НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Aвтоматизованих систем обробки інформації і управління

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з Основи програмування

(назва дисципліни)

на тему: Розв'язання СЛАР точними методами

Студента I курсу ІП-51 групи

напряму підготовки 6.050103 «Програмна інженерія»

спеціальності «Програмне забезпечення систем»

Булатова Д.Є.

(прізвище та ініціали)

Керівник Головченко Максим Миколайович

Асистент

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

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Члени комісії |  |  | Муха І.П. |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |
|  |  |  | Головченко М.М. |
|  | (підпис) |  | (вчене звання, науковий ступінь, прізвище та ініціали) |

Київ - 2016 рік

Додаток Г Лист завдання

Національний технічний університет України “КПІ”

(назва вищого навчального закладу)

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Основи програмування

Напрям "Програмна інженерія"

Курс 1 Група ІП-51 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Булатова Дмитра Єгоровича |

(прізвище, ім’я, по батькові)

|  |  |
| --- | --- |
| 1. Тема роботи | Розв'язання СЛАР точними методами |
|  | |
|  | |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченої роботи |  |

|  |  |
| --- | --- |
| 3. Вихідні дані до роботи | Технічне завдання (додаток А) |
|  | |
|  | |
|  | |

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

|  |
| --- |
|  |
|  |
|  |
|  |

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

|  |
| --- |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
| 6. Дата видачі завдання | 17 березня 2016 року |

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи | 14.03.16-20.03.16 |  |
| 2. | Підготовка ТЗ | 21.03.16-03.04.16 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 04.04.16-17.04.16 |  |
| 4. | Розробка алгоритму вирішення задачі | 18.04.16-24.04.16 |  |
| 6. | Узгодження алгоритму з керівником | 18.04.16-24.04.16 |  |
| 5. | Розробка сценарію роботи програми | 25.04.16-01.05.16 |  |
| 6. | Узгодження сценарію роботи програми з керівником | 25.04.16-01.05.16 |  |
| 7. | Узгодження з керівником інтерфейсу користувача | 25.04.16-01.05.16 |  |
| 8. | Розробка програмного забезпечення | 02.05.16-22.05.16 |  |
| 9. | Налагодження розрахункової частини програми | 02.05.16-22.05.16 |  |
| 10. | Розробка та налагодження інтерфейсної частини програми | 23.05.16-29.05.16 |  |
| 11. | Узгодження з керівником набору тестів для контрольного прикладу | 23.05.16-29.05.16 |  |
| 12. | Тестування програми | 30.05.16-05.06.16 |  |
| 13. | Підготовка пояснювальної записки | 06.06.16-12.06.16 |  |
| 14. | Здача курсової роботи на перевірку | 06.06.16-12.06.16 |  |
| 15. | Захист курсової роботи | 09.06.16 |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Булатов Д.Є. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| Керівник |  |  | Головченко М.М. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 р.

**АНОТАЦІЯ**

Пояснювальна записка до курсової роботи: 70 сторінок, 20 рисунків, 12 таблиць, 3 посилання.

**Об’єкт дослідження:** розв’язання СЛАР.

**Мета роботи:** дослідження методів розв’язання СЛАР методами LUP, Матричним та Обертання.

Дана курсова робота включає в себе: опис методів, застосування методу до конкретного завдання, код програми вирішення перерахованих вище методів на мові програмування С++, а також описання детального процесу розв’язання кожного з них.

Визначено порівняльну характеристику швидкості роботи методів. У висновках узагальнено підсумок проведеної роботи.

ВИРІШЕННЯ СЛАР, LUP МЕТОД, МАТРИЧНИЙ МЕТОД, МЕТОД ОБЕРТАННЯ, ОБЕРЕНА МАТРИЦЯ.

**Зміст**

[Вступ 6](#_Toc419641933)

[1 Постановка задачі 6](#_Toc419641934)

[2 Теоретичні відомості 8](#_Toc419641935)

[3 Опис алгоритмів 14](#_Toc419641936)

[3.1. Загальний алгоритм 14](#_Toc419641937)

[3.2. Алгоритм LUP методу 17](#_Toc419641938)

[3.3. Алгоритм Матричного методу 18](#MatrixMethod)

[3.4. Алгоритм Методу обертань 18](#RotationMethod)

[4 Опис програмного забезпечення 19](#_Toc419641939)

[4.1. Функціональна структура програмного забезпечення 20](#_Toc419641940)

[4.2. Опис функцій частин програмного забезпечення 21](#_Toc419641941)

[4.2.1. Користувацькі функції 21](#_Toc419641942)

[4.2.2. Стандартні функції 24](#_Toc419641943)

[5 Тестування програмного забезпечення 25](#_Toc419641944)

[5.1. План тестування 25](#_Toc419641945)

[5.2. Приклади тестування 25](#_Toc419641946)

[6 Інструкція користувача 34](#_Toc419641947)

[7 Аналіз результатів 40](#_Toc419641948)

[Висновки 45](#_Toc419641949)

[Перелік посилань 46](#_Toc419641950)

**Вступ**

Метою данної курсової роботи було створення прикладної компьютерної программи для вирішення певної задачі. В ході роботи використовувались інтегроване середовище розробки Microsoft Visual Studio Community 2015. Программа була написана на мові C++. Також у ході роботи використовувались .Net бібліотеки для роботи з віконною системую операційної системи Microsoft Windows.

**1 Постановка задачі**

Дана система лінійних алгебраїчних рівнянь з n рівнянь. Треба знайти розв’язок системи або вказати що він не існує. Задачу треба вирішити 3ма методами: LUP метод, Матричний метод та метод обертання. Порахувати кількість арифметичних операцій необхідних для виконання программи. Для систем з 2х рівнянь також вивести графік що демонструє точку перетину рівнянь 2х прямих (тобто точку розв’язку)

**2 Теоретичні відомості**

**2.1 LUP – метод**

LUP-розклад матриці — це представлення в формі

Алгоритм LUP-розкладу

Знаходження LUP-розкладу проводиться в дукілька кроків. Нехай матрица C - копія матриці А. На кожоному i-м кроці спочатку виконуємо пошук опорного элементу - максимального по модулю элемента серед елементів i-го стовпчика, що знаходиться не вище i-го рядку, після чого рядок з опорним елементом міняєтся місцями з i-м рядком. Одночасно проводиться такий же обмік в матриці Р. При цьому, якщо матриця невироджена, то опорный елемент гарантованно буде відрізнятись від нуля. После цього всі елементи поточного i-го рядку, що знаходяться нижче i-го рядку, діляться на опорний. Далі з усіх элементов ща знаходяться нижче i-го рядку и i-го стовпчику (тобто таких що j>i і k>i) віднімається добуток . Після цього лічильник i збільшується на 1 и процесс повторюється поки i<n де n — розмірність вихідної матриці. Після того як всі кроки будут виконані матриця C буде представляти собою наступну сумму:

де E — одинична матриця.

