

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії

КУРСОВА РОБОТА

з Основи програмування 2. Модульне програмування
(назва дисципліни)

на тему: Розв'язання СЛАР точними методами

Студента 1 курсу, групи ІП-15
Рибалки Іллі Сергійовича

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник старший викладач Головченко Максим Миколайович
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: _____

Національна оцінка _____

Члени комісії

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

(вчене звання, науковий ступінь, прізвище та ініціали)

Київ- 2022 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-15

Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Рибалки Іллі Сергійовича

(прізвище, ім'я, по батькові)

1. Тема роботи _____

2. Строк здачі студентом закінченої роботи _____

3. Вихідні дані до роботи _____

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

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

6. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

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

Студент _____
(підпис)

Керівник _____
(підпис)

Головченко М. М.
(прізвище, ім'я, по батькові)

"__" _____ 20__ р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 66 сторінок, 16 рисунки, 17 таблиць, 3 посилання.

Об'єкт дослідження: розв'язок СЛАР точними методами.

Мета роботи: дослідження методів розв'язання СЛАР, створення програмного забезпечення для розв'язку СЛАР трьома методами.

Вивчено метод розробки програмного забезпечення з використанням принципів ООП. Приведені змістовні постановки задач, їх індивідуальні математичні моделі, а також описано детальний процес розв'язання кожної з них.

Виконана програмна реалізація розв'язку СЛАР точними методами.

СИСТЕМА ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ, МЕТОД КРАМЕРА, МЕТОД ГАУСА З ОДИНИЧНОЇ ДІАГОНАЛЛЮ, МЕТОД ГАУСА З ВИБОРОМ ГОЛОВНОГО ЕЛЕМЕНТУ.

	4
ЗМІСТ	
ЗАВДАННЯ	2
АНОТАЦІЯ	4
ВСТУП	5
1 ПОСТАНОВКА ЗАДАЧІ	6
2 ТЕОРЕТИЧНІ ВІДОМОСТІ	7
2.1 Метод Крамера	7
2.2 Метод Гауса з одиничною діагоналлю	7
2.3 Метод Гауса з вибором головного елементу	8
3 ОПИС АЛГОРИТМІВ	11
3.1 Загальний алгоритм	11
3.2 Алгоритм методу Крамера	13
3.3 Алгоритм методу Гауса з одиничною діагоналлю	13
3.4 Алгоритм методу Гауса з вибором головного елементу	14
4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	16
4.1 Діаграма класів програмного забезпечення	16
4.2 Опис методів частин програмного забезпечення	17
4.2.1 Користувацькі методи	17
4.2.2 Стандартні методи	20
5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	23
5.1 План тестування	23
5.2 Приклади тестування	25
6 ІНСТРУКЦІЯ КОРИСТУВАЧА	34
6.1 Робота з програмою	34
6.2 Формат вхідних та вихідних даних	37
6.3 Системні вимоги	38
7 АНАЛІЗ РЕЗУЛЬТАТІВ	39
ВИСНОВКИ	44
ПЕРЕЛІК ПОСИЛАНЬ	45
ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ	46
ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ	49

ВСТУП

Метою розробки є фіксація практичних навичок з курсу основи програмування. Ідеєю даної роботи є створення програмного забезпечення для розв'язку систем лінійних алгебраїчних рівнянь з використанням парадигм ООП. В ході розробки будуть вирішені наступні задачі: надати можливість користувачу розв'язувати СЛАР трьома методами, за вибором, а саме: Крамера, Гауса з одиничною діагоналлю і Гауса з вибором головного елементу; виводити графічний і текстовий розв'язок системи. Дане програмне забезпечення має полегшити процес розв'язку СЛАР.

1 ПОСТАНОВКА ЗАДАЧІ

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

- а) метод Крамера;
- б) метод Гауса з одиничною діагоналлю;
- в) метод Гауса з вибором головного елементу;

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 2 до 7.

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 не перевищує двох невідомих, 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

Квадратичну систему з n лінійних рівнянь можна задати наступним чином:

$$Ax = b \quad (2.1)$$

де:

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

Тоді якщо $|A| \neq 0$, то система (2.1) має розв'язок та він єдиний. Якщо система має єдиний розв'язок, то його можна знайти одним із наступних методів.

2.1 Метод Крамера

Сутність методу Крамера полягає в тому, щоб отримати розв'язок через визначники квадратної матриці коефіцієнтів, отриманих шляхом заміни одного стовпця матриці коефіцієнтів вектор-стовпцем правої частини рівняння. Тобто, нехай в нас є система:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \quad (2.2)$$

Для даної системи (2.2) обчислюємо визначник:

$$\Delta = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \cdot a_{22} \cdot a_{33} + a_{12} \cdot a_{23} \cdot a_{31} + a_{21} \cdot a_{32} \cdot a_{13} - a_{13} \cdot a_{22} \cdot a_{31} - a_{12} \cdot a_{21} \cdot a_{33} - a_{23} \cdot a_{32} \cdot a_{11};$$

Аналогічним чином обчислюємо допоміжні визначники:

$$\Delta_1 = \begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}, \quad \Delta_2 = \begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}, \quad \Delta_3 = \begin{vmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ a_{31} & a_{32} & b_3 \end{vmatrix},$$

Використовуючи формули Крамера, знаходимо рішення системи:

$$x_1 = \frac{\Delta_1}{\Delta}, \quad x_2 = \frac{\Delta_2}{\Delta}, \quad x_3 = \frac{\Delta_3}{\Delta}$$

2.2 Метод Гауса з одиничною діагоналлю

Сутність методу Гауса полягає у приведенні матриці коефіцієнтів до одиничної діагоналі. Тобто:

$$\begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \ddots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix},$$

Прямий хід:

Припустимо що коефіцієнт матриці(2.1) $a_{11} \neq 0$, тоді поділивши коефіцієнти першого рівняння системи на a_{11} отримаємо:

$$x_1 + c_{12}x_2 + c_{13}x_3 + \dots + c_{1n}x_n = d_1 \quad (2.3)$$

$$\text{де } c_{1j} = \frac{a_{1j}}{a_{11}} (j > 1), d_1 = \frac{b_1}{a_{11}}.$$

Користуючись цим рівнянням(2.3), легко виключити з 2, 3, 4, ..., n рівняння системи(2.1) невідому x_1 . Для цього достатньо від другого рівняння системи відняти отримане рівняння, помножене на a_{21} ; від третього рівняння системи, відняти отримане рівняння, помножене на a_{31} , і так далі.

Таким чином, отримаємо систему трикутної форми:

$$\begin{cases} x_1 + c_{12}x_2 + c_{13}x_3 + \dots + c_{1n}x_n = d_1 \\ \quad x_2 + c_{23}x_3 + \dots + c_{2n}x_n = d_2 \\ \quad \quad x_3 + \dots + c_{3n}x_n = d_3 \\ \quad \quad \quad \dots \dots \dots \dots \dots \dots \dots \dots \dots \\ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad x_n = d_n \end{cases} \quad (2.4)$$

Зворотній хід:

Оскільки ми звели матрицю(2.1) до трикутної форми, то $x_n = d_n$, користуючись цим легко виключити з $n - 1, n - 2, \dots, 1$ рівняння системи невідому x_{n-1n} . Для цього достатньо від $n - 1$ рівняння системи(2.4) відняти n рівняння, помножене на c_{n-1n} ; від $n - 2$ рівняння системи, відняти n рівняння, помножене на c_{n-2n} , і так далі.

Таким чином, отримаємо одиничну діагональ в матриці коефіцієнтів, а перетворений вектор вільних членів і буде рішенням системи.

2.3 Метод Гауса з вибором головного елемента

Сутність методу Гауса з вибором головного елемента полягає у тому, щоб за рахунок найбільшого за модулем елемента в кожному стовпці звести матрицю коефіцієнтів до трикутної форми. Тобто:

$$\begin{bmatrix} a_{p1} & a_{i2} & \dots & a_{qn} \\ 0 & a_{p2} & \dots & a_{jn} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{pn} \end{bmatrix},$$

Розглянемо розширену прямокутну матрицю, що складається з коефіцієнтів системи (2.1) та її вільних членів:

$$M = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1j} & \dots & a_{1q} & \dots & a_{1n} & a_{1n+1} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2j} & \dots & a_{2q} & \dots & a_{2n} & a_{2n+1} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3j} & \dots & a_{3q} & \dots & a_{3n} & a_{3n+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \dots & \vdots & \dots & \vdots & \vdots \\ a_{i1} & a_{i2} & a_{i3} & \dots & a_{ij} & \dots & a_{iq} & \dots & a_{in} & a_{in+1} \\ \vdots & \vdots & \vdots & \dots & \vdots & \ddots & \vdots & \dots & \vdots & \vdots \\ a_{p1} & a_{p2} & a_{p3} & \dots & a_{pj} & \dots & a_{pq} & \dots & a_{pn} & a_{pn+1} \\ \vdots & \vdots & \vdots & \dots & \vdots & \dots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nj} & \dots & a_{nq} & \dots & a_{nn} & a_{nn+1} \end{bmatrix} \quad (2.5)$$

де:

$$a_{1n+1} = b_1, a_{2n+1} = b_2, a_{3n+1} = b_3 \dots, a_{nn+1} = b_n$$

Для даної матриці(2.5), згідно з алгоритмом методу Гауса з вибором головного елемента, виберемо ненульовий, як правило, найбільший за модулем елемент, який не належить стовпцю вільних членів, тобто $q \neq n + 1$, нехай це буде елемент a_{pq} .

Далі, для кожного рядка матриці (2.5), крім рядка під номером p , обчислюємо множники:

$$m_i = -\frac{a_{iq}}{a_{pq}}, (i \neq p)$$

На наступному кроці виконуємо наступні дії: до кожного неголовного рядка розширеної матриці (2.5) додамо головний рядок (p рядок матриці M) помножений на відповідний для нього множник m_i :

$$a_{ij} = a_{ij} + a_{pj} \cdot m_i, (i \neq p)$$

В результаті отримаємо нову матрицю, q -й стовпець якої складається з нульових елементів.

Викреслюючи даний стовпець і p -й (головний) рядок, отримаємо матрицю M_1 , яка складається з меншого на одиницю числа рядків та стовпців.

Над матрицею M_1 повторюємо тіж операції, після чого отримаємо деяку матрицю M_2 . Продовжуючи обчислювальний процес далі, отримуємо послідовність матриць $M, M_1, M_2, \dots, M_{n-1}$, остання з яких є матрицею-рядком, яка складається з двох елементів.

Для визначення невідомих x_i , об'єднуємо в систему, починаючи з останнього, який входить в матрицю M_{n-1} , всі головні рядки. Далі провівши відповідну перестановку рядків, отримаємо систему з трикутною матрицею, за допомогою якої, знаходимо розв'язок системи рівнянь (2.1).

