

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 ІІІ-35
Адаменко Арсен Богданович

Спеціальності 121 «Інженерія програмного
забезпечення»

Керівник
ст. викладач, 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- 2024 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрямок "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 БУДЕ ОТРИМАТНО

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	22 квітня 2024	
9.	Налагодження розрахункової частини програми	15 травня 2024	
10.	Розробка та налагодження інтерфейсної частини програми	28 травня 2024	
11.	Узгодження з керівником набору тестів для контрольного прикладу	24 травня 2024	
12.	Тестування програми	25 травня 2024	
13.	Підготовка пояснювальної записки	БУДЕ ОТРИМАТНО	
14.	Здача курсової роботи на перевірку	БУДЕ ОТРИМАТНО	
15.	Захист курсової роботи	БУДЕ ОТРИМАТНО	

Студент _____
(підпис)

Керівник _____
(підпис)

Головченко Максим Миколайович _____
(прізвище, ім'я, по батькові)

"__" _____ 20__ р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 62 сторіна, 14 рисунків, 12 таблиць, 3 посилання. **(МОЖЕ ЗМІНИТИСЯ)**

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.....	5
1 ПОСТАНОВКА ЗАДАЧІ.....	6
2 ТЕОРЕТИЧНІ ВІДОМОСТІ.....	7
3 ОПИС АЛГОРИТМІВ.....	8
3.1. Загальний алгоритм.....	8
3.2. Алгоритм LUP-методу.....	8
3.3. Алгоритм методу Гауса-Холеского.....	8
3.4. Алгоритм методу обертання.....	8
4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	9
4.1. Діаграма класів програмного забезпечення.....	9
4.2. Опис методів частин програмного забезпечення.....	9
4.2.1. Користувацькі методи.....	9
4.2.2. Стандартні методи.....	9
5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	10
5.1. План тестування.....	10
5.2. Приклади тестування.....	10
6 ІНСТРУКЦІЯ КОРИСТУВАЧА.....	11
7 АНАЛІЗ РЕЗУЛЬТАТІВ.....	12
ВИСНОВКИ.....	13
ПЕРЕЛІК ПОСИЛАНЬ.....	14
ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ.....	15
ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ.....	18

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, алгоритмами 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 додакового функціоналу для програмного забезпечення з призначенням цього виду.

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 з графічним інтерфейсом, 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 обробляти 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 СЛАР метод дає коректний результат. Якщо це не так, то програма повинна вивести відповідне повідомлення. Якщо розмірність системи рівне двом невідомим, то програмне забезпечення повинно виводити графік системи та його розв'язок якщо вхідну СЛАР можна вирішити вказаним методом. Якщо система не має розв'язків, 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:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Тоді якщо ранг матриці коефіцієнтів (2.1) є рівним кількості лінійних рівнянь, то система (2.1) має розв'язок та він є 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 [1]:

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

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

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

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 умов 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 методів 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 замовчуванню 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_{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 переписати 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:

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

де P — вектор перестановки для матриці A .

Так як U теж є трикутною матрицею, то вектор-стовпець розв'язок X можна переписати 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 ґрунтується на розкладанні матриці на добуток матриць L , D та L^+ [1]:

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

де L — нижня трикутна матриця, D — діагональна матриця, L^+ — комплексно спряженена матриця до матриці L та є верхньо трикутною до неї.

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

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

де \bar{l}_{jk} — елемент, комплексно-спряжений до l_{jk} .

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

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

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

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

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

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

Таким чином, LDL факторизація матриці відбувається за n ітерацій. Всі елементи $l_{ii}, i=\overline{1, n}$ дорівнюють 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:

$$LDL^+ X = B \quad (2.17)$$

Вираз (2.17) можна переписати як:

$$LY = B, \quad Y = DL^+ X \quad (2.18)$$

далі як:

$$DZ = Y, \quad Z = L^+ X \quad (2.19)$$

в кінці вже як:

$$L^+ X = Z \quad (2.20)$$

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) впливає остаточною вираз обчислення вектора X :

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

2.3. Метод обертання

Цей метод опирається на обнуленні значень матриці коефіцієнтів A та зміні вектора вільних членів B до верхньотрикутного вигляду та використанні етапу знаходження вектора значень змінних метода Гауса [2].

Давайте розглянемо систему лінійних рівнянь наступного вигляду:

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2 \\ \dots & & \dots & & \dots & & \dots & & \dots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \dots & + & a_{nn}x_n & = & b_n \end{array} \quad (2.21)$$

Нехай c_1 і s_1 — значення, відмінні від нуля. Нехай вони будуть рівними наступним значенням:

$$c_1 = \frac{a_{11}}{\sqrt{a_{11}^2 + a_{21}^2}}, \quad s_1 = \frac{a_{21}}{\sqrt{a_{11}^2 + a_{21}^2}} \quad (2.22)$$

Нехай значення матриці коефіцієнтів та вектора вільних членів будуть мати наступні значення:

$$\begin{array}{ll} a_{1j}^{(1)} = c_1 a_{1j} + s_1 a_{2j} & j = \overline{1, n}, b_1^{(1)} = c_1 b_1 + s_1 b_2 \\ a_{2j}^{(1)} = -s_1 a_{1j} + c_1 a_{2j} & j = \overline{2, n}, b_2^{(1)} = -s_1 b_1 + c_1 b_2 \end{array} \quad (2.23)$$

Другий елемент першого рівняння матриці коефіцієнтів буде мати наступне значення під час розкриття значення змінних:

$$\begin{aligned} a_{21} &= -s_1 a_{1j} + c_1 a_{2j} = -a_{21} k a_{11} + a_{11} k a_{21} = k(a_{11} a_{21} - a_{11} a_{21}) = 0 \\ k &= \frac{1}{\sqrt{a_{11}^2 + a_{21}^2}} \end{aligned}$$

Під час підстановки нових значень матриці коефіцієнтів та вектора вільних членів (2.21) по формулам (2.22) та (2.23), система лінійних алгебраїчних рівнянь буде мати наступний вигляд:

$$\begin{array}{ccccccc}
a_{11}^{(1)} x_1 & + & a_{12}^{(1)} x_2 & + & \dots & + & a_{1n}^{(1)} x_n & = & b_1^{(1)} \\
& & a_{22}^{(1)} x_2 & + & \dots & + & a_{2n}^{(1)} x_n & = & b_2^{(1)} \\
& \dots & \dots & & & & \dots & & \dots \\
a_{n1} x_1 & + & a_{n2} x_2 & + & \dots & + & a_{nn} x_n & = & b_n
\end{array}$$

