МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії
(повна назва кафедри, циклової комісії)

КУРСОВА РОБОТА

з _____Основи програмування_____
(назва дисципліни)

на тему:__Розв'язання СЛАР точними методами_____
_____

Студента (ки, ів) __1__ курсу, групи _ІП-35_
Адаменко Арсен Богданович

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник
_____Головченко М.М_
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: _____
Національна оцінка _____

Члени комісії _____ _____
(підпис) (посада, вчене звання, науковий ступінь, прізвище та ініціали)

_____ _____
(підпис) (посада, вчене звання, науковий ступінь, прізвище та ініціали)

Київ- 202_ рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрям "ІПЗ"

Курс 1 Група ІП-35 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Адаменко Арсен Богданович

(прізвище, ім’я, по батькові)

1. Тема роботи Розв'язання СЛАР точними методами

2. Строк здачі студентом закінченої роботи !!!!!!!!!!!!!!

3. Вихідні дані до роботи Технічне завдання додаток А

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

ВСТУП, ПОСТАНОВКА ЗАДАЧІ, ТЕОРЕТИЧНІ ВІДОМОСТІ, ОПИС АЛГОРИТМІВ, ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ВИСНОВКИ, ПЕРЕЛІК ПОСИЛАНЬ, ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ, ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6. Дата видачі завдання 03 квітня 2024

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	03 квітня 2024	
2.	Підготовка ТЗ	03 квітня 2024	
3.	Пошук та вивчення літератури з питань курсової роботи	22 квітня 2024	
4.	Розробка сценарію роботи програми	06 травня 2024	
6.	Узгодження сценарію роботи програми з керівником	15 травня 2024	
5.	Розробка (вибір) алгоритму рішення задачі	06 травня 2024	
6.	Узгодження алгоритму з керівником	15 травня 2024	
7.	Узгодження з керівником інтерфейсу користувача	8 травня 2024	
8.	Розробка програмного забезпечення	01 травня 2024	
9.	Налагодження розрахункової частини програми	15 травня 2024	
10.	Розробка та налагодження інтерфейсної частини програми	15 травня 2024	
11.	Узгодження з керівником набору тестів для контрольного прикладу	17 травня 2024	
12.	Тестування програми	18 травня 2024	
13.	Підготовка пояснювальної записки		
14.	Здача курсової роботи на перевірку		
15.	Захист курсової роботи		

Студент _____
(підпис)

Керівник _____ Головченко Максим Миколайович

(підпис) (прізвище, ім’я, по батькові)

"___" _____20__ р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 55 сторінок, 24 рисунки, 28 таблиць, 4 посилання.

Мета роботи: Метою курсової роботи є забезпечення надійності та коректності програмного забезпечення для розв’язання СЛАР різними точними методами.

Вивчено методи: LUP-метод, метод Гауса-Холецького, метод обертання.

Виконана програмна реалізація алгоритму LUP-метода, метода Гауса-Холецького, метода обертання.

СИСТЕМА ЛІНІЙНИХ РІВНЯНЬ, ТОЧНІ МЕТОДИ ВИРІШЕННЯ, LUP-МЕТОД, МЕТОД ГАУСА-ХОЛЕЦЬКОГО, МЕТОД ОБЕРТАННЯ.

ЗМІСТ

ВСТУП

Дана робота висвітлює методи розв’язання систем лінійних алгебраїчних рівнянь для вирішення великої кількості задач та проблем у напрямках галузі інформаційних технологій, які пов’язані з такими речами, алогритмами та методами, як графіка, штучний інтелект і машинне навчання.

Дана робота та програмне забезпечення є актуальним через використання нових підходів до розробки та супроводу програмного продукту, забезпечення додактового функціоналу для програмного забезпечення з цим призначенням.

Дане програмне забезпечення призначене для розв’язання систем лінійних алгебраїчних рівнянь різними методами з підрахунком практичної часової складності (кількість ітерацій), візуалізації розв’язків систем рівнянь, а також вивід розв’язків систем рівнянь до текстового файлу.

1 Постановка задачі

Розробити програмне забезпечення, що буде знаходити рішення для заданої СЛАР наступними методами:

а) LUP-метод;

б) метод Гауса-Холецького;

в) метод обертання;

Вхідними даними для даної роботи є СЛАР, яка задана в матричному вигляді:

$$AX = B$$

, де A – матриця коефіцієнтів, X – вектор шуканих значень (рішення системи),

B – вектор вільних членів. Програмне забезпечення повинно оброобробляти матрицю коефіцієнтів та стовпець вільних членів для СЛАР розмірність яких знаходиться в межах від 1 до 10.

Вихідними даними для даної роботи являється сукупність дійсних чисел, що є розв’язками даної системи, які виводяться на екран. Програмне забезпечення повинно видавати розв’язок за умови, що для вхідних даних обраний метод сходиться. Якщо це не так, то програма повинна вивести відповідне повідомлення. Якщо розмірність системи рівне двом невідомим, то програмне забезпечення повинно виводити графік системи. Якщо система не має розв’язків або їх нескінченна кількість, то програма повинна видати відповідне повідомлення.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

Систему лінійних алгебраїчних рівнянь (далі — СЛАР) з n рівнянь можна задати наступним чином:

$$AX = B \quad (2.1)$$

де:

$$A = \begin{bmatrix} a_{11} & a_{21} & \dots & 2 \\ a_{12} & a_{22} & \dots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \vdots & a_{nn} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Тоді якщо , то система (2.1) має розв’язок та він є лише єдиним. Якщо система має єдиний розв’язок, то його можна знайти одним із наступних методів.

2.1. LUP-метод

Цей метод опирається на розкладання матриці коефіцієнтів A у вигляді добутку матриць L та U:

$$A = LU \quad (2.2)$$

де L — нижня трикутна матриця, а U — верхня трикутна матриця, де усі діагональні елементи дорівнюють 1:

$$L = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \dots & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & u_{12} & \dots & u_{1n} \\ 0 & 1 & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \quad (2.3)$$

Тоді з цього твердження випливає, що LUP-метод розв’язання СЛАР складається з двох головних етапів:

1. етапу факторизації матриці A;
2. етапу отримання X.

Під час факторизації вектор B не змінюється, як ви бачите.

Із виразів (2.2) та (2.3) випливає, що значення всіх елементів матриці A можна записати наступним компактним чином:

$$\begin{aligned} a_{ij} &= \left(\sum_{k=1}^{j-1} l_{ik} u_{kj} \right) + l_{ij} \\ a_{ji} &= \left(\sum_{k=1}^{j-1} l_{jk} u_{ki} \right) + l_{jj} u_{ji} \end{aligned} \quad (2.4)$$

де $i=\overline{1,n}$, $j=\overline{1,i}$.

