина чисел машинної арифметики з плаваючою точкою має скінчену кількість елементів. Зверніть увагу на те, що кожне число цієї множини має значення

$$x = \pm\left(\frac{d_1}{N} + \frac{d_2}{N^2} + \ldots + \frac{d_t}{N^t}\right) \cdot N^b \ ,$$

де цілі числа $d_1, \ldots d_t$, (цифри мантиси) задовольняють нерівностям

$$1 \le d_i \le N - 1 \quad (i = 1) \ ,$$

$$0 \le d_i \le N - 1 \quad (i = 2, \ldots t) \ ,$$

а показник b знаходиться в певних межах $L \le b \le U$.

Покажіть, що вся множина характеризується лише чотирма параметрами: основою N, точністю t та інтервалом показників $[L, U]$.

2. Розгляньте уявну систему з плаваючою точкою, що складається з наступних чисел:

$$\{ \pm 0.d_1\, d_2\, d_3 \times 2^{\pm y} \} \ ,$$

де кожне з чисел d_2 , d_3 та y приймає одне із значень 0 або 1, а d_1 завжди дорівнює 1, за винятком випадку $d_1 = d_2 = d_3 = y = 0$.

Зобразіть фрагмент дійсної осі з нанесеними на неї елементами цієї множини чисел. Покажіть, що ця множина містить 25 елементів. Які значення машинного епсілон, найменшого та найбільшого з машинних чисел?

3. Доведіть, що множина чисел машинної арифметики з плаваючою точкою нараховує рівно $2\,(N–1)\,N^{t-1}\,(U–L+1) + 1$ елементів.

4. Поясніть, чому машинні числа з плаваючою точкою розподілені вздовж числової осі нерівномірно, зокрема більш густо поблизу нуля.

5. Який параметр характеризує відносну точність машинної арифметики?

6. Покажіть, що найменше та найбільше з чисел машинної арифметики визначаються головним чином значеннями L та U, тобто кількістю біт машинної пам'яті, відведених для зберігання порядку.

7. Покажіть, що машинне епсілон визначається кількістю біт машинної пам'яті, відведених для зберігання мантиси.

Лабораторна робота № 2.

Обчислення рядів

Мета роботи: обчислення нескінчених сум та добутків із заданою точністю.

Що зробити: отримати результат та оцінити його похибку, порівнявши результат розрахунків з величиною, отриманою за допомогою вбудованих функцій транслятора.

Стислі теоретичні відомості

Число x^*, яке не можна представити в комп'ютері точно, піддається округленню, тобто замінюється найближчим числом x, яке подається в комп'ютерній пам'яті. Це можна виразити таким записом:

$$x = x^*(1+\delta_x), \quad \text{або} \quad x - x^* = x^*\delta_x, \quad |\delta_x| \le \varepsilon_{маш},$$

Таким чином, відносна похибка наближення до числа, що зберігається в пам'яті комп'ютера, може досягати $\varepsilon_{маш}$.

Якщо в сумі $x + y$ доданки – додатні числа з плаваючою точкою, і $y < x$, то при $y < x\varepsilon_{маш}$ сума с плаваючою точкою цих чисел співпадає з x.

При обчисленні суми чисел x та y в типовому випадку ці числа є округленими версіями чисел x^* та y^* і мають відносні похибки, що не перевищують $\varepsilon_{маш}$. Однак відносна похибка їхньої суми $z = x + y$ може бути значно більшою:

$$z - z^* = z^*\delta_z = (x+y) - (x^* + y^*) = x^*\delta_x + y^*\delta_y.$$

Оскільки $|\delta_x|$ та $|\delta_y|$ можуть досягати $\varepsilon_{маш}$, то

$$|z^*| \cdot |\delta_z| \le (|x^*| + |y^*|)\,\varepsilon_{маш}, \qquad |\delta_z| \le \frac{|x^*| + |y^*|}{|x^* + y^*|}\varepsilon_{маш}$$

Особливо це явище проявляється при віднімання двох близьких чисел (тобто фактично мова йде про додавання двох близьких за абсолютною величиною, але протилежних за знаками чисел), коли знаменник останнього виразу стає малим.

Завдання

1. У варіантах завдань подані представлення деяких елементарних функцій у вигляді нескінченної суми або добутку. Складіть програму, яка би проводила обчислення цього ряду для заданого значення аргументу x із певною точністю обчислень ε. Під точністю розуміється деяке мале число (наприклад, 10^{-6}). Якщо черговий доданок (множник) відрізняється від нуля (одиниці) менш, ніж на це число, подальше додавання (множення) припиняється.

   Обчислення нескінченої суми $s = \sum_{k=1}^{\infty} u_k$ може проводитися за алгоритмом, блок-схема якого подана на рис. 2.1

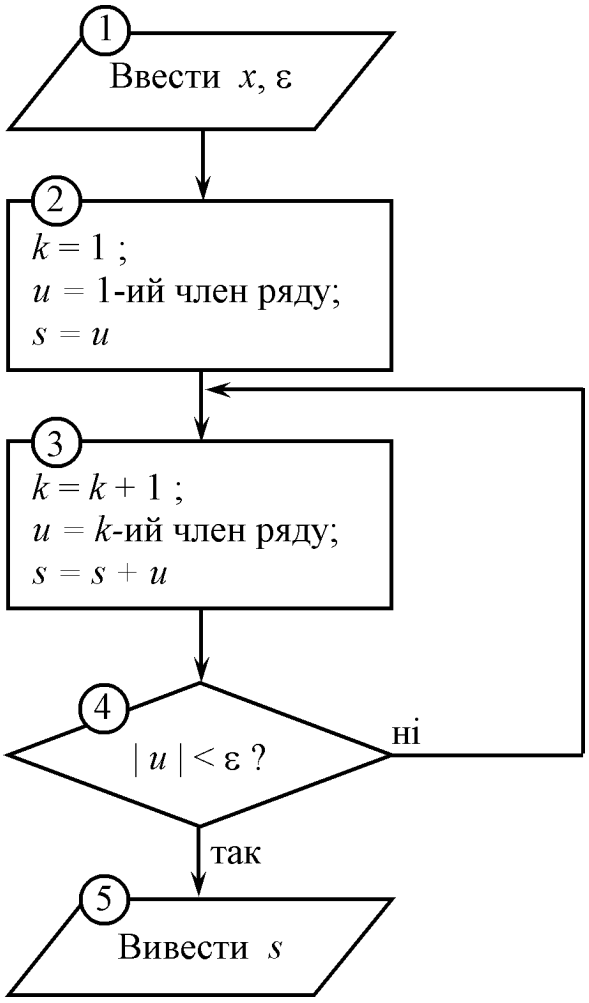


Рис. 2.1. Обчислення нескінченної суми s

Алгоритм обчислення нескінченого добутку $s = \prod_{k=1}^{\infty}(1+u_k)$ є аналогічним, за винятком перевизначення часткових добутків: $s = 1 + u$ (блок 2) та $s = s\cdot(1+u)$ (блок 3).

Прийміть до уваги, що в деяких випадках (особливо це стосується виразів, що містять факторіали) k-ий член ряду (блок 3) зручно обчислювати через попередній член за допомогою рекурентної формули. Наприклад, для

$$\exp x = 1 + \sum_{k=1}^{\infty}\frac{x^k}{k!}$$

k-ий член доречно обчислювати як $u_{\text{наступн}} = u_{\text{попередн}} \cdot x/k$.

Зверніть також увагу, що позначення n!! (подвійний факторіал), яке часто зустрічається у визначенні рядів, означає добуток всіх послідовних парних (якщо n парне) або непарних (якщо n непарне) натуральних чисел до n включно, тобто:

$$(2k)!! = 2\cdot 4\cdot 6\cdots 2k;$$
$$(2k-1)!! = 1\cdot 3\cdot 5\cdots (2k-1).$$

За означенням 0!! = 1.

Щоб уникнути зациклення програми із-за можливих помилок в програмуванні, доцільно передбачити примусовий вихід із циклу, якщо номер члена ряду k стає більшим, ніж певне велике число (наприклад, 10 000).

2. Порівняйте обчислене значення зі значенням, отриманим за допомогою вбудованих в транслятор функцій. Обчисліть і надрукуйте фактичну похибку ваших обчислень. Зіставте її з точністю обчислень ε, яку ви зажадали. Поясніть результати.

3. При обчисленні ряду в циклі обчислень передбачте виведення номера кожного члена ряду k , його величини і кожну часткову суму (добуток). Потурбуйтеся, щоб результати, що виводяться, мали вигляд охайної таблиці.

   Діалог вашої програми може бути приблизно таким (наведено обчислення ряду для exp x з прикладу п. 1 завдання)

```
введіть аргумент x  : 1
введіть похибку eps : 1E-4

k= 2    u= 0.50000000   s= 1.50000000
k= 3    u= 0.16666667   s= 1.66666663
k= 4    u= 0.04166667   s= 1.70833325
k= 5    u= 0.00833333   s= 1.71666658
k= 6    u= 0.00138889   s= 1.71805549
k= 7    u= 0.00019841   s= 1.71825385
k= 8    u= 0.00002480   s= 1.71827865

exp(x) за обчисл. рядом  = 2.71827864
exp(x) вбудована функція = 2.71828174
              їх різниця = 3.09944152E-006
```

4. Задаючи різні значення максимальної похибки ε (10^{-5}, 10^{-6} і так далі, аж до $\varepsilon_{маш}$), прослідкуйте за кількістю членів ряду, що знадобилися для обчислень, і фактичною похибкою результату. Поясніть отримані результати.
5. Зажадайте, щоб ваша програма проводила обчислення з подвоєною точністю, і прослідкуйте за подальшими змінами.

Варіанти для самостійної роботи

Варіанти 1, 13:

$$\arccos x = \operatorname{arctg}\frac{\sqrt{1-x^2}}{x} = \sqrt{2-2x}\left(1+\sum_{k=1}^{\infty}\frac{(2k-1)!!}{k!}\frac{z^k}{2k+1}\right) =$$

$$= \sqrt{2-2x}\left(1+\frac{z}{3}+\frac{1\cdot 3}{1\cdot 2}\cdot\frac{z^2}{5}+\frac{1\cdot 3\cdot 5}{1\cdot 2\cdot 3}\cdot\frac{z^3}{7}+\ldots\right), \quad \text{де } z=\frac{1-x}{4};\ |x|<1;$$

використовуйте діапазон $0.1 \le x \le 0.4$ (***вар. 1***) або $0.5 \le x \le 0.8$ (***вар. 13***).

Варіанти 2, 14:

$$\operatorname{arctg} x = \frac{x}{1+x^2}\left(1+\sum_{k=1}^{\infty}\frac{(2k)!!}{(2k+1)!!}z^k\right) =$$

$$= \frac{x}{1+x^2}\left(1+\frac{2}{3}z+\frac{2\cdot 4}{3\cdot 5}z^2+\frac{2\cdot 4\cdot 6}{3\cdot 5\cdot 7}z^3+\ldots\right), \quad \text{де } z=\frac{x^2}{1+x^2};\ |x|<\infty;$$

використовуйте діапазон $0.5 \le x \le 0.9$ (***вар. 2***) або $1 \le x \le 2$ (***вар. 14***).

Варіанти 3, 15:

$$\ln x = 2\sum_{k=1}^{\infty}\frac{z^{2k-1}}{2k-1} = 2\left(z+\frac{z^3}{3}+\frac{z^5}{5}+\ldots\right), \quad \text{де } z=\frac{x-1}{x+1};\ x>0;$$

використовуйте діапазон $0.2 \le x \le 0.8$ (***вар. 3***) або $1.2 \le x \le 2$ (***вар. 15***).

Варіанти 4, 16:

$$\sin x = x\prod_{k=1}^{\infty}\left(1-\frac{x^2}{(k\pi)^2}\right) = x\left(1-\frac{x^2}{\pi^2}\right)\left(1-\frac{x^2}{(2\pi)^2}\right)\left(1-\frac{x^2}{(3\pi)^2}\right)\ldots; \quad |x|<\infty;$$

використовуйте діапазон $0.2 \le x \le 0.7$ (***вар. 4***) або $0.8 \le x \le 1.3$ (***вар. 16***).

Варіанти 5, 17:

$$\operatorname{cosec} x = \frac{1}{\sin x} = \frac{1}{x}+2x\sum_{k=1}^{\infty}\frac{(-1)^k}{x^2-(k\pi)^2} =$$

$$= \frac{1}{x}+2x\left(-\frac{1}{x^2-\pi^2}+\frac{1}{x^2-(2\pi)^2}-\frac{1}{x^2-(3\pi)^2}\ldots\right); \quad x\neq 0,\ \pm\pi,\ \pm 2\pi,\ \ldots;$$

використовуйте діапазон $0.2 \le x \le 0.7$ (***вар. 5***) або $0.8 \le x \le 1.3$ (***вар. 17***).

Варіанти 6, 18:

$$\operatorname{cosec}^2 x = \frac{1}{\sin^2 x} = \frac{1}{x^2} + \sum_{k=1}^{\infty}\left(\frac{1}{(x-k\pi)^2} + \frac{1}{(x+k\pi)^2}\right) =$$

$$= \frac{1}{x^2} + \left(\frac{1}{(x-\pi)^2} + \frac{1}{(x+\pi)^2}\right) + \left(\frac{1}{(x-2\pi)^2} + \frac{1}{(x+2\pi)^2}\right) + \ldots;$$

$$x \neq 0,\ \pm\pi,\ \pm 2\pi,\ \ldots;$$

використовуйте діапазон $0.2 \le x \le 0.7$ (***вар. 6***) або $0.8 \le x \le 1.3$ (***вар. 18***).

Варіанти 7, 19:

$$\operatorname{arsh} x = \ln\left(x + \sqrt{x^2+1}\right) = x + \sum_{k=1}^{\infty}(-1)^k \frac{(2k-1)!!}{(2k)!!} \cdot \frac{x^{2k+1}}{2k+1} =$$

$$= x - \frac{1}{2}\cdot\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\cdot\frac{x^5}{5} - \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\cdot\frac{x^7}{7} + \ldots; \quad |x| < 1;$$

використовуйте діапазон $0.2 \le x \le 0.5$ (***вар. 7***) або $0.6 \le x \le 0.9$ (***вар. 19***).

Варіанти 8, 20:

$$\operatorname{arsh} x = \ln\left(x + \sqrt{x^2+1}\right) = \ln 2x + \sum_{k=1}^{\infty}(-1)^{k-1} \frac{(2k-1)!!}{(2k)!!} \cdot \frac{1}{2kx^{2k}} =$$

$$= \ln 2x + \frac{1}{2}\cdot\frac{1}{2x^2} - \frac{1\cdot 3}{2\cdot 4}\cdot\frac{1}{4x^4} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\cdot\frac{1}{6x^6} - \ldots; \quad |x| > 1;$$

використовуйте діапазон $1.5 \le x \le 3$ (***вар. 8***) або $4 \le x \le 10$ (***вар. 20***).

Варіанти 9, 21:

$$\sin x = \sum_{k=1}^{\infty}(-1)^{k-1} \frac{x^{2k-1}}{(2k-1)!} =$$

$$= x - \frac{x^3}{1\cdot 2\cdot 3} + \frac{x^5}{1\cdot 2\cdot 3\cdot 4\cdot 5} - \frac{x^7}{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6\cdot 7} + \ldots; \quad |x| < \infty;$$

використовуйте діапазон $0.2 \le x \le 0.7$ (***вар. 9***) або $0.8 \le x \le 1.3$ (***вар. 21***).

Варіанти 10, 22:

$$\cos x = \prod_{k=1}^{\infty}\left(1-\frac{4x^2}{(2k-1)^2\pi^2}\right) =$$

$$= \left(1-\frac{4x^2}{\pi^2}\right)\left(1-\frac{4x^2}{(3\pi)^2}\right)\left(1-\frac{4x^2}{(5\pi)^2}\right)\ldots; \quad |x|<\infty;$$

використовуйте діапазон $0.2 \le x \le 0.7$ (***вар. 10***) або $0.8 \le x \le 1.3$ (***вар. 22***).

Варіанти 11, 23:

$$\operatorname{sh} x = \frac{e^x - e^{-x}}{2} = x\prod_{k=1}^{\infty}\left(1+\frac{x^2}{(k\pi)^2}\right) =$$

$$= x\left(1+\frac{x^2}{\pi^2}\right)\left(1+\frac{x^2}{(2\pi)^2}\right)\left(1+\frac{x^2}{(3\pi)^2}\right)\ldots; \quad |x|<\infty;$$

використовуйте діапазон $0.5 \le x \le 0.9$ (***вар. 11***) або $1 \le x \le 2$ (***вар. 23***).

Варіанти 12, 24:

$$\operatorname{ch} x = \frac{e^x + e^{-x}}{2} = 1 + \sum_{k=1}^{\infty}\frac{x^{2k}}{(2k)!} =$$

$$= 1 + \frac{x^2}{1\cdot 2} + \frac{x^4}{1\cdot 2\cdot 3\cdot 4} + \frac{x^6}{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6} + \ldots; \quad |x|<\infty;$$

використовуйте діапазон $0.5 \le x \le 0.9$ (***вар. 12***) або $1 \le x \le 2$ (***вар. 24***).

Контрольні запитання

1. З якою метою в наведеному прикладі алгоритму обчислення нескінченної суми, що наведено в п. 1 завдання, перший член ряду програмується перед циклом явно?
2. Доведіть, що $(2k)!! = 2^k k!$; $(2k-1)!! = \dfrac{(2k)!}{2^k k!}$.

3. При виконанні прикладу з п. 3 завдання були отримані такі результати:

```
введіть аргумент x  : -10
введіть похибку eps : 1E-5

k= 2  u=   50.00000000  s=   40.00000000
k= 3  u= -166.66667175  s= -126.66667175
k= 4  u=  416.66668701  s=  290.00000000
k= 5  u= -833.33337402  s= -543.33337402
k= 6  u= 1388.88891602  s=  845.55554199
k= 7  u=-1984.12707520  s=-1138.57153320
k= 8  u= 2480.15893555  s= 1341.58740234
k= 9  u=-2755.73217773  s=-1414.14477539
k=10  u= 2755.73217773  s= 1341.58740234
k=11  u=-2505.21118164  s=-1163.62377930
k=12  u= 2087.67602539  s=  924.05224609
k=13  u=-1605.90466309  s= -681.85241699
k=14  u= 1147.07470703  s=  465.22229004
k=15  u= -764.71649170  s= -299.49420166
k=16  u=  477.94781494  s=  178.45361328
k=17  u= -281.14578247  s= -102.69216919
k=18  u=  156.19210815  s=   53.49993896
.....................................
k=28  u=    0.03279890  s=   -0.99160767
k=29  u=   -0.01130997  s=   -1.00291765
k=30  u=    0.00376999  s=   -0.99914765
k=31  u=   -0.00121613  s=   -1.00036383
k=32  u=    0.00038004  s=   -0.99998379
k=33  u=   -0.00011516  s=   -1.00009894
k=34  u=    0.00003387  s=   -1.00006509
k=35  u=   -0.00000968  s=   -1.00007474

exp(x) за обчисл. рядом  =   -7.47442245E-005
exp(x) вбудована функція =    4.53999309E-005
              їх різниця =    1.20144155E-004
```

Як видно, фактична похибка обчислень склала більше 100% (різняться навіть знаки отриманих результатів). Поясніть причини такого явища. Чи буде усунута проблема, якщо задати меншу, ніж 10^{-5} , максимальну похибку ε ? А якщо всі обчислення проводити з подвійною точністю, залишаючи ε без змін?

4. Доведіть, що відносна похибка добутку двох чисел $z = x \cdot y$ не перевищує суму відносних похибок множників: $|\delta_z| \le |\delta_x| + |\delta_y|$. При доведенні можете знехтувати квадратичними за δ членами.

5. Доведіть те ж саме для частки: при $z = x / y$ відносна похибка результата $|\delta_z| \le |\delta_x| + |\delta_y|$.
6. Дайте оцінку похибки δ_f обчислення значення функції $f(x, y)$. Як ця величина пов'язана з δ_x та δ_y ? Узагальніть результат на функцію багатьох змінних $f(x_1, x_2, x_3...)$.
7. Чим ще можуть бути викликані похибки в числах, що використовуються для комп'ютерних розрахунків, крім похибки округлення машинної арифметики?

Лабораторна робота № 3.

Дослідження функцій

Мета роботи: табулювання і зображення графіка функції для дослідження її характерних точок і особливостей поведінки

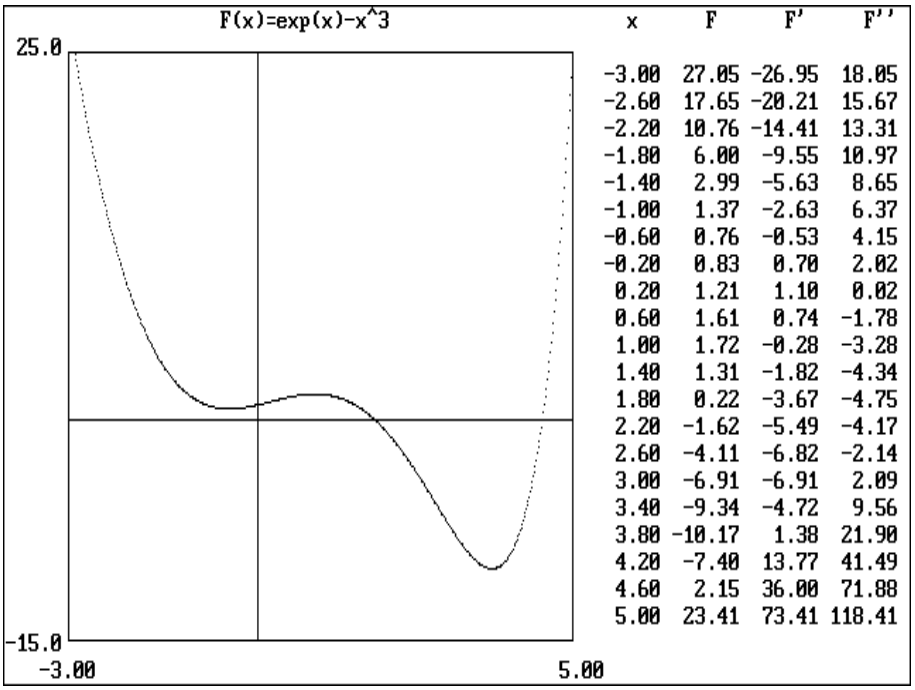
Що зробити: побудувати графік функції, попередньо дослідивши її аналітично. Впевнитись, що побудований фрагмент відображає всі особливості поведінки функції, які їй притаманні.

Стислі теоретичні відомості

Дослідити функцію означає *встановити її область визначення, область допустимих значень, розташування її особливих точок, нулів, екстремумів, перегинів, асимптот і асимптотичної поведінки при великих абсолютних значеннях аргументу, наявність тієї чи іншої симетрії, періодичності тощо*.

Завдання

1. Виберіть функцію для дослідження згідно з вашим варіантом та дослідіть її аналітично якомога більш детально. Визначте інтервал аргументу, який включає в себе всі особливості поведінки функції.
2. Складіть програму, що друкує таблицю значень функції і двох її похідних, а також креслить графік функції, як схематично показано на рис. 3.1. Обчислення функції $f(x)$ та двох її похідних $f'(x)$, $f''(x)$ рекомендується оформити у вигляді відповідних процедур. Потурбуйтеся, щоб графік наочно відображав всі особливості поведінки функції на її області визначення.
3. Порівняйте отримані результати з висновками аналітичного дослідження п. 1.



x	F	F'	F''
-3.00	27.05	-26.95	18.05
-2.60	17.65	-20.21	15.67
-2.20	10.76	-14.41	13.31
-1.80	6.00	-9.55	10.97
-1.40	2.99	-5.63	8.65
-1.00	1.37	-2.63	6.37
-0.60	0.76	-0.53	4.15
-0.20	0.83	0.70	2.02
0.20	1.21	1.10	0.02
0.60	1.61	0.74	-1.78
1.00	1.72	-0.28	-3.28
1.40	1.31	-1.82	-4.34
1.80	0.22	-3.67	-4.75
2.20	-1.62	-5.49	-4.17
2.60	-4.11	-6.82	-2.14
3.00	-6.91	-6.91	2.09
3.40	-9.34	-4.72	9.56
3.80	-10.17	1.38	21.90
4.20	-7.40	13.77	41.49
4.60	2.15	36.00	71.88
5.00	23.41	73.41	118.41

Рис. 3.1. Схематичний графік функції і таблиця функції та її похідних

Варіанти для самостійної роботи

Варіант 1: $f(x) = x^5 - 4x^3 + 1.5x/(1+x^2)$

Варіант 2: $f(x) = 2\cos 5x + x^2$

Варіант 3: $f(x) = \operatorname{sh} x - 2x^2 + 3$

Варіант 4: $f(x) = \operatorname{th} x - 0.5x - 0.2$

Варіант 5: $f(x) = \operatorname{tg} x - 8\sin(1.5x)$

Варіант 6: $f(x) = \sin x - 0.2x + 1.2$

Варіант 7: $f(x) = x^4 - 3x^3 - 1/(1+x^2) + 4x$

Варіант 8: $f(x) = \operatorname{arctg} x - (x+1)/(x+2)$

Варіант 9: $f(x) = 2x - 3 - (\ln x)/x^2$

Варіант 10: $f(x) = 1/\sqrt{7x - 3x^2} - x$

Варіант 11: $f(x) = \operatorname{arctg} x + 0.1x^2 - 0.8x$

Варіант 12: $f(x) = \sin x - 1/x$

Варіант 13: $f(x) = 1/(1+3\sin x) + 3\cos x$

Варіант 14: $f(x) = x^3 - 6x^2 + 2\sqrt{x+7}$

Варіант 15: $f(x) = 6\text{th}\, x - 5\text{arctg}\, x$

Варіант 16: $f(x) = \cos x + \cos(3x)$

Варіант 17: $f(x) = \text{ch}\, x - 4x^2$

Варіант 18: $f(x) = x^2 - \ln(x+0.5) - 3x$

Варіант 19: $f(x) = 20e^{-0.1x} + 3x - 50$

Варіант 20: $f(x) = \sqrt{x} - 4\cos(0.5x)$

Варіант 21: $f(x) = 10x^2e^{-x} - 3x$

Варіант 22: $f(x) = 2\text{sh}\, x - x^3$

Варіант 23: $f(x) = x^2 + 3 - 1/x$

Варіант 24: $f(x) = 5x - 8\ln x - 8$

Контрольні запитання

1. Який сенс вкладається в поняття «дослідження функції»?
2. Викладіть ваш спосіб дій при виборі інтервалу табулювання функції. Чи можете ви описати його за допомогою формальної схеми?

Лабораторна робота № 4.

Розв'язання нелінійних рівнянь з одним невідомим. Методи поділу навпіл (бісекції) та хорд

Мета роботи: вивчення алгоритмів і налаштування програм для розв'язання нелінійних рівнянь методом поділу навпіл (бісекції) і методом хорд.

Що зробити: знайти корені рівняння $f(x) = 0$ методом бісекції. Впевнитись, що їх значення узгоджуються з результатами аналітичного дослідження функції $f(x)$. Визначити порядок збіжності методу бісекції. Додатково – провести аналогічні дослідження методу хорд.

Стислі теоретичні відомості

А. Метод бісекції

Нехай потрібно розв'язати рівняння виду $f(x) = 0$. Припускається, що на інтервалі $x \in [a, b]$ функція $f(x)$ всюду визначена, є неперервною і змінює знак рівно один раз. За *методом поділу навпіл (бісекції)* наближення до кореню визначається як середня точка інтервалу (рис. 4.1).

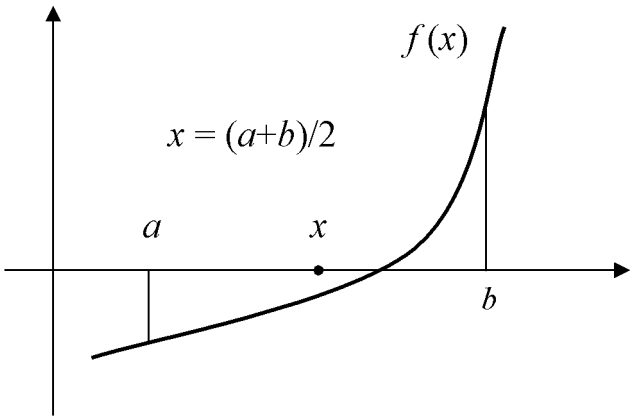


Рис. 4.1. Метод бісекції

В залежності від знака функції в точці x інтервал пошуку звужується вдвічі: до $[a, x]$ або $[x, b]$. Процес продовжується ітеративно доти, доки не буде виконана певна умова збіжності: наприклад, інтервал стане меншим за наперед задане мале число ε .

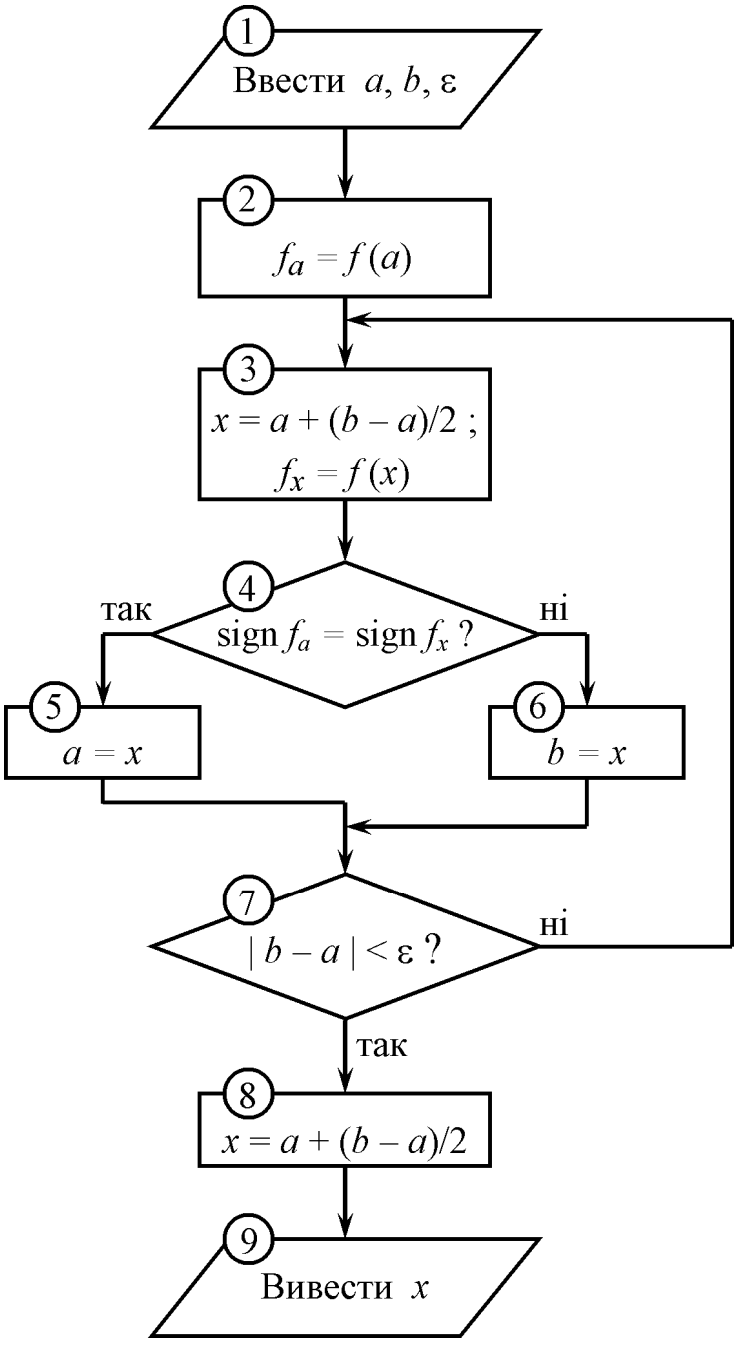


Рис. 4.2. Схема алгоритму методу бісекції

Б. Порядок збіжності ітераційного методу

Важливою характеристикою швидкості збіжності ітераційного методу є його порядок збіжності. Нехай $x^{(i)}$ – наближення до точного значення кореня x^* після проведення k ітерацій і $e^{(i)} = x^{(i)} - x^*$ – похибка i-го наближення. (Ми позначатимемо номер наближення верхнім індексом в дужках, резервуючи використання нижніх індексів для позначення в подальшому компонентів векторів і матриць.) Для методів, що збігаються, послідовність $e^{(1)}$, $e^{(2)}$, $e^{(3)}$, ... збігається до нуля. Для багатьох методів похибка наступного наближення пов'язана з похибкою попереднього приблизно як

$$e^{(i+1)} \approx C \left[e^{(i)}\right]^p .$$

Число p називають *порядком збіжності*.

Строго кажучи, p визначається як таке число, при якому границя

$$\lim_{i\to\infty} \frac{e^{(i+1)}}{\left[e^{(i)}\right]^p} = C$$

відмінна від 0 і від ∞. Можна показати, що при намаганні обчислити цю границю при іншому значенні p, що хоч якнайменше відрізняється від істинного порядку збіжності, її значення негайно стає або 0, або ∞.

Вочевидь, для методу бісекції $e^{(i+1)} = \frac{1}{2} e^{(i)}$ і його порядок збіжності дорівнює 1.

Оцінку порядку збіжності p ітераційного методу можна легко зробити, аналізуючи як зростає точність в послідовності наближень $x^{(i)}$. Нехай похибки становлять $e^{(i)} = 10^{-k^{(i)}}$, де $k^{(i)}$ – кількість вірних десяткових знаків в i-му наближенні. Тоді, зважаючи на попереднє співвідношення, маємо

$$10^{-k^{(i+1)}} \approx C\left[10^{-k^{(i)}}\right]^p;$$

$$k^{(i+1)} \approx -\lg C + pk^{(i)}.$$

Тобто, якщо $p = 1$, то кожна ітерація додає в наближення одне й те саме число десяткових знаків ($-\lg C$) і послідовність чисел $k^{(i)}$ складає арифметичну прогресію; якщо ж $p > 1$, то (за винятком, можливо, кількох найперших ітерацій) $k^{(i+1)} / k^{(i)} \sim p$, і кожна ітерація збільшує число вірних знаків в p разів, а послідовність чисел $k^{(i)}$ складає геометричну прогресію із знаменником p.

Завдання

1. Уясніть призначення окремих блоків схеми алгоритму для розв'язання рівняння виду $f(x) = 0$ методом бісекції. Складіть програму, що реалізує цей алгоритм. Фрагмент програми, що власне розв'язує рівняння, оформте у вигляді окремої процедури на зразок:

```
SUB Bisection (a,b,x,eps,ErrCode)
'
' ------------------------------------------------------
' Розв'язання рівняння f(x)=0 методом бісекції.
'
' Вхідні параметри:
'    a,b     - початкові границі інтервалу пошуку кореня,
'              не зберігаються при обчисленнях,
'              f(a) та f(b) повинні мати різні знаки;
'    eps     - точність розрахунків;
' Вихідні параметри:
'    x       - корінь рівняння f(x)=0;
'    ErrCode - код помилки:
'         ErrCode=0 якщо процедура завершилася успішно,
'         ErrCode=-1 якщо розв'язок не здобуто.
' ------------------------------------------------------
'
...
END SUB
```

   Передбачте в ній додатковий вихідний параметр – код помилки. Йому буде присвоєно одне певне значення (скажімо, 0), якщо процедура розв'язання пройшла успішно і помилок немає, і інше (скажімо, -1), якщо розв'язок не здобуто.

2. Після блоку 2 введіть додаткову перевірку правильності вибору початкового інтервалу пошуку – прийнятого припущення про те, що функція $f(x)$ на кінцях початкового інтервалу має різні знаки.

3. Візьміть ту ж саму функцію $f(x)$, яку ви досліджували при виконанні лабораторної роботи № 3. За допомогою вашої програми знайдіть її

найменший за модулем ненульовий корінь. Початковий інтервал пошуку кореня виберіть самостійно.

4. З метою налагодження програми і усвідомлення деталей роботи алгоритму введіть в програму після блоку 3 проміжний друк номера ітерацій *i,* а також значень *a, x, b,* $|b-a|$, $f(x)$ на кожній ітерації. Потурбуйтеся, щоб результати, що виводяться, мали вигляд охайної таблиці.
5. Дослідіть, як похибки поточного наближення до кореня $e^{(i)} = |b-a|$ залежать від номера ітерації *i*. Побудуйте графік залежності $\lg e^{(i)}$ від *i*. На основі цих даних впевніться, що порядок збіжності методу бісекції дорівнює 1.
6. Задавайте ε = 10^{-5}, 10^{-6}, 10^{-7}, ... Зменшуйте ε доти, доки програма не почне зациклюватися. Порівняйте цю величину з величиною машинного епсілон.
7. Знайдіть решту коренів рівняння $f(x) = 0$.

Стислі теоретичні відомості (продовження)

B. Метод хорд

Метод хорд відрізняється від методу бісекції тим, що нове наближення до кореня x визначається не як середина інтервалу пошуку, а як точка перетину хорди з віссю абсцис (рис. 4.3).

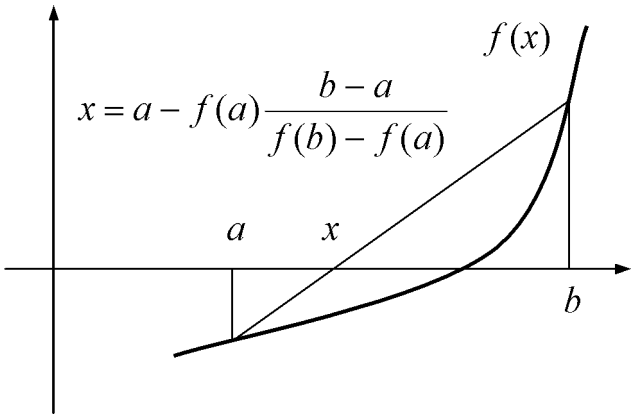


Рис. 4.3. Метод хорд

Зауважте, що в методі хорд не можна використовувати критерій збіжності ітераційного процесу $|b-a| < \varepsilon$. Пошук кореня повинен

закінчуватися при достатній близькості двох послідовних значень x , для чого в програмі необхідно передбачити спеціальну змінну, яка зберігає значення попереднього наближення до x.

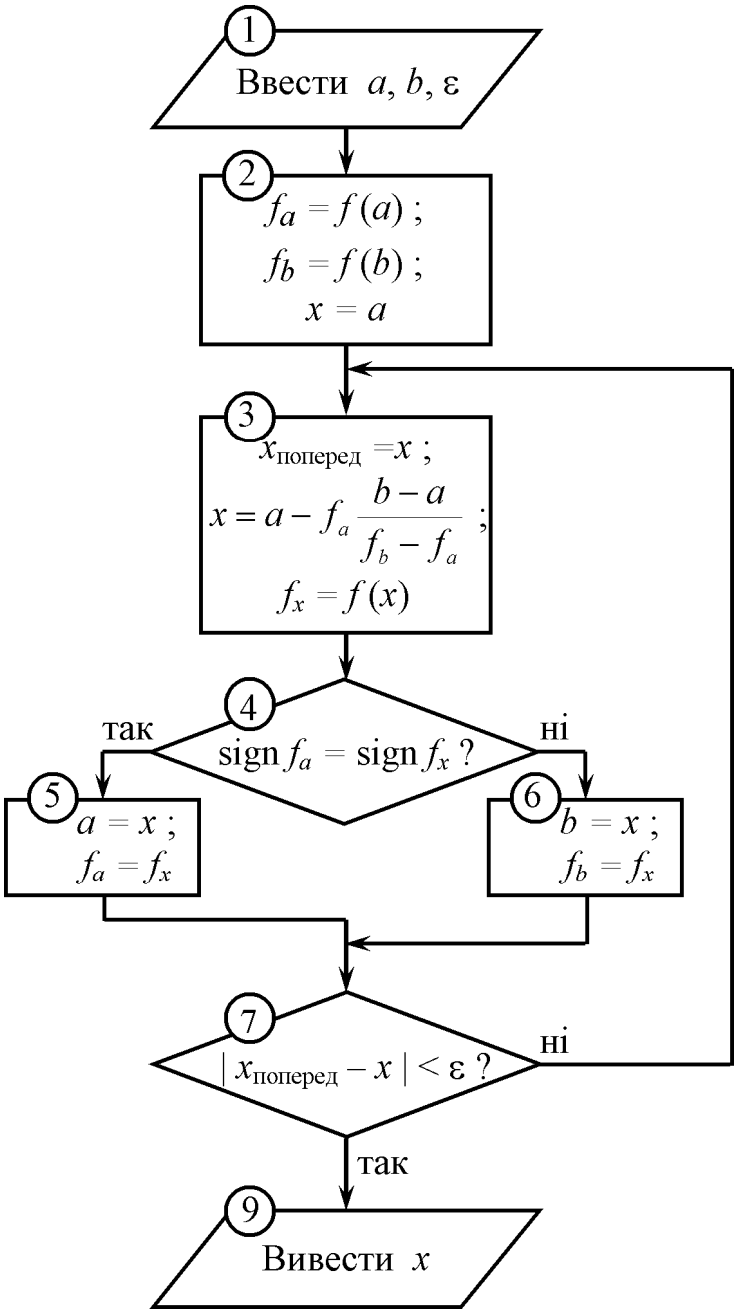


Рис. 4.4. Схема алгоритму методу хорд

Додаткове завдання

8. Порівняйте схеми алгоритмів обох методів і внесіть відповідні зміни в процедуру `Bisection` для реалізації методу хорд.
9. Додайте в рядок проміжного друку вашої програми виведення значення $|x_{\text{попер}} - x|$, що характеризує досягнуту точність поточного наближення $e^{(i)}$.
10. Знайдіть один або декілька коренів вашого рівняння за допомогою методу хорд. Порівняйте результати зі значеннями, знайденими методом бісекції.
11. Дослідіть, як похибки поточного наближення до кореня $e^{(i)}$ залежать від номеру ітерації i. Побудуйте графік залежності $\lg e^{(i)}$ від i. На основі цих даних з'ясуйте порядок збіжності методу хорд. Порівняйте порядки збіжності методів бісекції та хорд.

Контрольні запитання

1. Чому кількість ітерацій при бісекції приблизно дорівнює $n \sim \log_2(|b-a|/\varepsilon)$?
2. Яким чином можна з'ясувати порядок збіжності методу, аналізуючи залежність похибки поточного наближення від номера ітерації?
3. Чому при надто малому значенні ε програма зациклюється?
4. Чому в алгоритмі бісекції середня точка інтервалу $[a, b]$ (блок 3, рис. 4.2) розраховується як $x = a + (b-a)/2$, а не як $x = (a+b)/2$? Для відповіді на запитання уявіть гіпотетичний комп'ютер, що обчислює з точністю 2 десяткових знака і розрахуйте значення x за обома формулами при $a = 0.72$, $b = 0.74$.
5. Навіщо перед входом в цикл алгоритму хорд в (блок 2, рис. 4.4) передбачено присвоєння $x = a$? Чи можна його замінити на $x = b$? А на $x = (a+b)/2$?
6. Поясніть, чому в методі хорд не можна використовувати критерій збіжності ітераційного процесу $|b-a| < \varepsilon$.
7. Чому в схемі алгоритму методу хорд відсутній блок 8 ?
8. Як поведуть себе методи бісекції і хорд, якщо на інтервалі $x \in [a, b]$ функція $f(x)$ лишається всюду визначеною, але припущення, що вона є неперервною і змінює знак рівно один раз, невірне? А якщо кількість перемін її знаку не дорівнює одному?

Лабораторна робота № 5.

Розв'язання нелінійних рівнянь з одним невідомим. Метод простих ітерацій

Мета роботи: вивчення алгоритмів і налаштування програм для розв'язання нелінійних рівнянь методом простих ітерацій.

Що зробити: привести рівняння виду $f(x) = 0$ до виду $x = g(x)$, придатного для застосування методу простих ітерацій, можливо, використовуючи різні види $g(x)$ для різних коренів. Знайти корені рівняння цим методом, попередньо впевнившись у збіжності ітераційного процесу. Впевнитись, що значення коренів узгоджуються з результатами аналітичного дослідження функції $f(x)$. Визначити порядок збіжності методу простих ітерацій.

Стислі теоретичні відомості

Рівняння з одним невідомим може бути приведено до форми

$$x = g(x) \tag{1}$$

Розглянемо можливість знаходження кореня рівняння в ході ітераційного процесу

$$x^{(i+1)} = g(x^{(i)}), \tag{2}$$

що продовжується до тих пір, доки не буде виконана деяка умова збіжності, наприклад, доки два послідовних наближення не стануть достатньо близькими.

Схематично цей ітераційний процес при різних видах функції $g(x)$ представлено на рис. 5.1, який ілюструє той факт, що достатньою умовою збіжності цього процесу є виконання нерівності

$$|g'(x)| < 1$$

в деякому околі кореня рівняння.

Строге доведення цього твердження може бути здійснено на основі таких міркувань. Нехай x^* – точне значення кореня рівняння (1), а $e^{(i)} = x^{(i)} - x^*$ – похибка i-го наближення. Тоді розкладаючи (2) в ряд Тейлора навколо точки x^*

$$x^* + e^{(i+1)} = g(x^* + e^{(i)}) = g(x^*) + g'(x^*)\cdot e^{(i)} + \ldots$$

і приймаючи до уваги, що при $x = x^*$ рівність (1) виконується, а також нехтуючи членами більш високих порядків по $e^{(i)}$, маємо:

$$e^{(i+1)} \approx g'(x^*)\cdot e^{(i)}\cdot \ .$$

Отже, похибки послідовних наближень приблизно утворюють геометричну прогресію, яка дійсно збігається до нуля, якщо її знаменник за абсолютною величиною менший за 1.

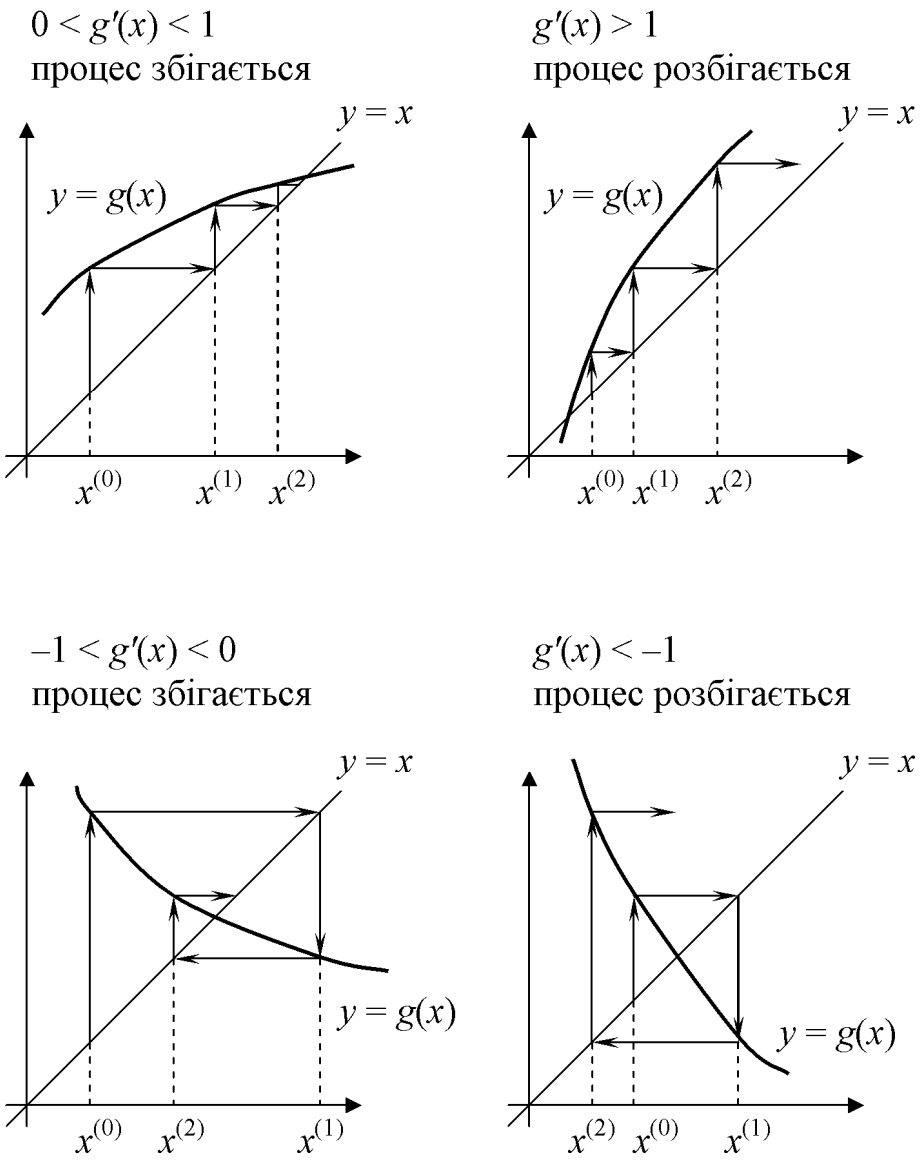


Рис. 5.1. Метод простих ітерацій

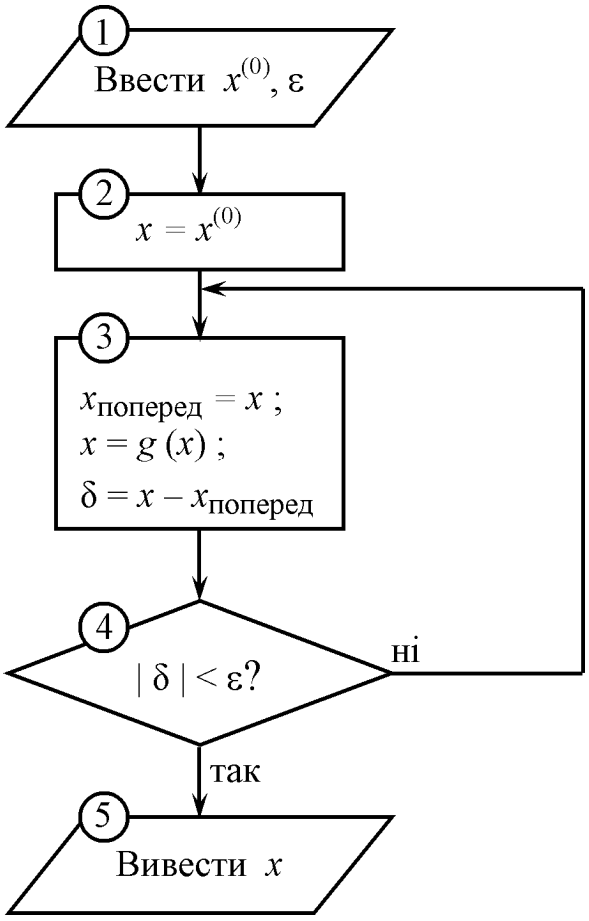


Рис. 5.2. Схема алгоритму методу простих ітерацій

Завдання

1. Приведіть рівняння виду $f(x) = 0$, що ви розв'язували при виконанні лабораторної роботи № 4, до виду $x = g(x)$, придатного для застосування методу простих ітерацій. (Функція $f(x)$ – та ж сама, яку ви досліджували при виконанні лабораторної роботи № 3.)
2. Уясніть призначення окремих блоків схеми алгоритму для розв'язання рівняння виду $x = g(x)$ методом простих ітерацій. Складіть програму, що реалізує цей алгоритм. Фрагмент програми, що власне розв'язує рівняння, оформте у вигляді окремої процедури.
3. З метою налагодження програми і усвідомлення деталей роботи алгоритму введіть в програму після блоку 2 проміжний друк номера ітерацій *i,* а також значень x , δ на кожній ітерації. Потурбуйтеся, щоб результати, що виводяться, мали вигляд охайної таблиці.

4. З метою гарантованого завершення програми навіть у випадку розбіжності ітераційного процесу запровадьте в програму обмеження на максимальну кількість ітерацій. Передбачте виведення відповідного повідомлення про незбіжність ітераційного процесу.
5. За допомогою вашої програми знайдіть найменший за модулем ненульовий корінь рівняння. Початкове наближення до кореня виберіть самостійно.
6. Дослідіть, як похибки поточного наближення до кореня $e^{(i)} = |\delta|$ залежать від номера ітерації i. Побудуйте графік залежності $\lg e^{(i)}$ від i. На основі цих даних з'ясуйте порядок збіжності методу простих ітерацій.

Додаткове завдання

7. Знайдіть решту коренів рівняння $f(x) = 0$. Для цього, можливо, доведеться перетворити рівняння до виду $x = g(x)$ іншим чином.

Контрольні запитання

1. Рівняння $f(x) = x^2 - c = 0$ з коренями $\pm\sqrt{c}$ було перетворено до трьох різних форм:

$$x = g_1(x) = \frac{c}{x};$$

$$x = g_2(x) = x + f(x) = x^2 + x - c;$$

$$x = g_3(x) = \frac{x + g_1(x)}{2} = \frac{x^2 + c}{2x}$$

   Застосуйте критерій і оцініть збіжність методу простих ітерацій у кожному випадку для обох коренів.
2. Рівняння $f(x) = 0$ можна привести до форми $x = g(x)$ перетворенням виду $x = x + \lambda f(x)$, де λ – довільна ненульова величина. Яке оптимальне значення цього параметру з точки зору забезпечення збіжності ітераційного процесу?
3. Розгляньте два сусідніх кореня рівняння $x = g(x)$ в припущенні, що функція $g(x)$ на ділянці між ними безперервна. Доведіть, що при намаганні визначити корені методом простих ітерацій принаймні один з них буде породжувати розбіжний ітераційний процес.

4. Обговоріть доцільність застосування критерію збіжності не за абсолютною близькістю двох послідовних значень $|\delta| < \varepsilon_{\text{абс}}$, а за відносною $|\delta| < |x|\varepsilon_{\text{відн}}$. При яких значеннях кореня слід користуватися тим чи іншим критерієм? Складіть комбінований критерій, що успішно працює при будь-яких значеннях кореня.
5. Чи можна в процесі обчислень діагностувати ситуацію незбіжності ітераційного процесу? Обговоріть наступне твердження: «ітераційний процес розбігається, якщо значення $|\delta|$ при поточній ітерації збільшилося порівняно з попереднім».