Далі рівняння системи (2.24) замінюємо новим, але операція буде проводитися вже для першого та третього рівняння:

$$\begin{array}{ccccccc}
a_{11}^{(2)} x_1 & + & a_{12}^{(2)} x_2 & + & \dots & + & a_{1n}^{(2)} x_n & = & b_1^{(2)} \\
& & a_{22}^{(1)} x_2 & + & \dots & + & a_{2n}^{(1)} x_n & = & b_2^{(1)} \\
& & a_{32}^{(1)} x_2 & + & \dots & + & a_{3n}^{(1)} x_n & = & b_2^{(1)} \\
& \dots & \dots & & & & \dots & & \dots \\
a_{n1} x_1 & + & a_{n2} x_2 & + & \dots & + & a_{nn} x_n & = & b_n
\end{array} \quad (2.25)$$

Для першого та третього рівняння (2.25) будуть використані наступні формули:

$$\begin{aligned}
c_2 &= \frac{a_{11}^{(1)}}{\sqrt{a_{11}^{(1)2} + a_{31}^2}}, & s_2 &= \frac{a_{31}}{\sqrt{a_{11}^{(1)2} + a_{31}^2}} \\
a_{1j}^{(2)} &= c_2 a_{1j}^{(1)} + s_2 a_{3j} & j &= \overline{1, n}, & b_1^{(2)} &= c_2 b_1^{(1)} + s_2 b_3 \\
a_{3j}^{(1)} &= -s_2 a_{1j}^{(1)} + c_2 a_{3j} & j &= \overline{2, n}, & b_3^{(1)} &= -s_2 b_1^{(1)} + c_2 b_3
\end{aligned}$$

Після двох ітерацій можна почати перетворювати систему рівнянь для четвертого рівняння і наступних. Після наступних ітерацій ми отримаємо підсистему рівнянь з (2.25) наступного вигляду:

$$\begin{array}{ccccccc}
a_{22}^{(1)} x_2 & + & \dots & + & a_{2n}^{(1)} x_n & = & b_2^{(1)} \\
\vdots & & & & \vdots & & \vdots \\
a_{n2}^{(1)} x_2 & + & \dots & + & a_{nn}^{(1)} x_n & = & b_n^{(1)}
\end{array} \quad (2.26)$$

Тепер залишається лише перетворити рівняння (2.26) за минулим алгоритмом. В результаті ми отримаємо наступну систему рівнянь:

$$\begin{array}{ccccccc}
a_{11}^{(n-1)} x_1 & + & a_{12}^{(n-1)} & \dots & + & a_{1n}^{(n-1)} & = & b_1^{(n-1)} \\
& & a_{22}^{(n-1)} & \dots & + & a_{2n}^{(n-1)} & = & b_2^{(n-1)} \\
& & & & & \vdots & & \vdots \\
& & & & & a_{nn}^{(n-1)} x_n & = & b_n^{(n-1)}
\end{array} \quad (2.27)$$

Після чого ми можемо отримати розв'язки системи лінійних рівнянь з остаточної системи рівнянь (2.27) за допомогою зворотнього методу Гауса.

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 стовпців 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 вивести 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 $|na_{ij}| = |\max na_{kj}|, j = \overline{1, n}, i = \overline{j, n}$:

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 $a_{jk} a_{ki}$.

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 всіх рядках нової матриці L лічильником j :

7.5.1.1. ЦИКЛ для змінної i від 1 до j :

7.5.1.1.1. Встановити значення l_{ji} як a_{ji} .

7.6. Заповнити матрицю L матрицею NA :

7.6.1. ЦИКЛ по всіх рядках 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 результату 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 2:
 - 7.2.1.1. Перейти до пункту 8.
 - 7.2.2. Встановити змінній *IsAmbiguous* 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 *IsAmbiguous* значення правди.
 - 7.2.4. ЯКЩО значення *IsAmbiguous* 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. Заповнити 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 $j \leq n$ 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 елемента матриці na_{ik} як $ca_{ik} + sa_{jk}$.

5.1.1.6.3. Встановити значення елемента 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.

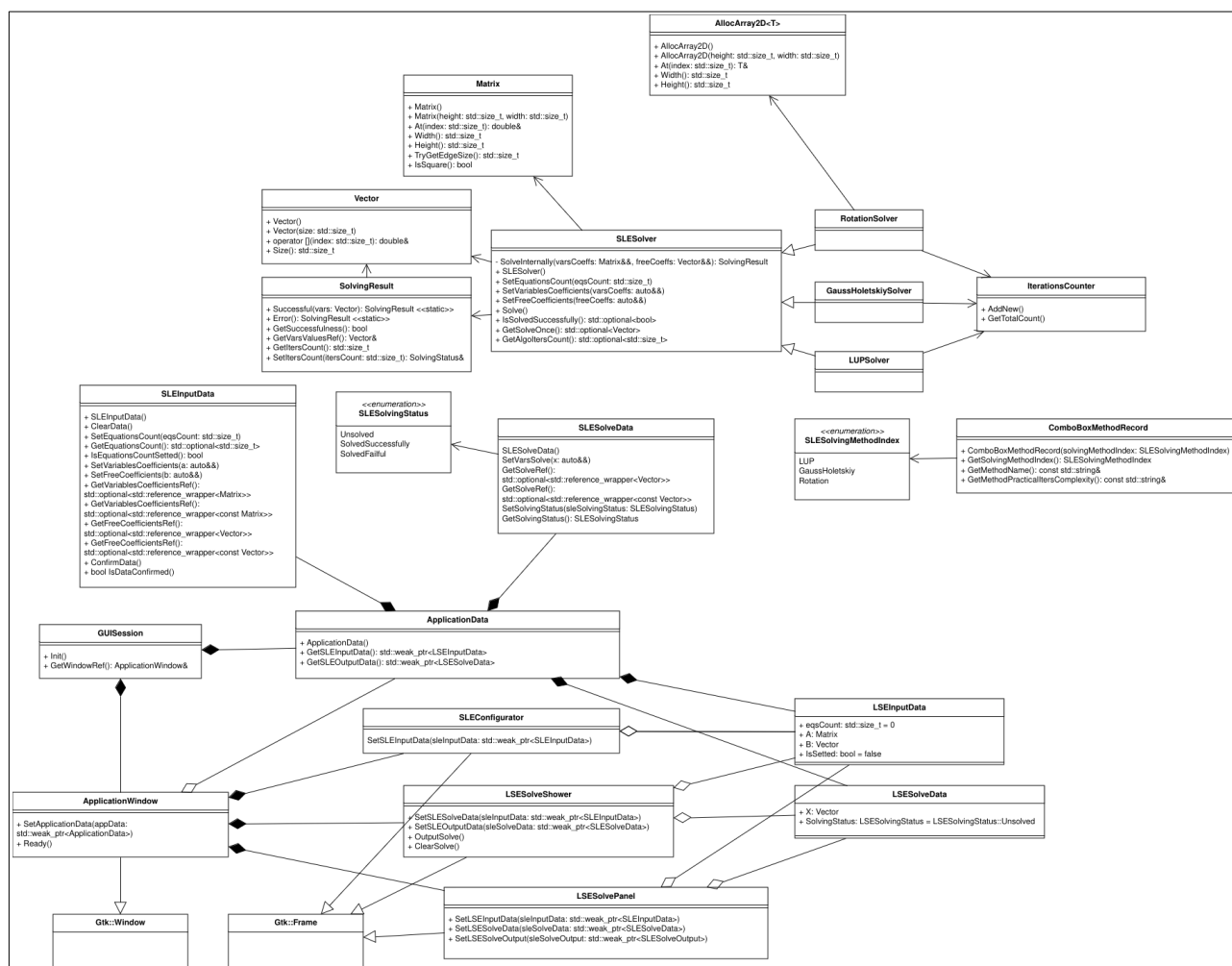
6.1.2.3. Встановити значення розв'язку x_i як $\frac{a_{in} - s}{a_{ii}}$.