У виразі (2.4) записано, що i — номер рядка матриці L та номер стовпця матриці U, а j — вже номер стовпця матриці L та номер рядка матриці U.

З рівняння (2.4) випливає, що факторизація відбувається за n стадій. На кожній стадії j поточний елемент a_{ji} матриці L наступним чином:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}, \quad i=\overline{j,n} \quad (2.5)$$

а елемент u_{ji} вже іншим:

$$u_{ji} = \frac{a_{ji} - \sum_{k=1}^{j-1} l_{jk} u_{ki}}{l_{jj}}, \quad i=\overline{j+1,n} \quad (2.6)$$

Елементи двох матриць обчислюються у шаховому порядку.

За деяких умовах за формулами (2.5) та (2.6) значення елементів матриці L та U можна побати як $l_{i1}=a_{i1}, i=\overline{1,n}$, а $u_{1j}=\frac{a_{1j}}{l_{11}}, j=\overline{2,n}$.

Так як метод називається як LUP-методом, а не LU-методом, то він має ще один вбудований крок для факторизації матриці A. Так як при обрахуванні елемента u_{ji} значення l_{jj} може набувати від'ємного значення, що зробить розв’язання неможливим, або настільки близьким до нуля, що точність вирішення СЛАР може погіршитися.

Одним з методов вирішення цієї проблеми є знаходження такого рядка $k=\overline{j,n}$, для якого:

$$|a_{kj}| = |\max a_{ij}|, \quad i=\overline{j,n} \quad (2.7)$$

Після чого треба зберігти вектор P, де $P_i \in \overline{1,n}, i=\overline{1,n}$ зберігає оригінальний номер рядка матриці A. По замовчуванням матриця P має наступну початкову конфігурацію перед факторизацією: $P_i=i, i=\overline{1,n}$

Так як перед обрахуванням елементів матриці L та U в ітерації j елементи рядків $i=\overline{j,n}$ є незмінними до цього часу, то слід зробити перестановку перед поточною ітерацією j, а також обміняти місцями значення P_j та P_j, де j — поточний номер ітерації, а i — значення рядка, для якого формула (2.7) коректною.

В результаті факторизації ми маємо наступне рівняння СЛАР:

$$LUX=B \quad (2.8)$$

Далі рівняння (2.8) можна перезаписати як:

$$LY=B, Y=UX \quad (2.9)$$

Потім в нас з (2.9) з'являється наступне рівняння для розв'язку Y:

$$LY=B \quad (2.10)$$

А також (2.9) можна перетворити для розв'язку X:

$$UX=Y \quad (2.11)$$

Так як L є трикутною матрицею, то для розв'язку Y ми можемо перетворити рівняння (2.10) на наступне:

$$y_i=\frac{b_{p_{ii}}-\sum_{j=1}^{i-1} l_{ij} y_i}{l_{ii}}, \quad i=\overline{1,n}$$

де P — вектор перестановки для матриці A.

Так як U теж є трикутною матрицею, то вектор-стовпець розв'язок X можна перезаписати за допомогою формули (2.11) як:

$$x_i=y_i-\sum_{j=i+1}^{n} u_{ij} x_j, \quad i=\overline{n,1}$$

2.2. Метод Гауса-Холецького

Цей метод використовується для розв'язання СЛАР з матрицею коефіцієнтів A, яка має бути емітовою. Він грунтується на розкладанні матриці на добуток матирць L, D та $L^\dagger$:

$$A=LDL^\dagger \quad (2.12)$$

де L — нижня трикутна матриця, D — діагональна матриця, $L^{\dagger}$ — комплексно спряженя матриця до матриці L та є верхньо трикутною до неї.

З виразу (2.12) випливає, що кожен елемент матриці A можна записати як:

$$a_{ij}=\sum_{k=1}^{i} l_{ik} d_{kk} \overline{l_{jk}}, \quad i \geq j \ (2.13)$$

де $\overline{l_{jk}}$ — елемент, комплексно-спряжений до l_{jk}.

У випадку, якщо $i=j$, то можна одержати наступне рівняння:

$$d_{jj} l_{jj}^2 = a_{jj} - \sum_{k-1}^{j-1} l_{jk}^2 d_{kk} \ (2.14)$$

Якщо $l_{ii}=1, i=\overline{1,n}$, то вираз (2.14) можна перезаписати вже ось так:

$$d_{jj} = a_{jj} - \sum_{k=1}^{j-1} (k_{jk}^2 d_{kk}), \quad j=\overline{1,n} \ (2.15)$$

Тепер якщо $i>j$, то вираз (2.14) вже буде мати наступний вигляд:

$$l_{ij} = \frac{a_{ij} - \sum_{j=1}^{j-1} l_{ik} d_{kk} \overline{l_{jk}}}{d_{jj}}, \quad j=\overline{1,n}, i=\overline{j+1,n} \ (2.16)$$

Таким чином, LDL факторизація матриці відбувається за n ітерацій. Всі елементи $l_{ii}, i=\overline{1,n}$ дорвівнюють 1. На кожній ітерації j треба обчислити значення елементів d_{jj} за формулою (2.15), а потім — l_{ij} за (2.16).

Після факторизації матриці A ми отримаємо наступне рівняння:

$$LDL^{\dagger} X = B \ (2.17)$$

Вираз (2.17) можна перезаписати як:

$$LY = B, \quad Y = DL^{\dagger} X \ (2.18)$$

далі як:

$$DZ = Y, \quad Z = L^{\dagger} X \ (2.19)$$

в кінці вже як:

$$L^{\dagger} X = Z \ (2.20)$$

З рівності (2.18) випливає, що:

$$y_i = b_i - \sum_{j=1}^{i-1} l_{ij} y_j, \quad i=\overline{1,n}$$

З виразу (2.19):

$$z=\frac{y_i}{d_{ii}},\quad i=\overline{1,n}$$

Далі з (2.20) випливає остаточним вираз обчислення вектора X:

$$x_i=z_i-\sum_{j=i+1}^{n}\overline{l_{ji}}\,x_j,\quad i=\overline{n,1}$$

2.3. Метод обертання

Немає відповідної літератури

3 ОПИС АЛГОРИТМІВ

Перелік всіх основних змінних та їхнє призначення наведено в таблиці 3.1.

Таблиця 3.1 — Основні змінні та їхні призначення.

