МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
до лабораторних робіт з дисципліни
«АЛГОРИТМИ ТА МЕТОДИ ОБЧИСЛЕНЬ»
для здобувачів освітнього ступеня бакалавра
спеціальності 123 «Комп'ютерна інженерія»
усіх форм навчання

Черкаси 2018

УДК 519.6(07)
М54

Затверджено вченою радою ФІТІС протокол № 11 від 15.05.2018 р., згідно з рішенням кафедри прикладної математики, протокол №10 від 18.04.2018 р.

Упорядник Мірошкіна І. В., к.т.н., доцент

Рецензент Щерба А.І., , к.ф.-м.н., доцент

М54 **Методичні** рекомендації до лабораторних робіт з дисципліни «Алгоритми та методи обчислень» для здобувачів освітнього ступеня бакалавра спеціальності 123 «Комп'ютерна інженерія» усіх форм навчання [Електронний ресурс] / [Упоряд.: І.В. Мірошкіна]; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси: ЧДТУ, 2018. – 106 с. – Назва з титульного екрана.

Викладено навчальні (навчально-методичного характеру) матеріали, наведено мету, теоретичні відомості, завдання та послідовність виконання лабораторних робіт з дисципліни «Алгоритми та методи обчислень», подано вимоги щодо їх виконання, контрольні запитання та перелік рекомендованої літератури.

Для студентів спеціальності 123 «Комп'ютерна інженерія» усіх форм навчання.

УДК 519.6(07)

Виробничо-практичне
електронне видання
комбінованого використовування

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
до лабораторних робіт з дисципліни
«АЛГОРИТМИ ТА МЕТОДИ ОБЧИСЛЕНЬ»
для здобувачів освітнього ступеня бакалавра
спеціальності 123 «Комп'ютерна інженерія»
усіх форм навчання

Упорядник **Мірошкіна** Ірина Володимирівна

В авторській редакції

ЗМІСТ

ВСТУП

Дисципліна «Алгоритми та методи обчислень» входить до циклу базових дисциплін математичної та природничо-наукової підготовки бакалаврів спеціальності 123 «Комп'ютерна інженерія».

Метою викладання даної дисципліни є отримання студентами знань з області проектування та розробки алгоритмів взагалі, методів обчислень і побудови алгоритмів для наближених обчислень. Оволодіння такими знаннями дозволить реалізовувати різні задачі моделювання за допомогою комп'ютерної техніки. Такі знання майбутній спеціаліст зможе застосовувати як при подальшому навчанні, так і після отримання вищої освіти у своїй професійній діяльності.

Окрім теоретичних знань, навчальною програмою дисципліни «Алгоритми та методи обчислень» передбачається формування у студентів практичних навичок при розв'язанні задач основних розділів дисципліни, таких як аналіз ефективності алгоритмів, побудова алгоритмів, розв'язування систем лінійних алгебраїчних рівнянь, розв'язування нелінійних рівнянь, методи наближення функцій, чисельне інтегрування, методи розв'язування звичайних диференціальних рівнянь.

Методичні рекомендації містять необхідні теоретичні відомості до кожної лабораторної роботи, завдання та порядок виконання, контрольні запитання та список рекомендованої літератури для підготовки до захисту лабораторної роботи. Кожний студент має свій варіант задачі, який повинен співпадати з його порядковим номером у списку групи.

Викладач чи інший повноважний співробітник кафедри на першому лабораторному занятті ознайомлює студентів із правилами поведінки у комп'ютерному класі та проводить інструктаж з техніки безпеки.

По виконанню лабораторної роботи студент оформлює звіт, в якому має бути наведено:

- формулювання індивідуального завдання;
- хід виконання роботи (лістинг розробленої програми, розрахунки);
- отримані результати;
- аналіз результатів;
- висновки.

ЛАБОРАТОРНА РОБОТА № 1
ЧАСОВА ЕФЕКТИВНОСТЬ АЛГОРИТМІВ

Мета: дослідити залежність часу виконання алгоритму сортування одновимірного масиву методом включень від кількості елементів масиву.

Теоретичні відомості

Алгоритм – це скінченна послідовність детермінованих арифметичних і логічних дій над вихідними та проміжними даними задачі обробки інформації, виконання яких призводить до її правильного розв'язку (тобто отриманню вихідних даних).

Аналіз алгоритмів передбачає доведення правильності (коректності) алгоритму та визначення його часових характеристик (як швидко алгоритм розв'язує задачу).

Під правильністю алгоритму розуміють те, що він правильно розв'язує задачу. Доведення правильності можливе:

- теоретичне (не завжди є простим, вимагає логічних висновків);
- експериментальне (прогін програми на різних тестах, розв'язки для яких отримано заздалегідь ручним способом, тобто тестування. Тести повинні охоплювати всі гілки алгоритму!).

Для оцінки часових характеристик теж застосовують теоретичний і експериментальний аналіз. Час розв'язання задачі тісно пов'язаний з об'ємом даних на вході. Тому функція часу алгоритму залежить від кількості вхідної інформації.

Для отримання експериментальних оцінок підбирають ряд задач, які відрізняються одна від одної кількістю вхідних даних. Для кожної кількості визначається статистика процесорного часу – отримується дискретна функція, яку за необхідністю апроксимують неперервною.

Час виконання алгоритму істотно залежить від якості програмування, характеристик комп'ютера. Тому для оцінки часової ефективності алгоритму не використовують поняття часу в явному вигляді. Замість нього розглядають сумарну кількість основних операцій: додавання, множення, порівняння, тощо. Щоб підвищити об'єктивність оцінки алгоритмів, учені, які працюють в галузі комп'ютерних наук, прийняли *асимптотичну часову складність* як основну міру ефективності виконання алгоритму.

Часто говорять, що час виконання алгоритму має порядок $T(n)$ від вхідних даних розміру n. Одиниця вимірювання $T(n)$ точно не визначена, але в більшості випадків під нею розуміють кількість інструкцій, які виконуються на ідеалізованому комп'ютері.

Для багатьох алгоритмів час виконання дійсно є функцією вхідних даних, а не їх розміру. У цьому випадку визначають $T(n)$ як час виконання в найгіршому випадку, тобто, як максимум часів виконання за всіма вхідними даними розміру n. Поряд з тим розглядають $T_{ср}(n)$ як середній (в статистичному розумінні) час виконання за всіма вхідними даними розміру n. Хоча $T_{ср}(n)$ є достатньо об'єктивною мірою виконання, але часто неможливо передбачити,

або обґрунтувати, рівнозначність усіх вхідних даних. На практиці середній час виконання знайти складніше, ніж найгірший час виконання, так як математично це зробити важко і, крім цього, часто не буває простого визначення поняття «середніх» вхідних даних. Тому, в основному, користуються найгіршим часом виконання як мірою часової складності алгоритмів. Аналогічно поняттю найгіршого часу, визначається поняття найкращого часу.

Функція складності алгоритму в деяких випадках може бути визначена точно у вигляді рекурентних співвідношень або сум чи добутків числових рядів, однак при збільшенні розміру задачі внесок постійних множників і доданків нижчих порядків, що фігурують у цих виразах стає вкрай незначним і у таких випадках для порівняння складності алгоритмів використовують так званий *асимптотичний аналіз*.

Мета асимптотичного аналізу полягає у визначенні гарного наближення точного значення складності алгоритмів, достатнього у багатьох випадках для порівняння їх величин не за точними, а за наближеними значеннями. Така асимптотична складність алгоритмів використовується також для оцінки розмірів задач, які можуть бути розв'язані за допомогою даного алгоритму. Для запису наближених формул складності використовуються асимптотичні позначення, зокрема «*O*» (О велике).

Кажуть, що $f(n) = O(g(n))$ якщо існують цілі C та n_0, такі що $|f(n)| \le C \cdot g(n)$ для усіх $n \ge n_0$.

В асимптотичних позначеннях для складності $f(n)$ використовуються такі назви:

$f(n)=O(\log_2 n)$ – логарифмічна;

$f(n)=O([\log_2 n] \cdot c)$ – полілогарифмічна складність («[°]» взяття цілої частини);

$f(n)=O(n)$ – лінійна складність;

$f(n)=O(n \cdot \log_2 n)$ – лінеаритмічна складність;

$f(n)=O(n^2)$ – квадратична складність;

$f(n)=O(n^c)$ – поліноміальна складність;

$f(n)=O(c^n)$ – експоненціальна складність;

$f(n)=O(n!)$ – факторіальна складність.

Оцінка з точністю до порядку дає верхню межу складності алгоритму. Те, що алгоритм має певний порядок складності, не означає, що алгоритм буде дійсно виконуватися так довго. При певних вхідних даних, багато алгоритмів виконується набагато швидше, ніж можна припустити на підставі їхнього порядку складності.

Аналіз часу виконання алгоритмів. Час виконання алгоритму можна оцінювати шляхом підрахунку кількості «базових операцій», що виконуються, - як функцію від розміру вхідних даних.

Приймаються наступні припущення про гіпотетичну обчислювальну машину (виконавця):

- для виконання «простої» операції (+, -, *, /, =, *if, call*) потрібно один часовий крок (такт обчислювальної машини);

- звертання до пам'яті (читання, запис) виконуються за один часовий крок;
- цикли та підпрограми складаються із послідовності простих операцій.

Розглянемо достатньо простий алгоритм сортування – сортування методом включення. Нагадаємо формулювання проблеми: маємо послідовність n чисел $\langle a_1, a_2, \ldots, a_n \rangle$. Необхідно виконати перестановку чисел $\langle a'_1, a'_2, \ldots, a'_n \rangle$ таким чином, щоб для них виконувалося співвідношення $a'_1 \le a'_2 \le \ldots \le a'_n$.

Алгоритм сортування включенням ефективно працює при сортуванні невеликої кількості елементів. Нижче наведений псевдокод методу включень під назвою *Insertion_sort*. На його вхід подається масив $A[1 \ldots n]$, який містить послідовність n чисел для сортування (кількість елементів тут позначена як *length*[*A*]). Вхідні числа сортуються без використання додаткової пам'яті: їх перестановка відбувається всередині масиву, а об'єм використаної для цього додаткової пам'яті не перевищує деяку сталу величину. По закінченню роботи алгоритму *Insertion_sort* вхідний масив міститиме відсортовану послідовність.

```
     Insertion_sort(A)
1  for j ← 2 to length[A]
2     do key ← A[j]
3          // Включення елементу A[j]  у відсортовану послідовність A[1..j-1]
4          i ← j-1
5          while i>0 and A[i]>key
6                    do A[i+1] ← A[i]
7                       i ← i-1
8          A[i+1] ← key
```

Будемо вважати, що для виконання кожного рядку псевдокоду потрібно фіксований час. Час виконання різних рядків може відрізнятись, але ми вважаємо, що один й той самий рядок *i* виконується за час c_i, де c_i – стала.

Почнемо з того, що введемо для процедури *Insertion_sort* час виконання кожної інструкції і кількість їх повторень. Для кожного $j = 2, 3, \ldots, n$, де n = *length*[*A*], позначимо через t_j кількість перевірок умови в циклі *while* (рядок 5). При нормальному закінченні циклів *for* та *while* (тобто коли перестає виконуватись умова циклу) умова перевіряється на один раз більше, аніж виконується тіло циклу. Звісно, коментарі не є виконуваними інструкціями, а тому не збільшують час роботи алгоритму.

Insertion_sort(A)	Час	Кількість повторень
1 **for** j ← 2 to length[A]	c_1	n
2 **do** key ← A[j]	c_2	n-1
3 // Включення елементу A[j]	0	n-1
4 i ← j-1	c_4	n-1
5 **while** i>0 and A[i]>key	c_5	$\sum_{j=2}^{n} t_j$
6 **do** A[i+1] ← A[i]	c_6	$\sum_{j=2}^{n}(t_j - 1)$
7 i ← i-1	c_7	$\sum_{j=2}^{n}(t_j - 1)$
8 A[i+1] ← key	c_8	n-1

Час роботи алгоритму – це сума проміжків часу, необхідних для виконання кожної інструкції, яка входить до його складу. Якщо виконання інструкції триває протягом часу c_i і вона повторюється в алгоритмі n разів, то її внесок в повний час роботи алгоритму становитиме $c_i \cdot n$. Позначимо через $T(n)$ час роботи алгоритму *Insertion_sort*:

$$T(n) = c_1 \cdot n + c_2 \cdot (n-1) + c_4 \cdot (n-1) + c_5 \cdot \sum_{j=2}^{n} t_j + c_6 \cdot \sum_{j=2}^{n} (t_j - 1) + \\ + c_7 \cdot \sum_{j=2}^{n} (t_j - 1) + c_8 \cdot (n-1).$$

Навіть якщо розмір вхідних даних є фіксованим, час роботи алгоритму може залежати від степені впорядкованості величин для сортування. Наприклад, найбільш сприятливий випадок для алгоритму *Insertion_sort* – це коли всі елементи масиву вже відсортовані. Тоді для кожного j=2,3,…,n буде виконуватись, що $A[i] \le key$ у рядку 5, ще коли i дорівнює своєму початковому значенню j–1. Таким чином, при j=2,3,…,n $t_j = 1$, і час роботи алгоритму в найбільш сприятливому випадку обчислюється так:

$$T(n) = c_1 \cdot n + c_2 \cdot (n-1) + c_4 \cdot (n-1) + c_5 \cdot (\mathrm{n}-1) + c_8 \cdot (\mathrm{n}-1) = \\ = n \cdot (c_1 + c_2 + c_4 + c_5 + c_8) - (c_2 + c_4 + c_5 + c_8) = a \cdot n + b,$$

де a та b – сталі, які залежать від величин c_i; тобто цей час є *лінійною функцією* від n.

Якщо елементи масиву відсортовані в порядку, який є зворотнім до потрібного (в даному випадку в порядку зменшення), то це найгірший випадок. Кожний елемент $A[j]$ необхідно порівнювати з усіма елементами вже відсортованої підмножини $A[1..j-1]$, так що для j=2,3,…,n значення $t_j = j$. З урахуванням того, що

$$\sum_{j=2}^{n} t_j = \frac{n(n+1)}{2} - 1, \qquad \text{та} \qquad \sum_{j=2}^{n} (t_j - 1) = \frac{n(n-1)}{2},$$

(за правилом суми арифметичної прогресії), отримуємо, що час роботи алгоритму *Insertion_sort* в найгіршому випадку визначається співвідношенням

$$T(n) = c_1 \cdot n + c_2 \cdot (n-1) + c_4 \cdot (n-1) + c_5 \cdot \left(\frac{n(n+1)}{2} - 1\right) + c_6 \cdot \frac{n(n-1)}{2} + \\ + c_7 \cdot \frac{n(n-1)}{2} + c_8 \cdot (n-1) = n^2 \cdot \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) + \\ + n \cdot \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right) - (c_2 + c_4 + c_5 + c_8) = a \cdot n^2 + b \cdot n + c,$$

де a, b, c залежать від c_i. Таким чином, це *квадратична функція* від n.

Було отримано результати аналізу щодо часу роботи алгоритму для сортування методом включення для найгіршого та найкращого випадку вхідних даних. У аналізі алгоритмів зазвичай досліджують час роботи алгоритму тільки у найгіршому випадку (тобто максимального часу роботи серед усіх вхідних даних розміром n) і на це є наступні причини.

По-перше, час роботи алгоритму в найгіршому випадку – це верхня межа цієї величини для будь-яких вхідних даних. Маючи це значення, можна точно зазначити, що для виконання алгоритму не потрібно буде більше часу.

По-друге, в деяких алгоритмах найгірший випадок зустрічається досить часто. Наприклад, якщо в базі даних відбувається пошук інформації, то найгіршому випадку відповідає ситуація, коли потрібна інформація в базі даних відсутня.

По-третє, характер поведінки «середнього» часу роботи алгоритму часто нічим не кращий поведінки часу роботи алгоритму для найгіршого випадку. Припустимо, що послідовність, для якої застосовується сортування методом включень, сформована випадковим чином. Скільки часу буде потрібно, щоби визначити, в яке місце підмасиву $A[1..j-1]$ потрібно помістити елемент $A[j]$? В середньому половина елементів підмасиву $A[1..j-1]$ менше ніж $A[j]$, а половина – більше його. Таким чином, в середньому потрібно перевірити половину елементів цього підмасиву, тому t_j приблизно дорівнює $j/2$. В результаті отримується, що середній час роботи алгоритму є квадратичною функцією від кількості вхідних елементів, тобто характер цієї залежності такий самий, що й для часу роботи в найгіршому випадку.

В деяких часткових випадках може бути цікавим середній час роботи алгоритму, тобто його математичне сподівання. Але при аналізі середнього часу роботи виникає одна проблема, яка полягає в тому, що не завжди очевидно, які вхідні дані для даної задачі будуть «середніми». Часто робиться припущення, що всі набори вхідних даних одного й того самого об'єму зустрічаються з однаковою ймовірністю. На практиці це припущення може не зберігатись, але тоді можна застосовувати *рандомізовані алгоритми*, в яких використовується випадковий вибір, і це дозволяє провести ймовірнісний аналіз.

Для полегшення аналізу процедури *Insertion_sort* були зроблені деякі припущення. По-перше, ми ігнорували фактичний час виконання кожної інструкції, представив цю величину у вигляді деякої константи c_i. Далі ми побачили, що облік всіх цих констант дає зайву інформацію: час роботи алгоритму в найгіршому випадку виражається формулою $an^2 + bn + c$, де a, b, c – деякі сталі, які залежать від вартостей c_i. Таким чином, ми ігноруємо не тільки фактичні вартості команд, але й їх абстрактні вартості c_i.

Зазвичай один алгоритм вважається ефективнішим за інший, якщо його час роботи в найгіршому випадку має більш низький порядок зростання. Через наявність сталих множників та другорядних членів ця оцінка може бути помилковою, якщо вхідні дані невеликі. Але якщо об'єм вхідних даних значний, то, наприклад, алгоритм $O(n^2)$ в найгіршому випадку працює швидше за алгоритм $O(n^3)$. Таким чином, можна зазначити, що швидким алгоритмом є такий алгоритм, для якого час роботи у найгіршому випадку зростає повільно по відношенню до зростання вхідних даних.

Завдання

1. Скласти процедуру сортування одновимірного масиву методом включень.
2. Скласти програму визначення часу виконання процедури сортування при різній вимірності масиву (n=100, 300, 500, 1000, 1500, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 15000,20000, 25000, 30000, 35000, 40000, 50000, 60000, 70000, 80000, 90000,100000). Передбачити заповнення масиву випадковими числами у діапазоні від 1 до n включно. Вимірювання часу для одного масиву проводити п'ять разів (передбачити, щоб масив не змінювався!). Остаточний час визначати як середній.

3. Побудувати графік залежності *T(n)* за отриманими результатами та виконати асимптотичний аналіз.
4. Для масиву найбільшої вимірності (*n*=100000) визначити мінімальний та максимальний час виконання.

Послідовність виконання лабораторної роботи

Розробимо процедуру сортування масиву методом включень. Формальні параметри процедури: масив, що упорядковується *B* вимірності *m*. У заголовку процедури у переліку параметрів перед описом масиву немає службового слова *var* тому, що нам не потрібно передавати упорядкований масив в головну програму.

```
procedure Insertion_Sort(m: integer; B: array[1..100000] of integer);
    var  i, j, key: integer;
begin
        for  j := 2 to m do
            begin
                    key := B[j];
                    i := j - 1;
                    while((i > 0) and (B[i] > key)) do
                        begin
                                B[i + 1] := B[i];
                                i := i - 1;
                        end;
                    B[i + 1] := key;
             end;
end;
```

Фрагмент програми заповнення масиву випадковими числами, визначення середнього часу сортування. Час визначаємо за допомогою стандартної функції *milliseconds*.

```
…….
type    mas = array[1..100000] of integer;
        tn = array [1..25]of integer;
        tm = array [1..25] of real;
var      n: tn; A: mas; t: tm;
        tp, tk, i, k: integer;
……..
begin

……..
    randomize;
    {Задаємо різні вимірності масивів}
    n[1]:=100; n[2]:=300; n[3]:=500; n[4]:=1000; n[5]:=1500; n[6]:=2000; n[7]:=3000;
    n[8]:=4000; n[9]:=5000; n[10]:=6000; n[11]:=7000; n[12]:=8000; n[13]:=9000;
    n[14]:=10000; n[15]:=15000; n[16]:=20000; n[17]:=25000; n[18]:=30000; n[19]:=40000;
    n[20]:=50000; n[21]:=60000; n[22]:=70000; n[23]:=80000; n[24]:=90000; n[25]:=100000;
    {Організовуємо головний цикл програми}
```

```
for k := 1 to 25 do
    begin
        {Заповнюємо масив A випадковими числами}
        for i := 1 to n[k] do A[i] := random(n[k]) + 1;
        t[k] := 0;
        for i := 1 to 5 do
            begin
                tp := milliseconds;   {Вимірюємо час до сортування}
                Insertion_Sort(n[k], A);
                tk := milliseconds;   {Вимірюємо час після сортування}
                t[k] := t[k] + (tk - tp);   {Визначаємо різницю}
            end;
        t[k] := t[k]/5;   {Визначаємо середній час}
    end;
  …….
end.
```

Подібно визначаємо найбільший та найменший час сортування масиву вимірності n=100000. Заповнюємо його числами від 100000 до 1 – для визначення найбільшого часу, і числами від1 до 100000 – для найменшого часу.

Результати обчислень приведено на рис. 1.1.

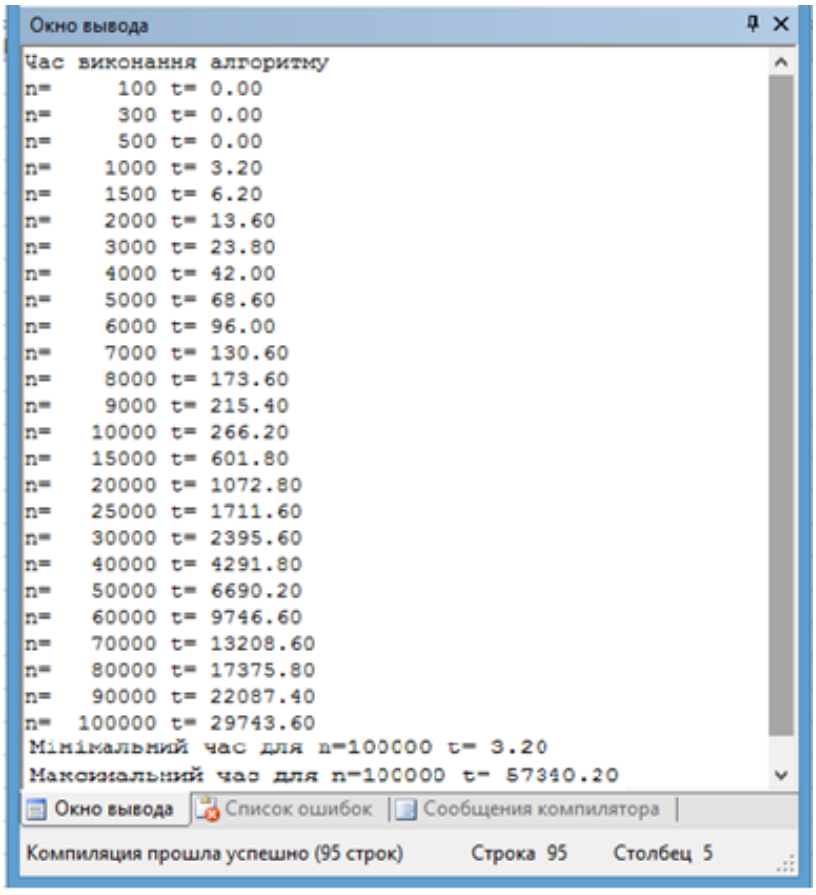


Рисунок 1.1 - Результати роботи програми

Побудуємо графік залежності $T(n)$ та верхню й нижню асимптоти (рис.1.2). Відомо, що алгоритм сортування за методом включень має квадратичну

складність $O(n^2)$. Тому для побудови асимптот нам необхідно лише підібрати відповідні коефіцієнти:

$$C_2 n^2 \leq T(n) \leq C_1 n^2, \text{ при } n \geq n_0.$$

За отриманими результатами маємо:

$$2 \cdot 10^{-6} \cdot n^2 \leq T(n) \leq 3{,}5 \cdot 10^{-6} \cdot n^2, \text{ при } n \geq 1000.$$

n	T	C1·n²	C2·n²
100	0,00	0,035	0,02
300	0,00	0,315	0,18
500	0,00	0,875	0,5
1000	3,20	3,5	2
1500	6,20	7,875	4,5
2000	13,60	14	8
3000	23,80	31,5	18
4000	42,00	56	32
5000	68,60	87,5	50
6000	96,00	126	72
7000	130,60	171,5	98
8000	173,60	224	128
9000	215,40	283,5	162
10000	266,20	350	200
15000	601,80	787,5	450
20000	1072,80	1400	800
25000	1711,60	2187,5	1250
30000	2395,60	3150	1800
40000	4291,80	5600	3200
50000	6690,20	8750	5000
60000	9746,60	12600	7200
70000	13208,60	17150	9800
80000	17375,80	22400	12800
90000	22087,40	28350	16200
100000	29743,60	35000	20000

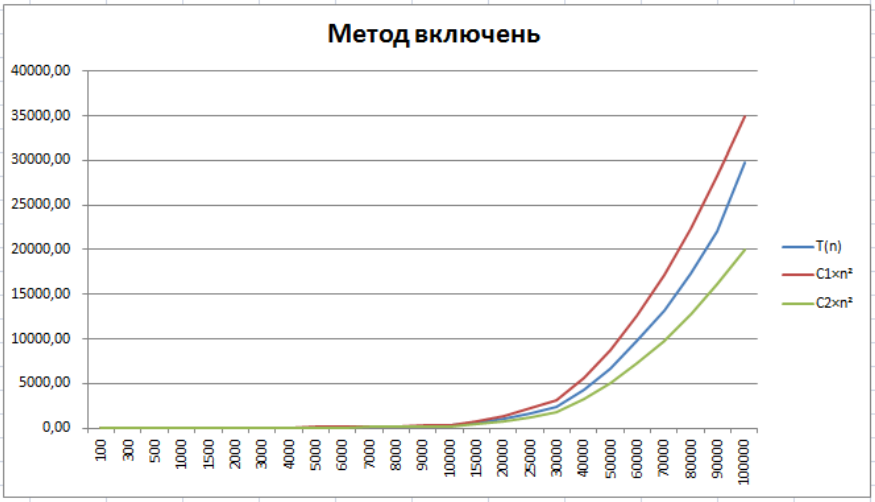


Рисунок 1.2 – Асимптотична оцінка алгоритму метода включень

Контрольні запитання

1. Дайте визначення поняття алгоритму.
2. В чому полягає асимптотичний аналіз алгоритмів.
3. Дайте визначення функції складності алгоритму.
4. Вкажіть правила для визначення функції складності.
5. Які види функції складності існують?
6. Яким чином визначається часова функція складності?
7. Назвіть способи аналізу функції складності за програмою.
8. Як працюють оператори циклу?
9. У якому випадку зручно використовувати оператор циклу *for*?

Перелік рекомендованої літератури

1. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест и др. – М. : ИД "Вильямс", 2011. – 1296 с. : ил.
2. Ахо А. Структуры данных и алгоритмы : учебн. пособ. / А. Ахо, Д. Хопкрофт, Д. Ульман ; пер. с англ. – М. : ИД "Вильямс", 2000. – 384 с.

ЛАБОРАТОРНА РОБОТА № 2
АНАЛІЗ ЧАСОВОЇ ЕФЕКТИВНОСТІ АЛГОРИТМІВ СОРТУВАННЯ

Мета: дослідити залежність часу виконання алгоритмів сортування одновимірного масиву методом бульбашковим, Шейкера та швидким від кількості елементів масиву.

Теоретичні відомості

Бульбашкове сортування. За методом бульбашкового сортування елементи масиву, що розміщені поряд, міняються місцями до тих пір, доки в масиві існують пари, для яких виконується умова сортування (тобто перший елемент більше другого у випадку сортування за зростанням). Щоб спростити пошук таких пар, масив переглядається в циклі від початку до кінця. В результаті найбільший елемент при кожному проході масиву начебто «випливає» нагору, звідки і назва алгоритму.

```
Bubble_sort(A)
   for i ← 1 to length(A)
      do for j ← 1 to (length(A)-i)
           do if A[j]>A[j+1]
                then  buf ←A[j]
                      A[j] ←A[j+1]
                      A[j+1] ←buf
```

Алгоритм має квадратичну асимптотичну складність $T(n)=O(n^2)$. Кількість перевірок умов перестановок в цьому методі значно перевищує кількість самих перестановок.

Сортування методом Шейкера. За один перегляд масиву з усіх його елементів обирається мінімальний та максимальний. Потім мінімальний елемент розміщується в початок, а максимальний – у кінець масиву. Після чого алгоритм виконується для інших даних (тобто для елементів, що розміщені між мінімальним та максимальним). За кожний прохід на свої місця стають одразу два елементи. Тому загальна кількість проходів буде вдвічі менше ніж в бульбашковому сортуванні – $^{n}/_{2}$.

```
Shaker_sort(A)
   for i ← 1 to (length(A) div 2)
      do  if A[i]>A[i+1]
                then min ← i+1
                     max ← i
                else min ← i
                     max ← i+1
          for j ← i+2 to (length(A)-i+1)
             do if A[j]>A[max]
                      then max ← j
                      else if A[j]<A[min]
                                then min ← j
          buf ←A[i]
```

```
            A[i] ←A[min]
            A[min] ←buf
            if max=i then max:=min
            buf ←A[length(A)-i+1]
            A[length(A)-i+1] ←A[max]
            A[max] ←buf
```

Алгоритм має теж квадратичну асимптотичну складність $T(n)=O(n^2)$.

Швидке сортування. Масив розбивається на дві частини, кожна з яких потім сортується окремо. Розбиття здійснюється відносно деякого елементу масиву. Вважається, що в першій частині всі числа менші або рівні цьому елементу, а в другій – більші або рівні. Кожна з частин переглядається з метою пошуку елементів, що потрапили не в свою групу. Якщо такі елементи знайдені, то їх міняють місцями.

Коли обидві частини упорядковані, кожна з них в свою чергу розбивається на дві частини і для кожної з них повторюється наведений алгоритм. Таке розбиття виконується до тих пір, доки одна з частин не буде зведена до одного елемента.

```
Quickly_sort(A)
   Sort(First, Last) //First, Last – номера першого та
                     //останнього елемента частини масиву, що сортується
      if First < Last //масив складається більш ніж з одного ел-та
           then Mid← A[random(Last-First)+First]
                //Випадково обираємо елемент відносно якого
                //буде здійснене розбиття масиву
                i← First-1
                j← Last+1
                //Пошук і перестановка елементів в двох частинах
                //масиву до тих пір, доки не виникне перетин в
                //точці розбиття
                while i<j
                   do repeat inc(i) until A[i]>=Mid
                      repeat dec(j) until A[j]<=Mid
                      if i<j
                             then
                                  buf ←A[i]
                                  A[i] ←A[j]
                                  A[j] ←buf
      Sort(First,j)  //Сортування першої частини
      Sort(j+1, Last)  //Сортування другої частини
Sort(1, length(A))
```

Процедура `Sort` – рекурсивна. Алгоритм має лінеаритмічну асимптотичну складність $T(n)=O(n \cdot \log n)$.

Завдання

1. Доповнити програму лабораторної роботи №1 процедурами сортування за методами бульбашковим, Шейкера, швидким.

2. Отримати час сортування масивів різної вимірності для вказаних методів.
3. Побудувати графіки залежності *T(n)* за отриманими результатами та виконати асимптотичний аналіз для кожного методу.
4. Для масиву найбільшої вимірності (*n*=100000) визначити мінімальний та максимальний час виконання.
5. Побудувати на одному графіку всі отримані часові залежності для методів включень, бульбашкового, Шейкера та швидкого.

Послідовність виконання лабораторної роботи

Розробимо процедури сортування масиву бульбашковим, Шейкера та швидким методами. Формальні параметри процедур: масив, що упорядковується *B* вимірності *m*. У заголовку процедур у переліку параметрів перед описом масиву немає службового слова *var* тому, що нам не потрібно передавати упорядкований масив в головну програму.

```
procedure Bubble_Sort(m: integer; B: mas);
var i, j, buf: integer;
begin
      for i := 1 to m do
              for j :=1 to (m-i) do
                      if B[j]>B[j+1] then
                                              begin
                                                      buf:=B[j];
                                                      B[j]:=B[j+1];
                                                      B[j+1]:=buf;
                                              end;
end;

procedure Shaker_sort(m: integer; B: mas);
var i, j, buf, min, max: integer;
begin
       for i :=1 to (m div 2) do
            begin
                  if B[i]>B[i+1]  then
                                      begin
                                              min :=i+1;
                                              max :=i;
                                      end
                                  else
                                      begin
                                              min :=i;
                                              max :=i+1;
                                      end;
                  for j :=i+2 to (m-i+1) do
                          if B[j]>B[max] then max:=j
                                       else if B[j]<B[min] then min:=j;
```

```
                buf:=B[i];
                B[i]:=B[min];
                B[min]:=buf;
                if max=i then max:=min;
                buf :=B[m-i+1];
                B[m-i+1]:=B[max];
                B[max]:=buf;
            end;
    end;

    procedure Quickly_sort(m: integer; B: mas);
        procedure Sort(First, Last: integer);
        var i, j, Mid, buf: integer;
        begin
            if First<Last then
                begin
                    Mid:=B[random(Last-First)+First];
                    i:=First-1;
                    j:=Last+1;
                    while i<j do
                        begin
                            repeat inc(i); until B[i]>=Mid;
                            repeat dec(j); until B[j]<=Mid;
                            if i<j then
                                begin
                                    buf:=B[i];
                                    B[i]:=B[j];
                                    B[j]:=buf;
                                end;
                        end;
                    Sort(First, j);
                    Sort(j+1, Last);
                end;
        end;
    begin
        Sort(1,m);
    end;
```

Тричі виконаємо програму, змінюючи кожного разу ім’я процедури сортування, отримаємо результати та проведемо асимптотичний аналіз для кожного методу (рис. 2.1 - 2.6).

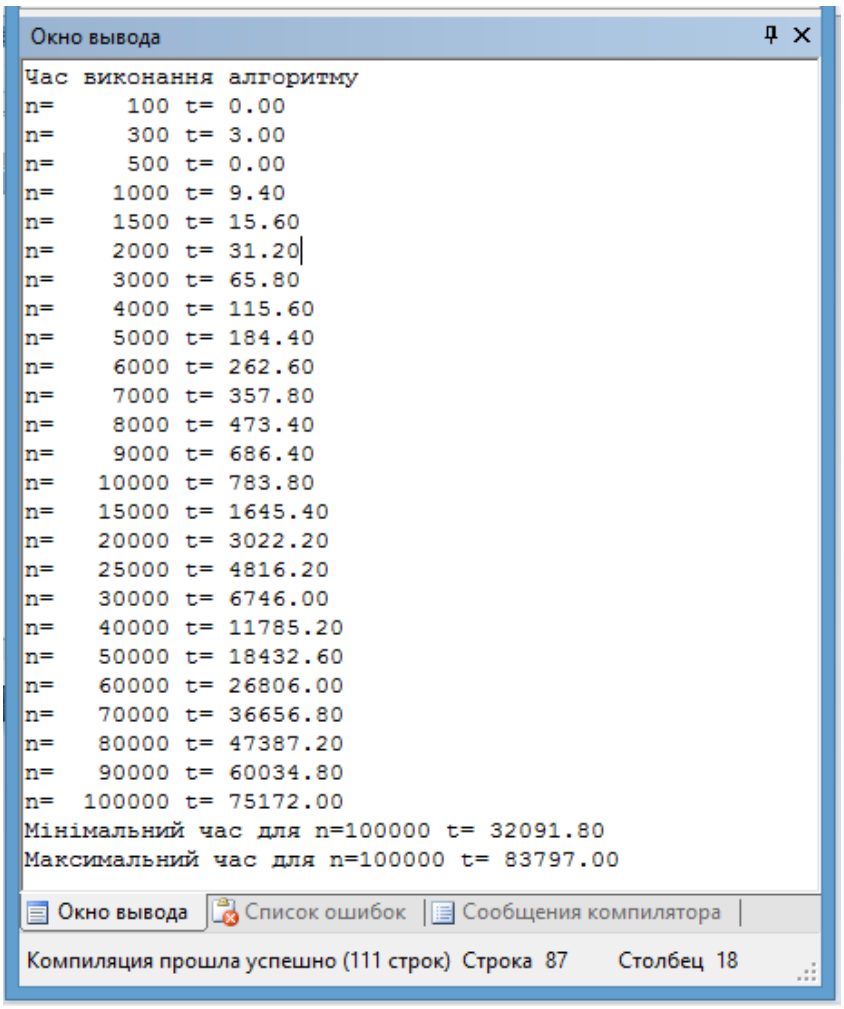
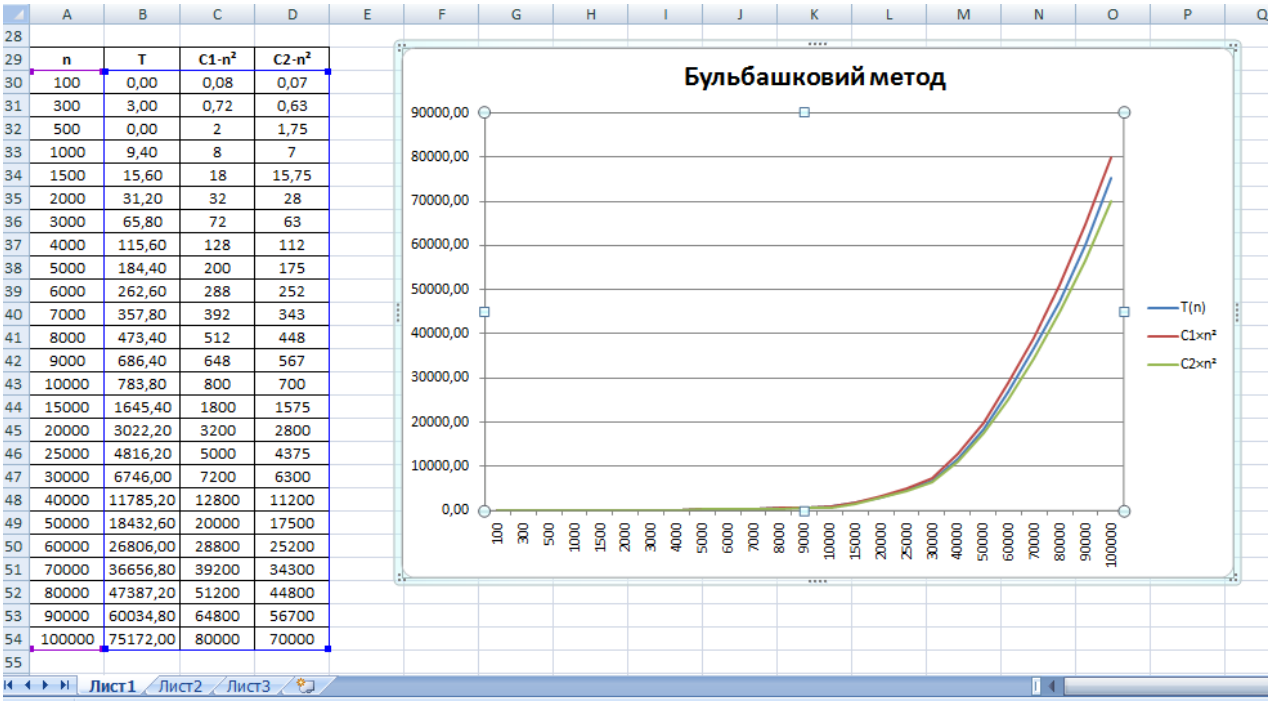


Рисунок 2.1 – Час сортування бульбашковим методом

Побудуємо графік залежності *T(n)* та верхню й нижню асимптоти.



n	T	C1·n²	C2·n²
100	0,00	0,08	0,07
300	3,00	0,72	0,63
500	0,00	2	1,75
1000	9,40	8	7
1500	15,60	18	15,75
2000	31,20	32	28
3000	65,80	72	63
4000	115,60	128	112
5000	184,40	200	175
6000	262,60	288	252
7000	357,80	392	343
8000	473,40	512	448
9000	686,40	648	567
10000	783,80	800	700
15000	1645,40	1800	1575
20000	3022,20	3200	2800
25000	4816,20	5000	4375
30000	6746,00	7200	6300
40000	11785,20	12800	11200
50000	18432,60	20000	17500
60000	26806,00	28800	25200
70000	36656,80	39200	34300
80000	47387,20	51200	44800
90000	60034,80	64800	56700
100000	75172,00	80000	70000

Рисунок 2.2 – Асимптотична оцінка алгоритму бульбашкового метода

Відомо, що алгоритм сортування за бульбашковим методом має квадратичну складність $O(n^2)$. Тому для побудови асимптот нам необхідно лише підібрати відповідні коефіцієнти:

$$C_2 n^2 \le T(n) \le C_1 n^2, \text{ при } n \ge n_0.$$

За отриманими результатами маємо:

$$7 \cdot 10^{-6} \cdot n^2 \le T(n) \le 8 \cdot 10^{-6} \cdot n^2, \text{ при } n \ge 1500.$$

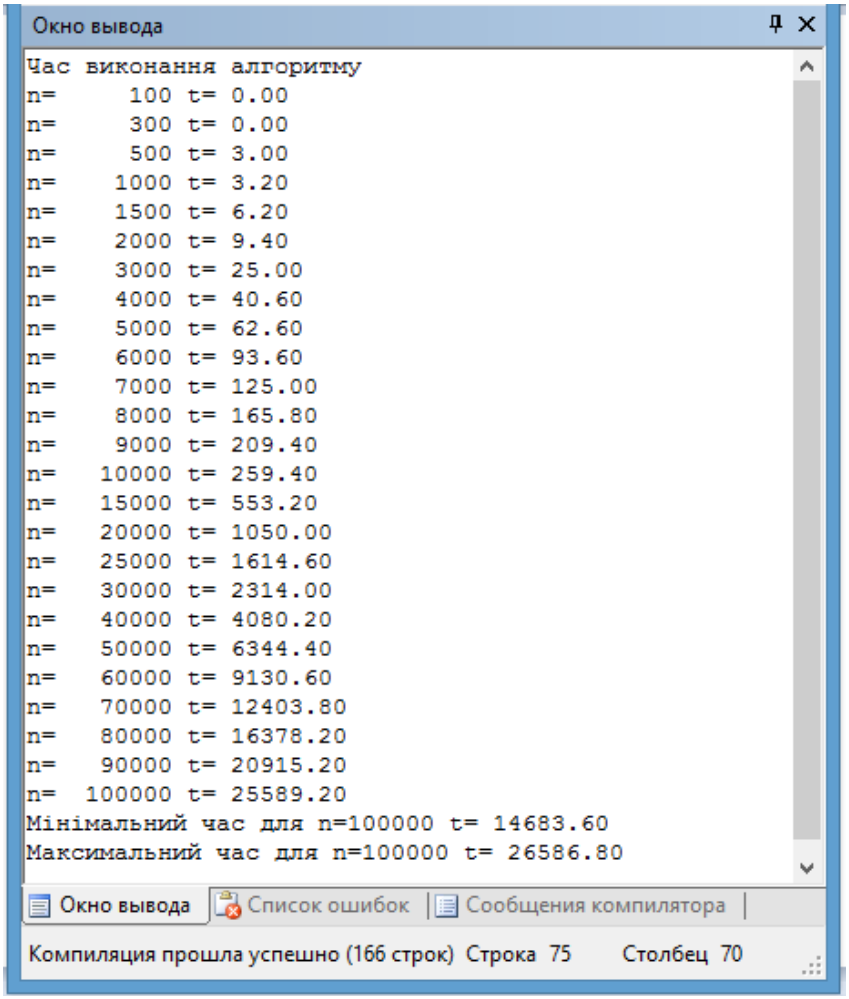


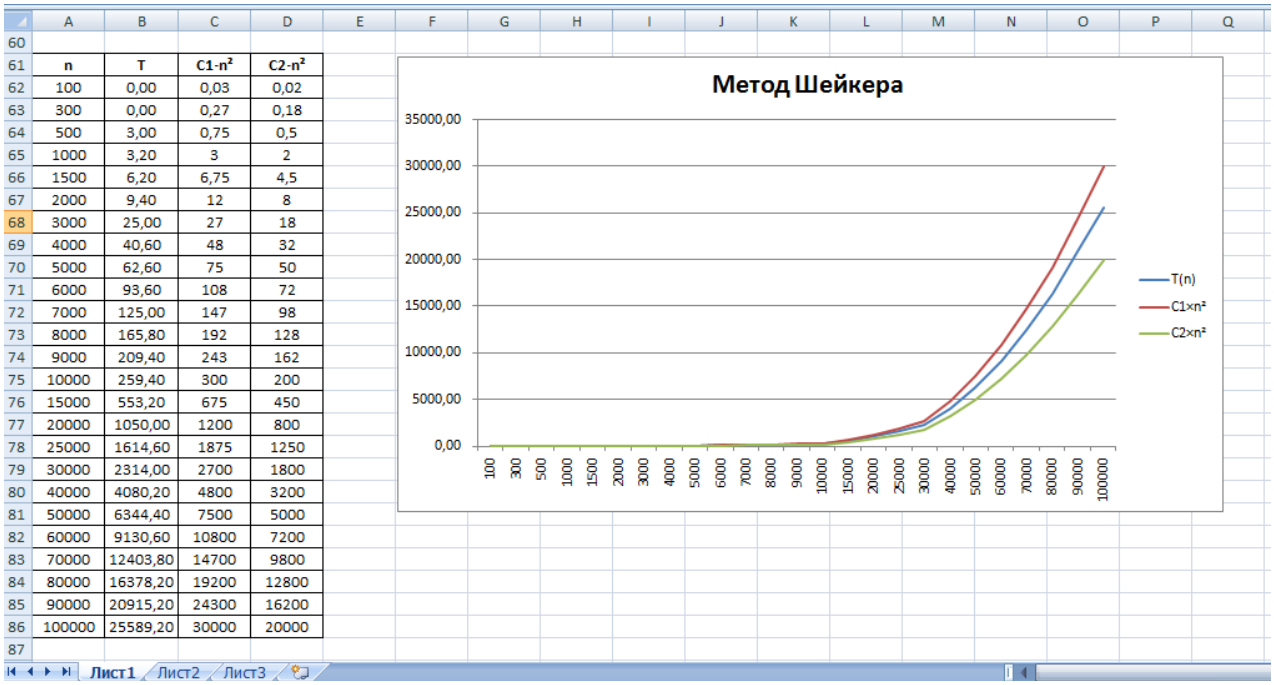
Рисунок 2.3 – Час сортування методом Шейкера