7. Встановити змінну *IsSolvable* як правда.

8. КІНЕЦЬ

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕНИЯ

4.1 Діаграма класів програмного забезпечення



4.2 Опис методів частин програмного забезпечення

4.2.1 Користувачькі методи

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

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
			елемент матриці			
8.	Matrix	Width	Отримати кількість стовпців матриці			Matrix.hp p
9.	Matrix	Height	Отримати кількість рядків матриці			Matrix.hp p
10.	Matrix	TryGetEdgeSize	Спробувати отримати розмірність квадратної матриці по її ширині			Matrix.hp p
11.	Matrix	IsSquare	Чи є матриця квадратною			Matrix.hp p
12.	AllocArray2D<T>	AllocArray2D	Створити пустий новий двовимірний масив			AllocArray2D.inc.h pp
13.	AllocArray2	AllocArray	Створити	height, width		AllocArra

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
	D<T>	2D	новий двовимірний масив шириною width та висотою height			y2D.inc.h pp
14.	AllocArray2 D<T>	At	Отримати посилання на елемент двовимірного масиву по ширині x та висоті y	y, x		AllocArra y2D.inc.h pp
15.	AllocArray2 D<T>	Width	Отримати ширину двовимірний масиву			AllocArra y2D.inc.h pp
16.	AllocArray2 D<T>	Height	Отримати висоту двовимірний масиву			AllocArra y2D.inc.h pp
17.	SolvingRes	Successful	Отримати			LSESolve

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
	ult		об'єкт типу класу цього методу. Об'єкт репрезентує, що статус значення є успішним при обробці			r.hpp
18.	SolvingRes ult	Error	Отримати об'єкт типу класу цього методу. Об'єкт репрезентує, що статус значення не є успішним при обробці			LSEsolve r.hpp
19.	SolvingRes ult	GetSuccess fulness	Отримати статус, чи є результат обрахунків успішним			

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
20.	SolvingResult	GetVarsValuesRef	Отримати посилання на значення невідомих змінних СЛАР			
21.	SolvingResult	GetItersCount	Отримати кількість ітерацій алгоритму			
22.	SolvingResult	SetItersCount	Встановити кількість ітерацій при виконанні алгоритмів та повернути оригінальний об'єкт	itersCount		LSESolver.hpp
23.	SLESolver	LSESolver	Конструктор стану об'єкта цього типу			LSESolver.hpp
24.	SLESolver	~LSESolver	Віртуальний деструктор			LSESolver.hpp

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
			для класів- нащадків			
25.	SLESolver	SetEquationsCount	Встановити кількість лінійних рівнянь єдиної	eqsCount		LSESolver.hpp
26.	SLESolver	SetVariablesCoefficients	Встановити матрицю коефіцієнтів	varsCoeffs		LSESolver.hpp
27.	SLESolver	SetFreeCoefficients	Встановити вектор вільних членів	freeCoeffs		LSESolver.hpp
28.	SLESolver	Solve	Почати процес розв'язку встановленої СЛАР			LSESolver.hpp
29.	SLESolver	IsSolvedSuccessfully	Отримати статус виконання алгоритму			LSESolver.hpp

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
			розв'язку			
30.	SLESolver	GetSolveOnce	Отримати розв'язок встановленої СЛАР єдиної			LSESolver.hpp
31.	SLESolver	GetAlgoIterCount	Отримати кількість ітерацій при виконанні алгоритму			LSESolver.hpp
32.	SLEInputData	ClearData	Скинути стан об'єкту даних СЛАР			
33.	SLEInputData	SetEquationsCount	Встановити кількість рівнянь	eqsCount		
34.	SLEInputData	IsEquationsCountSet	Перевірити, чи встановлено кількість рівнянь			
35.	SLEInputData	SetVariable	Встановити	a		

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
	ta	sCoefficients	матрицю коефіцієнтів			
36.	SLEInputData	SetFreeCoefficients	Встановити вільні коефіцієнти	b		
37.	SLEInputData	GetVariablesCoefficients	Отримати посилання на матрицю коефіцієнтів			
38.	SLEInputData	GetFreeCoefficients	Отримати посилання на вільні коефіцієнти			
39.	SLEInputData	ConfirmData	Підтвердити дані СЛАР			
40.	SLEInputData	IsDataConfirmed	Перевірити, чи є слар підтверджено ю			
41.	SLESolveData	SetVarsSolve	Встановити розв'язок СЛАР	x		

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
42.	SLESolveData	GetSolveRef	Отримати посилання на розв'язок СЛАР			
43.	SLESolveData	SetSolvingStatus	Встановити статус вирішення СЛАР	sleSolvingStatus		
44.	SLESolveData	GetSolvingStatus	Отримати статус вирішення СЛАР			
45.	GUISession	GUISession	Створити об'єкт типу сесії графічного інтерфейсу			GUI.hpp
46.	GUISession	Init	Ініціалізувати вікно графічної сесії аргументами командного			GUI.hpp