де *L* та *U* — нижня та верхня трикутна матриця того ж розміру, а *P* — матриця перестановки.

Якщо в рівнянні

задано *A* та *b*. Тоді розв'язок отримується в два кроки:

1. Розв'язуємо рівняння  і знаходимо *y*
2. Розв'язуємо рівняння  і знаходимо *x*.

Ці рівняня ми можемо розв’язати завдяки тому що матриці L та U трикутні. Завдяки цій властивості ми одразу можемо викинути з розгляду 1 змінну так як знаємо її значення. А після підстановки цієї змінної у інші рівняня зможемо знайти ще 1 змінну значення котрої знаходиться очевидним чином. Продовжуючи такі дії ми знайдемо усі шукані змінні.

[1]

Для вирішення рівняня виду Ly=b (L - трикутна матриця, y та b - матриці ширини 1) треба вирішити систему з n рівнянь де в i’му рівняні присутні i змінних.

Таким чином отримаємо систему

З системи отримуємо

Підставивши значення в інші рівняня системи отримуємо наступну систему з n-1 рівняня

Візьмемо

Отримаємо систему

Далі продовжуючи такі самі дії як з системою 2.1.5 можемо отримати значення всіх змінних тобто вирішити рівняня виду Ly=b (Що дозволяє вирішити рівняня 2.1.3, 2.1.4)

**2.2 Матричний метод**

Позначимо через X – матрицю-стовпець невідомих, через B– матрицю-стовпець вільних членів. Тоді попередню систему рівнянь можна записати у вигляді матричного рівняння:

Якщо квадратна матриця має відмінний від нуля визначник , то для неї існує обернена . Помноживши зліва в цьому рівнянні на , одержимо

Враховуючи, що добуток оберненої матриці на саму матрицю дає одиничну , а також формулу , одержимо матричний розв'язок системи

[2]

**2.3 Метод обертань**

Розглянемо спочатку метод обертань з допомогою якого будемо відшукувати розв'язок системи лінійних рівнянь наступного виду:

Прямий хід методу обертання.

На першому етапі прямого ходу здійснюємо виключення невідомої qr_rozklad_matruci4 з усіх рівнянь системи (2.3.1) крім першого. Для виключення qr_rozklad_matruci4 з другого рівняння обчислюють наступні коефіцієнти:

для яких обов'язковою вимогою є виконання наступних умов  . Після цього, перше рівняння системи (2.3.1) замінюють лінійною комбінацією першого і другого рівняння з коефіцієнтами  та , а друге рівняння — аналогічною лінійною комбінацією з коефіцієнтами  та  . В результаті виконання даних дій система (2.3.1) набуде наступного вигляду:

де

.

Неважко переконатися, що перетворення системи (2.3.1) до виду (2.3.4) еквівалентно множенню зліва матриці A і вектора b на матрицю , де

Виходячи з того, що коефіцієнти  і  матриці  підібрані таким чином, що , то їх можна представити наступним чином: . Тоді матриця  являтиме собою **матрицю обертання** на кут  в площині . Неважко догадатися, що саме з таких міркувань появилась назва даного методу.

Відмітимо, що у випадку коли коефіцієнт  здійснювати виключення невідомої  з другого рівняння системи немає необхідності. В цьому випадку покладають, що  і .

Далі переходимо до виключення невідомої qr_rozklad_matruci4 з третього рівняння. Для цього, як і на попередньому кроці, обчислюємо коефіцієнти виду:

для яких також повинні виконуватись наступні умови . Після цього, перше рівняння системи (2.3.4) замінюємо лінійною комбінацією першого і третього рівняння з коефіцієнтами  та  і третє рівняння — аналогічною лінійною комбінацією з коефіцієнтами  та  . Таким чином ми отримали систему, значення коефіцієнтів при невіломій  у другому і третьому рівнянні рівні нулю. Дане перетворення системи еквівалентне множенню зліва її матриці коефіцієнтів на наступну **матрицю обертання**:

Аналогічним чином проводимо виключення невідомої  з решти рівнянь системи. В результаті виконання першого етапу, який складається з (n-1)-гокроку система (2.3.1) зводиться до наступного вигляду:

або у матрично-векторній формі , де .

Далі переходимо до другого етапу (складається  з  -х кроків) з допомогою якого з рівнянь під номерами  системи (2.3.7)  здійснюємо виключення невідомої . Для цього, кожне -те рівняння замінюють лінійною комбінацією  -го рівняння з другим. В результаті виконання даного етапу система (2.3.7) прийме наступний вигляд:

або  , де  .

І так далі продовжуючи даний процес, після виконання -го етапу система лінійних рівнянь прийме наступний вигляд:

або у матрично-векторній формі  , де  .

Позначивши через R верхню трикутну матрицю, отриману з допомогою прямого ходу **методу обертань**  , де T — матриця результуючого обертання, являється ортогональною, як добуток ортогональних матриць. Отримуємо QR — розклад матриці A, де  .

Обернений хід метоту обертань.

Обернений хід **метоту обертань** здійснюється аналогічно як і у **методі Гаусса**. Тобто з останнього рівняння системи (9) знаходимо значення невідомої qr_rozklad_matruci60. Після чого решта значень  обчислюють за наступною формулою:

[3]

**3 Опис алгоритмів**

3.1 Загальний алгоритм

3.1.1 Кнопка “Сгенерировать”

1.Змінній n присвоюэться значення введене в поле Number

2. Задання випадкових значень таблиці SLAR

3.1.2 Кнопка “Сохранить результат”

1. ЯКЩО ні одна з таблиць LUPSol MatrixSol RotationSol не видима сповіщаємо про помилку і виходимо з функціі
2. Відкриваємо файл output.txt для запису
3. Виводимо в файл умову системи
4. Якщо видима таблиця LUPSol виводимо в файл рішення з цієї таблиці, кількість оперіцій алгоритму, час роботи алгоритму і кількість ітерацій
5. Якщо видима таблиця MatrixSol виводимо в файл рішення з цієї таблиці, кількість оперіцій алгоритму, час роботи алгоритму і кількість ітерацій
6. Якщо видима таблиця RotationSol виводимо в файл рішення з цієї таблиці, кількість оперіцій алгоритму, час роботи алгоритму і кількість ітерацій
7. Закриваємо файл