Змінна	Призначення
A	Матриця коефіцієнтів
B	Вектор вільних коефіцієнтів
IsSolvable	Ознака можливості вирішення СЛАР

3.1 Загальний алгоритм

1. ПОЧАТОК
2. Зчитати розмірність системи.
3. Зчитати матрицю системи та стовпець вільних членів:
   - 3.1. Зчитати матрицю коефіцієнтів:
     - 3.1.1. Цикл проходу по всіх рядках матриці системи (a_i — поточний рядок):
       - 3.1.1.1. Цикл проходу всіх стовпцях матриці системи (a_{ij} — поточний рядок):
         - 3.1.1.1.1. ЯКЩО поточний елемент матриці – вірно записане число, ТО записати його в відповідну комірку A. ІНАКШЕ видати повідомлення про помилку та перейти до пункту 8.
   - 3.2. Зчитати вектор вільних коефіцієнтів:
     - 3.2.1. Цикл проходу по всіх елементах стовпця вільного членів:
       - 3.2.1.1. Якщо поточний елемент вектора вільних коефіцієнтів – вірно записане число, ТО записати його в відповідну комірку B. ІНАКШЕ видати повідомлення про помилку та перейти до пункту 8.

4. ЯКЩО обраний LUP-метод, ТО обробити дані згідно алгоритму методу Якобі (підрозділ 3.2).
5. ЯКЩО обраний метод Гауса-Зейделя, ТО обробити дані згідно алгоритму методу Якобі (підрозділ 3.3).
6. ЯКЩО обраний метод обертання, ТО обробити дані згідно алгоритму методу Якобі (підрозділ 3.4).
7. ЯКЩО IsSolvable дорівнює правді, ТО:
   7.1. ЯКЩО обрана система на дві невідомих, ТО побудувати та вивести графік системи.
   7.2. Вивести рішення системи.
   7.3. Записати систему та її рішення у файл.
8. ІНАКШЕ вивсети повідомлення про неможливість розв’язання СЛАР вказаним методом.
9. КІНЕЦЬ

3.2 Алгоритм LUP-методу

1. ПОЧАТОК
2. Отримати розмірність матриці *A* як *n*.
3. Створити новий вектор *P* розміром *n*.
4. Заповнення вектора *P* початковими значеннями:
   4.1. ЦИКЛ по всім індексам нової матриці P лічильником *i*:
      4.1.1. Встановити значення p_i як *i*.
5. Створити нову матрицю *NA* розмірністю в *n* рядків та *n* стовпців.
6. Копіювання матриці *A* до матриці *NA*:
   6.1. ЦИКЛ по всім рядкам нової матриці *NA* лічильником *i*:
      6.1.1. ЦИКЛ по всім стовпцям нової матриці *NA* лічильником *j*:
         6.1.1.1. Встановити значення na_{ij} як у елемента a_{ij}.
7. Знайти факторизацію матриці *NA*:

7.1. Обміняти значення двох рядків матриці *NA* для забезпечення можливості розв'язання у випадку, коли $na_{jj}=0$:

7.1.1. ЦИКЛ для змінної *j* від *1* до *n*:

7.1.1.1. Знайти рядок з максимальним значенням *i*, де $|na_{ij}|=|\max na_{kj}|$, $j=\overline{j,n}$, $i=\overline{j,n}$:

7.1.1.1.1. Встановити значення для змінної *MaxColumnValue* як a_{jj}.

7.1.1.1.2. Встановити значення для змінної *MaxColumnIndex* як *j*.

7.1.1.1.3. ЦИКЛ для змінної *i* від *j* до *n*:

7.1.1.1.3.1. ЯКЩО $|na_{ij}|>|na_{MaxColumnIndex,j}|$, ТО

7.1.1.1.3.2. Встановити значення для змінної *MaxColumnValue* як na_{ij}.

7.1.1.2. ЯКЩО $MaxColumnValue=0$ є правдою, ТО

7.1.1.2.1. Встановити змінну *IsSolvable* як хиба.

7.1.1.2.2. Перейти до пункту 11.

7.1.2. ЦИКЛ для змінної *i* від *1* до *n*:

7.1.2.1. Обміняти елементи матриці *NA* $a_{j,i}$ та $na_{MaxColumnIndex,i}$ місцями.

7.1.3. Обміняти елементи вектора *P* p_j та $p_{MaxColumnIndex}$ місцями.

7.2. Ітерація факторизації матриці *NA*:

7.2.1. ЦИКЛ для змінної *i* від *j* до *n*:

7.2.1.1. Встановити змінну *s* як суму циклу з початковим значенням 0.

7.2.1.2. ЦИКЛ для змінної *k* від *1* до *j - 1*:

7.2.1.2.1. Збільшити значення змінної *s* як суми циклу на $na_{ik}a_{kj}$.

7.2.1.3. Встановити значення елементу матриці a_{ij} як $a_{ij}-s$.

7.2.1.4. ЯКЩО $i>j$, ТО

7.2.1.4.1. Встановити змінну *s* як суму циклу з початковим значенням 0.

7.2.1.4.2. ЦИКЛ для змінної *k* від *0* до *j - 1*:

7.2.1.4.2.1. Збільшити значення змінної *s* як суми циклу на $na_{jk}a_{ki}$.

7.2.1.4.3. Встановити значення елементу матриці a_{ji} як $\frac{a_{ji}-s}{a_{jj}}$.

7.3. Створити нову матрицю *L* розмірністю в *n* рядків та *n* стовпців.

7.4. Створити нову матрицю *U* розмірністю в *n* рядків та *n* стовпців.

7.5. Заповнити матрицю *L* матрицею *NA*:

7.5.1. ЦИКЛ по всім рядкам нової матриці *L* лічильником *j*:

7.5.1.1. ЦИКЛ для змінної *i* від *1* до *j*:

7.5.1.1.1.1. Встановити значення l_{ji} як na_{ji}.

7.6. Заповнити матрицю *L* матрицею *NA*:

7.6.1. ЦИКЛ по всім рядкам нової матриці *U* лічильником *j*:

7.6.1.1. ЦИКЛ для змінної *i* від *j* до *n*:

7.6.1.1.1. ЯКЩО $i>j$, ТО

7.6.1.1.1.1. Встановити значення u_{ji} як a_{ji}.

7.6.1.1.2. ІНАКШЕ

7.6.1.1.2.1. Встановити значення u_{ji} як 1.

8. Обчислити значення вектора *Y*:

8.1. Створити новий вектор *Y* розміру *n*.

8.2. ЦИКЛ для змінної *i* від *1* до *n*:

8.2.1. Встановити змінну *s* як суму циклу з початковим значенням 0.

8.2.2. ЦИКЛ для змінної *k* від *0* до *i - 1*:

8.2.2.1. Збільшити значення змінної *s* як суми циклу на $l_{ik}y_k$.

8.2.3. Встановити значення вектора y_i як $\frac{b_{p_i}-s}{l_{ii}}$.