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
MatrixA	Матриця коефіцієнтів
LstB	Стовпець вільних членів
n	Розмірність матриці(Довжина стовпця вільних членів)
d, dt	Визначник матриці
MatrixAcopy	Копія матриці коефіцієнтів (необхідна для заміни стовпців на елементи стовпця вільних членів в методі Крамера)
X, res	Розв'язок СЛАР
max_element	Максимальний елемент(використовується в методі Гауса з вибором головного елемента)
max_col	Індекс рядка максимального елемента в стовпці (використовується в методі Гауса з вибором головного елемента)
div	Множник (використовується в методі Гауса з вибором головного елемента)

3.1 Загальний алгоритм

1. ПОЧАТОК
2. Зчитати метод розв'язання
3. Зчитати розмірність системи
4. Зчитати матрицю коефіцієнтів і стовпець вільних членів:
 - 4.1. Зчитати матрицю системи:
 - 4.1.1. Цикл проходу по всіх рядках матриці системи (a_i – поточна строка):
 - 4.1.1.1. Цикл проходу всіх стовпцях матриці системи (a_{ij} – поточний елемент):
 - 4.1.1.1.1. ЯКЩО поточний елемент матриці – вірно записане число, ТО записати його в відповідну

комірку *MatrixA*. ІНАКШЕ видати повідомлення про помилку та перейти до пункту 4.

4.2. Зчитати стовпець вільних членів:

4.2.1. Цикл проходу по всіх елементах стовпця вільного членів:

4.2.1.1. Якщо поточний елемент стовпцю вільних членів – вірно записане число, ТО записати його в відповідну комірку *LstB*. ІНАКШЕ видати повідомлення про помилку та перейти до пункту 4.

5. ЯКЩО визначник *MatrixA* не рівний нулю, ТО:

5.1. ЯКЩО обраний метод Гауса з одиничною діагоналлю, ТО:

5.1.1. ЯКЩО *MatrixA* сумісна, ТО:

5.1.1.1. Обробити дані згідно алгоритму методу Гауса з одиничною діагоналлю (пункт 3.3).

5.1.1.2. ЯКЩО розмірність дорівнює двом, ТО побудувати та вивести графік системи.

5.1.1.3. Вивести рішення системи.

5.1.1.4. Записати систему та її рішення у файл.

ІНАКШЕ видати повідомлення про помилку та перейти до пункту 4.

5.2. ІНАКШЕ:

5.2.1. ЯКЩО обраний метод Крамера, ТО обробити дані згідно алгоритму методу Крамера (пункт 3.2).

5.2.2. ЯКЩО обраний метод Гауса з вибором головного елемента, ТО обробити дані згідно алгоритму методу Гауса з вибором головного елемента (пункт 3.4).

5.2.3. ЯКЩО розмірність дорівнює двом, ТО побудувати та вивести графік системи.

5.2.4. Вивести рішення системи.

5.2.5. Записати систему та її рішення у файл.

6. ІНАКШЕ видати повідомлення про помилку та перейти до пункту 4.

7. КІНЕЦЬ

3.2 Алгоритм методу Крамера

1. ПОЧАТОК
2. Обчислити визначник матриці коефіцієнтів (d).
3. Ініціалізація нової матриці *MatrixAcopy* значеннями матриці *MatrixA*.
4. ЦИКЛ проходу по всіх стовпцях матриці (*MatrixAcopy*):
 - 4.1. ЦИКЛ проходу по всіх рядках матриці (*MatrixAcopy*):
 - 4.1.1. Присвоїти значення $LstB[рядок]$ до поточного елементу *MatrixAcopy*
 - 4.2. Визначити визначник (dt) отриманої матриці (*MatrixAcopy*)
 - 4.3. Присвоїти $X[стовпець]$ значенням виразу dt/d
 - 4.4. Ініціалізація матриці *MatrixAcopy* значеннями матриці *MatrixA*.
5. КІНЕЦЬ

3.3 Алгоритм методу Гауса з одиничною діагоналлю

1. ПОЧАТОК
2. ЦИКЛ проходження по кожному рядку(i) матриці(*MatrixA*):
 - 2.1. Поділити $LstB[i]$ на $MatrixA[i][i]$
 - 2.2. ЦИКЛ проходження по кожному стовпцю(j) матриці(*MatrixA*) в зворотньому порядку більшого за значення рядку:
 - 2.2.1. Поділити $MatrixA[i][j]$ на $MatrixA[i][i]$
 - 2.3. ЦИКЛ проходження по кожному рядку(j) матриці(*MatrixA*) в зворотньому порядку меншого за значення рядку(i):
 - 2.3.1. Відняти від $LstB[j]$ значення виразу $MatrixA[j][i] * LstB[i]$
 - 2.3.2. ЦИКЛ по кожному стовпцю(j) матриці(*MatrixA*) в зворотньому порядку
 - 2.3.2.1. Відняти від $MatrixA[j][k]$ значення виразу $MatrixA[j][i] * MatrixA[i][k]$
3. ЦИКЛ проходження по кожному рядку(i) матриці(*MatrixA*) в зворотньому порядку:

3.1. ЦИКЛ проходження по індексу кожного елементу(j) стовпця вищого за рядок матриці($MatrixA$):

3.1.1. Відняти від $LstB[j]$ значення виразу $LstB[i] * MatrixA[j][i]$

3.1.2. Відняти від $MatrixA[j][i]$ значення виразу $MatrixA[j][рядок]$

4. Присвоїти результату значення $LstB$

5. КІНЕЦЬ

3.4 Алгоритм методу Гауса з вибором головного елементу

1. ПОЧАТОК

2. ЦИКЛ проходу по індексам кожного елменту(i) масиву($LstB$):

2.1. Додаємо $LstB[i]$ в кінець $MatrixA[i]$

3. ЦИКЛ проходу по стовпцях матриці не включаючи останній:

3.1. Присвоїти $max_element = MatrixA[стовпець][стовпець]$

3.2. Присвоїти $max_col = номер\ стовпця$

3.3. ЦИКЛ проходу по рядках матриці ($MatrixA$) нищих за номер стовпця:

3.3.1. ЯКЩО модуль $MatrixA[рядок][стовпець]$ більший за модуль $max_element$ ТО:

3.3.1.1. $max_element = MatrixA[рядок][стовпець]$

3.3.1.2. $max_col = номер\ рядку$

3.3.2. ЯКЩО max_col не дорівнює номеру стовпця ТО поміняти місцями $MatrixA[стовпець]$ і $MatrixA[max_col]$

3.4. ЦИКЛ проходу по рядках матриці ($MatrixA$) нищих за номер стовпця:

3.4.1. Присвоїти div значення ділення $MatrixA[рядок][стовпець]$ на $MatrixA[стовпець][стовпець]$

3.4.2. Відняти від $MatrixA[рядок][n]$ значення виразу $div * MatrixA[k][n]$

3.4.3. ЦИКЛ проходу стовпцях(j) більших за значення поточного стовпця матриці ($MatrixA$):

3.4.3.1. Відняти від $MatrixA[рядок][j]$ значення виразу div
 $* MatrixA[стовпець][j]$

4. ЦИКЛ проходу по рядках матриці ($MatrixA$) в зворотньому порядку:

4.1. Присвоїти $res[рядок]$ значення виразу:

4.1.1. Відняти від останнього елементу рядку матриці суму всіх елементів рядку (до діагонального елементу з індексами $[рядок][рядок]$) помножених на вже знайдені значення $res[індекс елемента відповідно]$

4.1.2. Поділити отриманий результат на діагональний елемент

5. КІНЕЦЬ

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Діаграма класів програмного забезпечення

Діаграма класів розробленого програмного забезпечення наведена на рисунку 4.1.

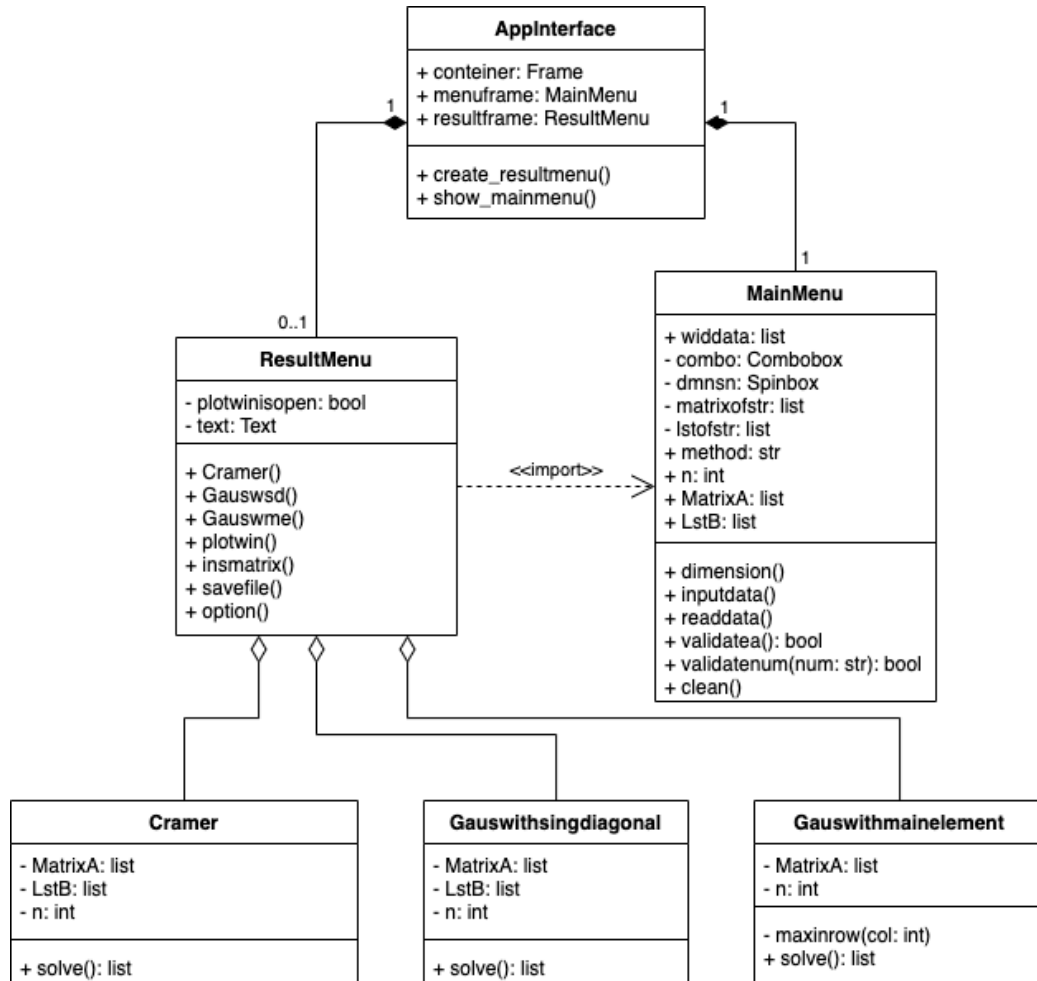


Рисунок 4.1 – Діаграма класів

4.2 Опис методів частин програмного забезпечення

4.2.1 Користувацькі методи

У таблиці 4.1 наведено користувацькі методи класів

Таблиця 4.1 – Користувацькі методи

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	AppInterface	__init__	Конструктор класу	Відсутні	Відсутні
2	AppInterface	create_result_menu	Створює об'єкт класу ResultMenu і відкриває відповідний фрейм.	Відсутні	Відсутні
3	AppInterface	show_main_menu	Відкриває головне меню.	Відсутні	Відсутні
4	MainMenu	__init__	Конструктор класу	Корінь(вікно), батьківський фрейм	Відсутні
5	MainMenu	dimension	Введення розмірності	Відсутні	Відсутні
6	MainMenu	inputdata	Введення матриці коефіцієнтів і стовпця вільних членів	Відсутні	Відсутні
7	MainMenu	readdata	Зчитування матриці коефіцієнтів і стовпця вільних членів у масив	Відсутні	Відсутні