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
			рядка			
47.	GUISession	GetWindow	Отримати посилання на вікно програми			GUI.hpp
48.	Application Data	GetSLEInputData	Отримати посилання на вхідні дані СЛАР			GUI.hpp
49.	Application Data	GetSLEOutputData	Отримати посилання на дані розв'язку СЛАР			GUI.hpp
50.	Application Window	SetApplicationData	Встановити посилання на об'єкт, що зберігає дані програми	appData		GUI.hpp
51.	Application Window	Ready	Встановити стан вікна як готового для опрацювання			GUI.hpp

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
52.	SLEConfigu rator	SetLSEInp utData	Встановити місце для запису конфігурації СЛАР	sleInputData		GUI.hpp
53.	SLESolveS hower	SetSLEOut putData	Встановити місце для зчитання даних СЛАР	sleSolveData		GUI.hpp
54.	SLESolveS hower	SetSLEInp utData	Встановити місце для зчитання даних про розв'язок СЛАР	sleInputData		GUI.hpp
55.	SLESolveS hower	OutputSolv e	Показати розв'язки СЛАР			GUI.hpp
56.	SLESolveS hower	ClearSolve	Приховати розв'язки СЛАР			GUI.hpp
57.	SLESolvePa nel	SetSLEInp utData	Встановити місце для	sleInputData		GUI.hpp

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
			зчитання даних СЛАР			
58.	SLESolvePanel	SetSLESolveData	Встановити місце для запису даних про розв'язок СЛАР	sleSolveData		GUI.hpp
59.	SLESolvePanel	SetSLESolveOutput	Встановити віджет для відображення розв'язків СЛАР	sleSolveOutput		GUI.hpp

4.2.2 Стандартні методи

№ п/п	Назва класу	Назва методу	Призначення методу	Опис вхідних параметрів	Опис вихідних параметр ів	Заголово чний файл
1.	std::vector<T>	vector	Конструктор порожнього динамічного			vector

			масиву			
2.	<code>std::vector<T></code>	<code>vector</code>	Конструктор динамічного масиву з size стандартних елементів	<code>size</code>		<code>vector</code>
3.	<code>std::vector<T></code>	<code>operator[]</code>	Отримати посилання на елемент	<code>index</code>	<code>T&</code>	<code>vector</code>
4.	<code>std::vector<T></code>	<code>size</code>	Отримати довжину		<code>std::size_t</code>	<code>vector</code>
5.	<code>std::istringstream</code>	<code>istringstream</code>	Новий рядковий вхідний потік з вхідних рядком <code>__str</code>	<code>__str</code>		<code>sstream</code>
6.	<code>std::istringstream</code>	<code>imbue</code>	Встановити локаль	<code>__loc</code>	<code>std::locale</code>	<code>sstream</code>
7.	<code>std::istringstream</code>	<code>operator>></code>	Вивести форматовані дані з потоку до змінної	<code>__f</code>	<code>std::istream</code>	<code>sstream</code>
8.	<code>std::istringstream</code>	<code>fail</code>	Перевірити, чи потік зазнав помилки		<code>bool</code>	<code>sstream</code>

9.	std::istringstream	eof	Перевірити, чи є потік у кінці файла		bool	sstream
10.	std::unique_ptr<T>	unique_ptr	Конструктор нового унікального розумного покажчика			memory
11.	std::unique_ptr<T>	reset	Перевстановити покажчик	__p		memory
12.	std::unique_ptr<T>	operator *	Отримати посилання на об'єкт покажчика		T&	memory
13.	std::shared_ptr<T>	shared_ptr	Конструктор нового спільного покажчика			memory
14.	std::shared_ptr<T>	operator *	Отримати посилання на об'єкт покажчика		T&	memory
15.	std::weak_ptr<T>	weak_ptr	Конструктор нового слабкого покажчика			memory
16.	std::weak_ptr	lock	Отримати		std::share	memory

	<code>tr<T></code>		новий спільний показчик, який посилається так же, як і слабкий показчик		<code>d_ptr<T></code>	
17.	<code>std::optional <T></code>	<code>optional</code>	Конструктор нового опціональног о значення, яке буде сконструйова не пустим конструкторо м			<code>optional</code>
18.	<code>std::optional <T></code>	<code>optional</code>	Конструктор нового опціональног о значення, яке буде тримати значення об'єкта <code>__t</code>	<code>__t</code>		<code>optional</code>
19.	<code>std::optional <T></code>	<code>optional</code>	Конструктор нового опціональног	<code><<немає назви>></code>		<code>optional</code>

			о значення, яке не буде мати об'єкта, буде зберігати статус відсутності об'єкта			
20.	std::optional <T>	value	Отримати посилання на об'єкт опціонального об'єкта		T&	optional
21.	std::optional <T>	has_value	Перевірити, чи має опціональний об'єкт об'єкт		bool	optional
22.	Gtk::Window	Window	Конструктор вікна			gtkmm.h
23.	Gtk::Window	set_title	Встановити підпис до вікна	title		gtkmm.h
24.	Gtk::Window	set_default _size	Встановити мінімальний розмір	width, height		gtkmm.h
25.	Gtk::Window	set_resizable	Перемкнути можливість змінювати	resizable		gtkmm.h

			розмір вікна			
26.	Gtk::Window	set_border_width	Встановити внутрішній відступ від рамки всіх внутрішніх віджетів вікна	border_width		gtkmm.h
27.	Gtk::Window	set_sensitive	Перемкнути можливість взаємодії з внутрішніми елементами вікна	sensitive		gtkmm.h
28.	Gtk::Frame	Frame	Конструктор іменованої рамки			gtkmm.h
29.	Gtk::Frame	set_label	Встановити видиму назву рамки	label		gtkmm.h
30.	Gtk::Frame	set_size_request	Встановити мінімальний розмір	width, height		gtkmm.h
31.	Gtk::Frame	add	Додати єдиний елемент як дитячий віджет	widget		gtkmm.h

32.	Gtk::Frame	set_border_width	Встановити внутрішній відступ від рамки всіх внутрішніх віджетів	border_width		gtkmm.h
33.	Gtk::Frame	show	Показати даний віджет			gtkmm.h
34.	Gtk::Frame	show_all_children	Показати всі віджети цього віджета			gtkmm.h
35.	Gtk::Fixed	Fixed	Конструктор поля для встановки віджетів по координатам			gtkmm.h
36.	Gtk::Fixed	put	Додати віджет по вказаним координатам відносно даного віджета	__v		gtkmm.h
37.	Gtk::Fixed	show	Показати даний віджет			gtkmm.h
38.	Gtk::Box	Box	Конструктор поля для			gtkmm.h