9. Обчислити значення вектора *X*:
    9.1. Створити новий вектор *Y* розміру *n*.
    9.2. ЦИКЛ для змінної *i* від *n* до *1* донизу:
        9.2.1. Встановити змінну *s* як суму циклу з початковим значенням 0.
        9.2.2. ЦИКЛ для змінної *k* від *i + 1* до *n*:
            9.2.2.1. Збільшити значення змінної *s* як суми циклу на $u_{ik}x_k$.
        9.2.3. Встановити значення розв'язка x_i як $y_i - s$.
10. Встановити змінну *IsSolvable* як правда.
11. КІНЦЕЬ

3.3 Алгоритм методу Гауса-Холецького

1. ПОЧАТОК
2. Задати змінній *n* значення розміру матриці A.
3. Обчислення розкладу Холецького:
    3.1. Створити квадратну матрицю L розміром *n*.
    3.2. Створити квадратну матрицю D розміром *n*.
    3.3. ЦИКЛ для змінної j від 1 до *n*:
        3.3.1. Встановити значення елементу матриці $L_{j,j}$ як 1.
        3.3.2. Встановити значення для змінної суми *s* значення 0.
        3.3.3. ЦИКЛ для змінної *k* від 1 до $j-1$:
            3.3.3.1. Збільшити змінну суми *s* на $L_{j,k}^2 D_{k,k}$.
        3.3.4. Встановити значення елементу матриці $D_{j,j}$ як $A_{j,j} - s$.
        3.3.5. ЦИКЛ для змінної *i* від j+1 до *n*:
            3.3.5.1. ЯКЩО $D_{j,j}$ дорівнює 0:
                3.3.5.1.1. Встановити змінну *IsSolvable* як хиба.
                3.3.5.1.2. Перейти до пункту 9.
            3.3.5.2. Встановити значення для змінної суми *s* значення 0.

3.3.5.3. ЦИКЛ для змінної k від 1 до $j-1$:

3.3.5.3.1. Збільшити змінну суми s на $L_{i,k} D_{k,k} L_{j,k}$.

3.3.5.4. Встановити значення елементу матриці $L_{i,j}$ як $\frac{A_{i,j}-s}{D_{j,j}}$.

4. Обчислення стовбця Y:

4.1. Створити стовпець-вектор Y розміром n.

4.2. ЦИКЛ для змінної i від 1 до n:

4.2.1. Встановити значення для змінної суми s значення 0.

4.2.2. ЦИКЛ для змінної k від 1 до $i-1$:

4.2.2.1. Збільшити змінну суми s на $L_{i,j} Y_j$.

4.2.3. Встановити значення елементу вектора Y_i як $B_i - s$.

5. Обчислення стовбця Z:

5.1. Створити стовпець-вектор Z розміром n.

5.2. ЦИКЛ для змінної i від 1 до n:

5.2.1. ЯКЩО $D_{j,j}$ дорівнює 0:

5.2.1.1. Встановити змінну *IsSolvable* як хиба.

5.2.1.2. Перейти до пункту 9.

5.2.2. Встановити значення елементу вектора Z_i як $\frac{Y_i}{D_{i,i}}$.

6. Обчислення стовбця X:

6.1. Створити стовпець-вектор X розміром n.

6.2. ЦИКЛ для змінної i від n до 1 змінюючи змінну на -1:

6.2.1. Встановити значення для змінної суми s значення 0.

6.2.2. ЦИКЛ для змінної k від $i+1$ до n:

6.2.2.1. Збільшити змінну суми s на $L_{j,i} X_k$.

6.2.3. Встановити значення елементу вектора X_i як $Z_i - s$.

7. Перевірити коректність результиту виконання методу:

7.1. Перевірити, чи є результат алгоритму коректним:

7.1.1. Створити новий вектор стовпець *NewB* розміром n.

7.1.2. ЦИКЛ для змінної y від 1 до n:

7.1.2.1. ЦИКЛ для змінної x від 1 до n:

7.1.2.1.1. Встановити значення елементу матриці $NewB_y$ як $A_{y,x}X_x$

.

7.1.3. ЦИКЛ для змінної y від 1 до n:

7.1.3.1. ЯКЩО $|NewB_y - New_x|$ не дорівнює 0:

7.1.3.1.1. Встановити змінну *IsSolvable* як хиба.

7.1.3.1.2. Перейти до пункту 9.

7.2. Перевірити, чи є результат алгоритму лише одним з рішенням СЛАР:

7.2.1. ЯКЩО n менше або дорівнює 2:

7.2.1.1. Перейти до пункту 8.

7.2.2. Встановити змінній *IsAmbigious* значення хиби.

7.2.3. ЦИКЛ для змінної i від 1 до n:

7.2.3.1. ЯКЩО B_i не дорівнює 0 або X_i не дорівнює 0:

7.2.3.1.1. Встановити змінній *IsAmbigious* значення правди.

7.2.4. ЯКЩО значення *IsAmbigious* дорівнює хибі.

8. Встановити змінну *IsSolvable* як правда.
9. КІНЕЦЬ

3.4 Загальний методу обертання

1. ПОЧАТОК
2. Отримати розмірність матриці A як n.
3. Створити нову матрицю NA розмірністю в n рядків та $n + 1$ стовпців.
4. Заповити нову матрицю NA значеннями з матриці A та B.

4.1. ЦИКЛ по всім рядкам нової матриці лічильником i:

4.1.1. ЦИКЛ по всім стовпцям нової матриці лічильником j:

4.1.1.1. ЯКЩО вираз $j <= n$ є правдивим, ТО

4.1.1.1.1. Встановити значення na_{ij} як у елемента a_{ij}.

4.1.1.2. ІНАКШЕ

4.1.1.2.1. Встановити значення na_{ij} як у елемента b_i.

5. Обчислити розкладання нової матриці NA:

5.1. ЦИКЛ для змінної *i* від *1* до *n - 1*:

5.1.1. ЦИКЛ для змінної *j* від *i* + *1* до *n*:

5.1.1.1. Встановити значення для змінної b як na_{ji}.

5.1.1.2. Встановити значення для змінної a як na_{ii}.

5.1.1.3. ЯКЩО значення a^2+b^2 не є позитивним, ТО

5.1.1.3.1. Встановити змінну *IsSolvable* як хиба.

5.1.1.3.2. Перейти до пункту 8.

5.1.1.4. Встановити значення для змінної c як $\frac{a}{\sqrt{(a^2+b^2)}}$.

5.1.1.5. Встановити значення для змінної b як $\frac{b}{\sqrt{(a^2+b^2)}}$.