Лабораторна робота № 6.

Розв'язання нелінійних рівнянь з одним невідомим. Методи Ньютона-Рафсона (дотичних) та січних

Мета роботи: вивчення алгоритмів і налаштування програм для розв'язання нелінійних рівнянь методом Ньютона-Рафсона (дотичних) і методом січних.

Що зробити: знайти корені рівняння $f(x) = 0$ методом Ньютона-Рафсона, попередньо впевнившись у збіжності ітераційного процесу. Впевнитись, що значення коренів узгоджуються з результатами аналітичного дослідження функції $f(x)$. Визначити порядок збіжності методу Ньютона-Рафсона. Додатково – провести аналогічні дослідження методу січних.

Стислі теоретичні відомості

А. Метод Ньютона-Рафсона (дотичних)

Ітераційний *метод Ньютона-Рафсона (дотичних)* розв'язку рівняння виду $f(x) = 0$ полягає в наступному. Нехай $x^{(i)}$ – деяке наближення до кореня рівняння. Наступне наближення $x^{(i+1)}$ до кореня можна обчислити як перетин дотичної до кривої $f(x)$ в точці ($x^{(i)}$, $f^{(i)}$) з віссю абсцис (рис. 6.1). (Тут і далі значення $f(x^{(i)})$ позначаються як $f^{(i)}$.)

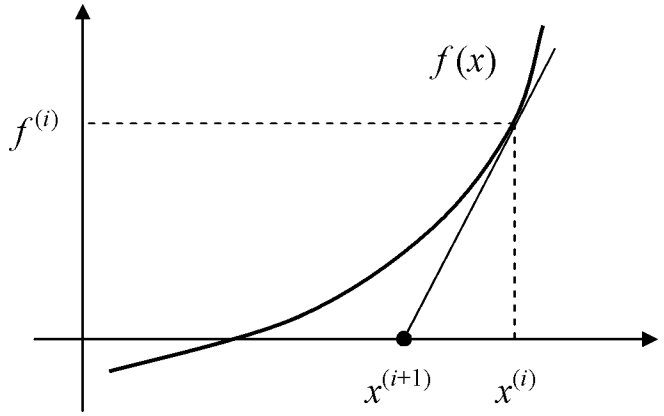


Рис. 6.1. Метод Ньютона-Рафсона

По суті, мі замінюємо в околі точки $x^{(i)}$ криву $f(x)$ прямою лінією – її дотичною, а точне рівняння $f(x) = 0$ – наближеним:

$$f(x) \approx f^{(i)} + (x - x^{(i)})\, f'^{(i)} = 0 .$$

Розв'язок наближеного рівняння трактується як чергове наближення до кореня точного рівняння:

$$x^{(i+1)} = x^{(i)} - \frac{f^{(i)}}{f'^{(i)}} \qquad (1)$$

Процес продовжується доти, доки не буде виконана деяка умова збіжності, наприклад, доки два послідовних наближення не стануть достатньо близькими.

Алгоритм методу аналогічний алгоритму методу простих ітерацій, що досліджувався при виконанні лабораторної роботи № 5. Його схема подана на рис. 6.2. Єдина відмінність полягає в змісті блоку 3, що містить власне ітераційну формулу (рис. 6.2).

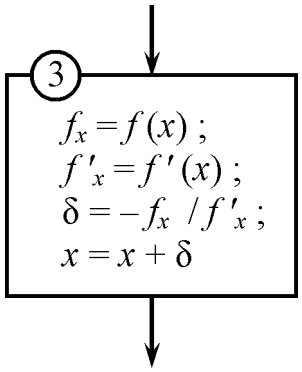


Рис. 6.2. Модифікація схеми алгоритму методу простих ітерацій для методу Ньютона-Рафсона

Завдання

1. Модифікуйте процедуру, яку ви налагодили для розв'язання рівняння методом простих ітерацій при виконанні лабораторної роботи № 5, для розв'язання рівняння виду $f(x) = 0$ методом Ньютона-Рафсона. (Функція $f(x)$ – та ж сама, яку ви досліджували при виконанні лабораторної роботи № 3.) Початкові наближення виберіть самостійно, намагаючись визначити якомога більше коренів.

2. Як і в лабораторній роботі № 5, дослідіть залежність похибки поточного наближення до кореня $e^{(i)} = |\delta|$ від номера ітерації i. Побудуйте графік залежності $\lg e^{(i)}$ від i. На основі цих даних з'ясуйте порядок збіжності методу Ньютона-Рафсона. Можливо, для цього прийдеться проводити розрахунки з подвійною точністю.

Стислі теоретичні відомості (продовження)

Б. Метод січних

Метод січних використовується тоді, коли обчислення похідної $f'(x)$ в явному вигляді утруднене. В цьому випадку дотична замінюється на січну, що проходить через точки двох останніх наближень (рис. 6.3).

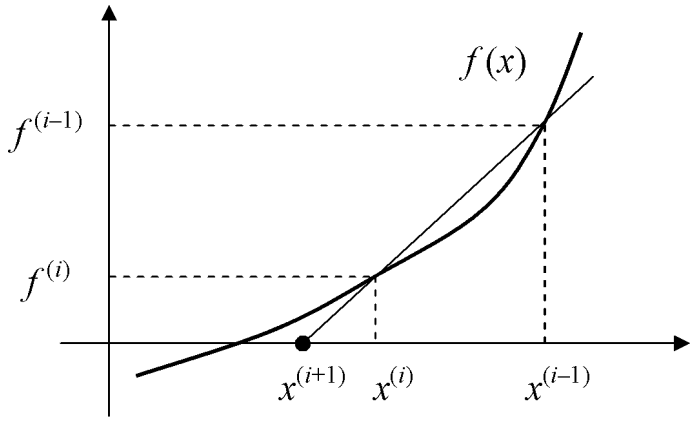


Рис. 6.3. Метод січних

Відповідно,

$$f'^{(i)} \approx \frac{f^{(i)} - f^{(i-1)}}{x^{(i)} - x^{(i-1)}}$$

і

$$x^{(i+1)} = x^{(i)} - f^{(i)} \frac{x^{(i)} - x^{(i-1)}}{f^{(i)} - f^{(i-1)}} \ . \qquad (2)$$

Зауважимо, що для старту алгоритму січних потрібні дві початкових точки: $x^{(0)}$ та $x^{(1)}$.

Схема алгоритму також подібна до схеми методу простих ітерацій, представленої на рис. 6.2. Відмінності подані на рис. 6.4.

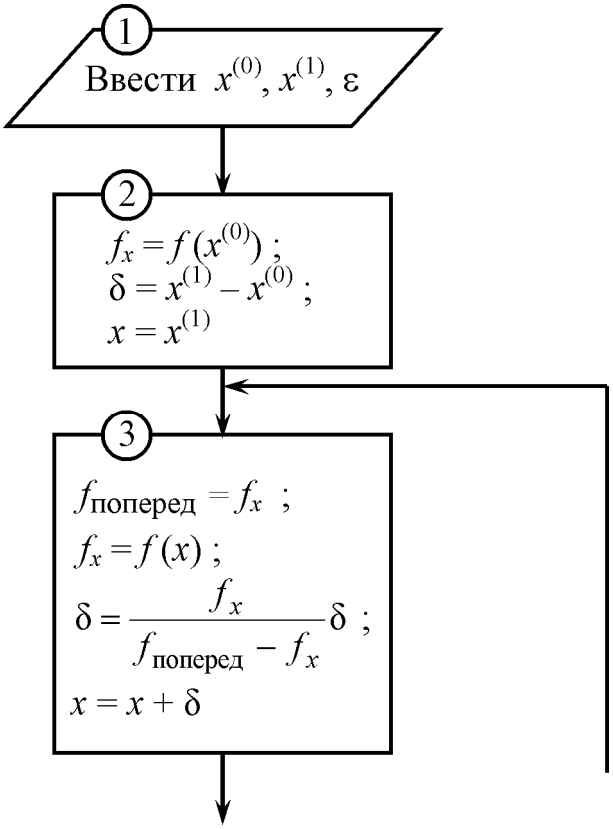


Рис. 6.4. Модифікація схеми алгоритму методу простих ітерацій для методу січних

Додаткове завдання

3. Введіть відповідні зміни в вашу програму для реалізації методу січних. Початкові наближення $x^{(0)}$ та $x^{(1)}$ виберіть самостійно.
4. Дослідіть, як похибки поточного наближення до кореня $e^{(i)} = |\delta|$ залежать від номера ітерації i. Побудуйте графік залежності $\lg e^{(i)}$ від i. На основі цих даних з'ясуйте порядок збіжності методу січних. Можливо, для цього прийдеться проводити розрахунки з подвійною точністю.

Контрольні запитання

1. Поясніть, чому ітераційні формули методів дотичних (1) і січних (2) не зведені до спільного знаменника, а виражають бажану величину як малу поправку до хорошого наближення?
2. Метод Ньютона-Рафсона еквівалентний методу простих ітерацій $x^{(i+1)} = g(x^{(i)})$, де

$$g(x) = x - \frac{f(x)}{f'(x)}$$

Розкладіть функцію $g(x)$ в ряд Тейлора навколо точки x^* і покажіть, що

$$e^{(i+1)} \approx \frac{f''(x^*)}{2f'(x^*)}\left[e^{(i)}\right]^2,$$

тобто метод Ньютона-Рафсона має порядок збіжності рівний 2.

3. Покажіть, що при $x = x^*$ виконується рівність $|g'(x)| = 0$, і, отже, навколо кожного кореня існує непуста область збіжності, де $|g'(x)| < 1$. Таким чином, якщо початкове наближення знаходиться в цій області, ітераційний процес методу Ньютона-Рафсона завжди збігається.
4. Розкладіть функцію $g'(x)$ в ряд Тейлора навколо точки x^* і покажіть, що межі області збіжності (тобто точки, де $g'(x)$ приймає значення ±1,) приблизно дорівнюють $x^* \pm \frac{f'(x^*)}{f''(x^*)}$.
5. Оцініть інтервали, в яких допустимо вибирати початкові наближення $x^{(0)}$ для знаходження методом Ньютона-Рафсона кожного з коренів досліджуваного вами конкретного рівняння.
6. Покажіть, що похибки послідовних наближень, обчислюваних за методом січних, пов’язані співвідношенням

$$e^{(i+1)} \approx \frac{f''(x^*)}{2f'(x^*)} e^{(i)} e^{(i-1)}.$$

7. З’ясуйте порядок збіжності методу січних. Підставте в попередню формулу співвідношення

$$e^{(i+1)} \approx C\left[e^{(i)}\right]^p$$

і покажіть, що порядок збіжності p задовольняє рівнянню $p^2 = p+1$, тобто $p = \left(\sqrt{5}+1\right)/2 \approx 1.618\ldots$

8. Чи може метод дотичних застосовуватися для знаходження кратних коренів? А інші відомі вам методи?

9. Обговоріть для кожного з методів можливість застосування критерію збіжності $|f(x)| < \varepsilon$.

10. За якими критеріями можна оцінювати різні методи розв’язання нелінійних рівнянь з одним невідомим? Який з досліджених вами методів показав себе найбільш ефективним?

Лабораторна робота № 7.

Дослідження неявно заданих функцій

Мета роботи: розрахунок таблиці значень і побудова графіка неявно заданої функції

Що зробити: побудувати графік функції, заданої неявно, використовуючи при розрахунках один з методів розв'язання нелінійних рівнянь з одним невідомим. Розрахувати таблицю функції та її двох похідних і впевнитися у взаємоузгодженості отриманих результатів.

Стислі теоретичні відомості

Неявно задана функція $y(x)$ задається рівняням $\varphi(x, y) = 0$. Для табуляції функції $y(x)$ слід, задаючи x в межах інтервалу табулювання, кожного разу розв'язувати рівняння $\varphi(x, y) = 0$ відносно y. Іноді розв'язок може бути найдено аналітично – в цьому разі функція $y(x)$ стає явною. В загальному ж випадку слід застосовувати чисельні методи розв'язання рівнянь з одним невідомим.

Терміни «явно» та «неявно» характеризують лише спосіб визначення функції і аж ніяк не мають відношення до її природи. Строго кажучи, протиставлення явного та неявного задання функції можливо лише якщо під явним заданням розуміти явне аналітичне задання. Якщо ж, як явне, допускати задання за допомогою будь-якого правила, то задання функції $y(x)$ за допомогою рівняння $\varphi(x,y) = 0$ нічим не гірше всякого іншого.

Можна показати, що похідні функції $y(x)$ обчислюються за формулами:

$$y' = -\frac{\varphi'_x(x, y)}{\varphi'_y(x, y)}, \qquad y'' = -\frac{\varphi''_{xx}\left(\varphi'_y\right)^2 - 2\varphi''_{xy}\,\varphi'_x\varphi'_y + \varphi''_{yy}\left(\varphi'_x\right)^2}{\left(\varphi'_y\right)^3}. \qquad (1)$$

Завдання

1. Згідно з вашим варіантом виведіть формули для обчислення похідних y' та y''.
2. Складіть програму для друкування таблиці значень функції $y(x)$ і двох її похідних, а також креслення графіку функції в заданому інтервалі значень x. За основу візьміть вашу програму з лабораторної роботи № 3. Якщо ви дослухалися рекомендації оформляти обчислення функції $f(x)$ та двох її похідних $f'(x)$, $f''(x)$ у вигляді відповідних процедур, то модифікація програми не вимагатиме багато зусиль. Досить зробити лише наступне.

   Процедуру обчислення функції $f(x)$ замініть на процедуру розв'язання рівняння $\varphi(x, y) = 0$ відносно y, вважаючи x відомим. Для цього скористайтеся вашим доробком з лабораторних робіт №№ 4–6.

   Процедури обчислення похідних замініть на вирази, отримані в п. 1.

Варіанти для самостійної роботи

Після номеру варіанта вказано метод розв'язання рівняння:

- бісекції (Б),
- простих ітерацій (І),
- Ньютона-Рафсона (Н)

Варіанти 1 (Б), 2 (І), 3 (Н): $x - y - 4xy^2\sqrt{x^2 + y^2} = 0;$ $0 \le x \le 5$

Варіанти 4 (Б), 5 (І), 6 (Н): $x - y + 5\exp(-x^2 - y^2) = 0;$ $-3 \le x \le 3$

Варіанти 7 (Б), 8 (І), 9 (Н): $y^3 + y^2\sin x + y + \cos x = 0;$ $-2\pi \le x \le 2\pi$

Варіанти 10 (Б), 11 (І), 12 (Н): $x + y - e^{x-y} = 0;$ $-5 \le x \le 5$

Варіанти 13 (Б), 14 (І), 15 (Н): $x^2 + y^2 + \ln(x + y) = 0;$ $-2 \le x \le 2$

Варіанти 16 (Б), 17 (І), 18 (Н): $xy - \sin(2x + y) = 0;$ $1 \le x \le 11$

Варіанти 19 (Б), 20 (І), 21 (Н): $x - y - 2\operatorname{arctg}(xy) = 0;$ $-3 \le x \le 5$

Варіанти 22 (Б), 23 (І), 24 (Н): $y^3 + x^2y - x + 1 = 0;$ $-6 \le x \le 6$

Контрольні запитання

1. Виведіть формули (1) для обчислення першої та другої похідної від неявно заданої функції.
2. Як програма повинна обробляти ситуацію, коли одна чи обидві похідні стають нескінченими?
3. Як програма повинна обробляти ситуацію, коли при вибраному x рівняння $\varphi(x,y) = 0$ відносно y не розв'язується (не має коренів)?
4. Можлива ситуація, коли рівняння $\varphi(x,y) = 0$ при фіксованому x має декілька коренів (іншими словами, функція $y(x)$ багатозначна). Викладіть ваш спосіб дій в такому випадку.

Лабораторна робота № 8.

Розв'язання систем лінійних алгебраїчних рівнянь. Метод Гауса

Мета роботи: вивчення алгоритмів і налаштування програм розв'язання систем лінійних алгебраїчних рівнянь (СЛАР) методом Гауса.

Що зробити: скласти процедуру для розв'язання СЛАР методом Гауса, яка б у випадку невиродженої системи знаходила її розв'язок, а для виродженої системи видавала відповідне попередження. Впевнитися в коректності роботи процедури, підставляючи в СЛАР отримані розв'язки і обраховуючи нев'язки. Додатково – передбачити оцінку числа обумовленості матриці системи.

Стислі теоретичні відомості

А. Метод Гауса

Метод Гауса дозволяє знаходити розв'язок будь-якої невиродженої системи *n* лінійних алгебраїчних рівнянь (СЛАР) виду:

$$
\begin{aligned}
a_{11}x_1 &+ a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\
a_{21}x_1 &+ a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
a_{n1}x_1 &+ a_{n2}x_2 + \ldots + a_{nn}x_n = b_n
\end{aligned}
$$

або, у матрично-векторному запису, $\mathbf{Ax} = \mathbf{b}$.

Процедура розв'язання проходить в два етапи:

- зведення системи до трикутного виду (так званий «прямий хід»)
- зворотня підстановка (так званий «зворотній хід»)

Зведення системи до трикутного виду полягає в її послідовних лінійних перетвореннях з метою обнулення коефіцієнтів, що знаходяться в її матриці нижче головної діагоналі.

Матриця вихідної системи загального виду перетворюється за Гаусом до трикутної форми в $n-1$ стадію. На 1-ій стадії обнулюються $n-1$ елементів 1-го стовпчика, що знаходяться нижче головної діагоналі; на 2-ій – $n-2$ елементів 2-го стовпчика; ... ; на i-ій стадії – $n-i$ елементів i-го стовпчика; ... ; на $n-1$-ій стадії – один нижній елемент передостаннього стовпчика. (Вочевидь, в останньому, n-му стовпчику елементів нижче головної діагоналі немає).

Перед i-ю стадією система рівнянь приймає вигляд:

$$
\begin{array}{llllll}
a_{11}x_1 + a_{12}x_2 & + \ldots + a_{1i}x_i & + \ldots + a_{1j}x_j & + \ldots + a_{1n}x_n & = & b_1 \\
\phantom{a_{11}x_1 +{}} a_{22}x_2 & + \ldots + a_{2i}x_i & + \ldots + a_{2j}x_j & + \ldots + a_{2n}x_n & = & b_2 \\
 & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & & \\
 & \quad a_{ii}x_i & + \ldots + a_{ij}x_j & + \ldots + a_{in}x_n & = & b_i \\
 & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & & \\
 & \quad a_{ki}x_i & + \ldots + a_{kj}x_j & + \ldots + a_{kn}x_n & = & b_k \\
 & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & & \\
 & \quad a_{ni}x_i & + \ldots + a_{nj}x_j & + \ldots + a_{nn}x_n & = & b_n
\end{array}
$$

На i-ій стадії i-те рівняння залишається без змін, а всі наступні – з $i+1$-го по n-е – перетворюються за зразком

(k-те рівняння) $\longleftarrow$ (k-те рівняння) – $p \cdot$ (i-те рівняння),

де

$$p = a_{ki} / a_{ii} \tag{1}$$

– так звані Гаусові множники. (Елементи a_{ii} називають ведучими елементами). Легко бачити, що при такому виборі p коефіцієнти при x_i нижче головної діагоналі обнулюються.

Отже, після зведення до трикутної, система набуває вигляду:

$$
\begin{array}{rrrrrcl}
a_{11}x_1 & + a_{12}x_2 & + \ldots + a_{1i}x_i & + \ldots + a_{1j}x_j & + \ldots + a_{1n}x_n & = & b_1 \\
 & a_{22}x_2 & + \ldots + a_{2i}x_i & + \ldots + a_{2j}x_j & + \ldots + a_{2n}x_n & = & b_2 \\
 & & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & & \\
 & & a_{ii}x_i & + \ldots + a_{ij}x_j & + \ldots + a_{in}x_n & = & b_i \\
 & & & \ldots\ldots\ldots & \ldots\ldots\ldots & & \\
 & & & & a_{nn}x_n & = & b_n
\end{array}
$$

Її розв’язок легко знаходиться зворотньою підстановкою:

$$
x_n = \frac{b_n}{a_{nn}}; \qquad x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij}x_j}{a_{ii}}, \quad i = n-1,\ n-2,\ ...,\ 1. \tag{2}
$$

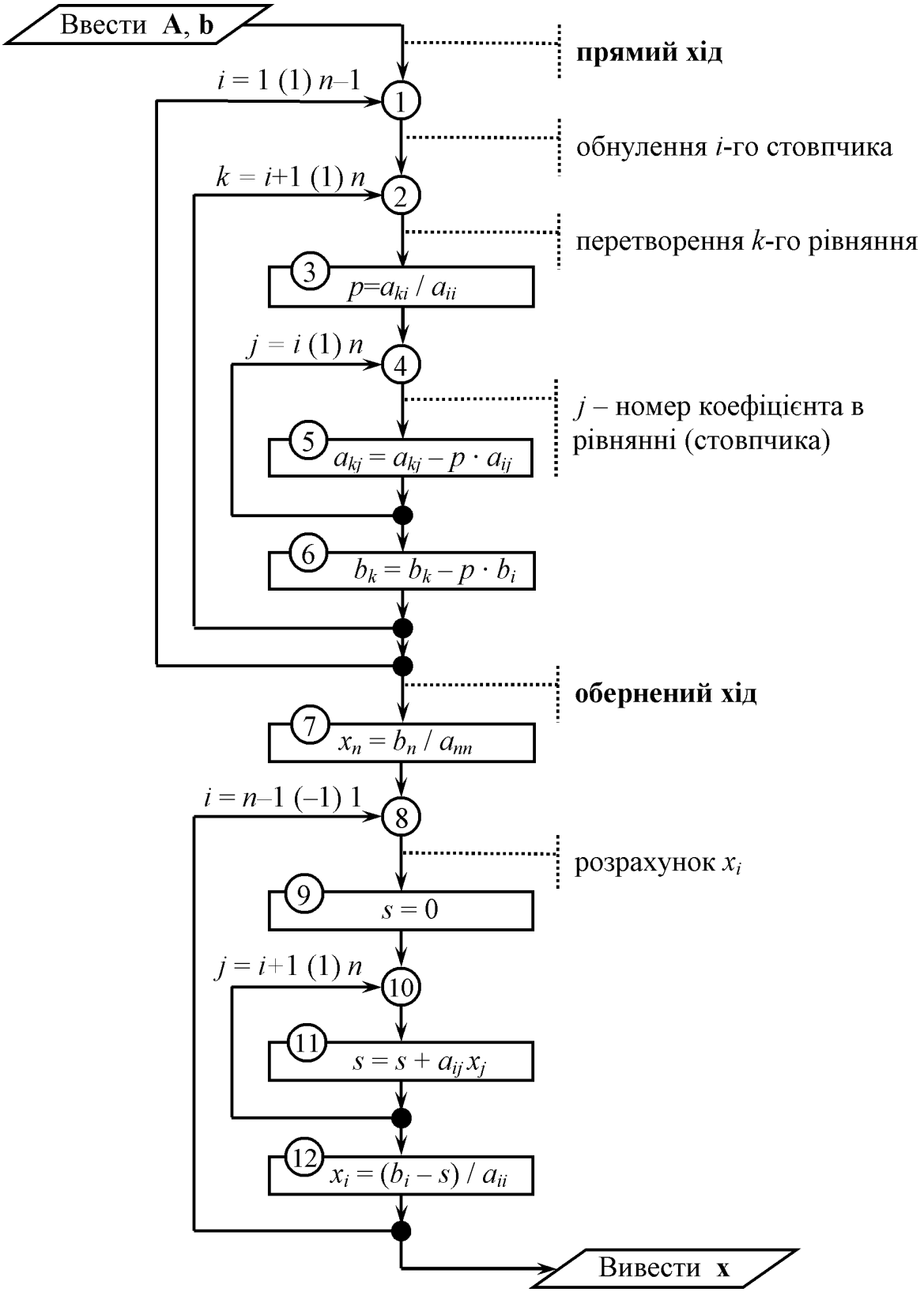


Рис. 8.1. Схема алгоритму метода Гауса для розв’язання СЛАР

Слід зауважити, що при всіх діленнях в формулах (1), (2) в ролі дільників виступають діагональні (ведучі) елементи. Основною проблемою при реалізації метода Гауса стає можливе ділення на нуль.

Для запобігання нульових ведучих елементів на i-ій стадії прямого ходу перед обнуленням i-го стовпчика застосовується перестановка рівнянь. В нижній частині i-го стовпчика (із числа елементів, що знаходяться під головною діагоналлю) вибирається елемент a_{mi}, максимальний за абсолютною величиною. Саме він служитиме ведучим елементом. Потім m-те та i-те рівняння міняються місцями, так що ведучий елемент стає діагональним.

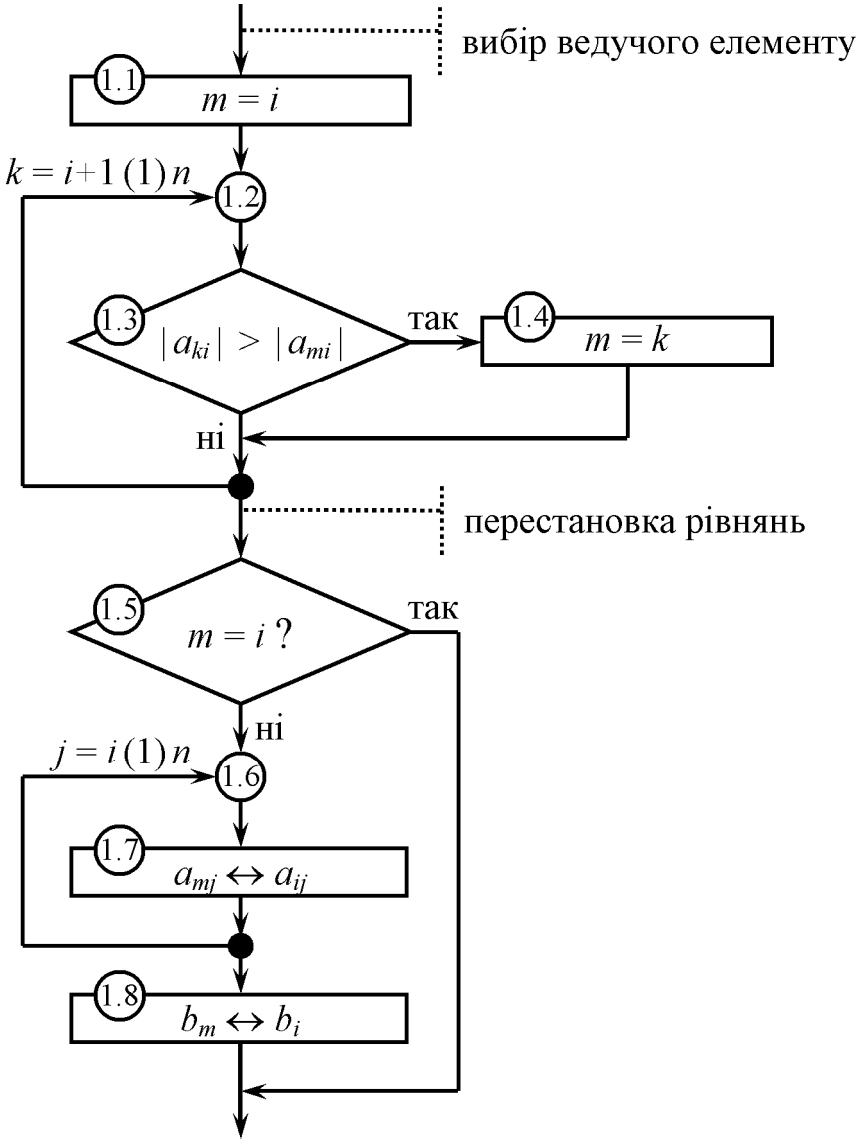


Рис. 8.2. Перестановка рівнянь перед обнуленням i-го стовпчика при застосуванні метода Гауса

Можна показати, що рівність нулю ведучого елементу a_{ii} на i-ій стадії прямого ходу методу Гауса з перестановками свідчить про те, що матриця системи вироджена.

Схема алгоритму методу Гауса наведена на рис. 8.1 – 8.2.

Завдання

1. Складіть процедуру для розв'язання СЛАР методом Гауса згідно зі схемою алгоритму на рис. 8.1. Передбачте в ній додатковий вихідний параметр – код помилки. Йому буде присвоєно одне певне значення (скажімо, 0), якщо процедура розв'язання пройшла успішно і помилок немає, і інше (скажімо, деяке від'ємне число), якщо розв'язок не здобуто (наприклад, система вироджена).

```
SUB Gauss(n,A(2),B(1),X(1),ErrCode)
'
' ------------------------------------------------------
' Розв'язання СЛАР методом Гауса.
'
' Вхідні параметри:
'    n      – порядок системи;
'    A[n,n] – матриця СЛАР;
'    B[n]   - вектор правої частини;
'             після виходу з процедури значення елементів
'             масивів A[] і B[] змінюються;
' Вихідні параметри:
'    X[n]   - вектор розв'язку;
'    ErrCode – код помилки:
'             ErrCode=0 якщо процедура завершилася успішно,
'             ErrCode<0 якщо матриця СЛАР вироджена.
' ------------------------------------------------------
'
...
END SUB
```

Налагоджуйте вашу процедуру `Gauss` поступово. В першому варіанті не включайте до неї фрагмент, що здійснює перестановку рівнянь (рис. 8.2), а коду помилки `ErrCode` просто присвойте нульове значення.

2. Складіть також окремі процедури:

   - для введення коефіцієнтів СЛАР (з клавіатури, файлу або безпосередньо в тексті програми – як ви вважаєте за доцільне);
   - для друку коефіцієнтів СЛАР на екран та/або в файл;
   - для друку заданого вектору, що буде застосовуватися для вектора розв'язку СЛАР **x** або вектора нев'язок **r = Ax – b**.

   Потурбуйтеся, щоб результати, що виводяться, мали вигляд охайної таблиці.

   Завжди починайте виконання вашої програми з введення коефіцієнтів вашої СЛАР і безпосередньо після цього, до початку будь-яких обчислень – негайного їх друку.

3. Введіть в процедуру `Gauss` проміжний друк коефіцієнтів СЛАР на кожній *i*-ій стадії прямого ходу, після обнулення *i*-го стовпчика. Скористайтеся для цього вищезазначеною процедурою. Її виклик буде останнім оператором в тілі циклу по *i* прямого ходу. Так ви зможете слідкувати за стадіями перетворення матриці СЛАР в трикутну – по завершенню кожної з них повинен обнулюватися наступний стовпчик під головною діагоналлю.

   Отримайте розв'язок **x** першої задачі. Надрукуйте його. Також для перевірки отриманого результату обчисліть і надрукуйте вектор нев'язок **r = Ax – b** , який у разі точного розв'язку має бути нульовим. Зауважте на те, що після виконання процедури `Gauss` значення елементів масивів **A** та **b** змінюються, тож в головній програмі заздалегідь потрібно зробити їх копії в пам'яті комп'ютера.

4. Спробуйте отримати розв'язок другої задачі. Прослідкуйте за перетвореннями матриці СЛАР в ході розв'язання. Поясніть отримані результати та труднощі, на які ви натрапили.

5. Включіть до процедури `Gauss` перед обнуленням *i*-го стовпчика (між блоками 1 та 2) фрагмент, що переставляє рівняння (рис. 8.2), і одразу ж за цим – проміжний друк коефіцієнтів СЛАР. Таким чином, на кожній стадії коефіцієнти СЛАР будуть виводитися двічі – після перестановки рівнянь і після обнулення відповідного стовпчика. Поясніть отримані результати.

6. Спробуйте отримати розв'язок третьої задачі. Прослідкуйте за перетвореннями матриці СЛАР в ході розв'язання. Поясніть отримані результати та труднощі, на які ви натрапили.

7. Основною причиною аварійної зупинки в процедурі `Gauss` є спроба ділення на нуль в формулах (1), (2), коли в ролі дільників виступають діагональні (ведучі) елементи. Тому якщо після перестановки рівнянь виявиться, що діагональний елемент $a_{ii} = 0$ (а це означатиме, що всі елементи i-го стовпчика від діагоналі і нижче дорівнюють нулю), необхідно перервати подальші розрахунки, присвоїти коду помилки `ErrCode` від'ємне значення $-i$, яке означатиме, що матриця СЛАР вироджена, причому це з'ясувалося при перевірці саме i-го діагонального елементу, після чого вийти із процедури `Gauss`. Включіть до процедури таку перевірку перед блоками 2 та 7.

   Передбачте в головній програмі після повернення з процедури `Gauss` перевірку параметра `ErrCode` і в залежності від його значення друкуйте або розв'язок і нев'язки системи, або повідомлення про виродженість.

Стислі теоретичні відомості (продовження)

Б. Число обумовленості матриці

При чисельному розв'язанні СЛАР замість точного розв'язку системи $\mathbf{Ax} = \mathbf{b}$ ми фактично, внаслідок округлень, отримуємо наближений розв'язок $\mathbf{x} + \mathbf{\Delta x}$ системи $(\mathbf{A}+\mathbf{\Delta A})(\mathbf{x}+\mathbf{\Delta x}) = (\mathbf{b} + \mathbf{\Delta b})$. В деяких випадках, навіть при невеликому збуренні коефіцієнтів СЛАР, наближений розв'язок може істотно відсуватися від точного (рис. 8.3).

В геометричному сенсі недолік таких систем полягає в тому, що гіперплощини, що описуються кожним з рівнянь, перетинаються під малими кутами, тобто матриця СЛАР «близька» до виродженої.

Для характеристики «близькості» чи «віддаленості» матриці СЛАР від виродженої вводять поняття *обумовленості* (conditionality) матриці. *Число обумовленості* $\mathrm{cond}(\mathbf{A})$ матриці $\mathbf{A}$ показує, наскільки можуть зрости відносні похибки розв'язку СЛАР при наявності відносних похибок у значеннях правих частин і елементів матриці:

$$\frac{\| \mathbf{\Delta x} \|}{\| \mathbf{x} \|} \le \mathrm{cond}(\mathbf{A}) \cdot \left(\frac{\| \mathbf{\Delta b} \|}{\| \mathbf{b} \|} + \frac{\| \mathbf{\Delta A} \|}{\| \mathbf{A} \|} \right).$$

Для будь-якої матричної норми число обумовленості не менше за 1.

Кажуть, що матриці з великим числом обумовленості є «погано обумовленими», а з таким, що лише незначно перевищує одиницю – «добре обумовленими».

Для погано обумовлених СЛАР малі збурення їх коефіцієнтів призводять до значної зміни розв'язку. Але в цьому випадку підстановка наближеного розв'язку $\mathbf{x}+\Delta\mathbf{x}$ в точну систему не призводить до великих нев'язок, і їх малість не гарантує малості похибки $\Delta\mathbf{x}$. Зокрема, якщо $\mathrm{cond}(\mathbf{A}) \geq 1/\varepsilon$, де ε – відносна похибка визначення коефіцієнтів СЛАР, (яка в принципі не може бути меншою за машинне епсілон), будь-який розв'язок СЛАР втрачає сенс і з практичної точки зору її можна вважати виродженою.

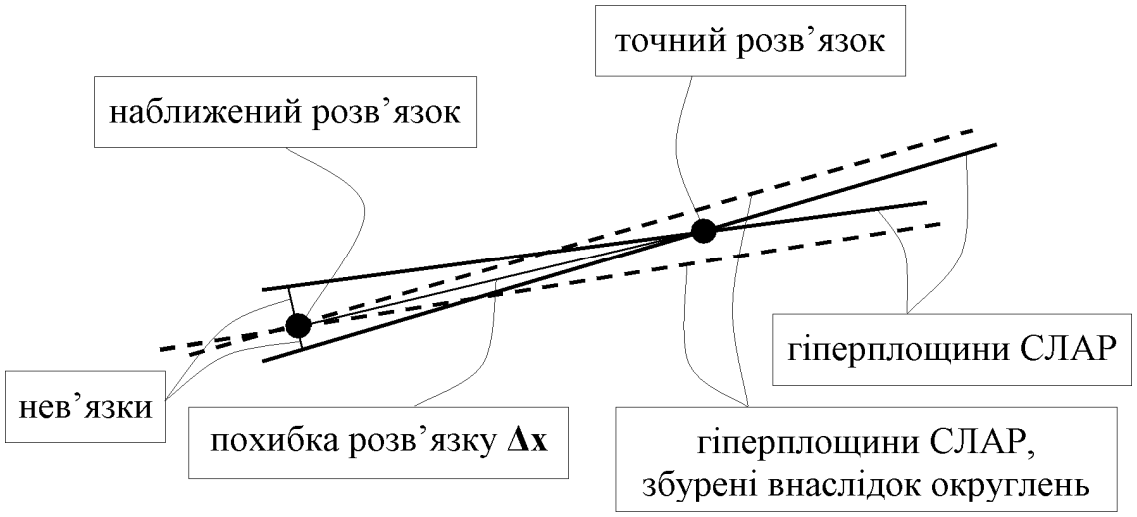


Рис. 8.3. До поняття обумовленості СЛАР

Можна показати, що число обумовленості дорівнює

$$\mathrm{cond}(\mathbf{A}) = \| \mathbf{A} \| \cdot \| \mathbf{A}^{-1} \| . \qquad (3)$$

Нажаль, практичне застосування критерію (3) обмежено тим, що обсяг обчислень, необхідних для його обрахунку, значно перевищує обсяг обчислень для власне розв'язання СЛАР. Тому часто використовують інші критерії, які дозволяють оцінити принаймні порядок числа обумовленості.

Один з таких критеріїв базується на величині детермінанту матриці, який у вироджених матриць дорівнює нулеві. Слід одразу відмітити, що сама по собі мала чи велика абсолютна величина детермінанту не може служити ознакою доброї чи поганої обумовленості матриці, оскільки матрицю можна помножити на довільне число q і змінити детермінант в q^n разів: $\det(q\mathbf{A}) = q^n \cdot \det(\mathbf{A})$.

Тому спочатку віднормуємо кожне рівняння таким чином, щоб сума квадратів його коефіцієнтів дорівнювала б одиниці:

$$\frac{a_{i1}}{\alpha_i}x_1 + \frac{a_{i2}}{\alpha_i}x_2 + \ldots + \frac{a_{in}}{\alpha_i}x_n = \frac{b_i}{\alpha_i}, \quad i = 1\ldots n,$$

де

$$\alpha_i = \sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \ldots + {a_{in}}^2} = \sqrt{\sum_{j=1}^{n} {a_{ij}}^2}. \tag{4}$$

Детермінант матриці такої СЛАР дорівнює

$$v = \det\begin{pmatrix} \frac{a_{11}}{\alpha_1} & \frac{a_{12}}{\alpha_1} & \ldots & \frac{a_{1n}}{\alpha_1} \\ \frac{a_{21}}{\alpha_2} & \frac{a_{22}}{\alpha_2} & \ldots & \frac{a_{2n}}{\alpha_2} \\ \ldots & \ldots & \ldots & \ldots \\ \frac{a_{n1}}{\alpha_n} & \frac{a_{n2}}{\alpha_n} & \ldots & \frac{a_{nn}}{\alpha_n} \end{pmatrix} = \frac{\det(\mathbf{A})}{\omega}, \tag{5}$$

де

$$\omega = \alpha_1\alpha_2\ldots\alpha_n = \sqrt{\prod_{i=1}^{n}\sum_{j=1}^{n} {a_{ij}}^2}. \tag{6}$$

Покажемо, що величину $|v|$, на відміну від $|\det(\mathbf{A})|$, можна використовувати для оцінки обумовленості матриці.

Мірою малості кутів, під якими перетинаються гіперплощини, що описуються кожним з рівнянь СЛАР, може служити об’єм косого паралелепіпеду, ребрами якого є n одиничних нормальних векторів до цих гіперплощин (рис. 8.4).

Компоненти одиничного нормального вектору до i-ї гіперплощини становлять

$$\frac{\left(a_{i1},\ a_{i2},\ \ldots,\ a_{in}\right)^{\mathrm{T}}}{\alpha_i},$$

де α_i дається виразом (4).

Абсолютна величина детермінанту (5), який містить компоненти цих векторів, тобто $|v|$, саме визначає об’єм такого паралелепіпеду.

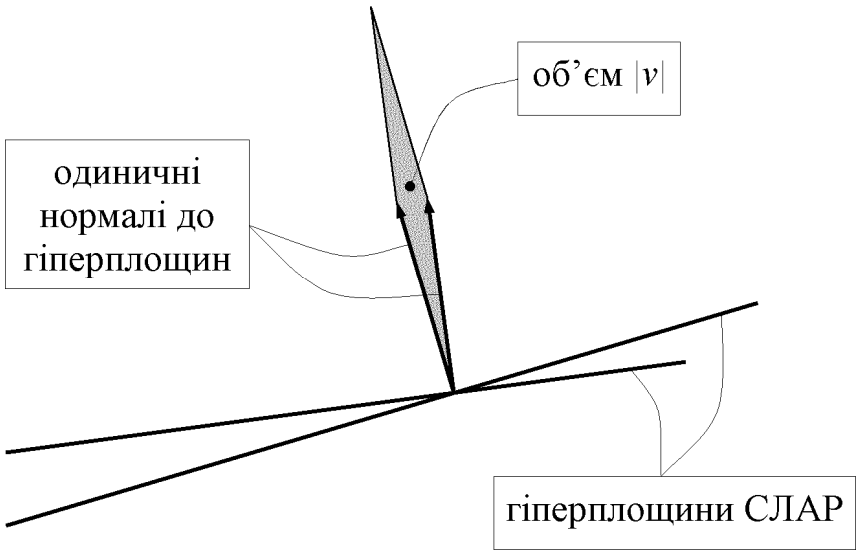


Рис. 8.4. Косий паралелепіпед, ребрами якого є одиничні нормалі до гіперплощин СЛАР

Якщо $v = 0$, матриця **A** вироджена. Значення $v = \pm 1$ відповідають ідеальній ситуації, коли всі гіперплощини взаємно-перпендикулярні. Проміжним ситуаціям відповідають випадки $0 < |v| < 1$.

Для грубої оцінки можна прийняти $\mathrm{cond}(\mathbf{A}) \sim 1/|v|$

Додаткове завдання

8. Доповніть процедуру `Gauss` обчисленням детермінанту матриці **A**. Зауважте, що його величина не змінюється при перетвореннях матриці, що здійснюються в ході обнулення *i*-го стовпчика. Лише перестановка *m*-го та *i*-го рівнянь може змінити його знак: він інвертується, якщо одне з чисел *m*, *i* – парне, а друге – непарне (тобто *m*+*i* є непарним).

   Оскільки детермінант отриманої після прямого ходу трикутної матриці дорівнює добутку її діагональних елементів, його значення можна здобути перемноженням ведучих елементів безпосередньо на етапі прямого ходу. (Не забудьте про елемент a_{nn} !). Можливо, буде потрібно інвертувати його знак внаслідок попередніх перестановок рядків.

9. Передбачте на початку процедури `Gauss` (коли матриця A ще не потерпала жодного перетворення) обчислення величини ω .

10. В оформленні самої процедури `Gauss` передбачте два додаткових вихідних параметра – значення детермінанту матриці та її числа обумовленості.

    В основній програмі виведіть ці два числа поруч із розв'язком системи.

11. Розв'яжіть за допомогою модернізованої таким чином процедури `Gauss` всі три задачі і оцініть обумовленість кожної з них.

12. В усіх трьох задачах введіть невелике збурення у праві частини СЛАР. Змініть один або декілька з компонент вектора **b** приблизно на 1%. Отримайте розв'язки систем та порівняйте їх з розв'язками незбурених систем. Зробіть висновки, зважаючи на число обумовленості матриці кожної системи.

Варіанти для самостійної роботи

Варіант	Задачі
1	А Г Ж
2	А Г К
3	А Г Л
4	А Д К
5	А Д Л
6	А Е Ж
7	А Е К
8	А Е Л

Варіант	Задачі
9	Б Г Ж
10	Б Г К
11	Б Г Л
12	Б Д К
13	Б Д Л
14	Б Е Ж
15	Б Е К
16	Б Е Л

Варіант	Задачі
17	В Г Ж
18	В Г К
19	В Г Л
20	В Д К
21	В Д Л
22	В Е Ж
23	В Е К
24	В Е Л

А)

$$\begin{aligned} 2x_1 + 2x_2 - x_3 + x_4 &= 3 \\ -3x_1 + 3x_3 &= -9 \\ -x_1 + 3x_2 + 3x_3 + 2x_4 &= -7 \\ x_1 + 4x_4 &= 4 \end{aligned}$$

Б)

$$\begin{aligned} 2x_1 + 3x_3 + x_4 &= 20 \\ -4x_1 + 3x_2 - 4x_3 - 2x_4 &= -34 \\ 4x_1 + 7x_2 + 9x_3 + x_4 &= 48 \\ 5x_1 + 7x_2 + 8x_4 &= 97 \end{aligned}$$

В)

$$\begin{aligned} 13x_1 - 5x_2 - 12x_3 &= 33 \\ -12x_1 + 5x_2 &= -19 \\ 4x_1 - x_2 - 22x_3 &= 29 \end{aligned}$$

Г)

$$\begin{aligned} -7x_1 - 6x_2 - 6x_3 + 6x_4 &= 144 \\ 7x_1 + 6x_2 + 8x_3 - 13x_4 &= -170 \\ 4x_1 + 17x_2 - 16x_3 + 10x_4 &= 21 \\ -5x_1 + 18x_2 + 19x_3 &= -445 \end{aligned}$$

Д)

$$\begin{aligned} 3x_2 - x_3 &= 7 \\ 9x_1 + 24x_2 + x_3 &= 20 \\ 21x_1 - x_2 - 16x_3 &= 63 \end{aligned}$$

Е)

$$\begin{aligned} -5x_1 + 7x_3 &= 109 \\ 4x_1 - 24x_2 + x_4 &= 168 \\ 3x_1 + 12x_2 - 7x_3 - 23x_4 &= -193 \\ -2x_1 + 42x_2 + 37x_3 - 21x_4 &= -95 \end{aligned}$$

Ж)

$$\begin{aligned} -2x_1 + 4x_2 + 7x_3 &= 42 \\ -7x_1 - 6x_2 - 6x_3 &= 7 \\ 11x_1 - 2x_2 - 8x_3 &= -91 \end{aligned}$$

К)

$$\begin{aligned} 5x_1 - 7x_3 &= -123 \\ -x_1 + 6x_2 + x_4 &= 60 \\ 2x_1 - 6x_2 - 4x_3 - 5x_4 &= -108 \\ -6x_1 - 6x_2 + 15x_3 + 7x_4 &= 159 \end{aligned}$$

Л)

$$\begin{aligned} 3x_1 - 2x_2 - 7x_3 - x_4 &= 2 \\ 7x_1 - 10x_2 - 5x_3 + x_4 &= 28 \\ 4x_1 - 15x_3 - 9x_4 &= -21 \\ -8x_1 + 8x_2 + 13x_3 + 4x_4 &= -11 \end{aligned}$$

Контрольні запитання

1. Скільки множень/ділень з плаваючою точкою потребує виконання алгоритму Гауса?
2. Покажіть, що алгоритм Гауса без перестановок не приводить до успіху тоді, коли один з головних мінорів вихідної матриці СЛАР дорівнює нулеві.
3. Проаналізуйте, як накопичуються похибки округлення в коефіцієнтах СЛАР при її перетвореннях за методом Гауса (блок 5). Доведіть. що саме перестановка рівнянь призводить до появи таких Гаусових множників p, які мінімізують можливе накопичення похибок округлення.
4. Нехай потрібно розв'язати декілька систем виду $\mathbf{Ax} = \mathbf{b}$ з однаковими матрицями $\mathbf{A}$ і різними правими частинами $\mathbf{b}$. Запропонуйте відповідну модифікацію алгоритму Гауса.
5. На базі алгоритму Гауса запропонуйте спосіб інвертування (знаходження оберненої) матриці.
6. Чому детермінант трикутної матриці дорівнює добутку її діагональних елементів?
7. Чому відносна похибка визначення коефіцієнтів СЛАР в принципі не може бути меншою за машинне епсілон?
8. Що таке норма вектора?
9. Що таке норма матриці?
10. Що таке число обумовленості матриці?
11. З якими складностями пов'язане обчислення числа обумовленості матриці за точною формулою (3)?

Лабораторна робота № 9.

Розв'язання систем лінійних алгебраїчних рівнянь. Ітераційні методи Якобі та Гауса-Зейделя

Мета роботи: вивчення алгоритмів та налагодження програм для розв'язання систем лінійних алгебраїчних рівнянь ітераційними методами Якобі та Гауса-Зейделя.

Що зробити: з'ясувати факт збіжності чи розбіжності ітераційних процесів Якобі та Гауса-Зейделя. У випадку збіжності знайти розв'язок СЛАР та перевірити його, підставляючи в СЛАР отримані розв'язки і обраховуючи нев'язки. Визначити порядок збіжності ітераційного процесу.

Стислі теоретичні відомості

При ітераційному розв'язанні системи n лінійних алгебраїчних рівнянь (СЛАР, див. постановку задачі в лабораторній роботі № 8) кожне з її рівнянь слід перетворити до виду:

$$
\begin{aligned}
x_1 &= [b_1 - (a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n)] / a_{11} \\
x_2 &= [b_2 - (a_{21}x_1 + a_{23}x_3 + \ldots + a_{2n}x_n)] / a_{22} \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
x_i &= [b_i - (a_{i1}x_1 + \ldots + a_{i,i-1}x_{i-1} + a_{i,i+1}x_{i+1} + \ldots + a_{in}x_n)] / a_{ii} \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
x_n &= [b_n - (a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{n,n-1}x_{n-1})] / a_{nn}
\end{aligned}
$$

або

$$
x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j \right), \quad (i = 1, 2, \ldots, n). \tag{1}
$$

Метод Якобі полягає в ітеративному застосуванні останньої формули. При підстановці до неї змінних використовуються їх значення, що отримані при попередній ітерації.

Для запуску ітераційного процесу необхідно задатися початковим наближенням. Можна показати, що вибір початкового наближення не впливає на характер збіжності процесу, тому можна вважати $\mathbf{x}^{(0)} = \mathbf{0}$ або $\mathbf{x}^{(0)} = \mathbf{b}$.

Закінчити обчислення можна при виконанні умови

$$\| \mathbf{x}^{(m+1)} - \mathbf{x}^{(m)} \| < \varepsilon , \tag{2}$$

де m – номер наближення, а ε – деяка мала величина. Норму вектора можна брати будь-яку, наприклад для ∞-норми

$$\| \mathbf{v} \|_{\infty} = \max_i \{ | v_i | \} \tag{3}$$

умова (2) запишеться у вигляді

$$\delta_{\max} = \max_i \left\{ \left| x_i^{(m+1)} - x_i^{(m)} \right| \right\} < \varepsilon .$$

Схема алгоритму метода Якобі з використанням такої норми наведена на рис. 9.1.

Завдання

1. Складіть програму для розв’язання СЛАР ітераційним методом Якобі. Скористайтеся розробленими вами в ході виконання лабораторної роботи № 8 процедурами для введення коефіцієнтів та для виведення розв’язку СЛАР.
2. Доповніть програму лічильником числа ітерацій та проміжним друком невідомих змінних після кожної ітерації і загальною похибкою наближення $\delta_{\max}$. Потурбуйтеся про охайність виведення результатів.
3. Іноді ітераційний процес може розбігатися. З метою гарантованого завершення програми навіть у випадку незбіжності до розв’язку, запровадьте в програму обмеження на максимальну кількість ітерацій. Передбачте виведення відповідного повідомлення про незбіжність ітераційного процесу, аналогічно тому, як ви це робили при виконанні п. 4 завдання лабораторної роботи № 5.
4. Спробуйте отримати розв’язок завдання вашого варіанту.
5. В разі збіжності ітераційного процесу для перевірки отриманого результату обчисліть і надрукуйте вектор нев’язок $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$.

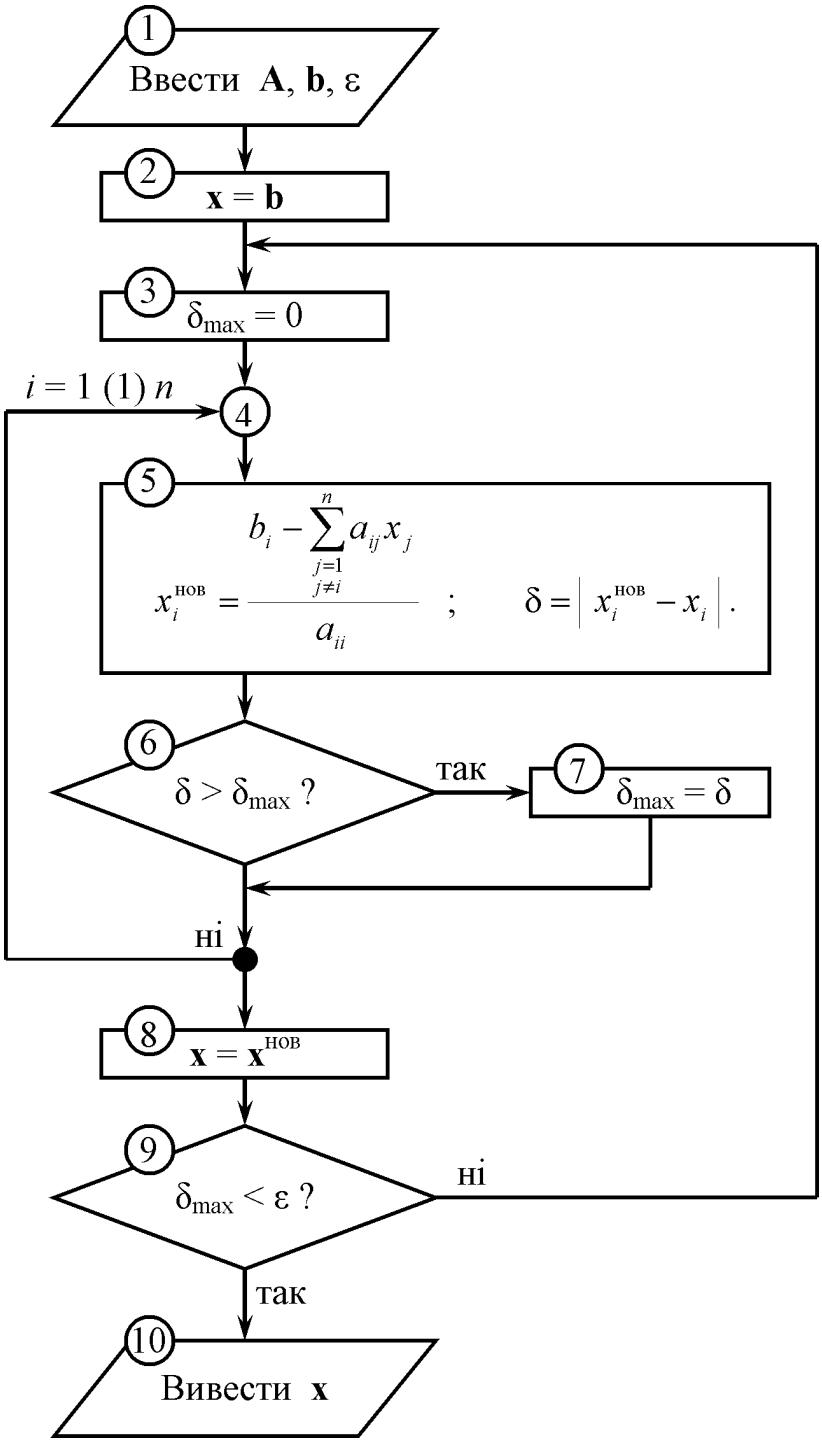


Рис. 9.1. Схема алгоритму ітераційного методу Якобі для розв’язку СЛАР

6. Дослідіть, аналогічно тому, як ви це робили при виконанні п. 6 завдання лабораторної роботи № 5, як похибки δ_{max} поточного наближення до розв'язку залежать від номера ітерації. На основі цих даних з'ясуйте порядок збіжності методу Якобі і порівняйте його з порядком збіжності методу простих ітерацій для розв'язання рівнянь з одним невідомим.