Побудуємо графік залежності $T(n)$ та верхню й нижню асимптоти. Відомо, що алгоритм сортування за методом Шейкера має квадратичну складність $O(n^2)$. Тому для побудови асимптот нам необхідно лише підібрати відповідні коефіцієнти:

$$C_2 n^2 \le T(n) \le C_1 n^2, \text{ при } n \ge n_0.$$

За отриманими результатами маємо:

$$2 \cdot 10^{-6} \cdot n^2 \le T(n) \le 3 \cdot 10^{-6} \cdot n^2, \text{ при } n \ge 1500.$$



n	T	C1·n²	C2·n²
100	0,00	0,03	0,02
300	0,00	0,27	0,18
500	3,00	0,75	0,5
1000	3,20	3	2
1500	6,20	6,75	4,5
2000	9,40	12	8
3000	25,00	27	18
4000	40,60	48	32
5000	62,60	75	50
6000	93,60	108	72
7000	125,00	147	98
8000	165,80	192	128
9000	209,40	243	162
10000	259,40	300	200
15000	553,20	675	450
20000	1050,00	1200	800
25000	1614,60	1875	1250
30000	2314,00	2700	1800
40000	4080,20	4800	3200
50000	6344,40	7500	5000
60000	9130,60	10800	7200
70000	12403,80	14700	9800
80000	16378,20	19200	12800
90000	20915,20	24300	16200
100000	25589,20	30000	20000

Рисунок 2.4 – Асимптотична оцінка алгоритму метода Шейкера

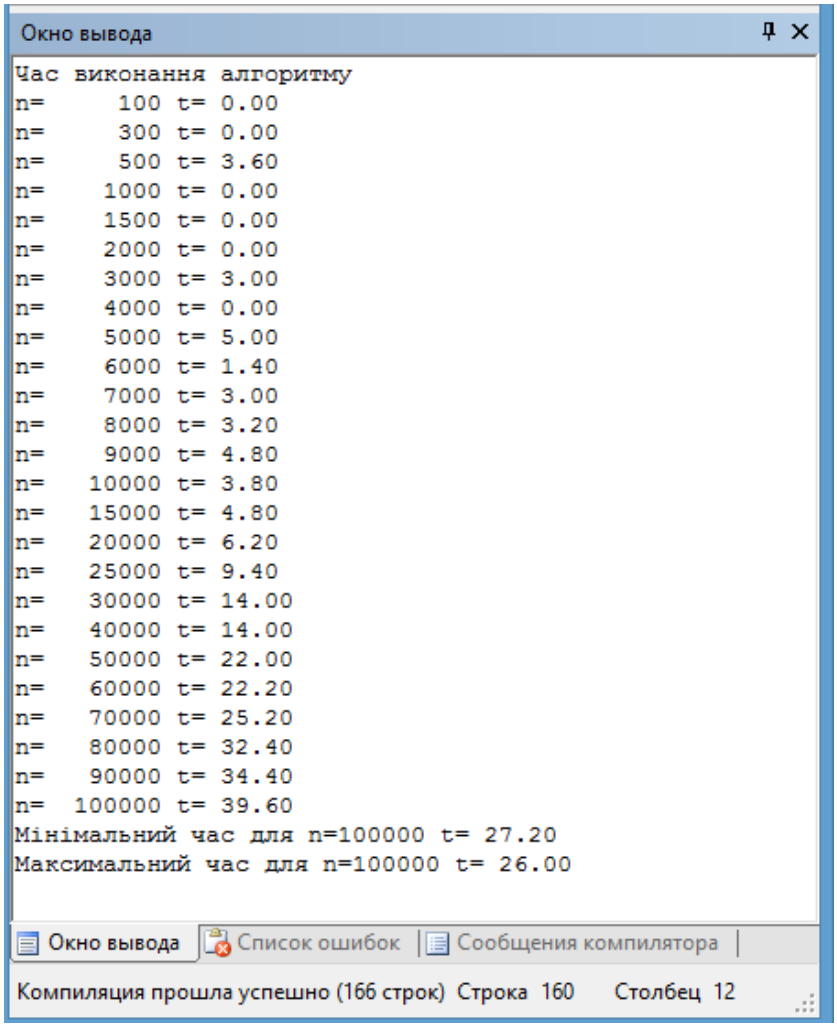
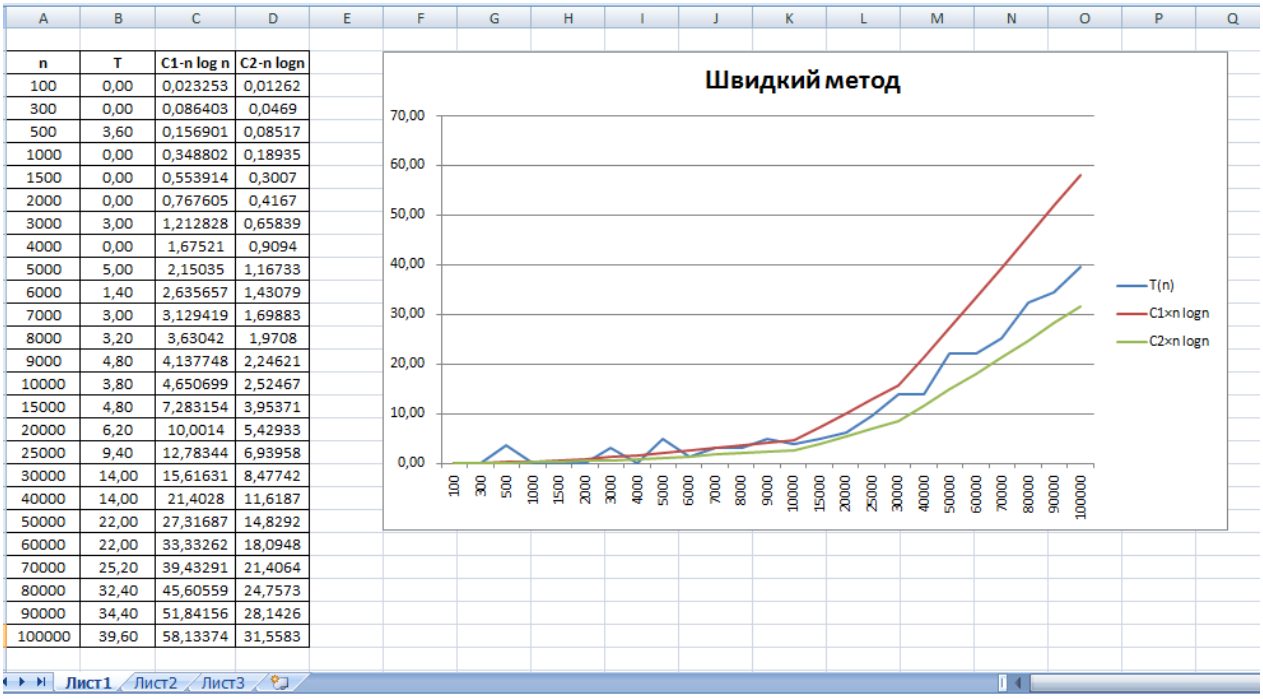


Рисунок 2.5 – Час сортування швидким методом

Побудуємо графік залежності *T(n)* та верхню й нижню асимптоти.



n	T	C1·n log n	C2·n log n
100	0,00	0,023253	0,01262
300	0,00	0,086403	0,0469
500	3,60	0,156901	0,08517
1000	0,00	0,348802	0,18935
1500	0,00	0,553914	0,3007
2000	0,00	0,767605	0,4167
3000	3,00	1,212828	0,65839
4000	0,00	1,67521	0,9094
5000	5,00	2,15035	1,16733
6000	1,40	2,635657	1,43079
7000	3,00	3,129419	1,69883
8000	3,20	3,63042	1,9708
9000	4,80	4,137748	2,24621
10000	3,80	4,650699	2,52467
15000	4,80	7,283154	3,95371
20000	6,20	10,0014	5,42933
25000	9,40	12,78344	6,93958
30000	14,00	15,61631	8,47742
40000	14,00	21,4028	11,6187
50000	22,00	27,31687	14,8292
60000	22,00	33,33262	18,0948
70000	25,20	39,43291	21,4064
80000	32,40	45,60559	24,7573
90000	34,40	51,84156	28,1426
100000	39,60	58,13374	31,5583

Рисунок 2.6 – Асимптотична оцінка алгоритму швидкого метода

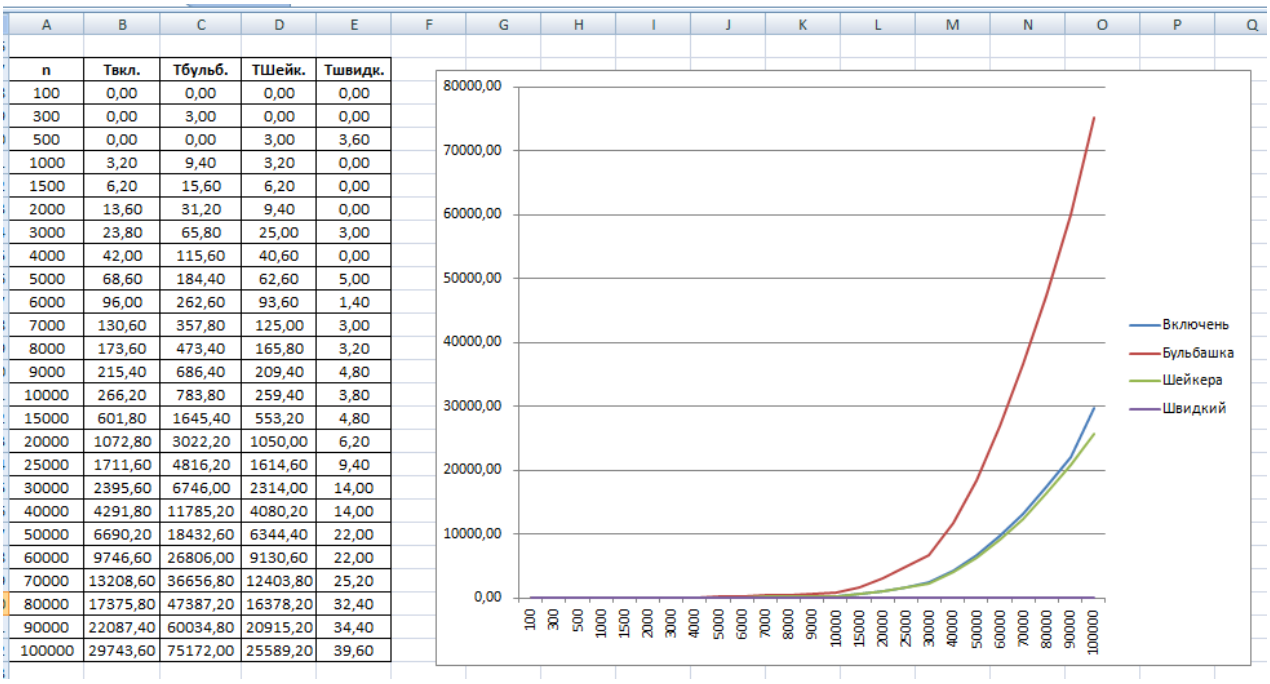
Відомо, що алгоритм сортування швидким методом має лінеаритмічну складність $O(n \cdot \log n)$. Тому для побудови асимптот нам необхідно лише підібрати відповідні коефіцієнти:

$$C_2 \cdot n \cdot \log n \le T(n) \le C_1 \cdot n \cdot \log n, \text{ при } n \ge n_0.$$

За отриманими результатами маємо:

$$1{,}9 \cdot 10^{-5} \cdot n \cdot \log_2 n \le T(n) \le 3{,}5 \cdot 10^{-5} \cdot n \cdot \log_2 n, \text{ при } n \ge 10000.$$

Побудуємо всі чотири отримані часові залежності на одному графіку (рис. 2.7).



n	Твкл.	Тбульб.	ТШейк.	Тшвидк.
100	0,00	0,00	0,00	0,00
300	0,00	3,00	0,00	0,00
500	0,00	0,00	3,00	3,60
1000	3,20	9,40	3,20	0,00
1500	6,20	15,60	6,20	0,00
2000	13,60	31,20	9,40	0,00
3000	23,80	65,80	25,00	3,00
4000	42,00	115,60	40,60	0,00
5000	68,60	184,40	62,60	5,00
6000	96,00	262,60	93,60	1,40
7000	130,60	357,80	125,00	3,00
8000	173,60	473,40	165,80	3,20
9000	215,40	686,40	209,40	4,80
10000	266,20	783,80	259,40	3,80
15000	601,80	1645,40	553,20	4,80
20000	1072,80	3022,20	1050,00	6,20
25000	1711,60	4816,20	1614,60	9,40
30000	2395,60	6746,00	2314,00	14,00
40000	4291,80	11785,20	4080,20	14,00
50000	6690,20	18432,60	6344,40	22,00
60000	9746,60	26806,00	9130,60	22,00
70000	13208,60	36656,80	12403,80	25,20
80000	17375,80	47387,20	16378,20	32,40
90000	22087,40	60034,80	20915,20	34,40
100000	29743,60	75172,00	25589,20	39,60

Рисунок 2.7 – Графіки залежності часу сортування
від вимірності вхідних даних

Контрольні запитання

1. Дайте визначення функції складності алгоритму.
2. Вкажіть правила для визначення функції складності.
3. Які види функції складності існують?
4. Яким чином визначається часова функція складності?
5. Поясніть принцип роботи бульбашкового методу сортування.
6. Поясніть принцип роботи методу Шейкера.
7. Поясніть принцип роботи швидкого методу сортування.

Перелік рекомендованої літератури

1. Седжвик Р. Алгоритмы на С++. Фундаментальные алгоритмы и структуры данных / Р. Седжвик. – М. : ИД "Вильямс", 2011. – 1056 с. : ил.
2. Красиков И.В. Алгоритмы. Просто как дважды два / И.В. Красиков, И.Е.°Красикова. –М.: Эксмо, 2007. – 256 с.

ЛАБОРАТОРНА РОБОТА № 3
РОЗВ'ЯЗУВАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ ПРЯМИМИ МЕТОДАМИ

Мета: опанувати прямими чисельними методами розв'язування систем лінійних алгебраїчних рівнянь – методом Гауса за схемою єдиного ділення та методом *LU*-розкладання.

Теоретичні відомості

Система лінійних алгебраїчних рівнянь (СЛАР) із *n*-невідомими має вигляд:

$$\left.\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1,\\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2,\\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ &a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{aligned}\right\}, \qquad (3.1)$$

або в компактному вигляді $\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1,2,\ldots,n.$ (3.2)

В матричній формі запишемо систему так:

$$A\vec{x} = \vec{b}, \qquad (3.3)$$

де $A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}$ - матриця коефіцієнтів системи; $\vec{b} = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix}$ - вектор вільних членів; $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}$ - вектор невідомих.

Система (3.1) буде мати єдиний розв'язок, якщо матриця *A* невироджена, тобто $\det A \neq 0$.

Чисельні методи розв'язування СЛАР діляться на дві групи: *прямі* та *ітераційні*.

Прямі методи дозволяють за скінчену кількість дій отримати точний розв'язок $\vec{x}$ системи (3.1), якщо елементи матриці *A* і вектор вільних членів $\vec{b}$ задано точно, і обчислення проводяться без округлень.

Ітераційні методи дозволяють знайти наближений розв'язок шляхом побудови послідовності наближень (ітерацій), починаючи з деякого довільного наближення.

Вибір методу розв'язування СЛАР залежить:

- від властивостей матриці А;
- від кількості рівнянь;
- від характеристик комп'ютера (швидкодії, розрядної сітки, об'єму операти-

вної пам'яті).

Прямі методи використовуються для розв'язування систем невеликої вимірності.

Ітераційні методи використовують зазвичай для систем великої вимірності, коли використання прямих методів є недоцільним через необхідність виконувати занадто велику кількість арифметичних операцій.

Метод Гауса є найрозповсюдженим прямим методом розв'язування систем лінійних алгебраїчних рівнянь. Ідея методу полягає у зведенні матриці коефіцієнтів системи *A* до трикутного вигляду, що досягається послідовним вилученням невідомих із рівнянь системи. Отримується еквівалентна система:

$$\left.\begin{aligned} x_1 + c_{12}x_2 + \ldots + c_{1n}x_n &= d_1, \\ x_2 + \ldots + c_{2n}x_n &= d_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ x_n &= d_n \end{aligned}\right\}, \qquad (3.4)$$

або в матричній формі запису: $C\vec{x} = \vec{d}$.

Зведення системи (3.1) до еквівалентної (3.4) називається *прямим ходом* метода Гауса, а розв'язування системи (3.4), тобто послідовне визначення невідомих, – *зворотним ходом* метода Гауса.

Прямий хід можна реалізувати за схемою єдиного ділення.

Схема єдиного ділення. Послідовно з системи (3.1) вилучаються невідомі $x_1, x_2, \ldots, x_i, \ldots, x_{n-1}$. Для вилучення *i*-ой невідомої з рівнянь системи з номерами $i+1, i+2, \ldots, n$ розділимо *i*-те рівняння на коефіцієнт a_{ii}. Потім від кожного $i+1, i+2, \ldots, n$ рівняння будемо віднімати *i*-те рівняння, помножене на відповідні коефіцієнти $a_{i+1,i}, a_{i+2,i}, \ldots, a_{n,i}$:

$$c_{kj} = \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}; \quad d_k = \frac{b_k^{(k-1)}}{a_{kk}^{(k-1)}}; \quad c_{1j} = \frac{a_{1j}}{a_{11}}; \quad d_1 = \frac{b_1}{a_{11}}; \qquad (3.5)$$

$k = 1,2,\ldots,n$ – крок перетворення

$i, j = k+1, k+2, \ldots, n.$

Зворотний хід відбувається за формулами:

$$\begin{aligned} x_n &= d_n; \\ x_{n-1} &= d_{n-1} - c_{n-1,n}x_n; \qquad (3.6) \\ x_i &= d_i - \sum_{j=i+1}^{n} c_{ij}x_j; \quad i = n, n-1, \ldots, 1. \end{aligned}$$

На практиці при розрахунках користуються розширеною матрицею коефіцієнтів системи, яку отримують із матриці *A*, доповнюючи її справа вектором $\vec{b}$. Блок-схема методу Гауса приведена на рис. 3.1.

Метод LU-розкладання. Даний метод є модифікацією методу Гауса. *LU*-розкладання – це представлення матриці *A* добутком двох матриць:

$$A = L \cdot U, \qquad (3.7)$$

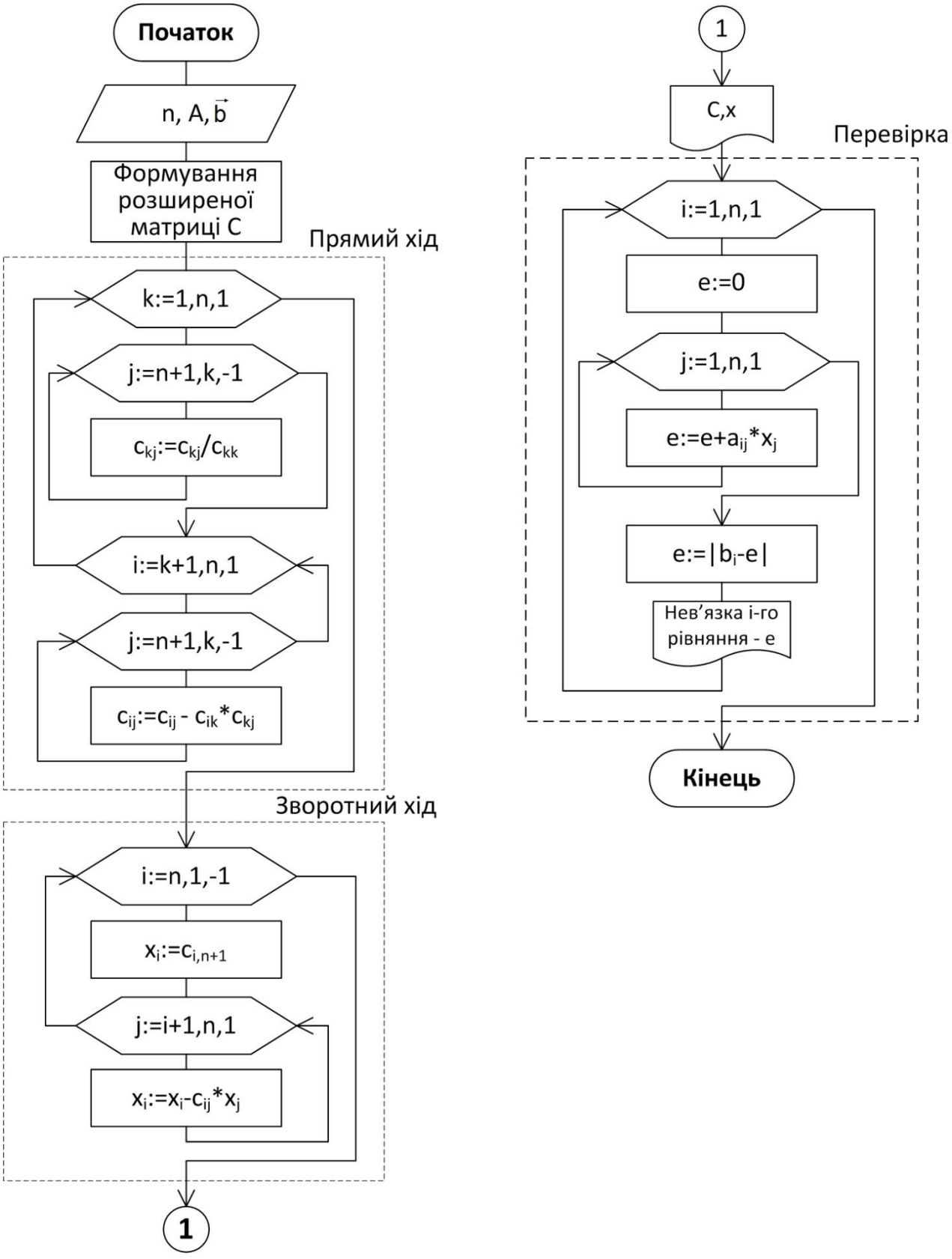


Рисунок 3.1 – Блок-схема методу Гауса

де $L=\begin{pmatrix} 1 & 0 & ... & 0 \\ l_{21} & 1 & ... & 0 \\ ... & ... & ... & ... \\ l_{n1} & l_{n1} & ... & 1 \end{pmatrix}$, $l_{ij}=\begin{cases} l_{ij}, \text{при } i>j \\ 1, \text{при } i=j \\ 0, \text{при } i<j \end{cases}$ - нижня трикутна матриця з фіксованими діагональними елементами;

$U=\begin{pmatrix} u_{11} & u_{12} & ... & u_{1n} \\ 0 & u_{22} & ... & u_{2n} \\ ... & ... & ... & ... \\ 0 & 0 & ... & u_{nn} \end{pmatrix}$, $u_{ij}=\begin{cases} u_{ij}, \text{при } i\le j \\ 0, \text{при } i<j \end{cases}$ - верхня трикутна матриця.

Якщо всі головні мінори матриці коефіцієнтів A системи (3.1) не дорівнюють нулю, то існують такі нижня L і верхня U трикутні матриці, що $A=L\cdot U$. Якщо елементи діагоналі однієї з матриць L або U фіксовані (ненульові), то таке розкладання буде єдиним.

Отримаємо формули для розкладання матриці A.

Утворимо матрицю $M^{(1)}$:

$$M^{(1)}=\begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ m_{21}^{(1)} & 1 & 0 & ... & 0 \\ m_{31}^{(1)} & 0 & 1 & ... & 0 \\ ... & ... & ... & ... & 0 \\ m_{n1}^{(1)} & 0 & 0 & ... & 1 \end{pmatrix}, \text{ де } m_{i1}^{(1)}=\frac{a_{i1}}{-a_{11}} \quad (i=2,\ldots,n).$$

Помножимо матрицю A зліва на $M^{(1)}$, отримаємо матрицю $A^{(1)}$:

$$M^{(1)}\cdot A=A^{(1)}=\begin{pmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & ... & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & ... & a_{3n}^{(1)} \\ ... & ... & ... & ... & 0 \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & ... & a_{nn}^{(1)} \end{pmatrix}.$$

З якої утворимо матрицю $M^{(2)}$:

$$M^{(2)}=\begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 \\ 0 & m_{32}^{(2)} & 1 & ... & 0 \\ ... & ... & ... & ... & 0 \\ 0 & m_{n2}^{(2)} & 0 & ... & 1 \end{pmatrix}, \text{ де } m_{i2}^{(2)}=\frac{a_{i2}^{(1)}}{-a_{22}^{(1)}} \quad (i=3,\ldots,n).$$

Помножимо матрицю $A^{(1)}$ зліва на $M^{(2)}$, отримаємо матрицю $A^{(2)}$:

$$M^{(1)}\cdot A^{(1)}=A^{(2)}=\begin{pmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & ... & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & ... & a_{3n}^{(2)} \\ ... & ... & ... & ... & 0 \\ 0 & 0 & a_{n3}^{(2)} & ... & a_{nn}^{(2)} \end{pmatrix}.$$

І так далі. Остання буде матриця $M^{(n-1)}$:

$$M^{(n-1)} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & m_{n,n-1}^{(n-1)} & 1 \end{pmatrix}, \text{ де } m_{n,n-1}^{(n-1)} = \frac{a_{n,n-1}^{(n-2)}}{-a_{n-1,n-1}^{(n-2)}}.$$

Тоді матриця $A^{(n-1)}$ буде верхньою трикутною матрицею:

$$M^{(n-1)} \cdot A^{(n-2)} = A^{(n-1)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n-1} & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n-1}^{(1)} & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3n-1}^{(2)} & a_{3n}^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & a_{nn}^{(n-1)} \end{pmatrix},$$

тобто матрицею *U*. Таким чином $U = M^{(n-1)} \cdot M^{(n-2)} \cdot \dots \cdot M^{(2)} \cdot M^{(1)} \cdot A$, а нижня трикутна матриця *L* отримується з ненульових стовпців матриць $M^{(1)}, \cdot M^{(2)}, \dots, M^{(n-2)}, M^{(n-1)}$:

$$L = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ -m_{21}^{(1)} & 1 & 0 & \dots & 0 & 0 \\ -m_{31}^{(1)} & -m_{31}^{(2)} & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & 1 & 0 \\ -m_{n1}^{(1)} & -m_{n1}^{(2)} & -m_{n1}^{(3)} & \dots & -m_{n,n-1}^{(n-1)} & 1 \end{pmatrix}, \text{ або } L = (M^{(n-1)} \cdot M^{(n-2)} \cdot \dots \cdot M^{(2)} \cdot M^{(1)})^{-1}.$$

Якщо матриця *A* системи (3.1) розкладена на добуток трикутних матриць *L* і *U*, то замість системи (3.3) можемо записати еквівалентне рівняння:

$$L \cdot U \cdot \vec{x} = \vec{b}. \tag{3.8}$$

Введемо допоміжний вектор змінних $\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} = U \cdot \vec{x}$., тоді перепишемо систему (3.8) у вигляді $\begin{cases} L \cdot \vec{y} = \vec{b}, \\ U \cdot \vec{x} = \vec{y}. \end{cases}$ Таким чином, розв'язування системи (3.1) звелося до послідовного розв'язання двох трикутних систем.

Запишемо перше рівняння $L\vec{y} = \vec{b}$ у розгорнутій формі:

$$\begin{cases} y_1 = b_1, \\ l_{21} y_1 + y_2 = b_2, \\ \dots\dots\dots\dots\dots\dots\dots \\ l_{n1} y_1 + l_{n2} y_2 + \dots + l_{n,n-1} y_{n-1} + y_n = b_n. \end{cases} \tag{3.9}$$

Звідки знаходимо значення компонент вектора $\vec{y}$ шляхом прямих підстановок:

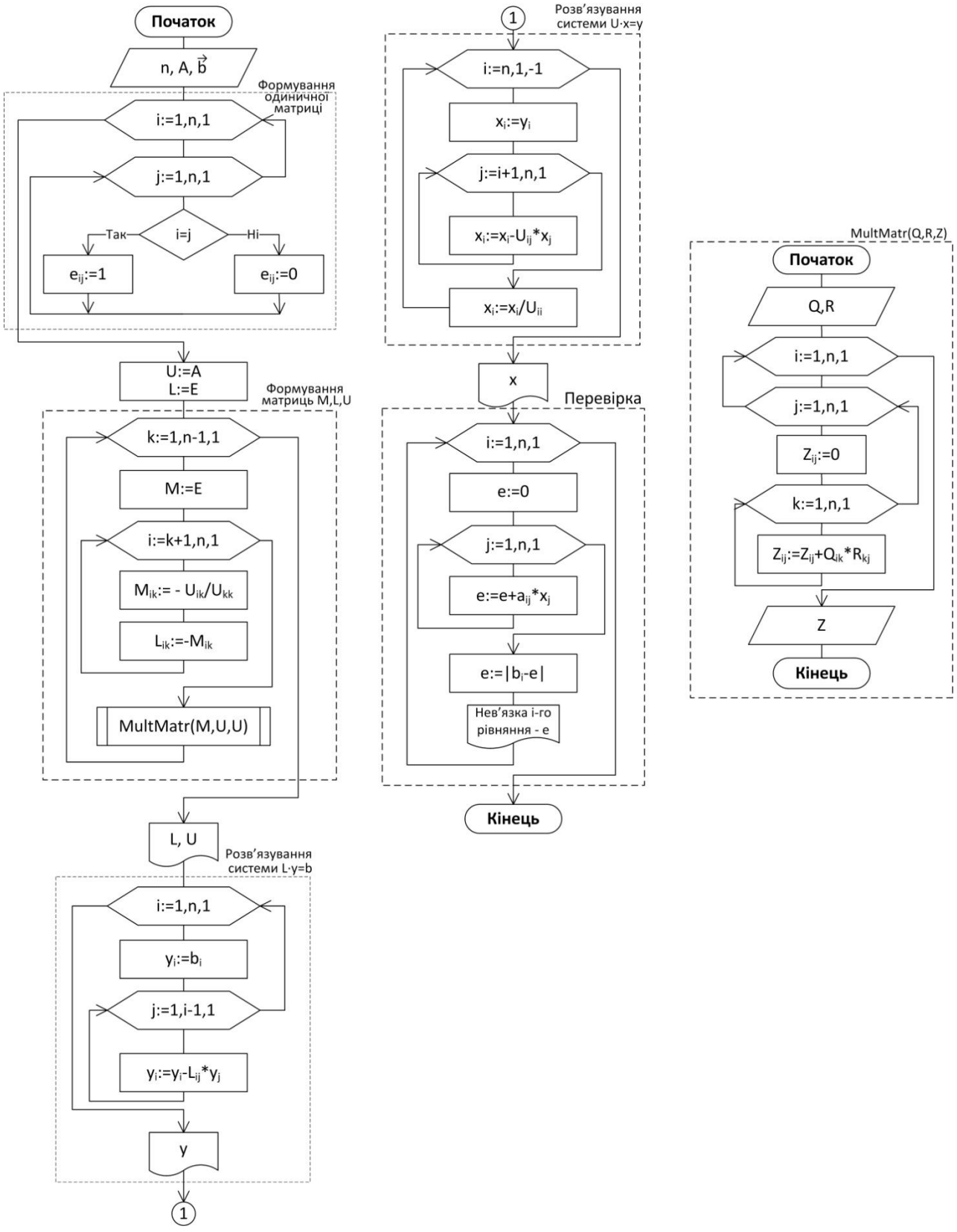


Рисунок 3.2 – Блок-схема методу *LU*-розкладання

$$\begin{cases} y_1 = b_1, \\ y_2 = b_2 - l_{21} y_1, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ y_n = b_n - l_{n1} y_1 - l_{n2} y_2 - \ldots - l_{n,n-1} y_{n-1}. \end{cases} \tag{3.10}$$

Розгорнемо тепер друге рівняння $U\vec{x} = \vec{y}$:

$$\begin{cases} u_{11} x_1 + u_{12} x_2 + \ldots + u_{1n} x_n = y_1, \\ u_{22} x_2 + \ldots + u_{2n} x_n = y_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ u_{nn} x_n = y_n. \end{cases} \tag{3.11}$$

Звідки знайдемо значення невідомих у зворотному порядку:

$$\begin{cases} x_n = \dfrac{y_n}{u_{nn}}, \\ x_{n-1} = \dfrac{y_{n-1} - u_{n-1,n} x_n}{u_{n-1,n-1}}, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_1 = \dfrac{y_1 - \sum\limits_{k=2}^{n} u_{1,k} x_k}{u_{11}}. \end{cases} \tag{3.12}$$

Блок-схема методу *LU*-розкладання приведена на рис.3.2.

Завдання

Методом Гауса за схемою єдиного ділення та методом *LU*-розкладання розв’язати СЛАР.

1. $\begin{cases} 8{,}2x_1 - 3{,}2x_2 + 14{,}2x_3 + 14{,}8x_4 = -8{,}4 \\ 5{,}6x_1 - 12x_2 + 15x_3 - 6{,}4x_4 = 4{,}5 \\ 5{,}7x_1 + 3{,}6x_2 - 12{,}4x_3 - 2{,}3x_4 = 3{,}3 \\ 6{,}8x_1 + 13{,}2x_2 - 6{,}3x_3 - 8{,}7x_4 = 14{,}3 \end{cases}$

2. $\begin{cases} 7{,}3x_1 + 12{,}4x_2 - 3{,}8x_3 - 14{,}3x_4 = 5{,}8 \\ 10{,}7x_1 - 7{,}7x_2 + 12{,}5x_3 + 6{,}6x_4 = -6{,}6 \\ 15{,}6x_1 + 6{,}6x_2 + 14{,}4x_3 - 8{,}7x_4 = 12{,}4 \\ 7{,}5x_1 + 12{,}2x_2 - 8{,}3x_3 + 3{,}7x_4 = 9{,}2 \end{cases}$

3. $\begin{cases} 3{,}8x_1 + 14{,}2x_2 + 6{,}3x_3 - 15{,}5x_4 = 2{,}8 \\ 8{,}3x_1 - 6{,}6x_2 + 5{,}8x_3 + 12{,}2x_4 = -4{,}7 \\ 6{,}4x_1 - 8{,}5x_2 - 4{,}3x_3 + 8{,}8x_4 = 7{,}7 \\ 17{,}1x_1 - 8{,}3x_2 + 14{,}4x_3 - 7{,}2x_4 = 13{,}5 \end{cases}$

4. $\begin{cases} 13{,}2x_1 - 8{,}3x_2 - 4{,}4x_3 + 6{,}2x_4 = 6{,}8 \\ 8{,}3x_1 + 4{,}2x_2 - 5{,}6x_3 + 7{,}7x_4 = 12{,}4 \\ 5{,}8x_1 - 3{,}7x_2 + 12{,}4x_3 - 6{,}2x_4 = 8{,}7 \\ 3{,}5x_1 + 6{,}6x_2 - 13{,}8x_3 - 9{,}3x_4 = -10{,}8 \end{cases}$

5. $\begin{cases} 4{,}3x_1 - 12{,}1x_2 + 23{,}2x_3 - 14{,}1x_4 = 15{,}5 \\ 2{,}4x_1 - 4{,}4x_2 + 3{,}5x_3 + 5{,}5x_4 = 2{,}5 \\ 5{,}4x_1 + 8{,}3x_2 - 7{,}4x_3 - 12{,}7x_4 = 8{,}6 \\ 6{,}3x_1 - 7{,}6x_2 + 1{,}34x_3 + 3{,}7x_4 = 12{,}1 \end{cases}$

6. $\begin{cases} 8{,}1x_1 + 1{,}2x_2 - 9{,}1x_3 + 1{,}7x_4 = 10 \\ 1{,}1x_1 - 1{,}7x_2 + 7{,}2x_3 - 3{,}4x_4 = 1{,}7 \\ 1{,}7x_1 - 1{,}8x_2 + 10x_3 + 2{,}3x_4 = 2{,}1 \\ 1{,}3x_1 + 1{,}7x_2 - 9{,}9x_3 + 3{,}5x_4 = 27{,}1 \end{cases}$

7. $\begin{cases} 1{,}7x_1 + 10x_2 - 1{,}3x_3 + 2{,}1x_4 = 3{,}1 \\ 3{,}1x_1 + 1{,}7x_2 - 2{,}1x_3 + 5{,}4x_4 = 2{,}1 \\ 3{,}3x_1 - 7{,}7x_2 + 4{,}4x_3 - 5{,}1x_4 = 1{,}9 \\ 10x_1 - 20{,}1x_2 + 20{,}4x_3 + 1{,}7x_4 = 1{,}8 \end{cases}$

8. $\begin{cases} 3{,}3x_1 - 2{,}2x_2 - 10x_3 + 1{,}7x_4 = 1{,}1 \\ 1{,}8x_1 + 21{,}1x_2 + 1{,}3x_3 - 2{,}2x_4 = 2{,}2 \\ -10x_1 + 1{,}1x_2 + 20x_3 - 4{,}5x_4 = 10 \\ 70x_1 - 1{,}7x_2 - 2{,}2x_3 + 3{,}3x_4 = 2{,}1 \end{cases}$

9. $\begin{cases} 6{,}1x_1 + 6{,}2x_2 - 6{,}3x_3 + 6{,}4x_4 = 6{,}5 \\ 1{,}1x_1 - 1{,}5x_2 + 2{,}2x_3 - 3{,}8x_4 = 4{,}2 \\ 5{,}1x_1 - 5{,}0x_2 + 4{,}9x_3 - 4{,}8x_4 = 4{,}7 \\ 1{,}8x_1 + 1{,}9x_2 + 2{,}0x_3 - 2{,}1x_4 = 2{,}2 \end{cases}$

10. $\begin{cases} 1{,}7x_1 + 9{,}9x_2 - 20x_3 - 1{,}7x_4 = 1{,}7 \\ 20x_1 + 0{,}5x_2 - 30{,}1x_3 - 1{,}1x_4 = 2{,}1 \\ 10x_1 - 20x_2 + 30{,}2x_3 + 0{,}5x_4 = 1{,}8 \\ 3{,}3x_1 - 0{,}7x_2 + 3{,}3x_3 + 20x_4 = -1{,}7 \end{cases}$

11. $\begin{cases} 2{,}2x_1 - 3{,}1x_2 + 4{,}2x_3 - 5{,}1x_4 = 6{,}01 \\ 1{,}3x_1 + 2{,}2x_2 - 1{,}4x_3 + 1{,}5x_4 = 10 \\ 6{,}2x_1 - 7{,}4x_2 + 8{,}5x_3 - 9{,}6x_4 = 1{,}1 \\ 1{,}2x_1 + 1{,}3x_2 + 1{,}4x_3 + 4{,}5x_4 = 1{,}6 \end{cases}$

12. $\begin{cases} 1{,}7x_1 - 1{,}3x_2 - 1{,}1x_3 - 1{,}2x_4 = 2{,}2 \\ 10x_1 - 10x_2 - 1{,}3x_3 + 1{,}3x_4 = 1{,}1 \\ 3{,}5x_1 + 3{,}3x_2 + 1{,}2x_3 + 1{,}3x_4 = 1{,}2 \\ 1{,}3x_1 + 1{,}1x_2 - 1{,}3x_3 - 1{,}1x_4 = 10 \end{cases}$

13. $\begin{cases} 35{,}8x_1 + 2{,}1x_2 - 34{,}5x_3 - 11{,}8x_4 = 0{,}5 \\ 27{,}1x_1 - 7{,}5x_2 + 11{,}7x_3 - 23{,}5x_4 = 12{,}8 \\ 11{,}7x_1 + 1{,}8x_2 - 6{,}5x_3 + 7{,}1x_4 = 1{,}7 \\ 6{,}3x_1 + 10x_2 + 7{,}1x_3 + 3{,}4x_4 = 20{,}8 \end{cases}$

14. $\begin{cases} 1{,}1x_1 + 11{,}3x_2 - 1{,}7x_3 + 1{,}8x_4 = 10 \\ 1{,}3x_1 - 11{,}7x_2 + 1{,}8x_3 + 1{,}4x_4 = 1{,}3 \\ 1{,}1x_1 - 10{,}5x_2 - 1{,}7x_3 - 1{,}5x_4 = 1{,}1 \\ 1{,}5x_1 - 0{,}5x_2 + 1{,}8x_3 - 1{,}1x_4 = 10 \end{cases}$

15. $\begin{cases} 35{,}1x_1 + 1{,}7x_2 + 37{,}5x_3 - 2{,}8x_4 = 7{,}5 \\ 45{,}2x_1 + 21{,}1x_2 - 1{,}1x_3 - 1{,}2x_4 = 11{,}1 \\ -21{,}1x_1 + 31{,}7x_2 + 1{,}2x_3 - 1{,}5x_4 = 2{,}1 \\ 31{,}7x_1 + 18{,}1x_2 - 31{,}7x_3 + 2{,}2x_4 = 0{,}5 \end{cases}$

16. $\begin{cases} 1{,}4x_1 + 2{,}1x_2 - 3{,}3x_3 + 1{,}1x_4 = 10 \\ 10x_1 - 1{,}7x_2 + 1{,}1x_3 - 1{,}5x_4 = 1{,}7 \\ 2{,}2x_1 + 34{,}4x_2 - 1{,}1x_3 - 1{,}2x_4 = 20 \\ 1{,}1x_1 + 1{,}3x_2 + 1{,}2x_3 + 1{,}4x_4 = 1{,}3 \end{cases}$

17. $\begin{cases} 1{,}1x_1 + 11{,}2x_2 + 11{,}1x_3 - 13{,}1x_4 = 1{,}3 \\ -3{,}3x_1 + 1{,}1x_2 + 30{,}1x_3 - 20{,}1x_4 = 1{,}1 \\ 7{,}5x_1 + 1{,}3x_2 + 1{,}1x_3 + 10x_4 = 20 \\ 1{,}7x_1 + 7{,}5x_2 - 1{,}8x_3 + 2{,}1x_4 = 1{,}1 \end{cases}$

18. $\begin{cases} 1{,}3x_1 - 1{,}7x_2 + 3{,}3x_3 + 1{,}7x_4 = 1{,}1 \\ 10x_1 + 5{,}5x_2 - 1{,}3x_3 + 3{,}4x_4 = 1{,}3 \\ 1{,}1x_1 + 1{,}8x_2 - 2{,}2x_3 - 1{,}1x_4 = 10 \\ 1{,}3x_1 - 1{,}2x_2 + 2{,}1x_3 + 2{,}2x_4 = 1{,}8 \end{cases}$

19. $\begin{cases} 7{,}5x_1 + 1{,}8x_2 - 2{,}1x_3 - 7{,}7x_4 = 1{,}1 \\ -10x_1 + 1{,}3x_2 - 20x_3 - 1{,}4x_4 = 1{,}5 \\ 2{,}8x_1 - 1{,}7x_2 + 3{,}9x_3 + 4{,}8x_4 = 1{,}2 \\ 10x_1 + 31{,}4x_2 - 2{,}1x_3 - 10x_4 = -1{,}1 \end{cases}$

20. $\begin{cases} 1{,}2x_1 + 1{,}8x_2 - 2{,}2x_3 - 4{,}1x_4 = 1{,}3 \\ 10x_1 - 5{,}1x_2 + 1{,}2x_3 + 5{,}5x_4 = 1{,}2 \\ 2{,}2x_1 - 30{,}1x_2 + 3{,}1x_3 + 5{,}8x_4 = 10 \\ 10x_1 + 2{,}4x_2 - 30{,}5x_3 - 2{,}2x_4 = 34{,}1 \end{cases}$