			ВСТАНОВКИ віджетів один за одним			
39.	Gtk::Box	set_orientat ion	Встановити орієнтацію списка віджетів	orientation		gtkmm.h
40.	Gtk::Box	set_spacing	Встановити відстань між віджетами	spacing		gtkmm.h
41.	Gtk::Box	pack_start	Додати до списку новий віджет	column		gtkmm.h
42.	Gtk::Box	show	Показати даний віджет	child		gtkmm.h
43.	Gtk::Box	show_all_c hildren	Показати всі віджети цього віджета			gtkmm.h
44.	Gtk::Grid	Grid	Конструктор сітки для відображення віджетів			gtkmm.h
45.	Gtk::Grid	attach	Додати новий віджет до сітки по координатам з вказаним	child, left, top, width, height		gtkmm.h

			розміром в клітинках			
46.	Gtk::Grid	remove_column	Прибрати один рядок	position		gtkmm.h
47.	Gtk::Grid	hide	Сховати даний віджет			gtkmm.h
48.	Gtk::Grid	show	Показати даний віджет			gtkmm.h
49.	Gtk::Grid	show_all_children	Показати всі віджети цього віджета			gtkmm.h
50.	Gtk::Label	Label	Конструктор віджета відображення тексту			gtkmm.h
51.	Gtk::Label	Label	Конструктор віджета відображення тексту з встановленим текстом	text		gtkmm.h
52.	Gtk::Label	set_text	Встановити текст для відображення	str		gtkmm.h
53.	Gtk::Label	set_width_chars	Встановити розмір віджета у	n_chars		gtkmm.h

			символах			
54.	Gtk::Label	set_size_request	Встановити мінімальний розмір	width, height		gtkmm.h
55.	Gtk::Entry	Entry	Конструктор поля вводу			gtkmm.h
56.	Gtk::Entry	set_placeholder_text	Встановити текст з замовчування	text		gtkmm.h
57.	Gtk::Entry	get_text	Отримати введений текст		Glib::ustring	gtkmm.h
58.	Gtk::Entry	set_width_chars	Встановити ширину поля у символах	n_chars		gtkmm.h
59.	Gtk::Entry	set_max_length	Встановити максимальну довжину введеного тексту	max		gtkmm.h
60.	Gtk::Entry	set_size_request	Встановити мінімальний розмір	width, height		gtkmm.h
61.	Gtk::Button	Button	Конструктор кнопки			gtkmm.h
62.	Gtk::Button	Button	Конструктор	text		gtkmm.h

			кнопки з встановленим текстом			
63.	Gtk::Button	signal_clicked	Встановити логіку для події на натискання	slot_	Glib::Sig nalProxy <void>	gtkmm.h
64.	Gtk::Combo Box	ComboBox	Конструктор поля зі списком			gtkmm.h
65.	Gtk::Combo Box	append	Додати новий пункт до списку	id, text		gtkmm.h
66.	Gtk::Drawin gArea	DrawingAr ea	Конструктор холста для програмного малювання			gtkmm.h
67.	Gtk::Drawin gArea	set_size_re quest	Встановити мінімальний розмір	width, height		gtkmm.h
68.	Gtk::Drawin gArea	signal_dra w	Встановити логіку для події на малювання	slot_	Glib::Sig nalProxy <void>	gtkmm.h
69.	Gtk::Drawin gArea	queue_dra w	Принудово відправити об'єкт до			gtkmm.h

			черги для відображення			
70.	Gtk::Allocation	Allocation	Конструктор об'єкта, що відображає геометричні параметри віджета			gtkmm.h
71.	Gtk::Allocation	get_width	Отримати реальну ширину		int	gtkmm.h
72.	Gtk::Allocation	get_height	Отримати реальну висоту		int	gtkmm.h
73.	Cairo::Context	Context	Конструктор об'єкта, який є інтерфейсом для малювання та отримання даних Gtk::Drawing Area			gtkmm.h
74.	Cairo::Context	get_allocation	Отримати об'єкт типу Gtk::Allocation		Gtk::Allocation	gtkmm.h

			n			
75.	Cairo::Cont ext	set_source_ rgb	Встановити колір для малювання	red, green, blue		gtkmm.h
76.	Cairo::Cont ext	rectangle	Намалювати прямокутник	x, y, iwidth, height		gtkmm.h
77.	Cairo::Cont ext	fill	Заповнити весь холст один кольором			gtkmm.h
78.	Cairo::Cont ext	stroke	Очистити буфер координат			gtkmm.h
79.	Cairo::Cont ext	move_to	Додати нову координату до буфера ліній	x, y		gtkmm.h
80.	Cairo::Cont ext	line_to	Намалювати лінію з останньої встановленої координати	x, y		gtkmm.h
81.	Cairo::Cont ext	set_font_si ze	Встановити розмір шрифту	size		gtkmm.h
82.	Cairo::Cont ext	show_text	Намалювати текст по раніше	utf8		gtkmm.h

			встановленим параметрам			
--	--	--	----------------------------	--	--	--

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 з цим 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 введенні 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 з довільними коефіцієнтами:
 - 1) Перевірка коректності роботи методу 1.
 - 2) Перевірка коректності роботи методу 2.
 - 3) Перевірка коректності роботи методу 3.
- г) Тестування побудови графіків.

5.2 Приклади тестування

Таблиця 5.1 — Приклад роботи програми при введенні некоректних символів.

Мета тесту	Перевірити можливість введення некоректних даних
Початковий стан програми	Відкрите вікно програми та кількість рівнянь (3) встановлена
Вхідні дані	1 2 3 8; 4 с 6 5; 7 8 9 100
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів та вектора вільних членів
Очікуваний результат	Повідомлення про помилку формату даних
Стан програми після проведення випробувань	Видано помилку «Введіть дійсне число»

Таблиця 5.2 — Приклад роботи програми при введенні замалих та завеликих значень.

Мета тесту	Перевірити можливість введення занадто великих або замалих чисел
Початковий стан програми	Відкрите вікно програми та кількість рівнянь (2) встановлена
Вхідні дані	100 300 990; -670 480 30000
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів та вектора вільних членів
Очікуваний результат	Повідомлення про помилку діапазону значень даних

Мета тесту	Перевірити можливість введення занадто великих або замалих чисел
Стан програми після проведення випробувань	Видано помилку «Комірка B[рівняння №2] не є в діапазоні [-10'000; 10'000]»

Таблиця 5.3 — Приклад роботи програми при нульовому значенні визначника.