3.1.3 Кнопка “Решить”

1. Змінній n присвоюэться значення введене в поле Number

2. Змінній badinput присвоюємо значення false

3. Створюємо матрицю solcopy

4. За допомогою структури try {} catch() {} намагаємося зчитати данні з таблиці SLAR і у разі помилки обробляємо виключну ситуацію

4.1 Зчитуємо таблицю SLAR у матриці SLARmatrix та fr (SLARmatrix – матриця коефіцієнтів fr – матриця вільних членів)

4.2 При виникненні помилки змінній badinput присвоюємо значення true

5. ЯКЩО детермінант матриці SLARmatrix (Система не сумісна) або badinput або не вибраний ні один метод вирішення то повідомляємо про помилку

5.1 ЯКЩО badinput повідомляємо що сталася помилка введення

5.2 ІНАКШЕ ЯКЩО не вибран ні один метод вирішення повідомляємо що не вибраний ні один метод

5.3 ІНАКШЕ повідомляємо що система не сумісна

5.4 Робимо таблиці LUPSol MatrixSol та RotationSol невидимими

5.5 Встановлюємо ширину вікна на 606

5.6 Вимикаємо графік

6. ІНАКШЕ вирішуємо систему

6.1 Очищаємо повідомлення про помилку

6.2 Якщо вибраний LUP метод використовуємо його для вирішення системи

6.2.1 Обнуляємо кількість операцій та ітерацій

6.2.2 Зберігаємо час початку виконання

6.2.3 Вирішуємо задачу за допомогою LUP методу

6.2.4 Зберігаємо і виводимо час роботи як час закінчення виконання мінус час початку

6.2.5 Зберігаємо і виводимо кількість операцій та ітерацій

6.2.6 Зберігаємо копію розв’язку у матриці solcopy

6.2.7 Робимо таблицю LUPSol видимою

6.2.8 Заносимо відповідь у таблицю LUPSol

6.3 ІНАКШЕ Таблицю LUPSol робимо невидимою та очищаємо поля виведення кількості операцій, часу виконання методу та кількості ітерацій

6.4 Якщо вибраний матричний метод використовуємо його для вирішення системи

6.4.1 Обнуляємо кількість операцій та ітерацій

6.4.2 Зберігаємо час початку виконання

6.4.3 Вирішуємо задачу за допомогою матричного методу

6.4.4 Зберігаємо і виводимо час роботи як час закінчення виконання мінус час початку

6.4.5 Зберігаємо і виводимо кількість операцій і ітерацій

6.4.6 Зберігаємо копію розв’язку у матриці solcopy

6.4.7 Робимо таблицю MatrixSol видимою

6.4.8 Заносимо відповідь у таблицю MatrixSol

6.5 ІНАКШЕ Таблицю MatrixSol робимо невидимою та очищаємо поля виведення кількості операцій та часу виконання методу

6.6 Якщо вибраний матричний метод використовуємо його для вирішення системи

6.6.1 Обнуляємо кількість операцій та ітерацій

6.6.2 Зберігаємо час початку виконання

6.6.3 Вирішуємо задачу за допомогою методу обертання

6.4.4 Зберігаємо і виводимо час роботи як час закінчення виконання мінус час початку

6.6.5 Зберігаємо і виводимо кількість операцій та ітерацій

6.6.6 Зберігаємо копію розв’язку у матриці solcopy

6.6.7 Робимо таблицю RotationSol видимою

6.6.8 Заносимо відповідь у таблицю RotationSol

6.7 ІНАКШЕ Таблицю RotationSol робимо невидимою та очищаємо поля виведення кількості операцій та часу виконання методу

6.8 ЯКЩО n=2 виводимо графік

6.8.1 Розширюємо вікно щоб звільнити місце під графік

6.8.2 Вмикаємо графік

6.8.3 Очищуємо графіки

6.8.4 ЯКЩО 2й коефіцієнт 1го рівняня нульовий задаємо 1й графік як вертикальну пряму

6.8.5 ІНАКШЕ будуємо графік по 2м точкам

6.8.6 ЯКЩО 2й коефіцієнт 2го рівняня нульовий задаємо 2й графік як вертикальну пряму

6.8.7 ІНАКШЕ будуємо графік по 2м точкам

6.9 ІНАКШЕ встановлюємо початкову ширину вікна і вимикаємо графік

3.2 Алгоритми методів

3.2.1 Алгоритм LUP методу

1. У матрицю C копіюємо матрицю коефіцієнтів
2. Для значень I від 0 до n-1
   1. Задаємо змінні OpV = 0 Op = -1
   2. Для значень n від I до n-1
   3. ЯКЩО C[j][i] більше по модулю за OpV то присвоюємо OpV = fabs(c[j][i]) Op=j
   4. Міняємо місцями рядки I та Op в в кожній з матриць C та fr
   5. Для значень j від i+1 до n-1
      1. Ділимо С[j][i] на C[i][i]
      2. Для значень k від i+1 до n-1
         1. від C[j][k] віднімаємо C[j][i] помножене на C[i][k]
3. Задаємо матрицю L як одиничну N x N а U як нульову N x N
4. Копіюємо значення матриці SLAR знизу від головної діагоналі у матрицю L не включаючи головну діагональ
5. Копіюємо значення матриці SLAR зверху від головної діагоналі у матрицю U включаючи головну діагональ
6. Задаємо матрицю y як матрицю N x 1
7. Вирішуємо
   1. Для значень I від 0 до n-1
      1. Для значень J від i+1 до n-1
         1. Віднімаємо від J-го рядка матриць L та fr I-й за таким коефіцієнтом щоб i-й елемент j-го рядка після віднімання дорівнював нулю
   2. Задаємо y[i] як fr[i][0] поділене на L[i][i]
8. Вирішуємо
   1. Для значень I від 0 до n-1
      1. Для значень J від i+1 до n-1
         1. Віднімаємо від J-го рядка матриць U та y I-й за таким коефіцієнтом щоб i-й елемент j-го рядка після віднімання дорівнював нулю
   2. Задаємо x[i] як y[i][0] поділене на U[i][i]
9. Повертаємо x

3.2.2 Алгоритм Матричного методу

1. Знаходимо обернену матрицю матриці коефіцієнтів
   1. Створюємо приєднану матрицю матриці коефіцієнтів
   2. Для i від 0 до кількості рядків