Послідовність виконання лабораторної роботи

1. ***Методом Гауса*** за схемою єдиного ділення розв'язати СЛАР:

$$\begin{cases} 0{,}11x_1 + 1{,}13x_2 - 0{,}17x_3 + 0{,}18x_4 = 1{,}00 \\ 0{,}13x_1 - 1{,}17x_2 + 0{,}18x_3 + 0{,}14x_4 = 0{,}13 \\ 0{,}11x_1 - 1{,}05x_2 - 0{,}17x_3 - 0{,}15x_4 = 0{,}11 \\ 0{,}15x_1 - 0{,}05x_2 + 0{,}18x_3 - 0{,}11x_4 = 1{,}00 \end{cases}$$

Сформуємо розширену матрицю коефіцієнтів C – до матриці коефіцієнтів A додаємо вектор вільних членів $\vec{b}$ стовпцем справа:

$$\begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & \vec{b} \end{matrix} \\ \left(\begin{array}{cccc|c} 0{,}11 & 1{,}13 & -0{,}17 & 0{,}18 & 1{,}00 \\ 0{,}13 & -1{,}17 & 0{,}18 & 0{,}14 & 0{,}13 \\ 0{,}11 & -1{,}05 & -0{,}17 & -0{,}15 & 0{,}11 \\ 0{,}15 & -0{,}05 & 0{,}18 & -0{,}11 & 1{,}00 \end{array}\right) \end{array}$$

Вилучимо з системи невідому x_1. Для цього перше рівняння оголосимо *головним* і розділимо його на коефіцієнт при x_1: $c_{1,1} = 0{,}11$.

$$\begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & \vec{b} \end{matrix} \\ \left(\begin{array}{cccc|c} 0{,}11 & 1{,}13 & -0{,}17 & 0{,}18 & 1{,}00 \\ 0{,}13 & -1{,}17 & 0{,}18 & 0{,}14 & 0{,}13 \\ 0{,}11 & -1{,}05 & -0{,}17 & -0{,}15 & 0{,}11 \\ 0{,}15 & -0{,}05 & 0{,}18 & -0{,}11 & 1{,}00 \end{array}\right) \end{array} : 0{,}11 \quad \sim \quad \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & \vec{b} \end{matrix} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0{,}13 & -1{,}17 & 0{,}18 & 0{,}14 & 0{,}13 \\ 0{,}11 & -1{,}05 & -0{,}17 & -0{,}15 & 0{,}11 \\ 0{,}15 & -0{,}05 & 0{,}18 & -0{,}11 & 1{,}00 \end{array}\right) \end{array}$$

Далі від другого рівняння віднімемо головне, помножене на коефіцієнт $c_{2,1}=0{,}13$; від третього рівняння віднімемо головне, помножене на коефіцієнт $c_{3,1}=0{,}11$; від четвертого рівняння віднімемо головне, помножене на коефіцієнт $c_{4,1}=0{,}15$.

$$\begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ (0{,}13-1\cdot 0{,}13) & (-1{,}17-10{,}273\cdot 0{,}13) & (0{,}18+1{,}545\cdot 0{,}13) & (0{,}14-1{,}636\cdot 0{,}13) & (0{,}13-9{,}091\cdot 0{,}13) \\ (0{,}11-1\cdot 0{,}11) & (-1{,}05-10{,}273\cdot 0{,}11) & (-0{,}17+1{,}545\cdot 0{,}11) & (-0{,}15-1{,}636\cdot 0{,}11) & (0{,}11-9{,}091\cdot 0{,}11) \\ (0{,}15-1\cdot 0{,}15) & (-0{,}05-10{,}2\cdot 0{,}15) & (0{,}18+1{,}545\cdot 0{,}15) & (-0{,}11-1{,}636\cdot 0{,}15) & (1{,}00-9{,}091\cdot 0{,}15) \end{array}\right) \sim \end{array}$$

$$\sim \begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0 & -2{,}505 & 0{,}381 & -0{,}073 & -1{,}052 \\ 0 & -2{,}18 & 0 & -0{,}33 & -0{,}89 \\ 0 & -1{,}591 & 0{,}412 & -0{,}355 & -0{,}364 \end{array}\right). \end{array}$$

Аналогічно вилучимо з системи невідомі x_2, x_3, x_4.

$$\begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0 & -2{,}505 & 0{,}381 & -0{,}073 & -1{,}052 \\ 0 & -2{,}18 & 0 & -0{,}33 & -0{,}89 \\ 0 & -1{,}591 & 0{,}412 & -0{,}355 & -0{,}364 \end{array}\right) : (-2{,}505) \end{array} \sim \begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0 & 1 & -0{,}152 & 0{,}029 & 0{,}420 \\ 0 & -2{,}18 & 0 & -0{,}33 & -0{,}89 \\ 0 & -1{,}591 & 0{,}412 & -0{,}355 & -0{,}364 \end{array}\right) \sim \end{array}$$

$$\sim \begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0 & 1 & -0{,}152 & 0{,}029 & 0{,}420 \\ 0 & (-2{,}18+1\cdot 2{,}18) & (0-0{,}152\cdot 2{,}18) & (-0{,}33+0{,}029\cdot 2{,}18) & (-0{,}89+0{,}420\cdot 2{,}18) \\ 0 & (-1{,}591+1\cdot 1{,}591) & (0{,}412-0{,}152\cdot 1{,}591) & (-0{,}355+0{,}029\cdot 1{,}591) & (-0{,}364+0{,}420\cdot 1{,}591) \end{array}\right) \sim \end{array}$$

$$\sim \begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0 & 1 & -0{,}152 & 0{,}029 & 0{,}420 \\ 0 & 0 & -0{,}331 & -0{,}267 & 0{,}025 \\ 0 & 0 & 0{,}170 & -0{,}309 & 0{,}304 \end{array}\right) : (-0{,}331) \end{array} \sim \begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0 & 1 & -0{,}152 & 0{,}029 & 0{,}420 \\ 0 & 0 & 1 & 0{,}805 & -0{,}076 \\ 0 & 0 & 0{,}170 & -0{,}309 & 0{,}304 \end{array}\right) \sim \end{array}$$

$$\sim \begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0 & 1 & -0{,}152 & 0{,}029 & 0{,}420 \\ 0 & 0 & 1 & 0{,}805 & -0{,}076 \\ 0 & 0 & (0{,}170-1\cdot 0{,}17) & (-0{,}309-0{,}805\cdot 0{,}17) & (0{,}304+0{,}076\cdot 0{,}17) \end{array}\right) \sim \end{array}$$

$$\sim \begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0 & 1 & -0{,}152 & 0{,}029 & 0{,}420 \\ 0 & 0 & 1 & 0{,}805 & -0{,}076 \\ 0 & 0 & 0 & -0{,}446 & 0{,}317 \end{array}\right) : (-0{,}446) \end{array} \sim \begin{array}{c} x_1 \quad x_2 \quad x_3 \quad x_4 \quad \vec{b} \\ \left(\begin{array}{cccc|c} 1 & 10{,}273 & -1{,}545 & 1{,}636 & 9{,}091 \\ 0 & 1 & -0{,}152 & 0{,}029 & 0{,}420 \\ 0 & 0 & 1 & 0{,}805 & -0{,}076 \\ 0 & 0 & 0 & 1 & -0{,}711 \end{array}\right). \end{array}$$

Таким чином початкову систему звели до трикутної.

$$\begin{cases} x_1 + 10{,}273x_2 - 1{,}545x_3 + 1{,}636x_4 = 9{,}091 \\ x_2 - 0{,}152x_3 + 0{,}029x_4 = 0{,}420 \\ x_3 + 0{,}805x_4 = -0{,}076 \\ x_4 = -0{,}711 \end{cases}$$

Виконаємо зворотний хід прямими підстановками: спочатку з останнього рівняння визначимо x_4. Потім з третього – x_3 через x_4; з другого – x_2 через x_3і x_2; і, нарешті, з першого – x_1 через x_2, x_3 і x_4.

$x_4 = -0,711;$

$x_3 = -0,076 - 0,805x_4 = -0,076 - 0,805 \cdot (-0,711) = 0,496;$

$x_2 = 0,420 + 0,152x_3 - 0,029x_4 = 0,420 + 0,152 \cdot 0,496 - 0,029 \cdot (-0,711) = 0,516;$

$x_1 = 9,091 - 10,273x_2 + 1,545x_3 - 1,636x_4 = 9,091 - 10,273 \cdot 0,516 + 1,545 \cdot 0,496 - 1,636 \cdot (-0,711) = 5,722.$

Відповідь: $x_1 = 5,722;\ x_2 = 0,516; x_3 = 0,496; x_4 = -0,711.$

2. ***Методом LU-розкладання*** розв'язати СЛАР:

$$\begin{cases} 0,11x_1 + 1,13x_2 - 0,17x_3 + 0,18x_4 = 1,00 \\ 0,13x_1 - 1,17x_2 + 0,18x_3 + 0,14x_4 = 0,13 \\ 0,11x_1 - 1,05x_2 - 0,17x_3 - 0,15x_4 = 0,11 \\ 0,15x_1 - 0,05x_2 + 0,18x_3 - 0,11x_4 = 1,00 \end{cases}$$

Сформуємо матрицю із коефіцієнтів системи: $A = \begin{pmatrix} 0,11 & 1,13 & -0,17 & 0,18 \\ 0,13 & -1,17 & 0,18 & 0,14 \\ 0,11 & -1,05 & -0,17 & -0,15 \\ 0,15 & -0,05 & 0,18 & -0,11 \end{pmatrix}$.

Формуємо M_1: $m_{21} = \dfrac{a_{21}}{-a_{11}} = \dfrac{0,13}{-0,11} = -1,1818;\ m_{31} = \dfrac{a_{31}}{-a_{11}} = \dfrac{0,11}{-0,11} = -1;$

$m_{41} = \dfrac{a_{41}}{-a_{11}} = \dfrac{0,15}{-0,11} = -1,3636.\quad M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1,1818 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1,3636 & 0 & 0 & 1 \end{pmatrix}$.

Помножимо матрицю A зліва на M_1, отримаємо матрицю A_1:

$$M_1 \cdot A = A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1,1818 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1,3636 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0,11 & 1,13 & -0,17 & 0,18 \\ 0,13 & -1,17 & 0,18 & 0,14 \\ 0,11 & -1,05 & -0,17 & -0,15 \\ 0,15 & -0,05 & 0,18 & -0,11 \end{pmatrix} = \begin{pmatrix} 0,11 & 1,13 & -0,17 & 0,18 \\ 0 & -2,5055 & 0,3809 & -0,0727 \\ 0 & -2,18 & 0 & -0.33 \\ 0 & -1,5909 & 0,4118 & -0,3555 \end{pmatrix}.$$

Утворимо матрицю M_2: $m_{32} = \dfrac{a_{32}^{(1)}}{-a_{22}^{(1)}} = \dfrac{-2,18}{2,5055} = -0,8701;\ m_{42} = \dfrac{a_{42}^{(1)}}{-a_{22}^{(1)}} = \dfrac{-1,5909}{2,5055} = -0,6350.$

$$M_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -0,8701 & 1 & 0 \\ 0 & -0,6350 & 0 & 1 \end{pmatrix}.$$

Помножимо матрицю A_1 зліва на M_2, отримаємо матрицю A_2:

$$M_2 \cdot A_1 = A_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -0,8701 & 1 & 0 \\ 0 & -0,6350 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0,11 & 1,13 & -0,17 & 0,18 \\ 0 & -2,5055 & 0,3809 & -0,0727 \\ 0 & -2,18 & 0 & -0.33 \\ 0 & -1,5909 & 0,4118 & -0,3555 \end{pmatrix} = \begin{pmatrix} 0,11 & 1,13 & -0,17 & 0,18 \\ 0 & -2,5055 & 0,3809 & -0,0727 \\ 0 & 0 & -0,3314 & -0.2667 \\ 0 & 0 & 0,1699 & -0,3093 \end{pmatrix}.$$

Утворимо матрицю M_3: $m_{43} = \dfrac{a_{43}^{(2)}}{-a_{33}^{(2)}} = \dfrac{0,1699}{0,3314} = 0,5127.\quad M_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0,5128 & 1 \end{pmatrix}$.

Помножимо матрицю A_2 зліва на M_3, отримаємо матрицю A_3, яка буде також матрицею U:

$$M_3 \cdot A_2 = A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0{,}5128 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0{,}11 & 1{,}13 & -0{,}17 & 0{,}18 \\ 0 & -2{,}5055 & 0{,}3809 & -0{,}0727 \\ 0 & 0 & -0{,}3314 & -0.2667 \\ 0 & 0 & 0{,}1699 & -0{,}3093 \end{pmatrix} = \begin{pmatrix} 0{,}11 & 1{,}13 & -0{,}17 & 0{,}18 \\ 0 & -2{,}5055 & 0{,}3809 & -0{,}0727 \\ 0 & 0 & -0{,}3314 & -0.2667 \\ 0 & 0 & 0 & -0{,}446 \end{pmatrix} = U .$$

Нижню трикутну матрицю L отримуємо з ненульових стовпців матриць $M_1, \cdot M_2, M_3$ (змінюємо знаки!):

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1{,}1818 & 1 & 0 & 0 \\ 1 & 0{,}8701 & 1 & 0 \\ 1{,}3636 & 0{,}635 & -0{,}5128 & 1 \end{pmatrix}.$$

Таким чином:

$$U = \begin{pmatrix} 0{,}11 & 1{,}13 & -0{,}17 & 0{,}18 \\ 0 & -2{,}5055 & 0{,}3809 & -0{,}0727 \\ 0 & 0 & -0{,}3314 & -0{,}2667 \\ 0 & 0 & 0 & -0{,}4460 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1{,}1818 & 1 & 0 & 0 \\ 1 & 0{,}8701 & 1 & 0 \\ 1{,}3636 & 0{,}635 & -0{,}5128 & 1 \end{pmatrix}.$$

Розв'яжемо систему $L\vec{y} = \vec{b}$:

$$\begin{cases} y_1 = 1{,}00 \\ 1{,}1818 y_1 + y_2 = 0{,}13 \\ 1 y_1 + 0{,}8701 y_2 + y_3 = 0{,}11 \\ 1{,}3636 y_1 + 0{,}635 y_2 - 0{,}5128 y_3 + y_4 = 1{,}00 \end{cases}$$

$$y_1 = 1{,}00$$
$$y_2 = 0{,}13 - 1{,}1818 y_1 = 0{,}13 - 1{,}1818 \cdot 1 = -1{,}0518$$
$$y_3 = 0{,}11 - 1 y_1 - 0{,}8701 y_2 = 0{,}11 - 1 \cdot 1{,}00 - 0{,}8701 \cdot (-1{,}0518) = 0{,}0252$$
$$y_4 = 1{,}00 - 1{,}3636 y_1 - 0{,}635 y_2 + 0{,}5128 y_3 = 1{,}00 - 1{,}3636 \cdot 1{,}00 - 0{,}635 \cdot (-1{,}0518) + 0{,}5128 \cdot 0{,}0252 = 0{,}3172$$

Розв'яжемо систему $U\vec{x} = \vec{y}$:

$$\begin{cases} 0{,}11 x_1 + 1{,}13 x_2 - 0{,}17 x_3 + 0{,}18 x_4 = 1{,}00 \\ -2{,}5055 x_2 + 0{,}3809 x_3 - 0{,}0727 x_4 = -1{,}0518 \\ -0{,}3314 x_3 - 0{,}2667 x_4 = 0{,}0252 \\ -0{,}4460 x_4 = 0{,}3172 \end{cases}$$

$$x_4 = \frac{0{,}3172}{-0{,}4460} = -0{,}7111$$
$$x_3 = \frac{0{,}0252 + 0{,}2667 x_4}{-0{,}3314} = \frac{0{,}0252 + 0{,}2667 \cdot (-0{,}7112)}{-0{,}3314} = 0{,}4962$$
$$x_2 = \frac{-1{,}0518 - 0{,}3809 x_3 + 0{,}0727 x_4}{-2{,}5055} = \frac{-1{,}0518 - 0{,}3809 \cdot 0{,}4962 + 0{,}0727 \cdot (-0{,}7111)}{-2{,}5055} = 0{,}5159$$
$$x_1 = \frac{1{,}00 - 1{,}13 x_2 + 0{,}17 x_3 - 0{,}18 x_4}{0{,}11} = \frac{1{,}00 - 1{,}13 \cdot 0{,}5159 + 0{,}17 \cdot 0{,}4962 - 0{,}18 \cdot (-0{,}7111)}{0{,}11} = 5{,}7217$$

Відповідь: $x_1 = 5{,}722$; $x_2 = 0{,}516$; $x_3 = 0{,}496$; $x_4 = -0{,}711$.

Наведемо фрагменти програм, що реалізують розглянуті алгоритми. Вхідними даними будуть вимірність системи n (n=4), матриця коефіцієнтів системи A та вектор вільних членів $\vec{b}$.

Метод Гауса. Формуємо розширену матрицю коефіцієнтів:

```
for i:=1 to n do
        begin
                for j:=1 to n do C[i,j]:=A[i,j];
                C[i,n+1]:=b[i];
        end;
```

Прямий хід:

```
for k:=1 to n do
        begin
                for j:=n+1 downto k do C[k,j]:=C[k,j]/C[k,k];
                for i:=k+1 to n do
                        for j:=n+1 downto k do C[i,j]:=C[i,j]-C[i,k]*C[k,j];
        end;
```

Зворотний хід:

```
for i:=n downto 1 do
        begin
                x[i]:=C[i,n+1];
                for j:=i+1 to n do x[i]:=x[i]-C[i,j]*x[j];
        end;
```

Здійснимо перевірку отриманих розв'язків. Підставимо їх в початкову систему (3.1) та обчислимо нев'язку:

```
writeln('Нев''язка системи:');
for i:=1 to n do
        begin
                e:=0;
                for j:=1 to n do e:=e+A[i,j]*x[j];
                e:=abs(e-b[i]);
                writeln('   e[',i,']=',e);
        end;
```

Результати роботи програми приведені на рис. 3.3.

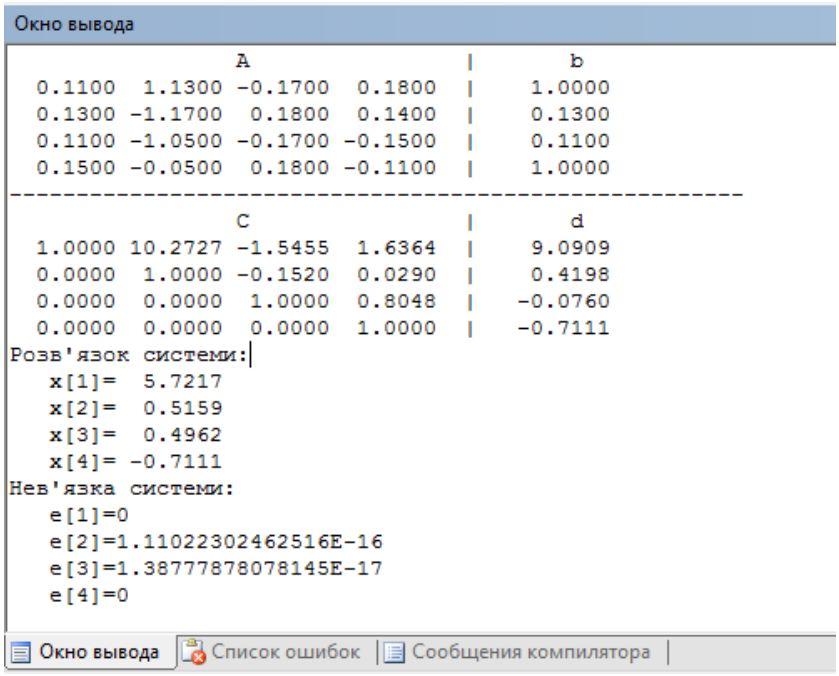


Рисунок 3.3 – Розв'язок СЛАР методом Гауса

Метод LU-розкладання. На кожному кроці обчислення матриці U нам приходиться виконувати множення матриць M та A. Тому доречно таке множення оформити у вигляді окремої процедури. Складемо процедуру обчислення добутку двох квадратних матриць:

```
…….
type  t=array[1..n,1..n] of real;
……
procedure MultMatr(Q,R:t; var Z:t);
var i,j,k:byte;
begin
        for i:=1 to n do
                for j:=1 to n do
                        begin
                                Z[i,j]:=0;
                                for k:=1 to n do Z[i,j]:=Z[i,j]+Q[i,k]*R[k,j];
                        end;
end;
```

Після введення вхідних даних, сформуємо одиничну матрицю $E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, на базі якої утворюються матриці M та L:

```
for i:=1 to n do
        for j:=1 to n do
                if i=j then E[i,j]:=1
                     else E[i,j]:=0;
U:=A; L:=E;
```

Обчислюємо матриці U та L:

```
for k:=1 to n-1 do
        begin
                {Формуємо матрицю M та L}
                M:=E;
                for i:=k+1 to n do
                                        begin
                                                M[i,k]:=-U[i,k]/U[k,k];
                                                L[i,k]:=-M[i,k];
                                        end;
                {Шукаємо матрицю U}
                MultMatr(M,U,U);
        end;
```

Шукаємо розв'язок системи $L\vec{y} = \vec{b}$:

```
for i:=1 to n do
                begin
                        y[i]:=b[i];
                        for j:=1 to i-1 do y[i]:=y[i]-L[i,j]*y[j];
                end;
```

```
writeln('Розв''язок системи  L*y=b');
for i:=1 to n do writeln('   y[',i,']=',y[i]:8:4);
```

Шукаємо розв'язок системи $U\vec{x} = \vec{y}$:

```
for i:=n downto 1 do
                    begin
                        x[i]:=y[i];
                        for j:=i+1 to n do x[i]:=x[i]-U[i,j]*x[j];
                        x[i]:=x[i]/U[i,i];
                    end;
 writeln('Розв''язок системи  U*x=y');
 for i:=1 to n do writeln('   x[',i,']=',x[i]:8:4);
```

Здійснимо перевірку отриманих розв’язків. Підставимо їх в початкову систему (3.1) та обчислимо нев’язку:

```
writeln('Нев''язка системи:');
for i:=1 to n do
                begin
                    eps:=0;
                    for j:=1 to n do eps:=eps+A[i,j]*x[j];
                    eps:=abs(eps-b[i]);
                    writeln('   e[',i,']=',eps);
                end;
```

Результати роботи програми приведені на рис. 3.4.

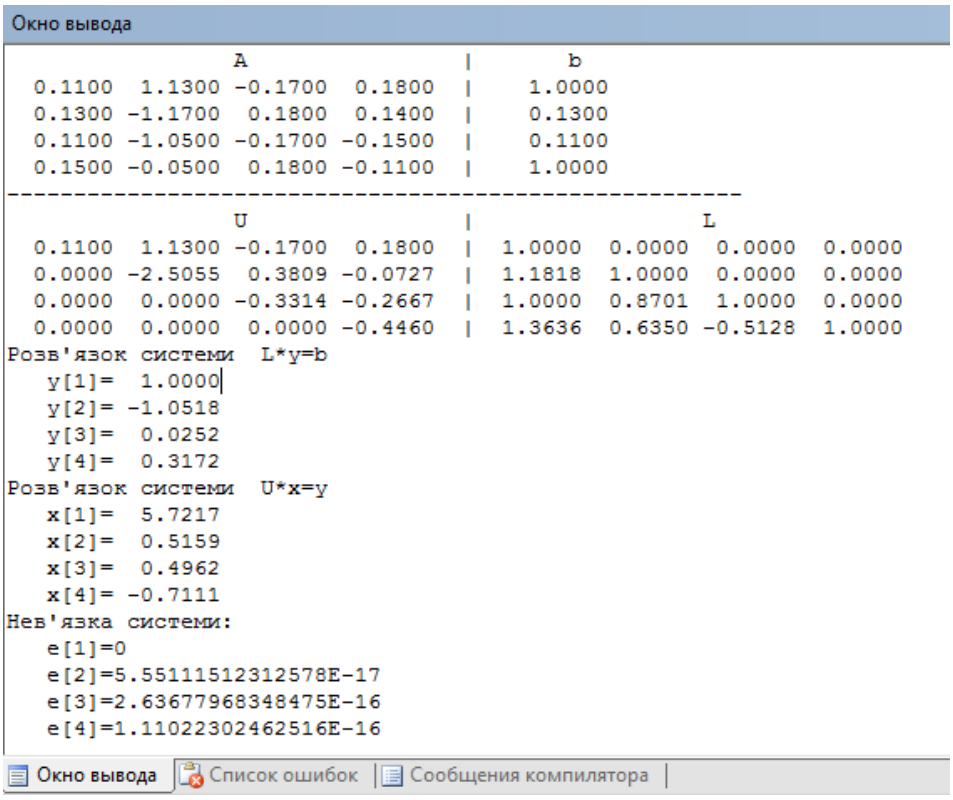


Рисунок 3.4 – Розв’язок СЛАР методом *LU*-розкладання

Контрольні запитання

1. На які дві групи поділяються методи розв’язання СЛАР?
2. Сутність методу Гауса. Поясніть поняття "прямий хід" та "зворотний хід".
3. Алгоритм схеми єдиного ділення.
4. Які схеми реалізації прямого ходу вам ще відомі?
5. Алгоритм методу *LU*-розкладання та випадки його використання.

Перелік рекомендованої літератури

1. Фельдман Л.П. Чисельні методи в інформатиці / Л.П. Фельдман, А.І. Петренко, О.А. Дмитрієва. –К.: BHV, 2006. – 480 с.
2. Пирумов У. Г. Численные методы: учеб. пособие для студ. втузов / У.Г. Пирумов. – 4-е изд., стереотип. – М.: Дрофа, 2007. – 221, [3] с.: ил.
3. Численные методы. Сборник задач: учеб. пособие для вузов / В.Ю. Гидаспов, И.Э. Иванов, Д.Л. Ревизников и др.; под ред. У.Г. Пирумова. – М.: Дрофа, 2007. – 144 с.: ил.

ЛАБОРАТОРНА РОБОТА № 4
РОЗВ'ЯЗУВАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ ІТЕРАЦІЙНИМИ МЕТОДАМИ

Мета: опанувати ітераційними методами розв'язування систем лінійних алгебраїчних рівнянь – методом простої ітерації та методом Зейделя.

Теоретичні відомості

Ітераційні методи використовують зазвичай для систем великої вимірності ($n \approx 100$), коли використання прямих методів є недоцільним через необхідність виконувати занадто велику кількість арифметичних операцій. Прямі методи дозволяють отримати точний розв'язок, але він не може бути досягнутий через накопичення похибок заокруглень при виконанні арифметичних операцій. Отриманий розв'язок може значно відрізнятися від точного. Ітераційними методами в таких випадках отримують розв'язки з більшою точністю ніж прямі.

Ітераційні методи передбачають побудову обчислювального (*ітераційного*) *процесу*, який дозволяє отримати послідовності наближень (ітерацій) розв'язку СЛАР, починаючи з деякого довільного наближення:

$\vec{x}^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \dots \\ x_n^{(0)} \end{pmatrix}$, $\vec{x}^{(1)} = \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \dots \\ x_n^{(1)} \end{pmatrix}$, $\vec{x}^{(2)} = \begin{pmatrix} x_1^{(2)} \\ x_2^{(2)} \\ \dots \\ x_n^{(2)} \end{pmatrix}$, …, $\vec{x}^{(s)} = \begin{pmatrix} x_1^{(s)} \\ x_2^{(s)} \\ \dots \\ x_n^{(s)} \end{pmatrix}$, які збігаються до точного $\lim\limits_{s\to\infty} \vec{x}^{(s)} = \vec{x}$.

Розглянемо систему лінійних алгебраїчних рівнянь із *n*-невідомими:

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n = b_2, \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n = b_n \end{array}\right\}, \qquad (4.1)$$

де $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$ - матриця коефіцієнтів системи; $\vec{b} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$ - вектор вільних членів; $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$ - вектор невідомих.

Для побудови ітераційного процесу систему (4.1) приведемо до вигляду:

$$\left.\begin{array}{l} x_1 = \beta_1 + \alpha_{11}x_1 + \alpha_{12}x_2 + \alpha_{13}x_3 + \dots + \alpha_{1n}x_n, \\ x_2 = \beta_2 + \alpha_{21}x_1 + \alpha_{22}x_2 + \alpha_{23}x_3 + \dots + \alpha_{2n}x_n, \\ x_3 = \beta_3 + \alpha_{31}x_1 + \alpha_{32}x_2 + \alpha_{23}x_3 + \dots + \alpha_{3n}x_n, \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ x_n = \beta_n + \alpha_{n1}x_1 + \alpha_{n2}x_2 + \alpha_{n3}x_3 + \dots + \alpha_{n,n-1}x_{n-1}, \end{array}\right\}, \qquad (4.2)$$

або $\vec{x} = \vec{\beta} + \alpha\vec{x}$. Така система називається *зведеною*, її можна отримати, наприклад, якщо кожне *i*-рівняння системи (4.1) розв'язати відносно змінної x_i:

$$\left.\begin{array}{l}
x_1 = \frac{b_1}{a_{11}} \qquad\quad - \frac{a_{12}}{a_{11}} x_2 - \frac{a_{13}}{a_{11}} x_3 - \; \dots \qquad\qquad - \frac{a_{1n}}{a_{11}} x_n, \\
x_2 = \frac{b_2}{a_{22}} - \frac{a_{21}}{a_{22}} x_1 \qquad\quad - \frac{a_{23}}{a_{22}} x_3 - \; \dots \qquad\qquad - \frac{a_{2n}}{a_{22}} x_n, \\
x_3 = \frac{b_3}{a_{33}} - \frac{a_{31}}{a_{33}} x_1 - \frac{a_{32}}{a_{33}} x_2 - \qquad \dots \qquad\qquad - \frac{a_{3n}}{a_{33}} x_n, \\
\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\
x_n = \frac{b_n}{a_{nn}} - \frac{a_{n1}}{a_{nn}} x_1 - \frac{a_{n2}}{a_{nn}} x_2 - \frac{a_{n3}}{a_{nn}} x_3 - \dots - \frac{a_{nn-1}}{a_{nn}} x_{n-1},
\end{array}\right\}$$

Тоді: $\beta_i = \frac{b_i}{a_{ii}}$; $\alpha_{ij} = -\frac{a_{ij}}{a_{ii}}$; $i, j = 1,2,...,n$; $i \neq j$; якщо $i = j$, то $\alpha_{ii} = 0$. (4.3)

При побудові ітерацій постають питання про початок і кінець процесу обчислень. Будь який ітераційний процес починається з того, що задається початкове наближення. Як правило припускають, що

$$\vec{x}^{(0)} = 0, \text{ або } \vec{x}^{(0)} = \vec{\beta}. \qquad (4.4)$$

Так як наближений розв'язок шукається з наперед заданою точністю ε, то послідовність повинна мати скінчену кількість членів, які отримуються за скінчену кількість ітерацій.

Найпростіша умова закінчення ітераційного процесу:

$$\max_{1 \le i \le n} \left| x_i^{(s+1)} - x_i^{(s)} \right| < \varepsilon. \qquad (4.5)$$

Тобто, обчислення продовжують до тих пір, доки абсолютна величина різниці між попереднім й наступним наближеннями не стане менше деякої наперед заданої точності ε:

Для дослідження збіжності ітераційного процесу користуються *теоремою про достатню умову збіжності:*

> Якщо для зведеної системи (4.2) будь-яка канонічна норма матриці α менше одиниці $\|\alpha\| < 1$, то ітераційний процес збігається до єдиного розв'язку цієї системи, незалежно від вибору початкового наближення.

Нормою матриці α називається дійсне число $\|\alpha\|$, що задовольняє певним умовам, найбільш важливі з яких такі:

- $\|\alpha\| \geq 0$, причому $\|\alpha\| = 0$ тільки коли α - нульова матриця;
- $\|c \cdot \alpha\| = |c| \cdot \|\alpha\|$, де c – дійсне число $c \in R$.

Норма називається *канонічною*, якщо $\|\alpha\| \geq \left|\alpha_{ij}\right|$, тобто вона не менше модуля будь-якого елементу матриці α.

На практиці зазвичай користуються трьома канонічними нормами:

- $\|\alpha\|_1 = \max_{1 \le i \le n} \sum_{j=1}^{n} \left|\alpha_{ij}\right|$ - додаються за модулем всі рядки матриці і максимальна

сума обирається нормою;

- $\|\alpha\|_2 = \max\limits_{1\le j\le n}\sum\limits_{i=1}^{n}|\alpha_{ij}|$ - додаються за модулем всі стовпці матриці і максимальна сума обирається нормою;

- $\|\alpha\|_3 = \sqrt{\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}|\alpha_{ij}|^2}$ - додаються квадрати всіх елементів матриці і корінь з цієї суми буде нормою.

Умова збіжності по відношенню до матриці A початкової системи (4.1) набуває такого змісту: процес ітерації буде збіжним, якщо модулі діагональних елементів матриці A будуть більші за суму модулів її сторонніх елементів (в рядку або стовпці):

$$|a_{ii}| > \sum_{\substack{j=1,\\ j\neq i}}^{n}|a_{ij}| \text{ або } |a_{jj}| > \sum_{\substack{i=1,\\ i\neq j}}^{n}|a_{ij}| \ (i,\ j=1,2,...,n). \qquad (4.6)$$

Виконання цих умов можна досягти, якщо застосувати правила лінійного комбінування рівнянь системи.

Метод простої ітерації. Кожне наступне наближення i-ой невідомої $x_i^{(s+1)}$, i=1,2,…, n визначається за допомогою системи рівнянь (4.2), в яких всі доданки правої частини беруться з попередній s-ітерації:

$$\left.\begin{aligned}
x_1^{(s+1)} &= \beta_1 + \alpha_{11}x_1^{(s)} + \alpha_{12}x_2^{(s)} + \alpha_{13}x_3^{(s)} + \ldots + \alpha_{1n}x_n^{(s)},\\
x_2^{(s+1)} &= \beta_2 + \alpha_{21}x_1^{(s)} + \alpha_{22}x_2^{(s)} + \alpha_{23}x_3^{(s)} + \ldots + \alpha_{2n}x_n^{(s)},\\
x_3^{(s+1)} &= \beta_3 + \alpha_{31}x_1^{(s)} + \alpha_{32}x_2^{(s)} + \alpha_{33}x_3^{(s)} + \ldots + \alpha_{3n}x_n^{(s)},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n^{(s+1)} &= \beta_n + \alpha_{n1}x_1^{(s)} + \alpha_{n2}x_2^{(s)} + \alpha_{n3}x_3^{(s)} + \ldots + \alpha_{n,n}x_n^{(s)}.
\end{aligned}\right\} \qquad (4.7)$$

Або система (4.7) в компактній формі:

$$x_i^{(s+1)} = \beta_i + \sum_{j=1}^{n}\alpha_{ij}x_j^{(s)}; \quad i=1,2,...,n; \ \ s=0,1,2,... \qquad (4.8)$$

Блок-схема алгоритму метода простої ітерації приведена на рис. 4.1.

Метод Зейделя являє собою деяку модифікацію метода простої ітерації. А саме, при обчисленні (s+1)-ого наближення невідомої x_i враховуються вже обчислені раніше значення невідомих на поточній ітерації $x_1^{(s+1)}, x_2^{(s+1)}, ..., x_{i-1}^{(s+1)}$:

$$\left.\begin{aligned}
x_1^{(s+1)} &= \beta_1 + \alpha_{11}x_1^{(s)} + \alpha_{12}x_2^{(s)} + \alpha_{13}x_3^{(s)} + \ldots + \alpha_{1n}x_n^{(s)},\\
x_2^{(s+1)} &= \beta_2 + \alpha_{21}x_1^{(s+1)} + \alpha_{22}x_2^{(s)} + \alpha_{23}x_3^{(s)} + \ldots + \alpha_{2n}x_n^{(s)},\\
x_3^{(s+1)} &= \beta_3 + \alpha_{31}x_1^{(s+1)} + \alpha_{32}x_2^{(s+1)} + \alpha_{33}x_3^{(s)} + \ldots + \alpha_{3n}x_n^{(s)},\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n^{(s+1)} &= \beta_n + \alpha_{n1}x_1^{(s+1)} + \alpha_{n2}x_2^{(s+1)} + \alpha_{n3}x_3^{(s+1)} + \ldots + \alpha_{n,n}x_n^{(s)}.
\end{aligned}\right\} \qquad (4.9)$$

або $$x_i^{(s+1)} = \beta_i + \sum_{j=1}^{i-1}\alpha_{ij}x_j^{(s+1)} + \sum_{j=i}^{n}\alpha_{ij}x_j^{(s)}; i=1,2,...,n; \ \ s=0,1,2,... \qquad (4.10)$$

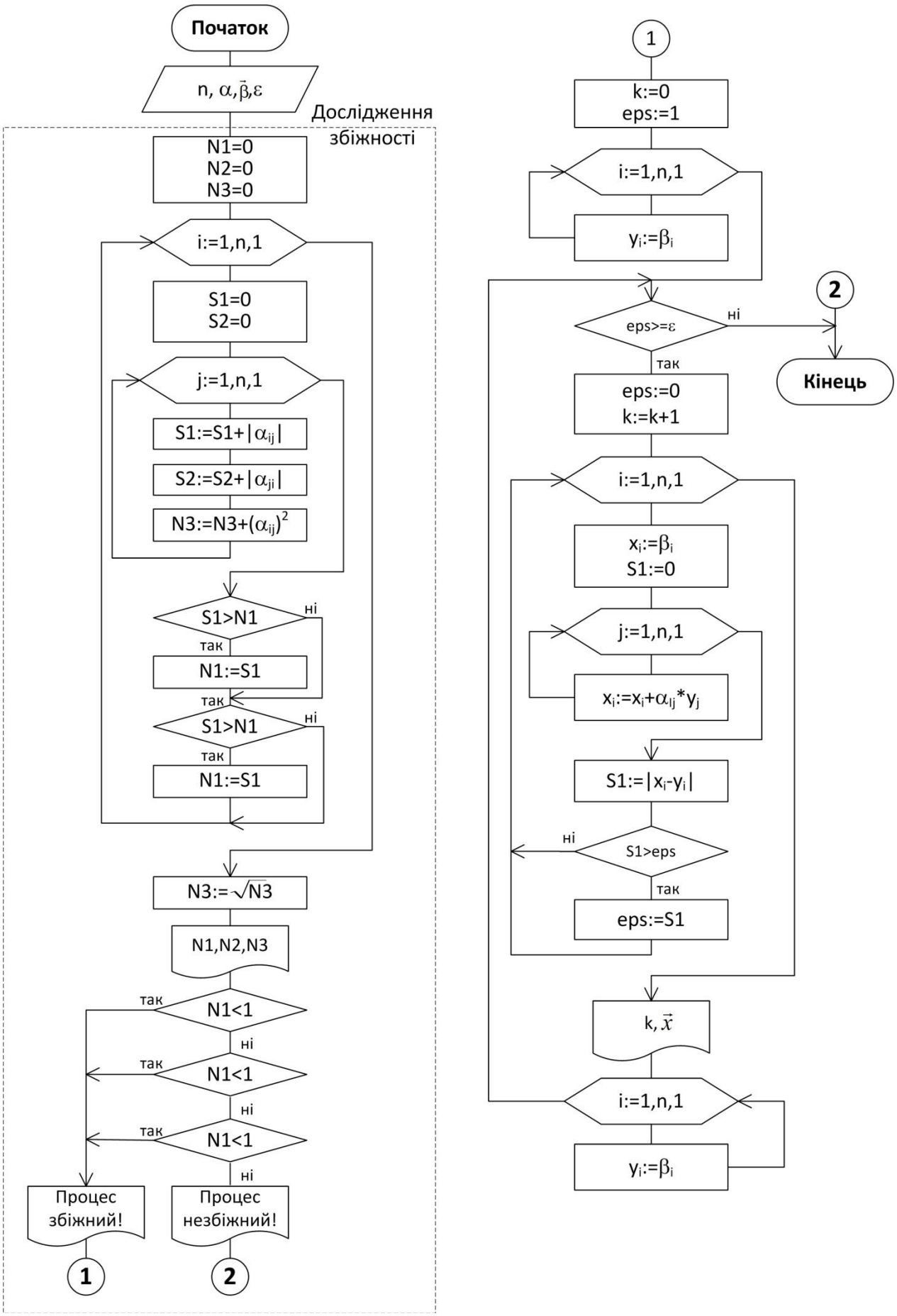


Рисунок 4.1 – Блок-схема методу простої ітерації

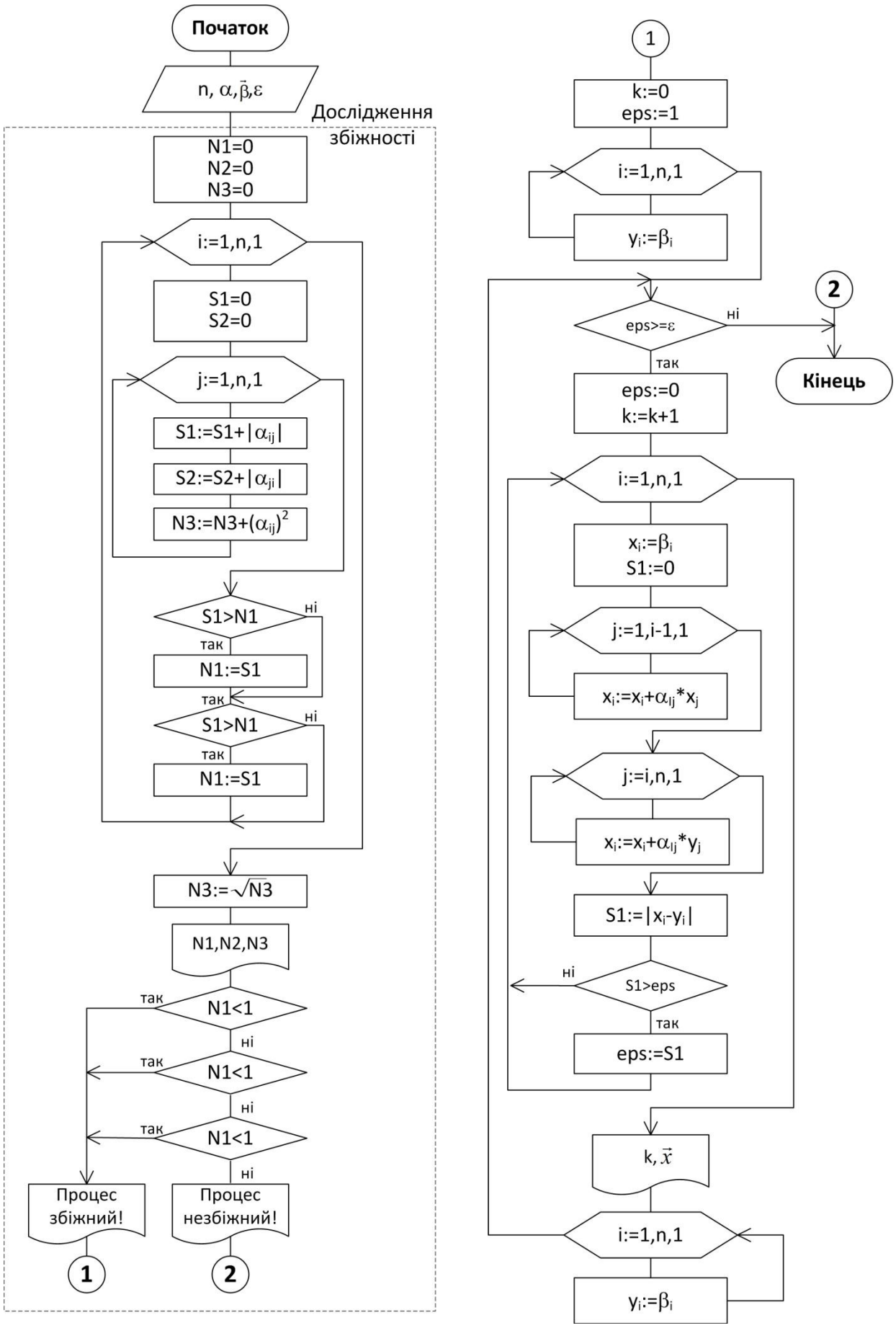


Рисунок 4.2 – Блок-схема методу Зейделя

Ітераційний процес закінчують при виконанні умови (4.5). Теорема про збіжність ітераційного процесу залишається вірною й для методу Зейделя. Блок-схема алгоритму метода Зейделя приведена на рис. 4.2.

Завдання

Розв'язати зведену СЛАР методами простої ітерації та Зейделя з точністю $\varepsilon = 0{,}001$. Порівняти швидкість збіжності обох методів.