Мета тесту	Перевірити можливість розв'язку СЛАР з нульовим визначником
Початковий стан програми	Відкрите вікно програми та кількість рівнянь (3) встановлена, матриця коефіцієнтів та вектор вільних членів встановлені
Вхідні дані	1 3 2 5; 4 8 9 9; 0 0 0 1
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів та вектора вільних членів
Очікуваний результат	Повідомлення про помилку вирішення СЛАР
Стан програми після проведення випробувань	Видано помилку «СЛАР неможливо вирішити цим методом»

Таблиця 5.4 — Приклад роботи методу 1 на несиметричній матриці.

Мета тесту	Перевірити можливість вирішення СЛАР з несиметричною матрицею коефіцієнтів
Початковий стан програми	Відкрите вікно програми та кількість рівнянь (3) встановлена, матриця коефіцієнтів та вектор вільних членів

Мета тесту	Перевірити можливість вирішення СЛАР з несиметричною матрицею коефіцієнтів
	встановлені
Вхідні дані	1 6 2 1; 5 9 7 2; 4 8 3 3
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів та вектора вільних членів
Очікуваний результат	Повідомлення про розв'язок СЛАР
Стан програми після проведення випробувань	Видано розв'язок «X1: - 0.1278195488721804; X2: 0.388926862110731; X3: - 012303485987696514»

Таблиця 5.5 — Приклад роботи методу 1 на не додатньо визначеній матриці.

Мета тесту	Перевірити можливість роботи методу 1 з не додатньо визначеною матрицею коефіцієнтів
Початковий стан програми	Відкрите вікно програми та кількість рівнянь (3) встановлена, матриця коефіцієнтів та вектор вільних членів встановлені
Вхідні дані	-1 -2 3 10; -4 5 6 40; 7 -8 -9 30
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів та вектора вільних членів
Очікуваний результат	Повідомлення про розв'язок СЛАР
Стан програми після проведення випробувань	Видано розв'язок «X1: 102.50000000000004; X2: 25.000000000000014; X3:

Мета тесту	Перевірити можливість роботи методу 1 з не додатньо визначеною матрицею коефіцієнтів
	54.16666666666668»

Таблиця 5.6 — Приклад перевірки коректності роботи методу 1.

Мета тесту	Перевірити коректність роботи методу 1 на довільних даних
Початковий стан програми	Відкрите вікно програми та кількість рівнянь (3) встановлена, матриця коефіцієнтів та вектор вільних членів встановлені, метод вирішення встановлений
Вхідні дані	146 136 102 3602; 136 155 100 3525; 102 100 114 3258
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів та вектора вільних членів
Очікуваний результат	Повідомлення про розв'язок «X1: 10; X2: 3; X3: 17»
Стан програми після проведення випробувань	Видано розв'язок «X1: 10; X2: 3; X3: 17»

Таблиця 5.7 — перевірки коректності роботи методу 2.

Мета тесту	Перевірити коректність роботи методу 2 на довільних даних
Початковий стан програми	Відкрите вікно програми та кількість рівнянь (3) встановлена, матриця коефіцієнтів та вектор вільних членів

Мета тесту	Перевірити коректність роботи методу 2 на довільних даних
	встановлені, метод вирішення встановлений
Вхідні дані	
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів та вектора вільних членів
Очікуваний результат	Повідомлення про розв'язок СЛАР «X1: 10; X2: 3; X3: 17»
Стан програми після проведення випробувань	Видано розв'язок «X1: 9.999999999999998; X2: 3.00000000000000027; X3: 16.999999999999996»

Таблиця 5.8 — перевірки коректності роботи методу 3.

Мета тесту	Перевірити коректність роботи методу 3 на довільних даних
Початковий стан програми	Відкрите вікно програми та кількість рівнянь (3) встановлена, матриця коефіцієнтів та вектор вільних членів встановлені, метод вирішення встановлений
Вхідні дані	8 7 5 138; 9 3 2 181; 6 4 1 224
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів та вектора вільних членів
Очікуваний результат	Повідомлення про розв'язок «X1: 10; X2: 3; X3: 17»
Стан програми після проведення	Видано розв'язок «X1: 9.999999999999996; X2:

Мета тесту	Перевірити коректність роботи методу 3 на довільних даних
випробувань	2.9999999999999996; X3: 17.0000000000000004»