1.2.1 Міняємо місцями i-й рядок з рядком в якому i-й коефіцієнт найбільший серед інших рядків

* + 1. Для j від i+1 до кількості рядків

1.2.2.1 До j-го рядка додаємо i-й з таким коефіцієнтом щоб позбутись i-го коефіцієнта в j-му рядку

* 1. Для i від кількості рядків до 0
     1. Для j від i-1 до 0

1.3.1.1 До j-го рядка додаємо i-й з таким коефіцієнтом щоб позбутись i-го коефіцієнта в j-му рядку

1.4 Ділимо кожний рядок на коефіцієнт що дорівнює його номеру

1.5 Отримана приєднана матриця буде оберненою

2. Домножаємо обернену матрицю на матрицю коефіцієнтів зліва

3. Повертаємо отриману матрицю

3.2.3 Алгоритм Методу обертань

1. Для значень I від 0 до n-1
   1. Для значень J від i+1 до n-1

1.1.1 Заміняємо I-е та J-е рівняня на їх лінійні комбінації.

Нове i-е рівняня =   множити на i-е рівняня + помножити на j-е рівняня, Нове j-е рівняня =   множити на i-е рівняня +  помножити на j-е рівняня

1. Для значень I від n-1 до 0
   1. Для значень J від i-1 до 0
      1. Віднімаємо від J-го рядка матриць SLAR та fr I-й за таким коефіцієнтом щоб i-й елемент j-го рядка після віднімання дорівнював нулю
2. Задаємо матрицю sol як матрицю N x 1
3. Задаємо sol[i][0] як fr[i][0] поділене на SLAR[i][i]
4. Повертаємо sol

**4 Опис програмного забезпечення**

**4.1 Функціональна структура програмного забезпечення**

Програмний додаток містить головний файл, 1 файл реалізації, 1 заголовний файл та заголовний файл для реалізації інтерфейсу (рисунок 4.1).

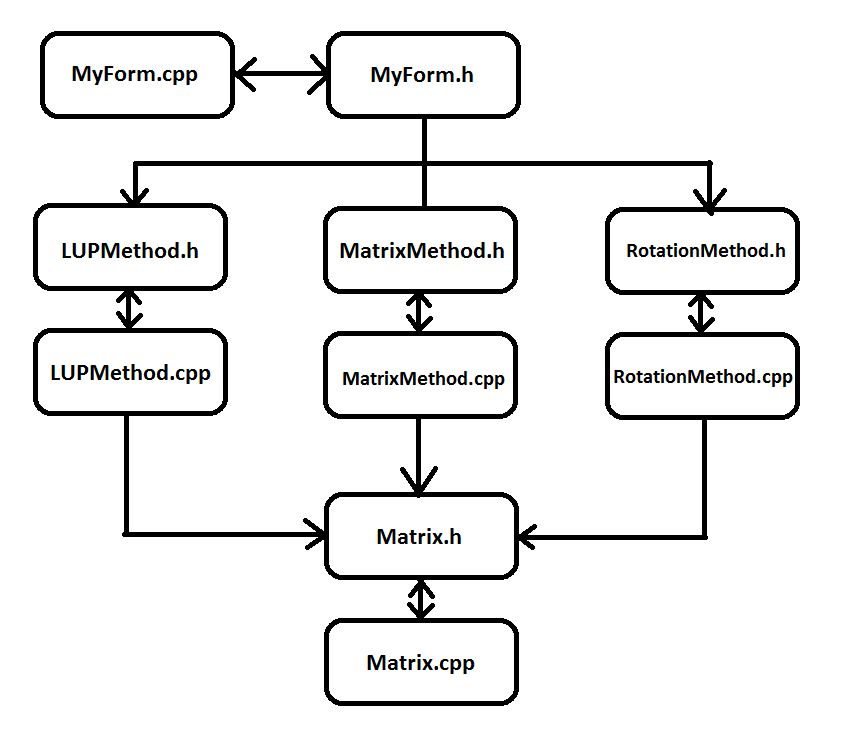


Рисунок 4.1 – Функціональна структура

**Опис функцій частин програмного забезпечення**

Программа поділена на наступні модулі:

* MyForm.h - реалізує інтерфейс користувача
* Matrix.h - реалізує структуру matrix для збереження і обробки матриць
* LUPMethod.h - Реалізує LUP метод
* MatrixMethod.h - реалізує матричний метод
* RotationMethod.h - реалізує метод обертань

**4.2 Опис функцій частин програмного забезпечення**

4.2.1 Користувацькі функції (Таблиця 4.1)

Таблиця 4.1 – Користувацькі функції

| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | matrix | makeidendity | Робить матрицю одиничною | Немає | Немає | matix.h |
| 2 | matrix | swaprows | Міняє місцями рядки | Int i,int j | Немає | matix.h |
| 3 | matrix | tr | Транспонує матрицю | Немає | Немає | matix.h |
| 4 | matrix | simplerows | Віднімає від b-го рядка a-й з таким коефіцієнтом щоб обнулити matrix[b][i] | Int a,int b,int i | Немає | matix.h |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 6 | matrix | transformrows | Змінює рядки згідно з алгоритмом методу обертання | int r1, int r2, int st, matrix &fr | Немає | matix.h |
| 7 | matrix | det | Рахує детермінант | Немає | double | matix.h |
| 8 | matrix | matrix | Ініціалізує матрицю с заданою висотою и шириною | Int dw,int dh | matrix | matix.h |
| 9 | matrix | operator[] | Дозволяє звертатися до єлементів матриці напряму | Int a | &std::vector < double > | matix.h |
| 10 | matrix | operator\* | Множить матрицю на число | double a | matrix | matix.h |

Продовження таблиці 4.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 12 | matrix | operator/ | Ділить матрицю на число | double a | matrix | matix.h |
| 13 | matrix | inverse | Повертає обернену матрицю | Немає | matrix | matix.h |
| 14 | - | LUP | Вирішує систему за допомогою LUP метода | matrix SLAR, matrix fr, int n | matrix | LUPMethod.h |
| 15 | - | MatrixMethod | Вирішує систему за допомогою Матричного метода | matrix SLAR, matrix fr | matrix | MatrixMethod.h |
| 16 | - | RotationMethod | Вирішує систему за допомогою метода обертань | matrix SLAR, matrix fr, int n | matrix | RotationMethod.h |

4.2.2 Стандартні функції (Таблиця 4.2)