1. $\begin{cases} x_1 = 2{,}13 + 0{,}23x_2 - 0{,}44x_3 - 0{,}05x_4 \\ x_2 = -0{,}18 + 0{,}24x_1 - 0{,}31x_3 + 0{,}15x_4 \\ x_3 = 1{,}44 + 0{,}06x_1 + 0{,}15x_2 - 0{,}23x_4 \\ x_4 = 2{,}42 + 0{,}72x_1 - 0{,}08x_2 - 0{,}05x_3 \end{cases}$

2. $\begin{cases} x_1 = -0{,}22 + 0{,}52x_2 + 0{,}08x_3 + 0{,}13x_4 \\ x_2 = 1{,}8 + 0{,}07x_1 - 0{,}05x_3 + 0{,}41x_4 \\ x_3 = -1{,}3 + 0{,}04x_1 + 0{,}42x_2 - 0{,}07x_4 \\ x_4 = 0{,}33 + 0{,}17x_1 + 0{,}18x_2 - 0{,}13x_3 \end{cases}$

3. $\begin{cases} x_1 = -1{,}71 + 0{,}31x_2 - 0{,}18x_3 + 0{,}22x_4 \\ x_2 = 0{,}62 - 0{,}21x_1 + 0{,}33x_3 + 0{,}22x_4 \\ x_3 = -0{,}89 + 0{,}32x_1 - 0{,}18x_2 - 0{,}19x_4 \\ x_4 = 0{,}94 + 0{,}12x_1 + 0{,}28x_2 - 0{,}14x_3 \end{cases}$

4. $\begin{cases} x_1 = -1{,}3 + 0{,}02x_2 - 0{,}62x_3 + 0{,}08x_4 \\ x_2 = 1{,}1 + 0{,}28x_1 + 0{,}33x_3 - 0{,}07x_4 \\ x_3 = -1{,}7 + 0{,}09x_1 + 0{,}13x_2 + 0{,}28x_4 \\ x_4 = 1{,}5 + 0{,}19x_1 - 0{,}23x_2 + 0{,}8x_3 \end{cases}$

5. $\begin{cases} x_1 = 1{,}21 + 0{,}27x_2 - 0{,}22x_3 - 0{,}18x_4 \\ x_2 = -0{,}33 - 0{,}21x_1 - 0{,}45x_3 + 0{,}18x_4 \\ x_3 = -0{,}48 + 0{,}12x_1 + 0{,}13x_2 + 0{,}18x_4 \\ x_4 = -0{,}17 + 0{,}33x_1 - 0{,}05x_2 + 0{,}06x_3 \end{cases}$

6. $\begin{cases} x_1 = -1{,}2 + 0{,}17x_2 - 0{,}33x_3 + 0{,}18x_4 \\ x_2 = 0{,}33 + 0{,}18x_1 + 0{,}43x_3 - 0{,}08x_4 \\ x_3 = 0{,}48 + 0{,}22x_1 + 0{,}18x_2 + 0{,}07x_4 \\ x_4 = -1{,}2 + 0{,}08x_1 + 0{,}07x_2 + 0{,}21x_3 \end{cases}$

7. $\begin{cases} x_1 = -0{,}81 - 0{,}07x_2 + 0{,}38x_3 - 0{,}21x_4 \\ x_2 = -0{,}64 - 0{,}22x_1 + 0{,}11x_3 + 0{,}33x_4 \\ x_3 = 1{,}71 + 0{,}51x_1 - 0{,}07x_2 - 0{,}11x_4 \\ x_4 = -1{,}21 + 0{,}33x_1 - 0{,}41x_2 \end{cases}$

8. $\begin{cases} x_1 = 0{,}43 - 0{,}05x_2 + 0{,}22x_3 - 0{,}33x_4 \\ x_2 = -1{,}8 + 0{,}22x_1 - 0{,}08x_3 + 0{,}07x_4 \\ x_3 = -0{,}8 + 0{,}33x_1 + 0{,}13x_2 - 0{,}05x_4 \\ x_4 = 1{,}7 + 0{,}08x_1 + 0{,}17x_2 + 0{,}29x_3 \end{cases}$

9. $\begin{cases} x_1 = 2{,}7 + 0{,}22x_2 - 0{,}11x_3 + 0{,}31x_4 \\ x_2 = -1{,}5 + 0{,}38x_1 - 0{,}12x_3 + 0{,}22x_4 \\ x_3 = 1{,}2 + 0{,}11x_1 + 0{,}23x_2 - 0{,}51x_4 \\ x_4 = -0{,}17 + 0{,}17x_1 - 0{,}21x_2 + 0{,}31x_3 \end{cases}$

10. $\begin{cases} x_1 = 0{,}11 + 0{,}22x_2 - 0{,}33x_3 + 0{,}07x_4 \\ x_2 = -0{,}33 + 0{,}45x_1 - 0{,}23x_3 + 0{,}07x_4 \\ x_3 = 0{,}85 + 0{,}11x_1 - 0{,}08x_2 + 0{,}18x_4 \\ x_4 = -1{,}7 + 0{,}08x_1 + 0{,}09x_2 + 0{,}33x_3 \end{cases}$

11. $\begin{cases} x_1 = -0{,}51 - 0{,}08x_2 + 0{,}11x_3 - 0{,}18x_4 \\ x_2 = 1{,}17 + 0{,}18x_1 + 0{,}52x_3 + 0{,}21x_4 \\ x_3 = -1{,}02 + 0{,}13x_1 + 0{,}31x_2 - 0{,}21x_4 \\ x_4 = -0{,}28 + 0{,}08x_1 - 0{,}33x_2 + 0{,}28x_3 \end{cases}$

12. $\begin{cases} x_1 = 2{,}42 - 0{,}16x_2 - 0{,}08x_3 + 0{,}15x_4 \\ x_2 = 1{,}43 + 0{,}16x_1 + 0{,}11x_3 - 0{,}21x_4 \\ x_3 = -0{,}16 + 0{,}05x_1 - 0{,}08x_2 + 0{,}34x_4 \\ x_4 = 1{,}62 + 0{,}12x_1 + 0{,}14x_2 - 0{,}18x_3 \end{cases}$

13. $\begin{cases} x_1 = -2{,}17 - 0{,}06x_2 - 0{,}12x_3 + 0{,}14x_4 \\ x_2 = 1{,}4 + 0{,}04x_1 + 0{,}08x_3 + 0{,}11x_4 \\ x_3 = -2{,}1 + 0{,}34x_1 + 0{,}08x_2 + 0{,}14x_4 \\ x_4 = -0{,}8 + 0{,}11x_1 + 0{,}12x_2 - 0{,}03x_3 \end{cases}$

14. $\begin{cases} x_1 = 1{,}34 + 0{,}08x_2 - 0{,}23x_3 + 0{,}32x_4 \\ x_2 = -2{,}33 + 0{,}16x_1 + 0{,}18x_3 + 0{,}16x_4 \\ x_3 = 0{,}34 + 0{,}15x_1 + 0{,}12x_2 - 0{,}18x_4 \\ x_4 = 0{,}63 + 0{,}25x_1 + 0{,}21x_2 - 0{,}16x_3 \end{cases}$

15. $\begin{cases} x_1 = -1{,}2 + 0{,}08x_2 - 0{,}03x_3 - 0{,}04x_4 \\ x_2 = 0{,}81 + 0{,}31x_2 + 0{,}27x_3 - 0{,}08x_4 \\ x_3 = -0{,}92 + 0{,}33x_1 - 0{,}07x_2 + 0{,}21x_4 \\ x_4 = 0{,}17 + 0{,}11x_1 + 0{,}03x_2 + 0{,}58x_3 \end{cases}$

16. $\begin{cases} x_1 = 2{,}43 + 0{,}18x_2 + 0{,}33x_3 + 0{,}16x_4 \\ x_2 = -1{,}12 + 0{,}32x_1 + 0{,}23x_3 - 0{,}05x_4 \\ x_3 = 0{,}43 + 0{,}16x_1 - 0{,}08x_2 - 0{,}12x_4 \\ x_4 = 0{,}83 + 0{,}09x_1 + 0{,}22x_2 - 0.13x_3 \end{cases}$

17. $\begin{cases} x_1 = 1{,}24 - 0{,}23x_2 + 0{,}25x_3 - 0{,}16x_4 \\ x_2 = 0{,}89 + 0{,}14x_1 - 0{,}18x_3 + 0{,}24x_4 \\ x_3 = 1{,}15 + 0{,}33x_1 + 0{,}03x_2 - 0{,}32x_4 \\ x_4 = -0{,}57 + 0{,}12x_1 - 0{,}05x_2 + 0{,}15x_3 \end{cases}$

18. $\begin{cases} x_1 = 1{,}42 + 0{,}34x_2 + 0{,}23x_3 - 0{,}06x_4 \\ x_2 = -0{,}66 + 0{,}11x_1 - 0{,}18x_3 + 0{,}36x_4 \\ x_3 = 1{,}08 + 0{,}23x_1 - 0{,}12x_2 - 0{,}35x_4 \\ x_4 = 1{,}72 + 0{,}12x_1 + 0{,}12x_2 - 0{,}47x_3 \end{cases}$

19. $\begin{cases} x_1 = 1{,}21 - 0{,}14x_2 + 0{,}06x_3 - 0{,}12x_4 \\ x_2 = -0{,}72 + 0{,}12x_1 + 0{,}32x_3 - 0{,}18x_4 \\ x_3 = -0{,}58 + 0{,}08x_1 - 0{,}12x_2 + 0{,}32x_4 \\ x_4 = 1{,}56 + 0{,}25x_1 + 0{,}22x_2 + 0{,}14x_3 \end{cases}$

20. $\begin{cases} x_1 = 0{,}67 - 0{,}23x_2 + 0{,}11x_3 - 0{,}06x_4 \\ x_2 = -0{,}88 + 0{,}18x_1 + 0{,}12x_3 - 0{,}33x_4 \\ x_3 = -0{,}18 + 0{,}12x_1 + 0{,}32x_2 + 0{,}07x_4 \\ x_4 = 1{,}44 + 0{,}05x_1 - 0{,}11x_2 + 0{,}09x_3 \end{cases}$

Послідовність виконання лабораторної роботи

1. ***Методом простої ітерації*** розв'язати СЛАР з точністю $\varepsilon = 0{,}05$:

$$\begin{cases} x_1 = 2{,}15 + 0{,}32x_1 - 0{,}05x_2 + 0{,}11x_3 - 0{,}08x_4, \\ x_2 = -0{,}83 + 0{,}11x_1 + 0{,}16x_2 - 0{,}28x_3 - 0{,}06x_4, \\ x_3 = 1{,}16 + 0{,}08x_1 - 0{,}15x_2 + 0{,}12x_4, \\ x_4 = 0{,}44 - 0{,}21x_1 + 0{,}13x_2 - 0{,}27x_3. \end{cases}$$

Сформуємо матрицю коефіцієнтів α і вектор вільних членів $\vec{\beta}$:

$$\alpha = \begin{pmatrix} 0{,}32 & -0{,}05 & 0{,}11 & -0{,}08 \\ 0{,}11 & 0{,}16 & -0{,}28 & -0{,}06 \\ 0{,}08 & -0{,}15 & 0 & 0{,}12 \\ -0{,}21 & 0{,}13 & -0{,}27 & 0 \end{pmatrix}, \quad \vec{\beta} = \begin{pmatrix} 2{,}15 \\ -0{,}83 \\ 1{,}16 \\ 0{,}44 \end{pmatrix}.$$

Визначимо збіжність ітераційного процесу для заданої системи. Знайдемо норми матриці α:

$$\|\alpha\|_1 = \max\begin{Bmatrix} |\alpha_{11}| + |\alpha_{12}| + |\alpha_{13}| + |\alpha_{14}| \\ |\alpha_{21}| + |\alpha_{22}| + |\alpha_{23}| + |\alpha_{24}| \\ |\alpha_{31}| + |\alpha_{32}| + |\alpha_{33}| + |\alpha_{34}| \\ |\alpha_{41}| + |\alpha_{42}| + |\alpha_{43}| + |\alpha_{44}| \end{Bmatrix} = \begin{Bmatrix} 0{,}32 + 0{,}05 + 0{,}11 + 0{,}08 \\ 0{,}11 + 0{,}16 + 0{,}28 + 0{,}06 \\ 0{,}08 + 0{,}15 + 0 + 0{,}12 \\ 0{,}21 + 0{,}13 + 0{,}27 + 0 \end{Bmatrix} = \max\begin{Bmatrix} 0{,}56 \\ 0{,}61 \\ 0{,}35 \\ 0{,}61 \end{Bmatrix} = 0{,}61;$$

$$\|\alpha\|_2 = \max\begin{Bmatrix} |\alpha_{11}| + |\alpha_{21}| + |\alpha_{31}| + |\alpha_{41}| \\ |\alpha_{12}| + |\alpha_{22}| + |\alpha_{32}| + |\alpha_{42}| \\ |\alpha_{13}| + |\alpha_{23}| + |\alpha_{33}| + |\alpha_{43}| \\ |\alpha_{14}| + |\alpha_{24}| + |\alpha_{34}| + |\alpha_{44}| \end{Bmatrix} = \begin{Bmatrix} 0{,}32 + 0{,}11 + 0{,}08 + 0{,}21 \\ 0{,}05 + 0{,}16 + 0{,}15 + 0{,}13 \\ 0{,}11 + 0{,}28 + 0 + 0{,}27 \\ 0{,}08 + 0{,}06 + 0{,}12 + 0 \end{Bmatrix} = \max\begin{Bmatrix} 0{,}72 \\ 0{,}49 \\ 0{,}66 \\ 0{,}26 \end{Bmatrix} = 0{,}72;$$

$$\|\alpha\|_3 = \sqrt{|\alpha_{11}|^2 + |\alpha_{12}|^2 + |\alpha_{13}|^2 + |\alpha_{14}|^2 + |\alpha_{21}|^2 + |\alpha_{22}|^2 + \ldots + |\alpha_{41}|^2 + |\alpha_{42}|^2 + |\alpha_{43}|^2 + |\alpha_{44}|^2} =$$
$$= \sqrt{0{,}32^2 + 0{,}05^2 + 0{,}11^2 + 0{,}08^2 + 0{,}11^2 + 0{,}16^2 + \ldots + 0{,}21^2 + 0{,}13^2 + 0{,}27^2 + 0^2} = 0{,}4203.$$

Таким чином, перша, друга та третя канонічні норми менші одиниці, тому ітераційний процес для даної системи буде збіжним. Оберемо початкове наближення розв'язку: $\vec{x}^{(0)} = \vec{\beta} = \begin{pmatrix} 2{,}15 \\ -0{,}83 \\ 1{,}16 \\ 0{,}44 \end{pmatrix}$. Тобто, $x_1^{(0)} = 2{,}15;\ x_2^{(0)} = -0{,}83;\ x_3^{(0)} = 1{,}16;$. $x_4^{(0)} = 0{,}44$ Для знаходження першого наближення вектора розв'язку $\vec{x}^{(1)}$ початкове наближення $\vec{x}^{(0)}$ підставимо в праву частину початкової системи: $\vec{x}^{(1)} = \vec{\beta} + \alpha \cdot \vec{x}^{(0)}$:

$$x_1^{(1)} = 2{,}15 + 0{,}32 \cdot 2{,}15 - 0{,}05 \cdot (-0{,}83) + 0{,}11 \cdot 1{,}16 - 0{,}08 \cdot 0{,}44 = 2{,}9719;$$
$$x_2^{(1)} = -0{,}83 + 0{,}11 \cdot 2{,}15 + 0{,}16 \cdot (-0{,}83) - 0{,}28 \cdot 1{,}16 - 0{,}06 \cdot 0{,}44 = -1{,}0775;$$
$$x_3^{(1)} = 1{,}16 + 0{,}08 \cdot 2{,}15 - 0{,}15 \cdot (-0{,}83) + 0{,}12 \cdot 0{,}44 = 1{,}5093;$$
$$x_4^{(1)} = 0{,}44 - 0{,}21 \cdot 2{,}15 + 0{,}13 \cdot (-0{,}83) - 0{,}27 \cdot 1{,}16 = -0{,}4326.$$

Перевіримо виконання умови закінчення ітераційного процесу:

$$\left|\vec{x}^{(1)} - \vec{x}^{(0)}\right| = \begin{pmatrix} |2{,}9719 - 2{,}15| \\ |-1{,}0775 - (-0{,}83)| \\ |1{,}5093 - 1{,}16| \\ |-0{,}4326 - 0{,}44| \end{pmatrix} = \begin{pmatrix} 0{,}8219 \\ 0{,}2475 \\ 0{,}3493 \\ 0{,}8726 \end{pmatrix}.$$ Умова не виконується, різниця між на-

ближеннями більше ε, тому продовжимо обчислення $\vec{x}^{(2)} = \vec{\beta} + \alpha \cdot \vec{x}^{(1)}$:

$$x_1^{(2)} = 2{,}15 + 0{,}32 \cdot 2{,}9719 - 0{,}05 \cdot (-1{,}0775) + 0{,}11 \cdot 1{,}5093 - 0{,}08 \cdot (-0{,}4326) = 3{,}3555;$$
$$x_2^{(2)} = -0{,}83 + 0{,}11 \cdot 2{,}9719 + 0{,}16 \cdot (-1{,}0775) - 0{,}28 \cdot 1{,}5093 - 0{,}06 \cdot (-0{,}4326) = -1{,}0721$$
$$x_3^{(2)} = 1{,}16 + 0{,}08 \cdot 2{,}9719 - 0{,}15 \cdot (-1{,}0775) + 0{,}12 \cdot (-0{,}4326) = 1{,}5075;$$
$$x_4^{(2)} = 0{,}44 - 0{,}21 \cdot 2{,}9719 + 0{,}13 \cdot (-1{,}0775) - 0{,}27 \cdot 1{,}5093 = -0{,}7317.$$

$\left|\vec{x}^{(2)} - \vec{x}^{(1)}\right| = \begin{pmatrix} |3{,}3555 - 2{,}9719| \\ |-1{,}0721 + 1{,}0775| \\ |1{,}5075 - 1{,}5093| \\ |-0{,}7317 + 0{,}4326| \end{pmatrix} = \begin{pmatrix} 0{,}3836 \\ 0{,}0054 \\ 0{,}0018 \\ 0{,}2991 \end{pmatrix}$. Умова не виконується, тому продовжимо обчислення $\vec{x}^{(3)} = \vec{\beta} + \alpha \cdot \vec{x}^{(2)}$:

$$x_1^{(3)} = 2{,}15 + 0{,}32 \cdot 3{,}3555 - 0{,}05 \cdot (-1{,}0721) + 0{,}11 \cdot 1{,}5075 - 0{,}08 \cdot (-0{,}7317) = 3{,}5017;$$
$$x_2^{(3)} = -0{,}83 + 0{,}11 \cdot 3{,}3555 + 0{,}16 \cdot (-1{,}0721) - 0{,}28 \cdot 1{,}5075 - 0{,}06 \cdot (-0{,}7317) = -1{,}0106$$
$$x_3^{(3)} = 1{,}16 + 0{,}08 \cdot 3{,}3555 - 0{,}15 \cdot (-1{,}0721) + 0{,}12 \cdot (-0{,}7317) = 1{,}5015;$$
$$x_4^{(3)} = 0{,}44 - 0{,}21 \cdot 3{,}3555 + 0{,}13 \cdot (-1{,}0721) - 0{,}27 \cdot 1{,}5075 = -0{,}8111.$$

$\left|\vec{x}^{(3)} - \vec{x}^{(2)}\right| = \begin{pmatrix} |3{,}3517 - 3{,}3555| \\ |-1{,}0106 + 1{,}0721| \\ |1{,}5015 - 1{,}5075| \\ |-0{,}8111 + 0{,}7317| \end{pmatrix} = \begin{pmatrix} 0{,}1462 \\ 0{,}0615 \\ 0{,}0060 \\ 0{,}0794 \end{pmatrix}$. Різниця між наближеннями більше ε, тому продовжимо обчислення $\vec{x}^{(4)} = \vec{\beta} + \alpha \cdot \vec{x}^{(3)}$:

$$x_1^{(4)} = 2{,}15 + 0{,}32 \cdot 3{,}5017 - 0{,}05 \cdot (-1{,}0106) + 0{,}11 \cdot 1{,}5015 - 0{,}08 \cdot (-0{,}8111) = 3{,}5511;$$
$$x_2^{(4)} = -0{,}83 + 0{,}11 \cdot 3{,}5017 + 0{,}16 \cdot (-1{,}0106) - 0{,}28 \cdot 1{,}5015 - 0{,}06 \cdot (-0{,}8111) = -0{,}9783$$
$$x_3^{(4)} = 1{,}16 + 0{,}08 \cdot 3{,}5017 - 0{,}15 \cdot (-1{,}0106) + 0{,}12 \cdot (-0{,}8111) = 1{,}4944;$$
$$x_4^{(4)} = 0{,}44 - 0{,}21 \cdot 3{,}5017 + 0{,}13 \cdot (-1{,}0106) - 0{,}27 \cdot 1{,}5015 = -0{,}8321.$$

$\left|\vec{x}^{(4)} - \vec{x}^{(3)}\right| = \begin{pmatrix} |3{,}5511 - 3{,}5017| \\ |-0{,}9783 - (-1{,}0106)| \\ |1{,}4944 - 1{,}5015| \\ |-0{,}8321 - (-0{,}8111)| \end{pmatrix} = \begin{pmatrix} 0{,}0494 \\ 0{,}0323 \\ 0{,}0071 \\ 0{,}0210 \end{pmatrix}$. Умова закінчення обчислень виконується, різниця між наближеннями менше ε. Останнє наближення приймаємо за остаточний розв'язок:

$$\vec{x} \approx \vec{x}^{(4)} = \begin{pmatrix} 3{,}55 \\ -0{,}98 \\ 1{,}49 \\ -0{,}83 \end{pmatrix}.$$

Відповідь: $x_1 = 3{,}55;\ x_2 = -0{,}98;\ x_3 = 1{,}49;\ x_4 = -0{,}83$, кількість ітерацій 4.

2. ***Методом Зейделя*** розв'язати СЛАР з точністю $\varepsilon = 0{,}05$:

$$\begin{cases} x_1 = 2{,}15 + 0{,}32x_1 - 0{,}05x_2 + 0{,}11x_3 - 0{,}08x_4, \\ x_2 = -0{,}83 + 0{,}11x_1 + 0{,}16x_2 - 0{,}28x_3 - 0{,}06x_4, \\ x_3 = 1{,}16 + 0{,}08x_1 - 0{,}15x_2 + 0{,}12x_4, \\ x_4 = 0{,}44 - 0{,}21x_1 + 0{,}13x_2 - 0{,}27x_3. \end{cases}$$

Оберемо початкове наближення розв'язку: $\vec{x}^{(0)} = \vec{\beta} = \begin{pmatrix} 2{,}15 \\ -0{,}83 \\ 1{,}16 \\ 0{,}44 \end{pmatrix}$. Тобто, $x_1^{(0)} = 2{,}15;\ x_2^{(0)} = -0{,}83;\ x_3^{(0)} = 1{,}16;\ x_4^{(0)} = 0{,}44$. Для знаходження першого наближення невідомої $x_1^{(1)}$ початкове наближення $\vec{x}^{(0)}$ підставимо в праву частину першого рівняння:

$$x_1^{(1)} = 2{,}15 + 0{,}32 \cdot 2{,}15 - 0{,}05 \cdot (-0{,}83) + 0{,}11 \cdot 1{,}16 - 0{,}08 \cdot 0{,}44 = 2{,}9719.$$

Для обчисленні першого наближення невідомої $x_2^{(1)}$ враховуються вже обчислене наближення $x_1^{(1)}$ та наближення $x_2^{(0)}, x_3^{(0)}, x_4^{(0)}$:

$$x_2^{(1)} = -0{,}83 + 0{,}11 \cdot 2{,}9719 + 0{,}16 \cdot (-0{,}83) - 0{,}28 \cdot 1{,}16 - 0{,}06 \cdot 0{,}44 = -0{,}9871.$$

Для обчисленні першого наближення невідомої $x_3^{(1)}$ враховуються вже обчислені наближення $x_1^{(1)}$, $x_2^{(1)}$ та наближення $x_3^{(0)}, x_4^{(0)}$:

$$x_3^{(1)} = 1{,}16 + 0{,}08 \cdot 2{,}9719 - 0{,}15 \cdot (-0{,}9871) + 0{,}12 \cdot 0{,}44 = 1{,}5986.$$

Для обчисленні першого наближення невідомої $x_4^{(1)}$ враховуються вже обчислені наближення $x_1^{(1)}$, $x_2^{(1)}$, $x_3^{(1)}$ та наближення $x_4^{(0)}$:

$$x_4^{(1)} = 0{,}44 - 0{,}21 \cdot 2{,}9719 + 0{,}13 \cdot (-0{,}9871) - 0{,}27 \cdot 1{,}5986 = -0{,}7441.$$

Перевіримо виконання умови закінчення ітераційного процесу:

$$\left|\vec{x}^{(1)} - \vec{x}^{(0)}\right| = \begin{pmatrix} |2{,}9719 - 2{,}15| \\ |-0{,}9871 - (-0{,}83)| \\ |1{,}5986 - 1{,}16| \\ |-0{,}7441 - 0{,}44| \end{pmatrix} = \begin{pmatrix} 0{,}8219 \\ 0{,}1571 \\ 0{,}4386 \\ 0{,}1840 \end{pmatrix}.$$ Умова не виконується, різниця між наближеннями більше ε, тому продовжимо обчислення.

$$x_1^{(2)} = 2{,}15 + 0{,}32 \cdot 2{,}9719 - 0{,}05 \cdot (-0{,}9871) + 0{,}11 \cdot 1{,}5986 - 0{,}08 \cdot (-0{,}7441) = 3{,}3857;$$
$$x_2^{(2)} = -0{,}83 + 0{,}11 \cdot 3{,}3857 + 0{,}16 \cdot (-0{,}9871) - 0{,}28 \cdot 1{,}5986 - 0{,}06 \cdot (-0{,}7441) = -1{,}0185;$$
$$x_3^{(2)} = 1{,}16 + 0{,}08 \cdot 3{,}3857 - 0{,}15 \cdot (-1{,}0185) + 0{,}12 \cdot (-0{,}7441) = 1{,}4943;$$
$$x_4^{(2)} = 0{,}44 - 0{,}21 \cdot 3{,}3857 + 0{,}13 \cdot (-1{,}0185) - 0{,}27 \cdot 1{,}4943 = -0{,}8069.$$

Перевіримо виконання умови закінчення ітераційного процесу:

$$\left|\vec{x}^{(2)} - \vec{x}^{(1)}\right| = \begin{pmatrix} |3{,}3857 - 2{,}9719| \\ |-1{,}0185 - (-0{,}9871)| \\ |1{,}4943 - 1{,}5986| \\ |-0{,}8069 - (-0{,}7441)| \end{pmatrix} = \begin{pmatrix} 0{,}4138 \\ 0{,}0314 \\ 0{,}1043 \\ 0{,}0628 \end{pmatrix}.$$ Умова не виконується, різниця між наближеннями більше ε, тому продовжимо обчислення.

$$x_1^{(3)} = 2{,}15 + 0{,}32 \cdot 3{,}3857 - 0{,}05 \cdot (-1{,}0185) + 0{,}11 \cdot 1{,}4943 - 0{,}08 \cdot (-0{,}8069) = 3{,}5133;$$
$$x_2^{(3)} = -0{,}83 + 0{,}11 \cdot 3{,}5133 + 0{,}16 \cdot (-1{,}0185) - 0{,}28 \cdot 1{,}4943 - 0{,}06 \cdot (-0{,}8069) = -0{,}9765;$$
$$x_3^{(3)} = 1{,}16 + 0{,}08 \cdot 3{,}5133 - 0{,}15 \cdot (-0{,}9765) + 0{,}12 \cdot (-0{,}8069) = 1{,}4907;$$
$$x_4^{(3)} = 0{,}44 - 0{,}21 \cdot 3{,}5133 + 0{,}13 \cdot (-0{,}9765) - 0{,}27 \cdot 1{,}4907 = -0{,}8272.$$

Перевіримо виконання умови закінчення ітераційного процесу:

$$\left|\vec{x}^{(3)}-\vec{x}^{(2)}\right|=\begin{pmatrix}|3.5133-3,3857|\\|-0,9765-(-1,0185)|\\|1,4907-1,4943|\\|-0,8272-(-0,8069)|\end{pmatrix}=\begin{pmatrix}0,1256\\0,0420\\0,0036\\0,0204\end{pmatrix}.$$ Умова не виконується, різниця між наближеннями більше ε, тому продовжимо обчислення.

$$x_1^{(4)}=2,15+0,32\cdot 3,5133-0,05\cdot(-0,9765)+0,11\cdot 1,4904-0,08\cdot(-0,8272)=3,5532;$$
$$x_2^{(4)}=-0,83+0,11\cdot 3,5532+0,16\cdot(-0,9765)-0,28\cdot 1,4904-0,06\cdot(-0,8272)=-0,9632;$$
$$x_3^{(4)}=1,16+0,08\cdot 3,5532-0,15\cdot(-0,9632)+0,12\cdot(-0,8272)=1,4895;$$
$$x_4^{(4)}=0,44-0,21\cdot 3,5532+0,13\cdot(-0,9632)-0,27\cdot 1,4895=-0,8335.$$

Перевіримо виконання умови закінчення ітераційного процесу:

$$\left|\vec{x}^{(4)}-\vec{x}^{(3)}\right|=\begin{pmatrix}|3,5532-3.5133|\\|-0,9632-(-0,9765)|\\|1,4895-1,4907|\\|-0,8335-(-0,8272)|\end{pmatrix}=\begin{pmatrix}0,0340\\0,0134\\0,0013\\0,0063\end{pmatrix}.$$ Умова закінчення обчислень виконується, різниця між наближеннями менше ε. Останнє наближення приймаємо за остаточний розв'язок:

$$\vec{x}\approx\vec{x}^{(4)}=\begin{pmatrix}3,55\\-0,96\\1,49\\-0,83\end{pmatrix}.$$

Відповідь: $x_1=3,55;\ x_2=-0,96;\ x_3=1,49;\ x_4=-0,83$, кількість ітерацій 4.

Методи показали однакову швидкість збіжності – чотири ітерації, але бачимо, що отримані розв'язки різняться за значеннями (зокрема для x_2). Це можна пояснити низькою заданою точністю ε.

Наведемо фрагменти програм, що реалізують розглянуті алгоритми. Вхідними даними будуть вимірність системи n (n=4), матриця коефіцієнтів зведеної системи α та вектор вільних членів $\vec{\beta}$.

Досліджуємо збіжність ітераційного процесу:

```
N1:=0; N2:=0; N3:=0;
for i:=1 to n do
                begin
                        S1:=0;S2:=0;
                         for j:=1 to n do
                                        begin
                                                S1:=S1+abs(A[i,j]);
                                                S2:=S2+abs(A[j,i]);
                                                N3:=N3+sqr(A[i,j]);
                                        end;
                        if S1>N1 then N1:=S1;
                        if S2>N2 then N2:=S2;
                end;
N3:=sqrt(N3);
writeln('Норми матриці A: N1=',N1:4:2,' N2=',N2:4:2,' N3=',N3:4:2);
if ((N1<1) or (N2<1) or (N3<1)) then writeln('Ітераційний процес збіжний! ',
```

```
                    'Знайдемо розв''язок з точністю e=',e:8:6)
                else
                    begin
                        writeln('Ітераційний процес ',
                        'розбіжний! Обчислення припиняємо!');
                        Halt;
                    end;
```

Метод простої ітерації.

```
eps:=1.0; k:=0; y:=b;
while eps>=e do
            begin
                eps:=0; k:=k+1; write('крок ',k);
                for i:=1 to n do
                        begin
                            x[i]:=b[i]; S1:=0;
                            for j:=1 to n do x[i]:=x[i]+A[i,j]*y[j];
                            write('   x[',i,']=',x[i]:8:4);
                            S1:=abs(x[i]-y[i]);
                            if S1>eps then eps:=S1;
                        end;
                y:=x;
                writeln('   eps=', eps:8:4);
            end;
writeln('Розв''язок системи:');
for i:=1 to n do writeln('   x[',i,']=',x[i]:8:4);
```

Результати роботи програми приведені на рис. 4.3.

Метод Зейделя.

```
eps:=1.0; k:=0; y:=b;
while eps>=e do
            begin
                eps:=0;  k:=k+1; write('крок ',k);
                for i:=1 to n do
                        begin
                            x[i]:=b[i]; S1:=0;
                            for j:=1 to i-1 do x[i]:=x[i]+A[i,j]*x[j];
                            for j:=i  to n do x[i]:=x[i]+A[i,j]*y[j]
                            write('   x[',i,']=',x[i]:8:4);
                            S1:=abs(x[i]-y[i]);
                            if S1>eps then eps:=S1;
                        end;
                y:=x;
                writeln('   eps=', eps:8:4);
            end;
writeln('Розв''язок системи:');
for i:=1 to n do writeln('   x[',i,']=',x[i]:8:4);
```

Результати роботи програми приведені на рис. 4.4.

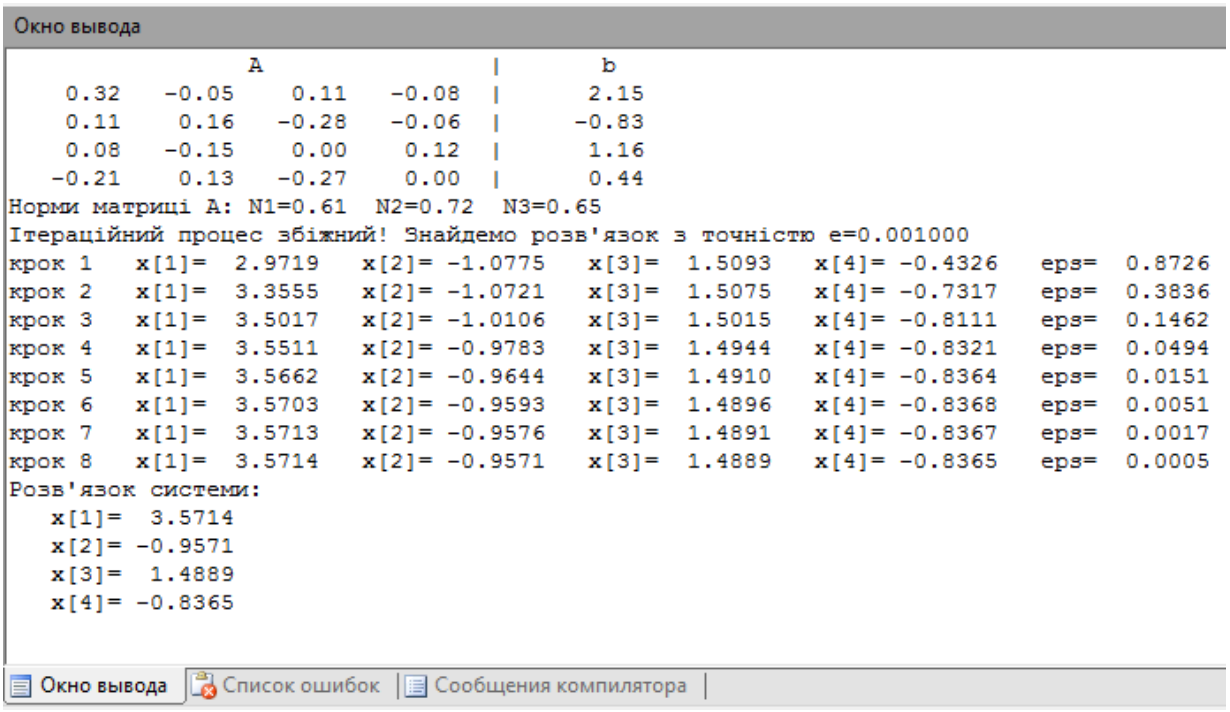


Рисунок 4.3 – Розв’язок СЛАР методом простої ітерації

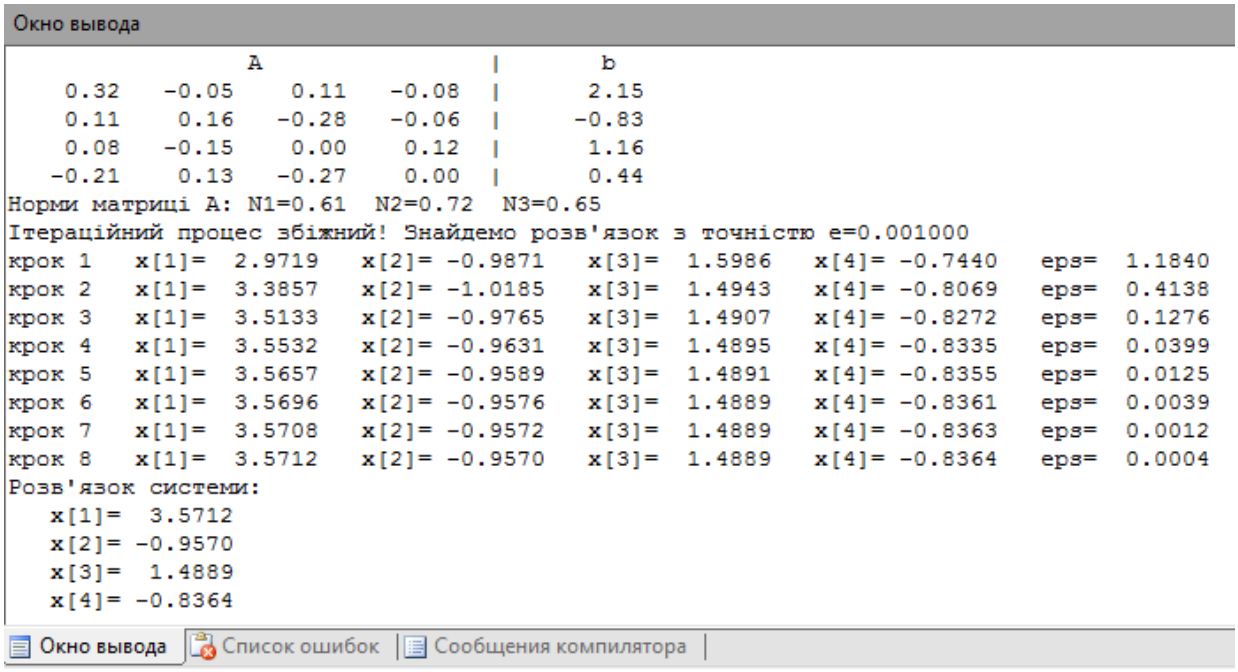


Рисунок 4.4 – Розв’язок СЛАР методом Зейделя

В програмній реалізації точність збільшили до ε=0,001. Відповідно збільшилася кількість ітерацій, а значення розв’язків співпадають.

Контрольні запитання

1. Охарактеризуйте ітераційні методи розв’язання СЛАР.
2. Наведіть формулу ітерацій в методі простої ітерації.
3. В чому міститься модифікація методу простої ітерації в методі Зейделя?
4. Яка умова закінчення ітераційних процесів?
5. Як перевірити збіжність процесу ітерації?

Перелік рекомендованої літератури

1. Фельдман Л.П. Чисельні методи в інформатиці / Л.П. Фельдман, А.І. Петренко, О.А. Дмитрієва. –К.: BHV, 2006. – 480 с.
2. Пирумов У. Г. Численные методы: учеб. пособие для студ. втузов / У.Г. Пирумов. – 4-е изд., стереотип. – М.: Дрофа, 2007. – 221, [3] с.: ил.
3. Численные методы. Сборник задач: учеб. пособие для вузов / В.Ю. Гидаспов, И.Э. Иванов, Д.Л. Ревизников и др.; под ред. У.Г. Пирумова. – М.: Дрофа, 2007. – 144 с.: ил.

ЛАБОРАТОРНА РОБОТА № 5
ВИЗНАЧЕННЯ КОРЕНІВ НЕЛІНІЙНИХ РІВНЯНЬ

Мета: опанувати чисельними методами визначення коренів нелінійних рівнянь – методом послідовного перебору для відокремлення коренів, методами половинного ділення, Ньютона та ітерацій для уточнення коренів.

Теоретичні відомості

Рівняння виду $f(x)=0,$ (5.1)

де $f(x)$ - нелінійна функція визначена на деякій числовій множині $x°\epsilon°X$, називається *нелінійним,* а множина X – *областю допустимих значень* рівняння. За виглядом функції $f(x)$ нелінійні рівняння поділяють на алгебраїчні та трансцендентні. Функція називається *алгебраїчною*, якщо вона містить арифметичні операції та піднесення в степінь з раціональним або ірраціональним показником. До *трансцендентних* функцій відносять всі неалгебраїчні функції: показникові – a^x, логарифмічні – $\log_a x$, $\ln x$, тригонометричні – $\sin x$, $\cos x$, $\operatorname{tg} x$, $\operatorname{ctg} x$, обернені тригонометричні $\arcsin x$, $\arccos x$, $\operatorname{arctg} x$, $\operatorname{arcctg} x$ та інші.

Будь яке значення ξ, яке при підстановці в початкове рівняння (5.1) перетворює його в тотожність $f(\xi)\equiv 0$, будемо називати *коренем рівняння* або *нулем.*

Процес розв'язання нелінійних рівнянь в методах обчислювальної математики поділяється на два етапи:

- відокремлення коренів;
- уточнення коренів.

Методи відокремлення коренів. Відокремити корені рівняння означає визначити на числовій осі Ox проміжки, на кожному з яких знаходиться єдиний (відокремлений) корінь рівняння. Будемо вважати, що рівняння (5.1) має лише ізольовані корені, тобто для кожного кореня існує окіл, в якому інших коренів немає.

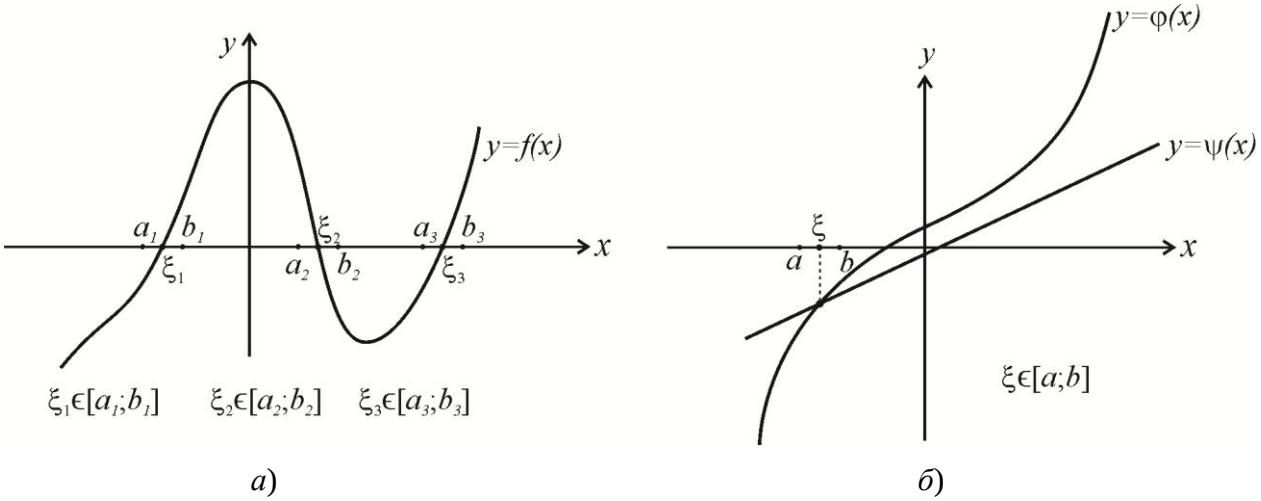
Відокремлення коренів можна здійснити трьома способами – графічним, аналітичним та методом послідовного перебору.

Графічний метод. Будують графік функції $y=f(x)$ для рівняння виду $f(x)=0$ (рис. 5.1, *а*) або представляють рівняння у вигляді $\varphi(x)=g(x)$ та будують графіки функцій $y=\varphi(x)$ та $y=\psi(x)$ (рис. 5.1, *б*). Значення дійсних коренів рівняння є абсцисами точок перетину графіка функції $y=f(x)$ з віссю Ox або абсцисами точок перетину графіків функцій $y=\varphi(x)$ та $y=\psi(x)$. Відрізки, в яких знаходиться тільки по одному кореню, легко знаходяться наближено.

Аналітичний метод. Аналітично корні рівняння $f(x)=0$ можна відокремити, використовуючи деякі властивості функцій та однією з розглянутих нижче теорем.

Теорема 1. Якщо функція $f(x)$ неперервна на відрізку $[a;b]$ і приймає на кінцях цього відрізку значення різних знаків, то всередині відрізка $[a;b]$ існує хоча б один корінь рівняння $f(x)=0$ (рис.5.2, *а*).

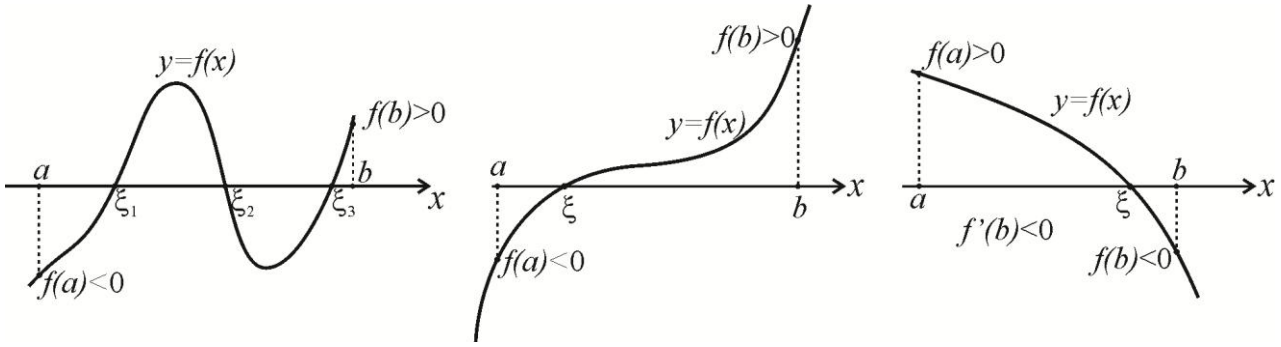
Теорема 2. Якщо функція $f(x)$ неперервна та монотонна на відрізку $[a;b]$ і приймає на кінцях відрізка значення різних знаків, то всередині відрізка $[a;b]$ існує корінь рівняння , і цей корінь єдиний (рис.5.2, *б*).