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
8	MainMenu	validatea	Перевіряє матрицю коефіцієнтів на сумісність, у разі вибору методу Гауса з одиничною діагоналлю.	Відсутні	Булева змінна
9	MainMenu	validatenum	Перевіряє чи є введений рядок числом, і чи задовольняє воно умові	Строка	Булева змінна
10	MainMenu	clean	Очищає фрейм від зайвих віджетів	Відсутні	Відсутні
11	ResultMenu	__init__	Конструктор класу	Корінь(вікно), батьківський фрейм	Відсутні
12	ResultMenu	Cramer	Виведення розв'язку СЛАУ методом Крамера	Відсутні	Відсутні
13	ResultMenu	Gauswsd	Виведення розв'язку СЛАУ методом Гауса з одиничною діагоналлю	Відсутні	Відсутні
14	ResultMenu	Gauswme	Виведення розв'язку СЛАУ методом Гауса з вибором головного елемента	Відсутні	Відсутні

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
15	ResultMenu	Plotwin	Створення вікна графіка, і побудова його	Відсутні	Відсутні
16	ResultMenu	insmatrix	Виведення вхідної матриці коефіцієнтів і стовпця вільних членів	Відсутні	Відсутні
17	ResultMenu	savefile	Збереження файлу	Тип файлу (розв'язок, графік)	Відсутні
18	ResultMenu	option	Опція закриття вікна програми, переходу в головне меню, закриття вікна графіка	Тип опції	Відсутні
19	Cramer	__init__	Конструктор класу	Матриця коефіцієнтів і стовпець вільних членів	Відсутні
20	Cramer	solve	Розв'язок СЛАУ методом Крамера	Відсутні	Розв'язок СЛАУ
21	Gauswithsing diagonal	__init__	Конструктор класу	Матриця коефіцієнтів і стовпець вільних членів	Відсутні
22	Gauswithsing diagonal	solve	Розв'язок СЛАУ методом Гауса з одиначною діагоналлю	Відсутні	Розв'язок СЛАУ

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
23	Gauswithmainelement	__init__	Конструктор класу	Матриця коефіцієнтів і стовпець вільних членів	Відсутні
24	Gauswithmainelement	solve	Розв'язок СЛАУ методом Гауса з вибором головного елемента	Відсутні	Розв'язок СЛАУ
25	Gauswithmainelement	maxinrow	Пошук максимального за модулем елемента в стовпці і перестановка його рядка на головну діагональ	Номер стовпця	Відсутні

4.2.2 Стандартні методи

У таблиці 4.2 наведено стандартні методи класів

Таблиця 4.2 – Стандартні методи

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	copy	deepcopy	Копіювання масиву	Масив	Копія вхідного масиву
2	math	floor	Округлення числа в меншу сторону	Число	Округлене число

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
3	math	ceil	Округлення числа в більшу сторону	Число	Округлене число
4	Frame	tkraise	Виведення фрейму	Об'єкт Frame	Відсутні
5	Combobox	current	Задання значення за умовчанням	Індекс елемента	Відсутні
6	Combobox	get	Отримати обраний елемент з випадючого списку	Відсутні	Строка
7	Spinbox	set	Задання значення за умовчанням	Число	Відсутні
8	Spinbox	get	Отримати обраний елемент	Відсутні	Число
9	messagebox	showerror	Виведення помилки	Повідомлен ня	Відсутні
10	Widget	destroy	Знищення віджета	Віджет	Відсутні
11	Tk	protocol	Присвоєння функції виконання певної команди (наприклад закриття вікна)	Команда, функція	Відсутні
12	Tk	title	Задання назви вікна	Строка	Відсутні
13	Tk	geometry	Задання розміру вікна	Строка	Відсутні
14	Tk	resizable	Задання можливості змінювати розмір вікна	Булеві змінні	Відсутні

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
15	Tk	Mainloop	Призупинка виконання коду до закриття вікна	Відсутні	Відсутні
16	filedialog	asksaveasfile	Виведення вікна збереження файлу	Назва файлу за умовчанням , розширення файлу	Відкриття створеного або перезаписан ого файлу
17	FigureCanvas TkAgg	get_tk_widget	Створення віджета з об'єкту фігури	Відсутні	Віджет
18	Text	insert	Вставка у віджет тексту	Строка	Відсутні

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 План тестування

Тестувати основний функціонал програмного забезпечення будемо за даним планом:

- а) Тестування правильності введених значень.
 - 1) Тестування при введенні некоректних символів.
 - 2) Тестування при введенні замалих та завеликих значень.
 - 3) Тестування при недостатній, для розв'язку, кількості введених даних.
- б) Тестування коректної роботи при введенні систем, що не мають коренів.
 - 1) Тестування роботи програми при нульовому значенні визначника.
- в) Тестування коректності роботи методів Крамера, Гауса з одиничною діагоналлю, Гауса з вибором головного елемента.
 - 1) Перевірка коректності роботи методу Крамера.
 - 2) Перевірка коректності роботи методу Гауса з одиничною діагоналлю.
 - 3) Перевірка коректності роботи методу Гауса з вибором головного елемента.
- г) Тестування коректності роботи методів Крамера, Гауса з одиничною діагоналлю, Гауса з вибором головного елемента з дробовими коефіцієнтами.
 - 1) Перевірка коректності роботи методу Крамера.
 - 2) Перевірка коректності роботи методу Гауса з одиничною діагоналлю.
 - 3) Перевірка коректності роботи методу Гауса з вибором головного елемента.
- д) Тестування побудови графіків.
 - 1) Перевірка коректності побудови графіків

2) Перевірка коректності побудови графіку при нульових
діагональних значеннях

5.2 Приклади тестування

Тестування правильності введення значень представлені в даних таблицях: некоректні символи(5.1), завеликі або замалі значення(5.2), заповнення не усієї матриці коефіцієнтів(5.3). Тестування коректної роботи при введенні систем, що не мають коренів приведені в таблиці (5.4). Тестування коректності роботи методів приведені в таблицях: Крамера(5.5), Гауса з одиничною діагоналлю(5.6), Гауса з вибором головного елемента(5.7). Тестування коректності роботи методів з дробовими коефіцієнтами приведені в таблицях: Крамера(5.8), Гауса з одиничною діагоналлю(5.9), Гауса з вибором головного елемента(5.10). Тестування коректності побудови графіків приведені в таблиці(5.11) і таблиці(5.12).

Таблиця 5.1 Приклад роботи програми при введенні некоректних символів

Мета тесту	Перевірити можливість введення некоректних символів
Початковий стан програми	Відкрите вікно
Вхідні дані	1 2 3 1 5 6 6 2 7 9 ф 10
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів
Очікуваний результат	Повідомлення про помилку формату даних
Стан програми після проведення випробувань	Видано помилку “Введені некоректні символи”

Таблиця 5.2 Приклад роботи програми при введенні замалих та завеликих значень

Мета тесту	Перевірити можливість введення завеликих або замалих чисел
Початковий стан програми	Відкрите вікно
Вхідні дані	10^{400} 1 1 30 10 5
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів
Очікуваний результат	Повідомлення про помилку величини даних
Стан програми після проведення випробувань	Видано помилку “Введене завелике число”

Таблиця 5.3 Приклад роботи програми при недостатній, для розв’язку, кількості введених даних

Мета тесту	Перевірити можливість розв’язку системи при недостатній кількості вхідних даних
Початковий стан програми	Відкрите вікно
Вхідні дані	Відсутні
Схема проведення тесту	Натиснути кнопку “Розв’язати” не вводячи дані в матрицю
Очікуваний результат	Повідомлення про недостатню кількість вхідних даних

Продовження таблиці 5.3

Стан програми після проведення випробувань	Виведено повідомлення “Не всі поля заповнені”
--	---

Таблиця 5.4 Приклад роботи програми при нульовому значенні визначника

Мета тесту	Перевірити можливість роботи програми, якщо визначник системи рівний нулю
Початковий стан програми	Відкрите вікно
Вхідні дані	0 0 10 0 1 10
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів
Очікуваний результат	Повідомлення про нульовий визначник
Стан програми після проведення випробувань	Видано повідомлення “Визначник рівен 0”

Таблиця 5.5 Приклад роботи методу Крамера

Мета тесту	Перевірити коректність роботи методу Крамера
Початковий стан програми	Відкрите вікно
Вхідні дані	2 5 -10 -10 5 2

Продовження таблиці 5.5

Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів, натиснення кнопки “Розв’язати”
Очікуваний результат	$x_1 = -1$ $x_2 = -1.6$
Стан програми після проведення випробувань	Отримано результат: $x_1 = -1$ $x_2 = -1.6$

Таблиця 5.6 Приклад роботи методу Гауса з одиничною діагоналлю

Мета тесту	Перевірити коректність роботи методу Гауса з одиничною діагоналлю
Початковий стан програми	Відкрите вікно
Вхідні дані	1 2 3 1 0 1 4 2 0 0 1 3
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів, натиснення кнопки “Розв’язати”
Очікуваний результат	$x_1 = 12$ $x_2 = -10$ $x_3 = 3$

Продовження таблиці 5.6

Стан програми після проведення випробувань	Отримано результат: $x_1 = 12$ $x_2 = -10$ $x_3 = 3$
--	---

Таблиця 5.7 Приклад роботи методу Гауса з вибором головного елемента

Мета тесту	Перевірити коректність роботи методу Гауса з вибором головного елемента
Початковий стан програми	Відкрите вікно
Вхідні дані	1 10 -1 15 1 20 -1 25 1 30 -1 35 1 40 -1 45 1 50 -1 55
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів, натиснення кнопки “Розв’язати”
Очікуваний результат	$x_1 = 0.01$ $x_2 = -0.9$ $x_3 = 1.1$ $x_4 = 0.8$
Стан програми після проведення випробувань	Отримано результат: $x_1 = 0.0127$ $x_2 = -0.928$ $x_3 = 1.12$ $x_4 = 0.759$

Таблиця 5.8 Приклад роботи методу Крамера з дробовими коефіцієнтами

Мета тесту	Перевірити коректність роботи методу Крамера з дробовими коефіцієнтами
Початковий стан програми	Відкрите вікно
Вхідні дані	1.5 1.8 1 1.7 1.6 1
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів, натиснення кнопки “Розв’язати”
Очікуваний результат	$x_1 = x_2 = 0.303$
Стан програми після проведення випробувань	Отримано результат: $x_1 = 0.303$ $x_2 = 0.303$

Таблиця 5.9 Приклад роботи методу Гауса з одиничною діагоналлю з дробовими символами

Мета тесту	Перевірити коректність роботи методу Гауса з одиничною діагоналлю з дробовими символами
Початковий стан програми	Відкрите вікно
Вхідні дані	1.1 1.3 1 1.2 1.4 1
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів, натиснення кнопки “Розв’язати”