5.1.1.6. ЦИКЛ для змінної *k* від *i* до *n* + *1*:

5.1.1.6.1. Встановити значення для змінної t як na_{ik}.

5.1.1.6.2. Встановити значення елементу матриці na_{ik} як $ca_{ik}+sa_{jk}$.

5.1.1.6.3. Встановити значення елементу матриці na_{jk} як $-st+ca_{jk}$.

6. Обчислити значення вектора X довжиною *n*:

6.1. ЦИКЛ для змінної *i* від *n* до *1* донизу:

6.1.1. Встановити змінну s сумми цикла як 0.

6.1.2. ЦИКЛ для змінної *j* від *i* + *1* до *n*:

6.1.2.1. Збільшити значення *s* змінної суми цикла на $a_{ij}\,x_j$.

6.1.2.2. ЯКЩО значення a_{ii} є нулем, ТО

6.1.2.2.1. Встановити змінну *IsSolvable* як хиба.

6.1.2.2.2. Перейти до пункту 8.

6.1.2.3. Встановити значення розв’язку x_i як $\frac{a_{in}-s}{a_{ii}}$.

7. Встановити змінну *IsSolvable* як правда.

8. КІНЕЦЬ

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Діаграма класів програмного забезпечення

Буде дуже скоро

4.2 Опис методів частин програмного забезпечення

4.2.1 Користувацькі методи

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметрів	Заголовочний файл
1	Vector	Vector	Створити новий пустий вектор			Vector.hpp
2	Vector	Vector	Створити новий вектор довжини size	size		Vector.hpp
3	Vector	operator[]	Отримати посилання на елемент вектора	index		Vector.hpp
4	Vector	Size	Отримати довжину вектора			
5	Matrix	Matrix	Створити нову пусту матрицю			Matrix.hpp
6	Matrix	Matrix	Створити матрицю розмірністю width на height	height, width		Matrix.hpp
7	Matrix	At	Отримати посилання на елемент	y, x		Matrix.hpp

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметрів	Заголовочний файл
			матриці			
8	Matrix	Width	Отримати кількість стовпців матриці			Matrix.hpp
9	Matrix	Height	Отримати кількість рядків матриці			Matrix.hpp
10	Matrix	TryGetEdgeSize	Спробувати отримати розмірність квадратної матриці по її ширині			Matrix.hpp
11	Matrix	IsSquare	Чи є матриця квадратною			Matrix.hpp
12	AllocArray2D<T>	AllocArray2D	Створини пустий новий двовимірний масив			AllocArray2D.inc.hpp
13	AllocArray2D<T>	AllocArray2D	Створини новий двовимірний масив шириною width та висотою height	height, width		AllocArray2D.inc.hpp
14	AllocArray2D<T>	At	Отримати посилання на елемент двовимірного масиву по ширині x та висоті y	y, x		AllocArray2D.inc.hpp

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметрів	Заголовочний файл
15	AllocArray2D<T>	Width	Отримати ширину двовимірний масиву			AllocArray2D.inc.hpp
16	AllocArray2D<T>	Height	Отримати висоту двовимірний масиву			AllocArray2D.inc.hpp
17	SolvingResult	Successful	Отримати об’єкт типу класу цього методу. Об’єкт репрезентує, що статус значення є успішним при обробці			LSESolver.hpp
18	SolvingResult	Error	Отримати об’єкт типу класу цього методу. Об’єкт репрезентує, що статус значення не є успішним при обробці			LSESolver.hpp
19	SolvingResult	SetItersCount	Встановити кількість ітерацій при виконанні алгоритмів та вернути оригінальний об’єкт	itersCount		LSESolver.hpp

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметрів	Заголовочний файл
20	LSESolver	LSESolver	Конструктор стану об’єкта цього типу			LSESolver.hpp
21	LSESolver	~LSESolver	Віртуальний деструктор для класів-нащадків			LSESolver.hpp
22	LSESolver	SetEquationsCount	Встановити кількість лінійних рівнянь єдиножди	eqsCount		LSESolver.hpp
23	LSESolver	SetVariablesCoefficients	Встановити матрицю коефіцієнтів			LSESolver.hpp
24	LSESolver	SetFreeCoefficients	Встановити вектор вільних членів			LSESolver.hpp
25	LSESolver	SolveLSE	Почати процес розв’язку встановленної ЛСАР			LSESolver.hpp
26	LSESolver	IsLSESolvedSuccessfully	Отримати статус виконання алгоритму розв’язку			LSESolver.hpp
27	LSESolver	GetSolvesValuesOnce	Отримати розв’язок встановленої СЛАР єдиножди			LSESolver.hpp
28	LSESolver	GetTotalIte	Отримати			LSESolve

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметрів	Заголовочний файл
		rationsCount	кількість ітерацій при викоанні алогритму			r.hpp
29	GUISession	GUISession	Створити об’єкт типу сесії графічного інтерфейсу			GUI.hpp
30	GUISession	Init	Ініціалізувати вікно графічної сесії аргументами командного рядка	argc, argv		GUI.hpp
31	GUISession	GetWindow	Отримати посилання на вікно програми			GUI.hpp
32	ApplicationData	GetLSEInputDataRef	Отримати посилання на вхідні дані СЛАР			GUI.hpp
33	ApplicationData	GetLSEOutputDataRef	Отримати посилання на дані розв’язку СЛАР			GUI.hpp
34	ApplicationWindow	SetApplicationData	Встановити посилання на об’єкт, що зберігає дані програми	appData		GUI.hpp
35	ApplicationWindow	ReadyWindow	Встановити стан вікна як			GUI.hpp

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметрів	Заголовочний файл
			готового для опрацювання			
36	LSEConfigurator	SetLSEInputData	Встановити місце для запису конфігурації СЛАР	lseInputData		GUI.hpp
37	LSESolveOutput	SetLSEOutputDataRef	Встановити місце для зчитання даних СЛАР	lseSolveData		GUI.hpp
38	LSESolveOutput	SetLSEInputDataRef	Встановити місце для зчитання даних про розв’язок СЛАР	lseInputData		
39	LSESolveOutput	OutputSolve	Показати розв’язки СЛАР			
40	LSESolveOutput	ClearSolve	Приховати розв’язки СЛАР			
41	LSESolverUI	SetLSEInputData	Встановити місце для зчитання даних СЛАР	lseInputData		
42	LSESolverUI	SetLSESolvesPlace	Встановити місце для запису даних про розв’язок СЛАР	lseSolvesPlace		
43	LSESolverUI	SetLSESolveOutputIORef	Встановити віджет для відображення	lseSolveOutput		

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметрів	Заголовочний файл
			розв’язків СЛАР			
44						

4.2.2 Стандартні методи

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметрів	Заголовочний файл

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 План тестування

Даний розділ посв’ячений тестуванню програмного забезпечення на наявність помилок, недоліків та інших відхилень від нормальної та коректної роботи програмного забезпечення.