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

6.1 Робота з програмою

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

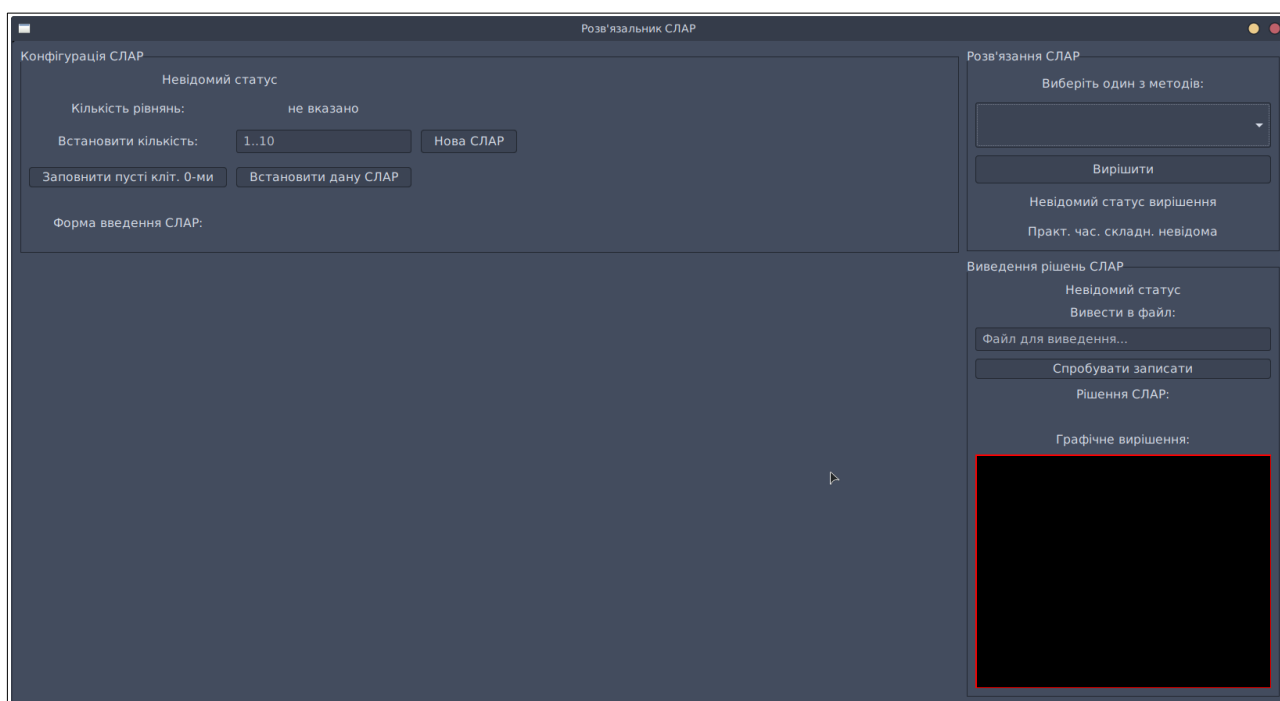


Рисунок 6.1 — Головне вікно програми

Далі треба встановити розмірність системи лінійних алгебраїчних рівнянь (далі — СЛАР) біля таблички «Встановити кількість», потім натиснути на клавішу «Нова СЛАР» 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).

Конфігурація СЛАР

Встановлено кількість рівнянь

Кількість рівнянь: 2

Встановити кількість: 1..10 Нова СЛАР

Заповнити пусті кліт. 0-ми Встановити дану СЛАР

Форма введення СЛАР:

a1,1	X1 +	a1,2	X2 =	b1
a2,1	X1 +	a2,2	X2 =	b2

Рисунок 6.3 — Форма введення СЛАР

Потім треба натиснути на клавішу «Встановити дану СЛАР», 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 неможливість 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 «Розв'язання СЛАР».

Розв'язання СЛАР

Виберіть один з методів:

LUP-метод
Складність: $\frac{1}{3}n^3 + \frac{9}{2}n^2 + \frac{19}{6}n$

Рисунок 6.4 — Список вибору методу розв'язання СЛАР

Потім треба натиснути на клавішу «Вирішити» для запуску 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, якщо кількість рівнянь рівна 2, а також у вигляді таблиці (рисунок 6.5), буде можливість вивести до введеного файлу.

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

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

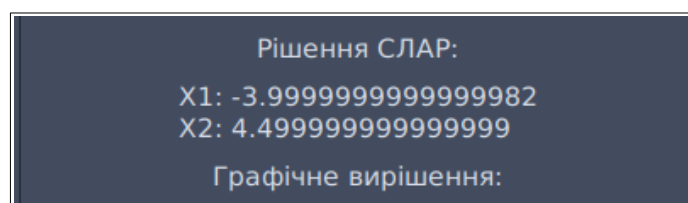


Рисунок 6.5 — Табличний метод відображення розв'язків СЛАР

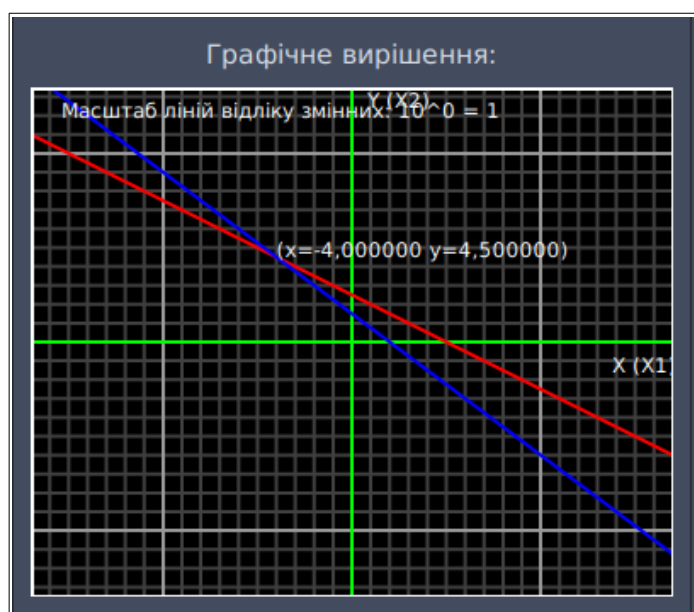


Рисунок 6.6 — Графічне вирішення СЛАР

6.2 Формат вхідних даних

Користувачу на вхід програми подається СЛАР у матричному вигляді, 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 стовпця 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 оновленнями) з ядром «6.9.1-arch1-1»	ArchLinux (з останніми оновленнями) з ядром «6.9.1-zen1-1-zen»
Процесор	Intel i7-6700HQ (8) @ 3.500GHz	
Оперативна пам'ять	8 GB RAM	16 GB RAM
Відеоадаптер	Intel HD Graphics 530	
Дисплей	1920x1080	1920x1080 або вище
Прилади введення	Клавіатура, комп'ютерна миша	

Додаткове програмне забезпечення	Пакети «base», «base-devel», «gtkmm3»

7 АНАЛІЗ РЕЗУЛЬТАТІВ

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:

The screenshot displays a software interface for solving a system of linear equations (СЛАР) using the LUP method. It is divided into two main panels.

Left Panel: Форма введення СЛАР (SLAR Input Form)

	x_1	x_2	x_3	=	
3	1	1	1		1
1	3	1	1		2
1	1	3	1		3

Right Panel: SLAR Results

СЛАР вирішено за 59 ітерацій

Виведення рішень СЛАР

Рішення виведено 13:23:39 UTC+00:00

Вивести в файл:

Файл для виведення...

Спробувати записати

Рішення СЛАР:

x_1 : -0.10000000000000003
 x_2 : 0.4
 x_3 : 0.9

Рисунок 7.1 – Результат виконання LUP-методу

Оскільки результат виконання (7.1) збігається з результатом в LibreOffice Calc (рисунок 7.2), то даний метод працює вірно.

fx Σ ▾ = =B3*\$F\$3+C3*\$F\$4+D3*\$F\$5										
	A	B	C	D	E	F	G	H	I	
		An1	An2	An3		X		New	Orig.	
		3	1	1		-0.1		1	1	
		1	3	1		0.4		2	2	
		1	1	3		0.9		3	3	

Рисунок 7.2 – Перевірка методу Якобі в LibreOffice Calc

б) Метод Гауса-Холецького.