Таблиця 4.2 - Стандартні функції

| № п/п | Назва классу | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | MessageBox | Show | Виведення повідомлення на єкран | String text, String caption | Немає | CLR |
| 2 | Ofstream | open | Відкриває файлу | Char\* filename, int mode | немає | Fstream.h |
| 3 | Ofstream | close | Закриває файл | немає | немає | Fstream.h |
| 4 | Convert | ToDouble | Конвертує данні у тип double | System::Object | double | CLR |
| 5 | Convert | ToString | Конвертує данні у тип String | int | double | CLR |
| 6 | - | srand | Ініціалізує генератор випадкових чисел | Unsigned int sid | немає | Stdlib.h |
| 7 | - | rand | Генерує випадкове число | Немає | int | Stdlib.h |

**5 Тестування програмного забезпечення**

5.1 План тестування

Программа має обробляти усі виключні випадки, давати кореткну відповідь на всіх корректних наборах данних та коректно обробляти помилку на всіх некоректних наборах данних

1. Тестування правильності введених значень.
   1. Тестування при введенні некоректних символів.
   2. Тестування при введенні замалих та завеликих значень.
2. Тестування коректної роботи при введені систем, що не мають коренів.
   1. Тестування роботи програми при нульовому значенні визначника.
3. Тестування коректності роботи методів.
   1. Перевірка коректності роботи LUP методу.
   2. Перевірка коректності роботи Матричного методу.
   3. Перевірка коректності роботи Методу обертань.
4. Тестування побудови графіків.
5. Тестування випадкової генерації
6. Тестування эфективності

Приклади тестування

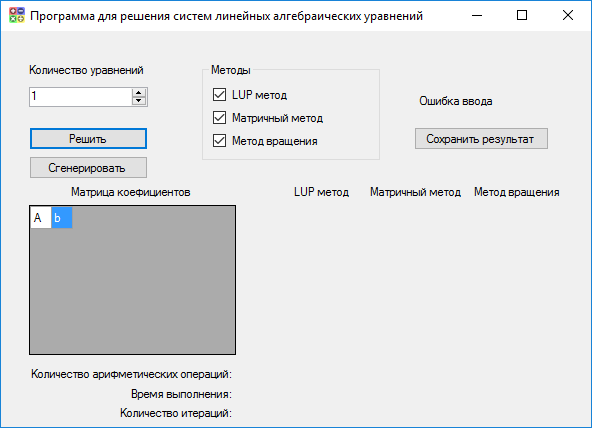
1. Тестування правильності введених значень.
   1. Тестування при введенні некоректних символів (Таблиця 5.1)

Таблиця 5.1 - Тестування при введенні некоректних символів

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення некоректних символів |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 1  A b |

Продовження таблиці 5.1

|  |  |
| --- | --- |
| Схема проведення тесту | Поелементне заповнення матриці коефіцієнтів |
| Очікуваний результат | Повідомлення про помилку  формату даних |
| Стан програми після проведення випробувань | Виведено помилку «Ошибка ввода» (Рисунок 5.1) |

Рисунок 5.1 - Тестування при введенні некоректних символів

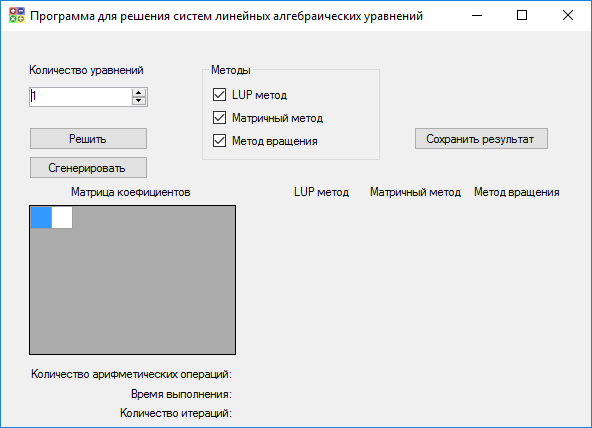
* 1. Тестування при введенні замалих та завеликих значень. (Таблиця 5.2)

Таблиця 5.2 - Тестування при введенні замалих та завеликих значень

|  |  |
| --- | --- |
| Мета тесту | Перевірити можливість введення замалих та завеликих значень |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 0 |
| Схема проведення тесту | Змінення значення кількості рівнянь |

Продовження таблиці 5.2

|  |  |
| --- | --- |
| Очікуваний результат | Неможливість введення недодатнього числа |
| Стан програми після проведення випробувань | Неможливо ввести недодатне число (Рисунок 5.2) |

Рисунок 5.2 - Тестування при введенні замалих та завеликих значень

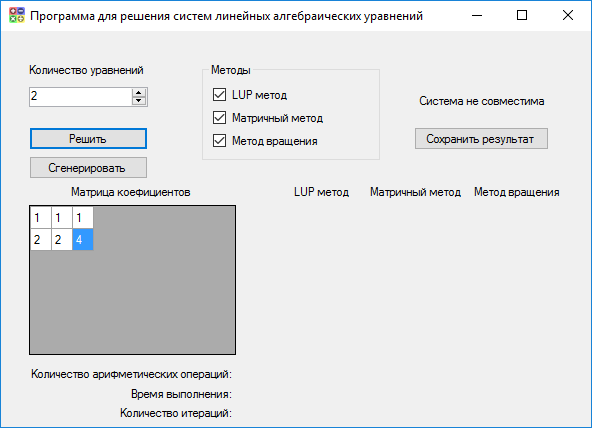
1. Тестування коректної роботи при введені систем, що не мають коренів
   1. Тестування роботи програми при нульовому значенні визначника (Таблиця 5.3)

Таблиця 5.3 - Тестування роботи програми при нульовому значенні визначника

|  |  |
| --- | --- |
| Мета тесту | Перевірити роботу при відсутності коренів |
| Початковий стан програми | Відкрите вікно програми |