Продовження таблиці 5.9

Очікуваний результат	$x_1 = -5$ $x_2 = 5$
Стан програми після проведення випробувань	Отримано результат: $x_1 = -5$ $x_2 = 5$

Таблиця 5.10 Приклад роботи методу Гауса з вибором головного елемента з дробовими коефіцієнтами

Мета тесту	Перевірити коректність роботи методу Гауса з вибором головного елемента з дробовими коефіцієнтами
Початковий стан програми	Відкрите вікно
Вхідні дані	2.5 5.5 1 3.5 3.555 1
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів, натиснення кнопки “Розв’язати”
Очікуваний результат	$x_1 = 0.9$ $x_2 = 0.1$
Стан програми після проведення випробувань	Отримано результат: $x_1 = 0.188$ $x_2 = 0.0965$

Таблиця 5.11 Приклад побудови графіку СЛАР

Мета тесту	Перевірити можливість будувати графічний розв'язок СЛАР
Початковий стан програми	Відкрите вікно
Вхідні дані	50 30 10 2 2 2
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів, натиснення кнопки “Графік” у вікні результатів
Очікуваний результат	Побудується графік двох прямих, що перетинаються в точці (-1,2)
Стан програми після проведення випробувань	Програма побудувала графік коректно (рисунок 5.1)

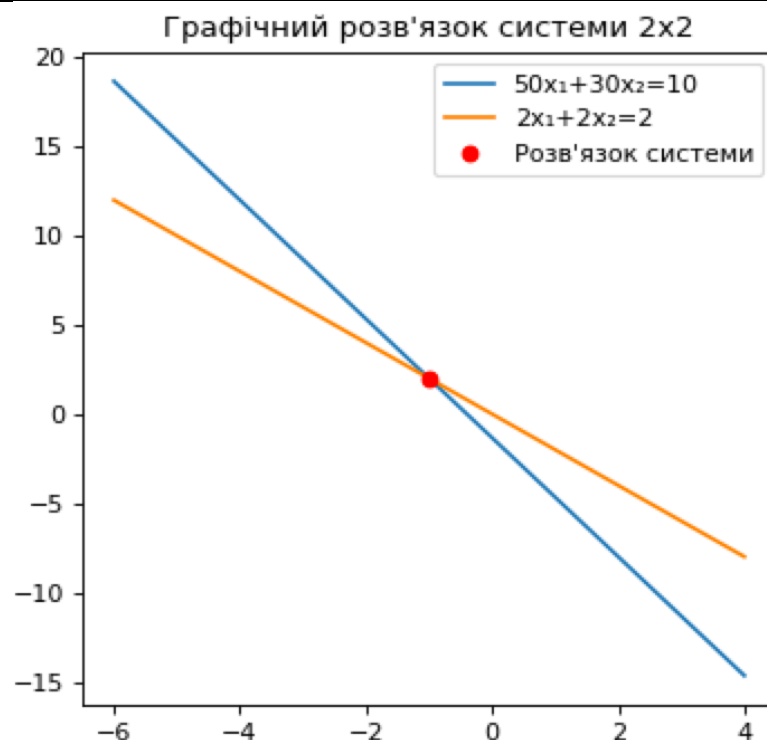


Рисунок 5.1 Результат побудови графіку для таблиці 5.11

Таблиця 5.12 Приклад побудови графіку при нульових значеннях діагоналі

Мета тесту	Перевірити можливість будувати графічний розв'язок системи при нульових діагональних коефіцієнтах
Початковий стан програми	Відкрите вікно
Вхідні дані	0 2 10 5 0 10
Схема проведення тесту	Поелементне заповнення матриці коефіцієнтів і стовпця вільних членів, натиснення кнопки “Графік” у вікні результатів
Очікуваний результат	Побудується графік двох прямих, що перетинаються в точці (2,5)
Стан програми після проведення випробувань	Програма побудувала графік коректно (рисунок 5.2)

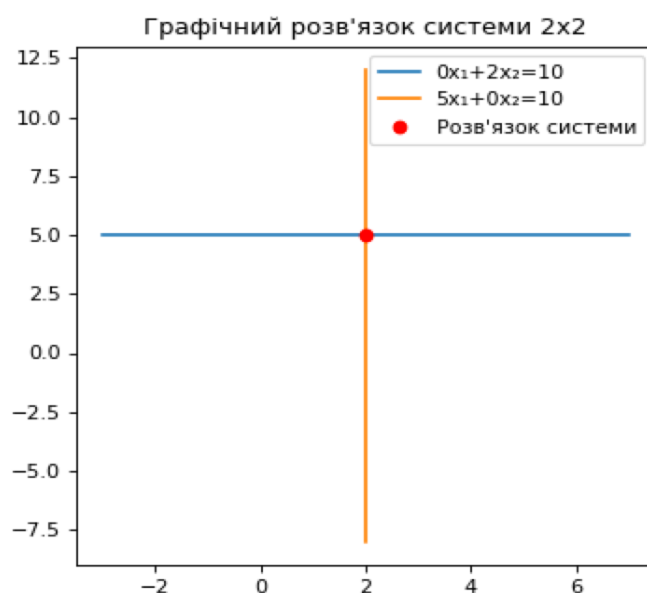


Рисунок 5.2 Результат побудови графіку для таблиці 5.12

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

6.1 Робота з програмою

Після запуску виконавчого файлу з розширенням *.app, відкривається головне вікно програми (Рисунок 6.1).

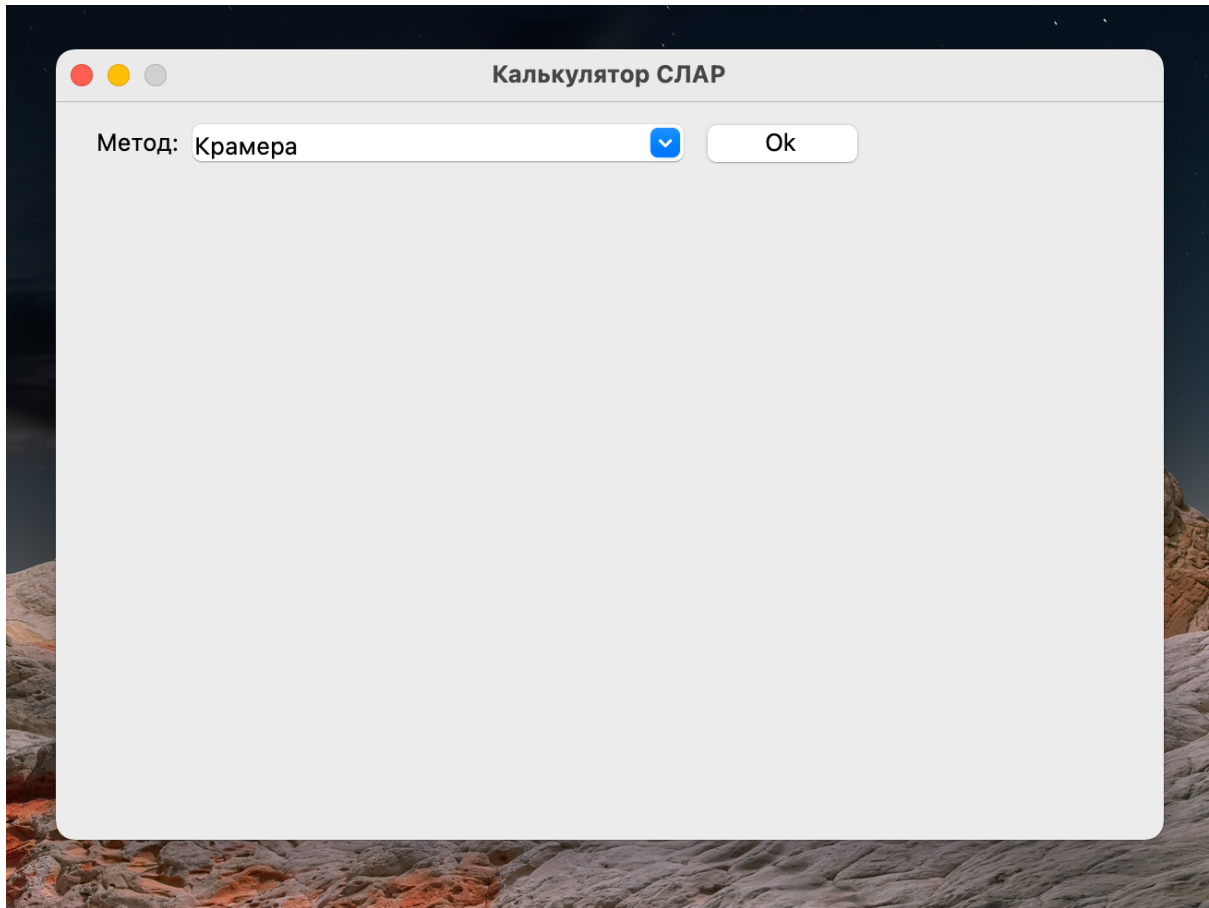


Рисунок 6.1 – Головне вікно програми

Далі за допомогою випадаючого списку під назвою «Метод», шляхом натиску на нього, необхідно обрати метод розв’язку системи, що буде оброблятися програмою (рисунок 6.2):

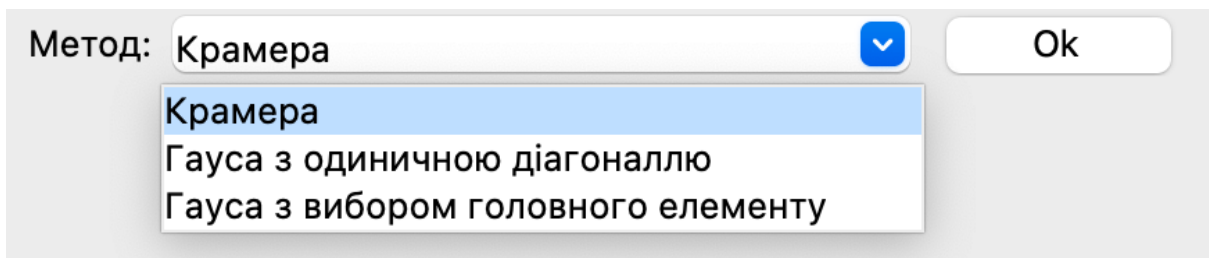


Рисунок 6.2 – Вибір необхідного методу розв’язку системи

Далі після натиску кнопки «Ok» за допомогою лічильника з назвою «Розмірність» шляхом натиску на стрілки або введенням числа з клавіатури

необхідно виставити розмір системи, що буде оброблятися програмою (рисунок 6.3):