Результат виконання методу Гауса-Холецького наведено на рисунку 7.2:

Форма введення СЛАР:					СЛАР вирішено за 34 ітерацій	
11	x1 +	7	x2 +	7	x3 =	10
7	x1 +	11	x2 +	7	x3 =	30
7	x1 +	7	x2 +	11	x3 =	90
<div>Виведення рішень СЛАР</div> <div>Рішення виведено 13:32:39 UTC+00:00</div> <div>Вивести в файл:</div> <div>Файл для виведення...</div> <div>Спробувати записати</div> <div>Рішення СЛАР:</div> <div>X1: -6.599999999999999</div> <div>X2: -1.60000000000000014</div> <div>X3: 13.4</div>						

Рисунок 7.3 – Результат виконання методу Гауса-Холецького

Оскільки результат виконання (7.3) збігається з результатом в LibreOffice Calc (рисунок 7.4), 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:

$$R = \begin{pmatrix} n & 1 & 1 & \cdots & 1 \\ 1 & n & 1 & \cdots & 1 \\ 1 & 1 & n & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & n \end{pmatrix}, \quad A = R^T R = \begin{pmatrix} n^2+2 & 2n+1 & 2n+1 & \cdots & 2n+1 \\ 2n+1 & n^2+2 & 2n+1 & \cdots & 2n+1 \\ 2n+1 & 2n+1 & n^2+2 & \cdots & 2n+1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 2n+1 & 2n+1 & 2n+1 & \cdots & n^2+2 \end{pmatrix} \quad (7.1)$$

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

Візуалізація результатів таблиці 7.1 наведено на рисунку 7.1:

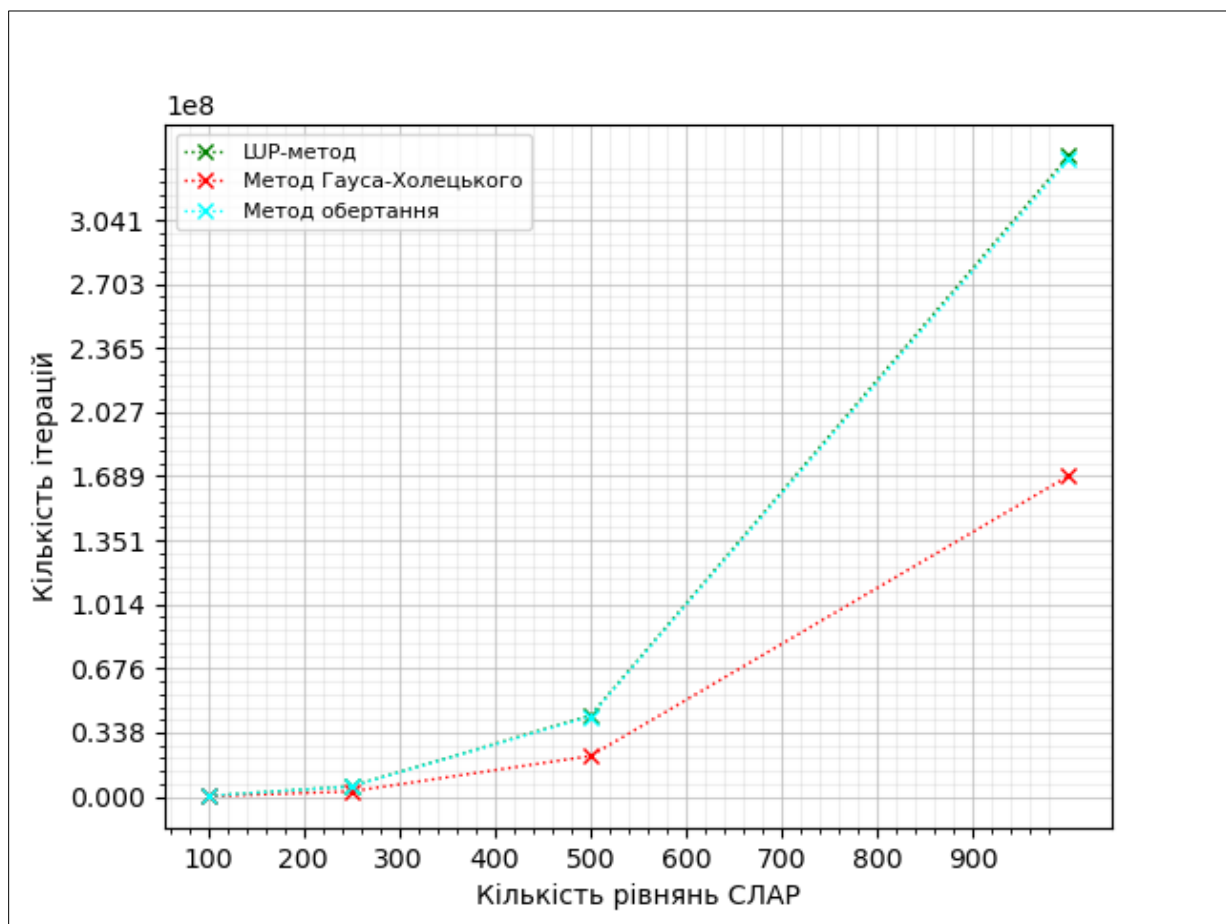


Рисунок 7.1 – Графік залежності кількості ітерацій методу від розміру вхідної системи

За результатами тестування можна зробити такі висновки:

- Всі розглянуті методи дозволяють знаходити розв'язки великих та надвеликих СЛАР.
- Складність всіх розглянутих методів є кубічною, тобто — $O(n^3)$, де n — кількість рівнянь СЛАР.
- З розглянутих методів найоптимальнішим для специфічних використання є метод Гауса-Холецького. Найбільш оптимальнішим для широкого

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) Метод обертання є найбільш точним з точки зору отриманих розв'язків введених СЛАР.
- 2) З точки зору кількості ітерацій, метод Гауса-Холецького є найбільш ефективним.

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-Холеского (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 15.05.2024 р.)
- 2) Об'єктно-орієнтоване проектування архітектури програмної системи (до 24.05.2024 р.)
- 3) Розробка програмного забезпечення (до 28.05.2024 р.)
- 4) Тестування розробленої програми (до 26.05.2024 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., байти)

студента групи ІІІ-35 І курсу

Адаменко А.Б.

<https://github.com/adamenko-arsen/SLE-Accurate-Solver> (МОЖЛИВО
ТРЕБА СТВОРИТИ НОВИЙ РЕПОЗИТОРІЙ АБО ВІДКРИТИ СТАРИЙ)