Продовження таблиці 5.3

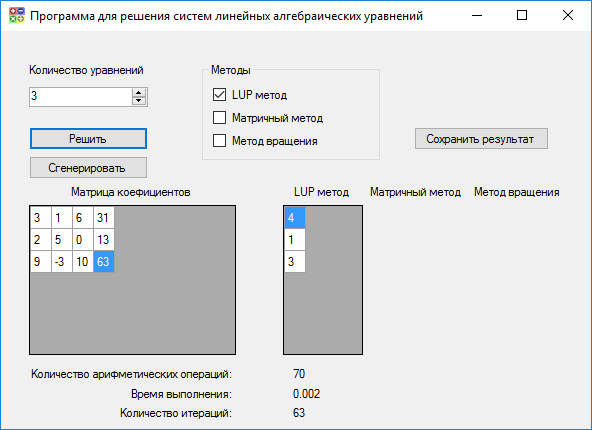
|  |  |
| --- | --- |
| Вхідні дані | 2  1 1 1  2 2 4 |
| Схема проведення тесту | Поелементне заповнення матриці коефіцієнтів |
| Очікуваний результат | Повідомлення про помилку |
| Стан програми після проведення випробувань | Виведено помилку «Система не совместима» (Рисунок 5.3) |

Рисунок 5.3 - Тестування роботи програми при нульовому значенні визначника

1. Тестування коректності роботи методів
   1. Перевірка коректності роботи LUP методу (Таблиця 5.4)

Таблиця 5.4 - Перевірка коректності роботи LUP методу

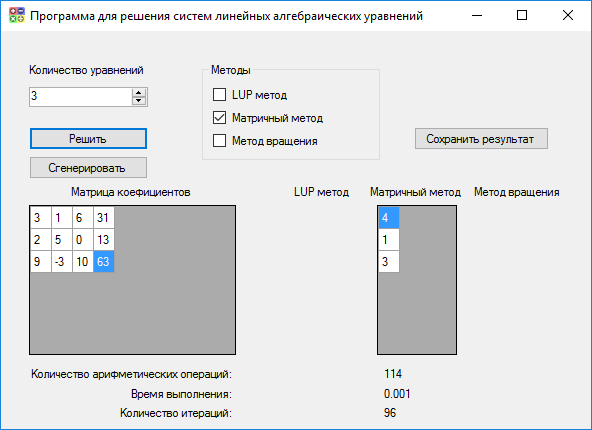
|  |  |
| --- | --- |
| Мета тесту | Перевірити роботу LUP методу |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 3  3 1 6 31  2 5 0 13  9 -3 10 63 |
| Схема проведення тесту | Поелементне заповнення матриці коефіцієнтів |
| Очікуваний результат | 4 1 3 |
| Стан програми після проведення випробувань | Виведена відповідь 4 1 3 |

Рисунок 5.4 - Перевірка коректності роботи LUP методу

* 1. Перевірка коректності роботи Матричного методу (Таблиця 5.5)

Таблиця 5.5 - Перевірка коректності роботи LUP методу

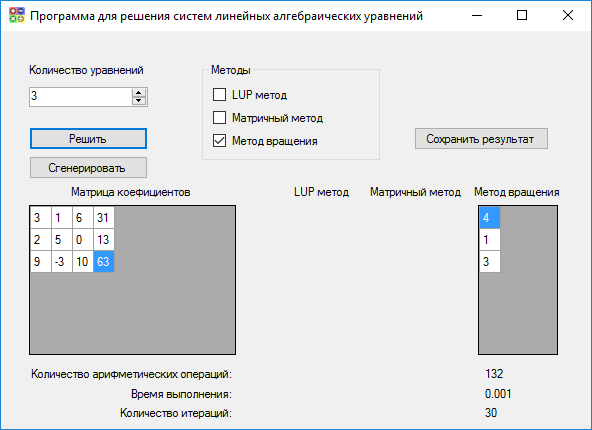
|  |  |
| --- | --- |
| Мета тесту | Перевірити роботу Матричного методу |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 3  3 1 6 31  2 5 0 13  9 -3 10 63 |
| Схема проведення тесту | Поелементне заповнення матриці коефіцієнтів |
| Очікуваний результат | 4 1 3 |
| Стан програми після проведення випробувань | Виведена відповідь 4 1 3 (Рисунок 5.5) |

Рисунок 5.5 - Перевірка коректності роботи LUP методу

* 1. Перевірка коректності роботи Методу обертань (Таблиця 5.6)

Таблиця 5.6 - Перевірка коректності роботи Методу обертань

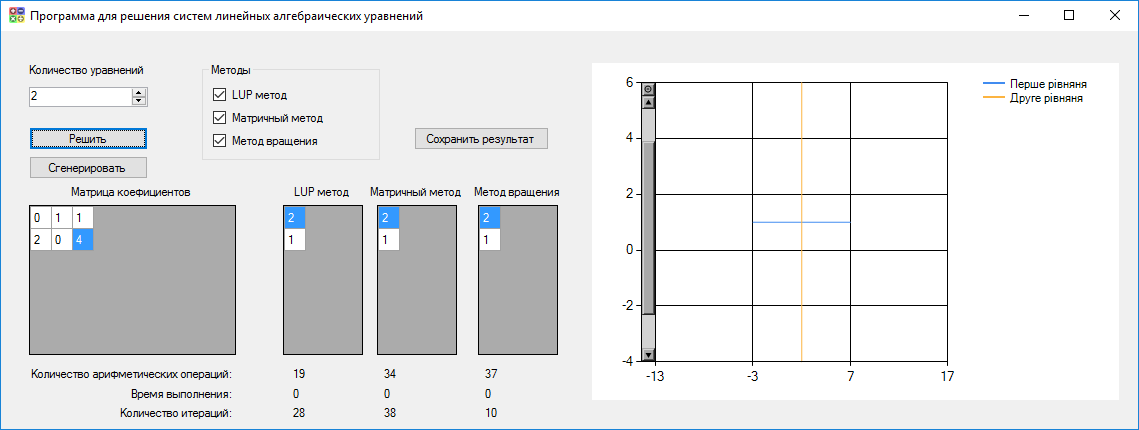
|  |  |
| --- | --- |
| Мета тесту | Перевірити роботу Методу обертань |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 3  3 1 6 31  2 5 0 13  9 -3 10 63 |
| Схема проведення тесту | Поелементне заповнення матриці коефіцієнтів |
| Очікуваний результат | 4 1 3 |
| Стан програми після проведення випробувань | Виведена відповідь 4 1 3 (Рисунок 5.6) |

 Рисунок 5.6 - Перевірка коректності роботи Методу обертань

1. Тестування побудови графіків (Таблиця 5.7)

Таблиця 5.7 – Тестування побудови графіків

|  |  |
| --- | --- |
| Мета тесту | Перевірити побудову графіків |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 2  0 1 1  2 0 4 |
| Схема проведення тесту | Поелементне заповнення матриці коефіцієнтів |
| Очікуваний результат | 2 перпендикулярних графіки зі спільною точкою (2;1) |
| Стан програми після проведення випробувань | Побудовано відповідний графік (Рисунок 5.7) |