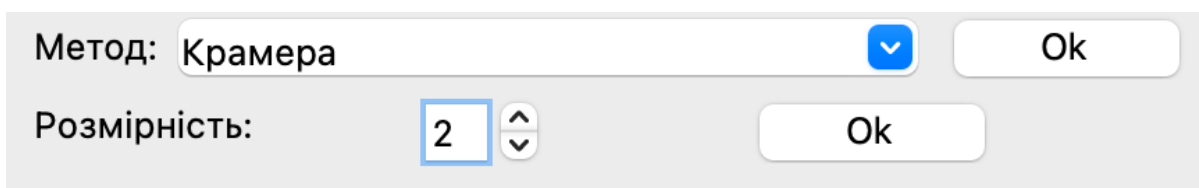


Рисунок 6.3 – Вибір необхідного розміру системи

Далі після натиску кнопки «Ok», шляхом введенням чисел з клавіатури необхідно заповнити матрицю коефіцієнтів і стовпець вільних членів, для подальшого розв’язку(рисунок 6.4):

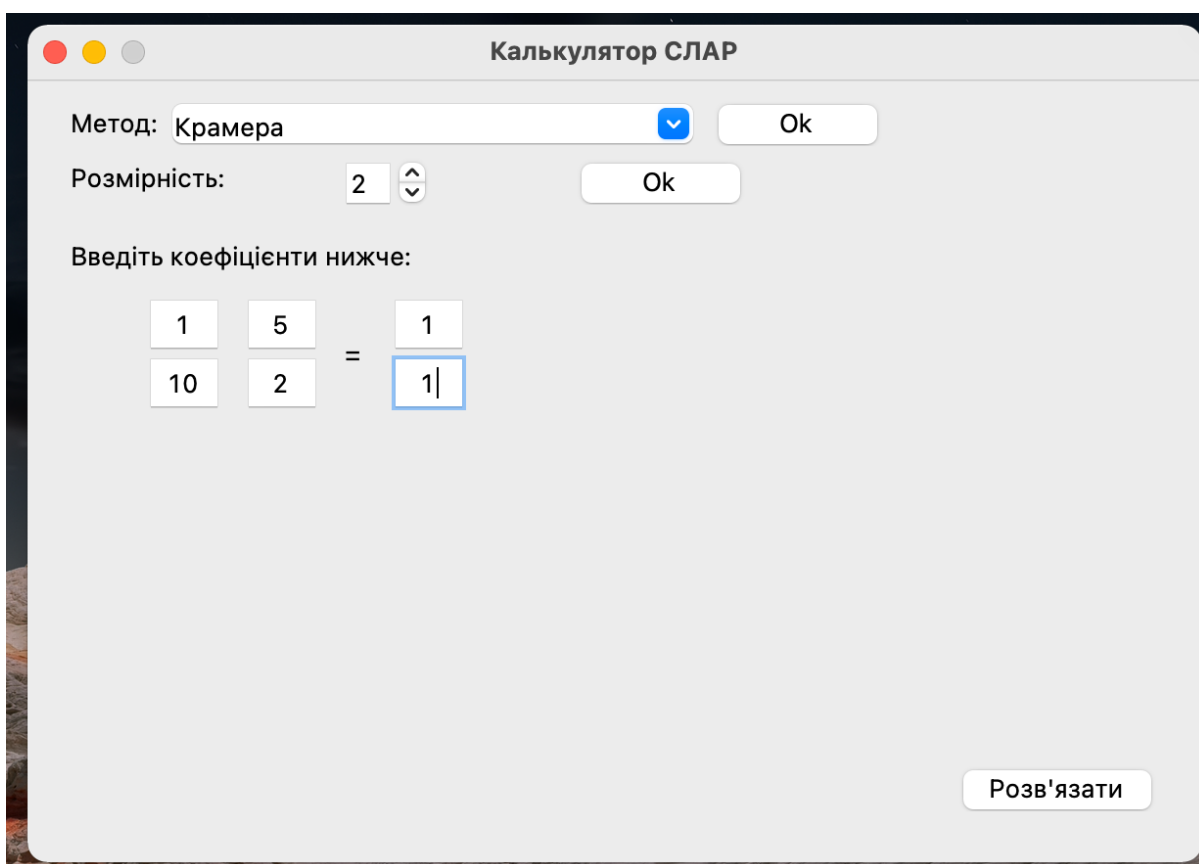


Рисунок 6.4 – Заповнення системи

Далі після натиску кнопки «Розв’язати», відкривається вікно розв’язку СЛАР(рисунок 6.5):

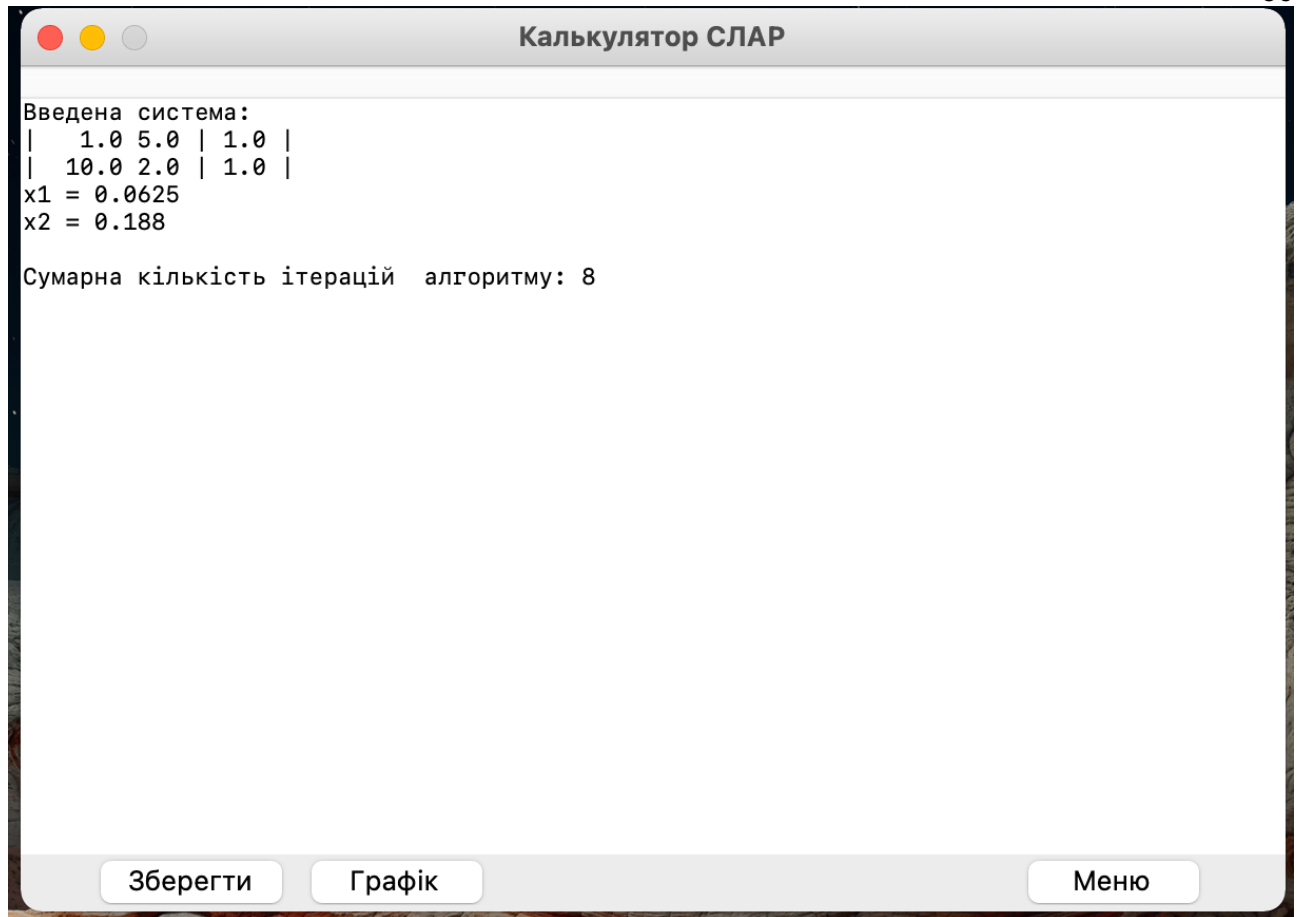


Рисунок 6.5 – Розв’язок системи

В цьому вікні користувачу надаються такі опції:

1. Перехід в головне меню;
2. Збереження розв’язку у файл з розширенням “.txt”;
3. За умови, що система має розмірність 2 на 2, користувачу надається можливість побачити графічний розв’язок системи, та зберегти його у файл формату “.png”(рисунок 6.6):

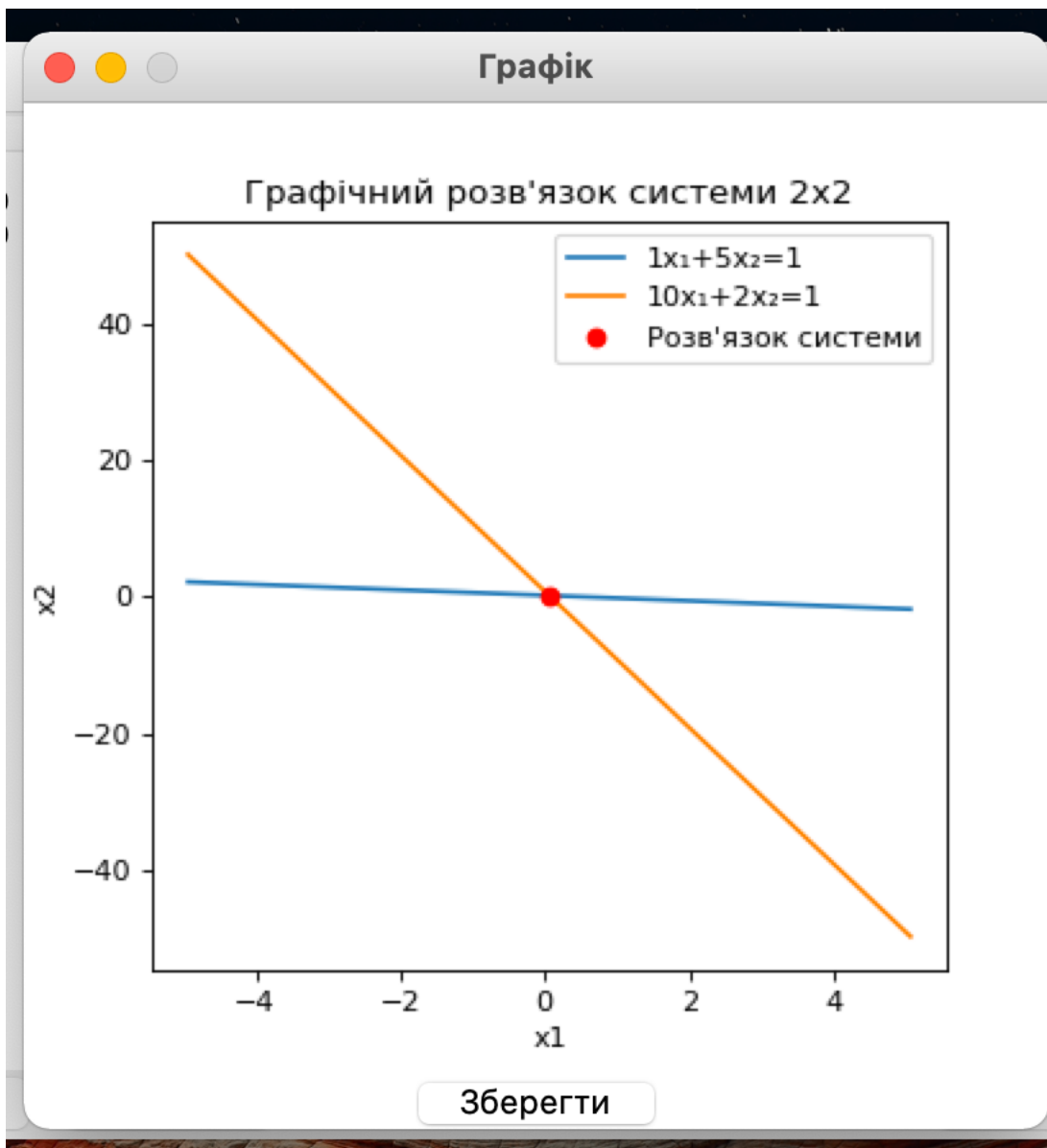


Рисунок 6.6 – Вікно графічного розв'язку системи

6.2 Формат вхідних та вихідних даних

Користувачем на вхід програми подається СЛАР у матричному вигляді, тобто задається за допомогою матриці системи та стовпця вільних членів, числа яких дійсні точністю не більше, ніж 3 знака після коми (якщо точність більша, то програма автоматично округлить їх за математичними правилами до 3-х знаків після коми).