7. Ітераційний метод Гауса-Зейделя відрізняється від методу Якобі лише тим, що уточнене значення x_1 одразу ж використовується для обчислення x_2. Далі по нових значеннях x_1 та x_2 обчислюють x_3 і т.д. Фактично, від ітерації Якобі

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(m)} \; - \sum_{j=i+1}^{n} a_{ij} x_j^{(m)} \right), \quad i = 1,\ 2,\ \ldots,\ n \qquad (4)$$

ітерація Гауса-Зейделя

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(m)} \right), \quad i = 1,\ 2,\ \ldots,\ n \qquad (5)$$

відрізняється лише номером наближення (m+1 замість m), що використовується при обрахуванні першої суми.

Модифікуйте вашу програму для реалізації цього методу (див. рис. 9.2). Зверніть увагу, що програма має спроститися. Зокрема, вам не знадобиться один із задекларованих масивів $\mathbf{x}^{нов}$. Розв'яжіть задачу вашого варіанту та порівняйте розв'язок СЛАР і кількість здійснених ітерацій з отриманими раніше результатами.

8. З'ясуйте порядок збіжності методу Гауса-Зейделя.

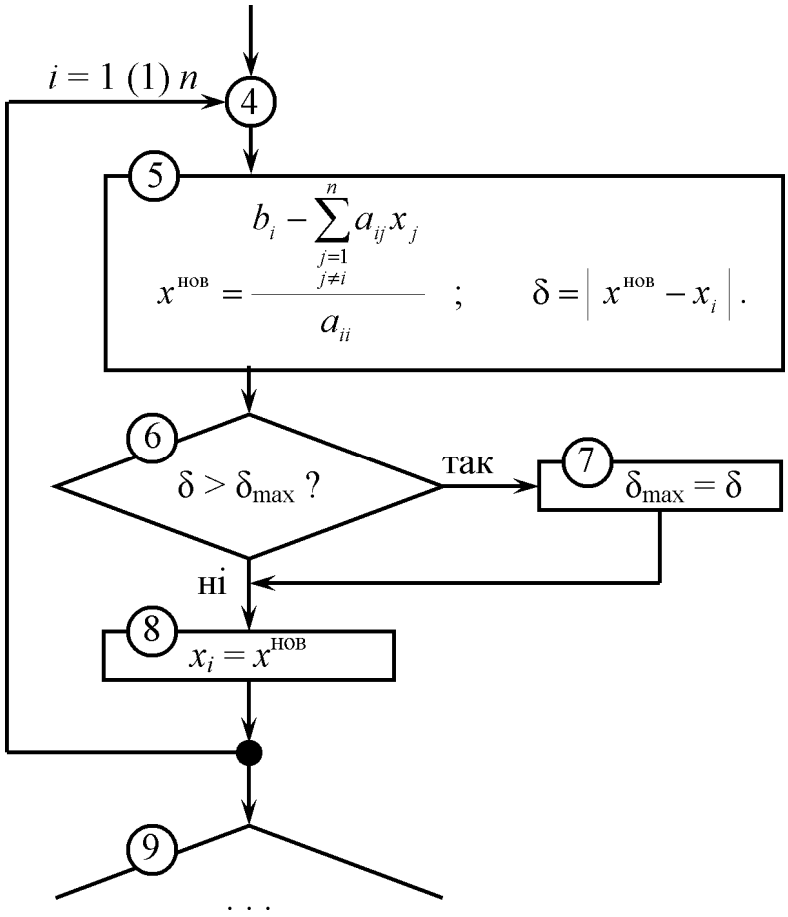


Рис. 9.2. Модифікація схеми алгоритму для методу Гауса-Зейделя

Варіанти для самостійної роботи

Варіант	Задачі
1	А $a = 2$, $b = 3$
2	Б $a = 10$, $b = 3$
3	В $a = 2$, $b = 5$
4	Г $a = 1$, $b = -1$
5	Д $a = 4$, $b = 4$
6	Е $a = 3$, $b = -2$
7	А $a = 3$, $b = -1$
8	Б $a = 8$, $b = 7$
9	В $a = -3$, $b = 4$
10	Г $a = 3$, $b = 2$
11	Д $a = -3$, $b = 3$
12	Е $a = 5$, $b = 2$

Варіант	Задачі
13	А $a = -1$, $b = 1$
14	Б $a = -4$, $b = 2$
15	В $a = 4$, $b = -2$
16	Г $a = 3$, $b = 4$
17	Д $a = -6$, $b = 2$
18	Е $a = 7$, $b = 4$
19	А $a = -2$, $b = 5$
20	Б $a = 5$, $b = -1$
21	В $a = 3$, $b = 3$
22	Г $a = -4$, $b = 5$
23	Д $a = 10$, $b = 1$
24	Е $a = 9$, $b = -6$

А) $\begin{pmatrix} 2a & a & 0 & 0 \\ b & 2a & a & 0 \\ 0 & b & 2a & a \\ 0 & 0 & b & 2a \end{pmatrix} \mathbf{x} = \begin{pmatrix} a \\ b \\ a \\ b \end{pmatrix}$

Б) $\begin{pmatrix} a & b & 0 & 0 \\ b & a & b & 0 \\ b & b & a & b \\ b & b & b & a \end{pmatrix} \mathbf{x} = \begin{pmatrix} 5 \\ 6 \\ 7 \\ 8 \end{pmatrix}$

В) $\begin{pmatrix} 4a & 0 & a & 0 \\ 0 & 3a & 0 & b \\ a & 0 & b & 0 \\ 0 & b & 0 & 2b \end{pmatrix} \mathbf{x} = \begin{pmatrix} 15 \\ 10 \\ 5 \\ 0 \end{pmatrix}$

Г) $\begin{pmatrix} 3a & b & 0 & 0 \\ a & 2b & b & 0 \\ 0 & a & 2a & b \\ 0 & 0 & a & 3b \end{pmatrix} \mathbf{x} = \begin{pmatrix} 2b \\ 2a \\ 2b \\ 2a \end{pmatrix}$

Д) $\begin{pmatrix} a & 2 & 0 & 0 \\ 2 & b & 2 & 0 \\ 0 & 2 & b & 2 \\ 0 & 0 & 2 & a \end{pmatrix} \mathbf{x} = \begin{pmatrix} b \\ 2b \\ 3b \\ 4b \end{pmatrix}$

Е) $\begin{pmatrix} b & 0 & 1 & 0 \\ a & 4b & 0 & a \\ 1 & 0 & 3b & 0 \\ 0 & 0 & 5 & 2b \end{pmatrix} \mathbf{x} = \begin{pmatrix} a \\ 10 \\ 5 \\ a \end{pmatrix}$

Контрольні запитання

1. Зверніть увагу, що за умови діагональної матриці системи метод Якобі дає розв'язок після першої ж ітерації. В загальному ж випадку збіжність методу забезпечується, якщо діагональні елементи матриці переважають над недіагональними. Зокрема, достатньою (хоча й не необхідною) умовою збіжності є виконання нерівностей

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

причому хоча б одна з нерівностей повинна виконуватися строго.

Аналогічним чином дослідіть ітерацію Гауса-Зейделя і сформулюйте достатній критерій збіжності цього методу.

2. Покажіть, що формула ітерацій Якобі (4) може бути записана в термінах нев'язок m-го наближення $\mathbf{r}^{(m)} = \mathbf{A}\mathbf{x}^{(m)} - \mathbf{b}$ як

$$x_i^{(m+1)} = x_i^{(m)} - \frac{r_i^{(m)}}{a_{ii}}, \quad i = 1,\ 2,\ ...,\ n.$$

Запропонуйте також аналогічний запис для формули ітерацій Гауса-Зейделя (5).

3. Підрахуйте кількість множень, що виконуються на одній ітерації методу Якобі та Гауса-Зейделя. Порівняйте з кількістю аналогічних дій в методі Гауса, який ви досліджували в лабораторній роботі № 8. За скільки ітерацій повинен збігатися ітераційний метод, щоб його застосування було доцільним в порівнянні з методом Гауса?

4. Обговоріть можливість та доцільність використання в умові (2) інших Гьольдерових норм виду

$$\| \mathbf{v} \|_p = \sqrt[p]{\sum_i |v_i|^p},$$

зокрема 1-норми (норми-суми)

$$\| \mathbf{v} \|_1 = \sum_i |v_i|$$

або 2-норми (евклідової норми)

$$\| \mathbf{v} \|_2 = \sqrt{\sum_i (v_i)^2}.$$

Вочевидь, ∞-норма виду (3) (норма-максимум) є граничним випадком $p \to \infty$.

Лабораторна робота № 10.

Власні вектори і власні числа матриць з елементами-дійсними числами

Мета роботи: наочна геометрична інтерпретація власних векторів і власних чисел матриць 2×2 з дійсними елементами.

Що зробити: в двовимірному просторі (площині) здійснити прямий перебор (з певним малим кроком) одиничних векторів **x** всіх можливих напрямків і для кожного з них порівняти напрямок результату добутку $\mathbf{y} = \mathbf{Ax}$, де **A** – матриця 2×2, з напрямком самого **x**. Визначити таким чином власні вектори і власні числа матриці **A** та порівняти їх з величинами, розрахованими аналітично за допомогою характеристичного рівняння.

Стислі теоретичні відомості

Нехай **A** – квадратна матриця, а **x** – вектор відповідного порядку. Результатом перетворювання вектору **x** за допомогою матриці **A**

$$\mathbf{y} = \mathbf{Ax}$$

є вектор **y**, напрямок якого, у загальному випадку, не співпадає з напрямком вектору **x**.

Можна поставити задачу: із усіх можливих векторів **x** знайти такі, які після перетворювання за допомогою цієї матриці, дають результат, колінеарний з початковим **x** , тобто виконується співвідношення

$$\mathbf{Ax} = \lambda\mathbf{x} , \qquad (1)$$

де λ – деякий скалярний множник. Вектор, що задовольняє співвідношенню (1), називають *власним вектором* (eigenvector), а число λ – відповідним йому *власним числом* (eigenvalue) матриці **A**. Зауважимо, що власні вектори визначаються з точністю до довільного множника, тобто правильніше було б говорити про «власні напрямки» матриці.

Геометричний сенс співвідношення (1) для матриць 2×2 є зовсім простим. Переведемо декартові координати векторів **x** та **y** в полярні:

$$(x_1 , x_2) \to (x , \varphi); \qquad (y_1 , y_2) \to (y , \psi),$$

де x , y – довжини векторів, а φ , ψ – їх полярні кути. Якщо $\varphi = \psi$ (або $\varphi = \psi \pm \pi$), то **x** (а також будь-який вектор цього напрямку) є власним вектором, а відношення $\lambda = y / x$ є відповідним йому власним числом. (Якщо $\varphi = \psi \pm \pi$, то власному числу λ треба приписати знак мінус).

Співвідношення (1) можна переписати у вигляді системи лінійних алгебраїчних рівнянь

$$(\mathbf{A} - \lambda \mathbf{E})\, \mathbf{x} = \mathbf{0} , \tag{2}$$

де **E** – одинична матриця. Щоб ця система мала розв'язки, відмінні від тривіального **x** = **0**, необхідно, щоб її матриця була виродженою, тобто

$$\det (\mathbf{A} - \lambda \mathbf{E}) = 0 , \tag{3}$$

або, позначаючи через n порядок матриці,

$$\det \begin{pmatrix} a_{11} - \lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} - \lambda & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} - \lambda \end{pmatrix} = 0 .$$

Це – рівняння відносно λ (характеристичне рівняння), що дозволяє знайти власні числа. Після розкриття детермінанту ліва частина буде містити поліном n-го порядку відносно λ (так званий «характеристичний многочлен») , отже, характеристичне рівняння є алгебраїчним і має рівно n-коренів. (Частина з них, можливо, буде комплексними, навіть якщо всі елементи матриці **A** є дійсними числами.) Знайшовши їх і підставивши до (2), можна відшукати і відповідні їм власні вектори (які у випадку комплексних власних чисел також можуть мати комплексні компоненти).

Інший спосіб знайти власні вектори матриці (щоправда, лише дійсні) полягає в прямому переборі векторів **x** всіх можливих напрямків і порівнянні для кожного з них напрямку результату добутку **y** = **Ax** з напрямком самого **x**. Вочевидь, можна обмежитися всіма векторами **x** одиничної довжини. Їх кінці складуть одиничну сферу в n-вимірному просторі. На практиці, звичайно, доведеться обмежитися перебором великого, але конечного числа точок, що вкриватимуть одиничну сферу густою сіткою, причому чим густіше буде ця сітка, тим точніше буде отриманий результат.

Завдання

1. Нехай **A** – довільна матриця 2×2. Задамося в площині вектором **x** одиничної довжини направленим під кутом φ до горизонтальної осі. В полярних координатах цей вектор запишеться як $(1, \varphi)$, а його декартові компоненти, вочевидь, становитимуть $x_1 = \cos\varphi$ та $x_2 = \sin\varphi$. Помножимо матрицю **A** на вектор **x**. В результаті отримаємо деякий вектор **y** = **Ax**.

   Примусимо тепер кут φ обертатися з малим кроком δ від 0 до 2π (360º). Тоді вектор **x** опише коло одиничного радіусу. Воно буде геометричним місцем точок, що є кінцями всіх можливих одиничних векторів на площині. Вектор **y** при цьому також опише деяку замкнену криву. Зауважимо, що форма цієї кривої повністю обумовлена матрицею **A**.

   Запрограмуйте побудову цих кривих. Використайте дві процедури: одну для введення матриці **A** (з файлу чи клавіатури – на ваш власний вибір), а іншу для обчислення добутку **y** = **Ax**. Виведіть на екран також компоненти матриці **A**.

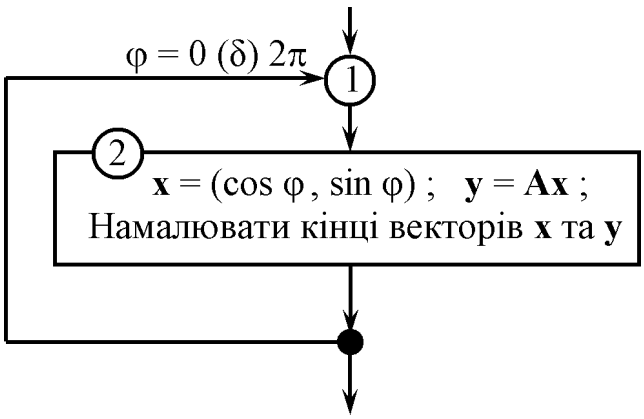


Рис. 10.1. Схема алгоритму перебору 2-компонентних векторів **x** одиничної довжини всіх можливих напрямків з кроком δ (за винятком процедур вводу-виводу)

Потурбуйтеся про те, щоб при геометричних побудовах на екрані застосовувалися однакові масштаби по горизонталі й вертикалі, інакше форми ваших кривих спотворяться, зокрема одиничне коло перетвориться на еліпс. Особливо уважно за масштабами треба слідкувати при використанні на комп'ютері екранних режимів з неквадратним пікселем.

2. Спробуйте виконати вашу програму з різними матрицями $\mathbf{A}_{(1)} - \mathbf{A}_{(4)}$ згідно з вашим варіантом. Впевніться, що крива, що її описує кінець вектора **y**, завжди є еліпсом, причому, у випадку, коли матриця **A** вироджена, він стягується у відрізок.

3. Скористайтеся прямим перебором одиничних векторів $\mathbf{x} = (\cos\varphi, \sin\varphi)$ всіх можливих напрямків для безпосереднього пошуку власних векторів матриці **A**.

   Для цього треба перевести декартові координати вектора $\mathbf{y} = \mathbf{A}\mathbf{x}$ в полярні:

   $$(y_1, y_2) \rightarrow (y, \psi),$$

   (для переводу координат зручно також скласти окрему процедуру) і порівняти кути φ та ψ. Якщо $\varphi = \psi$ (або $\varphi = \psi \pm \pi$), то **x** є власним вектором. Оскільки довжина **x** дорівнює 1, то довжина **y** дорівнюватиме відповідному йому власному числу. (Якщо $\varphi = \psi \pm \pi$, то власному числу треба приписати знак мінус).

   Для розв'язання поставленої задачі достатньо, щоб кінець одиничного вектора **x** описав не повне коло, а лише півколо, відповідно кут φ змінювався б з малим кроком δ від 0 до π (180º).

   Особливість обчислень полягає в тому, що внаслідок дискретного кроку по φ рівність $\varphi = \psi$ або $\varphi = \psi \pm \pi$ може виконуватися лише наближено. Фактично, треба робити перевірку:

   $$|\varphi - \psi| < \varepsilon \quad \text{або} \quad ||\varphi - \psi| - \pi| < \varepsilon, \tag{4}$$

   де ε – мале число, за порядком величини близьке до кроку δ.

   Поекспериментуйте з різними значеннями ε. Занадто велике ε призведе до того, що умова (4) буде виконуватися при декількох суміжних значеннях φ, і вектори **x** та **y** заметуть цілий сектор. Замале ж значення ε може призвести до того, що ви проскочите напрямок власного вектора, не помітивши його.

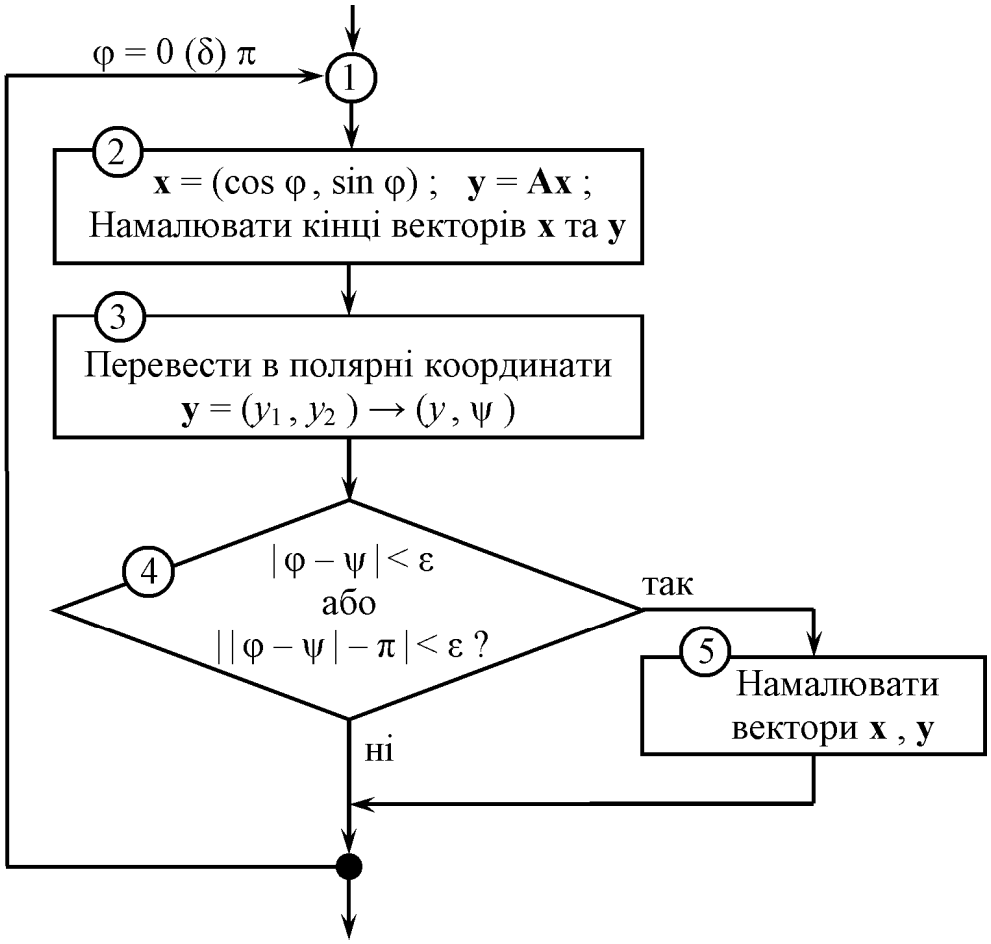


Рис. 10.2. Схема алгоритму пошуку власних векторів прямим перебором одиничних векторів всіх напрямків (за винятком процедур вводу-виводу)

4. Складіть для цих матриць характеристичні рівняння типу (3) та розрахуйте аналітично їхні власні числа λ_I, λ_{II} і власні вектори $\mathbf{x}_I$, $\mathbf{x}_{II}$, а також, якщо вони дійсні, то й кути θ_I, θ_{II}, які власні вектори утворюють з віссю абсцис.

   Виконайте вашу програму при симетричних та несиметричних, вироджених та невироджених матрицях **A.** Порівняйте результати, отримані за допомогою програми, з аналітичними розрахунками. Чи є якісні відмінності між результатами, отриманими для різних типів матриць? Чи можете ви сформулювати певні припущення щодо закономірностей, що спостерігаються? Спробуйте їх довести аналітично.

5. Впевніться, що для симетричних матриць власні числа λ_I, λ_{II} завжди є дійсними, а власні вектори $\mathbf{x}_I$, $\mathbf{x}_{II}$ – взаємно-ортогональними.

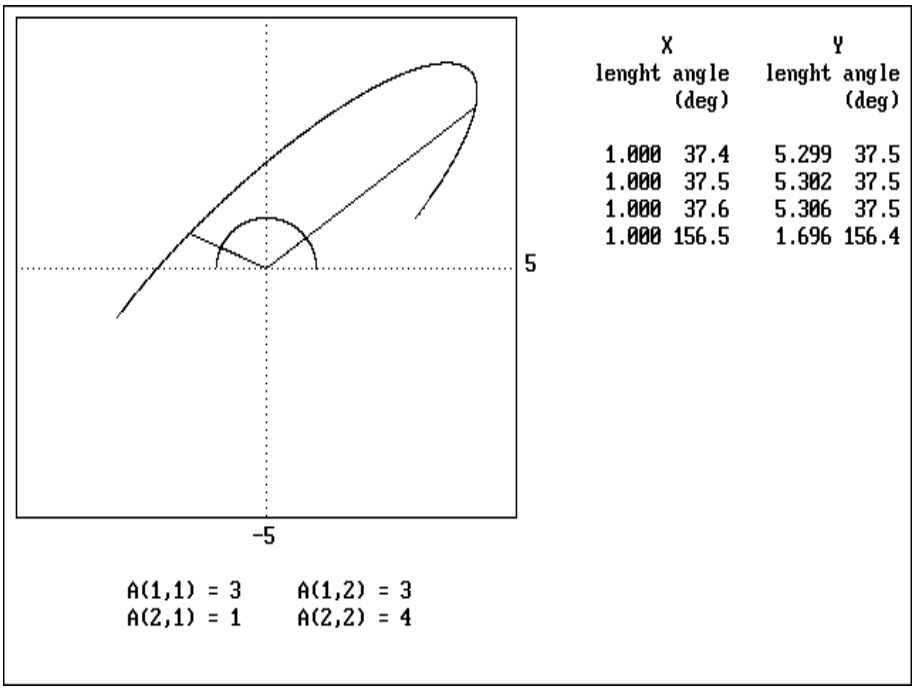


Рис. 10.3. Пошук власних векторів і власних чисел прямим перебором напрямків. Крок по куту $\delta = \pi/1800$ (0.1°), $\varepsilon = \pi/1800$

Варіанти для самостійної роботи

Заздалегідь приготуйте для експериментів 4 матриці 2×2 з дійсними елементами:

- $\mathbf{A}_{(1)}$ – несиметрична з дійсними власними числами;
- $\mathbf{A}_{(2)}$ – несиметрична з комплексними власними числами;
- $\mathbf{A}_{(3)}$ – несиметрична вироджена;
- $\mathbf{A}_{(4)}$ – симетрична (тобто $a_{21} = a_{12}$) невироджена.

Прийміть елемент $a_{11} = \mathrm{V}/2$, де V – номер варіанту, а решту елементів виберіть самостійно з діапазону (–20…+20).

Можна показати, що добуток всіх власних чисел матриці дорівнює її детермінанту. Тому у виродженої матриці $\mathbf{A}_{(3)}$ серед власних чисел обов’язково є нульове.

Контрольні запитання

1. Що являє собою геометричне місце точок, що є кінцями всіх можливих одиничних векторів в 3-вимірному просторі? А в просторі більшого числа вимірювань?
2. Що являє собою геометричне місце точок, яке описує кінець вектора $\mathbf{y} = \mathbf{A}\mathbf{x}$, якщо одиничний вектор $\mathbf{x}$ приймає всіх можливі напрямки в 3-вимірному просторі? А в просторі більшого числа вимірювань?
3. Що таке власні числа і власні вектори матриці?
4. Скільки власних чисел у квадратної матриці розміром 4×4? А власних векторів?
5. Опишіть ваш спосіб дій, якщо б перед вами стояла задача визначення прямим перебором власних векторів і власних чисел матриці 3×3. Зокрема, яким чином ви б встановлювали факт тотожності або близькості напрямків векторів $\mathbf{x}$ та $\mathbf{y}$?
6. Доведіть, що власні числа будь-якої матриці 2×2 з дійсними компонентами або обидва дійсні, або обидва комплексні. Сформулюйте узагальнення цього твердження на випадок матриці будь-якого порядку з дійсними компонентами.
7. Доведіть, що добуток всіх власних чисел матриці дорівнює її детермінанту. Якщо ви утруднюєтеся довести це для матриць довільної розмірності, обмежтеся випадком 2×2.
8. Доведіть, що у симетричної матриці з дійсними компонентами всі власні числа і власні вектори – дійсні. Якщо ви утруднюєтеся довести це для матриць довільної розмірності, обмежтеся випадком 2×2.
9. Доведіть, що у симетричної матриці з дійсними компонентами всі власні вектори взаємно-ортогональні. Якщо ви утруднюєтеся довести це для матриць довільної розмірності, обмежтеся випадком 2×2.
10. Яку особливість мають власні вектори симетричної матриці виду $\begin{pmatrix} a & b \\ b & a \end{pmatrix}$ крім того, що вони є взаємно-ортогональними?
11. Доведіть, що у симетричної матриці, отриманої як результат добутку $\mathbf{A}^{\mathrm{T}}\mathbf{A}$, де $\mathbf{A}$ – довільна матриця з дійсними компонентами, власні числа не тільки дійсні, але і невід'ємні. (Покажчик T означає транспонування.) Якщо ви утруднюєтеся довести це для матриць довільної розмірності, обмежтеся випадком 2×2.

Лабораторна робота № 11.

Знаходження власних векторів і власних чисел симетричних матриць. Метод обертань Якобі

Мета роботи: вивчення алгоритму і налаштування програми для знаходження власних векторів і власних чисел симетричних матриць методом обертань Якобі.

Що зробити: знайти власні числа і власні вектори симетричної матриці **A** шляхом її декомпозиції методом обертань Якобі в добуток виду $\mathbf{A} = \mathbf{QDQ}^T$, де **D** – діагональна матриця власних чисел, а **Q** – ортогональна матриця власних векторів. Впевнитися у правильності результату шляхом перевірки співвідношень $\mathbf{Q}^T\mathbf{Q} = \mathbf{E}$ та $\mathbf{QDQ}^T = \mathbf{A}$. Додатково – впевнитися в інваріантності сліду матриці **A** та її норми Фробеніуса при послідовних ітераціях. Оцінити порядок збіжності методу обертань Якобі.

Стислі теоретичні відомості

A. Подібні матриці

Квадратні матриці **A** та **B** називаються *подібними*, якщо існує невироджена матриця **P** (називається матрицею переходу), що виконується співвідношення:

$$\mathbf{AP} = \mathbf{PB} .$$

Можна показати, що у подібних матриць багато характеристик збігаються, а саме: детермінант, ранг матриці, слід матриці (тобто сума діагональних елементів), характеристичний многочлен, власні числа (хоча, власні вектори можуть бути різними) та ін.

Задачу на власні значення

$$\mathbf{Ax} = \lambda\mathbf{x} ,$$

можна сформулювати в інший спосіб, в термінах подібних матриць. Складемо діагональну матрицю **D**, елементами якої є власні числа $\lambda_1 \ldots \lambda_n$:

$$\mathbf{D} = \begin{pmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ldots & \\ 0 & & & \lambda_n \end{pmatrix}$$

а також матрицю **X**, стовпчиками якої є власні вектори $\mathbf{x}_1 \ldots \mathbf{x}_n$:

$$\mathbf{X} = \begin{array}{c} \left(\begin{array}{cccc} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n1} & x_{n2} & \ldots & x_{nn} \end{array}\right) \\ \begin{array}{cccc} \uparrow & \uparrow & & \uparrow \\ \mathbf{x}_1 & \mathbf{x}_2 & & \mathbf{x}_n \end{array} \end{array}$$

Добуток **XD** являє собою матрицю, в якій кожен із стовпчиків $\mathbf{x}_j$ помножений на відповідне число λ_j. Тому замість умови (1), що має виконуватися для кожної з пар $(\lambda_j, \mathbf{x}_j)$, можна написати єдине співвідношення для всього ансамблю власних векторів і власних чисел:

$$\mathbf{AX} = \mathbf{XD} . \qquad (1)$$

Таким чином, задача на власні значення матриці **A** може бути сформульована як задача відшукання подібної до **A** діагональної матриці **D**. Елементи λ_j на діагоналі матриці **D** є власними числами матриці **A** і визначені з точністю до їх перестановки.

Б. Ортогональні матриці.

Нехай $\mathbf{q}_1, \ldots \mathbf{q}_n$ – ортонормовані вектори в n-вимірному просторі:

$$\mathbf{q}_i^{\mathrm{T}}\mathbf{q}_j = \begin{cases} 1, \text{ при } i = j \\ 0, \text{ при } i \neq j \end{cases} \qquad (2)$$

(Покажчик T означає транспонування і в даному контексті позначає вектор-рядок). Складемо матрицю $\mathbf{Q}$, стовпчиками якої є вектори $\mathbf{q}_1, \ldots \mathbf{q}_n$:

$$\mathbf{Q} = \begin{array}{|cccc|} \hline q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ q_{n1} & q_{n2} & \cdots & q_{nn} \\ \hline \end{array}$$

$$\uparrow \quad \uparrow \qquad \uparrow$$

$$\mathbf{q}_1 \quad \mathbf{q}_2 \qquad \mathbf{q}_n$$

Така матриця називається *ортогональною*. Внаслідок властивості (2)

$$\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{E}\,, \qquad (3)$$

де $\mathbf{E}$ – одинична матриця. Тобто, транспонована ортогональна матриця збігається з оберненою:

$$\mathbf{Q}^{\mathrm{T}} = \mathbf{Q}^{-1}.$$

Легко бачити, що добуток ортогональних матриць є також ортогональною матрицею.

Розглянемо перетворення, яке описує ортогональна матриця. Нехай $\mathbf{y} = \mathbf{Q}\mathbf{x}$. Тоді

$$|\mathbf{y}|^2 = \mathbf{y}^{\mathrm{T}}\mathbf{y} = \mathbf{x}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}}\mathbf{Q}\mathbf{x} = \mathbf{x}^{\mathrm{T}}\mathbf{x} = |\mathbf{x}|^2,$$

тобто ортогональне перетворювання не змінює довжин векторів.

Нехай φ_x – кут між деякими векторами $\mathbf{x}'$ та $\mathbf{x}''$:

$$\cos\varphi_x = \frac{\mathbf{x}'^{\mathrm{T}}\,\mathbf{x}''}{|\mathbf{x}'|\cdot|\mathbf{x}''|}$$

Тоді кут φ_y між $\mathbf{y}' = \mathbf{Q}\mathbf{x}'$ та $\mathbf{y}'' = \mathbf{Q}\mathbf{x}''$ визначиться співвідношенням:

$$\cos\varphi_y = \frac{\mathbf{y}'^{\mathrm{T}}\,\mathbf{y}''}{|\mathbf{y}'|\cdot|\mathbf{y}''|} = \frac{\mathbf{x}'^{\mathrm{T}}\,\mathbf{Q}^{\mathrm{T}}\mathbf{Q}\,\mathbf{x}''}{|\mathbf{x}'|\cdot|\mathbf{x}''|} = \cos\varphi_x\,,$$

тобто $\varphi_y = \pm\varphi_x$, отже ортогональне перетворювання не змінює і кутів і таким чином описує поворот в просторі (можливо, разом із дзеркальним відображенням) без розтягування або стиснення.

Наприклад, в 2-вимірному просторі поворот вектора на кут φ проти годинникової стрілки описується ортогональною матрицею

$$\mathbf{Q} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}.$$

Можна показати, що у симетричної матриці (тобто такої, що $\mathbf{A}^{\mathrm{T}} = \mathbf{A}$) власні числа $\lambda_1, \ldots \lambda_n$ завжди є дійсними, а власні вектори $\mathbf{x}_1, \ldots \mathbf{x}_n$ – взаємно-ортогональними. Таким чином, якщо віднормувати довжину кожного власного вектора $\mathbf{x}_j$ на одиницю

$$\mathbf{q}_j = \frac{\mathbf{x}_j}{\left|\mathbf{x}_j\right|},$$

і скласти матрицю **Q**, стовпчиками якої є вектори $\mathbf{q}_1, \ldots \mathbf{q}_n$, то ця матриця буде ортогональною. Отже, для симетричних матриць співвідношення (1) з урахуванням властивості (3) може бути записане у вигляді декомпозиції

$$\mathbf{A} = \mathbf{QDQ}^{\mathrm{T}},$$

де **Q** – ортогональна матриця, а **D** – діагональна, причому такий розклад на множники завжди може бути виконаний в дійсних числах. Тоді діагональ **D** складатиметься із власних чисел матриці **A**, а стовпчики матриці **Q** містимуть її власні вектори, нормовані на одиничну довжину. Таку процедуру називають також діагоналізацією матриці **A**.

B. Метод обертань Якобі.

Метод Якобі полягає у виконанні ітераційних перетворень, які зводять матрицю **A** до діагонального виду. Будується така послідовність пар матриць ($\mathbf{Q}^{(k)}$; $\mathbf{D}^{(k)}$), щоб вона збігалася до (**Q**; **D**).

При цьому вимагається, щоб на кожній ітерації

(i) матриця $\mathbf{Q}^{(k)}$ була ортогональною;

(ii) матриця $\mathbf{D}^{(k)}$ – симетричною;

(iii) вони всі були пов’язані з початковою матрицею **A** співвідношенням

$$\mathbf{A} = \mathbf{Q}^{(0)}\mathbf{D}^{(0)}\mathbf{Q}^{(0)\mathrm{T}} = \mathbf{Q}^{(1)}\mathbf{D}^{(1)}\mathbf{Q}^{(1)\mathrm{T}} = \ldots = \mathbf{Q}^{(k)}\mathbf{D}^{(k)}\mathbf{Q}^{(k)\mathrm{T}} = \ldots \to \mathbf{QDQ}^{\mathrm{T}};$$

(iv) матриця $\mathbf{D}^{(k+1)}$ була б в певному сенсі більш наближеною до діагональної, ніж $\mathbf{D}^{(k)}$.

Вочевидь, якщо процес буде починатися з $\mathbf{D}^{(0)} = \mathbf{A}$, то, зважаючи на умову (iii), початковим значенням для **Q** слід взяти одиничну матрицю: $\mathbf{Q}^{(0)} = \mathbf{E}$.

В подальшому поточне (k-те) і наступне (k+1-ше) наближення матриць в цій послідовності будемо позначати (**Q** ; **D**) та (**Q'** ; **D'**) без зазначення номеру наближення верхнім індексом.

Оскільки згідно з умовою (i) кожне наступне наближення **Q'** , так само як і попереднє **Q** , є ортогональною матрицею, то воно має утворюватися з **Q** домноженням її на деяку іншу ортогональну матрицю **P** (добуток ортогональних матриць є також ортогональною матрицею):

$$\mathbf{Q'} = \mathbf{QP}. \tag{4}$$

Тоді для виконання умови (iii) $\mathbf{QDQ}^{\mathrm{T}} = \mathbf{Q'D'Q'}^{\mathrm{T}}$ необхідно, щоб наступне наближення **D'** було пов’язане з попереднім **D** співвідношенням $\mathbf{QDQ}^{\mathrm{T}} = (\mathbf{QP})\mathbf{D'}(\mathbf{P}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}})$, тобто $\mathbf{D} = \mathbf{PD'P}^{\mathrm{T}}$, або ж

$$\mathbf{D'} = \mathbf{P}^{\mathrm{T}}\mathbf{DP}. \tag{5}$$

Легко бачити, що умова (ii) симетричності матриці **D'** при цьому виконується.

Отже, при розрахунку наступного наближення (**Q'** ; **D'**) за формулами (4), (5) за допомогою будь-якої ортогональної матриці **P** умови (i), (ii) та (iii) будуть задовольнятися тотожно, а задача, таким чином, зводиться до такого вибору матриці **P** на кожному кроці, щоб задовольняти умові (iv).

В методі обертань Якобі за **P** беруть ортогональну матрицю найпростішого виду, а саме – матрицю повороту Ґівенса, яка описує поворот в площині (*lm*) на кут φ і відрізняється від одиничної матриці лише 4 елементами:

$$\mathbf{P} = \begin{array}{c} \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \\ \begin{matrix} & & l\text{-й стовпчик} & & m\text{-й стовпчик} & & \end{matrix} \end{array} \begin{matrix} \\ \\ l\text{-й рядок} \\ \\ m\text{-й рядок} \\ \\ \\ \end{matrix}$$

В ній $c = \cos\varphi$, $s = \sin\varphi$.

Легко бачити, що в матрицях **Q'** і **Q** відрізняються лише *l*-й та *m*-й стовпчики:

$$\left.\begin{aligned} q'_{il} &= q_{il}c + q_{im}s \\ q'_{im} &= -q_{il}s + q_{im}c \end{aligned}\right\}, \quad i = 1,\ \ldots,\ n, \tag{6}$$

(де *n* – порядок матриці), а решта елементів залишаються без змін. Аналогічно, в матрицях **D'** та **D** відрізняються лише *l*-й та *m*-й рядки та стовпчики. Їх можна обчислити за формулами:

$$\begin{aligned} d'_{ll} &= d_{ll}c^2 + 2d_{lm}cs + d_{mm}s^2 \\ d'_{mm} &= d_{ll}s^2 - 2d_{lm}cs + d_{mm}c^2 \\ d'_{lm} &= d'_{ml} = (-d_{ll} + d_{mm})\,cs + d_{lm}(c^2 - s^2) \\ \left.\begin{aligned} d'_{il} &= d'_{li} = d_{il}c + d_{im}s \\ d'_{im} &= d'_{mi} = -d_{il}s + d_{im}c \end{aligned}\right\}, & \quad i = 1,\ \ldots,\ n, \quad i \neq l, m. \end{aligned} \tag{7}$$

Площину (lm) та кут повороту φ визначають так. В матриці **D** вибирають опорний елемент – максимальний за модулем елемент d_{lm} поза діагоналлю, і вимагають, щоб після повороту елемент на його місці став рівним нулю. Таким чином, максимальний позадіагональний елемент в **D'** буде меншим, ніж в **D**, і в цьому сенсі **D'** буде «ближче до діагональної», ніж **D**, як того вимагає умова (iv).

Тоді, вимагаючи, щоб в (7) $d'_{lm} = 0$, на тригонометричні функції кута φ накладається умова:

$$\frac{c^2 - s^2}{cs} = \frac{d_{ll} - d_{mm}}{d_{lm}}.$$

Позначаючи

$$t = \frac{d_{ll} - d_{mm}}{d_{lm}} = \frac{c^2 - s^2}{2cs} = \operatorname{ctg} 2\varphi\ , \tag{8}$$

обраховуємо значення тригонометричних функцій кута повороту:

$$u = \frac{t}{\sqrt{1 + t^2}} = \cos 2\varphi \tag{9}$$

$$c = \sqrt{\frac{1+u}{2}} = \cos\varphi\,; \qquad s = \sqrt{\frac{1-u}{2}} = \sin\varphi\,. \tag{10}$$

(Така схема розрахунків дозволяє визначити c і s уникаючи безпосереднього обчислення тригонометричних функцій.)

На жаль, утворення нового нульового елементу часто веде до появи ненульового елементу там, де раніше був нуль. Тому метод Якобі виходить ітераційним. Він постійно повторює повороти Гівенса поки матриця **D** не стане майже діагональною, тобто максимальний недіагональний елемент не стане за модулем менше наперед заданого малого числа ε.

Завдання

1. Складіть процедуру діагоналізації симетричної матриці методом Якобі. Основу програмного коду запозичте з наведенного фрагменту.
2. Задайте симетричну матрицю **A** згідно з вашим варіантом. Оскільки матриця досить громіздка, не рекомендується вводити її компоненти з клавіатури, це занадто подовжить етап налагодження програми. Введіть її з файлу або ж згенеруйте безпосередньо в програмі, наприклад, за допомогою генератора випадкових чисел.
3. Виведіть початкову матрицю **A**, а також, по завершенню процедури діагоналізації, – ортогональну матрицю її власних векторів **Q** і діагональну матрицю власних чисел **D** (для запуску процедури необхідно скопіювати **A** в **D**). Потурбуйтеся, щоб результати, що виводяться, мали вигляд охайних таблиць.

```
SUB Jacobi(n,D(2),Q(2),eps)
'
' ------------------------------------------------------------
' Діагоналізація симетричної матриці методом Якобі.
'
' Вхідні параметри:
'    n      – порядок матриць;
'    D[n,n] – симетрична матриця для визначення власних
'             векторів і чисел, руйнується при обчисленнях;
'    eps    – точність розрахунків;
' Вихідні параметри:
'    D[n,n] – діагональна матриця, містить власні числа
'             початкової матриці D[n,n];
'    Q[n,n] – ортогональна матриця, містить нормовані
'             власні вектори початкової матриці A[n,n].
' ------------------------------------------------------------
'
FOR i=1 TO n                                  ' ініціація Q оди-
   FOR j=1 TO n                               ' ничною матрицею
      IF i=j THEN Q[i,j]=1 ELSE Q[i,j]=0
   NEXT j
NEXT i
```

```
DO                                              ' початок ітерацій

   Dlm=D[2,1] : l=2 : m=1                       ' пошук найбільш.
   FOR i=2 TO n                                 ' недіагоального
      FOR j=1 TO i-1                            ' елемента
         IF abs(D[i,j]) > abs(Dlm) THEN
             Dlm=D[i,j] : l=i : m=j
         END IF
      NEXT j
   NEXT i
                                                ' вихід, якщо до-
   IF abs(Dlm)<eps THEN EXIT LOOP               ' сягнута точність

   t=(D[l,l]-D[m,m])/(2*D[l,m])                 ' cos та sin
   u=t/SQR(1+t*t)                               ' кута повороту
   c=SQR((1+u)/2)
   s=SQR((1-u)/2)

   FOR i=1 TO n                                 ' перерахунок Q
      Qil= Q[i,l]*c+Q[i,m]*s
      Qim=-Q[i,l]*s+Q[i,m]*c
      Q[i,l]=Qil
      Q[i,m]=Qim
   NEXT i

   Dll=D[l,l]*c*c+2*D[l,m]*c*s+D[m,m]*s*s  ' перерахунок D:
   Dmm=D[l,l]*s*s-2*D[l,m]*c*s+D[m,m]*c*c  ' 4 елементи
   D[l,l]=Dll
   D[m,m]=Dmm
   D[l,m]=0 : D[m,l]=0

   FOR i=1 TO n                                 ' решта елементів
      IF i<>l AND i<>m THEN
         Dil= D[i,l]*c+D[i,m]*s
         Dim=-D[i,l]*s+D[i,m]*c
         D[i,l]=Dil : D[l,i]=D[i,l]
         D[i,m]=Dim : D[m,i]=D[i,m]
      END IF
   NEXT i

LOOP                                            ' кінець циклу

END SUB
```

4. Перевірте правильність роботи вашої програми. Для цього перевірте співвідношення $\mathbf{Q}^T\mathbf{Q} = \mathbf{E}$ та $\mathbf{QDQ}^T = \mathbf{A}$.

Додаткове завдання

5. *Слід матриці* Sp(**A**) (від нім. Spur – слід) – це сума усіх її діагональних елеменітів:

$$\mathrm{Sp}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii},$$

а *норма Фробеніуса* $\|\mathbf{A}\|_F$ – корінь квадратний із суми квадратів всіх елементів матриці:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}.$$

Можна показати, що при перетвореннях типу (5) слід матриці та її норма Фробеніуса повинні залишатися незмінними. Впевніться в цьому, обчислюючи $\mathrm{Sp}(\mathbf{D})$ та $\|\mathbf{D}\|_F$ на кожній ітерації.

6. Кожного разу після визначення максимального за модулем (опорного) елемента d_{lm} виводіть його значення (вивід зручно дублювати у файл). Аналізуючи послідовність $|d_{lm}|$ з'ясуйте, чи дорівнює порядок збіжності методу обертань Якобі одиниці, чи перевищує 1. Для отриамння послідовності максимальної довжини задавайте точність ε на рівні машинного епсілон. Можливо, для ще більшої подовженості послідовності, знадобиться проводити всі обчислення з подвійною точністю.

Варіанти для самостійної роботи

Використовуйте симетричні матриці заданої розмірності. Деякі з елементів матриці вказані в завданні, решту елементів виберіть самостійно з діапазону (–20…+20).

Варіанти 1, 6, 11, 16, 21: матриця 4 x 4, $a_{11} = V/2$, $a_{13} = a_{31} = 8.0$

Варіанти 2, 7, 12, 17, 22: матриця 5 x 5, $a_{22} = V/4$, $a_{24} = a_{42} = -15.0$

Варіанти 3, 8, 13, 18, 23: матриця 6 x 6, $a_{33} = V/2$, $a_{35} = a_{53} = 3.0$

Варіанти 4, 9, 14, 19, 24: матриця 4 x 4, $a_{44} = V/5$, $a_{23} = a_{32} = -5.0$

Варіанти 5, 10, 15, 20: матриця 5 x 5, $a_{55} = V$, $a_{34} = a_{43} = 12.0$

де V – номер варіанту.

Контрольні запитання

1. Нехай матриці загального виду (не обов'язково симетричні) **A** та **B** подібні: $\mathbf{AP} = \mathbf{PB}$, де **P** – матриця переходу. Нехай **X** є матрицею власних векторів (не обов'язково нормованих на одиничну довжину) матриці **A**, а **D** – діагональною матрицею її власних чисел: $\mathbf{AX} = \mathbf{XD}$. Доведіть, що власні вектори матриці **B** дорівнюють $\mathbf{P}^{-1}\mathbf{X}$, а власні числа такі ж, як у матриці **A**.
2. Доведіть, що добуток ортогональних матриць є також ортогональною матрицею.
3. Доведіть, що коли $\mathbf{A} = \mathbf{QDQ}^{\mathrm{T}}$ та $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{E}$, то $\mathbf{D} = \mathbf{Q}^{\mathrm{T}}\mathbf{AQ}$.
4. Покажіть, що при перетворенні (5), якщо **D** – симетрична матриця, то й **D'** також залишається симетричною.
5. Виведіть формули (6) для розрахунку елементів матриці **Q** після повороту Ґівенса.
6. Виведіть формули (7) для розрахунку елементів матриці **D** після повороту Ґівенса.
7. Аналізуючи формули (7) покажіть, що сума квадратів діагональних елементів в кожному наступному наближенні **D'** більша, ніж в попередньому:

   $$d_{ll}'^2 + d_{mm}'^2 = d_{ll}^2 + 2d_{lm}^2 + d_{mm}^2 \ .$$

   Поясніть, чому цей факт є гарантією того, що ітераційний процес збігається. Спирайтеся на незмінність норми Фробеніуса матриці **D**.
8. Іноді формули (8), (9) записують у формі:

   $$\tilde{t} = \frac{2d_{lm}}{d_{ll} - d_{mm}} = \frac{2cs}{c^2 - s^2} = \operatorname{tg} 2\varphi \ ,$$

   $$u = \frac{1}{\sqrt{1 + \tilde{t}^2}} = \cos 2\varphi \ .$$

   Поясніть, чому так робити не слід.
9. Запропонуйте метод розрахунку матриці $\mathbf{A}^{-1}$, оберненої до симетричної мариці **A**, із застосуванням декомпозиції $\mathbf{A} = \mathbf{QDQ}^{\mathrm{T}}$.
10. Доведіть, що при перетворенні (5) слід матриці залишається незмінним. Якщо ви утруднюєтеся довести це для матриць довільної розмірності, обмежтеся випадком 2×2.

11. Доведіть, що при перетворенні (5) норма Фробеніуса матриці залишається незмінною. Якщо ви утруднюєтеся довести це для матриць довільної розмірності, обмежтеся випадком 2×2.

12. Оскільки матриця **D** симетрична і залишається такою на всіх етапах обчислень, немає потреби зберігати в пам'яті комп'ютера її всю. Якщо обмежитися лише елементами головної діагоналі і одного з симетричних трикутників, то економія пам'яті становитиме майже половину, особливо для матриць великої розмірності. Запропонуйте спосіб економного зберігання в комп'ютерній пам'яті симетричних матриць і спосіб адресації її елементів. Які зміни зазнає програмний код для методу обертань Якобі?

Лабораторна робота № 12.

Сингулярний розклад матриці

Мета роботи: Наочна геометрична інтерпретація сингулярного розкладу (SVD) матриць 2×2 з дійсними елементами.

Що зробити: в двовимірному просторі (площині) здійснити прямий перебор (з певним малим кроком) одиничних векторів **x** всіх можливих напрямків, побудувати еліпс, який описує вектор добутку $\mathbf{y} = \mathbf{A}\mathbf{x}$, де **A** – матриця 2×2, визначити довжини та напрямки його півосей, а також ті вектори **x**, які породжуюся **y**, що відповідають півосям. Порівняти ці величини з сингулярними числами та сингулярними векторами матриці **A**, розрахованими аналітично.

Стислі теоретичні відомості

Можна показати, що будь-яка дійсна матриця **A** може бути представлена у вигляді добутку

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}, \qquad (1)$$

де **U** та **V** – ортогональні матриці, а **Σ** – діагональна, причому лише з невід'ємними числами σ_j на діагоналі. Ці елементи називаються *сингулярними числами* матриці **A** і визначені з точністю до їх перестановки. Зазвичай вимагають, щоб вони розташовувалися в матриці **Σ** в незростаючому порядку: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0$.

Тоді **Σ** (але не **U** та **V**) однозначно визначається по матриці **A**. Індекс r елемента σ_r є рангом матриці **A** , і якщо **A** невироджена, то всі її сингулярні числа строго додатні.

Така декомпозиція називається *сингулярним розкладом матриці на множники* (singular value decomposition, SVD)

Одним із способів відшукання сингулярного розкладу є розв'язок задачі на власні значення симетричної матриці $\mathbf{A}^{\mathrm{T}}\mathbf{A}$:

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = (\mathbf{V}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}})(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}) = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^{\mathrm{T}}. \qquad (2)$$

Таким чином, **V** є матрицею власних векторів матриці $\mathbf{A}^{\mathrm{T}}\mathbf{A}$, а σ_j^2 – власні числа цієї матриці. Можна показати, що всі вони не тільки дійсні, а

й невід'ємні, тому можна взяти всі $\sigma_j > 0$. Матриця **U** визначається із (1) з урахуванням ортогональності **U** та **V** :

$$\mathbf{U} = \mathbf{A}\mathbf{V}\mathbf{\Sigma}^{-1} . \tag{3}$$

Співвідношення (1) можна записати в одній з двох інших еквівалентних форм:

$$\begin{cases} \mathbf{A}\mathbf{V} = \mathbf{U}\mathbf{\Sigma} \\ \mathbf{A}^{\mathrm{T}}\mathbf{U} = \mathbf{V}\mathbf{\Sigma} \end{cases} \tag{4}$$

Якщо стовпчики матриць **U** та **V** вважати наборами одиничних векторів $\mathbf{u}_j$ та $\mathbf{v}_j$, то (4) можна записати у вигляді:

$$\begin{cases} \mathbf{A}\mathbf{v}_j = \sigma_j \mathbf{u}_j \\ \mathbf{A}^{\mathrm{T}}\mathbf{u}_j = \sigma_j \mathbf{v}_j \end{cases}$$

Вектори **u** та **v** називаються відповідно *сингулярним зліва вектором* та *сингулярним справа вектором* для σ.

Зауважимо, що для транспонованої матриці праві та ліві сингулярні вектори міняються ролями, а сингулярні числа залишаються незмінними.