а) *б*)

Рисунок 5.1 – Графічний метод відокремлення коренів

Теорема 3. Якщо функція $f(x)$ неперервна на відрізку $[a;b]$ і приймає на кінцях цього відрізку значення різних знаків, а похідна $f'(x)$ всередині відрізка $[a;b]$ існує і зберігає постійний знак, то всередині відрізка $[a;b]$ існує єдиний корінь рівняння (рис.5.2, *в*).



а) *б*) *в*)

Рисунок 5.2 – Графічна інтерпретація теорем

Алгоритм відокремлення коренів аналітичним методом:

1. Дослідити дане рівняння на монотонність і неперервність, визначити область допустимих та граничних значень.
2. Знайти $f'(x)$ – першу похідну, прирівняти її до нуля та визначити критичні точки.
3. Скласти таблицю знаків функції $f(x)$, використовуючи для x значення критичних точок, граничних значень з ОДЗ і точок, отриманих на першому кроці при аналізі даного рівняння.
4. Визначити інтервали, на кінцях яких функція приймає значення протилежних знаків. Всередині цих інтервалів існує по одному і тільки одному кореню.

Метод послідовного перебору. Виходячи з приблизного графіка функції $f(x)$ або з граничних значень ОДЗ, визначають інтервал $[a;b]$, на якому будуть шукатися корені рівняння. Далі табулюють функцію $f(x)$, починаючи з точки $x=a$, рухаючись управо з деяким кроком *h* до точки $x=b$ (рис.5.3).

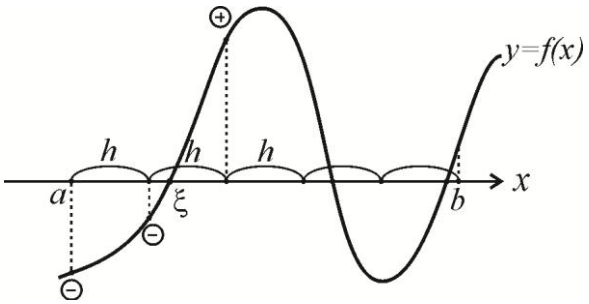


Рисунок 5.3 – Графічна інтерпретація методу послідовного перебору

Як тільки виявиться пара сусідніх значень $f(x)$, яка має різні знаки, і функція $f(x)$ монотонна на цьому відрізку, так відповідні значення аргументу *x* (попереднє й наступне) можна вважати кінцями відрізку, що містить корінь. Метод добре алгоритмізується. Блок-схема алгоритму метода послідовного перебору приведена на рис. 5.4.

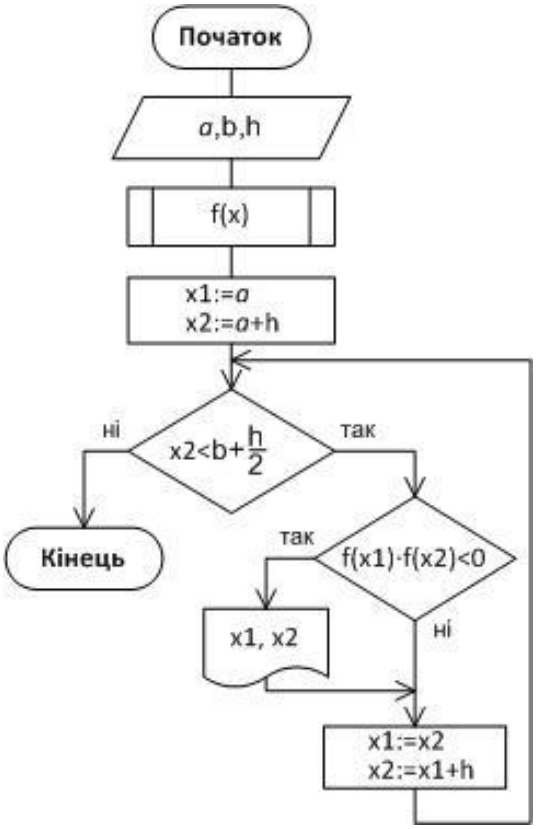


Рисунок 5.4 – Блок-схема методу послідовного перебору

Методи уточнення коренів. Розглянемо другий етап наближеного розв'язання нелінійних рівнянь – уточнення коренів, тобто доведення їх до заданої точності. Для уточнення коренів нелінійного рівняння із заданою похибкою ε на деякому відрізку $[a;b]$ в інженерній практиці найбільш широко використовують: метод половинного ділення, метод пропорційних частин, метод Ньютона та метод ітерацій.

Всі ці методи являються ітераційними, тобто побудовані на алгоритмах, в яких одна з їх частин повторюється багаторазово, при чому кількість повторень залежить від початкових даних (від заданої користувачем похибки, від відрізка дослідження, тощо). Розглянемо особливості цих методів та алгоритмів, на яких вони базуються.

Метод половинного ділення. Нехай маємо рівняння $f(x)=0$, де $f(x)$ – неперервна, монотонна нелінійна функція, яка має на відрізку $[a;b]$ єдиний корінь ξ, тобто добуток $f(a)\cdot f(b)<0$, причому $b-a>\varepsilon$, де ε – задана похибка обчислень. Потрібно знайти значення кореня ξ з заданою похибкою ε.

Алгоритм методу оснований на багатократному ділені навпіл (рис. 5.5) і звужуванні досліджуваного відрізка$[a;b]$, який отримали в результаті попереднього дослідження функції $f(x)$ (відокремлення коренів):

1. На відрізку $[a;b]$ вибираємо точку $c=\frac{a+b}{2}$, яка розділяє його на два рівних відрізки $[a;c]$ і $[c;b]$.
2. Перевіряємо чи $f(c)=0$, якщо так, то c – точний корінь початкового рівняння.
3. У випадку, коли $f(c)\neq 0$, то з двох отриманих відрізків $[a;c]$ і $[c;b]$ вибираємо той, на кінцях якого функція $f(x)$ приймає значення протилежних знаків, тобто, якщо $f(a)\cdot f(c)<0$, тоді залишаємо відрізок $[a;c]$ і точку b переносимо в точку c ($b=c$); якщо $f(a)\cdot f(c)>0$, то залишаємо відрізок $[c;b]$ і переносимо точку a в точку c ($a=c$) і переходимо до пункту 1.
4. Процес ділення відрізка навпіл виконується доти, поки на якомусь кроці, або середина відрізка буде коренем, або буде виконана умова закінчення ітераційного процесу: $|b-a|<\varepsilon$. У цьому випадку за наближене значення кореня вибирають $\xi\approx\frac{a+b}{2}$.

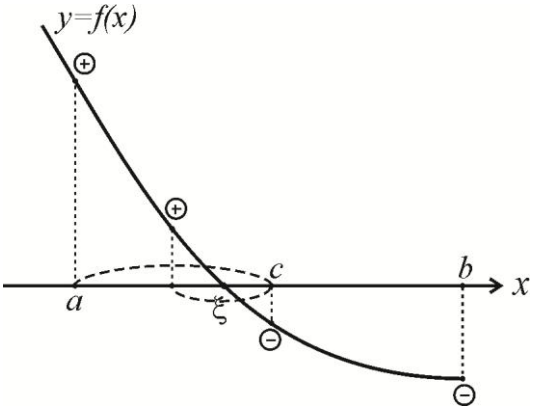


Рисунок 5.5 – Графічна інтерпретація методу половинного ділення

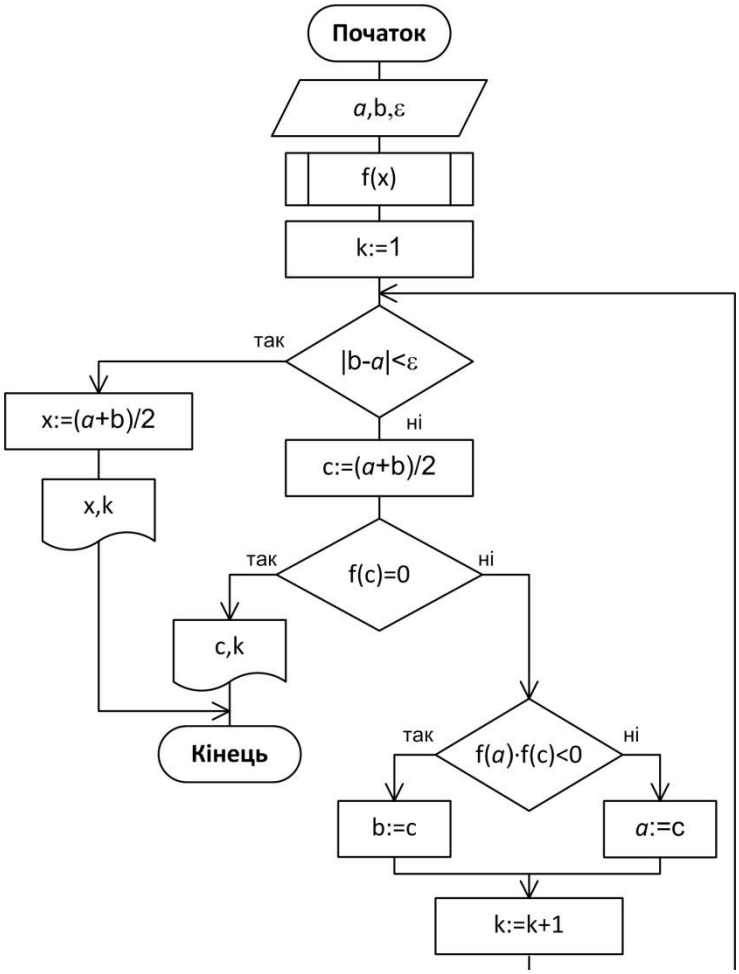
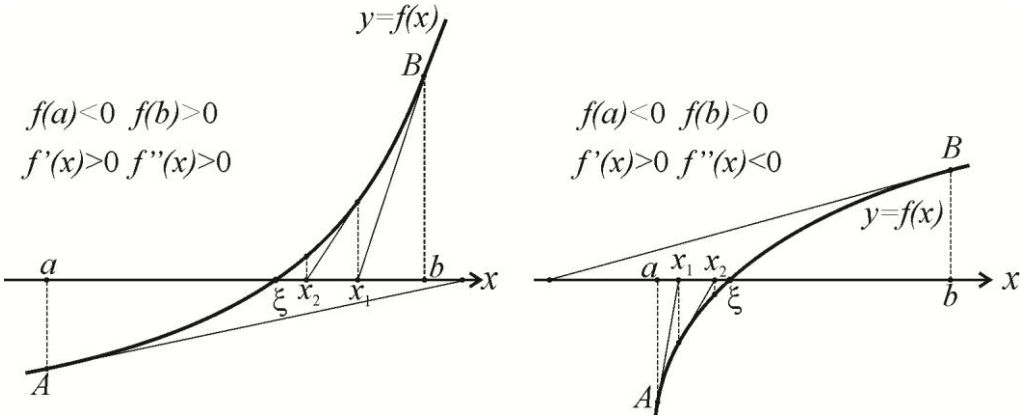


Рисунок 5.6 – Блок-схема методу половинного ділення

Метод половинного ділення – це найпростіший метод уточнення кореня рівняння. Він сходиться для будь-яких неперервних функцій $f(x)$, в тому числі недиференційованих. Блок-схема методу приведена на рис. 5.6.

Метод Ньютона дуже широко використовується при побудові ітераційних алгоритмів. Його популярність обумовлена тим, що на відміну від попереднього методу замість інтерполяції по двом значенням функції, в методі Ньютона здійснюється екстраполяція за допомогою дотичної до кривої в одній точці.

Нехай корінь рівняння *f(x)*=0 відокремлений на відрізку $[a;b]$, на якому нелінійна функція *f(x)* монотонна і має різні знаки на кінцях відрізку, причому похідні $f'(x)$ та $f''(x)$ неперервні та зберігають постійні знаки на всьому відрізку $[a;b]$. Потрібно знайти наближене значення кореня ξ з заданою похибкою ε.



а) *б*)

Рисунок 5.7 – Графічна інтерпретація методу Ньютона

Геометричний зміст метода Ньютона полягає в тому, що дуга кривої $y = f(x)$ на відрізку $[a;b]$ замінюється дотичною до цієї кривої, а наближене значення кореня визначається як точка перетину дотичної з віссю *Ox,* проведеної з одного з кінців досліджуваного відрізку (рис. 5.7). В результаті отримана послідовність наближених значень x_1, x_2, ..., x_k, ..., кожний наступний член якої ближчій до кореня ξ, ніж попередній. Однак всі x_k залишаються більше (рис.°5.7, *а*) або менше (рис.°5.7, *б*) істинного кореня ξ, тобто x_k - наближене значення кореня ξ з надлишком (з недостачею). Процес визначення кореня продовжується багаторазово доти, поки не одержано наближений корінь із заданим степенем точності. Рівняння дотичної має вигляд:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \qquad (5.2)$$

Порівнюючи рис. 5.7, *а* і *б,* помічаємо, що вони відрізняються один від одного тільки вибором початкового наближення (точкою побудови першої дотичної): в першому випадку за x_0 приймався кінець b відрізка $[a;b]$, в другому – кінець a. В протилежному випадку, значення першого наближення кореня x_1 знаходилося б за межами відрізку $[a;b]$. Точка, в якій будується дотична, обирається з умови співпадання знаків функції та її другої похідної $f(x)\cdot f''(x) > 0$.

При виборі початкового наближення кореня необхідно використовувати наступне правило: *за початкову точку слід вибирати той кінець відрізка* $[a;b]$, *в якому знак функції співпадає зі знаком другої похідної.* В першому випадку $f(b)\cdot f''(b)>0$ і початкова точка $x_0 = b$, в другому $f(a)\cdot f''(a)>0$ і в якості початкового наближення беремо $x_0 = a$.

Розрахунки продовжують до одночасного виконання умов $|f(x_{k+1})|<\varepsilon$ та $|x_{k+1}-x_k|<\varepsilon$. Блок-схема методу приведена на рис. 5.8.

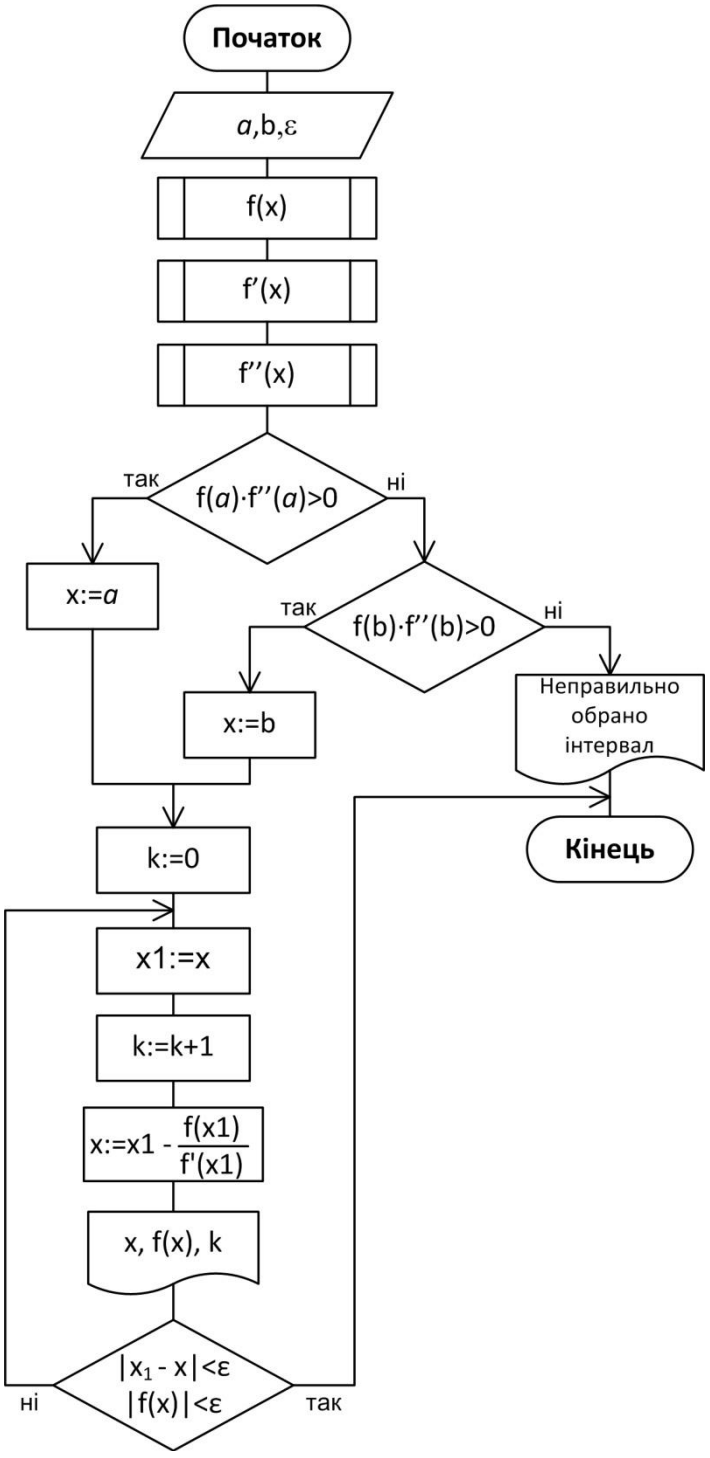


Рисунок 5.8 – Блок-схема методу Ньютона

Метод ітерацій. Суть методу полягає у заміні початкового рівняння (5.1) еквівалентним йому рівнянням

$$x = \varphi(x). \qquad (5.3)$$

Нехай задано рівняння $f(x)=0$, де $f(x)$ – неперервна нелінійна функція. Потрібно визначити корінь ξ цього рівняння, який знаходиться на відрізку $[a;b]$ з заданою похибкою ε. Виберемо довільним способом $x_0 \in [a;b]$ і підставимо його в праву частину рівняння (5.3); тоді отримаємо $x_1 = \varphi(x_0)$. Потім це значення x_1 підставимо знову в праву частину рівняння (5.3) і отримаємо $x_2 = \varphi(x_1)$ (рис.°5.9, *а, б*). Повторюючи цей процес, отримаємо послідовність чисел $x_{k+1} = \varphi(x_k)$. При цьому можливі два випадки:

- послідовність $x_1, x_2, ..., x_k, ...$, збігається, тобто має границю, і тоді ця границя буде коренем рівняння (5.1).
- послідовність $x_1, x_2, ..., x_k, ...$, розбігається, тобто не має границі.

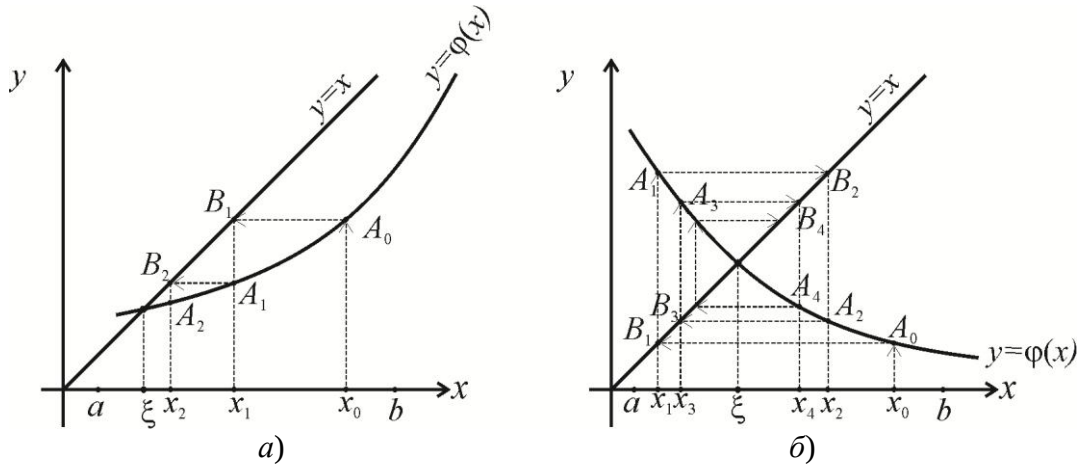


Рисунок 5.9 – Графічна інтерпретація методу ітерацій

Приведемо без доказу теорему, яка виражає умову, при якій ітераційний процес розв'язку нелінійного рівняння методом ітерацій на ЕОМ збігається.

Теорема | Нехай на відрізку $[a;b]$ знаходиться єдиний корінь рівняння $x = \varphi(x)$ та у всіх точках цього відрізку похідна $\varphi'(x)$ задовольняє нерівності $|\varphi'(x)| \le q < 1$. Якщо при цьому виконується і умова $a \le \varphi(x) \le b$, то ітераційний процес збігається, а за нульове наближення x_0 можна взяти число з відрізку $[a;b]$.

При використанні методу простих ітерацій основною проблемою є вибір функції $\varphi(x)$ в рівнянні $x = \varphi(x)$, яка б задовольняла умові збіжності. Чим менше значення $|\varphi'(x)|$, тим швидша збіжність. Як правило, обирають $\varphi(x) = x - \dfrac{f(x)}{q}$, де $q \approx \max\limits_{[a;b]} |f'(x)|$. Знак q має збігатися зі знаком $f'(x)$ на $[a;b]$.

Блок-схема алгоритму приведена на рис. 5.10.

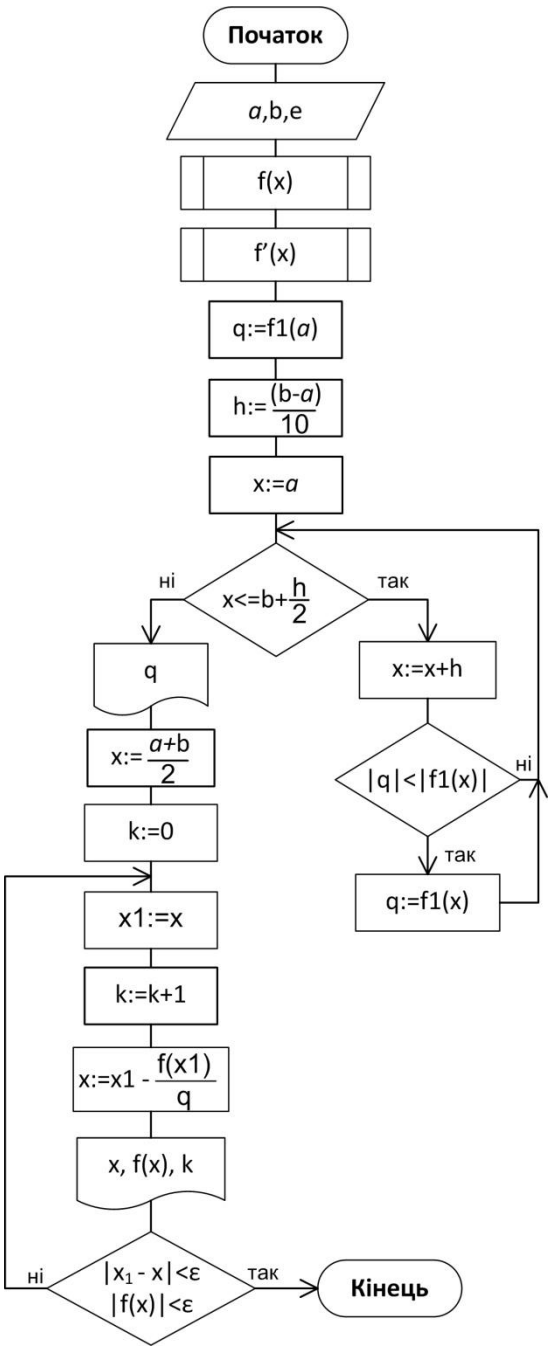


Рисунок 5.10 – Блок-схема методу ітерацій

Умова закінчення ітераційного процесу аналогічна умові для методу Ньютона: $\left|f(x_{k+1})\right| < \varepsilon$ та $\left|x_{k+1} - x_k\right| < \varepsilon$.

Завдання

Відокремити корені нелінійного рівняння методом послідовного перебору. Уточнити відокремлені корені з точністю $\varepsilon = 10^{-3}$ методами половинного ділення, Ньютона, ітерацій.

1. $2\ln(x) - \frac{1}{x} = 0$
2. $2\ln(x) - \frac{x}{2} + 1 = 0$
3. $3x^3 + 8x^2 + 4x - 8 = 0$
4. $2 - xe^x = 0$

5. $2^x - 2x^2 - 1 = 0, \quad x > 0$
6. $e^x - 2^x - 1 = 0$
7. $x^2 - \cos(x) = 0, \quad x < 0$
8. $tg(x) - x = 0, \quad x > 0$
9. $5x - 8\ln(x) - 8 = 0$
10. $\sqrt{x+1} - \frac{1}{x} = 0$
11. $x + \ln(x) - 0{,}5 = 0$
12. $x\ln(x) - 100 = 0$
13. $2 - x - \ln(x) = 0$
14. $xtg(x) - 1{,}28 = 0, \quad x > 0$
15. $x + e^x + e^{-3x} - 4 = 0, \quad x > 0$
16. $\ln(x) - tg(x) = 0, x < \frac{\pi}{2}$
17. $6x - 5\sin(x) - 6 = 0, \quad x < 0$
18. $x^4 + x^3 + 2x - 5 = 0$
19. $x^5 - 2x + 4 = 0$
20. $3x - 4\ln(x) - 5 = 0$

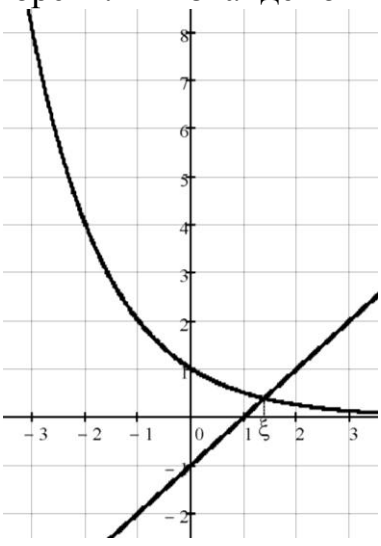
Результати обчислень подати у вигляді таблиці:

Метод	Корінь ξ	Значення $f(\xi)$	Кількість ітерацій *k*

Послідовність виконання лабораторної роботи

Знайти розв'язок нелінійного рівняння $x - 0{,}5^x - 1 = 0$ з точністю $\varepsilon = 10^{-3}$. Рівняння має вигляд $f(x)=0$: тоді $f(x) = x - 0{,}5^x - 1$.

Відокремимо корені рівняння спочатку ***графічним методом***. Функція *f(x)* складається з різниці двох елементарних функцій $x-1$ і $0{,}5^x$, тобто $f(x) = (x-1) - (0{,}5^x) = g(x) - h(x) = 0$. Побудуємо їх окремо. Абсциса точки перетину графіків двох функції буде наближеним значенням кореня. Ми знайдемо інтервал ізоляції кореня.



$g(x) = x - 1$ – лінійна функція, для побудови її графіка достатньо двох точок.

x	-1	2
g(*x*)	-2	1

$h(x) = 0{,}5^x$ – показникова функція, її графік не буде лінійним, отримаємо її значення в декількох точках.

x	-3	-2	-1	0	1	2	3
h(*x*)	8	4	2	1	0,5	0,25	0,125

Бачимо, що рівняння має один корінь в інтервалі [1;°2]. Зменшимо інтервал ***методом табулювання***. Протабулюємо функцію на інтервалі [1; 2] з кроком 0,1 та визначимо знаки функції в отриманих точках.

x	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	2,0
sign *f*(*x*)	–	–	–	–	+	+	+	+	+	+	+

Функція має різні знаки на кінцях інтервалу [1,3;°1,4]. Тому корінь рівняння буде знаходитися в середині цього інтервалу.

Уточнимо значення кореня до заданої точності ε=0,001.

Метод половинного ділення. Шукаємо середину відрізку $[a;b]$: $c=\frac{a+b}{2}=\frac{1{,}3+1{,}4}{2}=1{,}35$. Отримали два підінтервали: [1,3;°1,35] та [1,35;°1,4]. Обчислюємо значення функції в точці c: $f(c)$=-0,0423. Для подальших розрахунків обираємо той інтервал, на кінцях якого функція має різни знаки, тобто $[c;b]$. Складемо таблицю обчислень. Контролюємо точність.

k	a	c	b	f(a)	f(c)	f(b)	|b-a|
1	1,3	1,35	1,4	-0,1061	-0,0423	0,0211	0,1
2	1,35	1,375	1,4	-0,0423	-0,0106	0,0211	0,05
3	1,375	1,3875	1,4	-0,0106	0,0053	0,0211	0,025
4	1,375	1,3813	1,3875	-0,0106	-0,0026	0,0053	0,0125
5	1,3813	1,3844	1,3875	-0,0026	0,0014	0,0053	0,0062
6	1,3813	1,3828	1,3844	-0,0026	-0,00066	0,0014	0,0031
7	1,3828	1,3836	1,3844	-0,00066	0,00033	0,0014	0,0015
8	1,3828	**1,3832**	1,3836	-0,00066	-0,00016	0,00033	**0,0008**

Задану точність досягнуто: $|b_8-a_8|=|1{,}3836-1{,}3828|=0{,}0008<\varepsilon$.

Відповідь: $\xi\approx1{,}383$; $f(\xi)=-1{,}6\cdot10^{-4}$; кроків k=8.

Метод Ньютона. Розрахункова формула методу:

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)},\ \text{де } k=0,1,2,\dots .$$

Оберемо початкове наближення x_0 за умови:

$x_0=a$, якщо $f(a)\cdot f''(a)>0$; або $x_0=b$, якщо $f(b)\cdot f''(b)>0$.

Для цього знайдемо першу та другу похідні функції $f(x)$:

$f(x)=x-0{,}5^x-1;\quad f'(x)=1-0{,}5^x\cdot\ln0{,}5;\quad f''(x)=-0{,}5^x\cdot(\ln0{,}5)^2$.

Знайдемо значення другої похідної на кінцях інтервалу ізоляції кореня.

$f''(1{,}3)=-0{,}5^{1{,}3}\cdot(\ln0{,}5)^2=-0{,}1951;\qquad f(1{,}3)=1{,}3-0{,}5^{1{,}3}-1=-0{,}1061.$

$f''(1{,}4)=-0{,}5^{1{,}4}\cdot(\ln0{,}5)^2=-0{,}1821;\qquad f(1{,}4)=1{,}4-0{,}5^{1{,}4}-1=0{,}0211.$

Знаки функції та другої похідної співпадають в кінці a=1,3. Тому $x_0=a=1{,}3$. Проведемо обчислення:

$$x_1=x_0-\frac{f(x_0)}{f'(x_0)}=1{,}3-\frac{1{,}3-0{,}5^{1{,}3}-1}{1-0{,}5^{1{,}3}\cdot\ln0{,}5}=1{,}3828;\quad f(1{,}3828)=-0{,}00067;$$

$$|x_1-x_0|=|1{,}3828-1{,}3|=0{,}0828>\varepsilon$$

$$x_2=x_1-\frac{f(x_1)}{f'(x_1)}=1{,}3828-\frac{1{,}3828-0{,}5^{1{,}3828}-1}{1-0{,}5^{1{,}3828}\cdot\ln0{,}5}=1{,}3833;\quad f(1{,}3833)=-0{,}00004;$$

$$|x_2-x_1|=|1{,}3833-1{,}3828|=0{,}0005<\varepsilon$$

Задану точність досягнуто за двома умовами.

Відповідь: $\xi\approx1{,}383$; $f(\xi)=-4\cdot10^{-5}$; кроків k=2.

Метод ітерацій. Розрахункова формула методу:

$$x_{k+1}=x_k-\frac{f(x_k)}{q},\ \text{де } k=0,1,2,\dots;\ |q|=\max_{[a;b]}|f'(x)|.$$

Знак q має збігатися зі знаком $f'(x)$ на $[a;b]$. Дослідимо першу похідну на інтервалі [1,3; 1,4].

$f'(1{,}3) = 1 - 0{,}5^{1{,}3} \cdot \ln 0{,}5 = 1{,}2815$; $f'(1{,}4) = 1 - 0{,}5^{1{,}4} \cdot \ln 0{,}5 = 1{,}2627$. Похідна змінюється незначно, її значення додатні. Тому $q = 2$ (заокруглимо значення у бік більшого цілого). Тоді $x_{k+1} = x_k - \dfrac{f(x_k)}{q}$. За x_0 приймемо середину інтервалу $[a;b]$:

$$x_0 = \frac{a+b}{2} = \frac{1{,}3+1{,}4}{2} = 1{,}35.$$

$$x_1 = x_0 - \frac{f(x_0)}{2} = 1{,}35 - \frac{1{,}35 - 0{,}5^{1{,}35} - 1}{2} = 1{,}3711; \quad f(1{,}3711) = -0{,}01543$$

$$x_2 = x_1 - \frac{f(x_1)}{2} = 1{,}3711 - \frac{1{,}3711 - 0{,}5^{1{,}3711} - 1}{2} = 1{,}3789; \quad f(1{,}3789) = -0{,}00566$$

$$x_3 = x_2 - \frac{f(x_2)}{2} = 1{,}3789 - \frac{1{,}3789 - 0{,}5^{1{,}3789} - 1}{2} = 1{,}3817; \quad f(1{,}3817) = -0{,}00208$$

$$x_4 = x_3 - \frac{f(x_3)}{2} = 1{,}3817 - \frac{1{,}3817 - 0{,}5^{1{,}3817} - 1}{2} = 1{,}3827; \quad f(1{,}3827) = -0{,}00076$$

$$x_5 = x_4 - \frac{f(x_4)}{2} = 1{,}3827 - \frac{1{,}3827 - 0{,}5^{1{,}3827} - 1}{2} = 1{,}3831; \quad f(1{,}3831) = -0{,}00028$$

Задану точність досягнуто за двома умовами:

$|x_5 - x_4| = |1{,}3831 - 1{,}3827| = 0{,}0004 < \varepsilon$, $|f(x_5)| = |-0{,}00028| = 0{,}00028 < \varepsilon$.

Відповідь: $\xi \approx 1{,}383$; $f(\xi) = 2{,}8 \cdot 10^{-4}$; кроків k=5.

Рівняння	Метод	Корінь ξ	Значення $f(\xi)$	Кількість ітерацій k
$x - 0{,}5^x - 1 = 0$	половинного ділення	1,383	$-1{,}6 \cdot 10^{-4}$	7
	метод Ньютона	1,383	$-2{,}5 \cdot 10^{-8}$	2
	метод ітерацій	1,383	$2{,}8 \cdot 10^{-4}$	5

Найбільшу швидкість збіжності показав метод Ньютона.

Наведемо фрагменти програм, що реалізують розглянуті алгоритми.

Метод послідовного перебору. Вхідними даними будуть границі інтервалу $[a,b]$, в якому шукаються корені рівняння, h – крок табулювання функції. Праву частину рівняння запрограмуємо як функцію.

```
program Perebor;
var a,b,h,x1,x2:real;
    function f(x:real):real;
        begin
            f:=x-exp(x*ln(0.5))-1;
        end;
begin
    writeln('I етап. Відокремлення коренів. Метод послідовного перебору.');
    write('Введіть ліву границю пошуку a='); read(a);
    write('Введіть праву границю пошуку b='); read(b);
    write('Введіть крок h='); read(h);
```

```
        writeln('Корені рівняння знаходяться в інтервалі:');
        x1:=a; x2:=a+h;
        while (x2<b+h/2) do
                begin
                        if f(x1)*f(x2)<0 then
                        writeln('a=',x1:4:2,'  b=',x2:4:2);
                        x1:=x2;  x2:=x1+h;
                end;
end.
```

Результати роботи програми приведені на рис. 5.11.

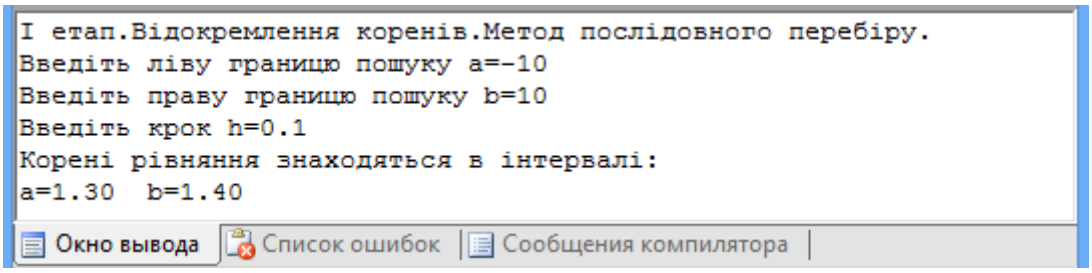


Рисунок 5.11 – Відокремлення коренів рівняння методом послідовного перебору

Метод половинного ділення. Вхідними даними будуть значення границь інтервалу ізоляції коренів, що отримано в попередній програмі, та задана точність $\varepsilon = 10^{-3}$.

```
program Polov_dilen;
var a,b,e,c:real;
k:integer;
    function f(x:real):real;
        begin
            F:=x-exp(x*ln(0.5))-1;
        end;
begin
    writeln('II етап.Уточнення коренів.');
    writeln('Метод половинного ділення.');
    write('Введіть a='); read(a);
    write('Введіть b='); read(b);
    write('Введіть точність e='); read(e);
    k:=1;
    while abs(b-a)>=e do
        begin
            c:=(a+b)/2;
            writeln('k=',k,'   x=',c:6:4,' f=',f(c):8:4);
            if f(c)=0 then
                begin
                    writeln('Корінь рівняння x=',c:6:3,' кроків ',k);
                    halt;
                end;
```

```
                if f(a)*f(c)<0 then b:=c
                                else a:=c;
                k:=k+1;
            end;
        writeln('Корінь рівняння x=',(a+b)/2:6:4,' f(',(a+b)/2:5:3,')=',f((a+b)/2):8,
                                                                    'кроків ',k);
end.
```

Результати роботи програми приведені на рис. 5.12.

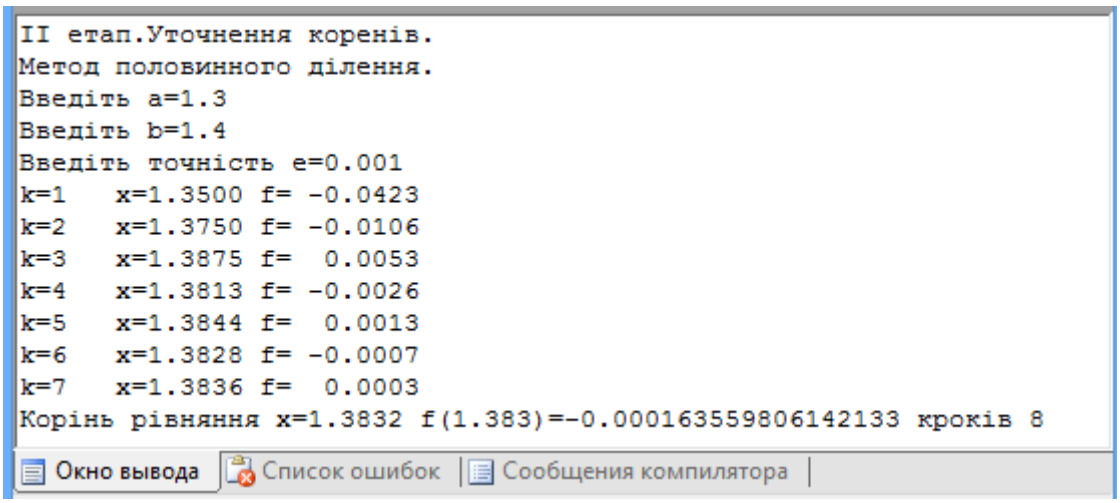


Рисунок 5.12 – Уточнення кореня рівняння
методом половинного ділення

Метод Ньютона. Вхідними даними будуть значення границь інтервалу ізоляції коренів та задана точність $\varepsilon = 10^{-3}$. Функціями f1 та f2 позначено відповідно першу та другу похідні $f'(x), f''(x)$.

```
program Newton;
var a,b,e,x,x1:real;
k:integer;
    function f(x:real):real;
      begin
            f:=x-exp(x*ln(0.5))-1;
      end;
     function f1(x:real):real;
      begin
            f1:=1-exp(x*ln(0.5))*ln(0.5);
      end;
     function f2(x:real):real;
      begin
           f2:=-exp(x*ln(0.5))*sqr(ln(0.5));
       end;
begin
      writeln('II етап.Уточнення коренів.');    writeln('Метод Ньютона.');
      write('Введіть a='); read(a);
      write('Введіть b='); read(b);
      write('Введіть точність e='); read(e);
```

```
        if f(a)*f2(a)>0 then x:=a
              else if f(b)*f2(b)>0 then x:=b
                    else
                         begin
                            writeln('Неправильно обрано відрізок ізоляції кореня.');
                            halt;
                         end;
        k:=0;
        repeat
              x1:=x;k:=k+1;
              x:=x1-f(x1)/f1(x1);
              writeln(' k= ',k,'  x=',x:6:4,' f=',f(x):8:4);
        until ((abs(x1-x)<e) and (abs(f(x))<e));
        writeln('Корінь рівняння x=',x:6:3,' f(',x:5:3,')=',f(x):8,' кроків ',k);
end.
```

Результати роботи програми приведені на рис. 5.13.

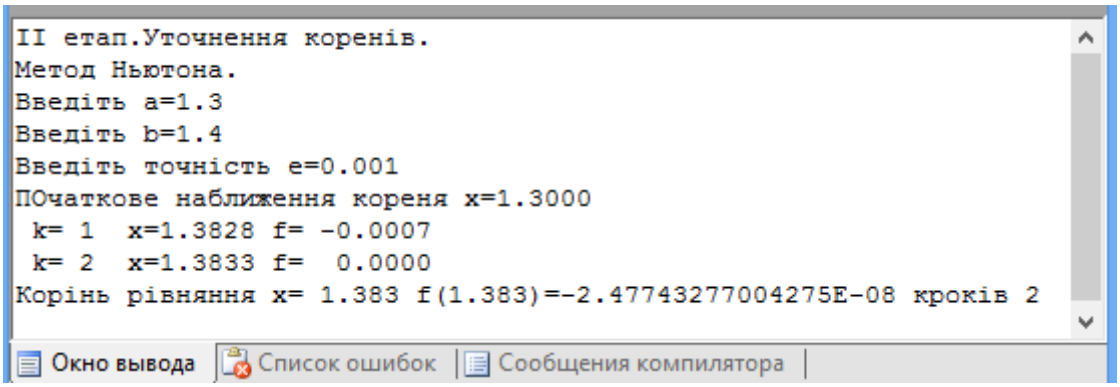


Рисунок 5.13 – Уточнення кореня рівняння методом Ньютона

Метод ітерацій. Обчислюється значення параметра q. На заданому інтервалі шукається найбільше за модулем значення першої похідної $f'(x)$ і заокруглюється до більшого цілого, але q має зберігати знак похідної.

```
program Newton;
var a,b,e,x,x1,q,h:real; k:integer;
    function f(x:real):real;
        begin
            f:=x-exp(x*ln(0.5))-1;
        end;
    function f1(x:real):real;
        begin
            f1:=1-exp(x*ln(0.5))*ln(0.5);
        end;
begin
        writeln('II етап.Уточнення коренів.');   writeln('Метод ітерацій.');
        write('Введіть a='); read(a);
        write('Введіть b='); read(b);
        write('Введіть точність e='); read(e);
        q:=f1(a);  h:=(b-a)/10;  x:=a;
```

```
        while x<=b+h/2 do
            begin
                x:=x+h;
                if abs(q)<abs(f1(x)) then q:=f1(x);
            end;
        q:=(trunc(abs(q))+1)*sign(q);
        writeln('Значення q=',q:4:2);
        x:=(a+b)/2; k:=0;
        repeat
            x1:=x; k:=k+1;
            x:=x1-f(x1)/q;
            writeln(' k= ',k,'  x=',x:6:4,' f(',x:5:3,')=',f(x):4);
        until ((abs(x1-x)<e) and (abs(f(x))<e));
        writeln('Корінь рівняння x=',x:6:3,' f(',x:5:3,')=',f(x):8,' кроків ',k);
end.
```

Результати роботи програми приведені на рис. 5.14.

```
II етап.Уточнення коренів.
Метод ітерацій.
Введіть a=1.3
Введіть b=1.4
Введіть точність e=0.001
Значення q=2.00
 k= 1  x=1.3711 f(1.371)=-0.0154380136082144
 k= 2  x=1.3789 f(1.379)=-0.00565614803915193
 k= 3  x=1.3817 f(1.382)=-0.00207504640832323
 k= 4  x=1.3827 f(1.383)=-0.00076163314229305
 k= 5  x=1.3831 f(1.383)=-0.000279602571534587
Корінь рівняння x= 1.383 f(1.383)=-0.000279602571534587 кроків 5
```

Окно вывода | Список ошибок | Сообщения компилятора

Рисунок 5.14 – Уточнення кореня рівняння методом ітерацій

Контрольні запитання

1. Що означає відокремити корені рівняння?
2. Які методи уточнення коренів нелінійних рівнянь Вам відомі?
3. Сутність методу половинного ділення. Пояснити на графіку.
4. Сутність методу Ньютона. Пояснити на графіку.
5. Сутність методу простої ітерації. Пояснити на графіку.