У таблиці 5.1-5.11 показано тестування програмного забезпечення на різних етапах його виконання та його поведінку при різних маніпуляціях зцим програмним додатком.

У списку нижче показано короткий зміст та весь план тестування цієї програми:

а) Тестування правильності введених значень.
   1) Тестування при введенні некоректних символів.
   2) Тестування при введенні замалих та завеликих значень.

б) Тестування коректної роботи при введені систем, що не мають коренів.
   1) Тестування роботи програми при нульовому значенні визначника.
   2) Тестування роботи методу 1 на несиметричній матриці.
   3) Тестування роботи методу 1 на не додатньо визначеній матриці.

в) Тестування коректності роботи методів 1, 2, 3.
   1) Перевірка коректності роботи методу 1.
   2) Перевірка коректності роботи методу 2.
   3) Перевірка коректності роботи методу 3.

г) Тестування коректності роботи методів 1,2,3 з дробовими коефіцієнтами.
   1) Перевірка правильності результатів.
   2) …

д) Тестування побудови графіків.

5.2 Приклади тестування

Таблиця 1.1 — Приклад роботи програми при введенні некоректних символів.

Мета тесту	Перевірити можливість введення некоректних даних
Початковий стан програми	Відкрите вікно програми
Вхідні дані	2 3 b 6 S 4 6 f f y 9 17
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів
Очікуваний результат	Повідомлення про помилку формату даних
Стан програми після проведення випробувань	Видано помилку «Введіть дійсне число»

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

6.1 Робота з програмою

Після запуску виконавчого файлу з розширенням *.exe, відкривається головне вікно програми:

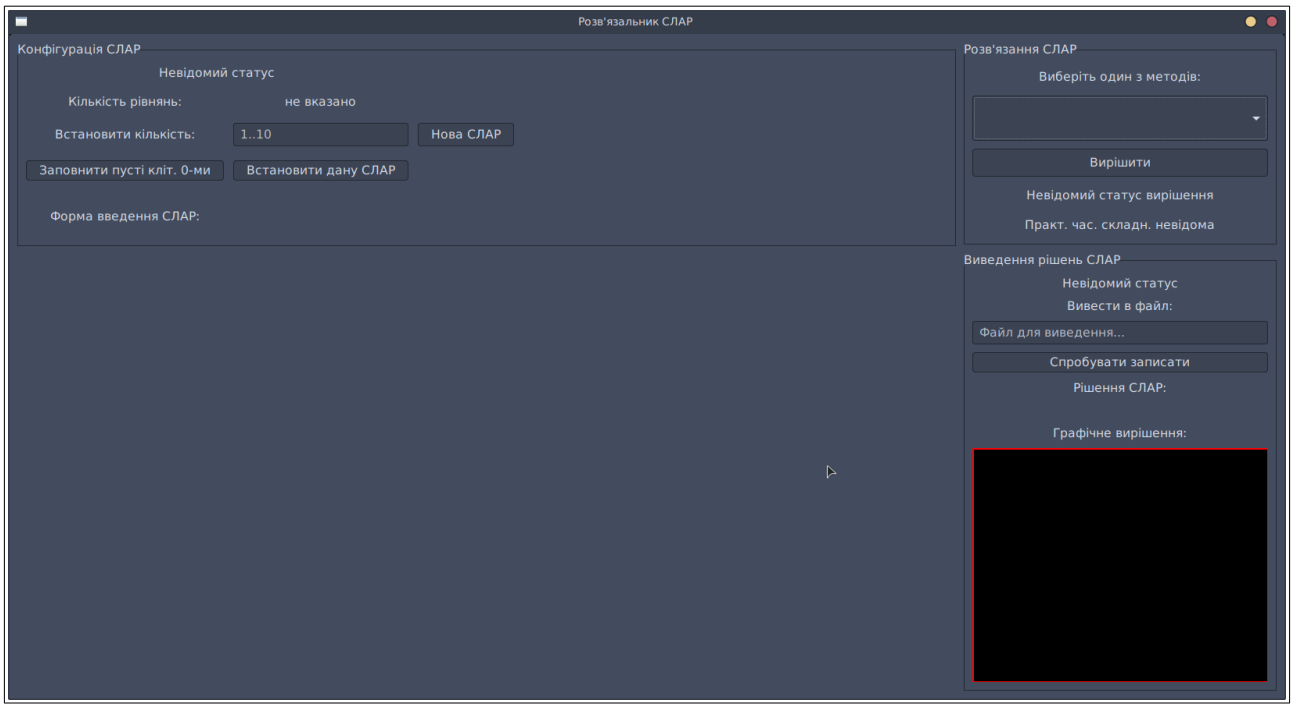

Рисунок 6.1 — Головне вікно програми

Далі треба встановити розмірність системи лінійних алгебраїчних рівнянь (далі — СЛАР) біля таблички «???», потів натиснути на клавішу «???» щоб встановити нову СЛАР, яка буде оброблятися програмою (рисунок 6.2).

Рисунок 6.2 — Вибір розміру системи

Після чого треба встановити всі коефіцієнти матриці та вільні коефіцієнти до всіх комірок форми введення (рисунок 6.3).