Вочевидь, для симетричних матриць сингулярні зліва і справа вектори співпадають між собою і є власними векторами матриці, а сингулярні числа – її власними числами.

Зважаючи на те, що ортогональна матриця описує поворот в просторі (можливо, разом із дзеркальним відображенням) без розтягування або стиснення, а діагональна – розтягування (стиснення) по осях без повороту, твердження (1) фактично означає наступне. Лінійне перетворення, що описує будь-яка матриця **A**, може розглядатися як послідовність трьох операцій: повороту $\mathbf{V}^{\mathrm{T}}$, розтягування по осях $\mathbf{\Sigma}$, та другого повороту **U**.

Рис. 12.1 ілюструє етапи такої процедури на прикладі лінійного перетворення одиничного кола (вгорі ліворуч) в еліпс (вгорі праворуч).

Спершу ми бачимо одиничне коло з двома векторами стандартного базису (позначені пунктирами зі стрілочками на кінцях) та сингулярними векторами **v** (позначені суцільними лініями з кружечками на кінцях).

Потім ми бачимо дію $\mathbf{V}^{\mathrm{T}} = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix}$, що повертає фігуру на кут φ за годинниковою стрілкою таким чином, що початкові вектори **v** набувають напрямки базисних.

Наступний етап полягає в дії $\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$, що розтягує одиничне коло вздовж координатних осей в еліпс з півосями σ_1 і σ_2, які є сингулярними числами **A**.

Кінцевий поворот $\mathbf{U} = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix}$ повертає фігуру на кут ψ проти годинникової стрілки, і початкові вектори **v** набувають напрямки векторів **u**.

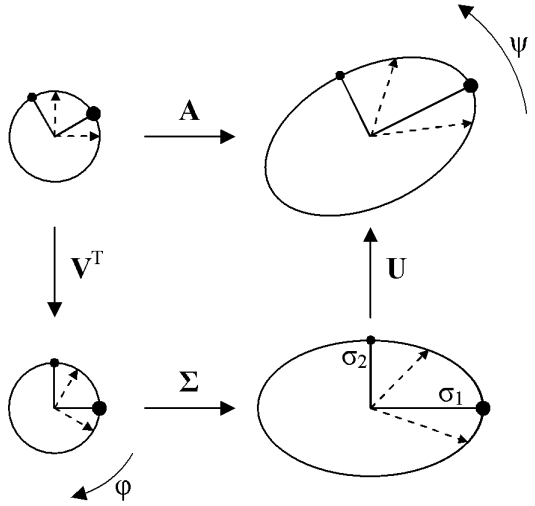


Рис. 12.1. Етапи лінійного перетворення
при сингулярному розкладі матриці $\mathbf{A} = \mathbf{U\Sigma V}^T$

Як ви бачили, вивчаючи декомпозицію симетричних матриць в лабораторінй роботі № 11, для них перший $\mathbf{Q}^T$ і другий **Q** повороти є взаємно-оберненими.

Завдання

1. Прийміть за матрицю **A** несиметричні матриці 2×2 з дійсними елементами $\mathbf{A}_{(1)}$ та $\mathbf{A}_{(2)}$, з якими ви проводили дослідження при виконанні лабораторної роботи № 10. Розв'яжіть аналітично задачу на власні значення матриці $\mathbf{A}^T\mathbf{A}$ і знайдіть сингулярний розклад матриці **A** за допомогою співвідношень (2), (3). Сингулярні зліва та справа вектори $\mathbf{u}_1, \mathbf{u}_2$ та $\mathbf{v}_1, \mathbf{v}_2$, що є відповідними стовпчиками ортогональних матриць **U** та **V**, переведіть в полярні координати. Впевніться, що в кожній парі $(\mathbf{u}_1, \mathbf{u}_2)$ та $(\mathbf{v}_1, \mathbf{v}_2)$ полярні кути відрізняються на $\pi/2$ (90º).

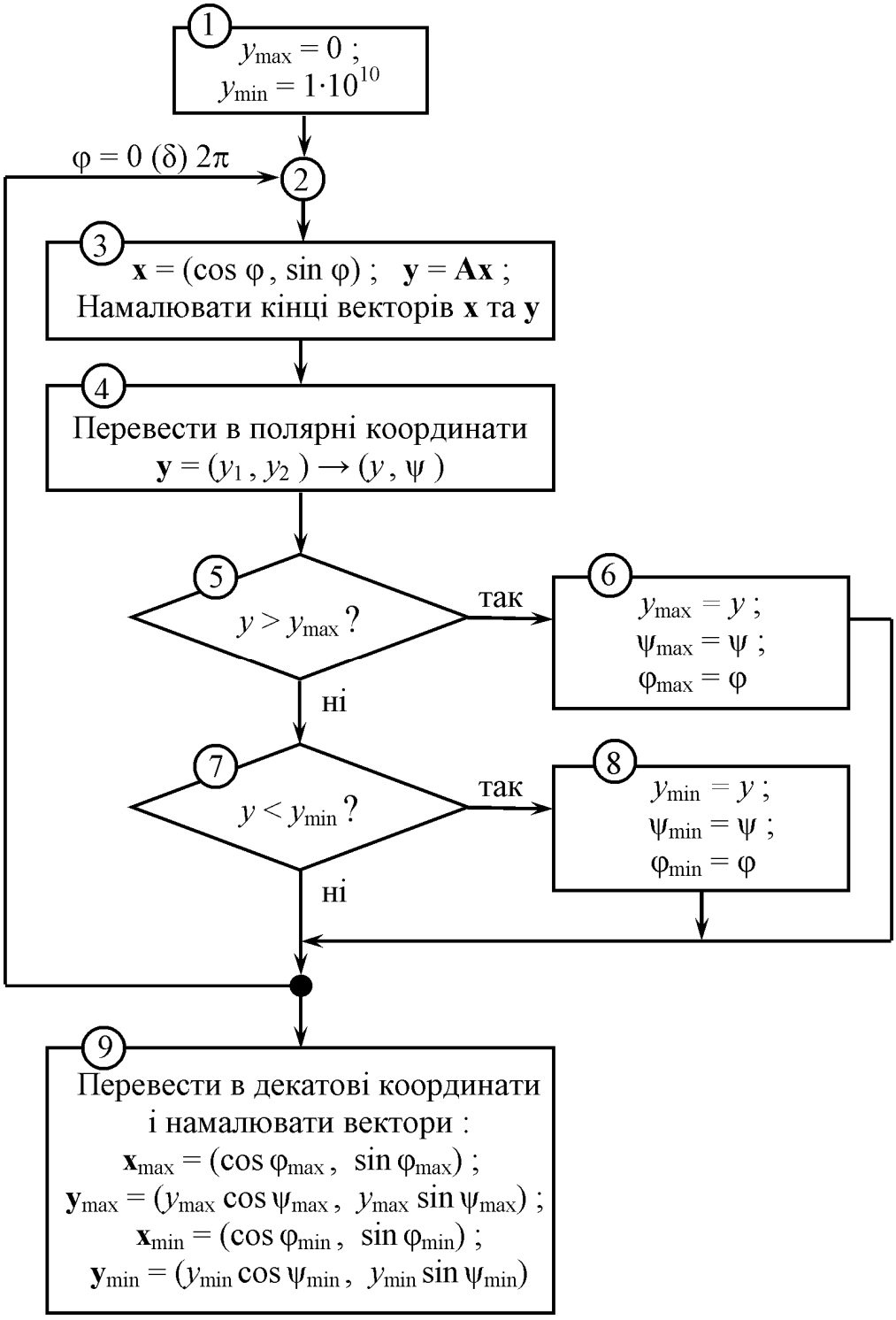


Рис. 12.2. Схема алгоритму визначення півосей еліпса,
що описується кінцем вектора **y** = **Ax**
при переборі одиничних векторів **x** всіх можливих напрямків
(за винятком процедур вводу-виводу)

2. Скористайтеся програмою, яку ви склали при виконанні завдань 1, 2 лабораторної роботи № 10. Доповніть її фрагментом, що відслідковує довжини і напрямки півосей еліпса, що його описує кінець вектора **y**. Для цього зафіксуйте, при яких напрямках вектора **x** (тобто значеннях полярного кута φ, назвемо їх φ_{max} і φ_{min}) вектор **y** набуває максимальної і мінімальної довжини y_{max} і y_{min}, і під якими кутами до горизонтальної осі ψ_{max} і ψ_{min} він при цьому направлений. (Не забудьте перед циклом по φ ініціювати значення y_{max} і y_{min} напевно заниженим і завищеним значеннями).
3. Відобразіть на екрані вектори **x** та **y**, що відповідають цим положенням. Надрукуйте значення довжин і полярних кутів цих векторів. Впевніться, що довжини y_{max} і y_{min} та напрямки ψ_{max} і ψ_{min} дійсно відповідають півосям еліпса.

   Схему алгоритму подано на рис. 12.2, а приклад результату роботи програми – на рис. 12.3.

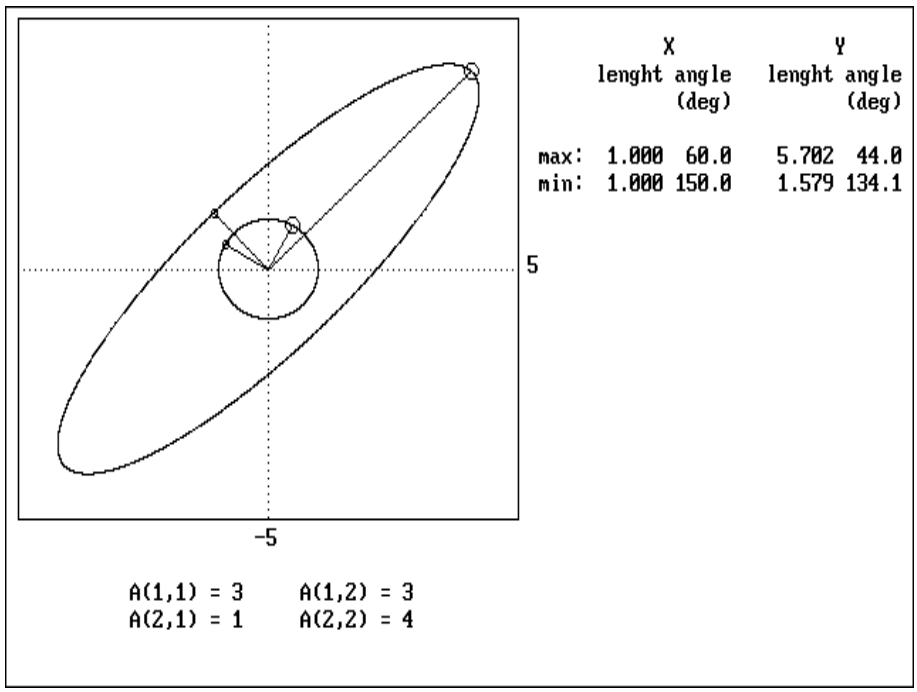


Рис. 12.3. Геометричні місця точок, що описуються при обертанні одиничного вектора **x** та відповідного йому вектора **y** = **Ax**.
Крок по куту δ = π/1800 (0.1º)

4. Виконайте вашу програму з матрицями $\mathbf{A}_{(1)}$ та $\mathbf{A}_{(2)}$ і порівняйте результати, отримані за допомогою програми (тобто φ_{max}, φ_{min}, а також

$y_{\max}$, $\psi_{\max}$, $y_{\min}$, $\psi_{\min}$), з аналітичними розрахунками сингулярних чисел і сингулярних векторів цих матриць. Поясніть результати.

5. Виконайте вашу програму з транспонованими матрицями $\mathbf{A}_{(1)}^{\mathrm{T}}$ та $\mathbf{A}_{(2)}^{\mathrm{T}}$. Чи можете ви сформулювати певні припущення щодо закономірностей, що спостерігаються? Спробуйте їх довести аналітично.

Контрольні запитання

1. Що таке сингулярний розклад матриці?
2. Нехай (σ_j, $\mathbf{u}_j$, $\mathbf{v}_j$) – набір сингулярних чисел і відповідних сингулярних зліва та справа векторів для матриці **A**. Доведіть, що для транспонованої матриці $\mathbf{A}^{\mathrm{T}}$ праві та ліві сингулярні вектори міняються ролями, а сингулярні числа залишаються незмінними.
3. Розгляньте, до яких результатів приводить аналогічний (2) розв'язок задачі на власні значення симетричної матриці $\mathbf{A}\mathbf{A}^{\mathrm{T}}$.
4. Розгляньте *n*-вимірний еліпсоїд, який описує кінець вектора $\mathbf{y} = \mathbf{A}\mathbf{x}$, коли одиничний вектор **x** приймає всіх можливі напрямки в *n*-вимірному просторі. Доведіть, що довжини його півосей дорівнюють сингулярним числам **A**, а напрямки – напрямкам сингулярним зліва векторів **u**. Якщо ви утруднюєтеся довести це для матриць довільної розмірності, обмежтеся випадком 2×2.
5. Опишіть ваш спосіб дій, якщо б перед вами стояла задача визначення прямим перебором сингулярних векторів і сингулярних чисел матриці 3×3. Зокрема, яким чином ви б відслідковували положення *середньої* за величиною півосі відповідного еліпсоїда?
6. Яким чином метод обертань Якобі для пошуку власних чисел і власних векторів симетричної матриці, який ви досліджували при виконанні лабораторної роботи № 11, може бути застосований для пошуку сингулярного розкладу?
7. Запропонуйте метод розрахунку матриці $\mathbf{A}^{-1}$, оберненої до довільної (не обов'язково симетричної) матриці **A**, із застосуванням сингулярного розкладу.

Лабораторна робота № 13.

Оптимізація функцій однієї змінної методом золотого перетину

Мета роботи: вивчення алгоритму і налаштування програми для пошуку мінімуму функції одного аргументу (одновимірної оптимізації) методом золотого перетину.

Що зробити: знайти екстремуми функції $f(x)$ методом золотого перетину. Впевнитись, що їх значення узгоджуються з результатами аналітичного дослідження функції $f(x)$. Визначити порядок збіжності методу золотого перетину. Оцінити максимальну кількість десяткових знаків, які можна визначити цим методом в положенні точки екстремуму і в значенні функції в цій точці.

Стислі теоретичні відомості

Функція $f(x)$ називається *унімодальною* на інтервалі $x \in [a, b]$, якщо існує єдине значення x^*, таке, що $f(x^*)$ – мінімум $f(x)$ на цьому відрізку: $f(x)$ строго спадає для $x < x^*$ і строго зростає для $x > x^*$. Зауважимо, що унімодальна функція не зобов'язана бути гладкою чи навіть неперервною.

Нехай стоїть задача пошуку мінімуму унімодальною функції. Для звуження початкового інтервалу невизначеності $[a, b]$ його слід розділити двома проміжними точками (назвемо їх c і d) на три ділянки (див. рис. 13.1).

Обчисливши значення функції в точках c і d, можна зробити висновок відносно відсутності мінімуму в правій чи лівій ділянці початкового інтервалу. Так, з того факту, що $f(c) > f(d)$ можна стверджувати, що мінімум знаходиться між c і b, а не між a та c, і тим самим звузити інтервал невизначеності для точки мінімуму. Процес продовжується ітеративно, доки не буде виконана певна умова збіжності: наприклад, інтервал стане меншим за наперед задане мале число ε.

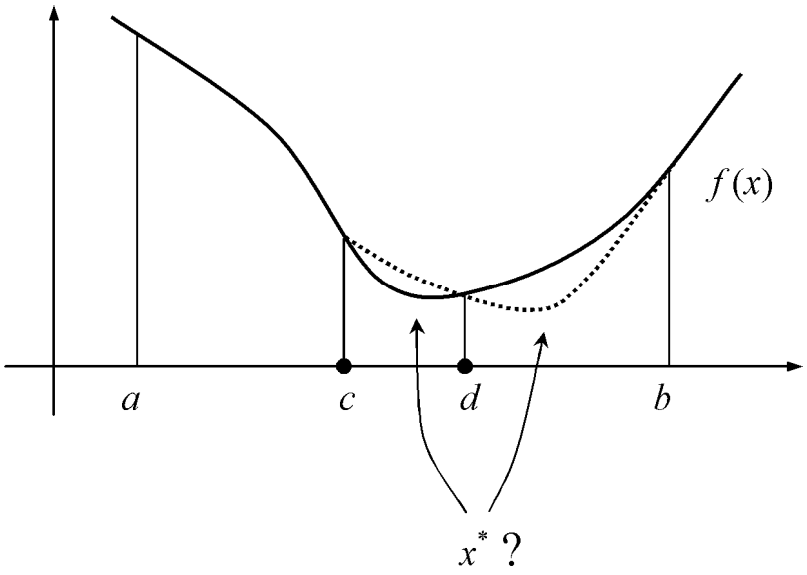


Рис. 13.1. Локалізація мінімуму унімодальної функції

Таким чином, вибираючи на кожній ітерації за тим чи іншим правилом дві точки всередині інтервалу невизначеності, ми отримуємо сімейство алгоритмів одномірної оптимізації, подібне за ідеологією методам подрібнення інтервалу для розв'язання рівнянь з одним невідомим (бісекції, хорд тощо).

З точки зору економії кількості обчислень функції $f(x)$, доцільно вибирати ці дві внутрішні точки таким чином, щоб після звуження інтервалу невизначеності одна з них стала межею нового інтервалу невизначеності, а друга була б використана як одна з проміжних точок наступної ітерації. Тоді кожна ітерація (крім першої) буде вимагати обчислення функції $f(x)$ лише при одному, а не при двох значеннях аргументу.

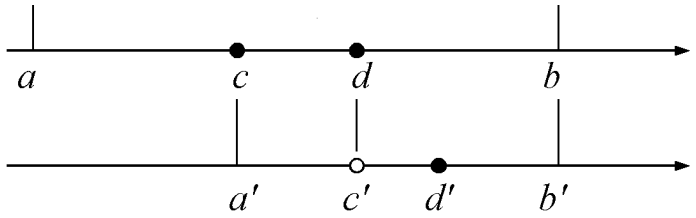


Рис. 13.2. Метод золотого перетину. Функція $f(x)$ обчислюється лише в одній з двох проміжних точок зменшеного інтервалу $[a', b']$

Така стратегія вимагає розбиття інтервалу $[a, b]$ точкою c таким чином, щоб відношення довжини всього інтервалу до більшої частини дорівнювало відношенню більшої частини до меншої:

$$\frac{b-a}{b-c} = \frac{b-c}{b-d}.$$

Точка d вибирається симетрично повністю аналогічно.

Легко перевірити, що

$$c = a + (1 - r)(b - a);$$
$$d = a + r(b - a) = b - (1 - r)(b - a),$$

де $r = (\sqrt{5} - 1) / 2 \approx 0.618\ldots$ – корінь рівняння $r^2 + r - 1 = 0$.

Число $1/r = (\sqrt{5}+1) / 2 \approx 1.618\ldots$ називають *золотим перетином*, звідки і походить назва алгоритму. Золотий перетин є фундаментальним числом, що проявляється в багатьох областях (див., наприклад, контрольне запитання 7 лабораторної роботи № 6).

Завдання

1. Уясніть призначення окремих блоків схеми алгоритму для пошуку мінімуму (оптимізації) функції $f(x)$ методом золотого перетину. Складіть програму, що реалізує цей алгоритм. Фрагмент програми, що власне відшукує мінімум, оформте у вигляді окремої процедури на зразок такої, яку ви складали в лабораторній роботі № 4 для розв'язку рівняння методом бісекції.
2. Візьміть ту ж саму функцію $f(x)$, яку ви досліджували при виконанні лабораторної роботи № 3. За допомогою вашої програми знайдіть її найменший за модулем локальний екстремум. (Якщо таким екстремумом є максимум, знайдіть мінімум функції $-f(x)$.) Початковий інтервал пошуку мінімуму, на якому функція буде унімодальною, виберіть самостійно.
3. З метою налагодження програми після блоків 5, 6 введіть в програму проміжний друк значень a, c, d, b, $|b - a|$, а також f_a, f_c, f_d та f_b на кожній ітерації. Бажано також передбачити лічильник ітерацій і виводити їх номер на початку кожного рядка друку. Потурбуйтеся, щоб результати, що виводяться, мали вигляд охайної таблиці.

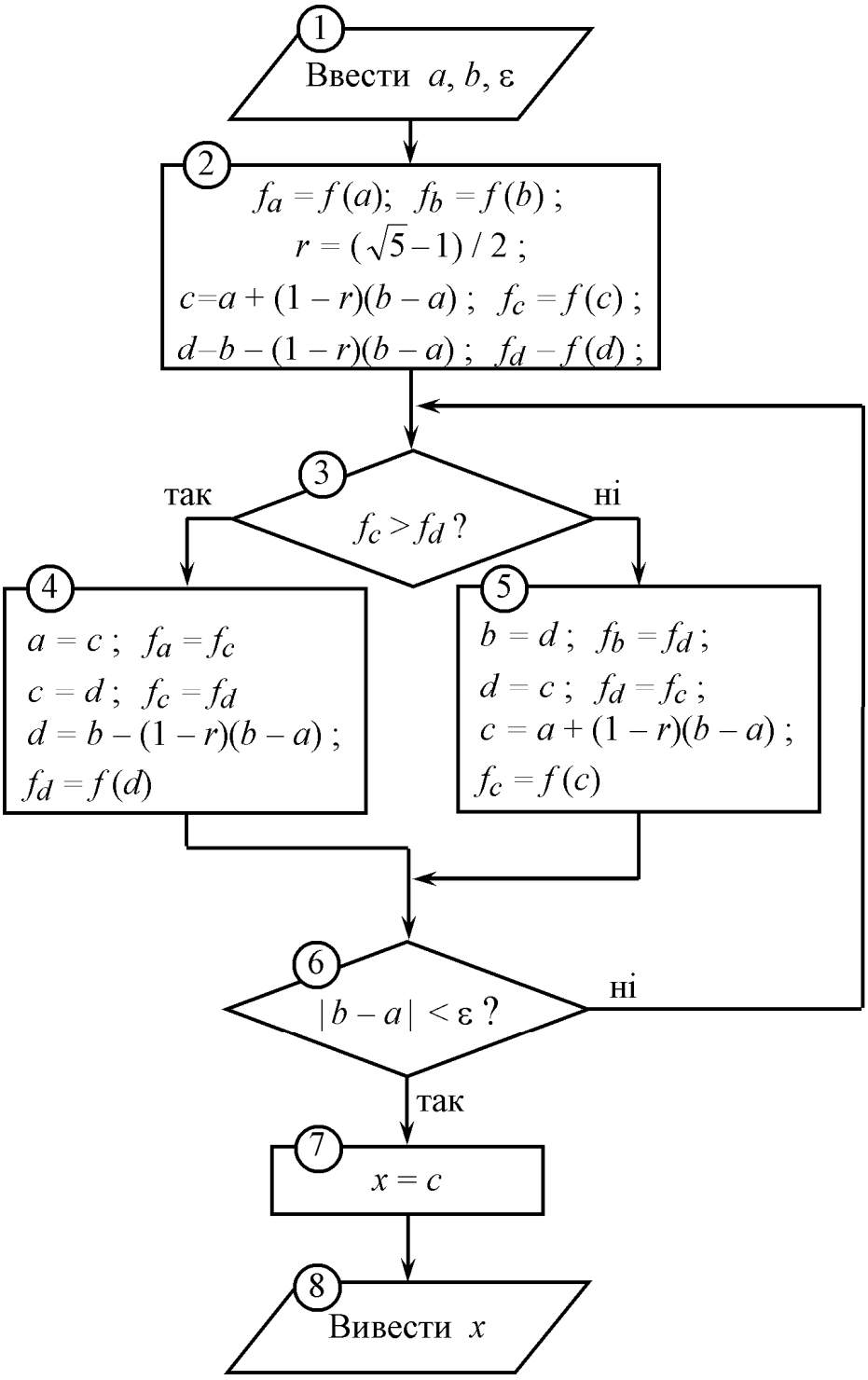


Рис. 13.3. Схема алгоритму методу золотого перетину

4. Дослідіть, як похибки поточного наближення до екстремуму $e^{(i)} = |b-a|$ залежать від номера ітерації i. Побудуйте графік

залежності $\lg e^{(i)}$ від i. На основі цих даних впевніться, що порядок збіжності методу золотого перетину дорівнює 1.

5. Зауважте, що при виникненні ситуації, коли

$$f(a) = f(c) = f(d) \quad \text{або} \quad f(c) = f(d) = f(b),$$

функція перестає бути унімодальною, і подальше звуження інтервалу $[a, b]$ втрачає сенс.

Додайте перевірку й цієї умови в блок 3, що регламентує вихід із циклу. Логічний вираз, що перевіряється, буде виглядати як

```
... abs(b-a)>eps OR (fa=fc AND fc=fd) OR (fc=fd AND fd=fb)
```

Задавайте $\varepsilon = 10^{-2}, 10^{-3}, \ldots$ Зменшуючи ε, оцініть граничну точність, з якою може бути знайдено мінімум функції методом золотого перетину. Порівняйте цю величину з величиною машинного епсілон. Впевніться, що найменше ε, при якому програма не веде себе аномально, приблизно дорівнює кореню квадратному з машинного епсілон, тобто кількість десяткових знаків, які можна визначити в положенні мінімуму, становить лише половину від кількості знаків машинної арифметики. Кажуть, що точка мінімуму може бути знайдена з *половинною машинною точністю*.

Контрольні запитання

1. Яку функцію називають унімодальною? Чим пояснюється вимога унімодальності функції на інтервалі, до якого застосовується метод золотого перетину?
2. Чим обумовлено таке розташування точок c і d, що вони ділять інтервал $[a, b]$ саме в пропорції $r = (\sqrt{5} - 1) / 2 \approx 0.618\ldots$?
3. Яким чином можна з'ясувати порядок збіжності методу, аналізуючи залежність довжини поточного інтервалу невизначеності (тобто похибки поточного наближення) від номера ітерації?
4. Поясніть, чому точка мінімуму може бути знайдена лише приблизно з половинною машинною точністю, на відміну від кореня рівняння, який зазвичай може бути обчислений з повною машинною точністю. Ця особливість є ознакою лише алгоритму золотого перетину, чи притаманна саме задачі оптимізації?
5. Чому в блоці 7 схеми алгоритму передбачено присвоєння $x = c$? Чи можна його замінити на $x = d$? А на $x = (c + d)/2$?

Лабораторна робота № 14.

Оптимізація функцій кількох змінних методом Хука-Дживса

Мета роботи: вивчення алгоритму і налаштування програми для пошуку безумовного мінімуму функції двох аргументів методом Хука-Дживса.

Що зробити: застосувати метод Хука-Дживса до двовимірної функції Розенброка – традиційної тестової функцій для випробувань алгоритмів багатомірної оптимізації. Побудувати на координатній площині ізолінії функції Розенброка і проілюструвати хід обчислень, відмічаючи точки послідовних наближень до екстремуму.

Стислі теоретичні відомості

Нехай потрібно знайти локальний мінімум скалярної цільової функції (Objective Function) декількох змінних $f(x_1, x_2, \ldots x_n) = f(\mathbf{x})$. Більшисть чисельних методів розв'язку такої задачі передбачають ітеративну побудову послідовності наближень, яка прямує до точного значення мінімуму $\mathbf{x}^*$:

$$\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots \mathbf{x}^{(i)}, \ldots \to \mathbf{x}^*,$$

при цьому значення цільової функції в цих точках монотонно зменшуються:

$$f(\mathbf{x}^{(0)}) > f(\mathbf{x}^{(1)}) > \ldots f(\mathbf{x}^{(m)}) > \ldots > f(\mathbf{x}^*).$$

Загальною стратегією побудови наступного наближення є

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \sigma^{(m)} \mathbf{s}^{(m)},$$

де $\mathbf{s}^{(m)}$ – напрямок пошуку, а $\sigma^{(m)}$ (точніше, $\sigma^{(m)} |\mathbf{s}^{(m)}|$) – довжина кроку в цьому напрямку.

Відмінність між методами багатомірної оптимізації полягає в способах вибору $\sigma^{(m)}$ та $\mathbf{s}^{(m)}$.

Метод Хука-Дживса (Hooke, Jeeves) являє собою комбінацію *просування за зразком* (pattern move) та *досліджуючого пошуку на шаблоні* (exploratory move).

Перед початком кожної ітерації ми маємо точку поточного наближення **x** та вектор зразку **s** (можливо, навіть нульовий), в напрямку якого цільова функція *ймовірно* зменшується. Ітерація полягає в тому, що ми знаходимо нову точку **x**, в якій значення цільової функції буде меншим, та коректуємо величину і напрямок вектору зразку **s** на підставі інформації, отриманої в ході обчислень.

Ітерація починається з того, що ми переміщуємося від точки поточного наближення **x** на вектор зразку **s** (*просування за зразком*) і отриману точку $\mathbf{p} = \mathbf{x} + \mathbf{s}$ призначаємо як центр шаблону.

Шаблоном називають сукупність точок, що складається з центральної точки **p** та точок в околі навколо неї, де кожній з координат дається приріст $-h_i$, 0 або $+h_i$ (рис. 14.1). Будемо позначати сукупність додатніх величин $(h_1, h_2, \ldots h_n)$ як **h**, а сам шаблон з центром **p** і відхиленнями **h** – як {**p; h**}.

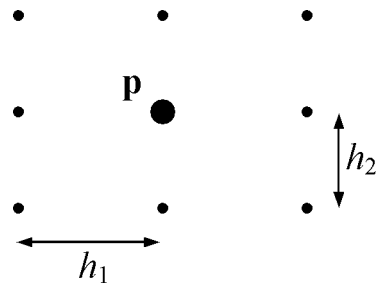


Рис. 14.1. Шаблон {**p; h**} для досліджуючого пошуку,
де $\mathbf{h} = \{h_1, h_2, \ldots h_n\}$

На цьому шаблоні проводиться *досліджуючий пошук*, який полягає в тому, що серед точок шаблону відшукують точку **b** (яку називають базовою), де значення цільової функції мінімальне. Не викючена ситуація, коли базова точка **b** співпадає з центром шаблону **p**. Базова точка служить кандидатом на наступне наближення до мінімуму. Якщо $f(\mathbf{b}) < f(\mathbf{x})$, то крок вважається вдалим, якщо ні – точка **b** відкидається.

Насправді, під час досліджуючого пошуку немає потреби перевіряти значеня цільової функції в *усіх* точках шаблону, їх кількість може бути значно зменшена.

Для цього треба на початку базову точку **b** помістити в центр шаблону **p**. Далі базовій точці надається варіація $\pm h_i$ в додатньому і від'ємному напрямках вздовж i-ї координатної осі, найкраща точка фіксується і базова точка переміщується туди. Такі варіації здійснюються

вздовж всіх координатних осей, і найкраща точка після останньої варіації розглядається як результат досліджуючого пошуку.

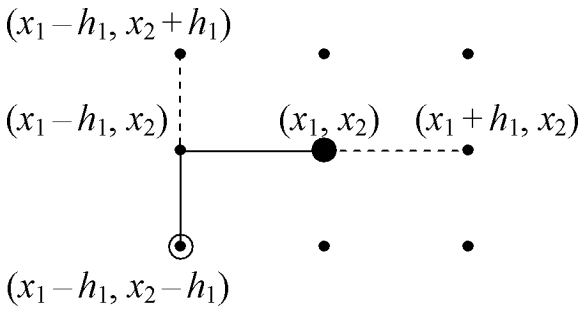


Рис. 14.2. Досліджуючий пошук на прикладі двох змінних.
Пунктирними лініями показані невдалі спроби
переміщення базової точки, а суцільними – успішні.
В частині точок шаблону цільова функція не обчислюється

Якщо крок вдалий, то це означає, що переміщення з точки **x** в точку **b** (яке визначалося вектором **s** та його корекцією за допомогою досліджуючого пошуку на величину, не більшу за **h**), було зроблено у вірному напрямку і при наступному просуванні цей напрямок доцільно зберегти, а довжину кроку – збільшити з метою пришвидшення пошуку (рис. 14.3.а). Таким чином, на наступній ітерації значення **s'** та **x'** визначаються як

$$\mathbf{s'} = \alpha\,(\mathbf{b} - \mathbf{x}),$$

де $\alpha > 1$ – *прискорюючий параметр* (зазвичай $\alpha \approx 2$), та

$$\mathbf{x'} = \mathbf{b}.$$

Якщо крок невдалий, то це означає, що напрямок вектору зразку **s** був неправильним, і точку **x** треба залишити без змін, а вектор **s** – обнулити (рис. 14.3.б):

$$\mathbf{s'} = \mathbf{0},$$

$$\mathbf{x'} = \mathbf{x}$$

і визначати новий напрямок пошуку виходячи з досліджуючого пошуку безпосередньо навколо точки **x** (рис. 14.3.в).

Якщо ж і в цьому випадку крок виявиться невдалим, тобто цільова функція на шаблоні {**p**; **h**} приймає найменше значення в його центрі, то це означає, що шаблон ймовірно накрив локальний мінімум (рис. 14.3.г). Таким чином, положення мінімума знайдено з точністю до **h.**

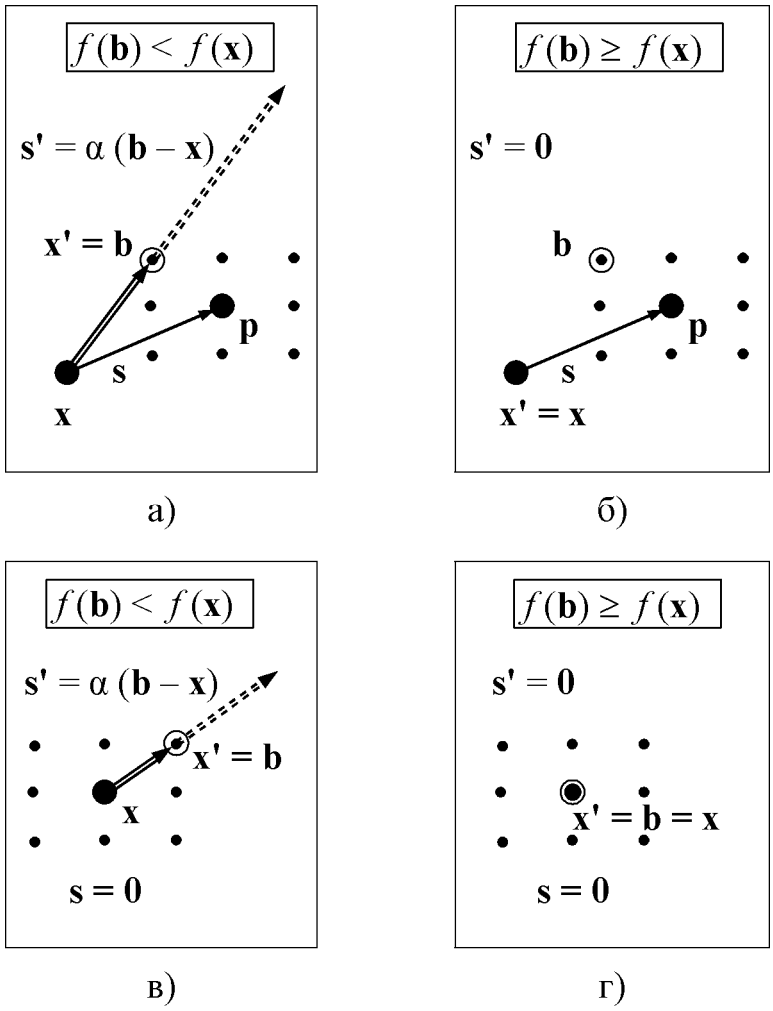


Рис. 14.3. Вдалий (а, в) і невдалий (б, г) крок метода Хука-Дживса при ненульовому (а, б) і нульовому (в, г) векторі зразку

Якщо ця точність є достатньою, тобто всі $h_i < \varepsilon_i$, де $(\varepsilon_1, \varepsilon_2, \ldots \varepsilon_n)$ – наперед задані малі числа, то пошук закінчується. В іншому випадку необхідно зробити *редукцію шаблону*, зменшивши величини всіх h_i:

$$\mathbf{h'} = \beta\, \mathbf{h},$$

де $\beta < 1$ – *параметр редукції шаблону* (зазвичай $\beta \approx 0.1$), і повторити досліджуючий пошук.

Загальна схема алгоритму Хука-Дживса представлена на рис. 14.4, а схема алгоритму досліджуючого пошуку – окремо на рис. 14.5.

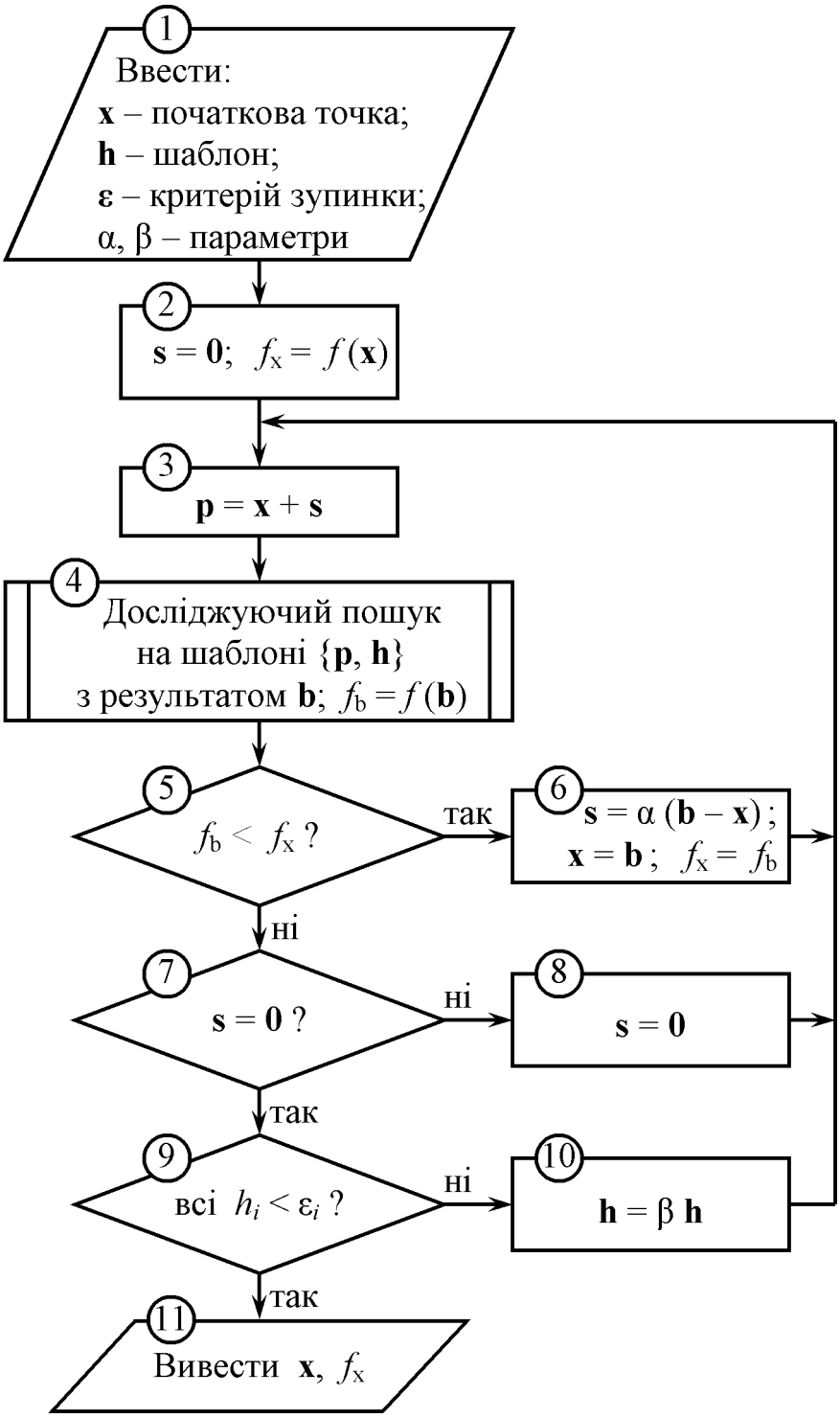


Рис. 14.4. Схема алгоритму Хука-Дживса

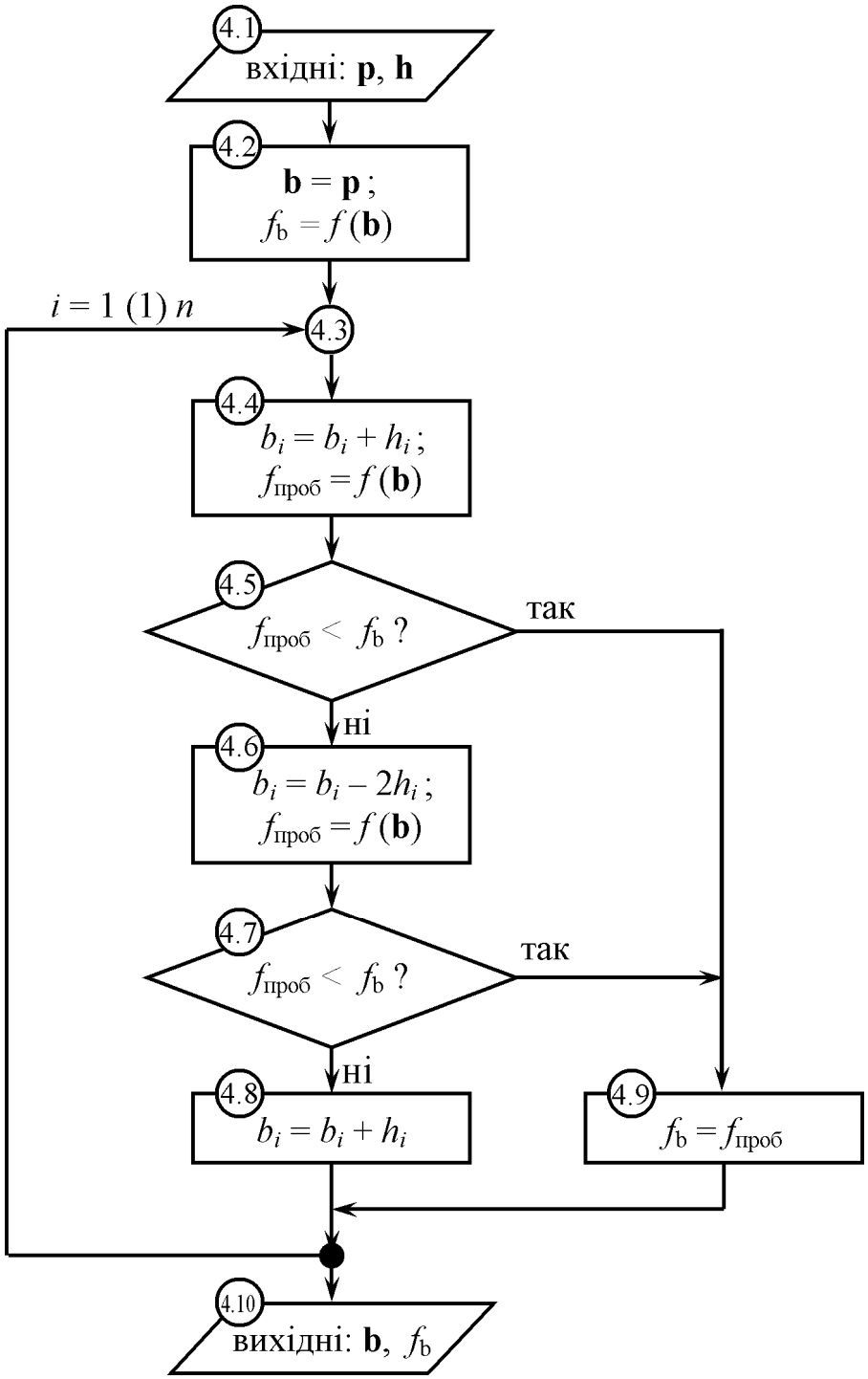


Рис. 14.5. Схема алгоритму досліджуючого пошуку.
На вході – центральна точка шаблону,
на виході – базова з мінімальним значенням цільової функції

Завдання

1. Однією із традиційних загальноуживаних тестових функцій для випробувань алгоритмів оптимізації функції кількох змінних є двовимірна функція Розенброка

$$f(x_1, x_2) = \left[1 - \frac{x_1}{a}\right]^2 + c\left[\frac{x_2}{b} - \left(\frac{x_1}{a}\right)^2\right]^2$$

Вона має улоговину вздовж кривої $\frac{x_2}{b} = \left(\frac{x_1}{a}\right)^2$ і глобальний мінімум в точці $(x_1, x_2) = (a, b)$ де $f(x_1, x_2) = 0$.

Побудуйте лінії рівня функції Розенброка з параметрами a, b, c згідно з вашим варіантом. Значення функції на ізолініях виберіть самостійно.

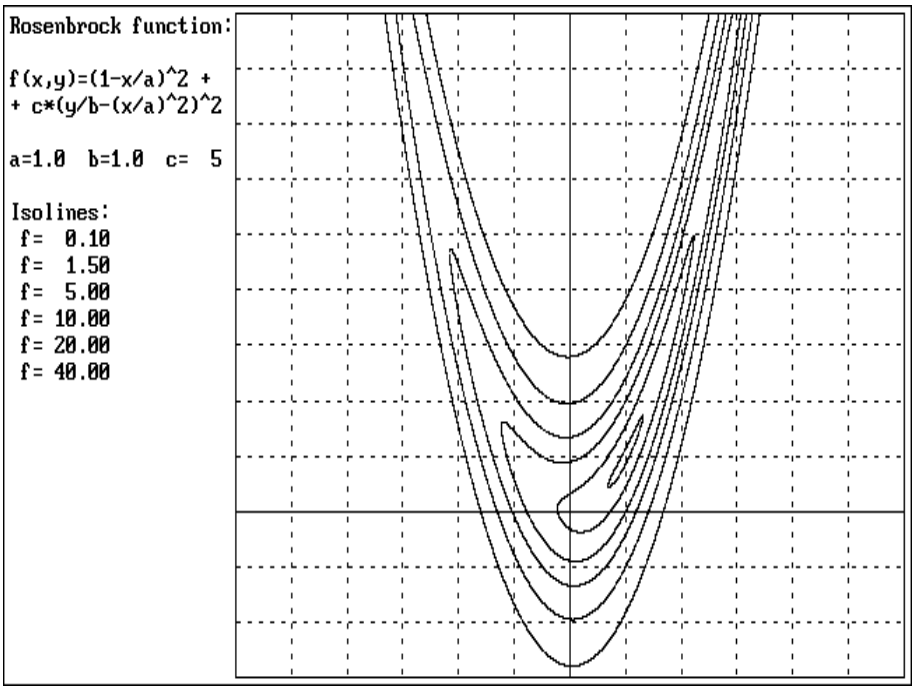


Рис. 14.6. Лінії рівня $f = 0.1, 1.5, 5, 10, 20, 40$
функції Розенброка з параметрами $a = 1$, $b = 1$, $c = 5$.
Координатна сітка нанесена з кроком 1

2. Уясніть призначення окремих блоків схеми алгоритму оптимізації функції $f(\mathbf{x})$ методом Хука-Дживса. Складіть програму, що реалізує цей алгоритм. Фрагмент програми, що власне відшукує мінімум, оформте у вигляді окремої процедури. Так само окремою процедурою оформте обчислення цільової функції.

```
SUB HookeJeeves(n,X(1),H(1),EPS(1),alpha,beta,fx)
'
' ------------------------------------------------------
' Пошук мінімуму функції n змінних методом Хука-Дживса.
'
' Вхідні параметри:
'    n      - кількість змінних;
'    X[n]   - початковий вектор аргументу,
'             при виході містить точку мінімуму;
'    H[n]   - початковий шаблон;
'    EPS[n] - критерій досягнутої точності;
'    alpha  - прискорюючий параметр;
'    beta   - параметр редукції шаблону;
' Вихідні параметри:
'    X[n]   - точка мінімуму;
'    fx     - значення цільової функції в точці мінімуму;
' Процедури, що використовуються:
'    ObjFunct(n,X(1),f) - обчислення цільової функції f(X).
' ------------------------------------------------------
...
' Декларування масивів вектору зразку S та базової точки B:
   DIM S[n], B[n]
...
END SUB

SUB ObjFunct(n,X(1),f)
'
' ------------------------------------------------------
' Обчислення цільової функції (Objective Function) f(X).
'
' Вхідні параметри:
'    n      - кількість змінних;
'    X[n]   - вектор аргументу;
' Вихідний параметр:
'    f      - значення цільової функції.
' ------------------------------------------------------
'
...
END SUB
```

3. Застосуйте алгоритм Хука-Дживса до функції Розенброка, розпочинаючи пошук із зазначеної початкової точки $\mathbf{x}_0$ згідно з вашим варіантом. Точність пошуку $(\varepsilon_1, \varepsilon_2)$ задайте $(10^{-3}, 10^{-3})$. Початкову величину шаблону **h**, прискорюючий параметр α та параметр редукції шаблону β виберіть самостійно.

   Проілюструйте хід обчислень на кресленні ізоліній. Відмічайте точки, в яких обчислюється цільова функція та додатково – ті базові точки, в яких крок пошуку виявився вдалим. Підрахуйте кількість тих і інших.

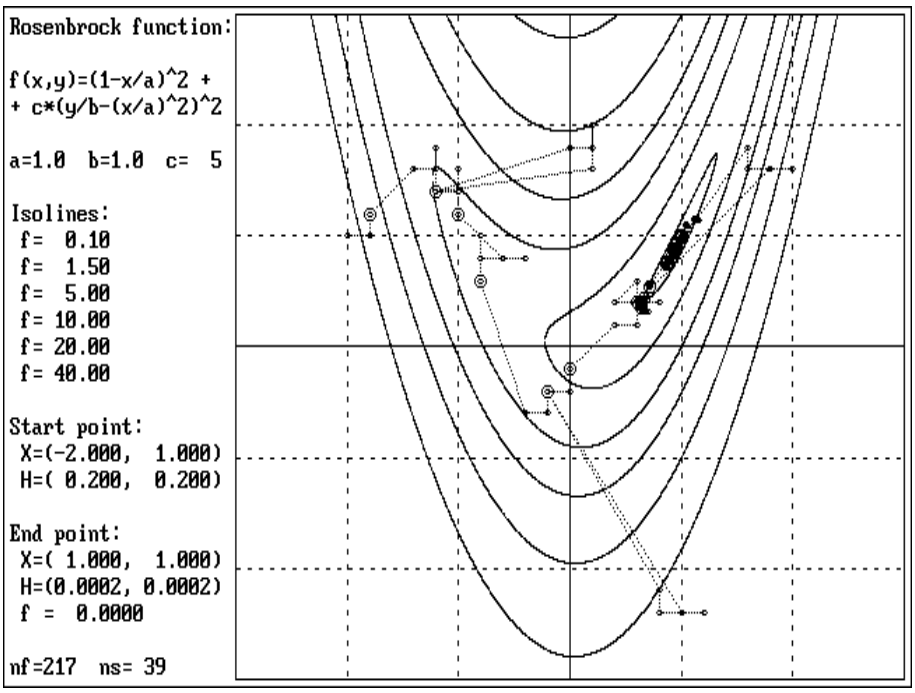


Рис. 14.7. Ілюстрація роботи алгоритму Хука-Дживса
на функції Розенброка.
Початкова точка $\mathbf{x} = (-2, +1)$, початковий шаблон $\mathbf{h} = (0.2, 0.2)$,
точність пошуку $(\varepsilon_1, \varepsilon_2) = (10^{-3}, 10^{-3})$, прискорюючий параметр $\alpha = 2$,
параметр редукції шаблону $\beta = 0.1$.

Точки, в яких обчислювалася цільова функція (їх 217),
відмічені та послідовно з'єднані пунктирною лінією.
Додатково великими пунсонами відмічені точки поточних наближень,
в яких цільова функція послідовно зменшується (їх 39)

Варіанти для самостійної роботи

Варіант	*a*	*b*	*c*	$\mathbf{x}_0$
1	1	0.5	1	(–1, 3)
2	0.5	0.5	2	(–1.5, 2)
3	1	1.5	3	(–1, 7.5)
4	3	1	4	(–7.5, 3)
5	2	1	5	(–2, 4.5)
6	2.5	1.5	6	(–6, 4)
7	2	2	7	(–2, 8)
8	2.5	2.5	8	(–5.5, 6)
9	2	3	9	(–2, 11.5)
10	1.5	1.5	10	(–3, 3.5)
11	1.5	2	11	(–1.5, 7)
12	3	2	12	(–6, 4)

Варіант	*a*	*b*	*c*	$\mathbf{x}_0$
13	4	2	13	(–4, 6.5)
14	4	2.5	14	(–7.5, 5)
15	2	1.5	15	(–2, 4.5)
16	3	1.5	16	(–5.5, 2.5)
17	3.5	2	17	(–3.5, 6)
18	3.5	2.5	18	(–6, 4)
19	2.5	3	19	(–2.5, 8)
20	2.5	2	20	(–4, 3)
21	2	2.5	21	(–2, 6.5)
22	3	2.5	22	(–5, 3.5)
23	3.5	3	23	(–3.5, 7.5)
24	3	3	24	(–4.5, 4)

Контрольні запитання

1. Покажіть, що лінії рівня $f = const$ функції Розенброка описуються співвідношеннями

$$x_2 = b\left(\left(\frac{x_1}{a}\right)^2 \pm \sqrt{\frac{f - \left(1 - \frac{x_1}{a}\right)^2}{c}}\right), \text{ де } a\left(1 - \sqrt{f}\right) \le x_1 \le a\left(1 + \sqrt{f}\right)$$

2. Скільки точок має шаблон {**p; h**} в *n*-вимірному просторі?
3. В скількох точках потрібно обрахувати ціьову функцію, щоб виконати досліджуючий пошук на шаблоні згідно з алгоритмом рис. 14.5 ? Порівняйте це число з кількістю точок всього шаблону.
4. З якою метою в алгоритм Хука-Дживса вводиться прискорюючий параметр α та параметр редукції шаблону β ?
5. Розгляньте функцію двох змінних $f(x_1, x_2) = x_1 + x_2 + 2|x_1 - x_2|$ і побудуйте схематично її лінії рівня. Дослідіть поведінку досліджуючого пошуку на шаблоні, центр якого припадає на пряму $x_1 = x_2$. Опишіть проблему, з якою ви стикнулись. Чи може така проблема з'явитися при застосуванні алгоритму Хука-Дживса до функції Розенброка? Запропонуйте шляхи її подолання.
6. Які ще алгоритми багатовимірної оптимізації вам відомі? Що в них спільного? Які переваги та недоліки притаманні цим алгоритмам?