Перелік рекомендованої літератури

1. Фельдман Л.П. Чисельні методи в інформатиці / Л.П. Фельдман, А.І. Петренко, О.А. Дмитрієва. –К.: BHV, 2006. – 480 с.
2. Пирумов У. Г. Численные методы: учеб. пособие для студ. втузов / У.Г. Пирумов. – 4-е изд., стереотип. – М.: Дрофа, 2007. – 221, [3] с.: ил.
3. Численные методы. Сборник задач: учеб. пособие для вузов / В.Ю. Гидаспов, И.Э. Иванов, Д.Л. Ревизников и др.; под ред. У.Г. Пирумова. – М.: Дрофа, 2007. – 144 с.: ил.

ЛАБОРАТОРНА РОБОТА № 6
НАБЛИЖЕННЯ ФУНКЦІЙ

Мета: опанувати чисельними методами наближення функцій многочленами Лагранжа та Ньютона, методом найменших квадратів.

Теоретичні відомості

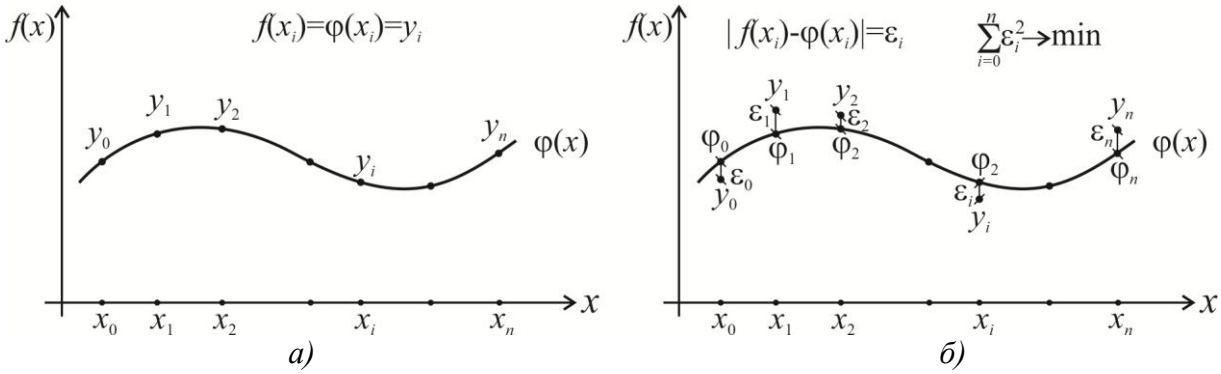
Наближенням функції називається заміна в розрахунках однієї функції $f(x)$ (відомої, невідомої або частково відомої) іншою функцією $\varphi(x)$, яка є близькою до $f(x)$ і має певні властивості, які дозволяють легко проводити над нею ті чи інші аналітичні або обчислювальні операції. Наприклад, заміна функції поліномом дозволяє отримати прості формули чисельного інтегрування і диференціювання. Заміна таблично заданої функції наближаючою функцією дозволяє отримати значення в її проміжних точках.

При підборі наближаючої функції $\varphi(x)$, треба:

- проаналізувати всі відомості про функцію $f(x)$ (як задана, яка її гладкість, які доступні значення похідних, як розміщені точки , в яких задана функція);
- визначити до якого класу повинна належати наближаюча функція $\varphi(x)$ (полином, тригонометрична, показникова, тощо);
- обрати критерій близькості між $f(x)$ і $\varphi(x)$. В якості критерію близькості можна вибрати, наприклад, точний збіг значень функції в заданих точках (*інтерполяція*, рис. 6.1, а) або мінімум відхилення наближаючої функції у заданих точках (*апроксимація*,рис. 6.2, б). Вибір критерію наближення визначається метою побудови наближаючої функції і може суттєво впливати на результати;
- вказати правило, яке дозволяє отримати значення наближаючої функції із заданою точністю в проміжних точках (які точки обрати, як їх розмістити).

Найбільшого розповсюдження в якості наближаючих функцій набули многочлени або функції, що складаються з многочленів. Така апроксимація називається поліноміальною апроксимацією або кусково-поліноміальною апроксимацією відповідно.

Серед інших сімейств функцій (тригонометричних, показникових, раціональних) многочлени привабливі тим,що вони є лінійними функціями своїх параметрів, тому їхнє обчислення зводиться до виконання простих арифметичних операцій.



а) *б)*

Рисунок 6.1 – Критерії близькості функцій

Нехай в точках $x_0, x_1, \ldots, x_n$ таких, що $a \le x_0 < x_1 < \cdots < x_n \le b$, і які будемо називати *вузлами*, відомі значення функції *f*(*x*): $f(x_0) = y_0$, $f(x_1) = y_1, \ldots$, $f(x_n) = y_n$. Тобто на $[x_0; x_n] \subseteq [a; b]$ задана таблична (сіткова) функція *f*(*x*). Функцію φ(*x*) називають *інтерполяційною* для *f*(*x*) на $[a; b]$, якщо її значення в заданих вузлах співпадають їз заданими значеннями функції *f*(*x*): $\varphi(x_0) = y_0$, $\varphi(x_1) = y_1, \ldots, \varphi(x_n) = y_n$.

Геометрична інтерпретація задачі інтерполяції: графік функції φ(*x*) проходить так, що принаймні в *n*+1 вузлі x_i він перетинає або торкається графіка функції *f*(*x*). Але зобразити таких графіків можна скільки завгодно, якщо не накладати на *f*(*x*) і φ(*x*) певних обмежень (рис. 6.2).

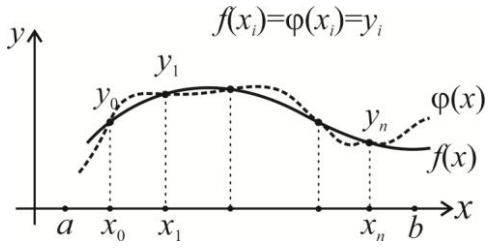


Рисунок 6.2 – Геометрична інтерпретація задачі інтерполяції

Якщо за інтерполяційну функцію обрати многочлен степеня *n*, тоді задача інтерполяції формулюється так: для функції *f*(*x*), що задана таблицею $f(x_i) = y_i$, $i = \overline{0,n}$, знайти многочлен

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n, \tag{6.1}$$

такий щоб виконувалися умови інтерполяції:

$$P_n(x_i) = y_i, \quad i = \overline{0,n}. \tag{6.2}$$

Знайти многочлен означає знайти *n*+1 його коефіцієнт $a_0, a_1, \ldots, a_n$. Для цього існує *n*+1 умова (6.2). Тоді коефіцієнти задовольняють системі рівнянь:

$$\left.\begin{aligned} &a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n = y_0, \\ &a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n = y, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n = y_n. \end{aligned}\right\} \tag{6.3}$$

Визначник системи (6.3) буде визначником Вандермонда і не дорівнюватиме нулю у випадку відсутності однакових вузлів:

$$\begin{vmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{vmatrix} = \prod_{j>i\ge 0}^{n} (x_j - x_i) = 0. \tag{6.4}$$

Тому розв'язок системи (6.3) буде існувати і буде єдиним. Тобто, інтерполяційний многочлен (6.1) теж існуватиме, і теж буде єдиним.

Многочлен Лагранжа. Розглянутий спосіб побудови інтерполяційного многочлена пов'язаний з великим обсягом обчислень. На практиці часто необхідно для таблично заданої функції отримати значення в її проміжних точках, а

знаходження інтерполяційного многочлена є необов'язковим. В такому випадку зручно застосовувати інтерполяційний многочлен в формі:

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{(x-x_0)(x-x_1)\cdot\ldots\cdot(x-x_{i-1})(x-x_{i+1})\cdot\ldots\cdot(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdot\ldots\cdot(x_i-x_{i-1})(x_i-x_{i+1})\cdot\ldots\cdot(x_i-x_n)}, \qquad (6.5)$$

що називається інтерполяційним многочленом Лагранжа. Наведемо основні переваги цієї форми запису інтерполяційного многочлена:

- кількість арифметичних операцій, які необхідні для побудови многочлена Лагранжа, пропорційно n^2 і є найменшим для усіх форм запису;
- формула (6.5) у явному вигляді містить значення функцій у вузлах інтерполяції;
- формулу (6.5) можна застосовувати як для рівновіддалених, так і для нерівно віддалених вузлів;
- інтерполяційний багаточлен Лагранжа зручний, коли значення функцій змінюються, а вузли інтерполяції незмінні, що має при багатьох експериментальних дослідженнях.

Недоліком формули Лагранжа є те, що при зміні числа вузлів треба всі обчислення виконувати знову.

Многочлен Ньютона. Іншу форму запису інтерполяційного многочлена являє собою многочлен в формі Ньютона:

$$\begin{aligned} P_n(x) = {} & f(x_0) + (x-x_0) f(x_0;x_1) + (x-x_0)(x-x_1) f(x_0;x_1;x_2) + \ldots + \\ & + (x-x_0)(x-x_1)\cdot\ldots\cdot(x-x_{n-1}) f(x_0;x_1;x_2;\ldots;x_n). \end{aligned} \qquad (6.6)$$

Тут $f(x_0;x_1), f(x_0;x_1;x_2), \ldots, f(x_0;x_1;x_2;\ldots;x_n)$ – *поділені різниці*. Уведемо поняття поділеної різниці.

Поділені різниці *нульового порядку* збігаються зі значеннями функції у вузлах $f(x_i) = y_i$. Поділені різниці *першого порядку* $f(x_i;x_j)$ визначаються через поділені різниці нульового порядку:

$$f(x_i;x_j) = \frac{y_j - y_i}{x_j - x_i};$$

поділені різниці *другого порядку* $f(x_i;x_j;x_k)$ визначаються через поділені різниці першого порядку:

$$f(x_i;x_j;x_k) = \frac{f(x_j;x_k) - f(x_i;x_j)}{x_k - x_i}.$$

Поділена різниця порядку *n* визначається співвідношенням .

$$f(x_i;x_j;x_k;\ldots;x_{n-1};x_n) = \frac{f(x_j;x_k;\ldots;x_n) - f(x_i;x_j;x_k;\ldots;x_{n-1})}{x_n - x_i}.$$

Таким чином, для *n*+1 вузла можна побудувати поділені різниці до *n*-ого порядку; поділені різниці більш високих порядків дорівнюють нулю. При побудові інтерполяційного многочлена Ньютона зручно користуватися таблицею поділених різниць. Наведемо послідовність отримання поділених різниць для числа *n*=3:

x_0	y_0	$f(x_0;x_1)$ $f(x_1;x_2)$ $f(x_2;x_3)$	$f(x_0;x_1;x_2)$ $f(x_1;x_2;x_3)$	$f(x_0;x_1;x_2;x_3)$
x_1	y_1			
x_2	y_2			
x_3	y_3			

За для підвищення точності інтерполяції до суми (6.6) можна додати нові члени, що потребує підключення нових вузлів. При цьому не має значення, в якому порядку підключаються нові вузли. Цим формула Ньютона вигідно відрізняється від формули Лагранжа.

Блок-схеми алгоритмів побудови многочленів Лагранжа і Ньютона приведені на рис. 6.3.

Метод найменших квадратів. Часто під час проведення експериментів виникає необхідність наближення функції, велика кількість точок якої знайдена з деякими похибками. В таких випадках інтерполяція стає недоцільною. Наближаючу функцію будують таким чином, щоб згладити вплив похибок вимірювання і кількості точок експерименту. Таке згладжування можливе при побудові наближаючої функції за методом найменших квадратів.

Вигляд наближаючої функції може бути довільним, але нехай це буде многочлен:

$$P_m(x)=a_0 + a_1x + a_2x^2 + \ldots + a_mx^m, \quad (m\le n). \tag{6.7}$$

Критерієм близькості може бути мінімізація суми квадратів відхилень значень заданої та наближаючої функцій у вузлах $x_i, i=\overline{0,n}$:

$$\Phi = \sum_{j=0}^{n}(P_m(x_j) - y_j)^2 \to min. \tag{6.8}$$

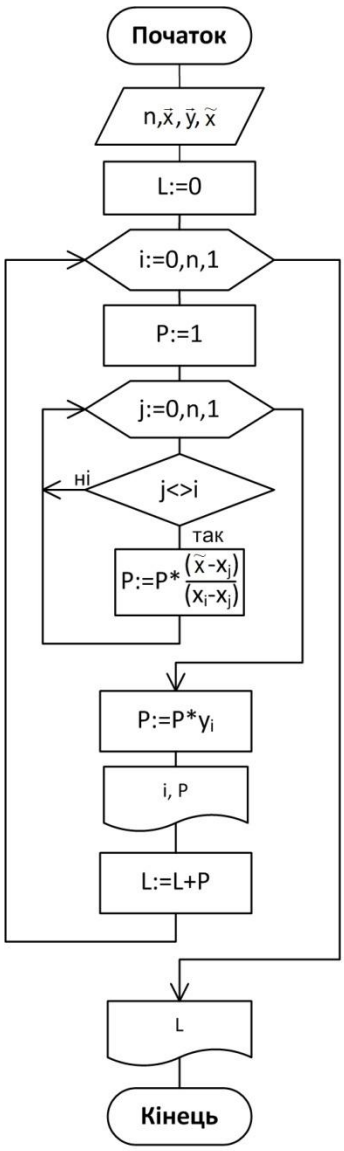
Мінімуму можливо досягти підбором коефіцієнтів многочлена (6.7). Оскільки сума квадратів відхилень апроксимуючої функції від заданої повинна бути мінімальною, тоді частинні похідні функції повинні дорівнювати нулю:

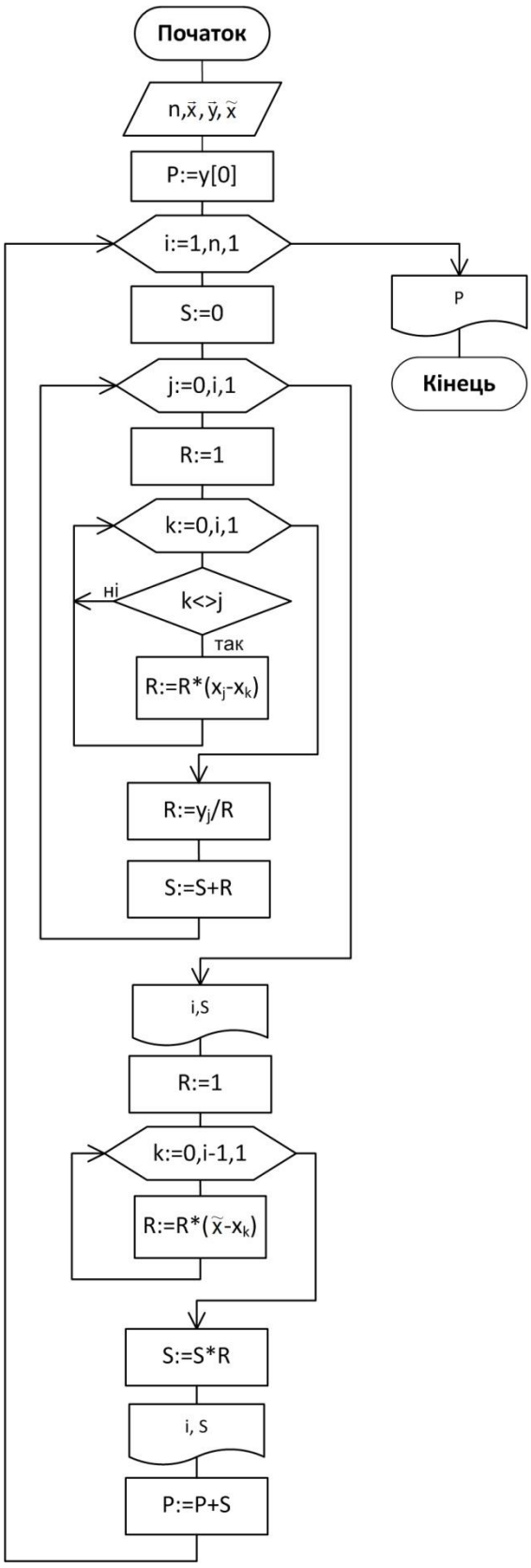
$$\frac{\partial\Phi}{\partial a_k} = 2\sum_{j=0}^{n}\Big[\sum_{i=0}^{m} a_i x_j^i - y_j\Big]x_j^k=0, \quad k=0, 1, \ldots, m.$$

Остаточно отримаємо систему:

$$\begin{cases} (n+1)\cdot a_0 + a_1\cdot\sum\limits_{j=0}^{n}x_j + a_2\cdot\sum\limits_{j=0}^{n}x_j^2 + \ldots + a_m\cdot\sum\limits_{j=0}^{n}x_j^m = \sum\limits_{j=0}^{n}y_j, \\ a_0\cdot\sum\limits_{j=0}^{n}x_j + a_1\cdot\sum\limits_{j=0}^{n}x_j^2 + a_2\cdot\sum\limits_{j=0}^{n}x_j^3 + \ldots + a_m\cdot\sum\limits_{j=0}^{n}x_j^{m+1} = \sum\limits_{j=0}^{n}y_jx_j, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_0\cdot\sum\limits_{j=0}^{n}x_j^m + a_1\cdot\sum\limits_{j=0}^{n}x_j^{m+1} + a_2\cdot\sum\limits_{j=0}^{n}x_j^{m+2} + \ldots + a_m\cdot\sum\limits_{j=0}^{n}x_j^{2m} = \sum\limits_{j=0}^{n}y_jx_j^m. \end{cases} \tag{6.9}$$

Система (6.9) називається *нормальною системою методу найменших квадратів* і являє собою систему лінійних алгебраїчних рівнянь відносно коефіцієнтів апроксимуючої функції a_0, a_1, a_2,…, a_m. Розв’язавши систему, можна побудувати многочлен (6.7), який наближає задану функцію *f(x)* і мінімізує квадратичне відхилення (6.8). Блок-схема алгоритму приведена на рис. 6.4.





а) многочлен Лагранжа *б) многочлен Ньютона*

Рисунок 6.3 – Блок-схема алгоритмів інтерполяції многочленами

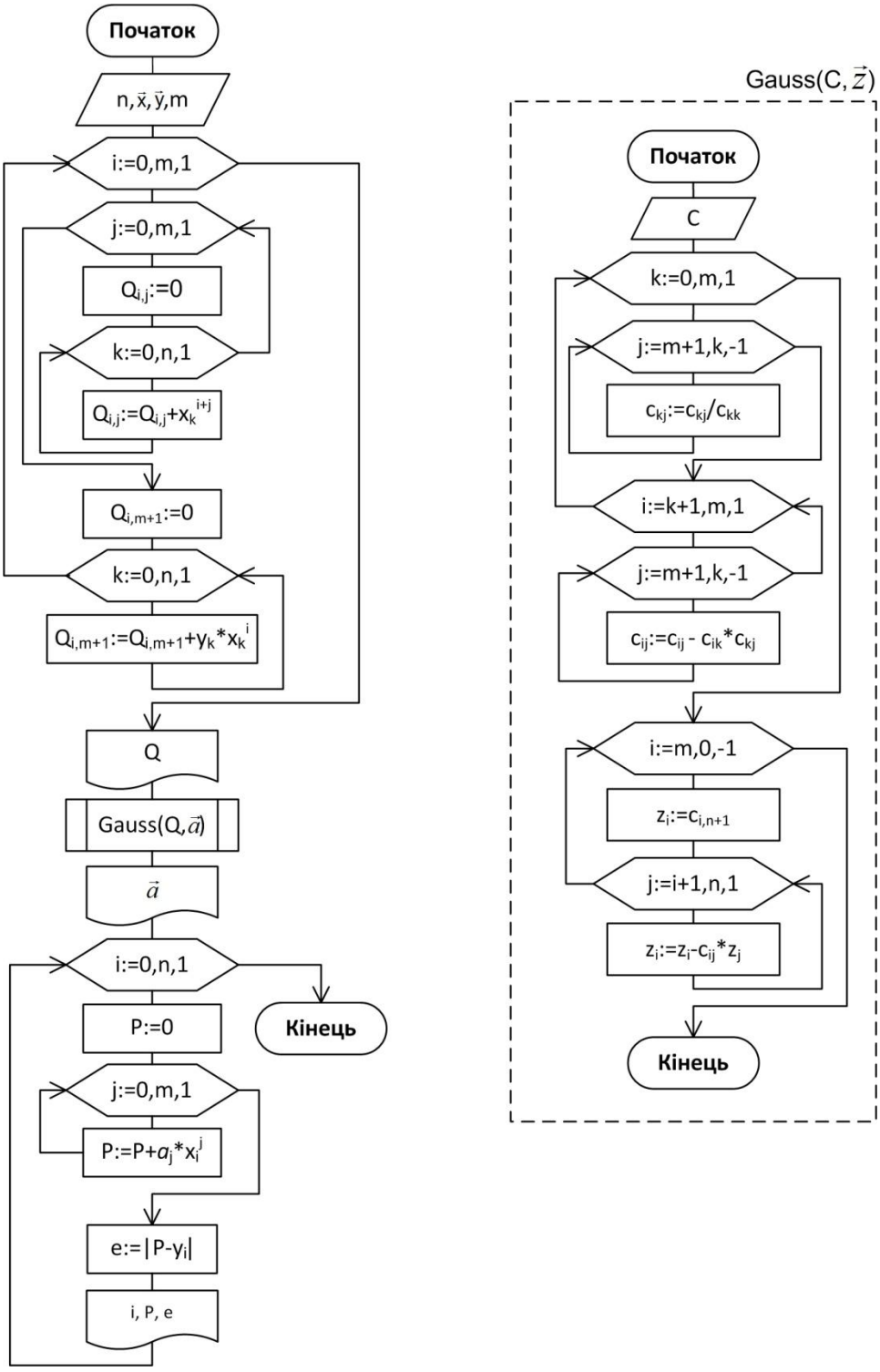


Рисунок 6.4 – Блок-схема методу найменших квадратів

У блок-схемі на рис. 6.4 прийняті такі позначення: Q - матриця коефіцієнтів системи, вектор $\vec{a}$ - вектор коефіцієнтів многочлена, P - значення многочлена в певній точці, e - похибка многочлена в певній точці. Систему рівнянь (6.9) пропонується розв'язувати за методом Гауса, який оформлений як процедура з формальними параметрами: C - розширена матриця коефіцієнтів, $\vec{z}$ - вектор невідомих.

Завдання

1. Використовуючи інтерполяційні поліноми Лагранжа й Ньютона, знайти наближене значення функції у точці $\tilde{x}$.

1.

x	0	1	2	3	4	$\tilde{x}=5$
y	1	-3	25	129	381	

2.

x	14	18	31	35	38	$\tilde{x}=30$
y	68	64	44	39	32	

3.

x	2,0	2,2	2,5	3,0	3,3	$\tilde{x}=2,4$
y	1,41	1,48	1,59	1,79	1,82	

4.

x	0,0	0,25	1,25	2,12	3,25	$\tilde{x}=1,2$
y	2,0	1,6	2,32	2,02	2,83	

5.

x	0,0	0,5	1,4	2,25	3,5	$\tilde{x}=1,45$
y	2,0	1,7	2,36	2,33	3,17	

6.

x	0,0	0,75	1,6	2,36	3,75	$\tilde{x}=2,6$
y	3,0	2,8	3,7	3,5	4,0	

7.

x	0,0	1,0	1,8	2,5	4,0	$\tilde{x}=2,0$
y	3,0	2,9	3,6	3,8	4,3	

8.

x	-0,25	0,0	1,0	2,1	3,0	$\tilde{x}=1,5$
y	4,0	3,6	4,56	4,02	3,83	

9.

x	-0,5	0,25	1,1	1,88	3,25	$\tilde{x}=2,5$
y	5,0	4,8	5,7	5,5	5,0	

10.

x	-2,5	-1,5	-0,7	0,0	1,5	$\tilde{x}=1,2$
y	6,0	5,9	6,9	6,7	5,83	

11.

x	-1,0	-0,5	0,4	1,25	2,5	$\tilde{x}=0,2$
y	1,9	1,4	2,3	2,5	3,1	

12.

x	0,0	0,25	1,25	2,12	3,25	$\tilde{x}=1,5$
y	4,0	3,9	4,5	4,7	4,95	

13.

x	-0,3	0	1,05	2,1	3,3	$\tilde{x}=2,0$
y	4,2	3,6	4,57	4,02	3,8	

14.

x	1,5	1,63	1,8	2,2	2,7	$\tilde{x}=1,6$
y	3,5	3,83	4,3	4,7	5,2	

15.

x	3,2	3,6	5,8	5,9	6,2	$\tilde{x}=4,0$
y	-9,1	-6,3	-2,0	1,2	3,5	

16.

x	0,3	0,55	1,55	2,32	3,8	$\tilde{x}=2,1$
y	3,0	2,6	3,32	3,02	5,0	

17.

x	-0,4	0,1	1,85	3,1	4,0	$\tilde{x}=1,3$
y	-9,1	-8,3	-2,1	1,5	3,4	

18.

x	10	12	22	31	35	$\tilde{x}=25$
y	43	21	17	13	8	

19.

x	-3,0	-2,8	-2,5	-2,0	-1,7	$\tilde{x}=-1,0$
y	1,41	1,5	1,57	1,79	1,82	

20.

x	4,0	5,0	6,0	7,0	8,0	$\tilde{x}=5,8$
y	6,0	7,0	8,0	9,0	10,0	

2. Для заданої таблично функції методом найменших квадратів знайти наближаючий многочлен третього степеня.

1.

x	-1,0	0,0	1,0	2,0	3,0	4,0
y	-0,5	0,0	0,5	0,8663	1,0	0,8663

2.

x	-1,0	0,0	1,0	2,0	3,0	4,0
y	0,866	1,0	0,866	0,5	0,0	-0,5

3.

x	-0,9	0,0	0,9	1,8	2,7	3,6
y	-0,369	0,0	0,369	0,854	1,786	6,314

4.

x	1,0	1,9	2,8	3,7	4,6	5,5
y	2,412	1,082	0,509	0,118	-0,240	-0,668

5.

x	0,1	0,5	0,9	1,3	1,7	2,1
y	-2,302	-0,693	-0,105	0,262	0,530	0,741

6.

x	-3,0	-2,0	-1,0	0,0	1,0	2,0
y	0,049	0,135	0,367	1,0	2,718	7,389

7.

x	0,0	0,2	0,4	0,6	0,8	1,0
y	1,0	1,003	1,0511	1,259	1,819	3,0

8.

x	-0,7	-0,4	-0,1	0,2	0,5	0,8
y	-0,775	-0,411	-0,100	-0,201	0,523	0,927

9.

x	-0,7	-0,4	-0,1	0,2	0,5	0,8
y	2,346	1,982	1,671	1,369	1,047	0,643

10.

x	-0,5	-3,0	-1,0	1,0	3,0	5,0
y	-1,373	-1,249	-0,785	0,785	1,249	1,373

11.

x	-0,5	-0,3	-0,1	1,0	3,0	5,0
y	2,944	2,819	2,356	0,785	0,321	0,197

12.

x	-1,0	0,0	1,0	2,0	3,0	4,0
y	-1,841	0,0	1,841	2,909	3,141	3,243

13.

x	-1,0	0,0	1,0	2,0	3,0	4,0
y	-0,459	1,0	1,540	1,583	2,010	3,346

14.

x	-0,9	0,0	0,9	1,8	2,7	3,6
y	-1,268	0,0	1,268	2,654	4,485	9,913

15.

x	1,0	1,9	2,8	3,7	4,6	5,5
y	3,414	2,981	3,309	3,818	4,359	4,831

16.

x	0,1	0,5	0,9	1,3	1,7	2,1
y	-2,202	-0,193	0,794	1,562	2,230	2,841

17.

x	-3,0	-2,0	-1,0	0,0	1,0	2,0
y	-2,950	-1,864	-0,632	1,0	3,718	9,389

18.

x	0,0	1,7	3,4	5,1	6,8	8,5
y	0,0	3,003	5,243	7,358	9,407	11,415

19.

x	-0,7	-0,4	-0,1	0,2	0,5	0,8
y	-1,475	-0,811	-0,200	0,401	1,023	1,727

20.

x	-0,7	-0,4	-0,1	0,2	0,5	0,8
y	1,646	1,582	1,571	1,569	1,547	1,443

Послідовність виконання лабораторної роботи

1. Використовуючи інтерполяційні поліноми Лагранжа й Ньютона, знайти наближене значення функції у точці $\widetilde{x}$.

x	-1,0	-0,2	1,8	2,7	4,0	$\widetilde{x}=2{,}0$
y	5,0	4,6	5,7	5,2	4,3	

Функція задана в п'яти вузлах, які пронумеруємо від 0 до 4 (n=4):

x_0=-1,0; x_1=-0,2; x_2=1,8; x_3=2,7; x_4=4,0;

y_0=5,0; y_1=4,6; y_2=5,7; y_3=5,2; y_4=4,3.

Побудуємо інтерполяційний поліном Лагранжа четвертого степеня:

$$L_4(x) = y_0\frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)} + y_1\frac{(x-x_0)(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)} +$$
$$+y_2\frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)} + y_3\frac{(x-x_0)(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_3-x_4)} +$$
$$+y_4\frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_1)(x_4-x_2)(x_4-x_3)}.$$

Підставимо задані числові значення і обчислимо кожний доданок окремо:

$$L_4^{(0)}(\tilde{x}) = 5\frac{(2+0{,}2)(2-1{,}8)(2-2{,}7)(2-4)}{(-1+0{,}2)(-1-1{,}8)(-1-2{,}7)(-1-4)} = 0{,}0743;$$

$$L_4^{(1)}(\tilde{x}) = 4{,}6\frac{(2+1)(2-1{,}8)(2-2{,}7)(2-4)}{(-0{,}2+1)(-0{,}2-1{,}8)(-0{,}2-2{,}7)(-0{,}2-4)} = -0{,}1983;$$

$$L_4^{(2)}(\tilde{x}) = 5{,}7\,\frac{(2+1)(2+0{,}2)(2-2{,}7)(2-4)}{(1{,}8+1)(1{,}8+0{,}2)(1{,}8-2{,}7)(1{,}8-4)} = 4{,}75;$$

$$L_4^{(3)}(\tilde{x}) = 5{,}2\,\frac{(2+1)(2+0{,}2)(2-1{,}8)(2-4)}{(2{,}7+1)(2{,}7+0{,}2)(2{,}7-1{,}8)(2{,}7-4)} = 1{,}0935;$$

$$L_4^{(4)}(\tilde{x}) = 4{,}3\,\frac{(2+1)(2+0{,}2)(2-1{,}8)(2-2{,}7)}{(4+1)(4+0{,}2)(4-1{,}8)(4-2{,}7)} = -0{,}0662;$$

і знайдемо значення суми всіх доданків:

$$L_4(\tilde{x}) = L_4^{(0)}(\tilde{x}) + L_4^{(1)}(\tilde{x}) + L_4^{(2)}(\tilde{x}) + L_4^{(3)}(\tilde{x}) + L_4^{(4)}(\tilde{x}) =$$
$$= 0{,}0743 - 0{,}1983 + 4{,}75 + 1{,}0935 - 0{,}0662 = 5{,}6534$$

Побудуємо інтерполяційний поліном Ньютона четвертого степеня:

$$P_4(x) = y_0 + (x-x_0)f(x_0;x_1) + (x-x_0)(x-x_1)f(x_0;x_1;x_2) +$$
$$+(x-x_0)(x-x_1)(x-x_2)f(x_0;x_1;x_2;x_3) +$$
$$+(x-x_0)(x-x_1)(x-x_2)(x-x_3)f(x_0;x_1;x_2;x_3;x_4).$$

Обчислимо поділені різниці:

i	x_i	y_i	$f(x_i;x_{i+1})$	$f(x_i;x_{i+1};x_{i+2})$	$f(x_i;x_{i+1};x_{i+2};x_{i+3})$	$f(x_i;x_{i+1};x_{i+2};x_{i+3})$
0	-1,0	5	-0,5	0,375	-0,2044	0,0561
1	-0,2	4,6	0,55	-0,3812	0,07598	
2	1,8	5,7	-0,5556	-0,0621		
3	2,7	5,2	-0,6923			
4	4,0	4,3				

$$f(x_0;x_1) = \frac{y_1-y_0}{x_1-x_0} = \frac{4{,}6-5}{-0{,}2+1} = -0{,}5; \qquad f(x_1;x_2) = \frac{y_2-y_1}{x_2-x_1} = \frac{5{,}7-4{,}6}{1{,}8+0{,}2} = 0{,}55;$$

$$f(x_2;x_3) = \frac{y_3-y_2}{x_3-x_2} = \frac{5{,}2-5{,}7}{2{,}7-1{,}8} = -0{,}5556; \qquad f(x_3;x_4) = \frac{y_4-y_3}{x_4-x_3} = \frac{4{,}3-5{,}2}{4{,}0-2{,}7} = -0{,}6923;$$

$$f(x_0;x_1;x_2) = \frac{f(x_1;x_2)-f(x_0;x_1)}{x_2-x_0} = \frac{0{,}55+0{,}5}{1{,}8+1{,}0} = 0{,}375;$$

$$f(x_1;x_2;x_3) = \frac{f(x_2;x_3)-f(x_1;x_2)}{x_3-x_1} = \frac{-0{,}5556-0{,}55}{2{,}7+0{,}2} = -0{,}3812;$$

$$f(x_2;x_3;x_4) = \frac{f(x_3;x_4)-f(x_2;x_3)}{x_4-x_2} = \frac{-0{,}6923+0{,}5556}{4{,}0-1{,}8} = -0{,}0621;$$

$$f(x_0;x_1;x_2;x_3) = \frac{f(x_1;x_2;x_3)-f(x_0;x_1;x_2)}{x_3-x_0} = \frac{-0{,}3812-0{,}375}{2{,}7+1{,}0} = -0{,}2044;$$

$$f(x_1;x_2;x_3;x_4) = \frac{f(x_2;x_3;x_4)-f(x_1;x_2;x_3)}{x_4-x_1} = \frac{-0{,}0621+0{,}3812}{4{,}0+0{,}2} = 0{,}07598;$$

$$f(x_0;x_1;x_2;x_3;x_4) = \frac{f(x_1;x_2;x_3;x_4)-f(x_0;x_1;x_2;x_3)}{x_4-x_0} = \frac{0{,}07598+0{,}2044}{4{,}0+1{,}0} = 0{,}0561;$$

підставимо числові значення:

$$P_4(\tilde{x}) = 5 + (2+1)(-0{,}5) + (2+1)(2+0{,}2)0{,}375 + (2+1)(2+0{,}2)(2-1{,}8)(-0{,}2044) +$$
$$+(2+1)(2+0{,}2)(2-1{,}8)(2-2{,}7)0{,}0561 = 5 - 1{,}5 + 2{,}475 - 0{,}2698 - 0{,}0518 = 5{,}6534.$$

Відповіді співпали, що говорить про правильність обчислень.

2. Для заданої таблично функції методом найменших квадратів знайти наближаючий многочлен третього степеня.

x	0,0	1,7	3,4	5,1	6,8	8,5
y	0,0	1,3038	1,8439	2,2583	2,6077	2,9155

Функція задана в шости вузлах, які пронумеруємо від 0 до 5 (n=5):
x_0=0,0; x_1=1,7; x_2=3,4; x_3=5,1; x_4=6,8; x_5=8,5;
y_0=0,0; y_1=1,3038; y_2=1,8439; y_3=2,2583; y_4=2,6077; y_5=2,9155.
Запишемо наближаючий многочлен третього степеня (m=3):

$$P_3(x)=a_0 + a_1x + a_2x^2 + a_3x^3.$$

Для знаходження невідомих коефіцієнтів a_0, a_1, a_2, a_3 запишемо нормальну систему метода найменших квадратів:

$$\begin{cases} (5+1)\cdot a_0 + a_1\cdot\sum\limits_{j=0}^{5} x_j + a_2\cdot\sum\limits_{j=0}^{5} x_j^2 + a_3\cdot\sum\limits_{j=0}^{5} x_j^3 = \sum\limits_{j=0}^{5} y_j, \\ a_0\cdot\sum\limits_{j=0}^{5} x_j + a_1\cdot\sum\limits_{j=0}^{5} x_j^2 + a_2\cdot\sum\limits_{j=0}^{5} x_j^3 + a_3\cdot\sum\limits_{j=0}^{n} x_j^{3+1} = \sum\limits_{j=0}^{n} y_j x_j, \\ a_0\cdot\sum\limits_{j=0}^{5} x_j^2 + a_1\cdot\sum\limits_{j=0}^{5} x_j^3 + a_2\cdot\sum\limits_{j=0}^{5} x_j^{3+1} + a_3\cdot\sum\limits_{j=0}^{5} x_j^{3+2} = \sum\limits_{j=0}^{5} y_j x_j^2, \\ a_0\cdot\sum\limits_{j=0}^{5} x_j^3 + a_1\cdot\sum\limits_{j=0}^{5} x_j^{3+1} + a_2\cdot\sum\limits_{j=0}^{5} x_j^{3+2} + a_3\cdot\sum\limits_{j=0}^{5} x_j^{2\cdot 3} = \sum\limits_{j=0}^{5} y_j x_j^3. \end{cases}$$

Обчислимо коефіцієнти системи:

$$\sum_{j=0}^{5} x_j = 25,5; \quad \sum_{j=0}^{5} x_j^2 = 158,95; \quad \sum_{j=0}^{5} x_j^3 = 1105,425;$$

$$\sum_{j=0}^{5} x_j^4 = 8176,706; \quad \sum_{j=0}^{5} x_j^5 = 62828,672; \quad \sum_{j=0}^{5} x_j^6 = 495182,228;$$

$$\sum_{j=0}^{5} y_j = 10,929; \quad \sum_{j=0}^{n} y_j x_j = 62,517; \quad \sum_{j=0}^{5} y_j x_j^2 = 415,047; \quad \sum_{j=0}^{5} y_j x_j^3 = 2988,87.$$

Запишемо остаточно систему:

$$\begin{cases} 6a_0 + 25,5a_1 + 158,95a_2 + 1105,425a_3 = 10,929, \\ 25,5a_0 + 158,95a_1 + 1105,425a_2 + 8176,706a_3 = 62,517, \\ 158,95a_0 + 1105,425a_1 + 8176,706a_2 + 62828,672a_3 = 415,047, \\ 1105,425a_0 + 8176,706a_1 + 62828,672a_2 + 495182,228a_3 = 2988,87. \end{cases}$$

Розв'язки системи можна знайти будь яким відомим методом, наприклад методом Гауса:

$$a_0=0,0241; \quad a_1=0,9023; \quad a_2= -0,1266; \quad a_3=0,0071;$$

тоді апроксимуюча функція буде мати вигляд: $P_3(x)=0,0241+0,9023x-0,1266x^2+0,0071x^3$.

Знайдемо значення функції в заданих точках і обчислимо похибку:

$P_3(0,0)=0,0241+0,9023\cdot0,0-0,1266\cdot0,0^2+0,0071\cdot0,0^3=0,0241$; $|0,0241-0,0|=0,0241$;
$P_3(1,7)=0,0241+0,9023\cdot1,7-0,1266\cdot1,7^2+0,0071\cdot1,7^3=1,2272$; $|1,2272-1,3038|=0,0766$;
$P_3(3,4)=0,0241+0,9023\cdot3,4-0,1266\cdot3,4^2+0,0071\cdot3,4^3=1,9092$; $|1,9092-1,8439|=0,0653$;
$P_3(5,1)=0,0241+0,9023\cdot5,1-0,1266\cdot5,1^2+0,0071\cdot5,1^3=2,2808$; $|2,2808-2,2583|=0,0225$;
$P_3(6,8)=0,0241+0,9023\cdot6,8-0,1266\cdot6,8^2+0,0071\cdot6,8^3=2,5526$; $|2,5526-2,6077|=0,0551$;
$P_3(8,5)=0,0241+0,9023\cdot8,5-0,1266\cdot8,5^2+0,0071\cdot8,5^3=2,9353$; $|2,9353-2,9155|=0,0198$.

Наведемо фрагменти програм, що реалізують розглянуті алгоритми.

Інтерполяційні многочлени Лагранжа і Ньютона. Вхідними даними

будуть *n* - вимірність задачі (номер останнього вузла, вузли нумеруються з 0), значення вузлів (масив *x*) і значення функції (масив *y*), а також точка, в якій шукається значення функції (*xx*).

Многочлен Лагранжа:

```
Program Lagrange;
var
    x,y:array[0..10] of real;
    n,i,j: integer;
    xx,L,P:real;
begin
        writeln('Задача інтерполяції. Поліном Лагранжа.');
        write('Задайте число: n='); read(n);
        writeln('Задайте значення вузлів:');
        for i:=0 to n do
                begin
                        write('x[',i,']=');
                        read(x[i]);
                end;
        writeln('Задайте значення функції:');
        for i:=0 to n do
            begin
                write('y[',i,']=');
                read(y[i]);
            end;
        write('Задайте значення точки, в якій шукаємо функцію: xx='); read(xx);
        L:=0;
        for i:=0 to n do
                begin
                        P:=1;
                        for j:=0 to n do if j<>i then P:=P*(xx-x[j])/(x[i]-x[j]);
                        P:=P*y[i];
                        writeln('Доданок L(',i,')=',P:6:4);
                        L:=L+P;
                end;
        writeln('L=',L:8:4);
end.
```

Результати роботи програми приведені на рис. 6.5.

Многочлен Ньютона:

```
Program Lagrange;
var
        x,y:array[0..10] of real;
        n,i,j,k: integer;
        xx,P,S,R:real;
begin
        writeln('Задача інтерполяції. Поліном Лагранжа.');
        write('Задайте число: n='); read(n);
        writeln('Задайте значення вузлів:');
```

```
        for i:=0 to n do
                begin
                        write('x[',i,']=');
                        read(x[i]);
                end;
        writeln('Задайте значення функції:');
        for i:=0 to n do
                begin
                        write('y[',i,']=');
                        read(y[i]);
                end;
        write('Задайте значення точки, в якій шукаємо функцію: xx='); read(xx);
        P:=y[0];
        for i:=1 to n do
                begin
                        S:=0;
                        for j:=0 to i do
                                begin
                                        R:=1;
                                        for k:=0 to i do if k<>j then R:=R*(x[j]-x[k]);
                                        R:=y[j]/R;
                                        S:=S+R;
                                end;
                        writeln('Поділена різниця порядку ',i,'  -  ',S:6:4);
                        R:=1;
                        for k:=0 to i-1 do R:=R*(xx-x[k]);
                        S:=S*R;
                        writeln('Доданок P(',i,')=',S:6:4);
                        P:=P+S;
                end;
        writeln('P=',P:8:4);
end.
```

Результати роботи програми приведені на рис. 6.6.