Результатом виконання програми є розв'язок заданої СЛАР, який видається у вигляді списку кожне число котрого записане з точністю до 3-х знаків після коми або повідомлення, що дана система не має розв'язків або не сходиться для обраного методу. За умови, що розмірність заданої СЛАР дорівнює двом,

користувачу надається додаткова можливість побачити графічний розв'язок СЛАР.

6.3 Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 6.1.

Таблиця 6.1 – Системні вимоги програмного забезпечення

	Мінімальні	Рекомендовані
Операційна система	MacOS 10.12	MacOS 11.6.1
Процесор	Влаштовані процесори Intel або Apple	Влаштовані процесори Intel або Apple
Оперативна пам'ять	256 MB RAM	512 MB RAM
Відеоадаптер	Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог)	
Дисплей	800x600	1024x768 або краще
Прилади введення	Клавіатура, комп'ютерна миша	

7 АНАЛІЗ РЕЗУЛЬТАТІВ

Головною задачею курсової роботи була реалізація програми для розв'язання СЛАР наступними методами: Крамера, Гауса з одиничною діагоналлю, Гауса з вибором головного елементу.

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, ретельно перевіряють на валідність і лише потім подаються на обробку програмі.

Для перевірки та доведення достовірності результатів виконання програмного забезпечення скористаюся WolframAlpha:

1) Метод Крамера.

Результат виконання методу Крамера наведено на рисунку 7.1:



Рисунок 7.1 – Результат виконання методу Крамера

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



Рисунок 7.2 – Перевірка методу Крамера в WolframAlpha

2) Метод Гауса з одиничною діагоналлю.

Результат виконання методу Гауса з одиничною діагоналлю наведено на рисунку 7.1:

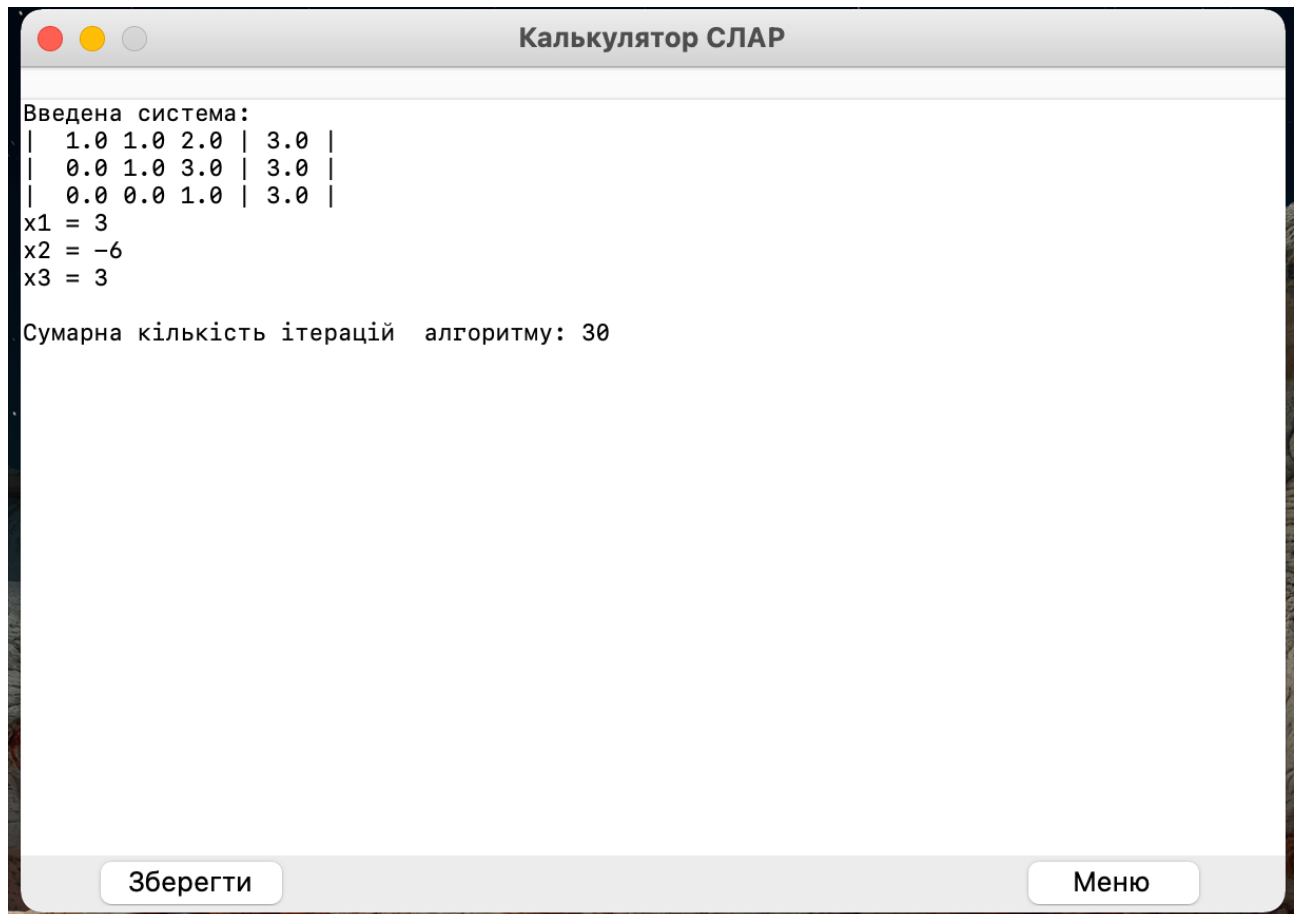


Рисунок 7.3 – Результат виконання методу Гауса з одиничною діагоналлю

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

Input interpretation

	$x + y + 2z = 3$
solve	$y + 3z = 3$
	$z = 3$

Result ☒ Step-by-step solution

$x = 3$ and $y = -6$

Рисунок 7.4 – Перевірка методу Гауса з одиничною діагоналлю в WolframAlpha

3) Метод Гауса з вибором головного елемента.

Результат виконання методу Крамера наведено на рисунку 7.1:

Калькулятор СЛАР

Введена система:

1.0	2.0	3.0	3.0
3.0	-1.0	-3.0	3.0
5.0	3.0	1.0	3.0

$x_1 = 2.57$
 $x_2 = -4.29$
 $x_3 = 3$

Сумарна кількість ітерацій алгоритму: 27

Зберегти Меню

Рисунок 7.5 – Результат виконання методу Гауса з вибором головного елемента

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

Input interpretation

	$x + 2y + 3z = 3$
solve	$3x - y - 3z = 3$
	$5x + 3y + z = 3$

Result

$x = \frac{18}{7}$ and $y = -\frac{30}{7}$ and $z = 3$

Approximate form ☒ Step-by-step solution

Download Page

POWERED BY THE WOLFRAM LANGUAGE

Рисунок 7.6 – Перевірка методу Гауса з вибором головного елемента в WolframAlpha

Для проведення тестування ефективності програми було створено матриці наступного вигляду:

$$\begin{pmatrix} 2n & 1 & 1 & 1 & \dots & 1 & n \\ 1 & 2n & 1 & 1 & \dots & 1 & n \\ 1 & 1 & 2n & 1 & \dots & 1 & n \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & 1 & 1 & \dots & 2n & n \end{pmatrix} \quad (7.1)$$

де n – розмірність системи.

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

Таблиця 7.1 – Тестування ефективності методів

Розмірність системи	Параметри тестування	Метод		
		Крамера	Гауса з одиничною діагоналлю	Гауса з вибором головного елемента
1000	Кількість ітерацій	10^9	501501000	335333998
2500	Кількість ітерацій	15625000000	7821877500	5220834998
5000	Кількість ітерацій	$225 \cdot 10^9$	62537505000	41716669998
10000	Кількість ітерацій	10^{12}	500150010000	333533339998

Продовження таблиці 7.1

Розмірність системи	Параметри тестування	Метод		
		Крамера	Гауса з одиничною діагоналлю	Гауса з вибором головного елемента
15000	Кількість ітерацій	$3375 \cdot 10^9$	1687837515000	1125450009998

Візуалізація результатів таблиці 7.1 наведено на рисунку 7.1:

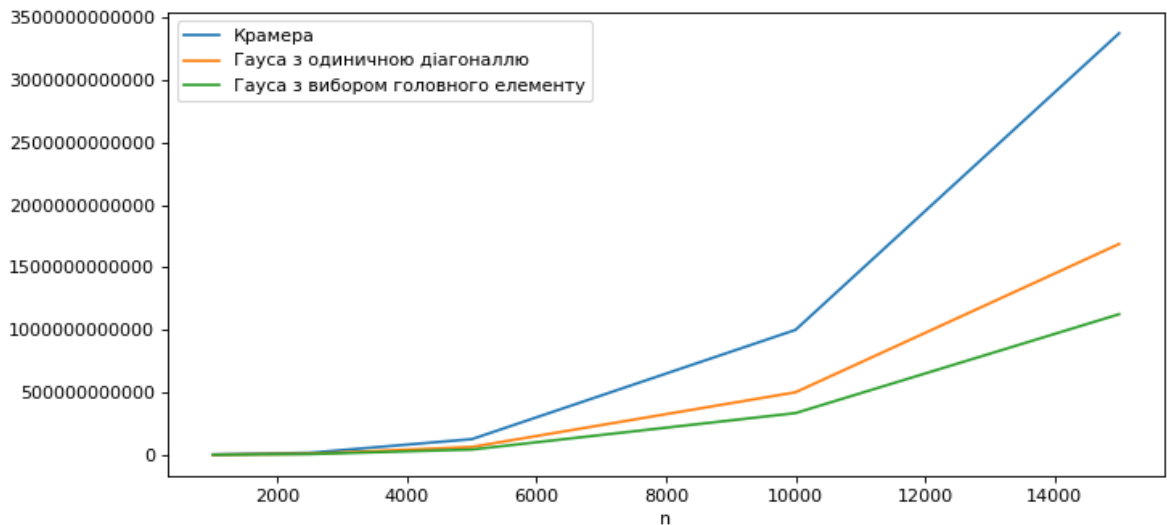


Рисунок 7.1 – Графік залежності кількості ітерацій методу від розміру вхідної системи

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 – розмір СЛАР.
- З розглянутих методів найоптимальнішим для практичного використання є метод Гауса з вибором головного елемента, оскільки він виконується найшвидше та має такі умови сумісності, що охоплюють найширший спектр СЛАР.