Лабораторна робота № 15.

Інтерполяція даних. Інтерполяційний поліном Лагранжа. Рівномірне (чебишовське) наближення функцій

Мета роботи: застосування алгоритмів інтерполяції для побудови поліноміального наближення функції.

Що зробити: побудувати поліноміальне наближення до функції $f(x)$ за допомогою інтерполяційного полінома з вузлами, що розташовані на кривій $f(x)$. Дослідити величину дефекту наближення в залежності від числа вузлів. Додатково – порівняти випадок рівновіддалених вузлів та вузлів з абсцисами Чебишова.

Стислі теоретичні відомості

А. Поліноміальна інтерполяція

Інтерполяція – спосіб знаходження проміжних значень величини за наявним дискретним набором відомих значень.

Нехай маємо n точок (x_1, y_1), (x_2, y_2), (x_3, y_3), … (x_n, y_n). Задача інтерполяції полягає в пошуку такої функції $L(x)$ (*інтерполяційна функція* або *інтерполянт*), яка в кожній з абсцис x_i (що називаються вузлами інтерполяції) приймає значення y_i:

$$L(x_i) = y_i, \quad i = 1, 2, 3, \ldots n. \tag{1}$$

В такій постановці задача (1) має безліч розв'язків, тому на $L(x)$ накладають додаткові обмеження, вимагаючи, щоб вона належала певному класу функцій. Зокрема, задача *поліноміальної інтерполяції* полягає в побудові $L(x)$ у вигляді полінома мінімального степеня, що задовольняє умові (1).

Оскільки (1) являє собою систему з n рівнянь, поліном $L(x)$ має бути поліномом n–1 -го степеня

$$L_{n-1}(x) = c_1 + c_2 x + c_3 x^2 + \ldots + c_n x^{n-1} \tag{2}$$

з n невідомими коефіцієнтами c_1, c_2, c_3, … c_n, що визначаються цією системою. Можна показати, що у випадку, коли серед значень x_i немає двох рівних, система лінійних алгебраїчних рівнянь (СЛАР), що визначає

коефіцієнти c_i, є невиродженою, і, таким чином, сам поліном – єдиним степеня n.

Лагранж запропонував спосіб безпосереднього обчислення таких поліномів, без розв'язання СЛАР, але із записом не в канонічній формі (2), а у вигляді лінійної комбінації

$$L(x) = \sum_{j=1}^{n} y_j l_j(x) \ ,$$

де базисні поліноми визначаються за формулою:

$$l_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i} = \frac{x - x_1}{x_j - x_1} \cdot \ldots \cdot \frac{x - x_{j-1}}{x_j - x_{j-1}} \cdot \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdot \ldots \cdot \frac{x - x_n}{x_j - x_n} \ .$$

Очевидно, що $l_j(x)$ мають такі властивості:

- Це поліноми степеня n–1 ;
- $l_j(x_j) = 1$;
- $l_j(x_i) = 0$ при $i \neq j$.

Звідси випливає, що $L(x)$, як лінійна комбінація $l_j(x)$, може мати степінь не більший від n–1, та $L(x_i) = y_i$.

Отже, розв'язком задачі (1) є єдиний поліном n–1 -го степеня

$$L_{n-1}(x) = \sum_{j=1}^{n} y_j \left(\prod_{\substack{i=1 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i} \right) , \qquad (3)$$

який після розкриття дужок може бути приведений до канонічної форми (2). (Зауважимо, що для практичних цілей в цьому немає потреби.)

Завдання

1. Для побудови поліноміального наближення візьміть згідно з вашим варіантом ту ж саму функцію $f(x)$, яку ви досліджували при виконанні лабораторної роботи № 3. Виберіть самостійно межі інтервалу інтерполяції [a, b], на якому функція $f(x)$ неперервна і ніде не стає нескінченою.

2. Призначте на $[a, b]$ певне число n рівновіддалених вузлів (почніть з порівняно невеликого $n \sim 3\ldots4$)

$$x_i = a + \frac{b-a}{n-1}(i-1), \quad i = 1, 2, 3, \ldots n \tag{7}$$

   та обрахуйте в них значення функції $y_i = f(x_i)$.

3. Запрограмуйте обчислення інтерполяційного полінома $L_{n-1}(x)$ за формулою (3) і побудуйте графіки функції $f(x)$ та полінома $L_{n-1}(x)$ аналогічно тому, як ви це робили в лабораторній роботи № 3.

   Окремо в укрупненому масштабі побудуйте графік методичної похибки інтерполяції $f(x) - L_{n-1}(x)$ та зафіксуйте її найбільшу на $[a, b]$ абсолютну величину (див. рис. 15.1). Це максимальне відхилення інтерполюючого полінома від функції, яку він наближує, зазвичай називають *дефектом наближення D*.

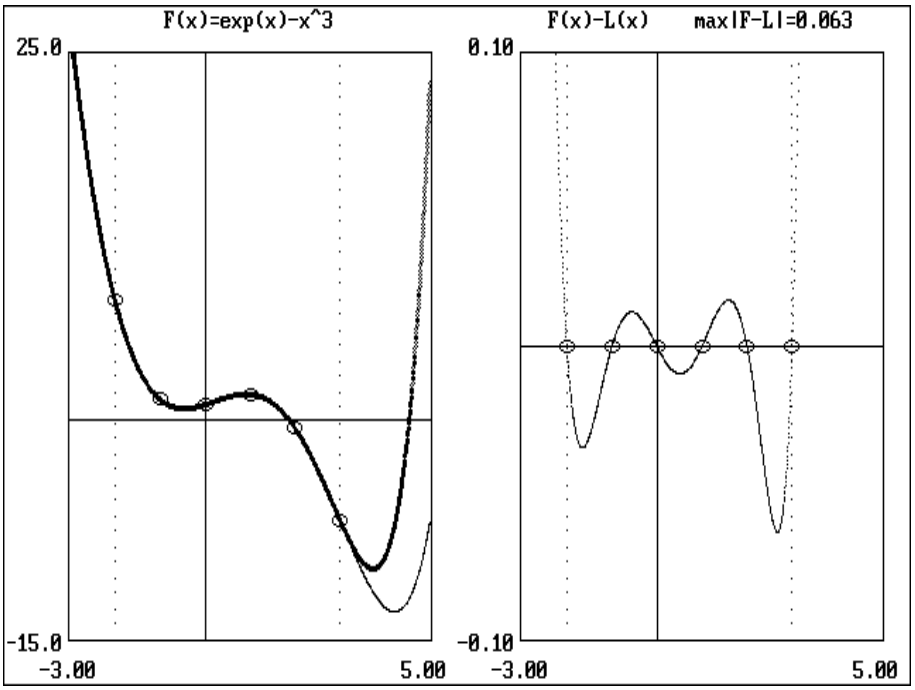


Рис. 15.1. Наближення функції $f(x) = e^x - x^3$ на інтервалі $[-2,\ 3]$ поліномом $L_5(x)$ 5-го степеня, що побудований як інтерполянт на 6 рівновіддалених вузлах з кроком 1.

Ліворуч: $f(x)$ – жирна лінія, $L_5(x)$ – тонка лінія.
На інтервалі інтерполяції лінії майже зливаються.
Праворуч: різниця $f(x) - L_5(x)$ в укрупненому масштабі по осі y

4. Збільшуючи число вузлів n до 10...20 дослідіть залежність дефекту наближення D від кількості вузлів. Поясніть закономірності, що спостерігаються. Чи прямує D до нуля, якщо кількість точок n необмежено зростає? (Не намагайтеся отримати дефект наближення D менший за машинний епсілон.)

Стислі теоретичні відомості (продовження)

Б. Похибка інтерполяції

Нехай y_i є значеннями деякої функції $f(x)$, визначеній на інтервалі $[a, b]$, що містить в собі всі вузли інтерполяції:

$$x_i \in [a, b], \quad f(x_i) = y_i, \quad i = 1, 2, 3, \ldots n.$$

Поставимо питання, наскільки добре інтерполяційний поліном $L_{n-1}(x)$ наближає функцію $f(x)$. Якщо $f(x)$ має неперервну n–у похідну $f^{(n)}(x)$ на інтервалі $[a, b]$, то різниця $R(x) = f(x) - L_{n-1}(x)$ (*залишковий член*, або *методична похибка* інтерполяції) оцінюється як

$$R(x) = f(x) - L_{n-1}(x) = \frac{f^{(n)}(\xi)}{n!}\omega_n(x), \tag{4}$$

де ξ – деяка точка в межах $[a, b]$, а

$$\omega_n(x) = \prod_{i=1}^{n}(x - x_i) = x^n + \ldots \tag{5}$$

– поліном n-го степеня з коефіцієнтом 1 при x^n, що приймає нульові зачення у всіх вузлах x_i.

Оскільки точне положення ξ залежить від x, ця формула дозволяє лише оцінити верхню межу похибки:

$$|R(x)| \le \frac{M_n}{n!}\omega_n(x), \tag{6}$$

де $M_n = \max\limits_{x \in [a,b]} \left| f^{(n)}(x) \right|$.

В. Рівномірне наближення функцій.

Абсолютним відхиленням D полінома $P(x)$ від функції $f(x)$ на інтервалі $[a, b]$ називають максимальне значення абсолютної величини різниці між ними на цьому інтервалі:

$$D = \max_{x \in [a, b]} \left| f(x) - P(x) \right|$$

Якщо функція $f(x)$ неперервна на конечному інтервалі $[a, b]$, то, згідно з *теоремою Вейерштраса про апроксимацію*, збільшенням степеня полінома $P(x)$ можна досягти того, що D (яке в цьому контексті називають *дефектом наближення*) стане скільки завгодно малим.

Задача *найкращого рівномірного наближення* полягає в тому, щоб при фіксованому степені апроксимуючого полінома $P(x)$ досягти якомога меншого дефекту наближення Δ.

Теорема Чебишова про альтернанс стверджує, що поліном найкращого рівномірного наближення m-го степеня $P_m(x)$ має ту властивість, що принаймні в $m+2$ точках інтервалу $[a, b]$ $\tilde{x}_i$, $i = 0, 1, 2, \ldots m+1$, занумерованих послідовно за зростанням величини

$$a \le \tilde{x}_0 < \tilde{x}_1 < \ldots < \tilde{x}_{m+1} \le b,$$

відхилення $f(x) - P_m(x)$ приймає своє максимальне за модулем значення, причому його знаки в цих точках чергуються:

$$f(\tilde{x}_i) - P_m(\tilde{x}_i) = \pm \Delta(-1)^i, \quad i = 0, 1, 2, \ldots m, m+1.$$

Сукупність цих $m+2$ точок x_i називають *чебишовським альтернансом* (рис. 15.2, вгорі).

Процедура точного відшукання коефіцієнтів полінома найкращого рівномірного наближення m-го степеня $P_m(x)$ довільної функції $f(x)$ досить громіздка, тому на практиці застосовують наближені методи, що дозволяють побудувати наближуючий поліном, хоча й не найкращий, але з дефектом D, близьким до мінімально можливого.

Одним із способів відшукання такого наближуючого полінома є побудова інтерполяційного полінома на певних вузлах x_i (див. рис. 15.2, внизу).

Зафіксуємо степінь наближуючого полінома $m = n-1$ і поставимо питання, яким чином слід вибирати вузли інтерполяції x_i, $i = 1, 2, \ldots n$, щоб верхня межа методичної похибки інтерполяції $R(x) = f(x) - P_{n-1}(x)$ (4) була якнайменшою.

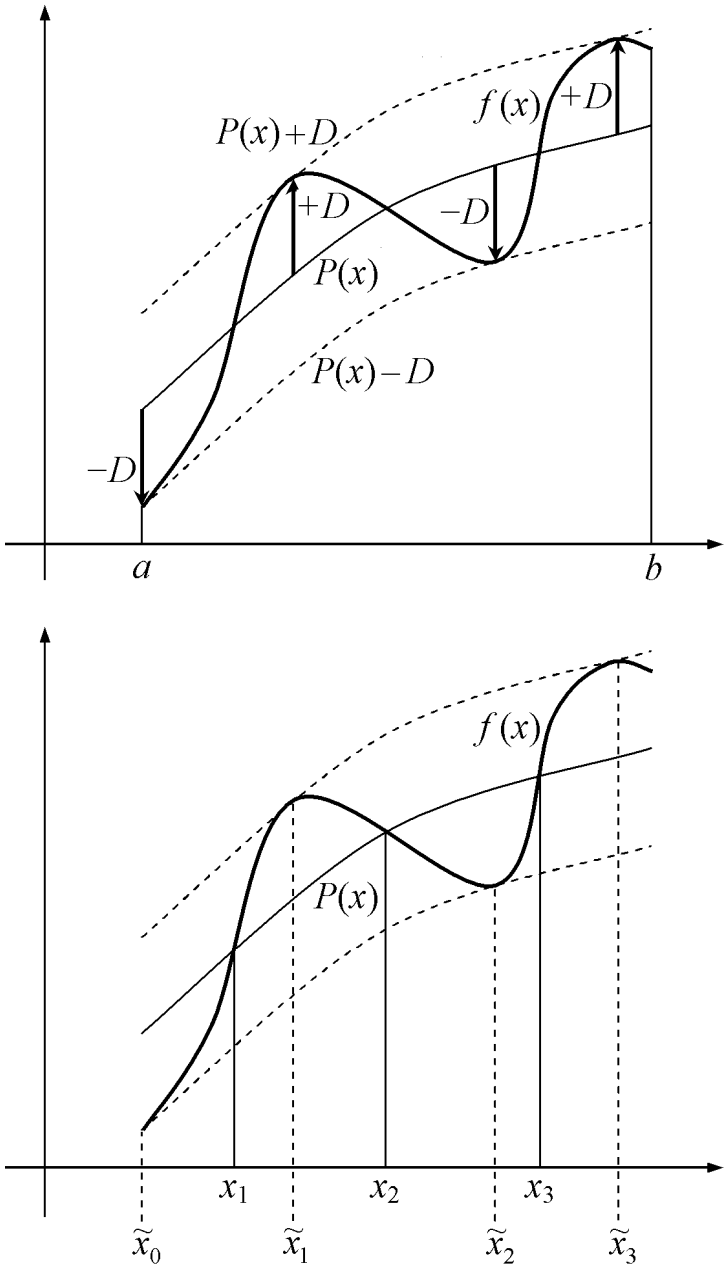


Рис. 15.2. Найкраще рівномірне наближення функції $f(x)$ (жирна лінія) поліномом $P(x)$ (тонка суцільна лінія) на інтервалі $[a, b]$.

Графік функції $f(x)$ затиснутий в смузі між кривими $P(x)+D$ та $P(x)-D$, і напівширина цієї смуги D служить мірою якості наближення.

Вгорі: в точках чебишовського альтернансу
відхилення $|f(x) - P(x)|$ сягають найбільшої величини D,
причому знаки різниці в цих точках чергуються.

Внизу: чебишовський альтернанс $\widetilde{x}_i$, $i = 0, 1, 2, \ldots$ та вузли інтерполяції x_i , $i = 1, 2, \ldots$ для полінома найкращого рівномірного наближення

Вочевидь, основний характер поведінки цієї похибки визначається поліномом $\omega_n(x)$ (5), а множник перед ним є величиною, що змінюється порівняно повільно, зокрема, якщо функція $f(x)$ сама є поліномом n-го степеня, то множник є константою, оскільки в цьому випадку $f^{(n)}(x) = const = M_n$.

Таким чином, задача відшукання полінома найкращого рівномірного наближення зводиться до пошуку такого набору вузлів x_i, що є коренями полінома $\omega_n(x)$, при якому сам поліном $\omega_n(x)$ найменше ухиляється від нуля.

Зазначимо, що при рівновіддалених вузлах типова поведінка цього полінома характеризується великими коливаннями на краях інтервалу інтерполяції і значно меншими – всередині (див. рис. 15.3, ліворуч), тому такий вибір вузлів є завідомо не найкращим.

Г. Поліноми Чебишова

Відомо (*теорема Чебишова*), що серед поліномів степеня n з коефіцієнтом 1 при x^n найменше ухиляється від нуля в діапазоні [–1, +1], а саме, на $\frac{1}{2^{n-1}}$, поліном $\frac{T_n(x)}{2^{n-1}}$, де $T_n(x)$ – поліном Чебишова, що визначається як

$$T_n(\cos\theta) = \cos(n\theta) \tag{8}$$

або ж

$$T_n(x) = \cos(n \arccos x), \quad |x| \le 1.$$

Зокрема,

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1 \\ &\dots \end{aligned}$$

і взагалі,

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{9}$$

Всі n нулів полінома $T_n(x)$ містяться на на відрізку [–1, +1] і становлять

$$x_i = \cos\left(\frac{n - i + ½}{n}\pi\right), \qquad i = 1, 2, \dots n. \tag{10}$$

Ці значення називають абсцисами або вузлами Чебишова. Вони розміщені нерівномірно на проміжку і ущільнюються на його кінцях (див. рис. 15.3, праворуч).

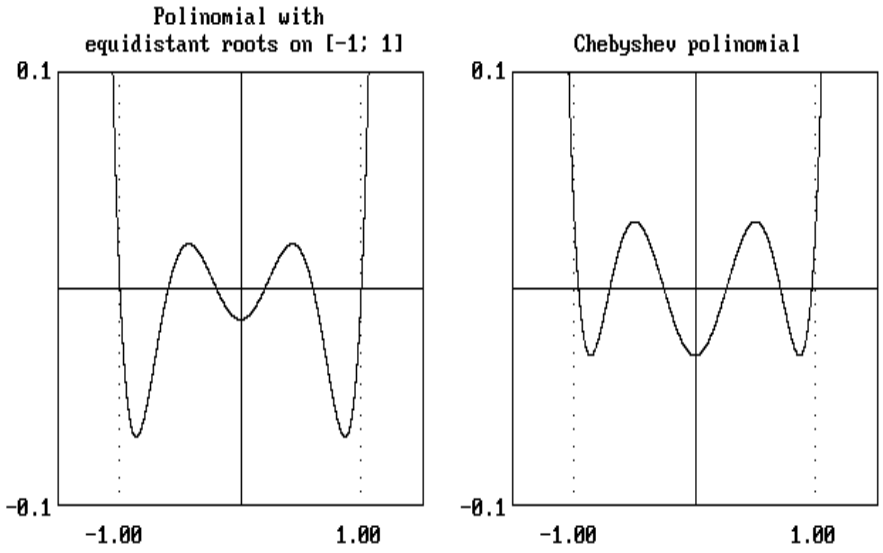


Рис. 15.3. Ліворуч: поліном 6-го порядку з рівновіддаленими коренями, розташованими на інтервалі [–1, +1] з кроком 2/5.
Праворуч: поліном Чебишова $T_6(x)/32$.
В обох випадках коефіцієнт при x^6 дорівнює 1.

Отже, якщо інтерполяційний поліном $P_{n-1}(x)$ з n вузлами в інтервалі [–1, +1] побудовано на чебишовських вузлах інтерполяції (10), то цей поліном є *близьким* до полінома найкращого рівномірного наближення, оскільки множник $\omega_n(x)$ в методичній похибці (4) є поліномом, що найменше ухиляється від 0. Такий поліном $P_{n-1}(x)$ буде *в точності* поліномом найкращого рівномірного наближення у випадку, якщо наближувана функція $f(x)$ сама є поліномом n-го степеня.

Якщо ж задача сформульована для інтервалу $[a, b]$, то вона зводиться до задачі на [–1, +1] перетворенням

$$x = \frac{a+b}{2} + \left(\frac{b-a}{2}\right)x'; \qquad x' \in [-1, +1], \qquad x \in [a, b],$$

тобто, замість (10), вузли інтерполяції слід вибрати як

$$x_i = \frac{a+b}{2} + \left(\frac{b-a}{2}\right)\cos\left(\frac{n-i+½}{n}\pi\right), \quad i = 1, 2 \ldots n. \tag{11}$$

Додаткове завдання

5. Виконайте пп. 1–4, беручи такі ж самі кількості вузлів, але визначаючи їх як абсциси Чебишова за формулою (11) (див. рис. 15.4). Порівняйте отримані результати.

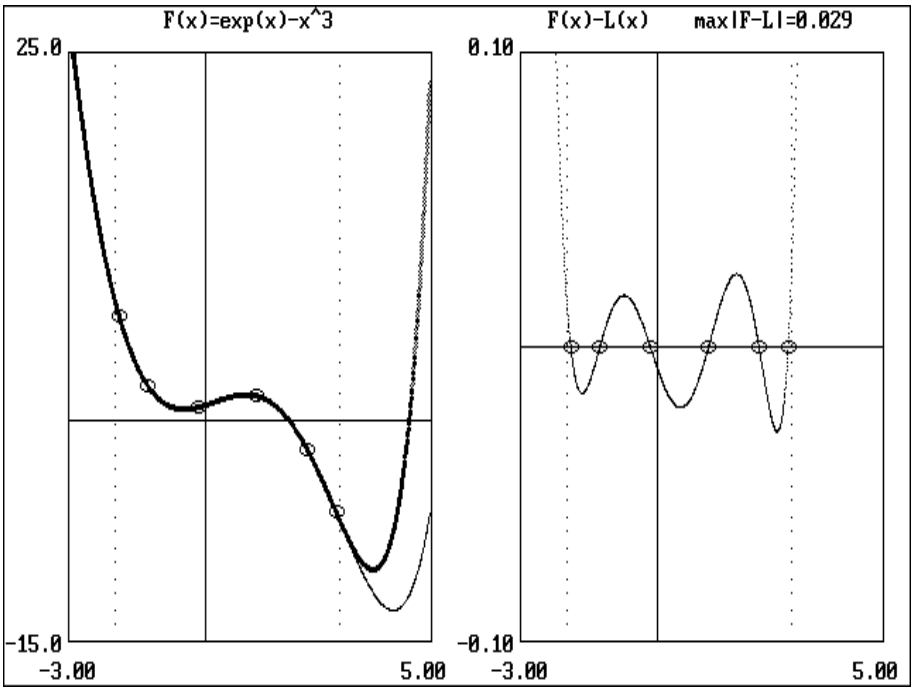


Рис. 15.4. Те ж саме, що на рис. 15.1,
але $L_5(x)$ побудований як інтерполянт на 6 вузлах,
що визначаються коренями полінома Чебишова $T_6(x)$

6. Проаналізуйте залежність дефекту наближення D від способу призначення вузлів. Отримані результати зручно аналізувати, якщо вони представлені у вигляді таблиці на зразок

n	D	
	рівновідділені вузли	вузли – абсциси Чебишова
3	…	…
4	…	…
5		
…		

Поясніть закономірності, що спостерігаються.

Контрольні запитання

1. Що являє собою задача інтерполяції? Опишіть загальну постановку такої задачі. Чи має вона єдиний розв'язок?
2. Що являє собою задача поліноміальної інтерполяції? За яких додаткових умов вона має єдиний розв'язок?
3. Складіть СЛАР для визначення коефіцієнтів c_i інтерполяційного полінома в канонічній формі (2).
4. Доведіть, що визначник матриці цієї СЛАР дорівнює $\prod_{1 \le i < j \le n} (x_j - x_i)$ (визначник Вандермонда). Він дорівнює нулю тоді і тільки тоді, коли серед x_i є хоча б одна пара рівних значень.
5. Опишіть постановку задачі інтерполяції, якщо відомо, що точки (x_0, y_0), (x_1, y_1), … (x_n, y_n) представляють періодичну функцію $f(x) = f(x+\tau)$. Яким класом функцій, на ваш погляд, доцільно обмежити інтерполянт $L(x)$?
6. Що таке похибка інтерполяції?
7. Що таке рівномірне наближення функції? Яким чином алгоритм побудови інтерполяційного полінома може бути застосовано для розв'язання задачі рівномірного наближення функції?
8. Базуючись на визначенні поліномів Чебишова (8) доведіть рекурентне співвідношення (9).
9. Доведіть, що в поліномі Чебишова $T_n(x)$ при $n \ge 1$ коефіцієнт при x^n дорівнює 2^{n-1}.
10. Доведіть, що поліном Чебишова $T_n(x)$ є парною функцією при парному n, і непарною – при непарному n.
11. Доведіть, що всі n нулів полінома Чебишова $T_n(x)$ дійсно містяться на відрізку [–1, +1] і визначаються формулою (10).
12. Доведіть, що поліном Чебишова $T_n(x)$ має n–1 екстремумів, всі вони також містяться в інтервалі [–1, +1] і становлять

$$x_i = \cos\left(\frac{n-i}{n}\pi\right), \qquad i = 1, 2 \ldots n-1 \tag{12}$$

В цих точках поліном приймає почергово значення ±1, тобто всі максимуми рівні +1 і всі мінімуми рівні –1. Крім точок екстремумів $T_n(x)$ приймає значення ±1 ще тільки в 2 точках на кінцях відрізка [–1, +1], які можна також визначати за формулою (12), покладаючи $i = 0$ та $i = n$.

Отже, $|T_n(x)| \leq 1$ для $|x| \leq 1$ та $|T_n(x)| > 1$ для $|x| > 1$.

13. Помножувачем частоти називають нелінійний електричний чотириполюсник, у якого при подачі на вхід гармонічного (синусоїдального) сигналу на виході утворюється також гармонічний сигнал, але n-кратної частоти, де n – певне ціле число. Якою має бути статична передаточна характеристика цього чотириполюсника?

Лабораторна робота № 16.

Кусково-поліноміальна інтерполяція. Кубічні сплайни

Мета роботи: застосування алгоритмів кусково-поліноміальної інтерполяції поліномами 1-го та 3-го степеня для побудови наближення функції.

Що зробити: побудувати поліноміальне наближення до функції $f(x)$ за допомогою кусково-лінійної інтерполяції з вузлами, що розташовані на кривій $f(x)$. Дослідити величину дефекту наближення в залежності від числа вузлів. Додатково – побудувати і дослідити сплайн-інтерполянт.

Стислі теоретичні відомості

А. Кусково-лінійна інтерполяція

При намаганні застосувати поліноміальну інтерполяцію на великій кількості вузлів (наприклад, для великих таблиць) сильно зростає степінь інтерполяційних поліномів, що робить їх незручними для обчислень. Уникнути цієї проблеми можна розбивши інтервал інтерполяції на кілька відрізків з побудовою на кожному з них окремого інтерполяційного полінома.

Найпростішим видом такого роду інтерполяції є *кусково-лінійна інтерполяція*. Нехай задані $n+1$ точок (x_0, y_0), (x_1, y_1), … (x_n, y_n), абсциси яких занумеровані за зростанням

$$x_0 < x_1 < \ldots < x_n .$$

Ці вузли з'єднуються прямолінійними ланками, тобто інтерполянт на кожному інтервалі є лінійною функцією $p_i(x)$, а загалом він являє собою неперервну ламану лінію з вершинами у цих точках (рис. 16.1).

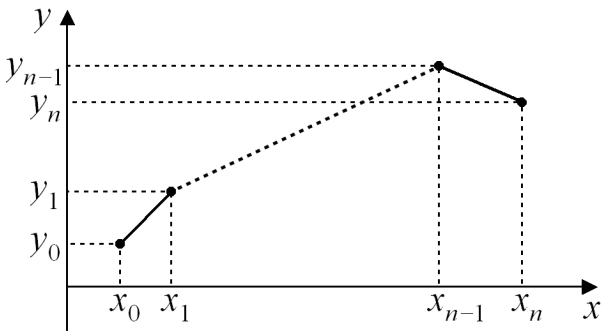


Рис. 16.1. Кусково-лінійна інтерполяція

Рівняння прямолінійної ланки на *i*-му інтервалі зручно записувати, використовуючи нормалізовану змінну

$$t_i = \frac{x - x_{i-1}}{x_i - x_{i-1}} = \frac{x - x_{i-1}}{h_i}, \qquad (1)$$

де h_i – довжина *i*-го відрізку (рис. 16.2). В межах цього відрізку змнна t_i приймає значення від 0 до 1.

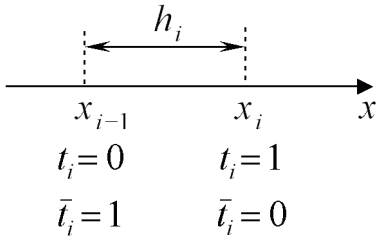


Рис. 16.2. Нормалізовані змінні t_i і $\bar{t}_i = 1 - t_i$ на *i*-му інтервалі

Легко бачити, що на *i*-му відрізку шукана пряма лінія, яка проходить через точки (x_{i-1}, y_{i-1}) та (x_i, y_i), має рівняння

$$p_i(x) = y_{i-1}(1 - t_i) + y_i t_i \ .$$

Якщо ввести додаткову змінну

$$\bar{t}_i = 1 - t_i, \qquad (2)$$

то це рівняння запишеться в красивій симетричній формі:

$$p_i(x) = y_{i-1}\bar{t}_i + y_i t_i \qquad (3)$$

Завдання

1. Для побудови кусково-лінійного наближення візьміть згідно з вашим варіантом ту ж саму функцію $f(x)$, яку ви досліджували при виконанні лабораторної роботи № 3. Виберіть самостійно межі інтервалу інтерполяції $[a, b]$, на якому функція $f(x)$ неперервна і ніде не стає нескінченою.

2. Розділіть $[a, b]$ на певне порівняно невелике число n відрізків ($n \sim 4\ldots6$), призначивши $n+1$ вузлів, зокрема, $x_0 = a$, $x_n = b$. Почніть з рівновіддалених вузлів

$$x_i = a + \frac{b-a}{n} i, \quad i = 0, 1, 2, \ldots n, \tag{4}$$

   але передбачте в програмі просту можливість призначення інших значень числу n та вузлам x_i (з клавіатури, файлу або безпосередньо в тексті програми – як ви вважаєте за доцільне). Обрахуйте у вузлах значення функції $y_i = f(x_i)$.

3. Запрограмуйте обчислення куково-лінійної інтерполяції за формулою (3) і побудуйте графіки функції $f(x)$ та інтерполяційної ламаної $p(x)$ аналогічно тому, як ви це робили в лабораторній роботі № 3.

   Окремо в укрупненому масштабі побудуйте графік похибки інтерполяції $f(x) - p(x)$ та зафіксуйте дефект наближення D, тобто найбільшу абсолютну величину відхилення наближуючої ламаної від функції.

4. Змінюючи положення вузлів інтерполяції спробуйте домогтися *рівномірного* наближення з якнайменшим дефектом D (одна з таких спроб показана на рис. 16.3). Дефект буде найменшим, якщо на кожному відрізку максимальні відхилення $p(x)$ від $f(x)$ будуть приблизно однаковими. Порівняйте це значення D з тим, що було при рівновіддалених вузлах.

5. Поверніться до рівновіддалених вузлів (4). Збільшіть кількість відрізків n в приблизно 3, 10, 30 і 100 разів порівняно з початковим значенням. Як веде себе дефект наближення D?

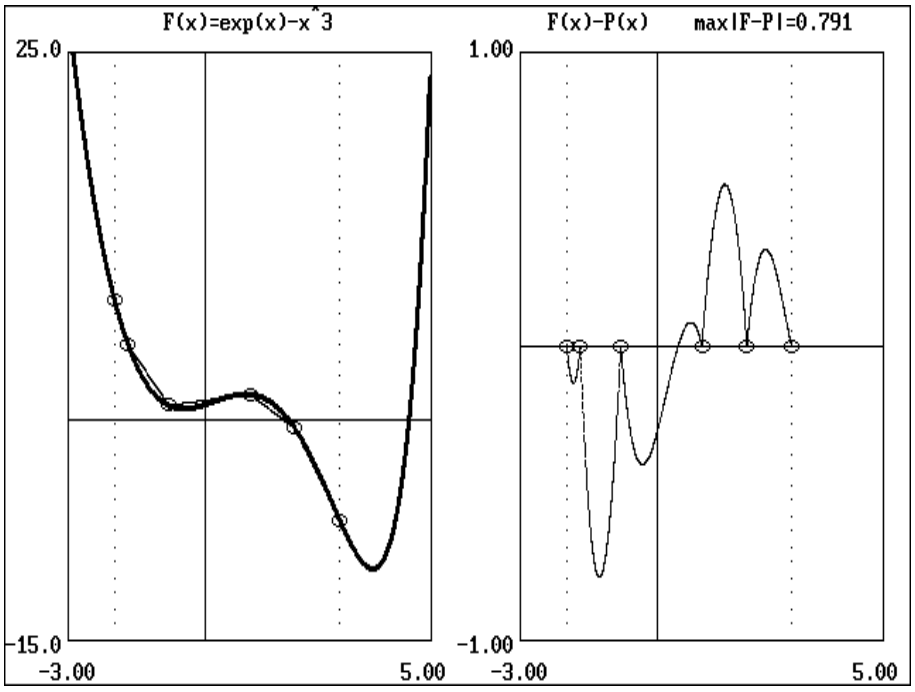


Рис. 16.3. Наближення функції $f(x) = e^x - x^3$ на інтервалі [–2, 3]
кусково-лінійною інтерполяцією $p(x)$
на n=5 відрізках (6 вузлах)
x_i = –2.0, –1.7, –0.8, +1.0, +2.0, +3.0 .

Ліворуч: $f(x)$ – жирна лінія, $p(x)$ – тонка лінія.
На інтервалі інтерполяції лінії майже зливаються.
Праворуч: різниця $f(x) - p(x)$ в укрупненому масштабі по осі y

Стислі теоретичні відомості (продовження)

Б. Сплайн-інтерполяція

При кусково-лнійній інтерполяції на стиках окремих ланок відбувається розрив похідних. Цього недоліку позбавлена так звана *сплайн-інтерполяція*.

На кожному відрізку інтерполянт є поліномом 3-го степеня $s_i(x)$. Поліноми зшиваються один з одним таким чином, щоб результуючий інтерполянт проходив через вузли інтерполяції, а також щоб на стиках суміжних ланок залишалися неперервними не тільки сама інтерполююча функція, але також її перша та друга похідні:

$$s_1(x_0) = y_0\,, \tag{5.1}$$

$$s_i(x_i) = s_{i+1}(x_i) = y_i\,, \qquad i = 1, 2, \ldots n-1, \tag{5.2}$$

$$s_i'(x_i) = s_{i+1}'(x_i)\,, \qquad i = 1, 2, \ldots n-1, \tag{5.3}$$

$$s_i''(x_i) = s_{i+1}''(x_i)\,, \qquad i = 1, 2, \ldots n-1, \tag{5.4}$$

$$s_n(x_n) = y_n\,. \tag{5.5}$$

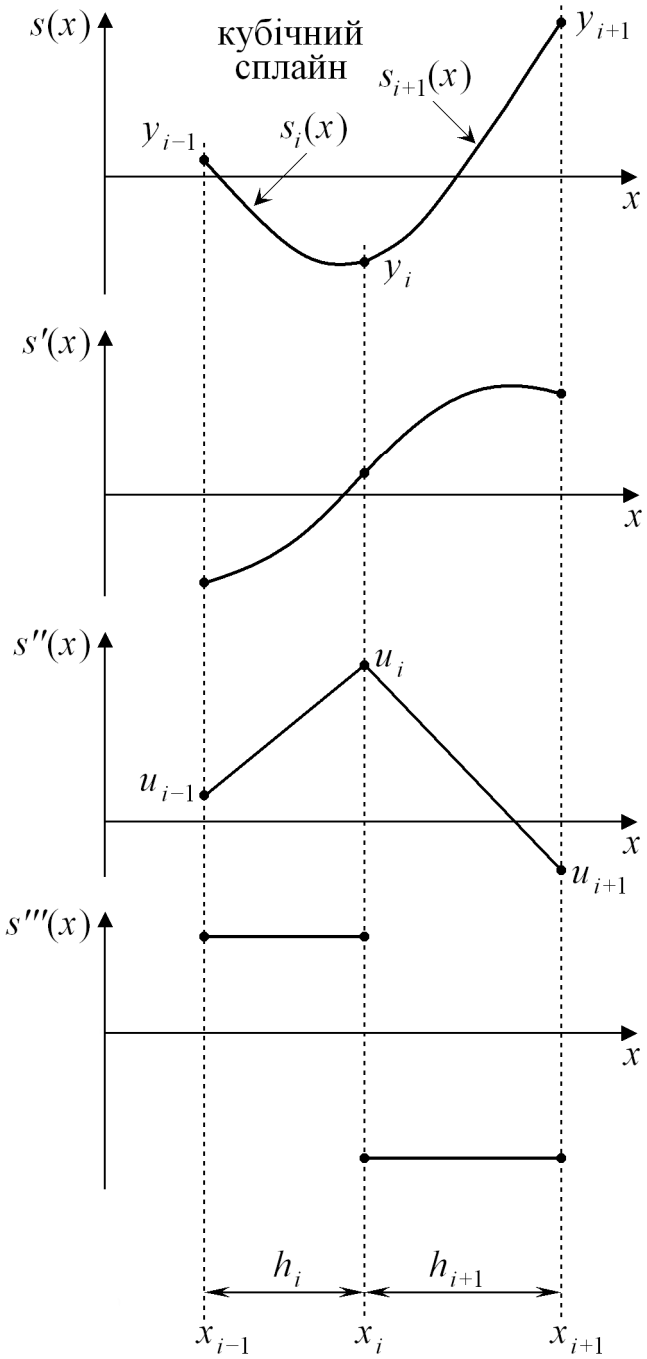


Рис. 16.4. Кубічний сплайн та три його похідних
на *i*-му та *i*+1-му відрізках

Визначення кожного з поліномів 3-го степеня $s_i(x)$ потребує 4 коефіцієнтів, тобто для опису всієї сукупності n поліномів знадобиться $4n$ параметрів. В той же час кількість рівнянь в (5) становить лише $1 + 2(n-1) + (n-1) + (n-1) + 1 = 4n-2$. Таким чином, іще дві умови мають бути накладені додатково.

Сплайни мають просту механічну аналогію. Саме такої форми набуває гнучка металічна лінійка (*сплайн*, spline), якщо її поставити на ребро та закріпити в точках (x_i, y_i), а в іншому полишити напризволяще. При цьому на кожному відрізку вона набуває форми фрагмента кубічної параболи, а її потенційна енергія пружної деформації, що пропорційна $\int_{x_0}^{x_n} y''(x)dx$, стає мінімальною.

Рівняння кубічного сплайна на i-му відрізку будемо записувати не в канонічній формі

$$s_i(x) = c_{1i} + c_{2i}x + c_{3i}x^2 + c_{4i}x^3,$$

а за допомогою інших чотирьох параметрів – значень на кінцях відрізка самої функції y_{i-1}, y_i та її другої похідної u_{i-1}, u_i.

Без обмеження загальності розглянемо сплайн $s_1(x)$ на відрізку $[x_0, x_1]$. Для полегшення проміжних записів індекс 1 при $s_1(x)$, а також при h_1, t_1, $\bar{t}_1$, які визначаються через (1), (2), писати не будемо.

З огляду на те, що $s''(x)$ є лінійною функцією, згідно з (3)

$$s''(x) = u_0\bar{t} + u_1 t\,.$$

Проінтегруємо цей вираз двічі, щоб відновити $s'(x)$ та $s(x)$. Оскільки

$$dx = h\cdot dt = -h\cdot d\bar{t}\,,$$

$$s'(x) = \int s''(x)dx = -u_0 h\int \bar{t}\,d\bar{t} + u_1 h\int t dt = -u_0 h\frac{\bar{t}^2}{2} + u_1 h\frac{t^2}{2} + const\,,$$

де $const$ – константа інтегрування;

$$s(x) = \int s'(x)dx = u_0 h^2\int\frac{\bar{t}^2}{2}d\bar{t} + u_1 h^2\int\frac{t^2}{2}dt + \int const\cdot dx =$$

$$= \frac{h^2}{6}\left(u_0\bar{t}^3 + u_1 t^3\right) + line(x)\,,$$

де $line(x)$ – довільна лінійна функція з двома коефіцієнтами, які походять від двох констант інтегрування.

Визначимо вигляд функції $line(x)$ з огляду на те, що $s(x)$ на кінцях відрізка $[x_0, x_1]$ має приймати значення y_0, та y_1.

$$s(x_0) = y_0 = \frac{h^2}{6}u_0 + line(x_0), \qquad s(x_1) = y_1 = \frac{h^2}{6}u_1 + line(x_1),$$

звідки

$$line(x_0) = y_0 - \frac{h^2}{6}u_0, \qquad line(x_1) = y_1 - \frac{h^2}{6}u_1,$$

тому, аналогічно (3),

$$line(x) = \left(y_0 - \frac{h^2}{6}u_0\right)\bar{t} + \left(y_1 - \frac{h^2}{6}u_1\right)t.$$

Тепер можна записати остаточний вираз для $s(x)$:

$$s(x) = y_0\,\bar{t} + y_1 t + \frac{h^2}{6}\Big[u_0(\bar{t}^3 - \bar{t}) + u_1(t^3 - t)\Big].$$

Відновлюючи індекси, повертаємося до довільного i-го відрізка $[x_{i-1}, x_i]$ та диференціюємо, щоб отримати вирази для першої та третьої похідної сплайна (див. рис. 16.4):

$$s_i(x) = y_{i-1}\,\bar{t}_i + y_i\,t_i + \frac{h_i^2}{6}\Big[u_{i-1}(\bar{t}_i^3 - \bar{t}_i) + u_i(t_i^3 - t_i)\Big] =$$

$$= y_{i-1}\,\bar{t}_i + y_i\,t_i - t_i\bar{t}_i\frac{h_i^2}{6}\Big[u_{i-1}(\bar{t}_i + 1) + u_i(t_i + 1)\Big], \qquad (6.1)$$

$$s_i'(x) = \frac{y_i - y_{i-1}}{h_i} - \frac{h_i}{6}\Big[u_{i-1}(3\bar{t}_i^2 - 1) - u_i(3t_i^2 - 1)\Big], \qquad (6.2)$$

$$s_i''(x) = u_{i-1}\bar{t} + u_i\,t, \qquad (6.3)$$

$$s_i'''(x) = \frac{u_i - u_{i-1}}{h_i}, \qquad i = 1, 2, \ldots n. \qquad (6.4)$$

Зауважимо, що в (6.1) перші два доданки являють собою просто кусково-лінійну інтерполяцію на відрізку $[x_{i-1}, x_i]$ (див. (3)), а третій доданок, який дорівнює нулю на кінцях i-го відрізку, – кубічну добавку, що забезпечує додаткову гладкість.

Легко бачити, що сплайни, визначені таким чином, задовольняють умовам (5.1), (5.2), (5.4) та (5.5) щодо неперервності функції та її другої похідної незалежно від значень $u_0, u_1, \ldots u_n$. Визначимо тепер сукупність значень $\{u_i\}$ таким чином, щоб забезпечити умови (5.3) щодо неперервності першої похідної. Згідно з (6.2) значення похідних в точці x_i зліва та справа становлять:

$$s_i'(x_i) = s_i'(x)\big|_{t_i=1} = \frac{y_i - y_{i-1}}{h_i} + \frac{h_i}{6}\left[u_{i-1} + 2u_i\right], \qquad (7.1)$$

$$s_{i+1}'(x_i) = s_{i+1}'(x)\big|_{t_{i+1}=0} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{h_{i+1}}{6}\left[2u_i + u_{i+1}\right]. \qquad (7.2)$$

Отже,

$$\frac{y_i - y_{i-1}}{h_i} + \frac{h_i}{6}\left[u_{i-1} + 2u_i\right] = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{h_{i+1}}{6}\left[2u_i + u_{i+1}\right],$$

або ж

$$h_i u_{i-1} + 2(h_i + h_{i+1})u_i + h_{i+1}u_{i+1} = 6(\Delta_{i+1} - \Delta_i), \qquad i = 1, 2, \ldots n-1, \qquad (8)$$

де позначено

$$\Delta_i = \frac{y_i - y_{i-1}}{h_i} = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}. \qquad (9)$$

Кількість невідомих $u_0, u_1, \ldots u_n$ становить $n+1$, в той час як число співвідношень типу (8) – лише $n-1$. Тому на ці параметри можна накласти довільно іще дві умови, з якихось додаткових міркувань, про що мова вже йшла вище.

Вважатимемо

$$u_0 = 0, \qquad u_n = 0. \qquad (10)$$

Цей тип граничних умов відповідає так званому «природному» або «натуральному» сплайну, що моделюється гнучкою лінійкою. Оскільки лінійка за межами інтервалу $[x_0, x_n]$ залишається недеформованою, то на його кінцях вона має нульову кривизну і, відповідно, нульову другу похідну.

Система рівнянь (8) відносно решти невідомих $u_1, \dots u_{n-1}$ в матричному вигляді запишеться:

$$\begin{pmatrix} 2(h_1+h_2) & h_2 & & & \dots & 0 \\ h_2 & 2(h_2+h_3) & h_3 & & & \dots \\ & h_3 & 2(h_3+h_4) & h_4 & & \\ & & & \dots & & \\ \dots & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \dots & & & h_{n-1} & 2(h_{n-1}+h_n) \end{pmatrix} \times$$

$$\times \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{n-2} \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} 6(\Delta_2 - \Delta_1) \\ 6(\Delta_3 - \Delta_2) \\ 6(\Delta_4 - \Delta_3) \\ \dots \\ 6(\Delta_{n-1} - \Delta_{n-2}) \\ 6(\Delta_n - \Delta_{n-1}) \end{pmatrix} \qquad (11)$$

Для її розв'язання застосуємо метод Гауса. Оскільки матриця системи є тридіагональною, доцільно застосувати схему «*прогонки*», яка не потребує зберігання в оперативній пам'яті двомірних масивів, а вимагає лише двох тимчасових одномірних масивів для проміжних результатів.

На етапі прямого ходу при обнуленні кожного стовпчика доводиться обнуляти єдиний елемент матриці на першій під-діагоналі, тому система, після її приведення до трикутної форми, матиме вигляд:

$$\begin{pmatrix} \alpha_1 & h_2 & & & \dots & 0 \\ & \alpha_2 & h_3 & & & \dots \\ & & \alpha_3 & h_4 & & \\ & & & \dots & & \\ \dots & & & & \alpha_{n-2} & h_{n-1} \\ 0 & \dots & & & & \alpha_{n-1} \end{pmatrix} \times \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_{n-2} \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \\ \beta_{n-2} \\ \beta_{n-1} \end{pmatrix}.$$

Тут

$$\alpha_1 = 2(h_1 + h_2)\,, \qquad \beta_1 = 6\big(\Delta_2 - \Delta_1\big), \tag{12.1}$$

гаусові множники на кожному етапі становлять $\dfrac{h_i}{\alpha_{i-1}}$, тому

$$\left.\begin{aligned} \alpha_i &= 2(h_i + h_{i+1}) - \frac{h_i^{\,2}}{\alpha_{i-1}} \\ \beta_i &= 6\big(\Delta_{i+1} - \Delta_i\big) - \frac{h_i \beta_{i-1}}{\alpha_{i-1}} \end{aligned}\right\}, \qquad i = 2,\, 3,\, \ldots\, n-1. \tag{12.2}$$

Зворотній хід:

$$u_{n-1} = \frac{\beta_{n-1}}{\alpha_{n-1}}\,, \tag{13.1}$$

$$u_i = \frac{\beta_i - h_{i+1} u_{i+1}}{\alpha_i}\,, \qquad i = n-2,\, n-3,\, \ldots\, 1\,. \tag{13.2}$$

Таким чином, алгоритм сплайн-інтерполяції полягає в наступному.

(i) На основі масивів $\{x_i\}$ та $\{y_i\}$ обчислити масив $\{u_i\}$ за формулами (12), (13). (Цей пункт виконується лише один раз).

(ii) Задатися значенням x, для якого треба обчислити інтерполянт. Визначити номер відрізку i, для якого $x \in [x_{i-1}, x_i]$.

(iii) Розрахувати інтерполянт за формулою (6.1). Для інших значень x повернутися до п. (ii).

Додаткове завдання

6. Скористайтеся тим самим набором вузлів, який ви створили згідно з пп. 1–2 завдання. Запрограмуйте обчислення сплайн-інтерполяції. Складіть окрему процедуру `SPLINE` для розрахунку масиву коефіцієнтів другої похідної $\{u_i\}$ для сплайн-інтерполяції «природним» сплайном за формулами (12), (13). Основу програмного коду запозичте з наведеного фрагменту.
7. Виконайте пп. 3–5 для сплайн-інтерполяції, розраховуючи замість інтерполяційної ламаної $p(x)$ сплайн-інтерполянт $s(x)$ за формулою (6.1) (див. рис. 16.5). Зверніть увагу, що в прикладах на рис. 16.3 і 16.5 масштаби для представлення похибки інтерполяції відрізняються в 10 разів.

```
SUB SPLINE (n,X(1),Y(1),U(1))
'
' ------------------------------------------------------------
' Розрахунок масиву коефіцієнтів другої похідної
' для сплайн-інтерполяції "природним" сплайном.
'
' Вхідні параметри:
'    n    - n+1 = кількість точок даних,
'           точки нумеруються від 0 до n;
'    X[n] - вузли інтерполяції (нумерація вузлів 0...n),
'           впорядковані за величиною X[0]<X[1]<...<X[n];
'    Y[n] - значення даних у вузлах інтерполяції;
' Вихідні параметри:
'    U[n] - значення величини другої похідної у вузлах,
'           в "природному" сплайні U[0]=U[n]=0.
' ------------------------------------------------------------
'
' Декларування службових масивів для проміжних розрахунків
   DIM AA[n], BB[n]

   h1=X[1]-X[0]                              ' Метод прогонки.
   h2=X[2]-X[1]                              ' Ініціація циклу
   D1=(Y[1]-Y[0])/h1                         ' прямого ходу
   D2=(Y[2]-Y[1])/h2
   AA[1]=2*(h1+h2)
   BB[1]=6*(D2-D1)

   FOR i=2 TO n-1                            ' Прямий хід
      h1=X[i]-X[i-1]
      h2=X[i+1]-X[i]
      D1=(Y[i]-Y[i-1])/h1
      D2=(Y[i+1]-Y[i])/h2
      AA[i]=2*(h1+h2)-h1*h1/AA[i-1]
      BB[i]=6*(D2-D1)-h1*BB[i-1]/AA[i-1]
   NEXT i

   U[n]=0                                    ' Ініціація циклу
   U[n-1]=BB[n-1]/AA[n-1]                    ' зворотнього ходу

   FOR i=n-2 TO 1 STEP -1                    ' Зворотній хід
      h2=X[i+1]-X[i]
      U[i] = (BB[i]-h2*U[i+1])/AA[i]
   NEXT i
   U[0] = 0

END SUB
```

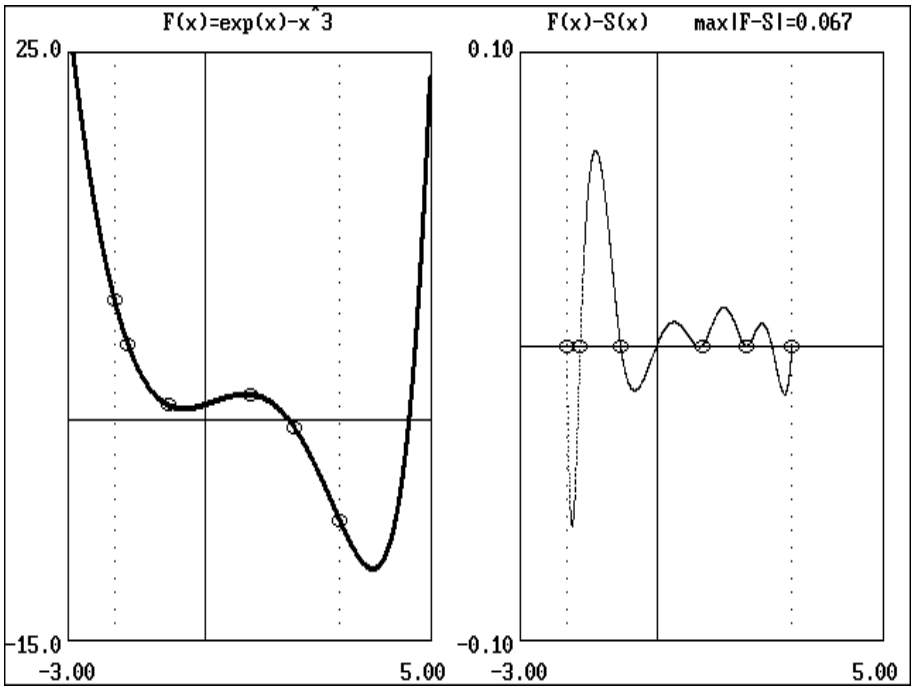


Рис. 16.5. Наближення функції $f(x) = e^x - x^3$ на інтервалі [–2, 3] сплайн-інтерполяцією «природним» сплайном $s(x)$ на n=5 відрізках (6 вузлах) $x_i = -2.0,\ -1.7,\ -0.8,\ +1.0,\ +2.0,\ +3.0$.

Ліворуч: $f(x)$ та $s(x)$, що зливаються у вибраному масштабі.
Праворуч: різниця $f(x) - s(x)$ в укрупненому масштабі по осі y.