Метод найменших квадратів. Вхідними даними будуть *n* - вимірність задачі (номер останнього вузла, вузли нумеруються з 0), значення вузлів (масив *x*) і значення функції (масив *y*), а також степінь апроксимуючого поліному (*m*).

```
Program MetKvad;
type matrix=array[0..10,0..10] of real;
        vector=array[0..10] of real;
var
        n,m,i,j,k: byte;
        x,y,a:vector;
        Q:matrix;
        P,e:real;
```

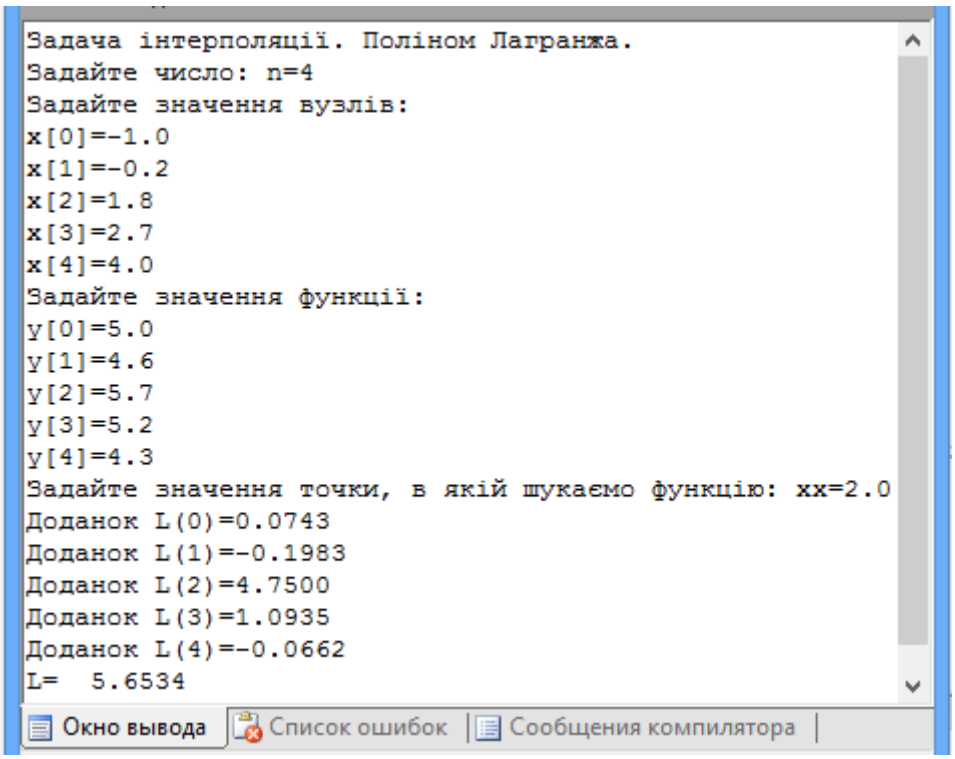


Рисунок 6.5 – Інтерполяція многочленом Лагранжа

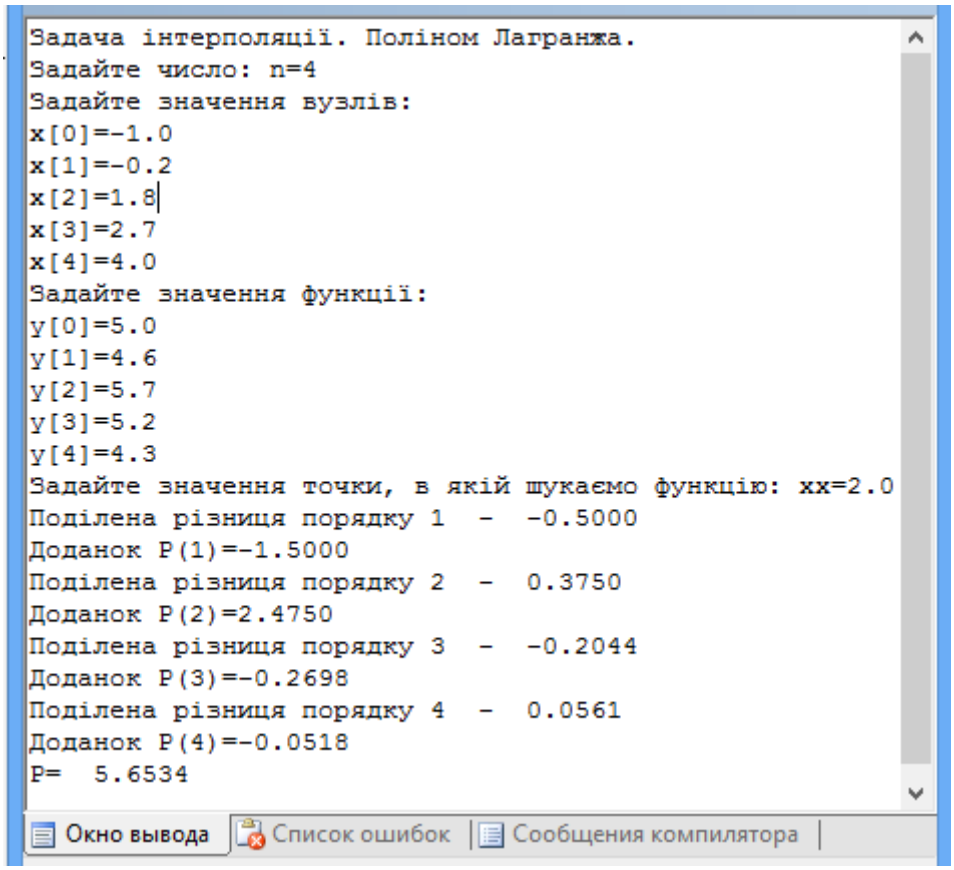


Рисунок 6.6 – Інтерполяція многочленом Ньютона

```
procedure Gauss(C:matrix;var z:vector);
var i,j,k: integer;
begin
	{Прямий хід}
	for k:=0 to m do
		begin
			for j:=m+1 downto k do C[k,j]:=C[k,j]/C[k,k];
			for i:=k+1 to m do
				for j:=m+1 downto k do C[i,j]:=C[i,j]-C[i,k]*C[k,j];
		end;
	{Зворотний хід}
	for i:=m downto 0 do
		begin
			z[i]:=C[i,m+1];
			for j:=i+1 to m do z[i]:=z[i]-C[i,j]*z[j];
		end;
end;

begin
	writeln('Задача апроксимації.Метод найменших квадратів.');
	write('Задайте число: n='); read(n);
	writeln('Задайте значення вузлів:');
	for i:=0 to n do
		begin
			write('x[',i,']=');
			read(x[i]);
		end;
	writeln('Задайте значення функції:');
	for i:=0 to n do
		begin
			write('y[',i,']=');
			read(y[i]);
		end;
	write('Задайте степінь апроксимуючого полінома m='); read(m);
	{Формування матриці системи}
	for i:=0 to m do
		begin
			for j:=0 to m do
				begin
					Q[i,j]:=0;
					for k:=0 to n do Q[i,j]:=Q[i,j]+power(x[k],i+j);
				end;
			Q[i,m+1]:=0;
			for k:=0 to n do Q[i,m+1]:=Q[i,m+1]+y[k]*power(x[k],i);
		end;
	write('Отримана система:');
	writeln('		Q		|	q ');
```

```
    for i:=0 to m do
            begin
                for j:=0 to m do write(Q[i,j]:12:3);
                writeln('  |  ', Q[i,m+1]:12:3);
            end;
    writeln('----------------------------------------------------------------------');
    Gauss(Q,a);
    writeln('Отримали поліном:'); Write('P(x)=');
    for i:=0 to m do write(a[i]:8:4,'*X^',i,'   ');
    writeln; writeln('Значення полінома в заданих точках:');
    for i:=0 to n do
            begin
                P:=0;
                 for j:=0 to m do P:=P+a[j]*power(x[i],j);
                 e:=abs(P-y[i]);
                writeln('P(x',i,')=', P:8:4,' похибка:',e:8:4);
            end;
end.
```

Результати роботи програми приведені на рис. 6.7.

```
Задача апроксимації.Метод найменших квадратів.
Задайте число: n=5
Задайте значення вузлів:
x[0]=0.0
x[1]=1.7
x[2]=3.4
x[3]=5.1
x[4]=6.8
x[5]=8.5
Задайте значення функції:
y[0]=0.0
y[1]=1.3038
y[2]=1.8439
y[3]=2.2583
y[4]=2.6077
y[5]=2.9155
Задайте степінь апроксимуючого полінома m=3
Отримана система:                   Q                |       q
       6.000      25.500     158.950    1105.425  |        10.929
      25.500     158.950    1105.425    8176.706  |        62.517
     158.950    1105.425    8176.706   62828.672  |       415.047
    1105.425    8176.706   62828.672  495182.228  |      2988.870
------------------------------------------------------------------------
Отримали поліном:
P(x)=  0.0241*X^0      0.9023*X^1     -0.1266*X^2      0.0071*X^3
Значення полінома в заданих точках:
P(x0)=  0.0241 похибка:  0.0241
P(x1)=  1.2272 похибка:  0.0766
P(x2)=  1.9092 похибка:  0.0653
P(x3)=  2.2808 похибка:  0.0225
P(x4)=  2.5526 похибка:  0.0551
P(x5)=  2.9353 похибка:  0.0198
```


Рисунок 6.7 – Апроксимація методом найменших квадратів

Контрольні запитання

1. Випадки застосування наближення функцій?
2. Що означає підібрати наближаючу функцію?
3. Пояснити поняття інтерполяції.
4. Переваги й недоліки інтерполяції поліномом Лагранжа.
5. Поняття поділеної різниці.
6. Інтерполяційний поліном Ньютона, його переваги.
7. Як оцінити похибку інтерполяції?
8. Пояснити поняття апроксимації.
9. Що являє собою нормальна система метода найменших квадратів?

Перелік рекомендованої літератури

1. Фельдман Л.П. Чисельні методи в інформатиці / Л.П. Фельдман, А.І. Петренко, О.А. Дмитрієва. –К.: BHV, 2006. – 480 с.
2. Пирумов У. Г. Численные методы: учеб. пособие для студ. втузов / У.Г. Пирумов. – 4-е изд., стереотип. – М.: Дрофа, 2007. – 221, [3] с.: ил.
3. Численные методы. Сборник задач: учеб. пособие для вузов / В.Ю. Гидаспов, И.Э. Иванов, Д.Л. Ревизников и др.; под ред. У.Г. Пирумова. – М.: Дрофа, 2007. – 144 с.: ил.

ЛАБОРАТОРНА РОБОТА № 7
ЧИСЕЛЬНЕ ІНТЕГРУВАННЯ

Мета: опанувати чисельними методами обчислення визначених однократних інтегралів за формулами лівих, правих, середніх прямокутників, трапецій, Сімпсона.

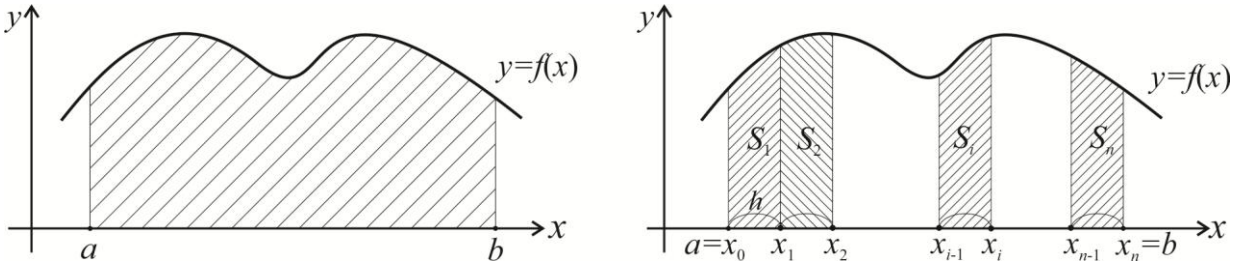
Теоретичні відомості

Якщо функція $f(x)$ – неперервна на відрізку $[a,b]$ і відома її первісна функція $F(x)$, то визначений інтеграл від цієї функції у границях від a до b може бути обчислений за формулою Ньютона-Лейбниця:

$$I = \int_a^b f(x)dx = F(b) - F(a), \quad \text{де} \quad F'(x) = f(x). \tag{7.1}$$

Але в багатьох випадках первісна функція $F(x)$ не може бути знайдена, або є дуже складною. Крім того, підінтегральна функція $f(x)$ може бути задана таблично, тоді поняття первісної функції взагалі втрачає сенс. Постає задача наближеного обчислення інтегралів за допомогою чисельних методів.

Усі чисельні методи обчислення інтегралів базуються на геометричній інтерпретації визначеного інтеграла, значення якого чисельно дорівнює площі фігури, що обмежена зверху – графіком функції $f(x)$, знизу – віссю $0x$, зліва та справа – межами інтегрування $x=a$, $x=b$ (рис. 7.1, *а*).



а *б*

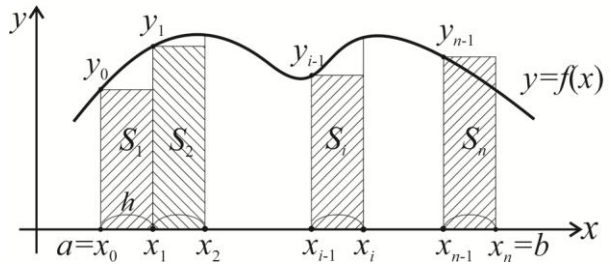
Рисунок 7.1 – Геометричний зміст інтегралу

Чисельне інтегрування основане на тому, що відрізок інтегрування $[a,b]$ розбивають на n рівних частин - відрізків $[x_i, x_{i+1}]$ ($i = \overline{0,n}$) довжиною h, де $h = \dfrac{b-a}{n}$. Кожен з відрізків є основою геометричної фігури (елементарної криволінійної трапеції), площу якої знаходять наближено як S_i, а значення інтегралу I визначають як суму таких площин S_i (рис. 7.1, *б*), тобто

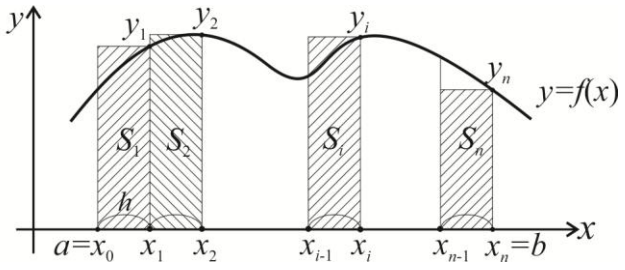
$$I = \sum_{i=1}^{n} S_i \tag{7.2}$$

Метод прямокутників - найпростіший метод наближеного обчислення інтеграла. Апроксимуємо площі елементарних криволінійних трапецій S_i площами прямокутників, висота яких дорівнюватиме значенню $f(x)$, а основа h. Значення $f(x)$ обчислюється або в лівому кінці інтервалу $[x_{i-1}, x_i]$ – $f(x_{i-1}) = y_{i-1}$ (рис. 7.2, *а*); або в правому кінці – $f(x_i)=y_i$ (рис. 7.2, *б*); або в середині інтервалу

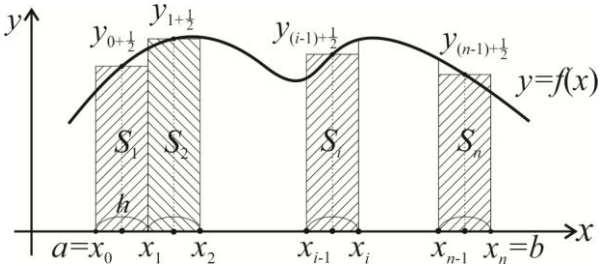
– $f\left(x_{(i-1)+\frac{1}{2}}\right) = f\left(x_{i-1}+\frac{h}{2}\right)$, (рис. 7.2, *в*).



а) метод лівих прямокутників



б) метод правих прямокутників



в) метод середніх прямокутників

Рисунок 7.2 – Графічна інтерпретація методу прямокутників

Відповідно отримаємо розрахункові формули:

- формула лівих прямокутників

$$I=\sum_{i=1}^{n}S_i=h\cdot y_0+h\cdot y_1+...+h\cdot y_{i-1}+...+h\cdot y_{n-1}=h\sum_{i=0}^{n-1}y_i=\frac{b-a}{n}\sum_{i=0}^{n-1}f(x_i); \qquad (7.3)$$

- формула правих прямокутників

$$I=\sum_{i=1}^{n}S_i=h\cdot y_1+h\cdot y_2+...+h\cdot y_i+...+h\cdot y_n=h\sum_{i=1}^{n}y_i=\frac{b-a}{n}\sum_{i=1}^{n}f(x_i); \qquad (7.4)$$

- формула середніх прямокутників

$$I=\sum_{i=1}^{n}S_i=h\cdot y_{0+\frac{1}{2}}+h\cdot y_{1+\frac{1}{2}}+...+h\cdot y_{(i-1)+\frac{1}{2}}+...+h\cdot y_{(n-1)+\frac{1}{2}}=h\sum_{i=0}^{n-1}y_{i+\frac{1}{2}}=$$

$$=\frac{b-a}{n}\sum_{i=0}^{n-1}f\left(x_i+\frac{h}{2}\right). \qquad (7.5)$$

Точність формул (7.3)-(7.4) можна оцінити так: $\varepsilon=\frac{b-a}{2}h\cdot f'(\xi)$, (7.6)

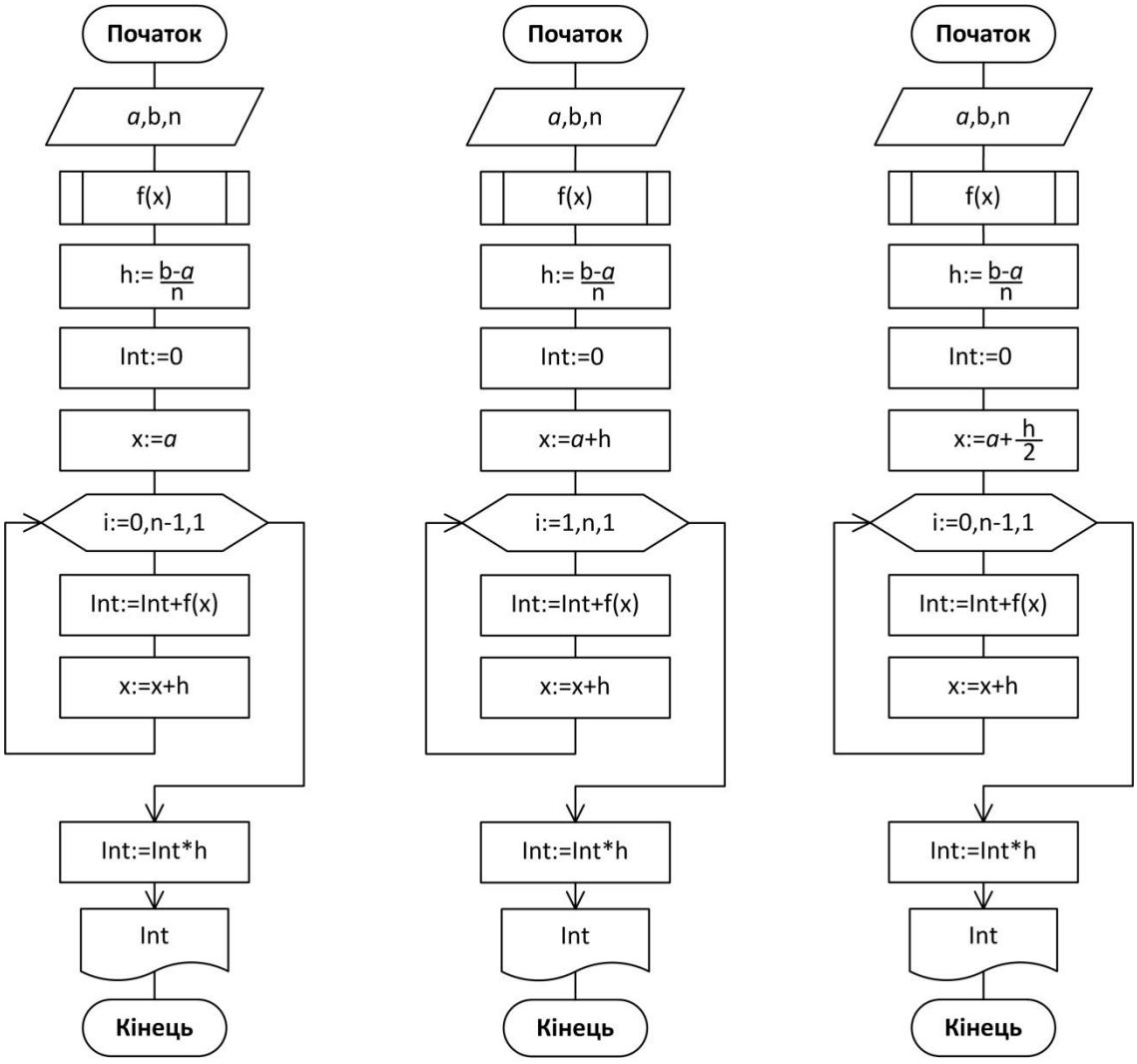
де $\xi\in[a,b]$ - точка, в якій перша похідна функції *f*(*x*) набуває найбільшого за модулем значення. Формула має перший порядок точності відносно кроку *h*.

Точність формули середніх прямокутників (7.5):

$$\varepsilon=\frac{b-a}{24}h^2 f''(\xi), \qquad (7.7)$$

де $\xi\in[a,b]$ - точка, в якій друга похідна функції *f*(*x*) набуває найбільшого за модулем значення. Формула має другий порядок точності відносно кроку *h*.

Блок-схеми методів прямокутників приведені на рис. 7.3.



формула лівих прямокутників *формула правих прямокутників* *формула середніх прямокутників*

Рисунок 7.3 – Блок-схема методів прямокутників

Метод трапецій. Апроксимуємо площі елементарних криволінійних трапецій S_i площами трапецій, висота яких дорівнюватиме h, а довжини основ значенням $f(x)$ у правому й лівому кінцях інтервалу $[x_{i-1}, x_i]$ - $f(x_{i-1}) = y_{i-1}$ й $f(x_i) = y_i$ (рис. 7.4). Загальну площу фігури обчислимо, як суму площ окремих елементарних трапецій:

$$I = \sum_{i=1}^{n} S_i = h \cdot \frac{y_0 + y_1}{2} + h \cdot \frac{y_1 + y_2}{2} + ... + h \cdot \frac{y_{i-1} + y_i}{2} + ... + h \cdot \frac{y_{n-1} + y_n}{2} =$$

$$= h \cdot \left(\frac{y_0}{2} + y_1 + ... + y_{n-1} + \frac{y_n}{2} \right) = h \cdot \left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i \right) =$$

$$= \frac{b-a}{n} \left(\frac{f(x_0) + f(x_n)}{2} + \sum_{i=1}^{n-1} f(x_i) \right). \qquad (7.8)$$

Точність формули трапецій можна оцінити так:

$$\varepsilon = \frac{b-a}{12} h^2 f''(\xi), \qquad (7.9)$$

де $\xi \in [a,b]$ - точка, в якій друга похідна функції $f(x)$ набуває найбільшого за модулем значення. Формула має другий порядок точності відносно кроку h, але її похибка вдвічі більше похибки формули середніх прямокутників.

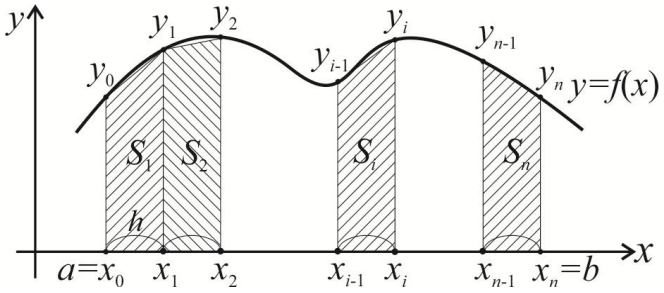


Рисунок 7.4 – Графічна інтерпретація методу трапецій

Метод Сімпсона. Апроксимуємо площі елементарних криволінійних трапецій S_i, площами фігур, обмежених зверху параболою, що проходить через три точки (x_{2i-2}, y_{2i-2}), (x_{2i-1}, y_{2i-1}), (x_{2i}, y_{2i}), $i = 0,1,...,n$ (рис. 7.5). Кількість точок для формули Сімпсона має бути завжди парною - $n = 2m$, так як парабола проводиться через три точки. Загальну площу фігури обчислимо, як суму площ окремих елементарних криволінійних трапецій:

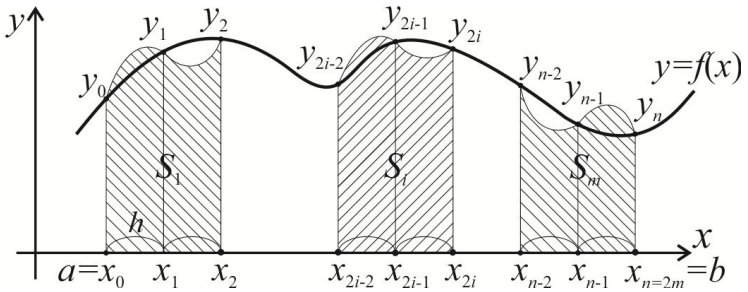


Рисунок 7.5 – Графічна інтерпретація методу Сімпсона

$$I = \sum_{i=1}^{m} S_i = \sum_{i=1}^{m} \frac{h}{3}\left(y_{2i-2} + 4y_{2i-1} + y_{2i}\right)_j =$$

$$= \frac{h}{3}\left(y_0 + 4y_1 + y_2\right) + \frac{h}{3}\left(y_2 + 4y_3 + y_4\right) + ... + \frac{h}{3}\left(y_{n-2} + 4y_{n-1} + y_n\right) =$$

$$= \frac{h}{3}\left((y_0 + y_n) + 4(y_1 + y_3 + ... + y_{n-1}) + 2(y_2 + y_4 + ... + y_{n-2})\right) =$$

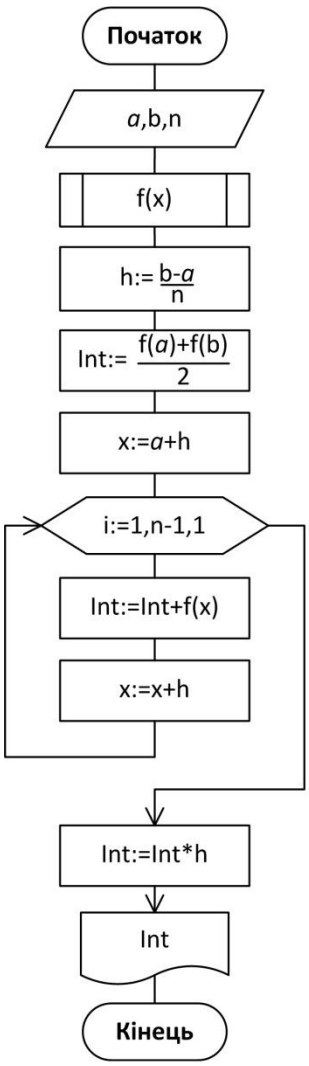
$$= \frac{b-a}{3n}\left[\left(f(x_0) + f(x_n)\right) + 4\left(f(x_1) + ... + f(x_{n-1})\right) + 2\left(f(x_2) + ... + f(x_{n-2})\right)\right] \quad (7.10)$$

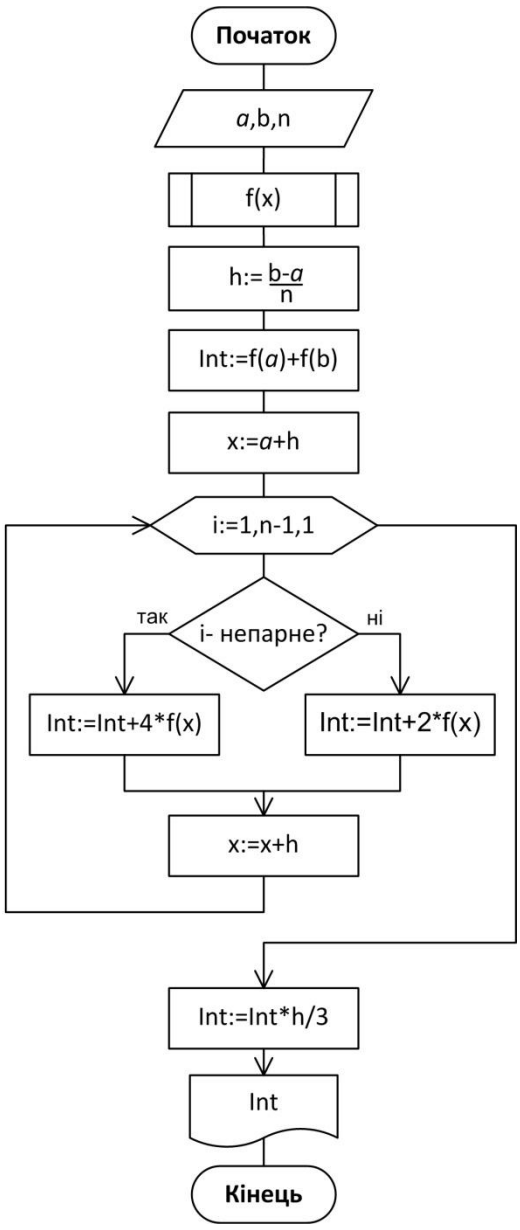
Точність формули Сімпсона можна оцінити так:

$$\varepsilon = \frac{(b-a)}{180} h^4 f^{IV}(\xi), \quad (7.11)$$

де $\xi \in [a,b]$ - точка, в якій четверта похідна функції $f(x)$ набуває найбільшого за модулем значення. Формула має четвертий порядок точності відносно кроку h.

Блок-схема методів трапецій і Сімпсона приведена на рис. 7.6.





формула трапецій *формула Сімпсона*

Рисунок 7.6 – Блок-схема методів трапецій і Сімпсона

Завдання

Методами прямокутників, трапецій та Сімпсона обчислити значення інтеграла. Значення *n* обрати рівним 10. Оцінити похибку для кожного методу інтегрування за формулами (7.6), (7.7), (7.9), (7.11).

1. $\int_{-1}^{1} \frac{x}{2x+5}dx$
2. $\int_{0}^{4} \frac{x}{(3x+4)^2}dx$
3. $\int_{-2}^{2} \frac{3x+4}{2x+7}dx$
4. $\int_{-1}^{1} \frac{1}{(2x+7)(px+4)}dx$
5. $\int_{-2}^{2} \frac{1}{x^2+4}dx$
6. $\int_{1}^{5} \sqrt{9+x^2}dx$

7. $\int_1^5 x^3\sqrt{4+x^2}\,dx$

8. $\int_{-2}^{2} \frac{1}{3x^2+4x+2}dx$

9. $\int_1^5 \frac{\sqrt{x}}{4+3x}dx$

10. $\int_0^2 \frac{x}{x^4+81}dx$

11. $\int_{\frac{\pi}{6}}^{\frac{5\pi}{4}} \frac{\cos x}{x}dx$

12. $\int_1^5 \frac{\sqrt{x}}{(1+2x)^2}dx$

13. $\int_{6.5}^{8.5} \sqrt{x^2-36}dx$

14. $\int_{9.5}^{7.5} x^3\sqrt{x^2-49}dx$

15. $\int_{-1}^{1} x\sqrt{2x+3}dx$

16. $\int_0^4 \frac{1}{\sqrt{(2x+7)(3x+4)}}dx$

17. $\int_{-1}^{1} \frac{x}{x^3+8}dx$

18. $\int_{-2}^{2} x\sqrt{49-x^2}\,dx$

19. $\int_{-2}^{2} \frac{x^2}{x^3-27}dx$

20. $\int_{-2}^{2} \frac{1}{x^3+64}dx$

Послідовність виконання лабораторної роботи

Методами прямокутників, трапецій та Сімпсона обчислити значення інтеграла. Значення *n* обрати рівним 10. Оцінити похибку для кожного методу інтегрування за формулами (7.6), (7.7), (7.9), (7.11).

$$\int_0^{1,3} \frac{1}{\sqrt{2x^2+0,3}}dx$$

Для обчислення інтеграла чисельними методами задамо сітку значень по x з кроком $h=\frac{b-a}{n}=\frac{1,3-0}{10}=0,13$, $x_0=a=0$; $x_{i+1}=x_i+h$; $i=\overline{0,n-1}$:

$$x_0=0;\ x_1=0,13;\ x_2=0,26;\ x_3=0,39;\ x_4=0,52;\ x_5=0,5;$$
$$x_6=0,78;\ x_7=0,91;\ x_8=1,04;\ x_9=1,17;\ x_{10}=1,3;$$

знайдемо значення підінтегральної функції в отриманих точках $y_i=f(x_i)=\frac{1}{\sqrt{2x_i^2+0,3}}$, $i=\overline{0,n}$:

$$y_0=1,826;\ y_1=1,731;\ y_2=1,516;\ y_3=1,287;\ y_4=1,091;\ y_5=0,935;$$
$$y_6=0,812;\ y_7=0,715;\ y_8=0,637;\ y_9=0,574;\ y_{10}=0,521\ .$$

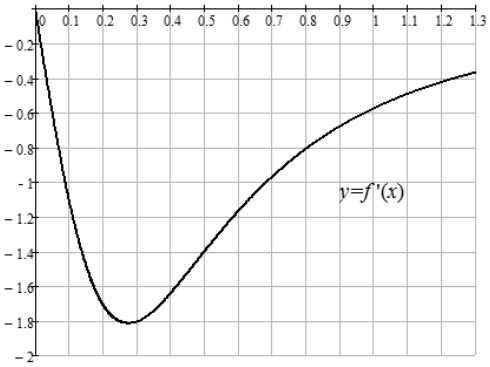
Знайдемо значення інтегралу за ***методом лівих прямокутників***:

$$I_{left}=h\sum_{i=0}^{n-1} y_i=0,13\cdot(1,826+1,731+1,516+1,287+1,091+0,935+0,812+0,715+0.637+0,574)=$$
$$=0,13\cdot 11,122=1,446;$$

та за ***методом правих прямокутників***:

$$I_{right} = h\sum_{i=1}^{n} y_i = 0{,}13\cdot(1{,}731+1{,}516+1{,}287+1{,}091+0{,}935+0{,}812+0{,}715+0{,}637+0{,}574+0{,}521) =$$
$$= 0{,}13\cdot 9{,}817 = 1{,}276\ .$$

Виконаємо оцінку точності розрахунків за формулою: $\varepsilon = \dfrac{b-a}{2} h\cdot f'(\xi)$, де $\xi \in [0;\ 1{,}3]$ - точка, в якій перша похідна функції $f(x)$ набуває найбільшого за модулем значення.



Знайдемо першу похідну $f'(x) = -\dfrac{2x}{\sqrt{(2x^2+0{,}3)^3}}$ і побудуємо її графік на інтервалі $[0;\ 1{,}3]$. Найбільше значення похідної $\max\limits_{[0;1.3]} |f'(x)| \approx 1{,}8$, приймемо його рівним $2{,}0$, тоді:

$$\varepsilon = \frac{1{,}3-0}{2}\cdot 0{,}13\cdot 2{,}0 = 0{,}169\ .$$

Порядок похибки співпадає з порядком h.

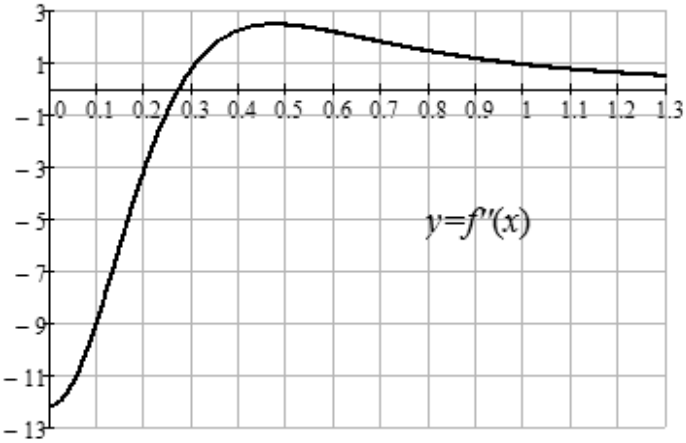
Знайдемо значення інтегралу за ***методом середніх прямокутників***. Для цього необхідно знайти значення функції в середніх точках $x_{i+\frac{1}{2}} = x_i + \dfrac{h}{2}$, $i = \overline{0, n-1}$:

$$x_{0+\frac{1}{2}} = 0{,}065;\ \ x_{1+\frac{1}{2}} = 0{,}195;\ \ x_{2+\frac{1}{2}} = 0{,}325;\ \ x_{3+\frac{1}{2}} = 0{,}455;\ \ x_{4+\frac{1}{2}} = 0{,}585;\ \ x_{5+\frac{1}{2}} = 0{,}715;$$
$$x_{6+\frac{1}{2}} = 0{,}845;\ \ x_{7+\frac{1}{2}} = 0{,}975;\ \ x_{8+\frac{1}{2}} = 1{,}105;\ \ x_{9+\frac{1}{2}} = 1{,}235;$$

знайдемо значення підінтегральної функції в отриманих точках :

$$y_{0+\frac{1}{2}} = 1{,}801;\ \ y_{1+\frac{1}{2}} = 1{,}631;\ \ y_{2+\frac{1}{2}} = 1{,}399;\ \ y_{3+\frac{1}{2}} = 1{,}183\ \ y_{4+\frac{1}{2}} = 1{,}008;\ \ y_{5+\frac{1}{2}} = 0{,}87;$$
$$y_{6+\frac{1}{2}} = 0{,}761;\ \ y_{7+\frac{1}{2}} = 0{,}674;\ \ y_{8+\frac{1}{2}} = 0{,}604;\ \ y_{9+\frac{1}{2}} = 0{,}546\ .$$

$$I_{mid} = h\sum_{i=0}^{n-1} y_{i+\frac{1}{2}} = 0{,}13\cdot(1{,}801+1{,}631+1{,}399+1{,}83+1{,}008+0{,}87+0{,}761+0{,}674+0{,}604+0{,}546) =$$
$$= 0{,}13\cdot 10{,}476 = 1{,}3618\ .$$



Виконаємо оцінку точності розрахунків за формулою:

$\varepsilon = \dfrac{b-a}{24} h^2 f''(\xi)$, де $\xi \in [0;\ 1{,}3]$ - точка, в якій друга похідна функції $f(x)$ набуває найбільшого за модулем значення. Знайдемо другу похідну:

$$f''(x) = \frac{12x^2}{\sqrt{(2x^2+0{,}3)^5}} - \frac{2}{\sqrt{(2x^2+0{,}3)^3}}$$

і побудуємо її графік на інтервалі $[0;\ 1{,}3]$.

Найбільше значення похідної $\max\limits_{[0;1,3]} |f''(x)| \approx 12$, приймемо його рівним 12, тоді:

$$\varepsilon = \frac{1,3-0}{24} \cdot 0,13^2 \cdot 12,0 = 0,0109 \ .$$

Порядок похибки співпадає з порядком h^2.

Знайдемо значення інтегралу за ***методом трапецій***:

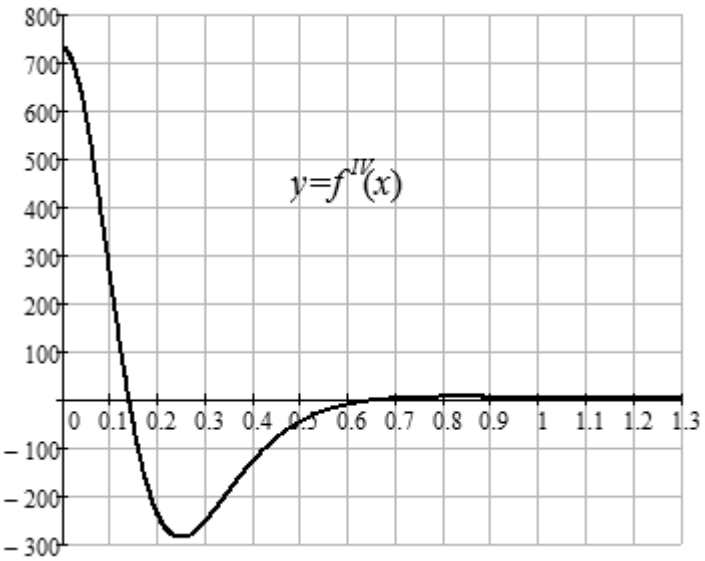
$$I_{trap} = h \cdot \left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i \right) = 0,13 \cdot \left(\frac{1,826 + 0,521}{2} + 1,731 + 1,516 + 1,287 + 1,091 + \right.$$
$$\left. + 0,935 + 0,812 + 0,715 + 0,637 + 0,574 \right) = 0,13 \cdot 10,47 = 1,3611 \ .$$

Точність формули трапецій має той же порядок точності по h, що і формула середніх прямокутників, але її значення вдвічі більше: $\varepsilon = \frac{b-a}{12} h^2 f''(\xi)$, де $\xi \in [0;\ 1,3]$ - точка, в якій друга похідна функції $f(x)$ набуває найбільшого за модулем значення. Тому отримаємо:

$$\varepsilon = \frac{1,3-0}{12} \cdot 0,13^2 \cdot 12,0 = 0,0218 .$$

Значення інтегралу за ***методом Сімпсона*** обчислимо за формулою:

$$I_{Simps} = \frac{h}{3} \left((y_0 + y_n) + 4 \cdot (y_1 + y_3 + \ldots + y_{n-1}) + 2 \cdot (y_2 + y_4 + \ldots + y_{n-2}) \right) =$$
$$= \frac{0,13}{3} \cdot \left(1,826 + 0,521 + 4 \cdot (1,731 + 1,287 + 0,935 + 0,715 + 0,574) + \right.$$
$$\left. + 2 \cdot (1,516 + 1,091 + 0,812 + 0,637) \right) = \frac{0,13}{3} \cdot 31,421 = 1,3616 \ .$$



Оцінимо точність формули Сімпсона $\varepsilon = \frac{b-a}{180} h^4 f^{IV}(\xi)$, тут $\xi \in [0;\ 1,3]$ - точка, в якій четверта похідна функції $f(x)$ набуває найбільшого за модулем значення. Знайдемо четверту похідну:

$$f^{IV}(x) = \frac{1680x^4}{\sqrt{(2x^2+0,3)^9}} - \frac{720x^2}{\sqrt{(2x^2+0,3)^7}} + \frac{36}{\sqrt{(2x^2+0,3)^5}}$$

і побудуємо її графік на інтервалі $[0;\ 1,3]$. Найбільше значення похідної $\max\limits_{[0;1,3]} |f^{IV}(x)| \approx 730$, тоді:

$$\varepsilon = \frac{1,3-0}{180} 0,13^4 \cdot 730 = 0,0015 \ .$$

Порядок похибки співпадає з порядком h^4.

Наведемо фрагменти програм, що реалізують розглянуті алгоритми.

Метод лівих прямокутників. Вхідними даними будуть границі інтегрування, кількість інтервалів *n*, підінтегральний вираз *f*(*x*) запрограмований функцією.

```
program Left;
  var   n,i:integer;
        a,b,h,x,Int:real;
  function f(x:real):real;
     begin
          f:=1/sqrt(2*sqr(x)+0.3);
     end;
begin
      writeln('Чисельне інтегрування. Метод лівих прямокутників.');
      write('Введіть нижню границю a='); read(a);
      write('Введіть верхню границю b='); read(b);
      write('Введіть кількість інтервалів  n='); read(n);
      h:=(b-a)/n;
      writeln('Крок h=',h:8:5);
      Int:=0;
      x:=a;
      for i:=0 to n-1 do
         begin
              Int:=Int+f(x);
              x:=x+h;
         end;
      Int:=Int*h;
      writeln('Значення інтегралу: I=',Int:8:4);
end.
```

Результати роботи програми приведені на рис. 7.7.

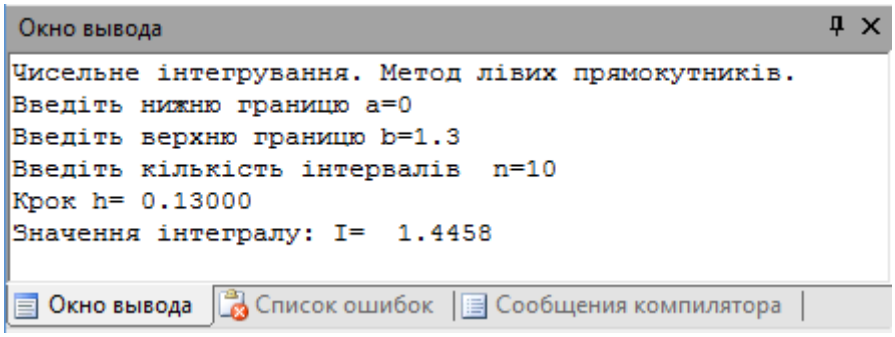


Рисунок 7.7 – Інтегрування методом лівих прямокутників

Метод правих прямокутників.

```
program Right;
  var   n,i:integer;
        a,b,h,x,Int:real;
  function f(x:real):real;
      begin
           f:=1/sqrt(2*sqr(x)+0.3);
      end;
```

```
begin
        writeln('Чисельне інтегрування. Метод правих прямокутників.');
        write('Введіть нижню границю a='); read(a);
        write('Введіть верхню границю b='); read(b);
        write('Введіть кількість інтервалів  n='); read(n);
        h:=(b-a)/n;
        writeln('Крок h=',h:8:5);
        Int:=0;
        x:=a+h;
        for i:=1 to n do
                        begin
                                Int:=Int+f(x);
                                x:=x+h;
                        end;
        Int:=Int*h;
        writeln('Значення інтегралу: I=',Int:8:4);
end.
```

Результати роботи програми приведені на рис. 7.8.

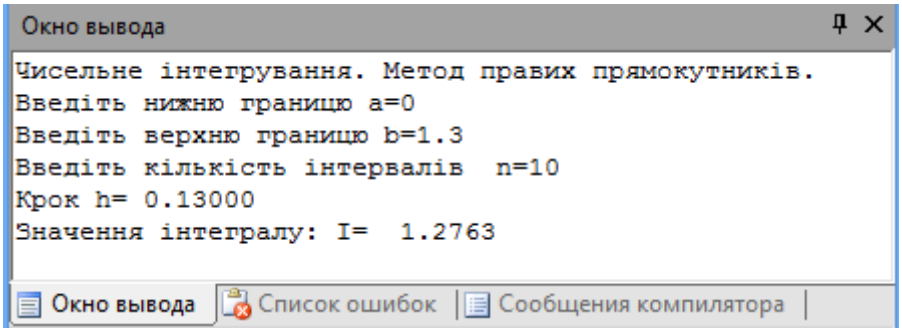


Рисунок 7.8 – Інтегрування методом правих прямокутників

Метод середніх прямокутників.

```
program Middle;
  var   n,i:integer;
        a,b,h,x,Int:real;
  function f(x:real):real;
          begin
                f:=1/sqrt(2*sqr(x)+0.3);
          end;
begin
       writeln('Чисельне інтегрування. Метод середніх прямокутників.');
       write('Введіть нижню границю a='); read(a);
       write('Введіть верхню границю b='); read(b);
       write('Введіть кількість інтервалів  n='); read(n);
       h:=(b-a)/n;
       writeln('Крок h=',h:8:5);
       Int:=0;
       x:=a+h/2;
       for i:=0 to n-1 do
```

```
                    begin
                        Int:=Int+f(x);
                        x:=x+h;
                    end;
    Int:=Int*h;
    writeln('Значення інтегралу: I=',Int:8:4);
end.
```

Результати роботи програми приведені на рис. 7.9.