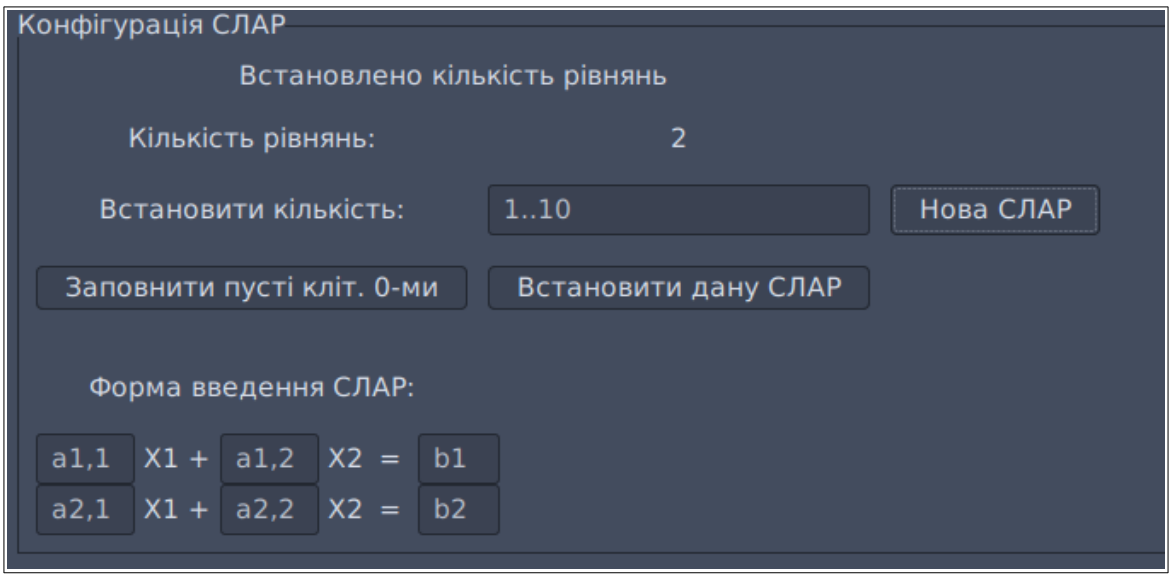

Рисунок 6.3 — Форма введення СЛАР

Потім треба натиснути на клавішу «???», що ви згодні встановити дану введену СЛАР до програми для подальшої обробки програмою (рисунок 6.3).

Якщо значення комірок форми введення СЛАР не є числами, то буде виведена помилка про немождивість підтвердити дану СЛАР. Якщо СЛАР введена правильно користувачем, то буде відображено час підтвердження СЛАР.

Далі користувач має вказати один з методів розв’язання введеної СЛАР до програми за допомогою вибору одного з методів у списку (рисунок 6.4) у підменю «???».

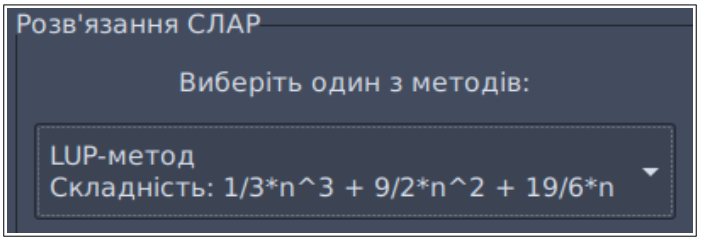

Рисунок 6.4 — Список вибору методу розв’язання СЛАР

Потім треба натиснути на клавішу «???» для запущення процесу розв’язання введеної СЛАР користувачем. Якщо СЛАР була розв’язана успішно, то програма відобразить час завершення розв’язання СЛАР та час можливості отримати розв’язок СЛАР графічно, у вигляді таблиці (рисунок 6.5) або виводом у файл в залежності від її розмірності.

Якщо СЛАР не була розв’язана вдало, то буде виведено повідомлення про неможливість розв’язку СЛАР.

Якщо СЛАР була розв’язана вдало та її розмірность дорівнює 2, то буде виведено її графічний розв’язок у вікні програми (рисунок 6.6).

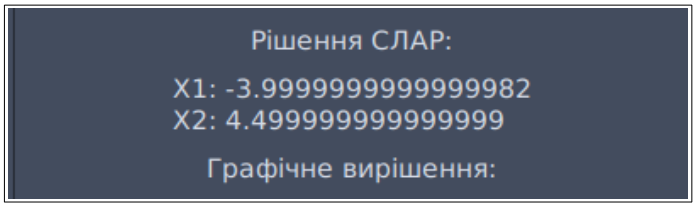

Рисунок 6.5 — Табличний метод відображення розв’язків СЛАР

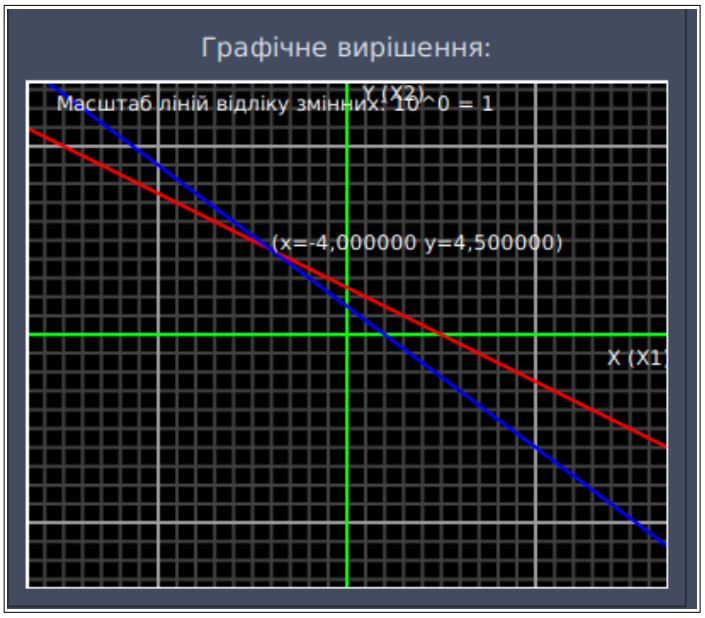

Рисунок 6.6 — Графічне вирішення СЛАР

6.2 Формат вхідних даних

Користувачу на вхід програми подається СЛАР у матричному вигяді, тобто задається за допомогою матриці коефіцієнтів та стовпця вільних членів, числа яких є дійсними. Значення кожного члена матриці коефіцієнтів має знаходитися в діапазоні $[-1000;1000]$, а значення кожного вільного члена — $[-10000;10000]$. При обробці вхідних даних введеної СЛАР користувачем у формі введення, числа коефіцієнтів матриці та стовпя вільних членів будуть округлень до 6 знака після точки донизу.

Результатом виконання програми є розв’язок встановленої СЛАР. У вікні програми невідомі змінні встановленої СЛАР будуть відображатися вертикальною таблицею значень розв’язків встановленої СЛАР. Якщо розв’язок буде виведено до файлу, то його змістом буде список дійсних чисел, де числа розділені одним пробілом, а ціла частина числа та її дробове значення буде відділене точкою.

6.3 Системні вимоги

	Мінімальні	Рекомендовані
Операційна система	ArchLinux (з останніми обновленнями)	ArchLinux (з останніми обновленнями)
Процесор	Intel i7-6700HQ (8) @ 3.500GHz	
Оперативна пам'ять	1 GB RAM	2 GB RAM
Відеоадаптер	Intel HD Graphics 530	
Дисплей	1920x1080	
Прилади введення	Клавіатура, комп’ютерна миша	
Додаткове програмне забезпечення	Пакети «base», «base-devel», «gtkmm3»	

7 АНАЛІЗ РЕЗУЛЬАТІВ

Головною задачею курсової роботи була реалізація програми для розв’язання СЛАР наступними методами: LUP, Гауса-Холецького, обертання.