8. Поясніть закономірності, що спостерігаються. Отримані результати зручно аналізувати, якщо вони представлені у вигляді таблиці на зразок

n	Δ			
	кусково-лінійна інтерполяція		сплайн-інтерполяція	
	рівновідділені вузли	вузли при рівномірному наближенні	рівновідділені вузли	вузли при рівномірному наближенні
5	…	…	…	…
15	…		…	
50	…		…	
150	…		…	
500	…		…	

Контрольні запитання

1. Чим відрізняється поліноміальна інтерполяція від кусово-поліноміальної? Які характерні риси сплайн-інтерполяції?
2. Як буде виглядати система рівнянь (11) для визначення $\{u_i\}$, якщо додаткові граничні умови (10) накладають на значення u_0 та u_n – *других* похідних в точках x_0 та x_n на кінцях інтервалу інтерполяції – фіксовані, але *не нульові* значення?
3. Як буде виглядати система рівнянь для визначення $\{u_i\}$, якщо дві додаткові граничні умови визначатимуть значення *перших* похідних на кінцях інтервалу інтерполяції? Інакше кажучи, замість (10) формулюються граничні умови $s_1'(x_0) = r_0$, $s_n'(x_n) = r_n$, де r_0 та r_n – певні числа, а вирази для $s_1'(x_0)$, $s_n'(x_n)$ даються формулами (7.2), (7.1). (Такого роду сплайн називають «фундаментальним».)
4. Одним з популярних способів накладання граничних умов є умова типу «сплайн, що завершується параболою». В цьому випадку перша та остання ланки сплайну представлюються квадратичними, а не кубічними параболами. Іноді це забезпечує більшу точність, аніж «природні» граничні умови. З огляду на те, що *третя* похідна на першій і останній ланках дорівнюватиме нулю $s_1''' = s_n''' = 0$ (див. 6.4), як буде виглядати система рівнянь для визначення $\{u_i\}$ в цьому випадку?
5. Як буде виглядати система рівнянь для визначення $\{u_i\}$, якщо дві граничні умови визначають *ненульові* значення *третіх* похідних на кінцях відрізка інтерполяції? Тобто граничні умови замість (10) матимуть вигляд $s_1'''(x_0) = w_0$, $s_n'''(x_n) = w_n$, де w_0 та w_n – певні числа (див. (6.4).)
6. Іноді при накладанні граничних умов на треті похідні w_0 та w_n в точках x_0 та x_n значення w_0 розраховується як третя похідна полінома 3-го степеня, що проходить через 4 перші вузли інтерполяції (x_i, y_i) при $i = 0, 1, 2, 3$, а w_n – через 4 останні при $i = n–3, n–2, n–1, n$. Розрахуйте, якими мають бути в цьому випадку значення w_0 та w_n. (Підказка: скористуйтеся інтерполяційним поліномом Лагранжа.)
7. Нехай точки даних (x_0, y_0), (x_1, y_1), … (x_n, y_n) представляють періодичну функцію $f(x) = f(x+\tau)$, довжина самого періоду становить $\tau = |x_n – x_0|$, а $y_n = y_0$. Які дві додаткові умови потрібно накласти замість (10) для побудови сплайн-інтерполянта?

8. *Екстраполяцією* називають такий тип апроксимації, при якому функція апроксимується *поза* заданим інтервалом, а не *між* заданими значеннями. Гнучка лінійка, форма якої описується «природним» сплайном, за межами інтервалу інтерполяції є прямолінійною. Ці прямі лінії можна використовувати для екстраполяції. Запишіть їх рівняння лівіше вузла x_0 та правіше вузла x_n. (Підказка: скористуйтеся формулами (7).)

9. Який вигляд матимуть рівняння екстраполянтів лівіше вузла x_0 та правіше вузла x_n. якщо для них використовувати поліноми 3-го степеню, що стикуються з першою та останньою ланкою сплайна так, що в точках x_0 та x_n залишаються неперервними *три* похідних?

10. Як, знаючи значення аргументу x, визначити номер інтервалу i, який його містить, $x \in [x_{i-1}, x_i]$? Прямий перебір потребуватиме в середньому $n/2$ перевірок, в той час як можна запропонувати алгоритм, що потребуватиме приблизно лише $\log_2 n$ перевірок. (Підказка: згадайте алгоритм бісекції для розв'язку рівнянь з одним невідомим).

11. Нехай задані $n+1$ точок даних $(x_i, y_i = f(x_i))$, $i = 0, 1, \ldots n$, і потрібно знайти точки, де $f(x) = 0$. Поясніть, чому не є хорошою ідея проінтерполювати дані поліномом Лагранжа n-го степеня $L_n(x)$, після чого розв'язати рівняння $L'(x) = 0$? Опишіть ефективний алгоритм розв'язання цієї задачі на основі використання кубічних інтерполяційних сплайнів.

Лабораторна робота № 17.

Апроксимація функціональних залежностей методом найменших квадратів

Мета роботи: застосування методу найменших квадратів для побудови поліноміального середньоквадратичного наближення функції.

Що зробити: побудувати наближення функції $f(x)$ лінійним поліномом за методом найменших квадратів, використовуючи набір із n псевдовипадкових «експериментальних» точок (x_i, y_i) ($i = 1, 2, \ldots n$), що розташовані на кривій $f(x)$ та імітують серію вимірювань. Дослідити поведінку коефіцієнтів МНК-полінома та средньоквадратичну похибку наближення σ в залежності від числа точок n. Провести аналогічні дослідження, додавши до «експериментальних» точок випадкові похибки. Додатково – побудувати МНК-наближення поліномами більш високих степенів та дослідити залежність σ від кількості точок n та степені МНК-полінома m.

Стислі теоретичні відомості

Нехай, вивчаючи невідому функціональну залежність між x та y, ми провели ряд вимірювань цих величин і в результаті серії експериментів отримали n точок (x_1, y_1), (x_2, y_2), … (x_n, y_n).

Задача полягає в тому, щоб знайти наближену залежність («емпіричну формулу») $y = \Phi(x)$, значення якої при $x = x_i$ ($i = 1, 2, \ldots n$) мало відрізняються від y_i. Застосування інтерполяції в данному випадку є недоцільним, оскільки значення y_i містять похибки вимірювань, і графік емпіричної залежності, взагалі кажучи, не проходить безпосередньо через точки (x_i, y_i), а певним чином згладжує випадкові похибки в даних.

Вважатимемо, що тип наближуючої формули вибрано, і вона може бути представлена у вигляді

$$y = \Phi(x; c_1, c_2, \ldots c_m) = \Phi(x; \mathbf{c}),$$

де Φ – відома функція, а $\mathbf{c} = \{c_1, c_2, \ldots c_m\}$ – невідомі постійні параметри, кількість яких менша (часто навіть значно менша) за кількість

екпериментальних точок ($m < n$), і які потрібно визначити таким чином, щоб емпірична формула давала в певному сенсі «найкраще наближення» до експериментальних даних.

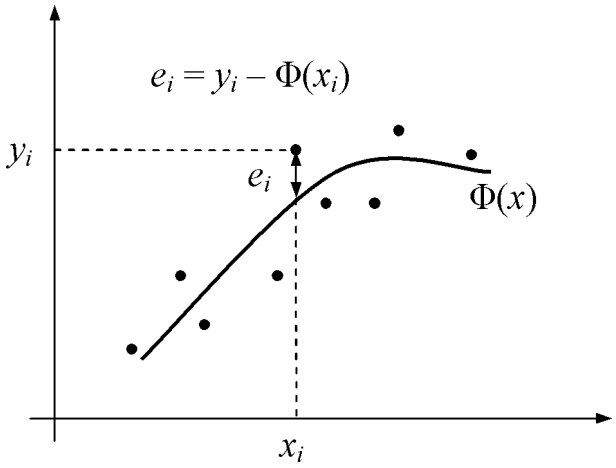


Рис. 17.1. Згладжування випадкових похибок в даних

Метод найменших квадратів (МНК) полягає в тому, що сукупність параметрів $\mathbf{c} = \{c_1, c_2, \ldots c_m\}$ визначається вимогою мінімізації суми квадратів похибок кожного експерименту:

$$S(\mathbf{c}) = \sum_{i=1}^{n} (e_i)^2 \to \min,$$

де

$$e_i = y_i - \Phi(x_i; \mathbf{c}), \quad i = 1, 2, \ldots n.$$

У випадку довільного вигляду функціональної залежності $\Phi(x; c_1, c_2, \ldots c_m)$ задача є задачею багатовимірної оптимізації і для свого розв'язання потребує складних чисельних алгоритмів.

Проте, якщо наближуюча функція МНК $\Phi(x)$ лінійна по параметрах $c_1, c_2, \ldots c_m$, тобто

$$\Phi(x) = c_1\varphi_1(x) + c_2\varphi_2(x) + \ldots + c_m\varphi_m(x) = \sum_{k=1}^{m} c_k\varphi_k(x), \qquad (1)$$

де $\varphi_k(x)$ – заздалегідь вибрані лінійно-незалежні базисні функції (зазвичай – поліноми, тригонометричні функції або експоненти), то задача

$$S(c_1, c_2, \ldots c_m) = \sum_{i=1}^{n} (c_1\varphi_1(x_i) + c_2\varphi_2(x_i) + \ldots + c_m\varphi_m(x_i) - y_i)^2 \to \min \qquad (2)$$

має аналітичний розв'язок. Прирівнюючи до нуля часткові похідні S, отримуємо так звану *систему нормальних рівнянь*:

$$\frac{\partial S}{\partial c_1} = 2\sum_{i=1}^{n} \varphi_1(x_i)\left(c_1\varphi_1(x_i) + c_2\varphi_2(x_i) + \ldots + c_m\varphi_m(x_i) - y_i\right) = 0$$

$$\frac{\partial S}{\partial c_2} = 2\sum_{i=1}^{n} \varphi_2(x_i)\left(- \text{ " } - \text{ " } - \text{ " } - \text{ " } - \text{ " } - \text{ " } - \right) = 0$$

……

$$\frac{\partial S}{\partial c_m} = 2\sum_{i=1}^{n} \varphi_m(x_i)\left(- \text{ " } - \text{ " } - \text{ " } - \text{ " } - \text{ " } - \text{ " } - \right) = 0$$

Це – система лінійних алгебраїчних рівнянь (СЛАР) відносно невідомих c_1, c_2, … c_m:

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{21} & a_{22} & \ldots & a_{2m} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{mm} \end{pmatrix} \times \begin{pmatrix} c_1 \\ c_2 \\ \ldots \\ c_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{pmatrix},$$

де

$$a_{k,l} = \sum_{i=1}^{n} \varphi_k(x_i)\varphi_l(x_i), \qquad b_k = \sum_{i=1}^{n} \varphi_k(x_i) y_i$$

Матриця системи нормальних рівнянь (*матриця Грама*) є невиродженою, відповідно її детермінант (*Граміан*) є відмінним від нуля, якщо тільки базисні функції $\varphi_1(x), \varphi_2(x), \ldots, \varphi_m(x)$ є лінійно-незалежними.

Зокрема, якщо за базисні вибрані степеневі функції $1, x, x^2, x^3, \ldots x^{m-1}$, і, відповідно, наближуюча функція МНК $\Phi(x)$ відшукується у формі полінома степеня m–1

$$\Phi_{m-1}(x) = c_1 + c_2 x + c_3 x^2 + \ldots + c_m x^{m-1}, \tag{3}$$

то система нормальних рівнянь, що визначає невідомі параметри c_1, c_2, … c_m, записується у вигляді

$$\begin{pmatrix} n & [x] & [x^2] & \ldots & [x^{m-1}] \\ [x] & [x^2] & [x^3] & \ldots & [x^m] \\ [x^2] & [x^3] & [x^4] & \ldots & [x^{m+1}] \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ [x^{m-1}] & [x^m] & [x^{m+1}] & \ldots & [x^{2m-2}] \end{pmatrix} \times \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \ldots \\ c_m \end{pmatrix} = \begin{pmatrix} [y] \\ [xy] \\ [x^2 y] \\ \ldots \\ [x^{m-1} y] \end{pmatrix}, \tag{4}$$

де квадратними дужками позначені суми по всіх експериментах (позначення Гауса): $[...] \equiv \sum_{i=1}^{n} ..._i$, тобто

$$[x^k] \equiv \sum_{i=1}^{n} (x_i)^k \; ; \qquad [x^k y] \equiv \sum_{i=1}^{n} (x_i)^k y_i \; .$$

(Тут враховано, що $[x^0] = x_1{}^0 + x_2{}^0 + \ldots x_n{}^0 = n$.)

Завдання

1. Для побудови поліноміального середньоквадратичного наближення візьміть згідно з вашим варіантом ту ж саму функцію $f(x)$, яку ви досліджували при виконанні лабораторної роботи № 3. Виберіть самостійно межі інтервалу апроксимації [a, b], на якому функція $f(x)$ неперервна і ніде не стає нескінченою (див. рис. 17.2).

2. Призначте на [a, b] певне число $n \sim 5 \ldots 20$ випадкових абсцис. Їх можна згенерувати за допомогою генератора псевдовипадкових чисел.

   Нехай r – випадкове число, рівномірно розподілене між 0 та 1. Більшість сучасних систем програмування мають подібний генератор псевдовипадкових чисел або як вбудовану функцію до транслятора, або в складі бібліотеки процедур для наукових розрахунків. Тоді можна вважати, що абсциси

   $$x_i = a + (b - a) \cdot r \, , \quad i = 1, 2 \ \ldots \ n$$

   рівномірно розподілені на [a, b]. Обрахуйте в цих точках значення функції $y_i = f(x_i)$. У такий спосіб ви отримаєте n точок (x_1, y_1), (x_2, y_2), … (x_n, y_n), що лежатимуть на кривій $f(x)$ та імітуватимуть серію вимірювань деякої фізичної величини.

3. Розрахуйте за методом найменших квадратів аналітичну залежність типу (3)

   $$\Phi_1(x) = c_1 + c_2 x \, , \tag{5}$$

   що наближує дані імітованого експерименту прямою лінією.

   Нормальні рівняння (4) міститимуть лише два невідомих параметри:

   $$\begin{pmatrix} n & [x] \\ [x] & [x^2] \end{pmatrix} \times \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} [y] \\ [xy] \end{pmatrix},$$

Обрахуйте коефіцієнти СЛАР, розв'яжіть її та визначте значення c_1 та c_2 у (5). Якщо після виконання лабораторної роботи № 8 ви маєте в своєму розпорядженні процедуру для розв'язання СЛАР, скористайтеся нею, інакше знайдіть розв'язок за правилом Крамера.

4. Розрахуйте середньоквадратичне ухилення σ отриманого наближення МНК, яке може служити мірою його якості:

$$\sigma = \sqrt{\frac{S}{n}},$$

де, як і раніше,

$$S = \sum_{i=1}^{n} \left(\Phi(x_i) - y_i\right)^2$$

5. Побудуйте графіки функції $f(x)$ та полінома МНК $\Phi_1(x)$ аналогічно тому, як ви це робили в лабораторній роботі № 3.

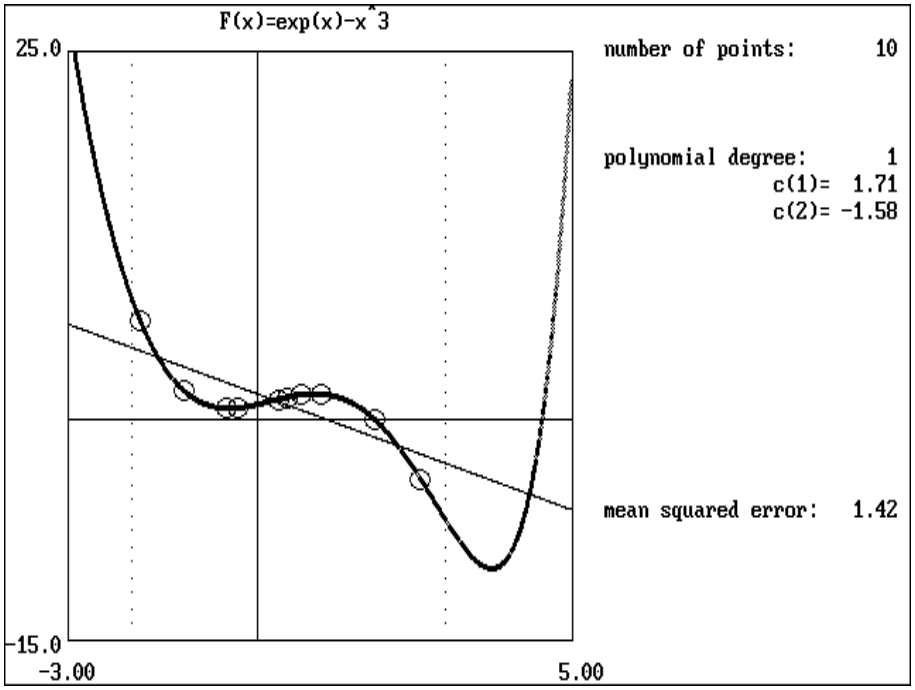


Рис. 17.2. МНК-наближення на інтервалі [–2, 3] набору даних $y_i = f(x_i)$ (кружки), де $f(x) = e^x - x^3$ (жирна лінія), лінійним поліномом $\Phi_1(x)$ з двома коефіцієнтами c_1 і c_2 (тонка лінія).

Праворуч зазначені кількість точок n, степінь полінома m, його коефіцієнти та средньоквадратична похибку наближення σ

6. Збільшуйте кількість точок n в приблизно 3, 10, 30 і 100 разів порівняно з початковим значенням. Чи прямує средньоквадратична похибка наближення σ до нуля, якщо кількість точок n необмежено зростає? Чи стабілізуються з ростом n значення коефіцієнтів c_1 і c_2 та похибки σ? Поясніть отримані результати.

7. Змоделюйте більш реальну ситуацію, коли вимірювання містять випадкові відхилення (шум). Цей шум можна представляти випадковою величиною, рівномірно розподіленою в діапазоні $[-W, +W]$, яку можна імітувати за допомогою того ж самого генератора псевдовипадкових чисел:

$$y_i = f(x_i) + (2r - 1)\cdot W.$$

   Задайте амплітуду W цього шуму на рівні приблизно в $3\sigma_0$, де σ_0 – типова величина средньоквадратичної похибки, що спостерігалася при виконанні пп. 4–6 завдання. Згенеруйте новий набір точок (x_1, y_1), (x_2, y_2), … (x_n, y_n), що лежать навколо кривої $f(x)$.

8. Розрахуйте для цього набору точок наближуючий МНК-поліном $\Phi_1(x)$ (див. рис. 17.3) і обрахуйте средньоквадратичні ухилення σ_W для даних, що містять шум. Порівняйте σ_W з величиною амплітуди шуму W. Поясніть результати.

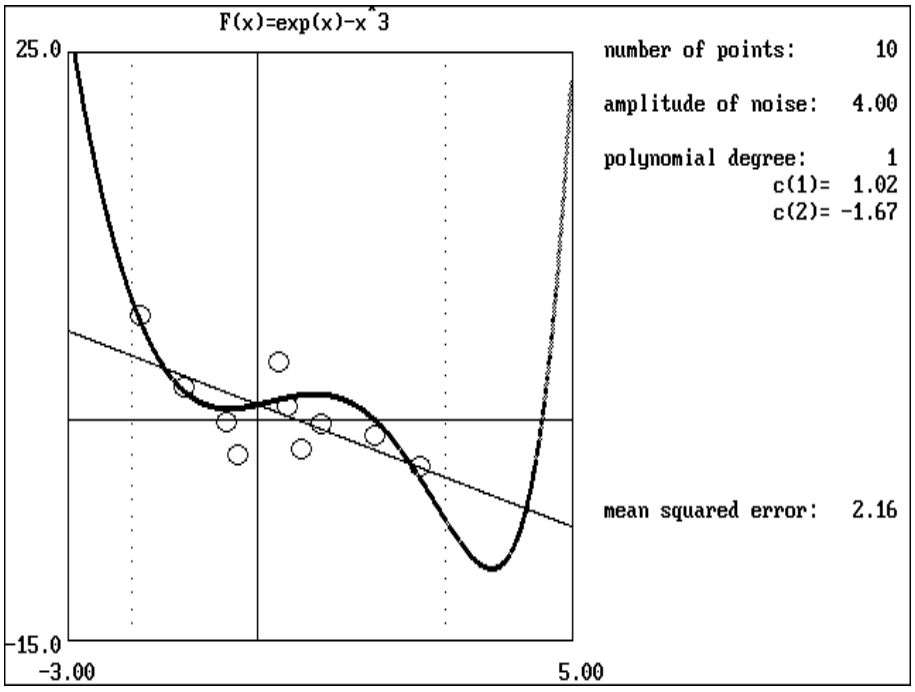


Рис. 17.3. Те ж саме, що на рис. 17.2,
але для набору даних $y_i = f(x_i)$ + шум з амплітудою 4.00

9. Так само, як в п. 6 завдання, збільшуйте кількість точок n і дослідіть поведінку значеннь коефіцієнтів c_1 і c_2 та средньоквадратичної похибка наближення σ_W. Отримані результати зручно аналізувати, якщо вони представлені у вигляді таблиці на зразок

n	c_1	c_2	σ	c_1	c_2	σ_W
	при $W = 0$			при $W = \ldots$		
10	…	…	…			
30						
100						
300						
…						

Додаткове завдання

10. Для виконання цього пункту завдання вам знадобиться процедура для розв'язння СЛАР, яку ви склали при виконанні лабораторної робти № 8. Використовуючи такий самий набір точок (x_1, y_1), (x_2, y_2), … (x_n, y_n), що й в п. 2 завдання, збільшуйте степінь m наближуючих МНК-поліномів $\Phi_m(x)$ та прослідкуйте за їх поведінкою (рис. 17.4). Дослідіть залежність середньоквадратичного ухилення σ від кількості точок n в наборі при різних степенях m наближуючих МНК-поліномів. Не забувайте про умову $m < n$! (Останній знак оклику означає не факторіал, а лише застереження бути уважним!)
11. Проведіть аналогічні дослідження за наявності шуму тієї ж самої амплітуди W, що і в п. 7 завдання (рис. 17.5) і проаналізуйте отримані результати. Це зручно робити, якщо вони представлені у вигляді таблиці на зразок

n	σ					σ_W				
	при $W = 0$					при $W = \ldots$				
	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = \ldots$	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = \ldots$
10	…	…	…							
30										
100										
300										
…										

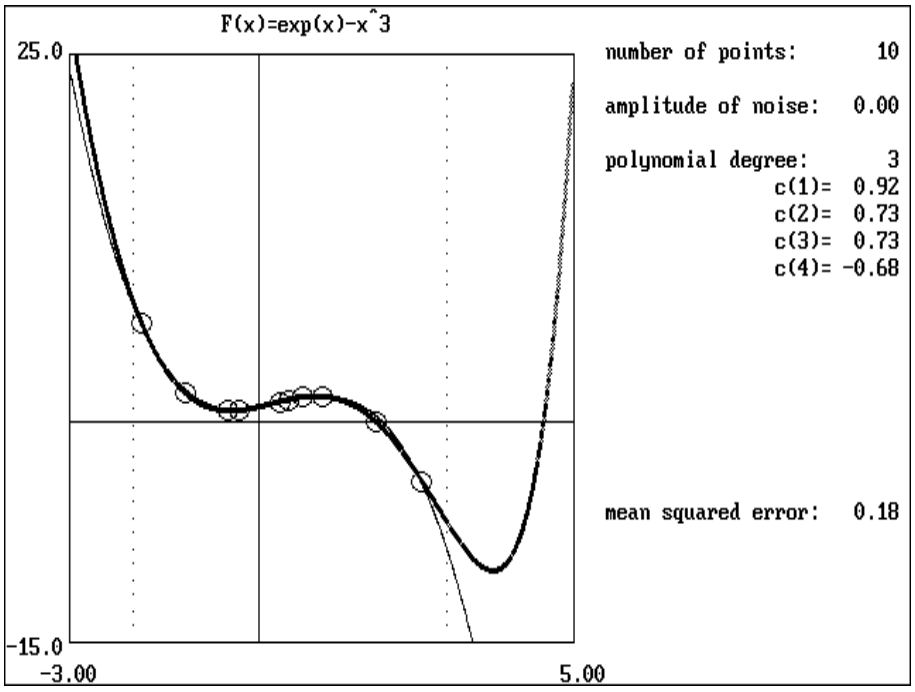


Рис. 17.4. Те ж саме, що на рис. 17.2,
але для МНК-наближення поліномом третього степеня $\Phi_3(x)$

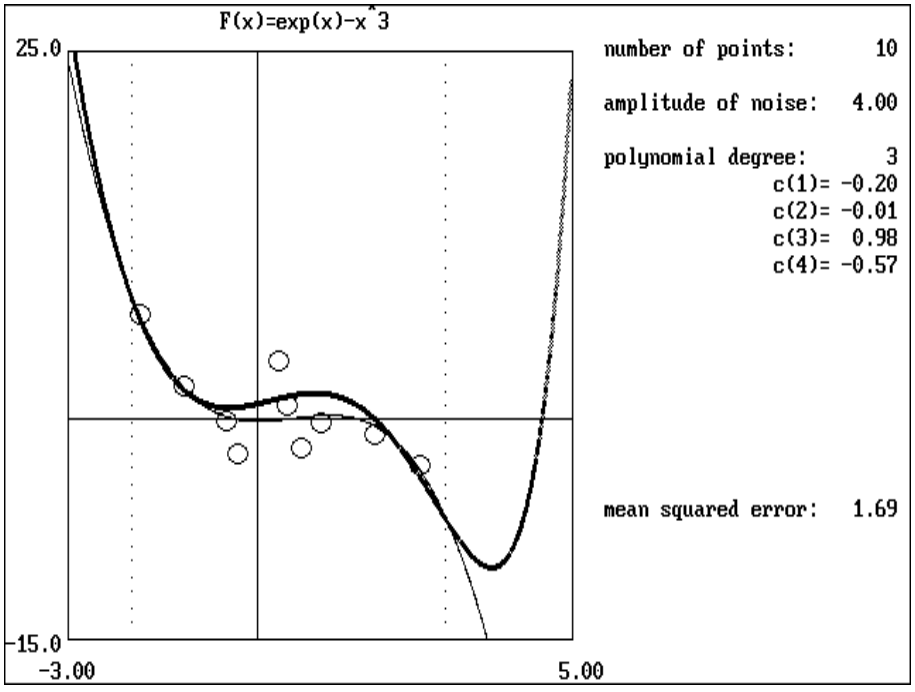


Рис. 17.5. Те ж саме, що на рис. 17.2,
але для набору даних $y_i = f(x_i)$ + шум з амплітудою 4.00
та МНК-наближення поліномом третього степеня $\Phi_3(x)$

Контрольні запитання

1. В чому полягає задача апроксимації за методом найменших квадратів? В чому її принципова різниця із задачею інтерполяції?
2. Які види наближуючих функцій зазвичай застосовуються в МНК?
3. Яким чином можна оцінювати якість наближення МНК?
4. Доведіть, що детермінант матриці Грама дорівнює нулю, якщо функції $\varphi_1(x), \varphi_2(x), ...\varphi_m(x)$ (1) не є лінійно-незалежними.
5. Для аналізу динаміки населення України за 1990-2013 рр. по наведених нижче даних будується поліном МНК 6-го степеня виду (3) $y = \Phi(x)$.

Чисельність населення України на початок року
(джерело: http://demoscope.ru/weekly/ssp/sng_pop.xls)

Рік	Населення
1990	51 556 500
1991	51 623 500
1992	51 708 200
1993	51 870 400
1994	51 715 400
1995	51 300 400
1996	50 874 100
1997	50 400 000

Рік	Населення
1998	49 973 500
1999	49 544 800
2000	49 115 000
2001	48 663 600
2002	48 240 900
2003	47 823 100
2004	47 442 100
2005	47 100 462

Рік	Населення
2006	46 749 170
2007	46 465 691
2008	46 192 309
2009	45 963 359
2010	45 782 592
2011	45 598 179
2012	45 453 283
2013	45 553 000

Обговоріть з точки зору обумовленості системи нормальних рівнянь (4) доцільність вибору змінних:

x = Рік ,
x = Рік / 1000 ,
x = Рік – 2000 ,
x = (Рік – 2000) /10 ;

y = Населення ,
y = Населення / 1 000 000 ,
y = Населення – 50 000 000 ,
y = (Населення – 50 000 000) / 1 000 000 .

Лабораторна робота № 18.

Чисельне інтегрування. Формули прямокутників, трапецій, Сімпсона

Мета роботи: вивчення алгоритмів Ньютона-Котеса чисельного інтегрування функції однієї змінної: квадратурних формул прямокутників, трапецій, Сімпсона та дослідження поведінки їх похибок.

Що зробити: обчислити інтеграл аналітично і за допомогою складеної квадратурної формули при різних кількостях підінтервалів n. Впевнитися у взаємоузгодженості отриманих результатів. Порівняти розбіжності між аналітичним і наближеними результатами при різних n і визначити порядок точності квадратурної формули.

Стислі теоретичні відомості

Квадратурні формули для наближеного обчислення інтегралу $I = \int_a^b f(x)dx$ отримують шляхом заміни підінтегральної функції $f(x)$ апроксимуючою функцією, яка легко інтегрується аналітично. Алгоритми Ньютона-Котеса передбачають апроксимацію функції $f(x)$ інтерполяційним поліномом $P(x)$ з вузлами інтерполяції на інтервалі $[a, b]$.

Формула прямокутників утворюється при використанні найпростішого інтерполяційного полінома нульового степеня: $P_0(x) = \text{const} = f(c)$, де єдиний вузол інтерполяції c є серединою інтервалу $[a, b]$ (рис. 18.1).

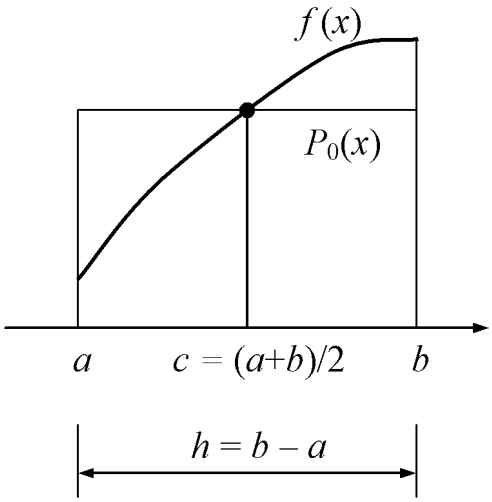


Рис. 18.1. Формула прямокутників

Шуканий інтеграл, чисельно рівний площі криволінійної трапеції під кривою $f(x)$, замінюється площею прямокутника під $P_0(x)$: $I \approx I_{\text{П}} = \int_a^b P_0(x)dx$. Вочевидь,

$$I_{\text{П}} = h \cdot f_c, \tag{1}$$

де $h = b - a$, $f_c = f(c)$.

Інтерполяційний поліном $P_1(x)$ першого степеню з вузлами інтерполяції a та b призводить до *формули трапецій*: $I \approx I_{\text{T}} = \int_a^b P_1(x)dx$.

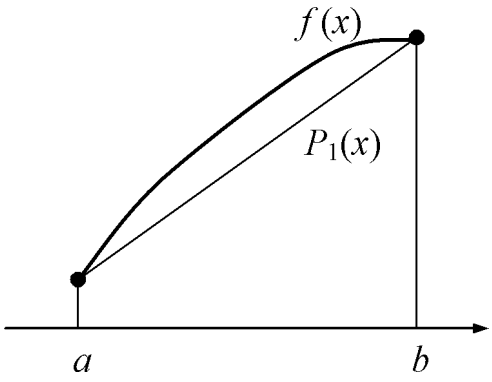


Рис. 18.2. Формула трапецій

Якщо записати інтерполяційний поліном у формі Лагранжа

$$P_1(x) = f_a \frac{x-b}{a-b} + f_b \frac{x-a}{b-a} \ , \tag{2}$$

де $f_a = f(a)$, $f_b = f(b)$, то безпосереднє інтегрування в межах від a до b приводить до формули:

$$I_{\text{T}} = h\frac{f_a + f_b}{2} \ . \tag{3}$$

Цей результат є досить очевидним також з геометричних міркувань (див. рис. 18.2), бо представляє собою площу відповідної прямолінійної трапеції.

Формула Сімпсона (парабол) утворюється при заміні підінтегральної функції $f(x)$ інтерполяційним поліномом $P_2(x)$ з вузлами в точках a, c, b : $I \approx I_{\text{C}} = \int_a^b P_2(x)dx$.

Записуючи $P_2(x)$ у формі Лагранжа аналогічно (2) і інтегруючи у межах від від a до b, знаходимо площу параболічної трапеції під кривою $P_2(x)$:

$$I_{\text{C}} = h\frac{f_a + 4f_c + f_b}{6} \ . \tag{4}$$

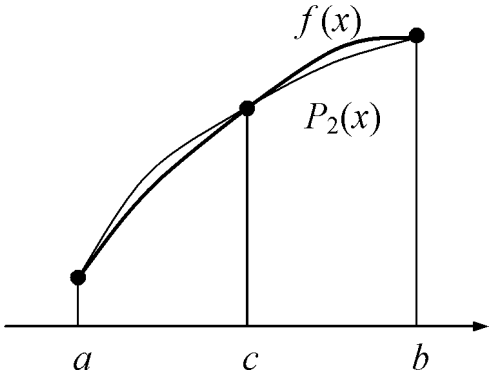


Рис. 18.3. Формула Сімпсона (парабол)

Результати, отримані за допомогою наближених квадратурних формул прямокутників, трапецій або Сімпсона, відрізняються від точного значення інтегралу I деякою похибкою. Можна показати, що ці похибки наближено дорівнюють

$$I - I_{\text{П}} \approx \frac{f_c'' h^3}{24} \tag{5.1}$$

$$I - I_{\text{Т}} \approx -\frac{f_c'' h^3}{12} \tag{5.2}$$

$$I - I_{\text{С}} \approx -\frac{f_c^{\text{IV}} h^5}{2880} \ . \tag{5.3}$$

Якщо інтервал інтегрування завеликий, то точності квадратурних формул (1), (3) або (4) недостатньо. За цих умов $f(x)$ не апроксимують поліномами високих степенів, а використовують кусочно-поліноміальну інтерполяцію, розбиваючи інтервал $[a, b]$ на більш дрібні підінтервали. Найпростіші формули виходять, якщо всі підінтервали мають однакову довжину $h = (b - a)/n$. На відміну від (1), (3), (4), які називають *простими квадратурними формулами,* формули з розбиттям інтервалу інтегрування на n рівних підінтервалів називають *складеними квадратурними формулами.*

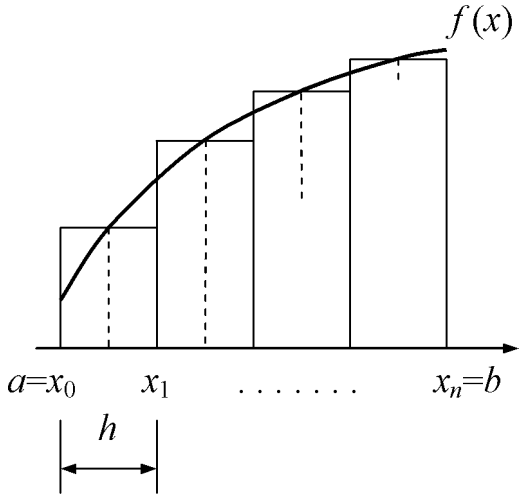


Рис. 18.4. Складена формула прямокутників

Складена формула прямокутників записується:

$$I_{\text{П}} = h\sum_{i=1}^{n} f\left(a + (i - ½)h\right) . \qquad (6)$$

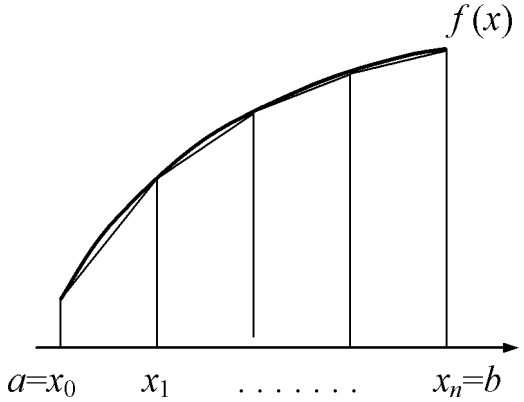


Рис. 18.5. Складена формула трапецій

Складена формула трапецій (скорочено позначаємо $f(x_i) = f_i$) має вид:

$$I_{\text{Т}} = h\frac{f_0 + f_1}{2} + h\frac{f_1 + f_2}{2} + \ldots + h\frac{f_{n-1} + f_n}{2} =$$

$$= h\left[\frac{f_0}{2} + f_1 + f_2 + \ldots + f_{n-1} + \frac{f_n}{2}\right]. \qquad (7)$$

Для побудови *складеної формули Сімпсона* інтервал розбивають на парне число підінтервалів довжиною h і застосовують просту формулу Сімпсона до кожної пари з них:

$$I_{\text{C}} = 2h\frac{f_0 + 4f_1 + f_2}{6} + 2h\frac{f_2 + 4f_3 + f_4}{6} + \ldots + 2h\frac{f_{n-2} + 4f_{n-1} + f_n}{6} =$$

$$= \frac{h}{3}\left[f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \ldots + 4f_{n-3} + 2f_{n-2} + 4f_{n-1} + f_n\right]. \quad (8)$$

Похибка складеної квадратурної формули приблизно в n разів (для формули Сімпсона – в $n/2$ разів) більша, ніж похибка відповідної простої формули:

$$I - I_{\text{П}} \approx \frac{f''}{24}\frac{(b-a)^3}{n^2} \quad (9.1)$$

$$I - I_{\text{T}} \approx -\frac{f''}{12}\frac{(b-a)^3}{n^2} \quad (9.2)$$

$$I - I_{\text{C}} \approx -\frac{f^{\text{IV}}}{180}\frac{(b-a)^5}{n^4}\ , \quad (9.3)$$

де f'' або f^{IV} є деякими середньозваженими значеннями відповідної похідної на інтервалі $[a, b]$, або, що те ж саме, є значенням похідної в деякій точці з цього інтервалу.

Таким чином, якщо число підінтервалів збільшити (а крок h відповідно зменшити) в деяке число k разів, то похибка результату, отриманого за складеною квадратурною формулою, зменшиться в k^p разів, де p=2 для формул прямокутників або трапецій і p=4 для формули Сімпсона. Показник p називають *порядком точності* формули.

Поряд з похибкою дискретизації, пропорційної $1/n^p$, в обчисленнях також присутня похибка округлення при розрахунку кожного із n значень f_i, яка в сумарному результаті може досягати величини порядку $n\,I\,\varepsilon_{маш}$.

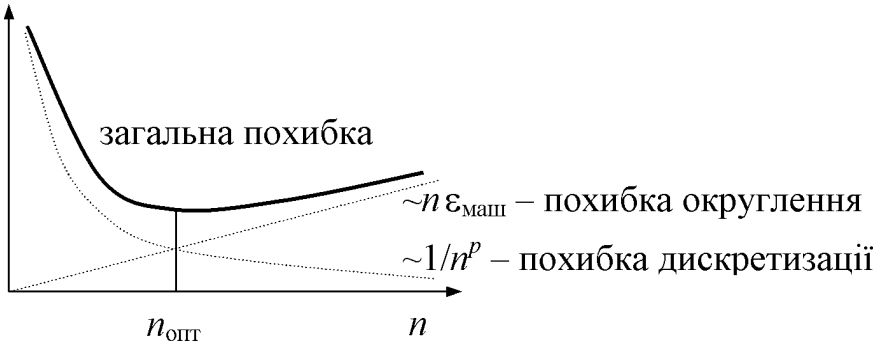


Рис. 18.6. Внесок похибок дискретизації і округлення в загальну похибку обчислень в залежності від числа підінтервалів

Сумарна похибка становить за порядком величини $\frac{1}{n^p} + n\varepsilon_{маш}$. Її мінімум досягається, коли обидва доданки приблизно рівні, тобто оптимальним з точки зору точності обчислень є число підінтервалів, приблизно рівне

$$n_{опт} \sim \frac{1}{\sqrt[p+1]{\varepsilon_{маш}}} \ . \qquad (10)$$

Наприклад, при $\varepsilon_{маш} = 10^{-8}$, для методів прямокутників або трапецій $n_{опт} \sim 500$, а для методу Сімпсона $n_{опт} \sim 40$. Звичайно, ці цифри є лише орієнтовними і справедливими лише за порядком величини, оскільки залежать від конкретного виду підінтегральної функції і меж інтегрування.

Завдання

1. За підінтегральну візьміть згідно з вашим варіантом ту ж саму функцію $f(x)$, яку ви досліджували при виконанні лабораторної роботи № 3. Виберіть самостійно межі інтегрування [a, b] і отримайте аналітичний вираз для $I = \int_a^b f(x)dx$. Можете скористатися таблицями невизначених інтегралів.
2. Запрограмуйте обчислення I за аналітичним виразом. В подальшому ви будете використовувати цю величину як еталонну, тож потурбуйтесь, щоб при обчисленнях марно не загубити значущі цифри.
3. Складіть програму для обчислення наближеного значення цього ж інтегралу за допомогою однієї зі складених квадратурних формул – прямокутників, трапецій або Сімпсона згідно з вашим варіантом і отримайте результат $I_{\text{наближ}}$ (тобто $I_{\text{П}}$, $I_{\text{Т}}$, або $I_{\text{С}}$). Порівняйте його з точним значенням I. Обчисліть похибку квадратурної формули
$$e = I - I_{\text{наближ}}.$$
4. Дослідіть залежність e від числа підінтервалів n. Задаючи n = 2, 4, 10, 20, 40, 100, ... складіть відповідну таблицю. Зауважте, що при дуже великих значеннях n накопичення інтегральної суми може вимагати значного часу роботи процесора, тож не виключено, що максимальна величина n буде обмежуватися швидкістю вашого комп'ютера. За даними таблиці побудуйте графік $e(n)$. Використовуйте логарифмічний масштаб по обох осях ($\lg e$ та $\lg n$). Поясніть характер отриманої залежності, визначте по ній порядок точності квадратурної формули і порівняйте його з теоретичним значенням. Оцініть оптимальне значення кількості підінтервалів квадратурної формули. Чи збігається воно із значенням, розрахованим за формулою (10) виходячи з машинного епсілон?
5. Виконайте дослідження п. 4, проводячи всі обчислення з подвійною точністю. Поясніть результати.

Варіанти для самостійної роботи

Варіанти 1, 2, 11, 12, 13, 14, 23, 24: формула прямокутників.

Варіанти 3, 4, 9, 10, 15, 16, 21, 22: формула трапецій.

Варіанти 5, 6, 7, 8, 17, 18, 19, 20: формула Сімпсона (парабол).

Контрольні запитання

1. Проінтегруйте $P_1(x)$ (див. (2)) в межах від a до b і самостійно отримайте формулу трапецій (3).
2. Виведіть самостійно формулу Сімпсона (4).
3. Доведіть формули (5) для похибок дискретизації квадратурних формул. Представте функцію $f(x)$ у вигляді ряду Тейлора навколо точки c (буде достатнім утримати члени до $f_c^{\mathrm{IV}}(x-c)^4$ включно) і виразіть за допомогою цього ряду f_a і f_b, а також – шляхом почленного інтегрування ряду – точне значення інтегралу I.
4. Доведіть формули (9) для похибок дискретизації складених квадратурних формул.
5. Нехай підінтегральна функція – поліном. Який має бути його степінь, щоб результат інтегрування, отриманий за допомогою квадратурної формули прямокутників, мав нульову похибку дискретизації? А за допомогою формули трапецій або Сімпсона?
6. Чи відрізняється відповідь на попереднє питання в залежності від того, використовується проста чи складена квадратурна формула?
7. Як зміниться похибка дискретизації при обчисленні інтегралу, якщо замість формули прямокутників з n=10 використати формулу трапецій з n=10? А формулу трапецій з n=20?
8. Вирази (9) мало придатні для практичної оцінки похибки дискретизації, оскільки вони містять похідні підінтегральної функції, обчислення яких може бути занадто обтяжливим. Деяке уявлення про похибку обчислень квадратурної формули може дати різниця між двома розрахунками інтегралу $I_{(1)}$ та $I_{(2)}$ із різними числами підінтервалів n_1 та n_2.

   Зважте на те, що $I \approx I_{\text{квадр}} + \alpha/n^p$, де α – деякий коефіцієнт, майже незалежний від n, і доведіть, що більш точна оцінка значення інтегралу дається виразом (так звана *екстраполяція Річардсона*):

$$I \approx I_{(2)} + \frac{I_{(2)} - I_{(1)}}{\left(\frac{n_2}{n_1}\right)^p - 1} .$$

9. Виходячи з формули трапецій з 1 та 2 підінтервалами за допомогою екстраполяції Річардсона виведіть уточнену квадратурну формулу. Чи знайома вона вам?

Лабораторна робота № 19.

Кратні інтеграли. Чисельне інтегрування методом Монте-Карло

Мета роботи: обчислення кратних інтегралів методом Монте-Карло за допомогою генератора псевдовипадкових чисел. Вивчення поведінки похибки отриманого наближення в залежності від кількості використаних випадкових точок.

Що зробити: обрахувати аналітично точну площу плоскої фігури. Оцінити її площу також методом Монте-Карло, генеруючи на площині достатню кількість n випадкових точок і аналізуючи, скільки з них потрапили всередину фігури. Додатково – впевнитись, що відносна похибка результату з ростом n має тенденцію зменшуватися пропорційно $1/\sqrt{n}$.

Стислі теоретичні відомості

Метод Монте-Карло є придатним для обчислення кратних інтегралів виду $A = \int\limits_{\boldsymbol{Q}} f(\mathbf{x}) dV_{\mathbf{x}}$ на області $\boldsymbol{Q}$ будь-якого числа вимірів. Під $\mathbf{x}$ розуміється сукупність змінних $\{x_1, x_2, \ldots\}$; $dV_{\mathbf{x}} = dx_1\, dx_2 \ldots$

Метод базується на тому факті, що середнє значення функції f в області $\boldsymbol{Q}$ дорівнює

$$f_{\text{сер}} = \frac{\int\limits_{\boldsymbol{Q}} f(\mathbf{x}) dV_{\mathbf{x}}}{V_{\boldsymbol{Q}}} ,$$

де $V_{\boldsymbol{Q}} = \int\limits_{\boldsymbol{Q}} dV_{\mathbf{x}}$ – об'єм області $\boldsymbol{Q}$.

Ідея методу полягає в тому, що генеруються n випадкових точок, рівномірно розподілених в області $\boldsymbol{Q}$, і обчислюється S_f – сума значень підінтегральної функції в цих точках. Тоді $f_{\text{сер}} \approx S_f / n$ і

$$A = \int\limits_{\boldsymbol{Q}} f(\mathbf{x}) dV_{\mathbf{x}} \approx \frac{S_f}{n} V_{\boldsymbol{Q}} .$$

Якщо область $\boldsymbol{Q}$ має складну форму, то складно і генерувати випадкові точки, що були б в ній розподілені рівномірно. В цьому випадку її обіймають областю $\boldsymbol{R}$ простої форми, і генерують точки в $\boldsymbol{R}$. Вважають, що $f(\mathbf{x}) = 0$, якщо $\mathbf{x} \notin \boldsymbol{Q}$. Тоді $\int_{\boldsymbol{Q}} = \int_{\boldsymbol{R}}$.

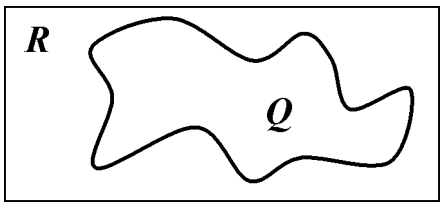


Рис. 19.1. Область інтегрування

Якщо область $\boldsymbol{R}$ прямокутна, то в ній випадкову точку $\mathbf{x}$ легко згенерувати за допомогою генератора рівномірно розподілених випадкових чисел. Для цього досить кожній координаті точки $\mathbf{x}$ присвоїти значення

$$x_i = x_i^{\min} + r \cdot (x_i^{\max} - x_i^{\min}),$$

де $x_i^{\max}$ та $x_i^{\min}$ – межі області вздовж i-ї координатної осі, а r – випадкове число, рівномірно розподілене між 0 та 1. Більшість сучасних систем програмування мають такий генератор псевдовипадкових чисел або як вбудовану функцію до транслятора, або в складі бібліотеки процедур для наукових розрахунків. Числа, які генеруються у такий спосіб, є результатом застосування спеціального алгоритму і в цьому сенсі вони, звичайно, не випадкові, а, навпаки, детерміновані. Їхня послідовність може бути відтворена, що дуже важливо при налаштуванні програми. Але ця послідовність «виглядає» як випадкова, задовольняючи ряду традиційних статистичних тестів.

Отже,

$$A = \int_{\boldsymbol{R}} f(\mathbf{x}) dV_{\mathbf{x}} \approx \frac{S_f}{n} V_{\boldsymbol{R}}, \qquad (1)$$

де

$$V_{\boldsymbol{R}} = \prod_i \left| x_i^{\max} - x_i^{\min} \right|.$$

Окремо можна виділити випадок, коли

$$f(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \boldsymbol{Q}; \\ 0, & \mathbf{x} \notin \boldsymbol{Q}. \end{cases}$$

Тоді S_f – число попадань випадкової точки в область $\boldsymbol{Q}$, S_f/n – доля попадань із загального числа випробувань, а сам інтеграл (1) дає оцінку розміру (об'єму) $V_{\boldsymbol{Q}}$ області $\boldsymbol{Q}$.

Інтуїтивно ясно, що із збільшенням кількості випробувань відносна похибка отриманого наближення буде зменшуватись. Методами математичної статистики можна показати, що очікуване середньоквадратичне відхилення результату приблизно дорівнює

$$\frac{\sigma_A}{A} \approx \frac{1}{\sqrt{n}} . \qquad (2)$$

Зазвичай фактична похибка, що спостерігається, за абсолютною величиною не перевищує потрійного середньоквадратичного відхилення.

Завдання

1. Для дослідження особливостей методу Монте-Карло візьміть двовимірну фігуру $\boldsymbol{Q}$ складної форми згідно з вашим варіантом. Розрахуйте аналітично значення її площі $V_{\boldsymbol{Q}}$. В подальшому ви будете використовувати цю величину як еталонну.
2. Задайте прямокутну область $\boldsymbol{R}$ $\{(x^{\min} \ldots x^{\max}), (y^{\min} \ldots y^{\max})\}$, що охоплює фігуру $\boldsymbol{Q}$. Визначте її площу $V_{\boldsymbol{R}} = (x^{\min} - x^{\max})(y^{\min} - y^{\max})$.
3. Почніть розробку вашої програми із фрагменту, що генерує випадкову точку з координатами (x, y) в прямокутній області $\boldsymbol{R}$ та перевіряє, чи належить ця точка області $\boldsymbol{Q}$.
4. Доповніть вашу програму графічним модулем. Відкрийте графічне вікно, намалюйте в ньому координатні осі, області $\boldsymbol{R}$ та $\boldsymbol{Q}$. Відмітьте також згенеровану випадкову точку (x, y). В залежності від того, чи потрапляє вона до області $\boldsymbol{Q}$, чи знаходиться поза нею, відмічайте її маркерами різних кольорів або розмірів. Це дозволить вам впевнитись, що перевірка умови $(x, y) \in \boldsymbol{Q}$ виконується програмою правильно.
5. Згенеруйте в $\boldsymbol{R}$ певну кількість n випадкових точок і підрахуйте, скільки з них (S_f) потрапило до області $\boldsymbol{Q}$. Розрахуйте наближене значення A площі фігури $\boldsymbol{Q}$ за формулою (1) та порівняйте її з точним значенням площі $V_{\boldsymbol{Q}}$, визначеному в п. 1.

6. Обчисліть відносну похибку отриманого значення

$$\delta_A = (A - V_{\boldsymbol{Q}}) / V_{\boldsymbol{Q}}. \qquad (3)$$

Дослідіть, як ця похибка залежить від числа випробувань n, задаючи $n = 10,\ 100,\ 1000,\ 10\,000,\ \ldots$ Бажано кожного разу застосовувати однакову послідовність псевдовипадкових чисел, для чого однаковим чином ініціювати генератор випадкових чисел на початку програми. Поясніть результати.

Додаткове завдання

7. Дослідіть більш детально, як змінюється похибка δ_A при послідовному збільшенні числа випробувань n від 1 до $n_{\max}$ (кількох десятків або сотень). Для цього після кожного випробування обчислюйте наближену оцінку A площі фігури $\boldsymbol{Q}$ за формулою (1) та її відносну похибку δ_A за формулою (3).

Залежність δ_A від n представте на окремому графіку (див. рис. 19.2). На цей же графік для кожного n нанесіть потрійне значення середньоквадратичного ухилення $\pm 3/\sqrt{n}$. Впевніться, що дві криві, що утворяться, окреслюють очікуваний «коридор», якого зазвичай дотримуватиметься поточна відносна похибка.