Рисунок 5.7 - Тестування побудови графіків

1. Тестування випадковії генерації (Таблиця 5.8)

Таблиця 5.8 - Тестування випадковії генерації

|  |  |
| --- | --- |
| Мета тесту | Перевірити побудову графіків |
| Початковий стан програми | Відкрите вікно програми |
| Вхідні дані | 3 |
| Схема проведення тесту | Нажимання на кнопку “Сгенерировать” |

Продовження таблиці 5.8

|  |  |
| --- | --- |
| Очікуваний результат | Таблиця коефіцієнтів заповниться випадковими числами |
| Стан програми після проведення випробувань | Матриця коефіцієнтів заповнилась випадковими значеннями (Рисунок 5.8) |

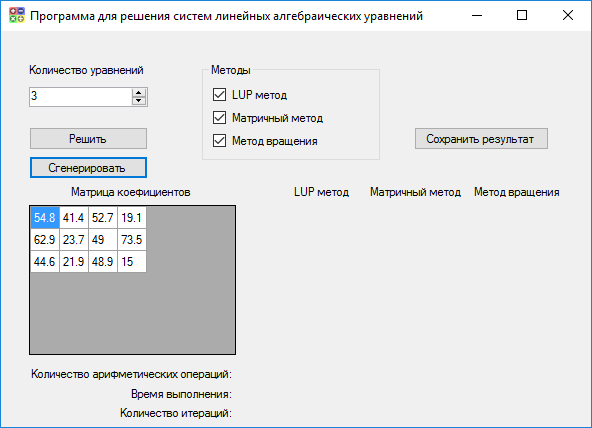


Рисунок 5.8 - Тестування випадковії генерації

**6 Інструкція користувача**

Робота з програмою

* 1. Відкрити программу (Рисунок 6.1)

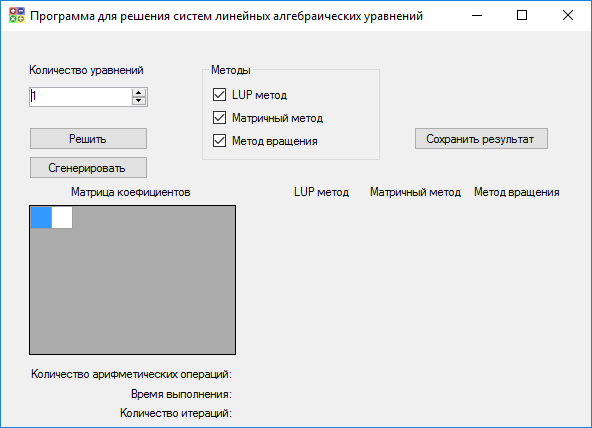


Рисунок 6.1 – Початковий стан вікна программи

* 1. Ввести необхідну кількість рівнянь (Рисунок 6.2)

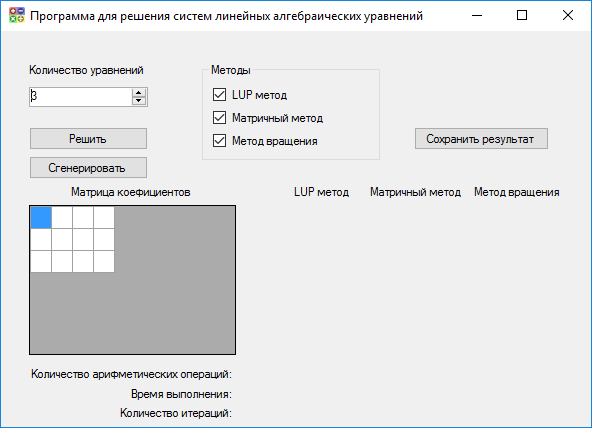


Рисунок 6.2 – Введення кількості рівнянь

* 1. Заповнити Матрицю коефіцієнтів або нажати на кнопку “Сгенерировать” (коефіцієнти що відповідають пустим клітинкам вважаються нульовими)