Висновки

В ході даної курсової роботи було проаналізовано хід розв'язку довільних СЛАР трьома методами, а саме: Крамера, Гауса з одиничною діагоналлю і Гауса з вибором головного елементу. Імплементовано алгоритми відповідних методів. Реалізовано їх було згідно парадигми ООП. Для класів розробленого програмного забезпечення було створено діаграму класів, і проаналізовано методи відповідних класів. В ході тестування програмного забезпечення були отримані задовільні результати по кожному з тестів. Також була підготовлена інструкція користувача, що включає в себе: опис алгоритму роботи з програмою, дані яка приймає програма та системні вимоги. В процесі аналізу алгоритмів було виявлено, що складність всіх розглянутих методів є кубічною.

ПЕРЕЛІК ПОСИЛАНЬ

1. Розв'язування систем лінійних рівнянь методом Крамера. URL: <https://www.mathros.net.ua/rozvjazok-systemy-linijnyh-algebraichnyh-rivnjan-metodom-kramera.html> (дата звернення: 16.04.2022).
2. Розв'язування систем лінійних рівнянь методом Гауса. URL: https://www.mathros.net.ua/metod-gaussa-rozvjazok-systemy-linijnyh-rivnjan-metodom-gaussa.html#gaussian_method_algorithm (дата звернення: 17.04.2022).
3. Метод Гаусса з вибором головного елемента. URL: https://www.mathros.net.ua/metod-gaussa-z-vyborom-golovnogo-elementa.html#gaussian_method_algorithm (дата звернення: 19.04.2022).

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ**КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського****Кафедра
інформатики та програмної інженерії****Затвердив****Керівник Головченко Максим Миколайович****«___» _____ 201_ р.****Виконавець:****Студент Рибалка Ілля Сергійович****«___» _____ 201_ р.****ТЕХНІЧНЕ ЗАВДАННЯ****на виконання курсової роботи****на тему: Розв'язання СЛАР точними методами****з дисципліни:****«Основи програмування»****Київ 2022**

1. *Мета:* Метою курсової роботи є розробка ефективного програмного забезпечення для розв'язання СЛАР.

2. *Дата початку роботи:* «12» квітня 2022 р.

3. *Дата закінчення роботи:* «__» _____ 202_ р.

4. *Вимоги до програмного забезпечення.*

1) Функціональні вимоги:

- Можливість задавати розмірність СЛАР в межах від 2 до 7;
- Можливість розв'язку СЛАР обраним методом:
 - o метод Крамера;
 - o метод Гауса з одиничною діагоналлю;
 - o метод Гауса з вибором головного елементу.
- Можливість перевіряти коректність введених даних;
- Можливість виведення графічного розв'язання СЛАР для системи 2 на 2;
- Можливість збереження графіку у файл;
- Можливість збереження результатів розв'язання у файл.

2) Нефункціональні вимоги:

- Можливість запускати програмне забезпечення на операційних системах MacOS Big Sur версії 11.6.1 і вище.

- Все програмне забезпечення та супроводжуюча 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.

5. *Стадії та етапи розробки:*

- 1) Об'єктно-орієнтований аналіз предметної області задачі (до __.__.202_ р.)
- 2) Об'єктно-орієнтоване проектування архітектури програмної системи (до __.__.202_ р.)
- 3) Розробка програмного забезпечення (до __.__.202_ р.)
- 4) Тестування розробленої програми (до __.__.202_ р.)
- 5) Розробка пояснювальної записки (до __.__.202_ р.).

6) Захист курсової роботи (до __.__.202_ р.).

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
розв'язку СЛАР точними методами*

(Найменування програми (документа))

Віртуальний носій

(Вид носія даних)

16 арк, 28 Кб

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

студента групи ІП-15 І курсу

Рибалки І.С.

ФАЙЛ interface.py

```

import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from numpy.linalg import det
from math import floor, ceil
import tkinter as tk
from tkinter import messagebox, filedialog
from tkinter.ttk import *
import re
from Cramer import Cramer
from Gauswithsingdiagonal import Gauswithsingdiagonal
from Gauswithmainelement import Gauswithmainelement

class AppInterface(tk.Tk):
    """
    Вікно програми
    """
    def __init__(self):
        tk.Tk.__init__(self)
        container = self.container = Frame(self)
        container.pack(side="top", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)
        frame = self.menuframe = MainMenu(container, self)
        frame.grid(row=0, column=0, sticky="nsew")
        frame.tkraise()

# Створення фрейму розв'язку
def create_resultmenu(self):
    frame = self.resultframe = ResultMenu(self.container, self)
    frame.grid(row=0, column=0, sticky="nsew")

```

```
frame.tkraise()
```

```
# Виведення фрейму головного меню
```

```
def show_mainmenu(self):
```

```
    frame = self.menuframe
```

```
    frame.tkraise()
```

```
class MainMenu(Frame):
```

```
    """
```

```
    Головне меню програми
```

```
    """
```

```
    def __init__(self, parent, root):
```

```
        Frame.__init__(self, parent)
```

```
        self.root = root
```

```
        widdata = self.widdata = []
```

```
        label = Label(self, text="Метод:")
```

```
        label.place(x=20, y=12)
```

```
        combo = self.__combo = Combobox(self, values=["Крамера", "Гауса з  
одиночною діагоналлю", "Гауса з вибором головного елементу"],  
state="readonly") # Випадаючий список для вибору метода розв'язку
```

```
        combo.current(0)
```

```
        combo.place(x=70, y=10, width=275)
```

```
        bt = Button(self, text='Ok', command=self.dimension)
```

```
        bt.place(x=350, y=10)
```

```
        widdata.append(label)
```

```
        widdata.append(combo)
```

```
        widdata.append(bt)
```

```
# Створення віджетів для вибору розмірності
```

```
def dimension(self):
```

```
    self.clean()
```

```

widdata = self.widdata
label2 = Label(self, text="Розмірність:")
label2.place(x=20, y=40)
dmnsn = self.__dmnsn = Spinbox(self, from_=2, to_=7, width=2)
dmnsn.set('3')
dmnsn.place(x=160, y=40)
btn1 = Button(self, text="Ok", command=self.inputdata)
btn1.place(x=280, y=40)
widdata.append(dmnsn)
widdata.append(btn1)
widdata.append(label2)

```

Створення полів для введення системи

```

def inputdata(self):
    self.clean()
    method = self.method = self.__combo.get()
    n = self.n = int(self.__dmnsn.get())
    if not n or not (n in range(2, 8)):
        messagebox.showerror("Помилка", "Неправильно введена розмірність")
        return
    matrixofstr = self.__matrixofstr = [[0] * n for i in range(0, n)]
    lstofstr = self.__lstofstr = [[0] for i in range(0, n)]
    if method == "Крамера":
        label4 = Label(self, text="=")
        label4.place(x=60+n*50, y=100 + n * 15)
        for i in range(0, n):
            for j in range(0, n):
                matrixofstr[i][j] = Entry(self, justify = tk.CENTER)
                matrixofstr[i][j].place(x=60 + j * 50, y=110 + i * 30, width=40, height=30)
            lstofstr[i] = Entry(self, justify = tk.CENTER)
            lstofstr[i].place(x=85+n*50, y=110 + i * 30, width=40, height=30)

```

```

else:
    for i in range(n):
        for j in range(n):
            if j == (n - 1):
                str = "x {}=" .format(j+1)
            else:
                str = "x {}+" .format(j+1)
            label4 = Label(self, text=str)
            matrixofstr[i][j] = Entry(self, justify = tk.CENTER)
            label4.place(x=120 + j * 65, y=110 + i * 30, width=40, height=30)
            matrixofstr[i][j].place(x=80 + j * 65, y=110 + i * 30, width=40, height=30)
            lstofstr[i] = Entry(self, justify = tk.CENTER)
            lstofstr[i].place(x=80 + n * 65, y=110 + 30 * i, width=40, height=30)
label3 = Label(self, text="Введіть коефіцієнти нижче:")
label3.place(x=20, y=80)
btn3 = Button(self, text="Розв'язати", command = self.readdata)
btn3.place(x=475, y=350)

```

Зчитування системи

```
def readdata(self):
```

```
    # Округлення числа за математичними правилами
```

```
def roundnum(num):
```

```
    tnum = num * 10 ** 3
```

```
    if abs(tnum) - abs(floor(tnum)) < 0.5:
```

```
        return floor(tnum) / 10 ** 3
```

```
    return ceil(tnum) / 10 ** 3
```

```
method = self.method
```

```
n = self.n
```

```
root = self.root
```

```
MatrixA = self.MatrixA = [[0] * n for i in range(n)]
```

```
LstB = self.LstB = [0 * n for i in range(n)]
```

```

matrixofstr = self.__matrixofstr
lstofstr = self.__lstofstr
for i in range(n):
    for j in range(n):
        num = matrixofstr[i][j].get()
        val = self.validatenum(num)
        if val[0]:
            MatrixA[i][j] = roundnum(float(num))
        else:
            return val[1]
for i in range(n):
    num = lstofstr[i].get()
    val = self.validatenum(num)
    if val[0]:
        LstB[i] = roundnum(float(num))
    else:
        return val[1]
if det(MatrixA) == 0:
    messagebox.showerror("Помилка", "Визначник рівен 0")
else:
    if method == "Гауса з одиничною діагоналлю":
        if self.validatea():
            messagebox.showerror("Помилка", "СЛАР неможливо розв'язати даним методом")
        else:
            self.clean()
            root.create_resultmenu()
    else:
        self.clean()
        root.create_resultmenu()

```

Валідація введеної системи для розв'язку методом Гауса з одиничною діагоналлю

```
def validatea(self):
    MatrixA = self.MatrixA
    LstB = self.LstB
    n = self.n
    for i in range(n):
        if MatrixA[i][i] == 0:
            for j in range(i+1,n):
                if MatrixA[j][i] != 0:
                    MatrixA[i], MatrixA[j] = MatrixA[j], MatrixA[i]
                    LstB[i], LstB[j] = LstB[j], LstB[i]
            elif j==n-1:
                return True
    return False
```

Валідація числа

```
def validatenum(self, num):
    if not num:
        return False, messagebox.showerror("Помилка", "Не всі поля заповнені")
    elif re.fullmatch(r'[-+]?[0-9]*\.?[0-9]*', num)==None:
        return False, messagebox.showerror("Помилка", "Введені некоректні симболи")
    elif float(num)>=float(10**6) or float(num)<=-float(10**6):
        return False, messagebox.showerror("Помилка", "Введене {} число".format("завелике" if float(num)>0 else "замале"))
    else:
        return True, None
```

Очищення фрейму від віджетів

```
def clean(self):
```



```

widdata = self.widdata
for widget in self.winfo_children():
    if widget not in widdata:
        widget.destroy()