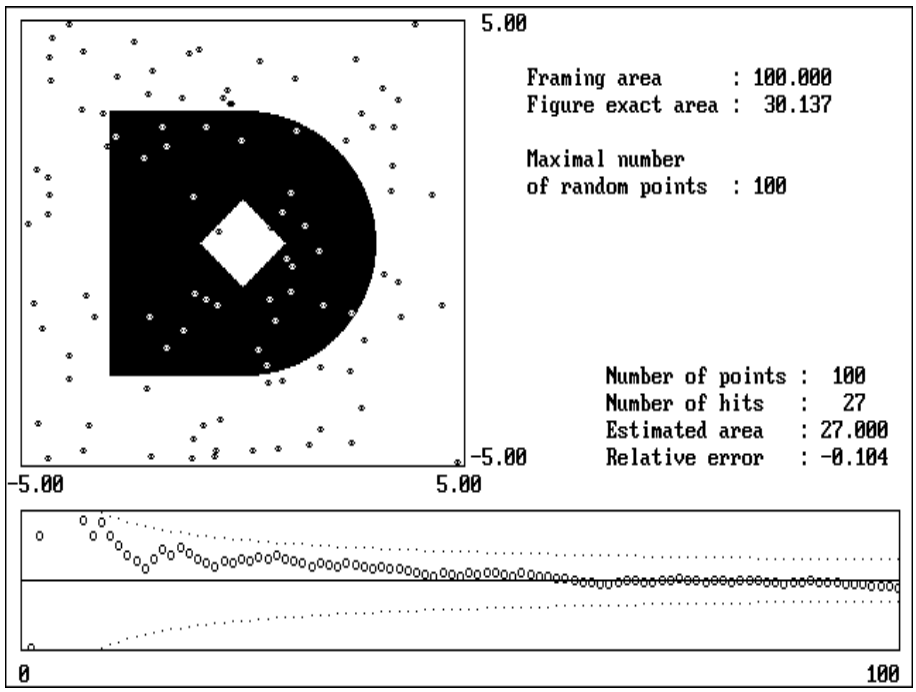


Рис. 19.2. Оцінка площі фігури методом Монте-Карло

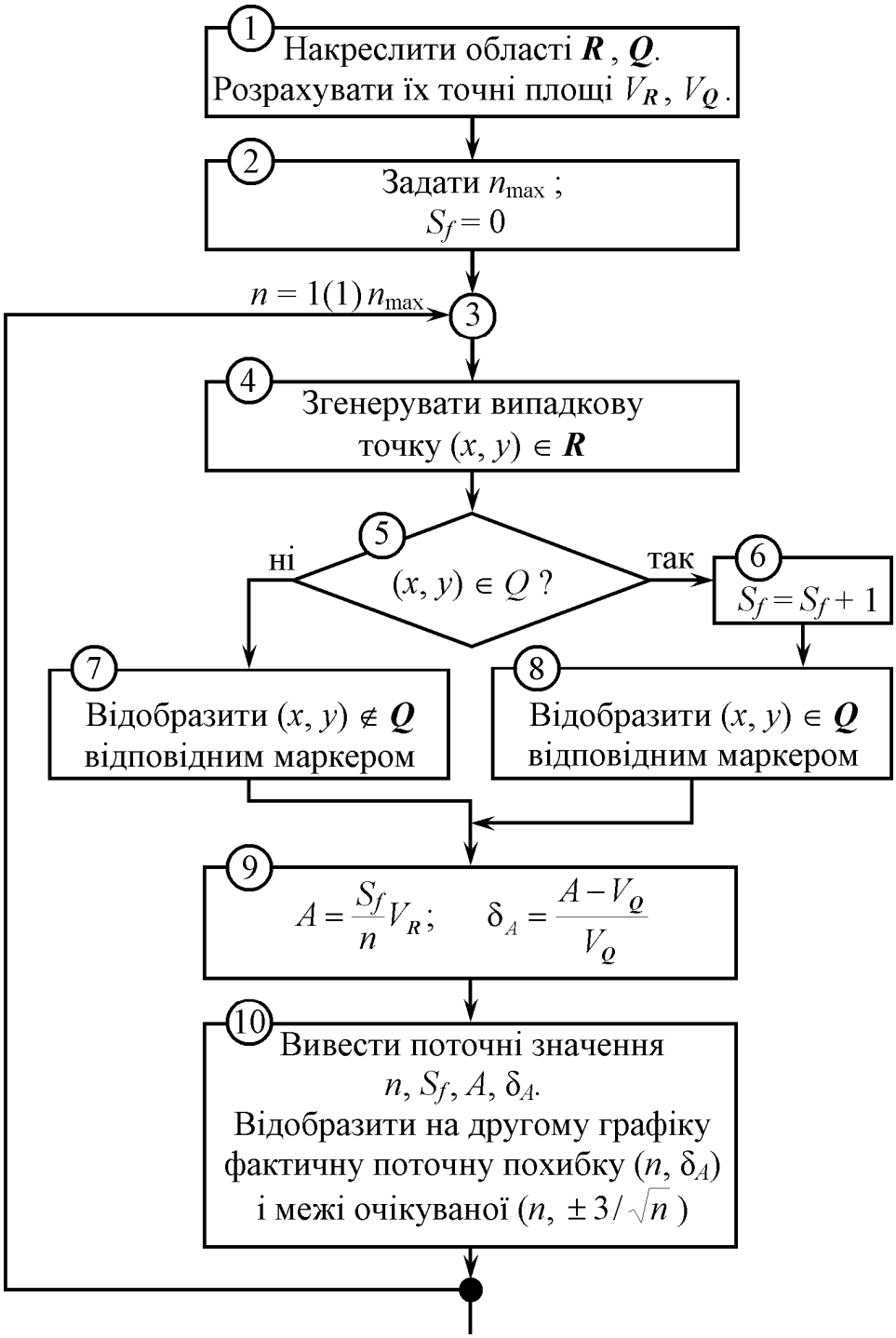
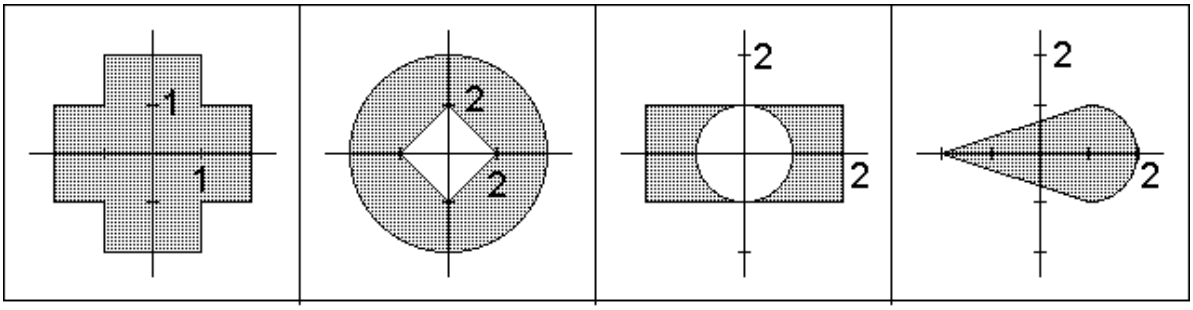


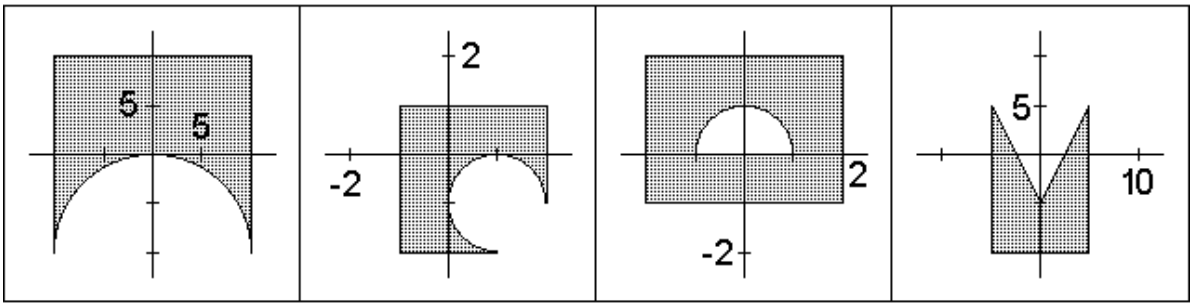
Рис. 19.3. Схема алгоритму для обчислення площі двовимірної фігури методом Монте-Карло

Варіанти для самостійної роботи

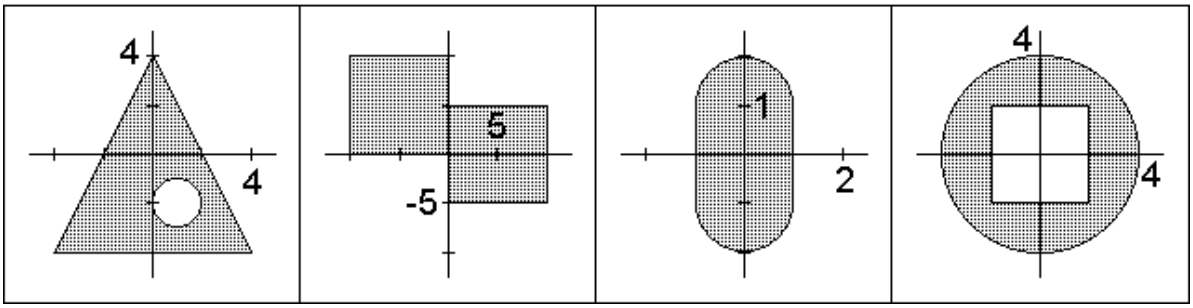
Варіанти 1, 24 ***Варіанти 2, 23*** ***Варіанти 3, 22*** ***Варіанти 4, 21***



Варіанти 5, 20 ***Варіанти 6, 19*** ***Варіанти 7, 18*** ***Варіанти 8, 17***



Варіанти 9, 16 ***Варіанти 10, 15*** ***Варіанти 11, 14*** ***Варіанти 12, 13***



Контрольні запитання

1. Що таке псевдовипадкова величина? Чим вона відрізняється від істинно випадкової?
2. Яким чином за допомогою генератора випадкових чисел, рівномірно розподілених між 0 та 1, можна згенерувати послідовність випадкових чисел, рівномірно розподілених в інших межах [a, b] ?
3. В чому полягає метод Монте-Карло для визначення площ?
4. Запропонуйте метод Монте-Карло для визначення об’ємів.
5. В яких випадках доцільно застосовувати метод Монте-Карло для визначення площ або об’ємів?
6. Скільки випробувань знадобиться, щоб очікувана похибка обчислення інтегралу методом Монте-Карло становила близько 1% ? А 0,1% ?

Лабораторна робота № 20.

Звичайні диференціальні рівняння. Задача Коші

Мета роботи: вивчення методу Рунге-Кутти розв'язку задачі Коші для звичайного диференціального рівняння.

Що зробити: отримати розв'язок звичайного диференціального рівняння на певному інтервалі аналітично і за допомогою однієї з квадратурних формул Рунге-Кутти. Впевнитися у взаємоузгодженості отриманих результатів. Побудувати графіки $y(t)$, $y'(t)$ та фазову траєкторію $y'(y)$. Порівняти розбіжності між аналітичним і наближеними результатами при різних кроках інтегрування і визначити порядок точності квадратурної формули.

Стислі теоретичні відомості

Зазвичай кажуть, що функція двох змінних $f(t,y)$ визначає певну поверхню над площиною (t,y). Кожна точка цієї поверхні знаходиться на висоті, чисельно рівній значенню функції f.

Якщо функція f є правою частиною диференціального рівняння

$$y'(t) = f(t, y), \tag{1}$$

то їй можна надати іншого геометричного змісту: оскільки $f(t, y)$ є похідною певної функції, тобто нахилом дотичної до її графіку, то кожній точці площини (t, y) можна співставити нахил, чисельно рівний значенню функції f в цій точці (див. приклад на рис. 20.1). Такий геометричний об'єкт називають *полем напрямків*.

Виключаючи ділянки області визначення функції $f(t, y)$, де вона має особливості (розриви, нескінченні похідні тощо), поле напрямків визначає онопараметричне сімейство інтегральних кривих $y(t)$, що різняться між собою значенням деякого параметра C. Таке сімейство називають *загальним розв'язком* диференціального рівняння (1) (див. приклад на рис. 20.2).

Щоб конкретизувати, про яку саме з інтегральних кривих $y(t)$ йде мова (*частковий розв'язок*), можна зазначити яку-небудь точку (t_0, y_0), через яку вона проходить.

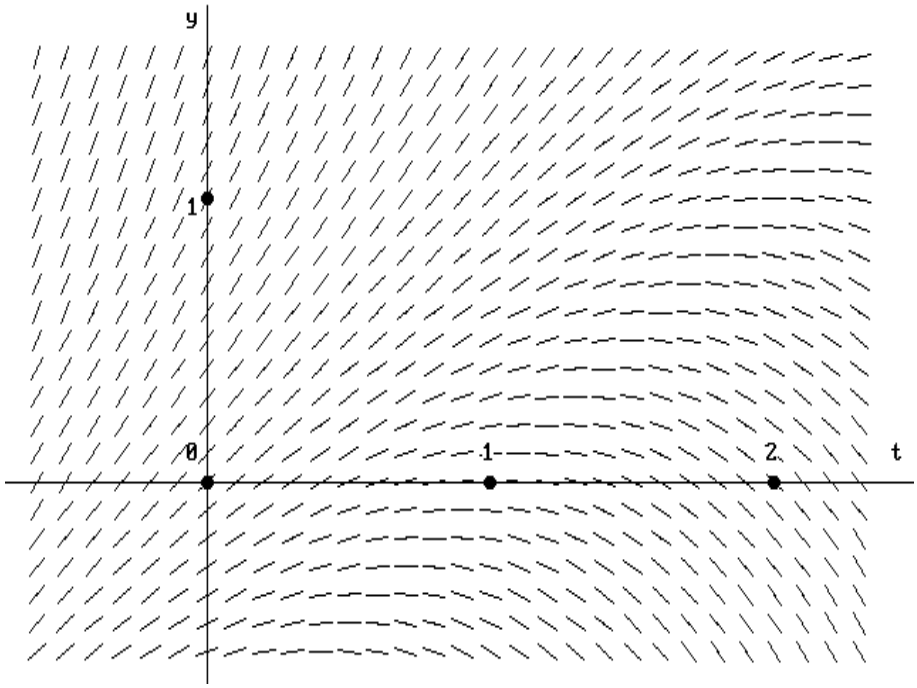


Рис. 20.1. Поле напрямків $y' = y - t + 1$

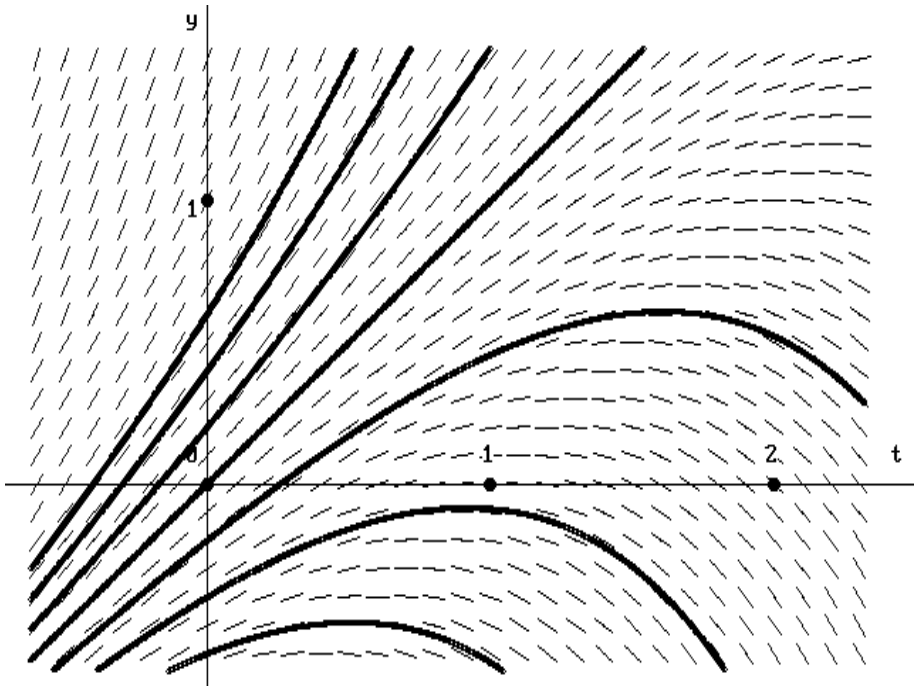


Рис. 20.2. Загальний розв'язок диференціального рівняння $y' = y - t + 1$.
Показані інтегральні криві $y(t) = t + Ce^t$
при C = +0.6, +0.4, +0.2, 0, –0.2, –0.4, –0.6 (згори вниз)

Постановка задачі у вигляді диференціального рівняння (1) з початковою умовою

$$y(t_0) = y_0 \tag{2}$$

називається *задачею Коші*.

Чисельний розв'язок задачі Коші відшукується в табличній формі:

t	y	$y'(t) = f(t, y)$
t_0	y_0	$y'_0 = f(t_0, y_0)$
t_1	y_1	$y'_1 = f(t_1, y_1)$
…	…	…
t_m	y_m	$y'_m = f(t_m, y_m)$
t_{m+1}	y_{m+1}	$y'_{m+1} = f(t_{m+1}, y_{m+1})$
…	…	…

Нульовий рядок таблиці – це початкова умова. Для розрахунку кожного наступного рядка використовуються дані попередніх рядків таблиці – одного (*однокрокові методи*) або декількох (*багатокрокові методи*). Вочевидь, для розрахунку першого рядка метод розрахунку має бути саме однокроковим.

Однокроковий метод передбачає послідовне просування по аргументу t на достатньо малі кроки h (не обов'язково однакової довжини). Використовуємо фрагмент ряду Тейлора, утримуючи похідні до p-го порядку (покажчик p називають *порядком точності* формули):

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2} y''(t) + \ldots + \frac{h^p}{p!} y^{(p)}(t) + O\left(h^{p+1}\right).$$

Похідні від $y(t)$ дорівнюють:

$$y'(t) = f(t, y)$$

$$y''(t) = \frac{df(t,y)}{dt} = \frac{f'_t dt + f'_y dy}{dt} = f'_t + f f'_y$$

$$y'''(t) = \frac{d\left(f'_t + f f'_y\right)}{dt} = \frac{f''_{tt} dt + f''_{ty} dy + (f'_t dt + f'_y dy) f'_y + f(f''_{ty} dt + f''_{yy} dy)}{dt} =$$

$$= f''_{tt} + 2 f f''_{ty} + f^2 f''_{yy} + f'_t f'_y + f (f'_y)^2$$

...

(всі часткові похідні від f обчислюються в точці (t, y), як і сама функція f).

Так, метод 1-го порядку (*метод Ейлера*) матиме вигляд:

$$y(t+h) = y(t) + hf(t,y) + O\left(h^2\right), \tag{3}$$

заснований на рядах Тейлора метод 2-го порядку –

$$y(t+h) = y(t) + hf(t,y) + \frac{h^2}{2}\left(f'_t + f f'_y\right) + O\left(h^3\right) \tag{4}$$

і т. д.

Незручність використання формули (4) та ще складніших формул більш високого порядку полягає в необхідності обчисленя часткових похідних функції f.

Методи Рунге-Кутти уникають обчислення часткових похідних, натомість на кожному кроці потребують обчислення функції $f(t, y)$ в кількох близьких точках.

Розглянемо обчислювальну схему з параметрами α, β, γ, ω:

$$k_1 = f(t,y)$$
$$k_2 = f(t + \alpha h, y + \beta h k_1)$$
$$y(t+h) = y(t) + \gamma h k_1 + \omega h k_2$$

Параметри α, β, γ, ω виберемо таким чином, щоб результат розрахунку $y(t+h)$ за цією схемою відрізнявся б від (4) на величину, не більшу ніж похибка самого результату (3), тобто $O\left(h^3\right)$.

Розкладаючи k_2 в ряд біля точки (t, y), маємо:

$$k_2 = f(t,y) + \alpha h f'_t + \beta h k_1 f'_y + O\left(h^2\right) = f + h\left(\alpha f'_t + \beta f f'_y\right) + O\left(h^2\right)$$

і

$$y(t+h) = y(t) + \gamma hf + \omega h\left[f + h\left(\alpha f'_t + \beta f f'_y\right) + O\left(h^2\right)\right] =$$
$$= y(t) + (\gamma + \omega)hf + \omega h^2\left(\alpha f'_t + \beta f f'_y\right) + O\left(h^3\right)$$

Співставлюючи коефіцієнти останньої формули з (4), отримуємо необхідні співвідношення між параметрами α, β, γ, ω:

$$\gamma + \omega = 1; \qquad \omega\alpha = \frac{1}{2}; \qquad \omega\beta = \frac{1}{2}.$$

Наявність трьох умов, накладених на чотри параметри, означає, що один з них, наприклад, ω, можна вибрати вільно.

Обчислювальна схема (точніше, сімейство схем) вигляду

$$\begin{aligned} k_1 &= f(t, y) \\ k_2 &= f\left(t + \frac{h}{2\omega}, y + \frac{hk_1}{2\omega}\right) \\ y(t+h) &= y(t) + (1-\omega)hk_1 + \omega\, hk_2 + O\left(h^3\right) \end{aligned} \tag{5}$$

де h – крок інтегрування по t, а $\omega \neq 0$ – вільний параметр, називається *методом Рунге-Кутти 2-го порядку*.

Ці формули, незалежно від вибраного ω, мають однаковий порядок точності – *локальна похибка* (тобто похибка на одному кроці інтегрування) пропорційна h^3, проте коефіцієнт при h^3, а отже і фактична величина похибки, буде (хоча і не дуже сильно) залежати від ω. Найпоширенішими є обчислювалні схеми з $\omega = 1$ (*модифікований метод Ейлера*) або з $\omega = ½$ (*метод Х'юна*), а найточнішою (з найменшим коефіцієнтом при h^3 в похибці) – з $\omega = ⅔$ (*метод Ральстона*).

Аналогічним чином можуть бути отримані обчислювальні схеми більш високих порядків, найпоширенішою з яких є «класичний» метод Рунге-Кутти 4-го порядку:

$$\begin{aligned} k_1 &= f(t, y) \\ k_2 &= f\left(t + \frac{h}{2}, y + \frac{hk_1}{2}\right) \\ k_3 &= f\left(t + \frac{h}{2}, y + \frac{hk_2}{2}\right) \\ k_4 &= f(t + h, y + hk_3) \\ y(t+h) &= y(t) + h\frac{k_1 + 2k_2 + 2k_3 + k_4}{6} + O\left(h^5\right) \end{aligned} \tag{6}$$

Знехтування в розрахунках членом $O\left(h^{p+1}\right)$ призводить до локальної похибки, пропорційної h^{p+1}. Якщо крок h зменшити в деяке число n разів, то локальні похибки на кожному кроці зменшаться в n^{p+1} разів, але похибка кінцевого результату, за рахунок збільшення в n разів числа кроків, зменшиться лише в n^p разів. Отже, *глобальна похибка* (тобто похибка на всьому інтервалі інтегрування рівняння) є пропорційною h^p.

Завдання

1. Складіть програму для розв'язання задачі Коші (1), (2) методом Рунге-Кутти на інтервалі $t \in [t_0, t_{\max}]$ згідно з вашим варіантом. Конкретну обчислювальну схему (один з методів 2-го порядку, «класичний» метод 4-го порядку, або інший метод 3-го або 4-го порядку, який ви знайдете у довіднику або виведете власноруч) виберіть самостійно. Розв'язок задачі отримайте у вигляді таблиці функції та її похідної.

   Основу програмного коду запозичте з наведеного фрагменту.

```
...
t=t0
y=y0
CALL Deriv(t,y,y1) ' похідна: y1=f(t,y)
PRINT t,y,y1       ' друк нульов. рядка таблиці результатів

DO WHILE t<tmax
   CALL RungeKutta(t,y,h)  ' один крок інтегрування
   CALL Deriv(t,y,y1)      ' похідна: y1=f(t,y)
   PRINT t,y,y1            ' друк рядка таблиці результатів
LOOP
...
END                ' кінець програми

SUB RungeKutta (t,y,h)
'
' ------------------------------------------------------
' Однокроковий інтегратор диференціального рівняння
' dy/dt = f(t,y) методом Рунге-Кутти.
'
' Вхідні параметри:
'    t    - початкове значення аргументу;
'    y    - початкове значення функції;
'    h    - крок інтегрування по аргументу;
' Вихідні параметри:
'    t    - збільшене на h початкове t;
'    y    - функція, проінтегрована від t до t+h;
'
' Процедура викликає додаткову процедуру Deriv(t,y,y1)
' для обчислення правої частини y1=f(t,y) рівняння.
' ------------------------------------------------------
...
END SUB
```

2. Скористуйтеся наведеним у вашому варіанті аналітичним виразом для загального розв'язку вашого рівняння. Впевніться в його правильності: розв'яжіть рівняння самостійно або просто підставте загальний розв'язок в рівняння та перевірте тотожність лівої та правої частин.

   Визначте частковий розв'язок, що відповідає вашій початковій умові, обчисливши відповідну константу інтегрування. Розрахуйте аналітичний розв'язок і надрукуйте його в таблиці поруч із обчисленим в п. 1 методом Рунге-Кутти. Порівняйте результати.

3. Зобразьте графічно $y(t)$ та $y'(t)$.

4. Дослідіть, як впливає величина кроку інтегрування на точність розв'язку задачі. Поясніть характер отриманої залежності, визначте по ній порядок точності квадратурної формули і порівняйте його з теоретичним значенням.

Варіанти для самостійної роботи

Варіант	Диференційне рівняння $y' = f(t, y)$	t_0	y_0	$t_{\max}$	Аналітичний розв'язок ($C = const$)
1.	$y' = \dfrac{y-3}{t(3t+1)}$	0.2	1.6	1.8	$y = \dfrac{Ct}{3t+1} + 3$
2.	$y' = t(y^2+1)$	0	–1.5	2.0	$y = \operatorname{tg}\left(\dfrac{t^2}{2} + C\right)$
3.	$y' = \dfrac{1}{(1+e^{-t})y}$	–3	0.6	7	$y = \sqrt{2\ln(1+e^t) + C}$
4.	$y' = -2tye^{-t^2}$	–2	2	2	$y = C\exp(\exp(-t^2))$
5.	$y' = 2te^{t^2-y}$	–1	1.3	2	$y = \ln(e^{t^2} + C)$
6.	$y' = -\dfrac{y}{t}\ln y$	0.1	0.1	0.5	$y = e^{\frac{C}{t}}$
7.	$y' = (y+1)\operatorname{tg} t$	–1.2	4	1.2	$y = \dfrac{C}{\cos t} - 1$
8.	$y' = -(2y+1)\operatorname{ctg} t$	0.5	1.2	2.5	$y = \dfrac{C}{\sin^2 t} - \dfrac{1}{2}$
9.	$y' = \dfrac{y}{t} - 2\sqrt{\dfrac{y}{t}}$	0.1	1	3.7	$y = t\ln^2\left\lvert\dfrac{C}{t}\right\rvert$

Варіант	Диференційне рівняння $y' = f(t, y)$	t_0	y_0	$t_{\max}$	Аналітичний розв'язок ($C = const$)
10.	$y' = \frac{y}{t}\left(\ln\frac{y}{t} + 1\right)$	0.5	0.4	8.5	$y = te^{Ct}$
11.	$y' = -\frac{y}{t} + y^2 \ln t$	0.1	–3.5	3.1	$y = \frac{2}{t(C - \ln^2 t)}$
12.	$y' = -\frac{3y}{t} + t^3 y^3$	0.6	2.9	2	$y = \frac{1}{t^2\sqrt{Ct^2 + 1}}$
13.	$y' = y + e^t$	–1.3	–1	2.8	$y = (t + C)e^t$
14.	$y' = y - e^{-t}$	–2	3.8	2	$y = Ce^t + \frac{e^{-t}}{2}$
15.	$y' = 2ty + 2t^3$	–2	1.5	2	$y = Ce^{t^2} - t^2 - 1$
16.	$y' = -2ty + te^{-t^2}$	–2	0.2	2	$y = \left(\frac{t^2}{2} + C\right)e^{-t^2}$
17.	$y' = -\frac{y}{t} + 3t$	0.2	2	2.2	$y = t^2 + \frac{C}{t}$
18.	$y' = \frac{y}{t} + t + 1$	0.2	–1	4.2	$y = t(t + \ln t + C)$
19.	$y' = \frac{2y}{t} + \frac{3}{t^2}$	0.5	–2.1	5.5	$y = Ct^2 - \frac{1}{t}$
20.	$y' = -\frac{t+1}{t}y + 3te^{-t}$	0.2	1	4.2	$y = \left(t^2 + \frac{C}{t}\right)e^{-t}$
21.	$y' = -\frac{y}{t} + 2e^{t^2}$	0.1	4	1.6	$y = \frac{e^{t^2} + C}{t}$
22.	$y' = -\frac{y}{t^2} + e^{\frac{1}{t}}$	0.3	2	1.3	$y = e^{\frac{1}{t}}(t + C)$
23.	$y' = \frac{1 - y\sin t}{\cos t}$	0	4	1.57	$y = C\cos t + \sin t$
24.	$y' = -y\cos t + \frac{1}{2}\sin 2t$	0	3	6.28	$y = \sin t - 1 + Ce^{-\sin t}$

Контрольні запитання

1. Що таке поле напрямків?
2. Що являє собою загальний і частковий розв'язок диференціального рівняння?
3. Що таке задача Коші?
4. В якому вигляді відшукується чисельний розв'язок задачі Коші? А аналітичниій?
5. Дайте наочну геометричну інтерпретацію методу Ейлера (3).
6. Запишіть формули модифікованого методу Ейлера (5) при $\omega = 1$ та дайте їм наочну геометричну інтерпретацію.
7. Запишіть формули методу Х'юна (5) при $\omega = ½$ та дайте їм наочну геометричну інтерпретацію.
8. Що таке локальна і глобальна похибки чисельного розв'язку диференціального рівняння?
9. Що таке порядок точності методу?
10. Загальний вигляд обчислювальних схем методів Рунге-Кутти задається формулами

$$
\begin{aligned}
&k_1 = f(t, y)\\
&k_2 = f(t + \alpha_2 h,\ y + \beta_{21} h k_1)\\
&k_3 = f(t + \alpha_3 h,\ y + \beta_{31} h k_1 + \beta_{32} h k_2)\\
&\ldots\\
&k_s = f(t + \alpha_s h,\ y + \beta_{s1} h k_1 + \beta_{s2} h k_2 + \ldots + \beta_{s,s-1} h k_{s-1})\\
&y(t + h) = y(t) + \gamma_1 h k_1 + \gamma_2 h k_2 + \ldots + \gamma_s h k_s
\end{aligned}
$$

Конкретний метод визначається числом s і коефіцієнтами α_i, β_{ij} та γ_i, для яких повинні справджуватися умови:

$$\sum_{j=1}^{i-1} \beta_{ij} = \alpha_i,\ i = 2, \ldots\ s; \qquad \sum_{i=1}^{s} \gamma_i = 1.$$

Ці коефіцієнти часто впорядковують в так звану *таблицю Батчера*, де сума чисел в рядку праворуч від вертикальної лінії дорівнює числу ліворуч від неї:

0					
α_2	β_{21}				
α_3	β_{31}	β_{32}			
⋮	⋮	⋮	⋱		
α_s	β_{s1}	β_{s2}	⋯	$\beta_{s,s-1}$	
1	γ_1	γ_2	⋯	γ_{s-1}	γ_s

Запишіть таблиці Батчера

- для методу Ейлера (3);
- для загального методу Рунге-Кутти 2-го порядку (5) і його окремих випадків – модифікованого методу Ейлера з $\omega = 1$, методу Х'юна з $\omega = ½$ та методу Ральстона з $\omega = ⅔$;
- для «класичного» методу Рунге-Кутти 4-го порядку (6).

11. Обговоріть вплив на загальну похибку розрахунків похибок дискретизації і округлення.

Лабораторна робота № 21.

Системи звичайних диференціальних рівнянь

Мета роботи: вивчення алгоритмів розв'язку системи звичайних диференціальних рівнянь на прикладі моделювання руху супутника Землі або іншого небесного тіла.

Що зробити: розрахувати висоту r_0 перицентру чи апоцентру орбіти, яку вам пропонується промоделювати, та швидкість v_0 супутника в цій точці. Використовуючи ці величини як початкові умови, розрахувати рух супутника протягом одного оберту по орбіті і відобразіити результати розрахунків графічно. Додатково – дослідити, як похибка розрахунків залежить від кроку по часу. Перевірити, наскільки точно повна енергія супутника та його момент імпульсу зберігають незмінне значення при розрахунках.

Стислі теоретичні відомості

А. Методи Рунге-Кутти для системи диференціальних рівнянь

Для розв'язку задачі Коші системи диференціальних рівнянь виду

$$\begin{cases} \dfrac{dy_1(t)}{dt} = f_1(t, y_1, y_2, \ldots y_n) \\ \dfrac{dy_2(t)}{dt} = f_2(t, y_1, y_2, \ldots y_n) \\ \quad \ldots \\ \dfrac{dy_n(t)}{dt} = f_n(t, y_1, y_2, \ldots y_n) \end{cases}$$

з початковими умовами

$$\begin{cases} y_1(t_0) = y_{10} \\ y_2(t_0) = y_{20} \\ \quad \ldots \\ y_n(t_0) = y_{n0} \end{cases},$$

або, в запису у векторній формі

$$\frac{d\mathbf{y}(t)}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0$$

можуть застосовуватися ті ж самі методи, що і для розв'язку одного рівняння із заміною скалярної функції $y(t)$ вектор-функцією скалярного аргументу $\mathbf{y}(t)$.

Наприклад, формули сімейства методів Рунге-Кутти 2-го порядку можуть бути записані у векторному вигляді у формі

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t, \mathbf{y}) \\ \mathbf{k}_2 &= \mathbf{f}\left(t + \frac{h}{2\omega}, \mathbf{y} + \frac{h\mathbf{k}_1}{2\omega}\right) \\ \mathbf{y}(t+h) &= \mathbf{y}(t) + (1-\omega)h\mathbf{k}_1 + \omega h\mathbf{k}_2 \end{aligned} \qquad (1)$$

де h – крок інтегрування по скалярній змінній t, а $\omega \neq 0$ – вільний параметр, найбільш уживаними значеннями якого є $\omega = 1$, або $\omega = ½$, або $\omega = ⅔$.

Б. Рух тіла в полі тяжіння (задача двох тіл)

Нехай космічний апарат (супутник) порівняно невеликої маси m знаходиться в полі тяжіння планети маси M. Оскільки $M >> m$, гравітаційним впливом малої маси супутника на рух планети можна знехтувати.

Сила тяжіння, яка діє на космічний апарат, призводить до його прискорення $\mathbf{r}''$, і згідно з другим законом Ньютона,

$$m\mathbf{r}'' = -\frac{\mathbf{r}}{r} \cdot \frac{GMm}{r^2}, \qquad (2)$$

де $\mathbf{r}$ – радіус-вектор космічного апарата, що відраховується від центру планети, G – гравітаційна константа, множник $-\mathbf{r}/r$ виражає той факт, що сила діє в напрямку, протилежному $\mathbf{r}$, а штрих означає диференціювання по часу. (Добуток GM називають *гравітаційним параметром* планети. Його визначають надзвичайно точно із астрономічних спостережень.)

Скорочуючи на m і вводячи додаткову змінну – швидкість космічного апарату **v**, приводимо рівняння руху (2) 2-го порядку до системи рівнянь 1-го порядку:

$$\begin{cases} \mathbf{r}' = \mathbf{v} \\ \mathbf{v}' = -\mathbf{r}\dfrac{GM}{r^3} \end{cases} \tag{3}$$

За початкові умови треба взяти положення і швидкість космічного апарату в певний момент часу t_0:

$$\begin{cases} \mathbf{r}(t_0) = \mathbf{r}_0 \\ \mathbf{v}(t_0) = \mathbf{v}_0 \end{cases} \tag{4}$$

Визначимо систему координат так, що її початок співпадає з центром планети, координатна вісь (1) спрямована вдовж вектору $\mathbf{r}_0$, а координатна площина (1-2) визначається парою векторів $\mathbf{r}_0$ та $\mathbf{v}_0$. Тоді вектори $\mathbf{r}(t)$ та $\mathbf{v}(t)$ залишатимуться в цій площині і в подальшому, тобто задачу можна розглядати як плоску (двовимірну). Позначимо компоненти **r** як (y_1, y_2), а **v** – як (y_3, y_4). Тоді система (3) запишеться:

$$\begin{cases} y_1' = y_3 \\ y_2' = y_4 \\ y_3' = -y_1 \dfrac{GM}{\left(y_1^{\,2} + y_2^{\,2}\right)^{3/2}} \\ y_4' = -y_2 \dfrac{GM}{\left(y_1^{\,2} + y_2^{\,2}\right)^{3/2}} \end{cases} \tag{5}$$

Для простоти приймемо, що початкова швидкість космічного апарату $\mathbf{v}_0$ перпендикулярна його радіус-вектору $\mathbf{r}_0$ (рис. 21.1). Це означатиме, що його рух починатиметься від перицентру або апоцентру орбіти, тобто точки, найближчої до центра тяжіння, або найвіддаленішої від нього. (Коли йдеться про рух навколо Землі, апоцентр і перицентр називають апогеєм та перигеєм). Тоді початкові умови матимуть вигляд:

$$\begin{cases} y_1(t_0) = r_0 \\ y_2(t_0) = 0 \\ y_3(t_0) = 0 \\ y_4(t_0) = v_0 \end{cases} \tag{6}$$

Можна показати, що розв'язком рівняння (2) є рівномірний рух по круговій орбіті радіуса r_0, якщо початкова швидкість дорівнює $v_0 = v_1$, де

$$v_1 = \sqrt{\frac{GM}{r_0}} \tag{7}$$

– так звана кругова швидкість. (Якщо r_0 дорівнює радіусу планети, то v_1 називають першою космічною швидкістю.)

Позначимо параметр

$$\beta = (v_0 / v_1)^2 . \tag{8}$$

Вочевидь, руху по круговій орбіті відповідає значення $\beta = 1$.

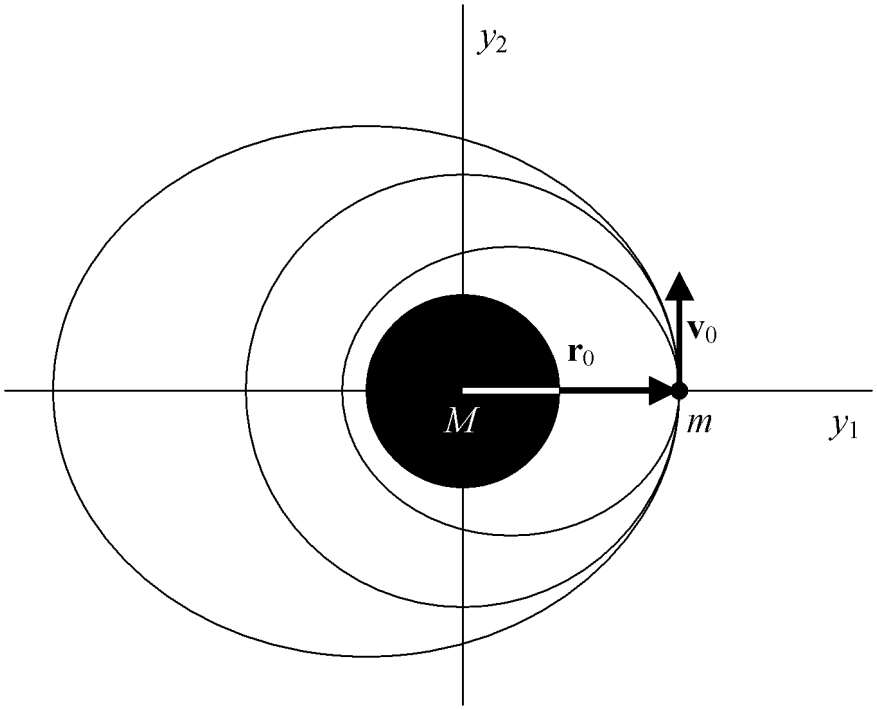


Рис. 21.1. Еліптичні орбіти при різних початкових швидкостях

Якщо $\beta < 1$ ($v_0 < v_1$), то супутник рухатиметься по еліптичній орбіті, причому початкова точка є апоцентром ($r_A = r_0$), а протилежна точка орбіти – перицентром, віддаленим від центра планети на

$$r_П = r_0 \frac{v_0^{\,2}}{2v_1^{\,2} - v_0^{\,2}} = r_0 \frac{\beta}{2 - \beta} . \tag{9}$$

Зауважимо, що якщо висота перицентра $r_П$ менша за радіус планети, космічний апарат впаде на її поверхню.

Якщо $1 < \beta < 2$ ($v_1 < v_0 < v_2$, де $v_2 = \sqrt{2}v_1$), то орбіта космічного апарату також буде еліптичною, але початкова точка буде перицентром ($r_П = r_0$), а протилежна точка орбіти – апоцентром, висота якого дається аналогічним виразом:

$$r_A = r_0 \frac{v_0^2}{2v_1^2 - v_0^2} = r_0 \frac{\beta}{2-\beta}. \tag{10}$$

Таким чином, в разі еліптичної траєкторії довжина великої півосі орбіти становить

$$a = \frac{r_A + r_П}{2} = \frac{1}{2}\left(r_0 + r_0 \frac{\beta}{2-\beta}\right) = \frac{r_0}{2-\beta}, \tag{11}$$

а її ексцентриситет –

$$e = \frac{r_A - r_П}{r_A + r_П} = |\beta - 1|. \tag{12}$$

Період обертання супутника в цьому випадку визначається формулою

$$T = 2\pi\sqrt{\frac{a^3}{GM}} = \frac{2\pi r_0}{v_1(2-\beta)^{3/2}}. \tag{13}$$

Якщо $\beta \geq 2$ ($v_0 \geq v_2$), то космічний апарат піде нескінченно далеко від центру тяжіння, описуючи розімкнуту траєкторію – параболу (при $\beta = 2$, тобто $v_0 = v_2$) або гіперболу (при $\beta > 2$, тобто $v_0 > v_2$). Величину

$$v_2 = \sqrt{2}v_1 = \sqrt{\frac{2GM}{r_0}} \tag{14}$$

називають швидкістю звільнення, а якщо r_0 дорівнює радіусу планети – то другою космічною.

Завдання

1. Згідно з вашим варіантом та даними таблиці *Орбітальні та фізичні характеристики деяких тіл Сонячної системи* визначте елементи орбіти, яку вам пропонується промоделювати, а саме:
   - довжину великої півосі орбіти a, ексцентриситет орбіти e, висоту її перицентру $r_П$ та апоцентру r_A;
   - швидкість супутника в перицентрі та апоцентрі;
   - період обертання по орбіті T.

2. Складіть програму для моделювання руху супутника навколо планети. Для розрахунків вам знадобиться процедура підходящого однокрокового інтегратора, наприклад, метода Рунге-Кутти 2-го порядку (1), а також процедура обчислення правих частин системи рівнянь (5). Головний модуль включатиме в себе лише операції вводу-виводу. Якщо після кожного кроку інтегрування компоненти радіус-вектора (y_1, y_2) позначати міткою на екрані, то вони відслідкують собою траєкторію руху супутника.

   Основу програмного коду запозичте з наведеного нижче фрагменту.

   Передбачте, щоб ваша програма перед початком розрахунків послідовно робила:

   - запит початкового радіусу r_0;
   - розрахунок і друк значень швидкостей v_1 та v_2;
   - запит початкової швидкості v_0;
   - розрахунок і друк протилежної точки орбіти – перицентру або апоцентру $r_П$ або r_A ;
   - розрахунок і друк орбітального періоду T;
   - запит загального часу розрахунку руху t_{max} ;
   - запит кроку інтегрування по часу.

   Приклад результату роботи програми наведено на рис. 21.2.

   Зауважте, що з метою полегшення читабельності в наведеному прикладі інтервали часу вводяться в годинах, після чого програма переводить їх у секунди, всі подальші розрахунки проводяться в секундах і кілометрах, а перед друком результатів час знову переводяться із секунд в години.

```
...
DO WHILE t<tmax
   CALL RungeKutta2(t,n,Y(),h)        ' 1 крок інтегрування
   ... ' позначити на екрані коорд-ти супутника (Y[1],Y[2])
LOOP
END                                   ' кінець програми
```

```
SUB RungeKutta2(t,n,Y(1),h)
'
' ------------------------------------------------------
' Однокроковий інтегратор системи диференціальних рівнянь
' dY/dt = F(t,Y) методом Рунге-Кутти 2-го порядку.
'
' Вхідні параметри:
'    t    - початкове значення скалярного аргументу;
'    n    - порядок системи рівнянь;
'    Y[n] - початкові значення компонентів вектор-функції;
'    h    - крок інтегрування по скалярному аргументу;
' Вихідні параметри:
'    t    - збільшене на h початкове t;
'    Y[n] - вектор-функція, проінтегрована від t до t+h;
'
' Процедура викликає додаткову процедуру Deriv(t,n,Y(),F())
' для обчислення правої частини dY/dt = F(t,Y).
' ------------------------------------------------------
'
...
' Декларування службових масивів для проміжних розрахунків:
   DIM K1[n], K2[n], Y2[n]
...
END SUB

SUB Deriv(t,n,Y(1),F(1))
'
' ------------------------------------------------------
' Права частина системи диф. рівнянь руху супутника
' навколо Землі.
'
' Вхідні параметри:
'    t    - час;
'    n    - порядок системи рівнянь, повинно бути n = 4;
'    Y[n] - масив координат і швидкостей супутника:
'           Y[1], Y[2] - координати вздовж осей 1, 2
'           Y[3], Y[4] - швидкості вздовж осей 1, 2;
' Вихідні параметри:
'    F[n] - вектор-функція, права частина системи рівнянь.
' ------------------------------------------------------
'
   GM=398600.4          ' Гравітаційний параметр Землі, км/с

   F[1]=Y[3]
   F[2]=Y[4]
   r3=(SQR(Y[1]^2+Y[2]^2))^3
   F[3]=-Y[1]*GM/r3
   F[4]=-Y[2]*GM/r3
END SUB
```

3. Використовуючи висоту r_0 перицентру чи апоцентру та швидкість супутника v_0 в цій точці як початкові умови, розрахуйте рух супутника протягом одного оберту по орбіті. Відображайте на екрані окремими позначками положення супутника після кожного кроку інтегрування рівнянь руху.

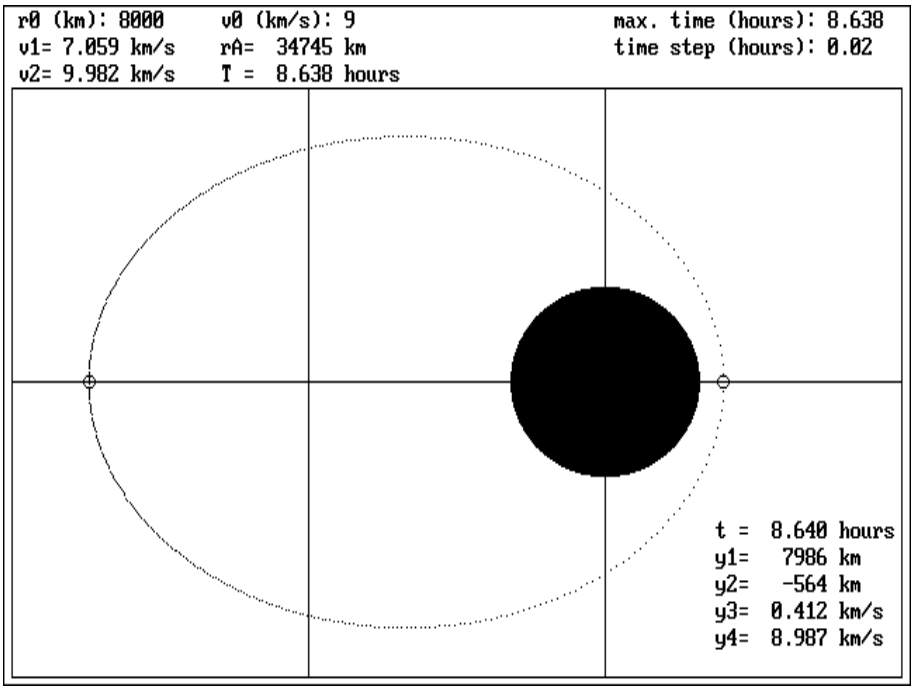


Рис. 21.2. Розрахунок руху супутника Землі.
В верхній частині екрану – запит початкових умов.
В правому нижньому кутку – друк поточних значень
аргументу і змінних системи диференціальних рівнянь.
Вивід значень y_1 та y_2 дублюється графічно

4. Дослідіть, як зміниться орбіта супутника, у випадках, якщо його початкова висота r_0 лишиться незмінною, а початкова швидкість v_0 збільшиться або зменшиться на 10%.

Додаткове завдання

5. Дослідіть, як похибка розрахунків залежить від кроку по часу h. Скористайтеся тим, що рух супутника періодичний з періодом T (13), і порівняйте положення супутника, розраховане вашою програмою в цей момент, з його початковою точкою.

   Аналогічне порівняння можна робити в момент $T/2$, порівнюючи розраховане програмою положення супутника з протилежною точкою орбіти (9), (10).

6. При русі в полі тяжіння деякі комбінації координат і швидкостей залишаються незмінними, зокрема, в кожний момент часу виконуються наступні співвідношення:

$$\frac{v^2}{2} - \frac{GM}{r} = const; \tag{15}$$

$$\mathbf{r} \times \mathbf{v} = const. \tag{16}$$

   Будучи помноженим на m, (15) виражає закон збереження повної енергії (суми кінетичної та потенціальної), а (16) – закон збереження моменту імпульсу космічного апарата.

   В позначеннях $\mathbf{r} = (y_1, y_2)$, $\mathbf{v} = (y_3, y_4)$ ці співвідношення приймають вигляд:

$$\frac{y_3^{\,2} + y_4^{\,2}}{2} - \frac{GM}{\sqrt{y_1^{\,2} + y_2^{\,2}}} = const; \tag{17}$$

$$y_1 y_4 - y_2 y_3 = const. \tag{18}$$

   Перевірте, наскільки точно ці величини зберігають незмінне значення при ваших розрахунках.

Орбітальні та фізичні характеристики деяких тіл Сонячної системи

Небесне тіло	Гравітаційний параметр GM, км^3/с^2	Велика піввісь орбіти a, км	Ексцентриситет орбіти e, 10^{-3}	Орбітальний період T, діб	Середній радіус R, км	Період обертання відносно зірок (сидерична доба), годин
Земля	398 600.4418	149 598 261	16.710	365.256366	6 371	23.93447
Місяць	4 902.7779	384 399	54.9	27.32166	1 737	*)
Марс	42 828	227 939 100	93.315	686.971	3 386	24.6229
Фобос	7.154×10^{-4}	9 377	15.1	0.3189	11.1	*)
Деймос	1.498×10^{-4}	23 460	0.2	1.26244	6.2	*)

*) Місяць, Фобос і Деймос обертаються навколо осі синхронно із рухом по орбіті, так що вони завжди повернуті до планети одним боком.

Варіанти для самостійної роботи

Варіанти 1, 12.

Низька навколоземна орбіта – орбіта, близька до кругової на висоті 200 км над поверхнею Землі (вища за атмосферу, але нижча за радіаційні пояси).

Використовується як опорна при виводі в космос космічних апаратів перед здійсненням подальших маневрів, а також в більшості пілотованих польотів.

Анімацію низької навколоземної орбіти див. тут:
http://www.youtube.com/watch?v=9snoBwilV1k .

Промоделюйте рух космічного корабля «Восток-2», параметри орбіти якого становили: перигей – 183 км над поверхнею Землі, апогей – 244 км, період обертання – близько 89 хв.

Варіанти 2, 13.

Геостаціонарна орбіта – кругова орбіта, розташована над екватором Землі, знаходячись на якій супутник обертається навколо планети синхронно з обертанням Землі навколо осі, здійснюючи оберт за *зоряну (сидеричну) добу*, тобто за 23 год. 56 хв. 4.1 сек. = 23.93447 год. Її радіус становить $\sqrt[3]{GM(T/2\pi)^2}$ = 42 164 км. Геостаціонарний супутник в небі (для наземного спостерігача) є практично нерухомим.

Використовується супутниками зв'язку.

Анімацію геостаціонарної орбіти з незначним нахилом до площини екватора див. тут: http://www.youtube.com/watch?v=AYA61xoxXhs .

Промоделюйте рух геостаціонарного супутника.

Варіанти 3, 14.

Гоманівською траєкторією називають еліптичну орбіту, що використовується для переходу між двома круговими орбітами, які знаходяться в одній площині. Вона дотикається до цих двох орбіт в своєму апоцентрі і перицентрі. Орбітальний маневр для переходу включає в себе 2 імпульси роботи двигуна на розгін – для входу на гоманівську траєкторію та для сходу с неї. Названа на честь німецького вченого Вальтера Гомана (Walter Hohmann), який її описав в 1925 році.

Геоперехідна орбіта – сильно витягнута гоманівська еліптична траєкторія космічного апарату, перигей якої лежить на відстані низької навколоземної орбіти (див. Варіант 1), а апогей – на відстані геостаціонарної орбіті (див. Варіант 2).

Використовується для виводу космічного апарату на геостаціонарну орбіту. При досягненні апогею двигун надає апарату додатковий розгінний імпульс, який перетворює його еліптичний рух в круговий з періодом обертання навколо Землі, рівним тривалості сидеричної доби.

Анімацію геоперехідної орбіти див. тут:
https://www.youtube.com/watch?v=gmR6W-c-kOE

Промоделюйте рух космічного апарату, що переходить з низької навколоземної орбіти на геостаціонарну. Вважайте, що низька навколоземна орбіта є круговою на відстані 200 км від земної поверхні.

Варіанти 4, 15, 23.

Орбіта «Тундра» – сильно нахилена до площини екватора еліптична геосинхронна орбіта з ексцентриситетом 0.25 – 0.4.

Геосинхронною називають орбіту, на якій період обертання супутника дорівнює зоряному періоду обертання Землі – сидеричній добі. (див. Варіант 2). У всіх геосинхронних орбіт (як кругових, так і еліптичних) велика піввісь становить 42 164 км. Окремим випадком геосинхронної орбіти є геостаціонарна (див. Варіант 2), коли треком супутника (його проекцією на земну поверхню) є єдина точка на екваторі. У загальному випадку, коли орбіта має ненульовий ексцентриситет або нахил до екваторіальної площини Землі, трек являє собою більш чи менш викривлену вісімку.

Супутники, що використовують високі еліптичні орбіти, рухаються з великою швидкістю в перигеї, але сильно уповільнюються в апогеї. Коли космічний апарат знаходиться поблизу апогею, у наземного спостерігача складається враження, що супутник майже не рухається протягом декількох годин, тобто його орбіта стає квазігеостаціонарною. З іншого боку, точка квазігеостаціонару може бути розташована над будь-якою точкою земної кулі, а не тільки над екватором, як у геостаціонарних супутників. Ця властивість використовується в приполярних широтах (вище 76 – 78° пн.ш./пд.ш.), де геостационарні супутники спостерігаються дуже низько над горизонтом або навіть знаходяться під ним. В цих зонах прийом із геостаціонарного супутника сильно ускладнений або зовсім неможливий.

Анімацію орбіти «Тундра» та концептуально близької до неї орбіти «Молнія» див. тут: http://www.youtube.com/watch?v=Yo9rFmfX42Q .

Промоделюйте рух супутника по орбіті «Тундра» з ексцентриситетом 0.2684.

Така орбіта з нахилом 63,4° до площини екватора використовується американською компанією «Sirius XM Radio», що для радіомовлення експлуатує угрупування з трьох супутників, які рухаються вздовж однієї орбіти з інтервалом 8 годин один за одним. Кожний з них розташовується над континентальною частиною США близько 16 годин на добу, причому весь час над країною знаходиться принаймні один супутник.