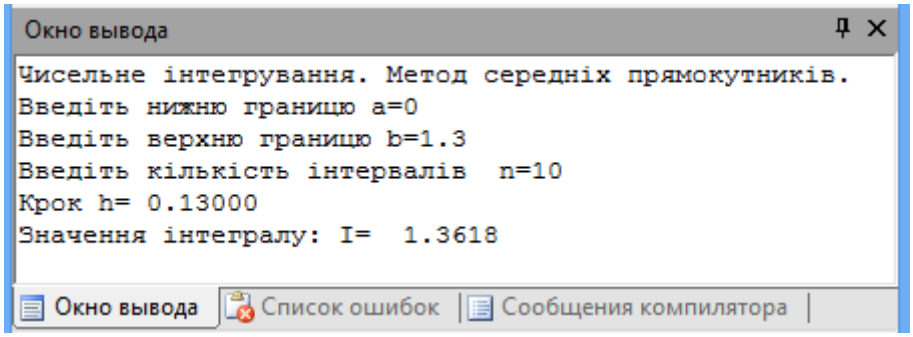


Рисунок 7.9 – Інтегрування методом середніх прямокутників

Метод трапецій.

```
program Trapec;
  var   n,i:integer;
        a,b,h,x,Int:real;
  function f(x:real):real;
        begin
            f:=1/sqrt(2*sqr(x)+0.3);
        end;
begin
        writeln('Чисельне інтегрування. Метод трапецій.');
         write('Введіть нижню границю a='); read(a);
         write('Введіть верхню границю b='); read(b);
         write('Введіть кількість інтервалів  n='); read(n);
         h:=(b-a)/n;
         writeln('Крок h=',h:8:5);
         Int:=(f(a)+f(b))/2.0;
         x:=a+h;
         for i:=1 to n-1 do
                        begin
                            Int:=Int+f(x);
                            x:=x+h;
                        end;
         Int:=Int*h;
         writeln('Значення інтегралу: I=',Int:8:4);
end.
```

Результати роботи програми приведені на рис. 7.10.

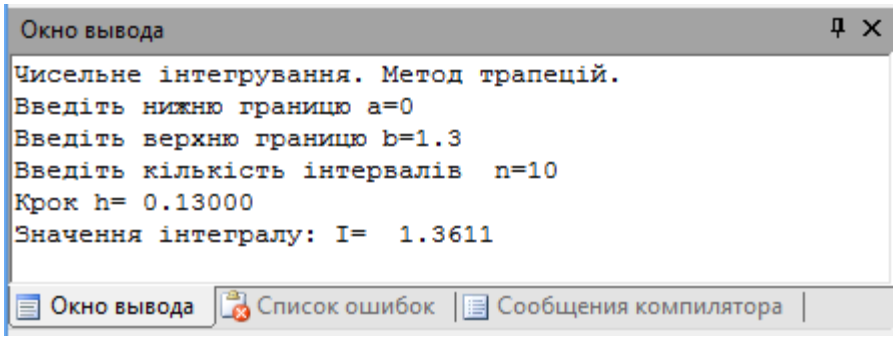


Рисунок 7.10 – Інтегрування методом трапецій

Метод Сімпсона.

```
program Middle;
  var   n,i:integer;
        a,b,h,x,Int:real;
  function f(x:real):real;
              begin
                    f:=1/sqrt(2*sqr(x)+0.3);
              end;
begin
      writeln('Чисельне інтегрування. Метод Сімпсона.');
      write('Введіть нижню границю a='); read(a);
      write('Введіть верхню границю b='); read(b);
      write('Введіть кількість інтервалів  n='); read(n);
      h:=(b-a)/n;
      writeln('Крок h=',h:8:5);
      Int:=f(a)+f(b);
      x:=a+h;
      for i:=1 to n-1 do
                        begin
                              if Odd(i) then Int:=Int+4*f(x)
                                        else Int:=Int+2*f(x);
                              x:=x+h;
                        end;
      Int:=Int*h/3.0;
      writeln('Значення інтегралу: I=',Int:8:4);
end.
```

Результати роботи програми приведені на рис. 7.11.

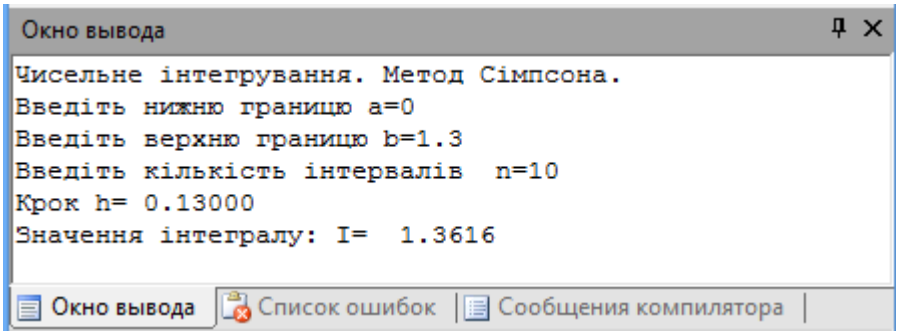


Рисунок 7.11 – Інтегрування методом Сімпсона

Контрольні запитання

1. Коли застосовується чисельне інтегрування?
2. Який загальний геометричний зміст чисельних методів обчислення визначених однократних інтегралів?
3. Зміст методів прямокутників, їхня похибка.
4. Зміст методу трапецій, його похибка.
5. Зміст методу Сімпсона, його похибка.
6. Як обчислити інтеграл із заданою точністю?

Перелік рекомендованої літератури

1. Фельдман Л.П. Чисельні методи в інформатиці / Л.П. Фельдман, А.І. Петренко, О.А. Дмитрієва. –К.: BHV, 2006. – 480 с.
2. Пирумов У. Г. Численные методы: учеб. пособие для студ. втузов / У.Г. Пирумов. – 4-е изд., стереотип. – М.: Дрофа, 2007. – 221, [3] с.: ил.
3. Численные методы. Сборник задач: учеб. пособие для вузов / В.Ю. Гидаспов, И.Э. Иванов, Д.Л. Ревизников и др.; под ред. У.Г. Пирумова. – М.: Дрофа, 2007. – 144 с.: ил.
4. Демидович Б.П. Основы вычислительной математики / Б.П. Демидович, И.А. Марон. - М.: Государственное изд-во физико-математ. лит., 1960. – 659 с.

ЛАБОРАТОРНА РОБОТА № 8
ЧИСЕЛЬНЕ РОЗВ'ЯЗУВАННЯ ЗАДАЧІ КОШІ

Мета: опанувати чисельними методами розв'язання задачі Коші для звичайного диференційного рівняння першого порядку.

Теоретичні відомості

Звичайним диференціальним рівнянням називають рівняння виду

$$F(x, y, y', y'',..., y^{(n)})=0, \qquad (8.1)$$

що зв'язує одну незалежну змінну x, шукану функцію $y(x)$ та її похідні до n-ого порядку. *Порядком* звичайного диференційного рівняння називається порядок старшої похідної від шуканої функції.

Диференціальне рівняння називається *лінійним*, якщо воно має вигляд:

$$a_n(x)y^{(n)} + a_{n-1}(x)y^{(n-1)} + ...+a_1(x)y' + a_0(x)y + f(x) = 0.$$

Загальним інтегралом рівняння (8.1) називають функцію

$$\Phi(x, y, c_1,..., c_n), \qquad (8.2)$$

що зв'язує незалежну змінну x, шукану функцію $y(x)$ та n сталих інтегрування $c_1,...,c_n$ за допомогою рівняння $\Phi(x, y, c_1,..., c_n) = 0$. Таким чином, функція $y(x)$ входить до функції (8.2) неявно, а кількість сталих інтегрування дорівнює порядку рівняння.

Загальним розв'язком звичайного диференціального рівняння називається функція

$$y(x) = \varphi(x, c_1,..., c_n), \qquad (8.3)$$

що зв'язує незалежну змінну та n сталих інтегрування, тобто рівняння (8.3) визначає функцію $y(x)$ явно.

Для визначення сталих інтегрування задаються *додаткові у мови*. Їх кількість дорівнює кількості сталих інтегрування. Якщо в додаткові умови підставити функцію (8.2) і розв'язати отриману систему відносно сталих інтегрування $c_1,...,c_n$, а потім отриманий розв'язок підставити в (8.2),то отримаємо частковий інтеграл $\Phi_1(x, y(x)) = 0$.

Подібні дії із загальним розв'язком (8.3) дають частковий розв'язок $y(x) = \varphi_1(x)$.

Якщо всі додаткові умови задаються в одній точці x_0, то сукупність звичайного диференціального рівняння і додаткових умов називають *задачею Коші*, а додаткові умови називають *початковими умовами*.

Розглянемо задачу Коші для звичайного диференційного рівняння першого порядку

$$y' = f(x, y), \quad y_0 = y(x_0). \qquad (8.4)$$

Потрібно знайти розв'язок $y=y(x)$ на відрізку $[a,b]$, де $x_0 = a$.

Уведемо різницеву сітку на відрізку $[a,b]$: $x_i = x_0 + h \cdot i$, $h = \dfrac{b-a}{n}$, $i = 0,1,...,n$. Точки x_i називаються *вузлами* різницевої сітки, відстань h між вуз-

лами - *кроком* різницевої сітки, а сукупність заданих у вузлах сітки значень деякої величини називається сітковою функцією $y^{(h)} = \{y_i, i = 0,1,...,n\}$.

Наближений розв'язок задачі Коші будемо шукати чисельно у вигляді сіткової функції $y^{(h)}$.

Метод Ейлера відіграє важливу роль в теорії чисельних методів розв'язання звичайних диференціальних рівнянь, але не часто використовується в практичних розрахунках через низьку точність.

Розглянемо геометричну інтерпретацію метода (рис. 8.1). Нехай розв'язок у вузлі x_0 відомий з початкових умов (8.4). Отримаємо розв'язок у вузлі x_1.

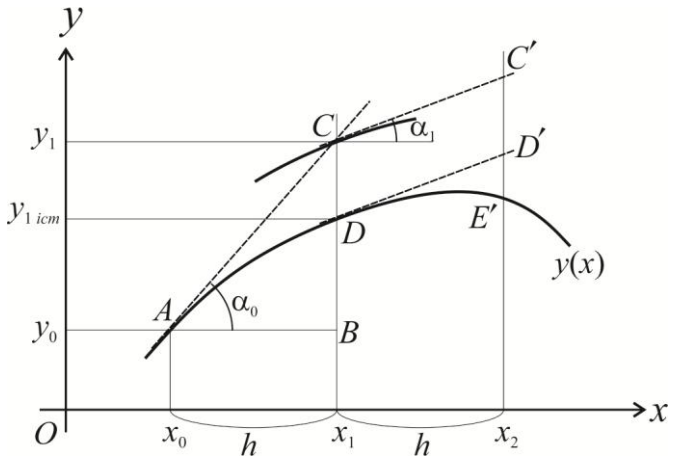


Рисунок 8.1 – Геометрична інтерпретація методу Ейлера

Графік функції $y^{(h)}$, що є розв'язком задачі Коші (8.4), являє собою гладку криву, проходить через точку (x_0, y_0) згідно умови $y(x_0)=y_0$ і має в цій точці дотичну. Тангенс кута нахилу дотичної до осі *Ox* дорівнює значенню похідної від розв'язку в точці x_0, тобто значенню правої частини диференціального рівняння в точці (x_0,) згідно виразу $\operatorname{tg}\alpha_0 = y'(x_0) = f(x_0, y_0)$. Якщо крок *h* різницевої сітки невеликий і графік дотичної не встигає суттєво розбігтися з графіком функції, то можна прийняти за значення функції у вузлі x_1 – $y(x_1) = y_{1\,icm}$ – значення дотичної y_1. Похибка $|y_1 - y_{1\,icm}|$ геометрично буде представлена відрізком *CD*. З прямокутного трикутника *ABC* знайдемо $CB = BA \cdot \operatorname{tg}\alpha_0$ або $\Delta y = h \cdot y'(x_0)$. Враховуючи, що $\Delta y = y_1 - y_0$, і замінюючи похідну $y'(x_0)$ на праву частину диференціального рівняння, отримаємо співвідношення

$$y_1 = y_0 + h \cdot f(x_0, y_0).$$

Вважаємо тепер точку (x_1, y_1) початковою. Повторимо всі попередні дії. Отримаємо значення y_2 у вузлі x_2. Перехід до довільних індексів дає формулу метода Ейлера:

$$y_{i+1} = y_i + h \cdot f(x_i, y_i). \qquad (8.5)$$

На кожному кроці метода Ейлера виникає *локальна похибка* по відношенню до точного розв'язку – *CD* (на першому), *C'D'* (на другому), і так далі. Крім того, на кожному кроці, починаючи з другого, накопичується *глобальна похибка* $|y_i - y_{i\,icm}|$ – *C'E'*. Локальна похибка на кожному кроці визначається співвід-

ношенням $\varepsilon_i^h = \frac{y''(\xi)}{2}h^2$, де $\xi \in [x_{i-1}, x_i]$. Глобальна похибка $\varepsilon_{гл}^h = Ch$ - має лінійну залежність від h, тому метод має перший порядок точності відносно кроку h.

Блок-схема алгоритму Ейлера приведена на рис. 8.4.

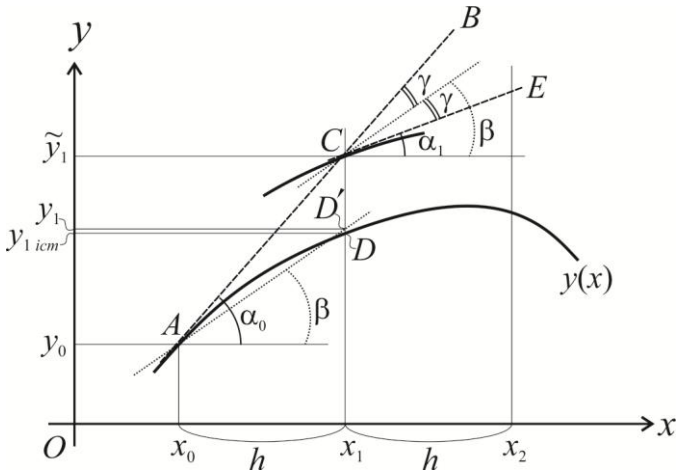


Рисунок 8.2 – Геометрична інтерпретація методу "предиктор-коректор"

Метод "предиктор-коректор" (Ейлера-Коші). Розрахунок проводиться в два етапи (рис. 8.2). На першому (етап *прогнозу*) шукається розв'язок на правому кінці інтервалу за методом Ейлера (за дотичною в точці A під кутом α_0: $\operatorname{tg}\alpha_0 = f(x_0, y_0)$), на другому (етап *корекції*) уточнюється значення розв'язку на правому кінці за допомогою полусуми тангенсів кутів нахилу на кінцях інтервалу $\left(\angle\beta = \frac{\angle\alpha_0 + \angle\alpha_1}{2}\right)$. Остаточний напрямок - бісектриса $\angle BCE$ (пряма під кутом $\angle\beta$):

$$\tilde{y}_{i+1} = y_i + h \cdot f(x_i, y_i),$$

$$y_{i+1} = y_i + h \cdot \frac{f(x_i, y_i) + f(x_{i+1}, \tilde{y}_{i+1})}{2}, \qquad (8.6)$$

$$x_{i+1} = x_i + h.$$

Локальна похибка по відношенню до точного розв'язку на першому кроці – DD'. Похибка методу "предиктор-коректор" має порядок h^2. Блок-схема алгоритму метода приведена на рис. 8.5.

Метод Рунге-Кутти. В обчислювальній практиці найбільшого розповсюдження отримала схема Рунге-Кутти четвертого порядку точності – h^4. Її використовують у більшості стандартних програм на комп'ютерах. Приведемо розрахункові формули схеми:

$$y_{i+1} = y_i + \Delta y_i, \quad \Delta y_i = \frac{h}{6}\left(K_1^i + 2K_2^i + 2K_3^i + K_4^i\right),$$

$$K_1^i = f(x_i, y_i),$$

$$K_2^i = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1^i\right),$$

$$K_3^i = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_2^i\right),$$

$$K_4^i = f\left(x_i + h, y_i + h \cdot K_3^i\right). \qquad (8.7)$$

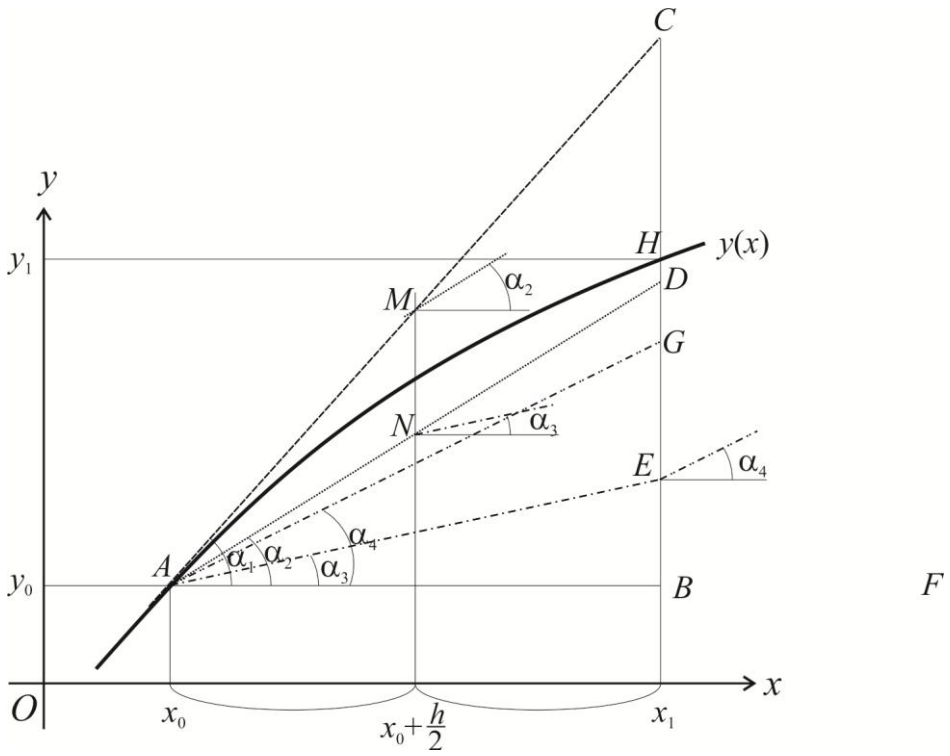


Рисунок 8.3 – Геометрична інтерпретація методу Рунге-Кутти

Дамо геометричне тлумачення методу (рис. 8.3). Значення величини Δy_i є середнєзваженою величиною поправок $h \cdot K_1^i$, $h \cdot K_2^i$, $h \cdot K_3^i$, $h \cdot K_4^i$ на кожному кроці (з ваговими коефіцієнтами $1/6$, $2/6$, $2/6$, $1/6$ відповідно). Проаналізуємо, як отримуються ці поправки. Спочатку створюється приріст $h \cdot K_1^i = h \cdot f(x_i, y_i)$, який відповідає поправці Ейлера. На рис. 8.3 йому відповідає відрізок *BC*. Точка *M* належить дотичній, що побудована в точці *A* до кривої $y(x)$ під кутом α_1, і має координати $\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1^i\right)$. Тому значення K_2^i буде дорівнювати значенню тангенса кута α_2, тобто тангенсу кута нахилу дотичної в точці *M* до кривої $y(x)$. Приросту $h \cdot K_2^i$ буде відповідати відрізок *BD*. Точка *N* належить прямій, що проходить через точку *A* під кутом α_2, і має координати $\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_2^i\right)$. Тому значення K_3^i буде дорівнювати значенню тангенса кута α_3, тобто тангенсу кута нахилу дотичної в точці *N* до кривої $y(x)$. Поправці $h \cdot K_3^i$ буде відповідати відрізок *BE*. Значення K_4^i буде дорівнювати значенню тангенса кута α_4, тобто тангенсу кута нахилу дотичної в точці *E* до кривої $y(x)$, тоді поправці с буде відповідати відрізок *BG*. Кінцева крокова поправка

$\Delta y_i = BH$ є результатом усереднення з указаними коефіцієнтами чотирьох поправок - довжин відрізків *BC*, *BD*, *BE* і *BG*. Точка *H* буде початковою для наступного кроку метода.

Блок-схема алгоритму Рунге-Кутти приведена на рис. 8.6.

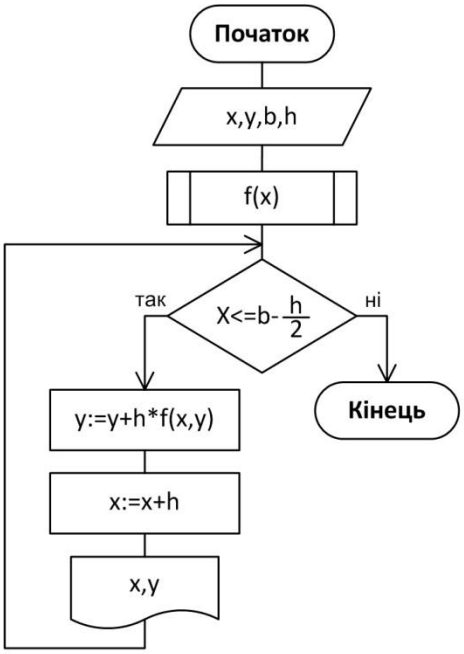


Рисунок 8.4 – Блок-схема методу Ейлера

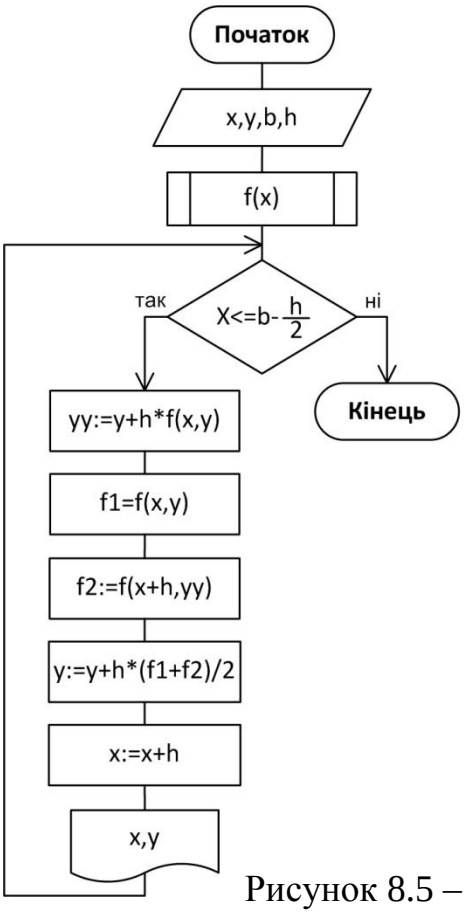


Рисунок 8.5 – Блок-схема методу "предиктор-коректор"

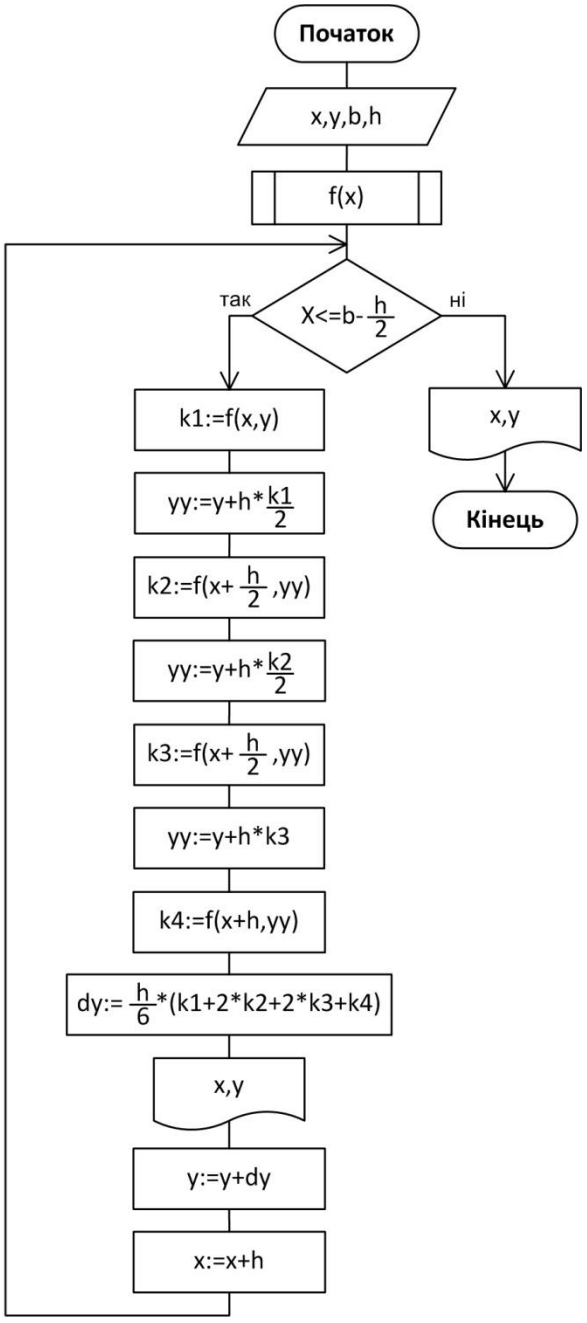


Рисунок 8.6 – Блок-схема методу Рунге-Кутти

Завдання

Методами Ейлера, “предиктор – коректор”, Рунге-Кутти знайти чисельний розв’язок задачі Коші для звичайного диференціального рівняння першого порядку на інтервалі $[x_0, b]$ з кроком h.

	$y' = f(x, y)$	x_0	$y(x_0) = y_0$	b	h
1.	$y' = \frac{1}{2}xy$	0,0	1,0	1,0	0,1
2.	$y' = xy^2 + 1$	0,0	0,0	1,0	0,1
3.	$y' = (2 - y)x$	1,0	1,0	2,0	0,1
4.	$y' = \ln(x + y)$	0,0	1,5	1,0	0,1
5.	$y' = \cos(x) + y$	-1,0	0,0	0,0	0,1
6.	$y' = 2y + 3e^x$	0,3	1,42	0,6	0,03
7.	$y' = y^2 - x^2$	1,0	1,0	2,0	0,1
8.	$y' = xy(y^2 - 1)$	0,0	0,5	1,0	0,1
9.	$y' = \frac{2y - x}{y}$	1,0	2,0	2,0	0,1
10.	$y' = x^3 - y$	1,0	-1,0	2,0	0,1
11.	$y' = 2xy + x^2$	0,0	0,0	0,5	0,05
12.	$y' = e^x - y^2$	0,0	0,0	0,4	0,04
13.	$y' = x^3 + y^2$	0,0	0,0	0,5	0,05
14.	$y' = y^2 e^x - 2y$	0,0	1,0	1,0	0,1
15.	$y' = \frac{y}{x}$	1,0	1,0	4,0	0,3
16.	$y' = x^2 - y^2$	0,0	0,0	1,0	0,1
17.	$y' = \frac{y}{x+1} - y^2$	0,0	1,0	1,0	0,1
18.	$y' = \sqrt{2x + y}$	2,0	0,75	3,0	0,1
19.	$y' = x + e^y$	1,0	1,0	2,0	0,1
20.	$y' = 2^x + y^2$	0,0	1,0	3,0	0,3

Послідовність виконання лабораторної роботи

Методами Ейлера, “предиктор – коректор”, Рунге-Кутти знайти чисельний розв’язок задачі Коші для звичайного диференціального рівняння першого порядку $y' = (y + x)^2$, $y(0) = 0$ на інтервалі $[0,\ 0{,}5]$ з кроком h=0,1 .

Метод Ейлера. Задамо сітку значень по x з кроком h=0,1, $x_0 = 0$; $x_{i+1} = x_i + h$:

$$x_0 = 0;\ \ x_1 = 0{,}1;\ \ x_2 = 0{,}2;\ \ x_3 = 0{,}3;\ \ x_4 = 0{,}4;\ \ x_5 = 0{,}5.$$

Виходячи з початкової умови $y_0 = 0$, знайдемо значення y_1 у вузлі $x_1 = 0{,}1$ за формулою Ейлера (8.5):

$$y_1 = y_0 + h \cdot f(x_0, y_0) = 0 + 0{,}1 \cdot (0+0)^2 = 0.$$

Аналогічно отримаємо розв'язок у наступному вузлі $x_2 = 0{,}2$:

$$y_2 = y_1 + h \cdot f(x_1, y_1) = 0 + 0{,}1 \cdot (0+0{,}1)^2 = 0{,}001.$$

Продовжимо обчислення для решти вузлів:

$$y_3 = y_2 + h \cdot f(x_2, y_2) = 0{,}001 + 0{,}1 \cdot (0{,}001 + 0{,}2)^2 = 0{,}0050401;$$

$$y_4 = y_3 + h \cdot f(x_3, y_3) = 0{,}0050401 + 0{,}1 \cdot (0{,}0050401 + 0{,}3)^2 = 0{,}0143450;$$

$$y_5 = y_4 + h \cdot f(x_4, y_4) = 0{,}0143450 + 0{,}1 \cdot (0{,}0143450 + 0{,}4)^2 = 0{,}0315132.$$

Остаточним розв'язком задачі Коші буде таблична функція:

i	0	1	2	3	4	5
x_i	0	0,1	0,2	0,3	0,4	0,5
y_i	0	0	0,001	0,0050401	0,0143450	0,0315132

Метод "предиктор-коректор". Виходячи з початкової умови $x_0 = 0$, $y_0 = 0$, знайдемо значення y_1 у вузлі $x_1 = 0{,}1$ за формулою Ейлера-Коші (8.6):

$$\tilde{y}_1 = y_0 + h \cdot f(x_0, y_0) = 0 + 0{,}1 \cdot (0+0)^2 = 0;$$

$$y_1 = y_0 + h \cdot \frac{f(x_0, y_0) + f(x_1, \tilde{y}_1)}{2};$$

$$f(x_0, y_0) = (0+0)^2 = 0; \qquad f(x_1, \tilde{y}_1) = (0+0{,}1)^2 = 0{,}01;$$

$$y_1 = 0 + 0{,}1 \cdot \frac{0 + 0{,}01}{2} = 0{,}0005.$$

Продовжимо обчислення для решти вузлів:

$$\tilde{y}_2 = y_1 + h \cdot f(x_1, y_1) = 0{,}0005 + 0{,}1 \cdot (0{,}0005 + 0{,}1)^2 = 0{,}001510025;$$

$$y_2 = y_1 + h \cdot \frac{f(x_1, y_1) + f(x_2, \tilde{y}_2)}{2};$$

$$f(x_1, y_1) = (0{,}0005 + 0{,}1)^2 = 0{,}01010025;$$

$$f(x_2, \tilde{y}_2) = (0{,}001510025 + 0{,}2)^2 = 0{,}0406063;$$

$$y_2 = 0{,}0005 + 0{,}1 \cdot \frac{0{,}01010025 + 0{,}0406063}{2} = 0{,}0030353.$$

Результати занесемо в таблицю:

i	x_i	y_i	$\tilde{y}_i$	$f(x_i, y_i)$	$f(x_{i+1}, \tilde{y}_{i+1})$
0	0	0	–	0	–
1	0,1	0,0005	0	0,0101003	0,01
2	0,2	0,0030353	0,001510025	0,0412233	0,0406063
3	0,3	0,0098138	0,0071576	0,0959846	0,0943458
4	0,4	0,0234083	0,0194123	0,1792746	0,1759067
5	0,5	0,0470243	0,0413358	–	0,2930445

Остаточним розв'язком задачі Коші буде таблична функція:

i	0	1	2	3	4	5
x_i	0	0,1	0,2	0,3	0,4	0,5
y_i	0	0,0005	0,0030353	0,0098138	0,0234083	0,0470243

Метод Рунге-Кутти. Спочатку знайдемо коефіцієнти схеми Рунге-Кутти за формулою (8.7):

$$K_1^0 = f(x_0, y_0) = f(0,0) = (0+0)^2 = 0;$$

$$K_2^0 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_1^0\right) = f\left(0 + \frac{0,1}{2}, 0 + \frac{0,1}{2}\cdot 0\right) = f(0,05\ ,0) = (0+0,05)^2 = 0,0025;$$

$$K_3^0 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_2^0\right) = f\left(0 + \frac{0,1}{2},\ 0 + \frac{0,1}{2}\cdot 0,0025\right) = f(0,05\ ,\ 0,000125) =$$

$$= (0,000125 + 0,05)^2 = 0,0025125;$$

$$K_4^0 = f\left(x_0 + h, y_0 + h\cdot K_3^0\right) = f\left(0+0,1\ ,\ 0+0,1\cdot 0,0025125\right) = f(0,1\ ,\ 0,00025125) =$$

$$= (0,00025125 + 0,1)^2 = 0,0100503.$$

Тепер знайдемо Δy_0:

$$\Delta y_0 = \frac{h}{6}\left(K_1^0 + 2K_2^0 + 2K_3^0 + K_4^0\right) = \frac{0,1}{6}\cdot(0 + 2\cdot 0,0025 + 2\cdot 0,0025125 + 0,0100503) =$$

$$= 0,0003346;$$

і остаточно $y_1 = y_0 + \Delta y_0 = 0 + 0,000335 = 0,0003346$.

Продовжимо обчислення для решти вузлів.

$$K_1^1 = f(x_1, y_1) = f(0,1\ ,0,0003346) = (0,0003346 + 0,1)^2 = 0,0100670;$$

$$K_2^1 = f\left(x_1 + \frac{h}{2}, y_1 + \frac{h}{2}K_1^1\right) = f\left(0,1 + \frac{0,1}{2}\ ,\ 0,0003346 + \frac{0,1}{2}\cdot 0,0100670\right) = f(0,15\ ,\ 0,0008379) =$$

$$= (0,0008379 + 0,15)^2 = 0,0227521;$$

$$K_3^1 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{h}{2}K_2^1\right) = f\left(0,1 + \frac{0,1}{2},\ 0,0003346 + \frac{0,1}{2}\cdot 0,0227521\right) = f(0,15,\ 0,0014722) =$$

$$= (0,0014722 + 0,15)^2 = 0,0229438;$$

$$K_4^1 = f\left(x_1 + h, y_1 + h\cdot K_3^1\right) = f\left(0,1+0,1\ ,\ 0,0003346 + 0,1\cdot 0,0229438\right) = f(0,2\ ,\ 0,0026290) =$$

$$= (0,0026290 + 0,2)^2 = 0,0410585;$$

$$\Delta y_1 = \frac{h}{6}\left(K_1^1 + 2K_2^1 + 2K_3^1 + K_4^1\right) = \frac{0,1}{6}\cdot(0,0100670 + 2\cdot 0,0227521 + 2\cdot 0,0229438 + 0,0410585) =$$

$$= 0,0023753;$$

$y_2 = y_1 + \Delta y_1 = 0,0003346 + 0,0023753 = 0,0027099$.

Результати занесемо в таблицю:

i	x_i	y_i	K_1^i	K_2^i	K_3^i	K_4^i	Δy_i
0	0	0	0	0,0025000	0,0025125	0,0100503	0,0003346
1	0,1	0,0003346	0,0100670	0,0227521	0,0229438	0,0410585	0,0023753
2	0,2	0,0027099	0,0410913	0,0649049	0,0655130	0,0956425	0,0066262
3	0,3	0,0093360	0,0956888	0,1325837	0,1339305	0,1786999	0,0134570
4	0,4	0,0227930	0,1787539	0,2320645	0,2346397	0,2983967	0,0235093
5	0,5	0,0463023	–	–	–	–	–

Остаточним розв'язком задачі Коші буде таблична функція:

i	0	1	2	3	4	5
x_i	0	0,1	0,2	0,3	0,4	0,5
y_i	0	0,0003346	0,0027099	0,0093360	0,0227930	0,0463023

Наведемо фрагменти програм, що реалізують розглянуті алгоритми.

Метод Ейлера. Вхідними даними будуть початкова умова – x_0, y_0, кінцева точка обчислень – b, крок – h, права частина рівняння $f(x)$ запрограмований функцією.

```
program Eyler;
var   b,h,x,y:real;
   function f(x,y:real):real;
      begin
            f:=sqr(y+x);
      end;
begin
      writeln('Задача Коші. Метод Ейлера');
      writeln('Введіть початкову умову:');
      write('x0='); read(x);
      write('y0='); read(y);
      write('Введіть кінцеву точку обчислень: b='); read(b);
      write('Введіть крок: h='); read(h);
      while x<=b-h/2 do
            begin
                  y:=y+h*f(x,y);
                  x:=x+h;
                  writeln('x=',x:5:2,'  y=',y:9:7);
            end;
end.
```

Результати роботи програми приведені на рис. 8.7.

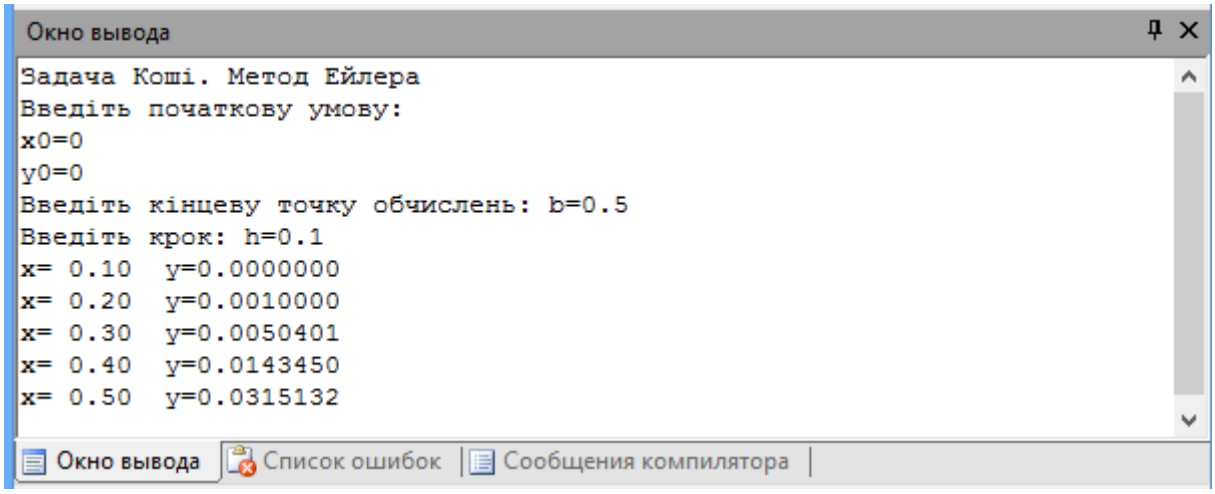


Рисунок 8.7 – Розв'язок задачі Коші методом Ейлера

Метод "предиктор-коректор".

```
program Pred-Kor;
```

```
var   b,h,x,y,yy,f1,f2:real;
   function f(x,y:real):real;
        begin
              f:=sqr(y+x);
        end;
begin
       writeln('Задача Коші. Метод предиктор-коректор.');
       writeln('Введіть початкову умову:');
       write('x0='); read(x);
       write('y0='); read(y);
       write('Введіть кінцеву точку обчислень: b='); read(b);
       write('Введіть крок: h='); read(h);
       while x<=b-h/2 do
              begin
                    yy:=y+h*f(x,y);
                    f1:=f(x,y);
                    f2:=f(x+h,yy);
                    y:=y+h*(f1+f2)/2.0;
                    x:=x+h;
                    writeln('x=',x:5:2,'  y=',y:9:7,'  y~=',yy:9:7,'  f1=',f1:9:7,'  f2=',f2:9:7);
              end;
end.
```

Результати роботи програми приведені на рис. 8.8.

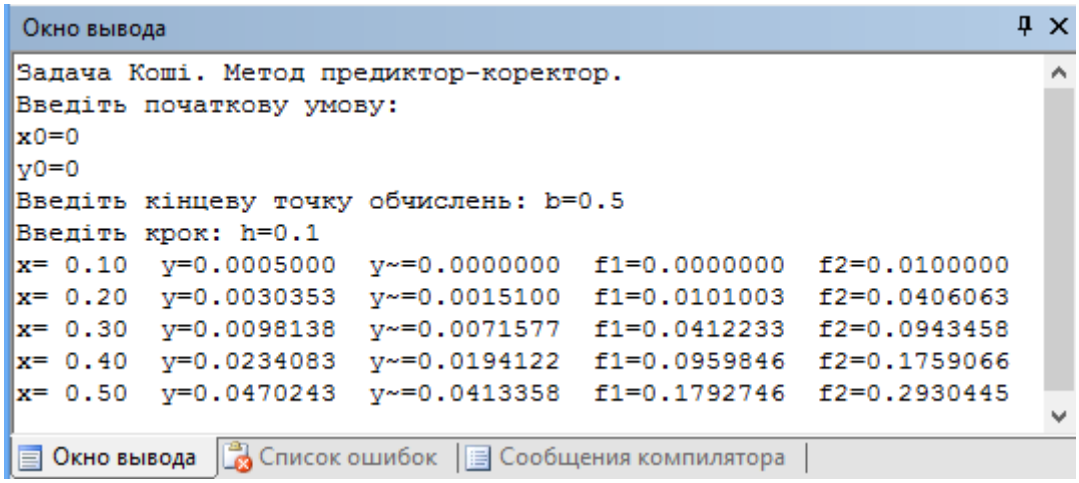


Рисунок 8.8 – Розв'язок задачі Коші методом “предиктор-коректор”

Метод Рунге-Кутти.

```
program Runge_Kutta;
var   b,h,x,y,k1,k2,k3,k4,dy,yy:real;
   function f(x,y:real):real;
      begin
            f:=sqr(y+x);
      end;
begin
       writeln('Задача Коші. Метод Рунге-Кутти');
```

```
        writeln('Введіть початкову умову:');
        write('x0='); read(x);
        write('y0='); read(y);
        write('Введіть кінцеву точку обчислень: b='); read(b);
        write('Введіть крок: h='); read(h);
        while x<=b-h/2 do
            begin
                k1:=f(x,y);
                yy:=y+h*k1/2.0;
                k2:=f(x+h/2.0,yy);
                yy:=y+h*k2/2.0;
                k3:=f(x+h/2.0,yy);
                yy:=y+h*k3;
                k4:=f(x+h,yy);
                dy:=h*(k1+2*k2+2*k3+k4)/6;
                writeln('x=',x:5:2,'  y=',y:9:7,'  k1=',k1:9:7,'  k2=',k2:9:7,'  k3=',k3:9:7,
                                                '  k4=',k4:9:7,'  dy=',dy:9:7);
                y:=y+dy;
                x:=x+h;
            end;
        writeln('x=',x:5:2,'  y=',y:9:7);
end.
```

Результати роботи програми приведені на рис. 8.9.

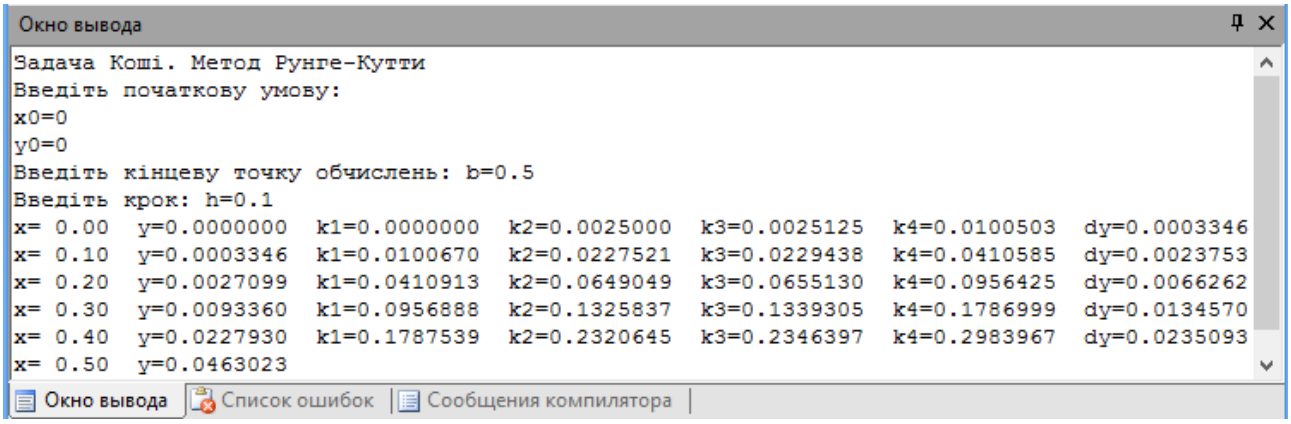


Рисунок 8.9 – Розв'язок задачі Коші методом Рунге-Кутти

Контрольні запитання

1. Сформулюйте задачу Коші для звичайного диференційного рівняння першого порядку.
2. Геометрична інтерпретація задачі Коші.
3. Сутність методу Ейлера, його точність.
4. Сутність модифікації метода Ейлера у методі “предиктор-коректор”.
5. Геометрична сутність метода Рунге-Кутти.
7. Які ще методи розв’язання задачі Коші Вам відомі?

Перелік рекомендованої літератури

1. Фельдман Л.П. Чисельні методи в інформатиці / Л.П. Фельдман, А.І. Петренко, О.А. Дмитрієва. –К.: BHV, 2006. – 480 с.
2. Пирумов У. Г. Численные методы: учеб. пособие для студ. втузов / У.Г. Пирумов. – 4-е изд., стереотип. – М.: Дрофа, 2007. – 221, [3] с.: ил.
3. Численные методы. Сборник задач: учеб. пособие для вузов / В.Ю. Гидаспов, И.Э. Иванов, Д.Л. Ревизников и др.; под ред. У.Г. Пирумова. – М.: Дрофа, 2007. – 144 с.: ил.