```

```

class ResultMenu(Frame):

```

```

    """

```

```

    Меню результатів розв'язку

```

```

    """

```

```

    def __init__(self, parent, root):

```

```

        Frame.__init__(self, parent)

```

```

        self.root = root

```

```

        self.MatrixA = root.menuframe.MatrixA

```

```

        self.LstB = root.menuframe.LstB

```

```

        self.n = root.menuframe.n

```

```

        self.method = root.menuframe.method

```

```

        self.__plotwinisopen = False

```

```

        self.__text = tk.Text(self, wrap=tk.NONE, highlightthickness = 0)

```

```

        self.__text.place(x=0, y=13, width= 600, height= 360)

```

```

        self.insmatrix()

```

```

        scl = Scrollbar(self, orient='horizontal')

```

```

        scl.pack(side = tk.TOP, fill = tk.X)

```

```

        scl.config(command=self.__text.xview)

```

```

        self.__text.config(xscrollcommand= scl.set)

```

```

        btn1 = Button(self, text="Меню", command = self.option)

```

```

        btn2 = Button(self, text="Зберегти", command = self.savefile)

```

```

        btn1.place(x=475, y=373)

```

```

        btn2.place(x=35, y=373)

```

```

        if self.method == "Крамера":

```

```

            self.Cramer()

```

```

        elif self.method == "Гауса з одиничною діагоналлю":

```

```

        self.Gauswsd()
    else:
        self.Gauswme()
    if self.n == 2:
        self.__btn3 = Button(self, text="Графік", command=self.plotwin)
        self.__btn3.place(x=135, y=373)
    root.protocol('WM_DELETE_WINDOW',
self.option("DELETE_WINDOW"))

```

Розв'язок СЛАР методом Крамера

```

def Cramer(self):
    MatrixA = self.MatrixA
    LstB = self.LstB
    n = self.n
    result = Cramer(MatrixA, LstB)
    x = self.x = result.solve()
    text = self.__text
    for i in range(n):
        text.insert(tk.END, "x {} = {:.3g}\n".format(i+1,x[i]))
    text.insert(tk.END, "\nСумарна кількість ітерацій
    {}").format(n**3))

```

Розв'язок СЛАР методом Гауса з одиничною діагоналлю

```

def Gauswsd(self):
    MatrixA = self.MatrixA
    LstB = self.LstB
    n = self.n
    result = Gauswithsingdiagonal(MatrixA, LstB)
    x = self.x = result.solve()
    text = self.__text
    for i in range(n):

```

```

text.insert(tk.END, "x {} = {:.3g}\n".format(i+1,x[i]))
text.insert(tk.END, "\nСумарна кількість ітерацій алгоритму:
{}".format(sum([(1+n)*(n-i)+(1+i) for i in range(n)])))

```

```

# Розв'язок СЛАР методом Гауса з вибором головного елемента
def Gauswme(self):
    MatrixA = self.MatrixA
    LstB = self.LstB
    n = self.n
    result = Gauswithmainelement(MatrixA, LstB)
    x = self.x = result.solve()
    text = self.__text
    for i in range(n):
        text.insert(tk.END, "x {} = {:.3g}\n".format(i+1,x[i]))
    text.insert(tk.END, "\nСумарна кількість ітерацій алгоритму:
{}".format(sum((n-k+1)*(n-k) for k in range(n-1))+n**2))

```

Створення вікна графічного розв'язку системи 2 на 2

```

def plotwin(self):
    MatrixA = self.MatrixA
    LstB = self.LstB
    res = self.x
    feq = "{:g}x1{:+g}x2={:g}".format(MatrixA[0][0], MatrixA[0][1], LstB[0])
    seq = "{:g}x1{:+g}x2={:g}".format(MatrixA[1][0], MatrixA[1][1], LstB[1])
    plotwindow = self.plotwindow = tk.Tk()
    plotwindow.title("Графік")
    plotwindow.geometry('400x400+150+150')
    plotwindow.resizable(False, False)
    self.__btn3["state"] = tk.DISABLED
    self.__plotwinisopen = True

```

```

tk.Button(plotwindow, text = "Зберегти", width=10, height=1, command =
lambda: self.savefile("plot")).pack(side=tk.BOTTOM)

fig = self.fig = plt.Figure(figsize=(10, 10), dpi=80)
x = [res[0]-10, res[0]+10]
y = lambda j: [(LstB[j] - MatrixA[j][0]*x[i])/MatrixA[j][1] for i in range(2)]
fpl = fig.add_subplot()
fpl.set_title('Графічний розв'язок системи 2x2')
for i in range(2):
    if MatrixA[i][1] == 0:
        fpl.plot([LstB[i]/MatrixA[i][0], LstB[i]/MatrixA[i][0]], x)
    else:
        fpl.plot([res[0]-5, res[0]+5], y(i))
fpl.plot(res[0], res[1], 'ro')
fpl.set_xlabel('x1')
fpl.set_ylabel('x2')
fpl.legend([feq, seq, "Розв'язок системи"])
canvas = FigureCanvasTkAgg(fig, plotwindow)
canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH)
self.plotwindow.protocol('WM_DELETE_WINDOW',
lambda:
self.option("close_plot"))
self.plotwindow.mainloop()

# Вставка в віджет Text введеної системи
def insmatrix(self):
    # Визначення відступу між елементами
    def maxincols():
        lst = [0 for i in range(n+1)]
        for i in range(n):
            lst[i] = max([len(str(MatrixA[j][i])) for j in range(n))]+1)
        lst[n] = max([len(str(i)) for i in LstB])
        lst = [10 if i>10 else i for i in lst]

```

```

        return lst

    MatrixA = self.MatrixA
    LstB = self.LstB
    n = self.n
    setw = maxincols()
    text = self.__text
    text.insert(tk.END, "Введена система:\n")
    for i in range(n):
        text.insert(tk.END, "| ")
        for j in range(n+2):
            if j == n:
                text.insert(tk.END, " | ")
            else:
                if j == self.n+1:
                    text.insert(tk.END, "{:>{}}".format(LstB[i], setw[n]))
                else:
                    text.insert(tk.END, "{:>{}}".format(MatrixA[i][j], setw[j]))
        text.insert(tk.END, "\n")

# Збереження файлу (текстового або графічного розв'язку)
def savefile(self, type = "text"):
    if type == "plot":
        file = filedialog.asksaveasfile(initialfile =
'Untitled.png',defaulttextextension=".png")
        if file:
            self.fig.savefig(file.name)
        else:
            method = "Cramer" if self.method == "Крамера" else "Gaussian"
            file = filedialog.asksaveasfile(mode='w', initialfile = '{}.txt'.format(method),
defaulttextextension=".txt")
            if file:

```

```

file.write(self.__text.get("1.0",tk.END))
file.close()

```

Метод відповідальний за коректне закриття меню результатів

```
def option(self, stat = "show_mainmenu"):
```

```
    if self.__plotwinisopen:
```

```
        self.__plotwinisopen = False
```

```
        self.plotwindow.destroy()
```

```
    if stat == "DELETE_WINDOW":
```

```
        self.root.destroy()
```

```
    elif stat == "close_plot":
```

```
        self.__btn3["state"] = tk.NORMAL
```

```
    else:
```

```
        self.root.show_mainmenu()
```

```
        self.destroy()
```

```
app = AppInterface()
```

```
app.title("Калькулятор СЛАР")
```

```
app.geometry('600x400+325+150')
```

```
app.resizable(False, False)
```

```
app.mainloop()
```

ФАЙЛ Cramer.py

```

from copy import deepcopy
from numpy.linalg import det
class Cramer:
    def __init__(self, Matrix, Lst):
        self.__MatrixA = Matrix # Матриця коефіцієнтів
        self.__LstB = Lst # Стовпець вільних членів
        self.__n = len(Lst) # Розмірність системи

# Метод для розв'язку СЛАР
def solve(self):
    MatrixA = self.__MatrixA
    LstB = self.__LstB
    n = self.__n
    res=[]
    d = det(MatrixA)
    MatrixAcopy = deepcopy(MatrixA)
    for i in range(n):
        for j in range(n):
            MatrixAcopy[j][i] = LstB[j]
        dt=det(MatrixAcopy)
        MatrixAcopy = deepcopy(MatrixA)
        res.append(dt/d)
    return res

```

ФАЙЛ Gauswithsingdiagonal.py

```

from copy import deepcopy

class Gauswithsingdiagonal:

    def __init__(self, Matrix, Lst):

        self.__MatrixA = Matrix # Матриця коефіцієнтів
        self.__LstB = Lst # Стовпець вільних членів
        self.__n = len(Lst) # Розмірність системи

# Метод для розв'язку СЛАР
def solve(self):

    MatrixA = deepcopy(self.__MatrixA)
    LstB = deepcopy(self.__LstB)
    n = self.__n

    # Прямий хід
    for i in range(n):
        LstB[i]/=MatrixA[i][i]
        for j in range(n-1, -1, -1):
            MatrixA[i][j] /= MatrixA[i][i]
        for j in range(n-1, i, -1):
            LstB[j]-=MatrixA[j][i] * LstB[i]
            for k in range(n-1, -1, -1):
                MatrixA[j][k] -= MatrixA[j][i] * MatrixA[i][k]

    # Зворотній хід
    for i in range(n-1, -1, -1):
        for j in range(i-1,-1,-1):
            LstB[j] -= LstB[i]*MatrixA[j][i]
            MatrixA[j][i] -= MatrixA[j][i]

    return LstB

```


ФАЙЛ Gauswithmainelement.py

```

from copy import deepcopy
class Gauswithmainelement:
    def __init__(self, Matrix, Lst):
        self.__MatrixA = Matrix # Матриця коефіцієнтів
        for i in range(len(Lst)):
            self.__MatrixA[i].append(Lst[i]) # Зклеювання матриці коефіцієнтів і
            # стовпця вільних членів
        self.__n = len(Lst) # Розмірність системи

    # Метод для пошуку максимального за модулем елементу в стовпці
    def __maxinrow(self, col):
        MatrixA = self.__MatrixA
        max_element = MatrixA[col][col]
        max_row = col
        for i in range(col + 1, len(MatrixA)):
            if abs(MatrixA[i][col]) > abs(max_element):
                max_element = MatrixA[i][col]
                max_row = i
        if max_row != col:
            MatrixA[col], MatrixA[max_row] = MatrixA[max_row], MatrixA[col]

    # Метод для розв'язку СЛАР
    def solve(self):
        MatrixA = deepcopy(self.__MatrixA)
        n = self.__n
        for k in range(n - 1):
            self.__maxinrow(k)
            for i in range(k + 1, n):
                div = MatrixA[i][k] / MatrixA[k][k]
                MatrixA[i][n] -= div * MatrixA[k][n]

```

```

    for j in range(k, n):
        MatrixA[i][j] -= div * MatrixA[k][j]
res = [0 for i in range(n)]
for k in range(n - 1, -1, -1):
    res[k] = (MatrixA[k][n] - sum([MatrixA[k][j] * res[j] for j in range(k + 1, n)]))
/ MatrixA[k][k] # Почергове знаходження коренів рівняння
return res

```