Варіанти 5, 16, 24.

Орбіта «Молнія» – сильно нахилена до площини екватора еліптична орбіта з періодом обертання, рівним точно половині сидеричної доби і великим ексцентриситетом, так що перигей знаходиться на висоті 500 – 1000 км над поверхнею Землі, а апогей сягає майже 40 000 км від Землі.

Орбіта «Молнія» концептуально схожа з орбітою «Тундра» (див. Варіант 4), але за зоряну добу супутник здійснює не один, а два витки, і під час проходження ним апогею під супутник потрапляють два райони Земної кулі, віддалених один від одного на 180° по довготі. Апогей цієї орбіти лежить нижче, ніж у «Тундри», що дозволяє використовувати передавачі меншої потужності.

Орбіта використовується російським (раніше – радянським) угрупуванням супутників «Молнія» та «Меридіан», що складається з восьми космічних апаратів на високоеліптичних орбітах з апогеєм в Північній півкулі, період обертання яких дорівнює половині сидеричної доби (тобто, трохи менше 12 годин). Космічні апарати розділені на чотири пари, в кожній з яких супутники рухаються вздовж однієї наземної траси з інтервалом в 6 годин один за одним. Траси пар зсунуті один відносно одного на 90° по довготі, тобто 8 супутників забезпечують покриття в усій Північній півкулі. Апогеї добових витків супутників двох пар знаходяться над територією Центрального Сибіру і Північної Америки, а у супутників двох інших пар – над Західною Європою та Тихим океаном.

Анімацію орбіти «Молнія» див. тут:
http://www.youtube.com/watch?v=G8DP4QrKLwI .

Промоделюйте рух супутника по орбіті «Молнія» з перигеєм 500 км над земною поверхнею.

Варіанти 6, 17.

Промоделюйте рух Місяця по орбіті навколо Землі.

Варіанти 7, 18.

Низька селеноцентрична орбіта – кругова орбіта на висоті близько 110 км над поверхнею Місяця. Використовується як орбіта очікування перед посадкою космічного апарату на Місяць.

Промоделюйте рух командного модуля корабля «Аполлон-11», який в очікуванні місячного модуля, що здійснив посадку, знаходився на орбіті з параметрами: перицентр (периселеній) – 100 км над поверхнею Місяця, апоцентр (апоселеній) – 122 км, період обертання – близько 119 хв.

Варіанти 8, 19.

Промоделюйте рух Фобоса по орбіті навколо Марса.

Варіанти 9, 20.

Промоделюйте рух Деймоса по орбіті навколо Марса.

Варіанти 10, 21.

Промоделюйте Гоманівську траєкторію (див. Варіант 3) для переходу з орбіти Деймоса на орбіту Фобоса. Ексцентриситетами орбіт Фобоса і Деймоса можна знехтувати і вважати їх круговими.

Варіанти 11, 22.

Промоделюйте рух супутника на стаціонарній орбіті навколо Марсу (див. Варіант 2).

Контрольні запитання

1. Як розрахувати гравітаційний параметр GM планети знаючи її радіус R та величину прискорення сили тяжіння g на її поверхні? Розрахуйте гравітаційний параметр Землі і порівняйте з наведеним у таблиці *Орбітальні та фізичні характеристики деяких тіл Сонячної системи*.
2. Поясніть, яким чином можна знайти гравітаційний параметр планети із астрономічних спостережень за її супутником? Скористайтеся формулою (13).

   Гравітаційний параметр Землі, визначений таким чином, становить $398\,600.4418 \pm 0.0008$ км^3/с^2, тобто можлива похибка становить

близько $2\cdot10^{-9}$, на відміну від величини гравітаційної константи G, яку сучасні методи дозволяють визначити лише з точністю $1\cdot10^{-4}$.

3. Як, на вашу думку, астрономи визначають масу Землі або маси віддалених небесних тіл?

4. Поясніть різницю між зоряною (сидеричною) добою, тривалістю 23 год. 56 хв. 4.1 с, і сонячною (синодичною), тривалістю 24 год. рівно.

5. Доведіть співвідношення (17) та (18). Для цього потрібно записати похідні лівих частин цих виразів та довести, що вони тотожньо дорівнюють нулю за умови, що змінні пов'язані системою диференціальних рівнянь (5).

6. Доведіть, що при русі космічного апарата (супутника) по замкненій еліптичній орбіті його швидкість v та відстань r від центру тяжіння в будь-який момент часу пов'язані між собою залежністю

$$v^2 = GM\left(\frac{2}{r} - \frac{1}{a}\right). \tag{19}$$

Для цього запишіть співвідношення (15) та (16) в точках апоцентру і перицентру і покажіть, що константа в правій частині (15) становить $-\dfrac{GM}{r_{\text{А}} + r_{\text{П}}}$.

7. Користуючись (16) та (19), виведіть формули (9) та (10).

8. Часто в довідниках при описі орбіти вказують не відстані перицентру та апоцентру $r_{\text{П}}$ та $r_{\text{А}}$, а велику піввісь орбіти a та її ексцентриситет e. Скориставшись (11) та (12) покажіть, що $r_{\text{П}}$ та $r_{\text{А}}$ можуть бути розраховані за формулами

$$\begin{cases} r_{\text{П}} = a(1-e) \\ r_{\text{А}} = a(1+e) \end{cases} \tag{20}$$

9. Поясніть, чому на рис. 21.2 дистанція між послідовними розрахованими точками траєкторії космічного апарату поблизу перицентра більша, ніж поблизу апоцентра?

10. Можна значно зменшити загальний обсяг розрахунків, якщо в процесі обчислень коригувати крок інтегрування по часу h: поблизу перицентру робити його меншим, а ближче до апоцентру – збільшувати. Запропонуйте простий алгоритм вибору h перед кожним кроком інтегрування.

Лабораторна робота № 22.

1-вимірне рівняння Шредінгера. Розв'язання методом скінченних різниць

Мета роботи: вивчення методики розв'язання крайових задач методом скінченних різниць (методом сіток).

Що зробити: розрахувати енергетичні рівні стаціонарних станів електрона і відповідні їм хвильові функції, розв'язуючи рівняння Шредингера мет одом сіток. Скористатися процедурою діагоналізації симетричної матриці методом обертань Якобі. Зобразити графічно кілька розрахованих нижніх енергетичних рівней, а також відповідні їм хвильові функції. Додатково – порівняти між собою розраховані значення енергетичних рівней з аналітичними розв'язками ідеалізованої задачі.

Стислі теоретичні відомості

Метод скінченних різниць (*метод сіток*) – чисельний метод розв'язку задач диференціального та інтегрального числення, оснований на заміні функцій неперервного аргументу функціями дискретного аргументу. Така заміна приводить до заміни інтегралів – сумами, диференціальних операторів – різницевими операторами, а диференціальних рівнянь – системами алгебраїчних рівнянь.

Розглянемо застосування методу скінченних різниць на прикладі *рівняння Шредингера*. Поведінка електрона в 1-вимірній потенційній ямі описується хвильовою функцією $\psi(x)$, що задовольняє рівнянню

$$-\frac{\hbar^2}{2m}\frac{d^2\psi(x)}{dx^2}+U(x)\psi(x)=W\psi(x)\ , \qquad (1)$$

де

x – лінійна координата,
$\hbar = 1.054\ 572 \times 10^{-34}$ Дж·с – стала Планка,
$m = 9.109\ 554 \times 10^{-31}$ кг – масса електрона,
$U(x)$ – профіль потенційної енергії ями,
W – енергетичні рівні стаціонарних станів (ми не позначаємо енергію більш звичною літерою E, щоб уникнути збігу з одиничною матрицею).

Вважатимемо, що профіль $U(x)$ має скінченну ширину L (рис. 22.1). В цьому випадку на $\psi(x)$ накладаються *граничні умови*:

$$\psi(0) = \psi(L) = 0. \tag{2}$$

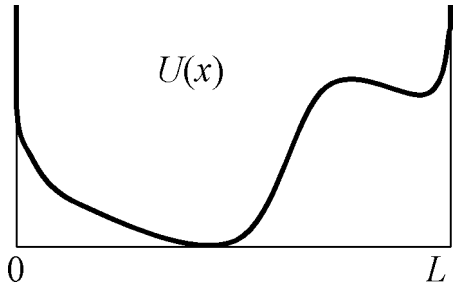


Рис. 22.1. Профіль потенційної енергії ями.
$U(x) = +\infty$ при $x < 0$ та $x > L$

Крім того, на $\psi(x)$ додатково накладається *умова нормування*

$$\int_0^L |\psi(x)|^2 dx = 1, \tag{3}$$

яка означає, що загальна ймовірність знаходження частинки в ямі дорівнює 1.

Перейдемо до функцій дискретного аргументу. Розіб'ємо ділянку $[0, L]$ на *сітку* з n маленьких інтервалів довжиною $h = L/n$ (рис. 22.2). (Ми змушено позначаємо суттєво різні величини $\hbar$ та h схожими літерами внаслідок усталеної традиції.)

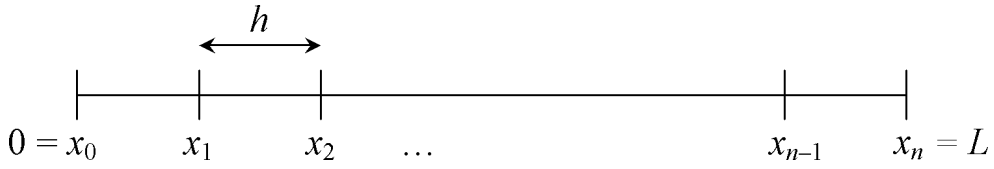


Рис. 22.2. Сітка $x_i = ih$

Функцію $\psi(x)$ шукатимемо лише у вузлах сітки x_i, позначаючи її значення в них як $\psi_i = \psi(x_i)$. Аналогічно, позначатимемо відомі значення $U(x)$ у вузлах сітки як $U_i = U(x_i)$.

Замінимо диференціальне рівняння (1) системою алгебраїчних рівнянь для знаходження ψ_0, ψ_1, … ψ_i, … ψ_n. (Звичайно, чим більшим буде n, тим точнішим буде розв'язок, але й тим складнішою система.)

Похідну у внутрішніх точках сітки (крім 0 та L) замінимо наближеною скінченнорізницевою формулою:

$$\left.\frac{d^2\psi(x)}{dx^2}\right|_{x=x_i} = \frac{1}{h^2}\left(\psi_{i-1} - 2\psi_i + \psi_{i+1}\right). \tag{4}$$

Тоді рівняння Шредингера (1) заміниться системою

$$-\left(\frac{\hbar^2}{2m}\right)\frac{1}{h^2}\left(\psi_{i-1} - 2\psi_i + \psi_{i+1}\right) + U_i\psi_i = W\psi_i\,, \qquad i = 1, 2, \ldots n-1, \tag{5}$$

граничні умови (2) заміняться співвідношенням

$$\psi_0 = \psi_n = 0\,, \tag{6}$$

а умова нормування (обчислюємо інтеграл за методом трапецій) прийме вигляд

$$h\left(\frac{|\psi_0|^2}{2} + |\psi_1|^2 + \ldots + |\psi_{n-1}|^2 + \frac{|\psi_n|^2}{2}\right) = 1,$$

або ж, з урахуванням (6),

$$h\sum_{i=1}^{n-1}|\psi_i|^2 = 1. \tag{7}$$

Позначимо

$$Q = \frac{\hbar^2}{2m} = 6.104\ 152\times10^{-39}\ \text{Дж·м}^2, \qquad R = \frac{Q}{h^2} = \frac{Q}{L^2}n^2.$$

Запишемо систему рівнянь (5) з урахуванням нових позначень:

$$-R\psi_{i-1} + (2R + U_i - W)\psi_i - R\psi_{i+1} = 0\,, \quad i = 1, 2, \ldots n-1$$

або, в матричній формі

$$\begin{pmatrix} (2R+U_1-W) & -R & & \ldots & 0 \\ -R & (2R+U_2-W) & -R & & \ldots \\ & -R & (2R+U_3-W) & & \\ \ldots & & & \ldots & -R \\ 0 & \ldots & & -R & (2R+U_{n-1}-W) \end{pmatrix} \times$$

$$\times \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \ldots \\ \psi_{n-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \ldots \\ 0 \end{pmatrix}$$

чи більш компактно

$$(\mathbf{H} - W\mathbf{E})\boldsymbol{\psi} = \mathbf{0}\,, \tag{8}$$

де квадратна симетрична матриця $(n-1) \times (n-1)$

$$\mathbf{H} = \begin{pmatrix} (2R+U_1) & -R & & \ldots & 0 \\ -R & (2R+U_2) & -R & & \ldots \\ & -R & (2R+U_3) & & \\ \ldots & & & \ldots & -R \\ 0 & \ldots & & -R & (2R+U_{n-1}) \end{pmatrix} \tag{9}$$

є гамільтоніан, $\mathbf{E}$ – одинична матриця, $\boldsymbol{\psi}$ – вектор невідомих значень хвильової функції $\{\psi_1, \psi_2, \ldots \psi_{n-1}\}$, що відшукуються.

Таким чином, рівняння Шредингера зводиться до задачі на власні значення гамільтоніана.

Після розв'язку задачі на власні значення отримуємо набір власних чисел гамільтоніана – енергетичних рівней стаціонарних станів електрона

$$W_{(1)},\ W_{(2)},\ \ldots\ W_{(n-1)}$$

та відповідний кожному власному числу власний вектор

$$\begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \ldots \\ \psi_{n-1} \end{pmatrix}_{(1)}, \quad \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \ldots \\ \psi_{n-1} \end{pmatrix}_{(2)}, \quad \ldots, \quad \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \ldots \\ \psi_{n-1} \end{pmatrix}_{(n-1)}.$$

Оскільки власні вектори визначаються з точністю до довільного множника, їх необхідно віднормувати згідно з (7):

$$\boldsymbol{\psi}_{(k)}^{\text{норм}} = \frac{\boldsymbol{\psi}_{(k)}}{\sqrt{h}\,\left|\boldsymbol{\psi}_{(k)}\right|}, \qquad \text{де} \qquad \left|\boldsymbol{\psi}_{(k)}\right| = \sqrt{\sum_{i=1}^{n-1}\left|\psi_{i(k)}\right|^2}. \tag{10}$$

Оцінимо межі придатності такого способу розрахунків.

Із аналітичної теорії рівнянь типу Шредингера відомо наступне. Якщо енергетичні рівні стаціонарних станів $W_{(1)}$, $W_{(2)}$, … (а їх кількість є нескінченою в реальній, недискретизованій задачі) занумерувати за зростанням

$$W_{(1)} < W_{(2)} < \ldots < W_{(k)} < \ldots,$$

то відповідна хвильова функція $\psi_{(k)}(x)$ матиме вигляд k напівхвиль і перетинатиме вісь абсцисс (k–1) раз.

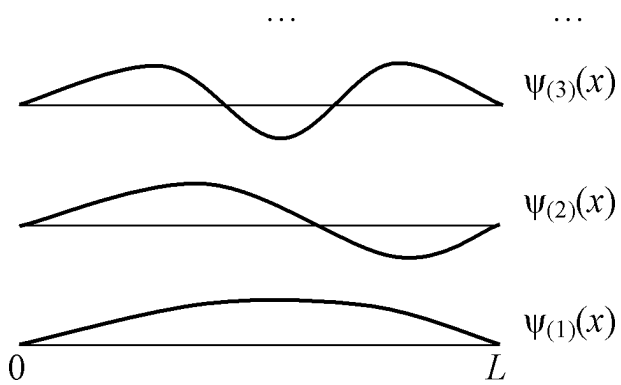


Рис. 22.3. Якісна поведінка хвильових функцій $\psi_{(k)}(x)$ нижніх енергетичних рівнів

Якщо в дискретизованій задачі $n >> k$, то скінченнорізницеве наближення (4) похідної є коректним. Але якщо k близьке до $(n-1)$, таке ствердження перестає бути справедливим (рис. 22.4)

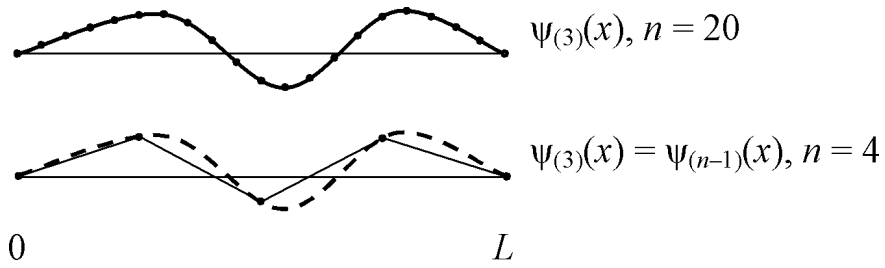


Рис. 22.4. До питання коректності
скінченнорізницевого наближення рівняння Шредингера

Таким чином, для розрахунку нижніх k стаціонарних станів електрону в потенційній ямі необхідно використовувати скінченнорізницеву сітку з числом вузлів n в кілька разів більше за k.

Завдання

1. Розрахуйте енергетичні рівні стаціонарних станів електрона і відповідні їм хвильові функції для трьох профілей потенційної ями $U(x)$ згідно з вашим варіантом.

   При розрахунках, зважаючи на те, що $Q = 6.104\ 152 \times 10^{-39}$ Дж·м^2 (на рівні найменшого машинного числа underflow level для деяких трансляторів), доцільно перейти від одиниць СІ до більш прийнятних одиниць вимірювання енергій та відстаней: 1 еВ = $1.602\ 177 \times 10^{-19}$ Дж та 1 Å = 10^{-10} м. Тоді $Q = 3.809\ 911$ еВ·Å^2, всі енергії мають виражатися в електрон-вольтах, а відстані – в ангстремах.

   Задайтеся числом вузлів сітки n в кілька десятків і скористайтеся процедурою діагоналізації симетричної матриці методом обертань Якобі, яку ви склали при виконанні лабораторної робти № 11. Після застосування цього алгоритму визначені власні вектори уже є нормованими $\sum_{i=1}^{n-1} \left|\psi_{i(k)}\right|^2 = 1$, що спрощує процедуру подальшого нормування (10).

2. Зобразіть на екрані кілька розрахованих нижніх енергетичних рівней $W_{(k)} \le (2\ldots3)U_0$, де U_0 – висота бар'єру між западинами для вашого варіанту, а також відповідні їм хвильові функції $\psi_{(k)}(x)$ (для їх зображення потрібно застосувати інший масштаб, див. рис. 22.5). За бажанням упорядкуйте значення $W_{(k)}$ за величиною.

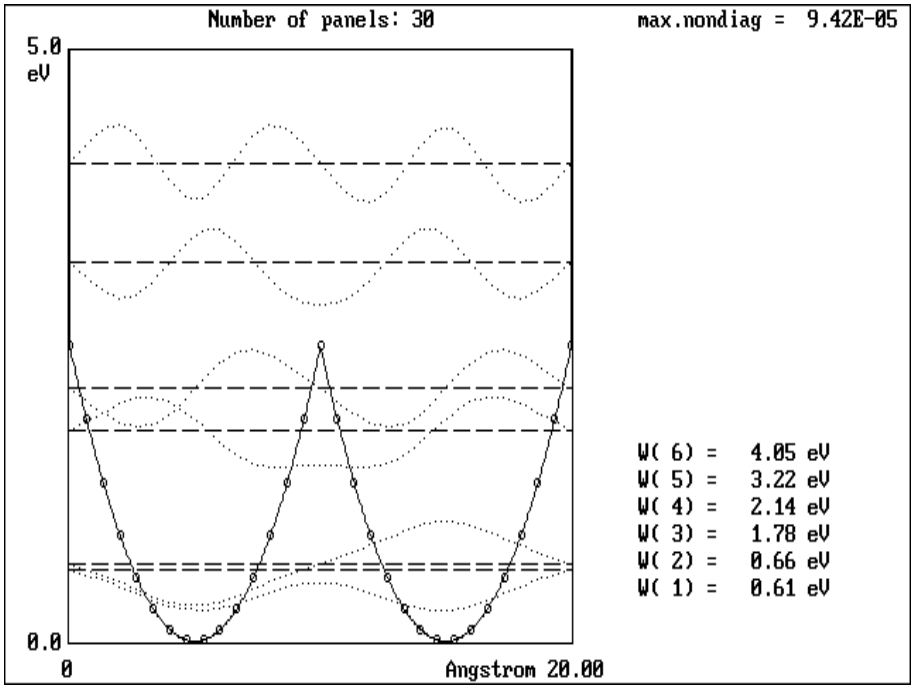


Рис. 22.5. Енергетичні рівні та хвильові функції стаціонарних станів електрона в потенціальній ямі шириною L = 20 Å з двома параболічними западинами типу $U(x) = \beta x^2$, β = 0.1 еВ/Å^2.

Суцільна лінія – профіль потенціальної ями $U(x)$, відмічені вузли сітки. Штрихові лінії – енергетичні рівні. Показані $W_{(k)} \le 5.0$ еВ. Поруч наведені значення $W_{(k)}$ (упорядковані за величиною). Пунктирні лінії – хвильові функції $\psi_{(k)}(x)$ (в іншому масштабі, вісь абсцис суміщена з висотою відповідного енергетичного рівня).

Для розрахунків застосовується скінченнорізницева сітка з 30 вузлами. Діагоналізація гамільтоніану проводиться методом обертань Якобі. Для контролю за ходом ітераційного процессу величина максимального недіагонального елементу матриці на кожній ітерації виводиться на екран (у правому верхньому куті). Ітерації припиняються, коли цей елемент стає меншим за 10^{-4}

Додаткове завдання

3. Порівняйте між собою розраховані значення $W_{(k)}$ для потенціальних профілей з однією, двома і трьома западинами, а також із аналітичними розв'язками ідеалізованої задачі з однією западиною.

   У випадку пласкої потенційної ями з нескінченно високими стінками (тобто $U(x) = 0$ при $0 < x < L$ та $U(x) = +\infty$ при $x < 0$ або $x > L$, непарні варіанти) енергії стаціонарних станів даються співвідношеннями

$$W_{(1)} = \pi^2 \frac{Q}{L^2}, \quad W_{(k)} = k^2 W_{(1)}, \quad k = 1, 2, 3, \ldots$$

   У випадку параболічної ями (тобто $U(x) = \beta x^2$, парні варіанти)

$$W_{(1)} = \sqrt{Q\beta}, \quad W_{(k)} = (2k-1) W_{(1)}, \quad k = 1, 2, 3, \ldots$$

Варіанти для самостійної роботи

Для *непарних варіантів* профілі $U(x)$ являють собою (рис. 22.6):

- пласку потенціальну яму з нескінченно високими стінками шириною $L = a$;
- яму шириною $L = 2a + b$ з двома западинами шириною a, що відділені між собою бар'єром шириною b та висотою U_0;
- яму шириною $L = 3a + 2b$ з трьома западинами шириною a, що відділені між собою бар'єрами шириною b та висотою U_0.

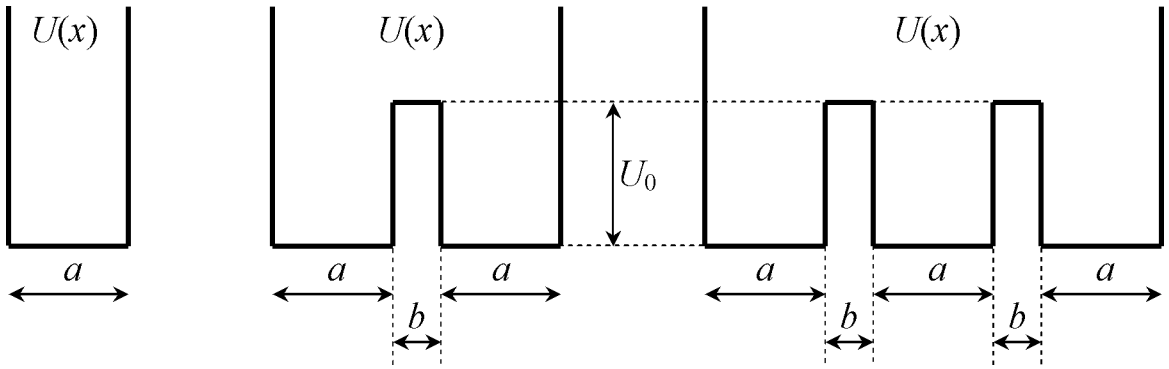


Рис. 22.6. Три профілі потенціальної ями U(x) для непарних варіантів.

Варіант	a, Å	b, Å	U_0, еВ
1	3	0.5	50
3	3	1	25
5	4	0.5	30
7	4	1	15
9	5	1	20
11	5	1.5	10
13	6	1	10
15	6	2	5
17	7	1	10
19	7	2	5
21	8	1	6
23	8	2	4

Для *парних варіантів* профілі $U(x)$ являють собою (рис. 22.7):

- потенціальну яму з нескінченно високими стінками шириною $L = a$ і параболічним дном $U = \beta(x–a/2)^2$;
- яму шириною $L = 2a$ з двома параболічними западинами $U = \beta(x–a/2)^2$ при $0 < x < a$ та $U = \beta(x–3a/2)^2$ при $a < x < 2a$;
- яму шириною $L = 3a$ з трьома параболічними западинами $U = \beta(x–a/2)^2$ при $0 < x < a$, $U = \beta(x–3a/2)^2$ при $a < x < 2a$ та $U = \beta(x–5a/2)^2$ при $2a < x < 3a$.

Як видно, бар'єр, що відділяє западини в ямах шириною $2a$ та $3a$, має висоту $U_0 = \beta(a/2)^2$.

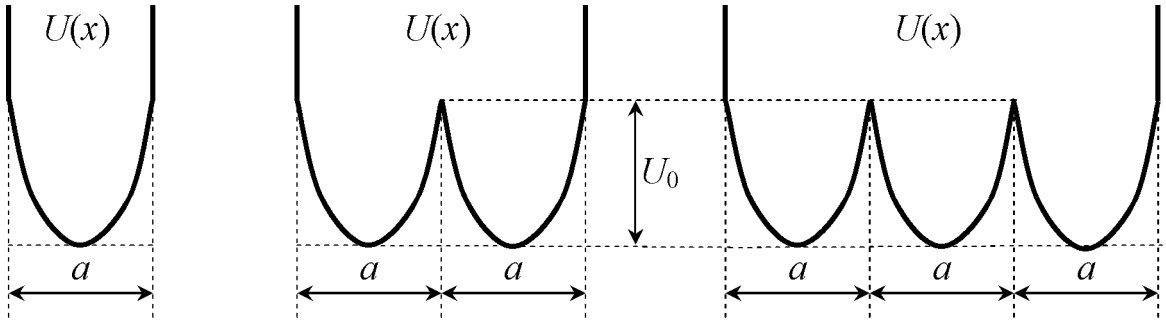


Рис. 22.7. Три профілі потенційної ями $U(x)$ для парних варіантів.

Варіант	a, Å	β, еВ/Å^2
2	3.5	5
4	4	5
6	4.5	5
8	5	1
10	5.5	1
12	6	1
14	6.5	0.5
16	7	0.5
18	7.5	0.5
20	8	0.2
22	8.5	0.2
24	9	0.2

Контрольні запитання

1. В чому полягає суть методу скінченних різниць?
2. Поясніть походження скінченнорізницевої формули (4) для наближення похідної. Які ще формули для наближення першої та другої похідних вам відомі?
3. Оцініть обсяг оперативної пам’яті, який потрібен для зберігання масивів, що використовуються в ваших розрахунках.
4. Яким чином ви визначали масштаб для зображення хвильових функцій?
5. Поясніть, чому у випадку потенціальної ями з двома або трьома западинами, нижні енергетичні рівні групуються відповідно у дублети або триплети з близькими величинами енергії? Як поведуть себе енергетичні рівні в ситуації, коли профіль потенціальної ями є періодичною функцією із макроскопічною кількістю однотипних западин?
6. Чи отримали ви моральне задоволення від виконання циклу лабораторних робіт з обчислювальноъ математики? Якщо у вас виникли додаткові запитання – поставте їх своєму викладачу, якщо з’явилися зауваження – повідомте про них автору.

Додаток.

Деякі елементи синтаксису мови програмування Borland Turbo Basic 1.1

Для ілюстрації алгоритмів в тексті практикуму подекуди використовуються фрагменти програм на Basic версії Borland Turbo Basic 1.1. Цей розділ ставить за мету допомогти читачеві адаптувати наведені фрагменти до будь-якої мови програмування і, аж ніяк не претендуючи на повноту опису цього діалекту Basic, містить короткий опис лише тих елементів його синтаксису, які використовуються в тексті.

При описі елементів синтаксису використовуються такі позначення:

- квадратні дужки **[ ]** означають, що поміщена в них інформація не є обов'язковою;
- фігурні дужки **{ }** вказують на вибір двох чи більше варіантів, один з яких має бути використаний. Варіанти відділяються один від одного вертикальною рискою (**|**);
- три крапки (. . .) указують на те, що частина команди може бути повторена необхідну кількість разів;
- три вертикально розташованих крапки указують на пропуск одного чи більше рядків програми.

Загальні риси

Оператори та коментарі

Програма складається з одного чи більше рядків программного коду. Кожний рядок може містити один чи декілька операторів, а також коментар, що може містити будь-який текст. Оператори розділяються двокрапкою. Коментар добавляється в кінці рядка та відокремлюється від власне тексту програми апострофом (**'**):

оператор **[**: *оператор***]** ... **[**'*коментар***]**

Великі і малі літери в ключових словах операторів не розрізняються. Коментарі транслятором ігноруються.

Дуже довгий оператор можна перенести на наступний рядок, скориставшись позначкою продовження – символом підкреслення (_) в кінці рядка. Тоді наступний рядок розглядатиметься як продовження попереднього, і вони разом утворюватимуть один логічний рядок.

Змінні

Змінна – це ідентифікатор, що представляє значення, яке зберігається в певному місці машинної пам'яті. Значення змінної може змінюватися при виконанні програми. Імена змінних повинні починатися з літери та можуть містити будь-яку кількість букв і цифр. Великі і малі літери в іменах змінних не розрізняються.

Масиви. Оператор DIM

Масив – це группа даних, що мають однакове ім'я змінної. Дані, з яких складається масив, називають елементами масива. Окремі елементи масиву помічаються індексами. Масиви можуть мати один чи декілька індексів. Одновимірний масив (тобто масив з одним індексом) – це просто список значень. Двувимірний масив являє собою таблицю чисел з рядками і стовпчиками даних. Можливі також і багатовимірні масиви.

Призначити масиву ім'я, а також визначити кількість та організацію його елементів, можна оператором `DIM` :

```
DIM змінна[індекси] [, змінна[індекси]] ...
```

Тут *змінна* – це ідентифікатор масиву, *індекси* – це взятий в квадратні дужки перелік з одного чи більше цілих чисел чи виразів, розділених комами, які визначають розмір масиву, тобто максимальне значення індексу по кожному з вимірів. Наприклад:

```
DIM payments[55], a[15]   ' два одновимірних списки
DIM b[15,20]              ' двувимірна таблиця
DIM c[5,5,10,20,3]        ' п'ятивимірний масив
```

Індекси елементів масиву – це цілочисельні вирази в квадратних дужках праворуч від імені масиву. Наприклад, `payments[3]` та `payments[44]` – це два елементи масиву `payments`. Елемент масиву можна використовувати в операторі або виразі як будь-яку іншу змінну.

Масив `payments` та звичайна («скалярна») змінна, також названа `payments` , – це різні змінні.

Введення даних з клавіатури. Оператор INPUT

Оператор `INPUT` призначений для введення даних з клавіатури:

```
INPUT ["текст_підказки",] перелік_змінних
```

Тут *текст_підказки* є необов'язковим параметром, *перелік_змінних* – одна чи декілька змінних, розділених комами.

Оператор `INPUT` чекає, доки користувач не введе дані з клавіатури, після чого присвоює значення заданим змінним. У випадку, коли *перелік_змінних* містить кілька змінних, дані при введенні повинні розділятися комами. Наприклад:

```
INPUT "Введіть ваш вік та вагу: ", Age, Weight
```

Числові вирази. Арифметичні операції. Вбудовані функції

Числові вирази складаються з числових констант та змінних, що розділяються в потрібних місцях арифметичними операціями. Перелік арифметичних операцій та порядок їх виконання наведені в таблиці.

Арифметичні операції в порядку зниження пріоритету їх виконання

Операція	Дія	Приклад
^	Піднесення до степеню	10^4
-	Унарний мінус	-16
*, /	Множення, ділення	45*19, 45/19
\	Цілочисельне ділення	45\19
MOD	Залишок цілочисельного ділення	45 MOD 19
+, -	Додавання, віднімання	45+19, 45-19

Операції в дужках мають вищий пріоритет і завжди виконуються першими. Всередині дужок діють загальні правила.

Числові вирази також можуть містити функції, деякі з яких вбудовані безпосередньо в транслятор. Приклади вбудованих математичних функцій наведені в таблиці.

Деякі вбудовані математичні функції

Функція	Дія
`ABS(x)`	Абсолютне значення аргументу
`INT(x)`	Найбільше ціле, що не перевищує аргументу
`SGN(x)`	Знак аргументу. Результат дорівнює `-1` при `x<0`, `0` при `x=0`, `+1` при `x>0`
`SQR(x)`	Квадратний корінь, аргумент має бути невід'ємним
`EXP(x)`	Експонента, тобто число *e*=2.71828… в степені `x`
`LOG(x)`	Натуральний логарифм, аргумент має бути додатнім
`SIN(x)`, `COS(x)`, `TAN(x)`	Синус, косинус, тангенс кута в радіанах
`ATN(x)`	Арктангенс. Результат в радіанах від $-\pi/2$ до $\pi/2$
`RND`	Псевдовипадкове число в діапазоні від 0 до 1. Аргумент відсутній

Оператор присвоювання

Оператор присвоювання присвоює змінній значення виразу:

змінна = вираз

Наприклад:

```
x = 37.4
y = 37.4/15
z[1] = a
z[2] = 37.4/a
abc = SQR((c + d)/z[1]) * SIN(37.4/a)
```

Виведення даних на екран. Оператор PRINT

Оператор `PRINT` надсилає дані на екран. Його синтаксис:

`PRINT` **[***перелік_виразів***]** **[;]**

Тут *перелік_виразів* – послідовність числових виразів та/або фрагментів тексту в лапках (`"`), що розділені крапками з комою (`;`) або комами (`,`). `PRINT` використовує ці знаки пунктуації для визначення місця на екрані, де розміщувати друковані дані.

Для швидкого і охайного виведення екран розділяється на «друкарські зони» по 14 колонок кожна. Кома в *переліку_виразів* означає, що наступні дані будуть виводитися з початку наступної друкарської зони. Крапка з комою задає виведення наступних даних безпосередньо після попередніх безвідносно до друкарських зон.

Якщо оператор PRINT завершується крапкою з комою, наступний оператор PRINT почне друк з того ж рядка. Якщо ж в кінці крапка з комою відсутня, після друку даних здійснюється перехід на новий рядок. Якщо і *перелік_виразів* відсутній, виконується лише перехід на новий рядок.

Наприклад:

```
a = 1
b = 2
c = 3
PRINT "a="; a, "b="; b, "c="; c
```

Логічні структури

Логічні вирази. Операції порівняння та логічні операції.

Логічний вираз – це вираз, результат якого є булєвим, тобто TRUE (вірно) або FALSE (невірно). Логічні вирази зазвичай використовуються в операторі IF або інших операторах прийняття рішення для визначення напрямку подальшого виконання програми.

Логічні вирази утворюються як результат операцій порівняння над числами.

Операції порівняння

Операція	Порівняння	Приклад
=	Рівність	x = y
<>	Нерівність	x <> y
<	Меньш, ніж	x < y
>	Більш, ніж	x > y
<=	Меньше чи дорівнює	x <= y
>=	Більше чи дорівнює	x >= y

Якщо арифметичні операції та операції порівняння зведені в одному виразі, пріоритет арифметичних операцій вищий, і вони завжди виконуються першими. Наприклад, `4+5 < 4*3` має результатом `TRUE`.

Логічні вирази можна компонувати один з одним за допомогою логічних операцій, які виконують дії над булєвими результатами.

Логічні операції в порядку зниження пріоритету їх виконання

Операція	Дія	Приклад
`NOT`	Заперечення	`NOT a>b`
`AND`	Та	`a>b AND x<y`
`OR`	Або	`a>b OR x<y`

Пріоритет логічних операцій нижчий за операції порівняння.

Оператор IF

Оператор `IF` перевіряє умову і змінює хід виконання програми:

`IF` *`логічний_вираз`* `THEN` *`оператор(и)`* **[**`ELSE` *`оператор(и)`***]**

Якщо умова виконується, тобто *`логічний_вираз`* має значення `TRUE`, то виконуються *`оператор(и)`*, що йдуть за `THEN` та перед необов'язковим `ELSE`. Якщо ж *`логічний_вираз`* має значення `FALSE`, тоді виконуються *`оператор(и)`*, що йдуть за `ELSE`. Наприклад:

```
INPUT "Enter a number", x
IF x>100 THEN PRINT "Big number" ELSE PRINT "Small number"
```

Якщо необов'язковий `ELSE` не включено, продовжується виконання наступного рядку програми. Наприклад:

```
IF day > 29 AND month = 2 THEN PRINT "Error in date"
```

(Операція `AND` має нижчий пріоритет, ніж операції порівняння `">"` та `"="`, тому дужки не потрібні).

Оператор `IF` та пов'язані з ним оператори, включно з тими, що йдуть після `ELSE`, повинні знаходитися в одному й тому ж логічному рядку. Якщо ж кількість операторів більша, ніж може вмістити один рядок, альтернативою може бути використання блоку `IF`.

Блок IF

Блок `IF` створює серію перевірок:

```
IF логічний_вираз THEN
   .
   .  оператор(и)
   .
[ELSEIF логічний_вираз THEN
   .
   . оператор(и)]
   .
[ELSE
   .
   . оператор(и)]
   .
END IF
```

При виконанні блоку `IF` спочатку перевіряється істинність умови в рядку `IF`. Якщо цей *логічний_вираз* має значення `FALSE`, по порядку перевіряється кожний з наступних `ELSEIF` (яких може бути скільки завгодно). Як тільки при виконанні програми буде підтверджена істинність умови в одному з `ELSEIF`, будуть виконані *оператор(и)*, що йдуть після пов'язаного з ним `THEN`, і програма перестрибне на `END IF` без подальших перевірок. *оператор(и)* після необов'язкового `ELSE` виконуються, якщо жодна з попередніх перевірок не була успішною. Наприклад:

```
IF x < 0 THEN
   PRINT "Number is less then zero"
ELSEIF x > 0 THEN
   PRINT "Number is greated then zero"
ELSE
   PRINT "Number is 0"
END IF
```

Після ключового слова `THEN` в першому рядку блока `IF` немає нічого. За цією ознакою транслятор відрізняє блок `IF` від звичайного оператора `IF`. Ключове слово `ELSE` є єдиним в рядку. Кінець структурного блоку визначає оператор `END IF`. (Зауважте, що `END IF` має пробіл, а `ELSEIF` – ні.)

Блоки `IF` можуть бути вкладені, тобто після будь-якого `THEN` будь-де може міститися інший блок `IF`.

Оператори DO / LOOP

Оператори DO/LOOP утворюють блок і являють собою універсальний побудовник циклу з перевіркою на його початку чи в кінці:

```
DO [{WHILE|UNTIL} логічний_вираз]
   .
   . оператор(и) [EXIT LOOP]
   .
LOOP [{WHILE|UNTIL} логічний_вираз]
```

Для здійснення перевірки в циклі DO/LOOP використовуються ключові слова WHILE або UNTIL. Конструкція з WHILE означає «виконувати цикл доки …». Тобто, якщо *логічний_вираз* має значення TRUE, цикл буде повторюватись, а якщо FALSE – програма вийде з циклу і перейде на оператор, що слідує безпосередньо після LOOP. Конструкція з UNTIL означає «вийти з циклу як тільки …» і викликає протилежний ефект, тобто цикл буде завершено, якщо умова – TRUE і повторено у випадку FALSE. (Таким чином, конструкції ...WHILE *логічний_вираз* та ...UNTIL NOT *логічний_вираз* є синонімічними.)

Наприклад:

```
x = 1
DO WHILE x <= 1024
   PRINT "X="; x, "Square Root of X = "; SQR(x)
   x = x * 2
LOOP
```

В будь-якому місці блоку DO/LOOP в нього можна включити оператор EXIT LOOP. Він дає можливість примусово вийти з циклу.

Оператори FOR/NEXT

Оператори FOR/NEXT утворюють блок і визначають цикл з автоматичним додатнім чи від'ємним прирощенням:

```
FOR змінна = початок TO кінець [STEP крок]
   .
   . оператор(и) [EXIT FOR]
   .
NEXT змінна
```

Тут *змінна* – лічильник циклу, *початок* – початкове значення лічильника, *кінець* – його кінцеве значення, а *крок* – необов'язкове значення прирощення, яке, якщо воно не зазначене, за замовченням дорівнює 1.

Виконання операторів між FOR та NEXT повторюється. З кожним проходженням через цикл *змінна* набуває прирощення *крок*. Цикл завершується, коли *змінна* буде більше чи рівна *кінець* (або, для від'ємного *крок*, – менше або рівна *кінець*). Тіло циклу повністю пропускається, якщо *початок* > *кінець* при додатньому *крок* або ж, *початок* < *кінець* при від'ємному *крок*.

Цикли FOR/NEXT можуть бути поміщені всередині інших циклів FOR/NEXT.

Для примусового выходу з циклу FOR/NEXT до його завершення використовується оператор EXIT FOR.

Наприклад:

```
FOR x = 0 TO 10 STEP 0.5
   x3 = x^3
   IF x3 > 100 THEN EXIT FOR
   PRINT "X="; x, "Cube of X = "; x3
NEXT x
```

Оператор END

Оператор END (без аргументів) закінчує виконання програми. Оператори END можуть бути поміщені в будь-яке місце програми, і їх може бути більше одного. Якщо в програмі закінчуються оператори, це викликає той самий ефект.

Процедури

Опис та виклик процедури. Оператори SUB / END SUB та CALL

Деякому фрагменту програми можна надати ім'я і викликати його при потребі в різних місцях програми. Такий фрагмент називається процедурою, а оператори SUB та END SUB відділяють в тексті програми її опис:

```
SUB ім'я_процедури [(перелік_параметрів)]
    .
    .оператор(и) [EXIT SUB]
    .
END SUB
```

Тут *ім'я_процедури* – це унікальне ім'я, пов'язане з процедурою, *перелік_параметрів* – це взята в дужки необов'язкова послідовність формальних параметрів, розділених комами, що передаються процедурі при її виклику. Ці параметри використовуються лише для опису процедури, вони не мають ніякого відношення до інших змінних з тими ж самими іменами в програмі.

Виконання процедури, визначеної за допомогою оператора SUB, ініціюється оператором CALL :

```
CALL ім'я_процедури [(перелік_параметрів)]
```

Тут *перелік_параметрів* є переліком фактичних параметрів, які повинні бути узгодженими за типом і кількістю з формальними параметрами, зазначеними в операторі SUB.

Опис процедури завершується оператором END SUB, який повертає виконання програми на оператор, наступний за CALL. Щоб вийти з процедури не з її кінця, а з якого-небудь іншого місця, використовується оператор EXIT SUB.

Описи процедур не повинні бути вкладені, тобто не можна описати процедуру всередині іншої процедури (хоча опис процедури може містити виклики інших процедур).

Положення описів процедур всередині програми неважливе, транслятор знайде опис, де б він не знаходився.

Описи процедур є «невидимими» при виконанні програми, тобто програма не може випадково «потрапити» в опис процедури. Наприклад, при виконанні наступної програми

```
CALL PrintSomething

SUB PrintSomething
   PRINT "Printed from PrintSomething"
END SUB
```

повідомлення з'являється лише один раз.

Формальні та фактичні параметри

Змінні, що складають *перелік_параметрів* в операторі SUB при описі процедури, називаються формальними параметрами. Вони використовуються лише для опису процедури і повністю відділені від інших інших змінних з тими ж самими іменами в програмі.

Значення, що передаються процедурі при її виклику оператором CALL, називають фактичними параметрами.

Фактичні параметри, що передаються в процедуру, можуть бути як змінними, так і виразами (зокрема, константами). Фактичні параметри, величини яких визначаються в процедурі та з процедури повертаються назад, можуть бути лише змінними.

Наступний приклад демонструє виклик процедури CylVol, де всі фактичні параметри є змінними в головній програмі.

```
Diameter = 9.4 : R = Diameter/2 : height = 12.1
CALL CylVol(R, height, V)
PRINT "Volume of the cylinder is "; V
END

SUB CylVol(Radius, height, Volume)
'
' ------------------------------------------------------------
' Розрахунок об'єму циліндра
'
' Вхідні параметри:
'    Radius – радіус
'    height - висота
' Вихідний параметр:
'    Volume - об'єм циліндра
' ------------------------------------------------------------
'
   Volume = Radius * Radius * 3.14159 * height
END SUB
```

При виклику процедури перші два фактичних параметри могли би бути й виразами (зокрема, константами), наприклад:

```
CALL CylVol(Diameter/2, 12.1, V)
```

але третій фактичний параметр обов’язково повинен бути змінною, оскільки його значення передається назад у головну програму.

Передача параметрів за значенням чи за посиланням

Передача параметрів процедурі означає насправді передачу їй покажчиків, що визначають адреси ділянок пам'яті, де розміщуються фактичні значення параметрів. Далі в процедурі над цими ділянками пам'яті виконуються певні дії.

Якщо фактичний параметр є змінною, то результати всіх дії з нею стають доступними в програмі, що викликала процедуру. Наприклад, результатом виконання такої програми

```
x=1
PRINT "x="; x
CALL ZP(x)
PRINT "x="; x
END

SUB ZP(a)
'
' ------------------------------------------------------
' ілюстрація передачі параметрів за значенням/посиланням
' Вхідний та вихідний параметр:  a
' ------------------------------------------------------
'
   a=a*2
   PRINT "a="; a
END SUB
```

буде:

```
x= 1
a= 2
x= 2
```

Такий спосіб передачі параметра носить назву *передачі за посиланням*.

Таким чином, при передачі параметра за посиланням процедура може змінювати його значення і повертати інформацію програмі, що її викликає.

Якщо ж фактичний параметр є виразом (зокрема, константою), то спочатку обчислюється його значення, воно заноситься в тимчасову ділянку пам'яті, після чого в процедуру передається покажчик на адресу саме цієї тимчасової ділянки пам'яті. Після повернення в головну программу вміст цієї тимчасової ділянки пам'яті стає недоступним.

Такий спосіб передачі параметра носить назву *передачі за значенням*.

Наприклад, якщо в попередньому прикладі виклик процедури змінити на

```
CALL ZP( (x) )
```

то результатом виконання програми буде:

```
x= 1
a= 2
x= 1
```

Такий ефект пояснюється тим, що наявність дужок примушує транслятор аналізувати `(x)` як вираз (такий самий ефект спостерігався б, якщо фактичним параметром замість `(x)` зазначити `x+0` або `x*1`), і вміст ділянки пам'яті, в якій зберігається змінна `x`, при виконанні процедури `ZP` залишається недоторканим.

Передача масивів процедурам

Процедури дозволяють передавати цілі масиви як аргументи. При описі процедури в операторі `SUB` формальний параметр-масив позначається в *`перелік_параметрів`* приєднанням до ідентифікатора взятої в дужки числової константи. Ця величина показує *кількість вимірів* масива, але *не розмір масива*. Наприклад:

```
SUB CountZeros(a(1), size, count)
'
' лічильник count повертає кількість нульових елементів
' в одновимірному масиві a[size] розміром size елементів
'
    count = 0
    FOR i = 1 TO size
      IF a[i] = 0 THEN count = count + 1
    NEXT i
END SUB
```

При виклику процедури оператором `CALL` фактичні параметри-масиви розрізняються завдяки наявності порожніх дужок після ідентифікатора масива. Наприклад:

```
size = 55 : DIM payments[size]
...
CALL CountZeros(payments(), size, count)
PRINT "There are made "; size - count; " payments"
END
```

Масиви передаються в процедури тільки за посиланням, хоча окремі елементи масивів можна передавати й за значенням в складі «скалярних» виразів.

Рекомендована література

Основна

1. *Бахвалов Н. С., Жидков Н. П., Кобельков Г. М.* Численные методы. – М.: Наука, 1987. – 600 с.
2. *Боглаев Ю. П.* Вычислительная математика и программирование. – М.: Высш. школа, 1990. – 544 с.
3. *Васильков Ю. В., Василькова Н. Н.* Компьютерные технологии вычислений в математическом моделировании. – М.: Финансы и статистика, 1999. – 255 с.
4. *Гловацкая А. П.* Методы и алгоритмы вычислительной математики. – М.: Радио и связь, 1999. – 406 с.
5. *Демидович Б. П., Марон И. А.* Основы вычислительной математики. – М.: Наука, 1965. – 665 с.
6. *Каханер Д., Моулер К., Нэш С.* Численные методы и программное обеспечение. – М.: Мир, 2001. – 576 с.
7. *Мак-Кракен Д., Дорн У.* Численные методы и программирование на ФОРТРАНЕ. – М.: Мир, 1977. – 293 с.
8. *Ортега Дж., Пул У.* Введение в численные методы решения дифференциальных уравнений. – М.: Наука, 1986. – 288 с.
9. *Пантина И. В., Синчуков А. В.* Вычислительная математика. – М.: Московский финансово-промышленный институт «Синергия», 2012. – 176 с.
10. *Прокопенко Ю. В., Татарчук Д. Д., Казиміренко В. А.* Обчислювальна математика. – К.: Політехніка, 2003. – 120 с.
11. *Рябенький В. С.* Введение в вычислительную математику – М.: Физматлит, 2000. – 296 с.
12. *Форсайт Дж., Малькольм М., Моулер К.* Машинные методы математических вычислений. – М.: Мир, 1980. – 279 с.
13. *Шуп Т. Е.* Решение инженерных задач на ЭВМ. – М.: Мир, 1990. – 235 с.

Додаткова

14. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 536 с.
15. *Бабенко К. И.* Основы численного анализа. – М.: Наука, 1986. – 744 с.
16. *Банди Б.* Методы оптимизации. Вводный курс. – М.: Радио и связь, 1988. – 128 с.
17. *Березин И. С., Жидков Н. П.* Методы вычислений. В 2 тт. – М.: Физматиздат. Т.1, 1966. – 632 с. Т.2, 1962. – 639 с.
18. *Вержбицкий В. М.* Численные методы. Линейная алгебра и нелинейные уравнения. – М.:Высш. школа, 2000. – 266 с.
19. *Вержбицкий В. М.* Численные методы. Математический анализ и обыкновенные дифференциальные уравнения. – М.:Высш. школа, 2001. – 382 с.
20. *Волков А. Е.* Численные методы. – М.: Наука, 1982. – 248 с.
21. *Воробьев А. Ф., Данелян Т. Я., Данилина Н. И.* Вычислительная математика. – М.: Статистика, 1966. – 164 с.
22. *Воробьева Г. Н., Данилова А. Н.* Практикум по вычислительной математике. – М.: Высш. шк., 1990. – 308 с.
23. *Данциг Дж. Б.* Линейное программирование, его применения и обобщения. – М.: Прогресс, 1966. – 600 с.
24. *Демидович Б. П., Марон И. А., Шувалова Э. З.* Численные методы анализа. – М.: Наука, 1962. – 367 с.
25. *Денис Дж., Шнабель Р.* Численные методы безусловной оптимизации и решения нелинейных уравнений. – М.: Мир, 1988. – 249 с.
26. *Ермаков С. М., Михайлов Г. А.* Курс статистического моделирования. – М.: Наука, 1976. – 320 с.
27. *Заварыкин В. М., Житомирский В. Г., Лапчик М. П.* Численные методы. – М.: Просвещение, 1991. – 176 с.
28. *Иванов В. В.* Методы вычислений на ЭВМ: Справочное пособие.- К.: Наук. Думка, 1986. – 584 с.
29. *Ильина В. А., Силаев П. К.* Численные методы для физиков-теоретиков. В 2 тт. – М. – Ижевск: Институт компьютерных исследований. Т.1, 2003. – 132 с. Т.2, 2004. – 118 с.

30. *Калиткин Н. Н.* Численные методы. – М.: Наука, 1978. – 500 с.

31. *Киреев В. И., Пантелеев А. В.* Численные методы в примерах и задачах — М.: Высш. школа, 2008. – 480 с.

32. *Копченова Н. В., Марон И. А.* Вычислительная математика в примерах и задачах. – М.: Наука, 1972. – 368 с.

33. *Крылов В. И., Бобков В. Б., Монастырный П. И.* Вычислительные методы. В 2 тт. – М.: Наука. Т.1, 1976. – 304 с. Т.2, 1977. – 400 с.

34. *Марчук Г. И.* Методы вычислительной математики. – М.: Наука, 1989. – 608 с.

35. *Муха В. С.* Вычислительные методы и компьютерная алгебра. – Минск: Бел. гос. ун-т информатики и радиоэлектроники, 2006. – 127 с.

36. *Мэтьюз Дж. Д., Финк К. Д.* Численные методы. Использование Matlab. – М.: Издат. дом «Вильямс», 2001. – 720 с.

37. *Мысовских И. П.* Лекции по методам вычислений. – СПб.: Изд-во СПбГУ, 1998. – 472 с.

38. *Самарский А. А., Гулин А. В.* Численные методы. – М.: Наука, 1989. – 432 с.

39. *Турчак Л. И., Плотников П. В.* Основы численных методов. – М.: Физматлит, 2003. – 304 с.

40. *Федоренко Р. П.* Введение в вычислительную физику. – М.: изд-во МФТИ, 1994. – 528 с.

41. *Форсайт Дж., Моулер К.* Численное решение систем линейных алгебраических уравнений. – М.: Мир, 1969. – 167 с.

42. *Хемминг Р. В.* Численные методы для научных работников и инженеров. – М.: Наука, 1972. – 399 с.

43. *Шноль Э. Э.* Семь лекций по вычислительной математике. – М.: Едиториал УРСС, 2004. – 112 с.