Критичні ситуації у роботі програми виявлені не були. Під час тестування було виявлено, що більшість помилок виникало тоді, коли користувачем вводилися не числові вхідні дані. Тому всі дані, які вводить користувач, перевіряються на коректність перед обробкою.

Для перевірки та доведення достовірності результатів виконання програмного забезпечення скористаюся LibreOffice Calc:

а) LUP-метод.

Результат виконання LUP-методу наведено на рисунку 7.1:

Рисунок 7.1 – Результат виконання LUP-методу

Оскільки результат виконання збігається з результатом в LibreOffice Calc (рисунок 7.2), то даний метод працює вірно.

Рисунок 7.2 – Перевірка методу Якобі в LibreOffice Calc

б) Метод Гауса-Холецького.

Результат виконання методу Гауса-Холецького наведено на рисунку 7.2:

Рисунок 7.3 – Результат виконання методу Гауса-Холецького

Оскільки результат виконання збігається з результатом в LibreOffice Calc (рисунок 7.3), то даний метод працює вірно.

Рисунок 7.4 – Перевірка методу методу Гауса-Холецького в LibreOffice Calc

в) Метод обертання.

Результат виконання методу обертання наведено на рисунку 7.5:

Рисунок 7.5 – Результат виконання методу обертання

Оскільки результат виконання збігається з результатом в LibreOffice Calc (рисунок 7.5), то даний метод працює вірно.

Рисунок 7.6 – Перевірка методу методу обертання в LibreOffice Calc

Для проведення тестування ефективності програми було створено матриці наступного вигляду:

$$R=\begin{pmatrix} n & 1 & 1 & \cdots & 1 \\ 1 & n & 1 & \cdots & 1 \\ 1 & 1 & n & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & n \end{pmatrix}, \quad A=R^T R \ (7.1)$$

де n — розмірність системи.

Матриця A (7.1) має підходити для всіх методів вирішення СЛАР, включно для методу Гауса-Холецького [3].

Результати тестування ефективності алгоритмів розв’язання СЛАР наведено в таблиці 7.1:

Таблиця 7.1 – Тестування ефективності методів

Розмірність системи	Параметри тестування	Метод		
		LUP	Гауса-Холецького	обертання
100	Кількість ітерацій циклів	378650	191900	358350
	Кількість елементарних операцій			

250	Кількість ітерацій циклів	5490375	2761000	5364625
	Кількість елементарних операцій			
500	Кількість ітерацій циклів	42793250	21459500	42291750
	Кількість елементарних операцій			
1000	Кількість ітерацій циклів	337836500	169169000	335833500
	Кількість елементарних операцій			

Візуалізація результатів табилиці 7.1 наведено на рисунку 7.1:

Рисунок 7.1 – Графік залежності кількості ітерацій методу від розміру вхідної системи

За результатами тестування можна зробити такі висновки:

а) Всі розглянуті методи дозволяю знаходити розвязки великих та надвеликих СЛАР.

б) Складність всіх розглянутих методів є кубічною, тобто — $O(n^3)$, n — розмір СЛАР.

в) З розглянутих методів найоптимальнішим для специфічних використань є метод Гауса-Холецького. Найбільш оптимальнішим для практичного використання є метод обертання, бо точність обрахунків при його використанні вища, ніж у LUP-метода.

ВИСНОВКИ

Було вивчено та реалізовано три точних методи вирішення СЛАР, які є головними алгоритмами в програмному додатку. Також було додано кілька корисних функціональних можливостей для програмного забезпечення.

Було проаналізовано практичну часову складність алгоритмів на СЛАР різного розміру на різних алгоритмах точного вирішення СЛАР за різними критеріями оцінювання ефективності виконання алгоритмів. Під час аналізування результатів було видвинуті наступні твердження:

1) 1
2) 2
3) 3

ПЕРЕЛІК ПОСИЛАНЬ

[1] Прокопенко, Ю. В., Татарчук, Д. Д., & Казміренко, В. А. (2003). Обчислювальна математика.
[2] Шахно, С. М., Дудикевич, А. Т., & Левицька, С. М. (2009). ББК В 162.73 я73 Ш81.
[3] Ayres Jr, F. (1962). Theory and problems of matrices. McGraw-Hill, 134.

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії

Затвердив
Керівник __Головченко М.М._____
«_03»___квітня_____2024 р.

Виконавець:
Студент __Адаменко А.Б.__
«_03»____квітня_____2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання курсової роботи
на тему: «Розв'язання СЛАР точними методами»
з дисципліни:
«Основи програмування»

Київ 2024

7.1 *Мета*: Метою курсової роботи є забезпечення надійності та коректності програмного забезпечення для розв’язання СЛАР різними точними методами.

7.2 *Дата початку роботи*: «_03»___квітня____2024 р.

7.3 *Дата закінчення роботи*: «___»_____ 2024 р.

7.4 *Вимоги до програмного забезпечення*.

1) Функціональні вимоги:

- Можливість ввести розмірність СЛАР.
- Можливість задавати вільні коефіцієнти та коефіцієнти невідомих змінних СЛАР.
- Можливість обирати один з методів розв’язання СЛАР.
- Розв’язати СЛАР обраним методом (LUP-методом, метод обертання, метод Гауса-Холецького (квадратного кореня)).
- Можливість зберігати результати розв’язання СЛАР у текстовий файл.
- Можливість графічного розв’язання СЛАР у випадку розмірності СЛАР у два рівняння.
- Відображати значення практичної складності алгоритмів розв’язання СЛАР.
- Нормальна обробка та реакція на некоректні дії користувача.
- Можливість відображення практичної складності алгоритму.

2) Нефункціональні вимоги:

- Можливість запускати та працювати з програмним забезпеченням на операційній системі «Linux».
- Все програмне забезпечення та супроводжувача технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 2015 - Розробка технічної документації.

7.5 *Стадії та етапи розробки*:

1) Об'єктно-орієнтований аналіз предметної області задачі (до__.__.2024 р.)
2) Об'єктно-орієнтоване проектування архітектури програмної системи (до __.__.2024 р.)
3) Розробка програмного забезпечення (до __.__.2024 р.)
4) Тестування розробленої програми (до __.__.2024 р.)
5) Розробка пояснювальної записки (до __.__.2024 р.).
6) Захист курсової роботи (до __.__.2024 р.).

7.6 *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду програмного забезпечення вирішення задачі знаходження розв'язків системи лінійних рівнянь

(Найменування програми (документа))

GIT

(Вид носія даних)

1 арк, 53 байтів

(Обсяг програми (документа), арк., Кб)

студента групи ІП-35 І курсу
Адаменко А.Б.

https://github.com/adamenko-arsen/SLE-Accurate-Solver