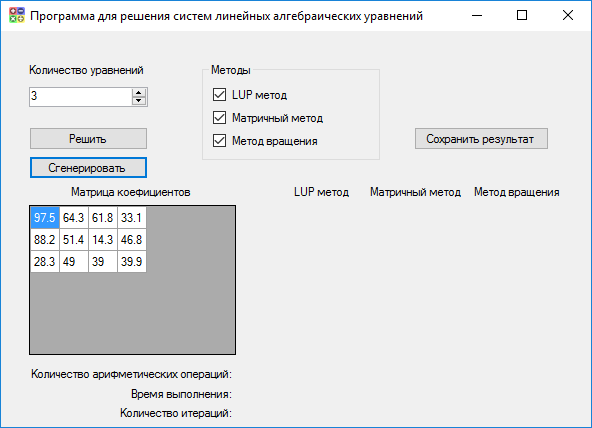


Рисунок 6.3 – Заповнення матриці коефіцієнтів

* 1. Вибрати мінімум 1 метод вирішення

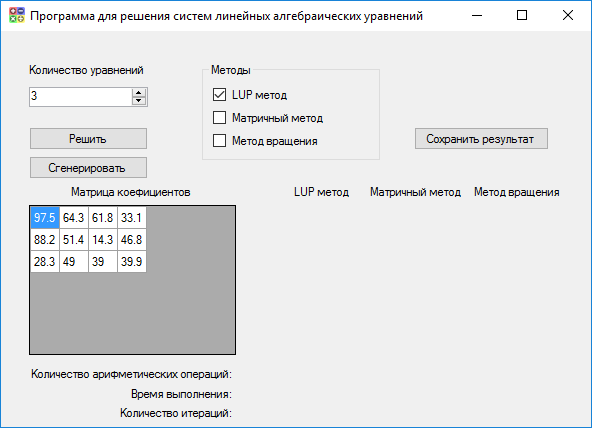


Рисунок 6.4 – Вибір методів

* 1. Отримати результат

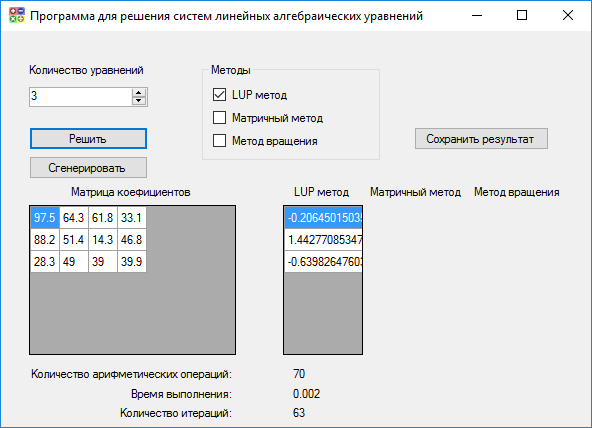


Рисунок 6.5 – Отримання результатів

* 1. За бажанням зберігти результат

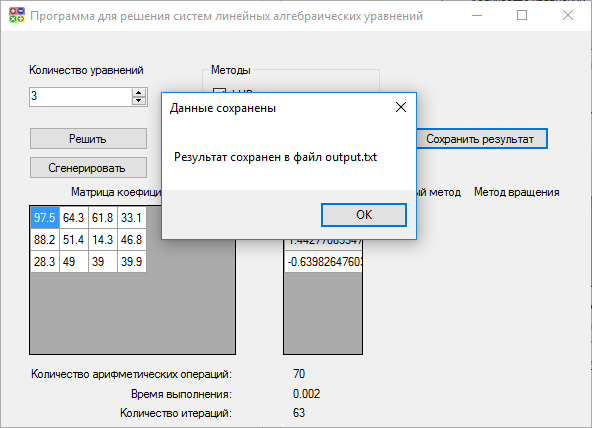


Рисунок 6.6 – Повідомлення про успішне збереження

Формат вхідних та вихідних даних

Вхідні данні :

Кількість рівнянь – ціле число від 1 до 500

Коефіцієнти – дійсні числа

Вихідні данні:

Розв’язки – Дійсні числа

Системні вимоги

Операційна система Windows 7 або новіша

Єкран з роздільною здатністю від 1366x768

.NET Framework версії 4.5.2 або новіше

**7 Аналіз і узагальнення результатів**

Данне програмне забезпечення вирішує СЛАР точними методами обробляє всі виключні ситуації а за потреби зберігає результати в файл та будує графіки рівнянь.

В результаті тестувань помилок в роботі программи не виявлено.

Аналіз ефективності роботи программи

* 1. Теоретичне обгрунтування складності
     1. LUP метод

Для знаходження розв’язку LUP метод Спочатку розкладає матрицю на L U та P матриці. Для знаходження розкладу потребується O() часу тому що алгоритм знаходження розкладу для кожного рядку виконує дію над кожним елементом що знаходиться справа знизу від елемента тобто результуюча кількість операцій для знаходження розкладу буде дорівнювати

Для знажодження розв’язків рівнянь та також треба O() часу. Для кожного рядка віднімаємо цей рядок від усіх наступних. Таким чином кількість операцій буде дорівнювати

Отже загальна асимптотика буде рівною O()

В коді використовується 3 вкладені цикли так що на практиці асимптотика співпадає з теоретичною

* + 1. Матричний метод

Цей метод складається з 2х частин, а саме множення оберненої матриці на матрицю вільних членів та пошук оберненої матриці.

Множення оберненої матриці на матрицю вільних членів очевидно потребує O() часу.

Порахуємо кількість дій необхідних для пошуку оберненої матриці.

Ми виконуємо операції над матрицею та приєднаною до неї одиничною матрицею тому розмір матриці над якою будуть проводитись всі подальші операції n x 2n

Для кожного рядку ми віднімаємо цей рядок від усіх наступних з деякими коефіцієнтами. Це потребує операцій. Потім ми кожен i-й рядок ділимо на його i-й коефіцієнт. Це потребує ще операцій. Тож в результаті пошук оберненої матриці потребує O() часу.

На практиці в методі знаходження оберненої матриці використовуються 2 вкладені цикли всередині яких викликається функція що містить ще 1 вкладений цикл тож на практиці асимптотика теж O()

* + 1. Метод обертань

Прямий хід методу обертань складається з n кроків при чому на i’му кроці виконуються операції над кожним рядком починаючи з i+1’го (Але тільки над елементами індекси яких не менше i). Таким чином кількість дій можна підрахувати як

Обернений хід аналогічний прямому тому результуючий час роботи алгоритму складає ().

На практиці в методі знаходження оберненої матриці використовуються 2 вкладені цикли всередині яких викликається функція що містить ще 1 вкладений цикл тож на практиці асимптотика теж O()

* 1. Тестування ефективності

Залежність кількості арифметичних дій, операцій та часу виконання від кількості рівнянь продемонстрована в таблиці 7.1

Таблиця 7.1 - Залежність швидкодії від кількості рівнянь

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кількість рівнянь | Параметри тестування | Метод | | |
| LUP | Матричний | Обертання |
| 100 | Кількість арифметичних дій | 2671450 | 4009900 | 4054150 |
| Час виконання методу (с) | 0.004 | 0.005 | 0.003 |
| Кількість ітерацій | 1348444 | 2049100 | 990100 |
| 200 | Кількість арифметичних дій | 21352900 | 32039800 | 32218300 |
| Час виконання методу (с) | 0.02 | 0.029 | 0.02 |
| Кількість ітерацій | 10727095 | 16198600 | 7960400 |

Продовження таблиці 7.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кількість рівнянь | Параметри тестування | Метод | | |
| LUP | Матричний | Обертання |
| 300 | Кількість арифметичних дій | 72044350 | 108089700 | 108492450 |
| Час виконання методу (с) | 0.068 | 0.095 | 0.068 |
| Кількість ітерацій | 36135645 | 54447900 | 26910600 |
| 400 | Кількість арифметичних дій | 170745800 | 256159600 | 256876600 |
| Час виконання методу (с) | 0.163 | 0.223 | 0.165 |
| Кількість ітерацій | 85574195 | 128797200 | 63840800 |
| 500 | Кількість арифметичних дій | 333457250 | 500249500 | 501367734 |
| Час виконання методу (с) | 0.315 | 0.461 | 0.313 |
| Кількість ітерацій | 167042244 | 251245500 | 124751000 |

Час виконання методів підраховувався на компьютері з процессором Intel Core i7-4790 (3.6 GHz)

Рисунок 7.1 – Графік залежності швидкодії від кількості рівнянь

Графік залежностей кількості арифметичних дій від кількості рівнянь (Рисунок 7.2)

Рисунок 7.2 – Графік залежності кількості арифметичних дій від кількості рівнянь

Графік залежностей кількості ітерацій від кількості рівнянь (Рисунок 7.3)

Рисунок 7.3 – Графік залежності кількості ітерацій від кількості рівнянь

* 1. Порівняня результатів з аналогом

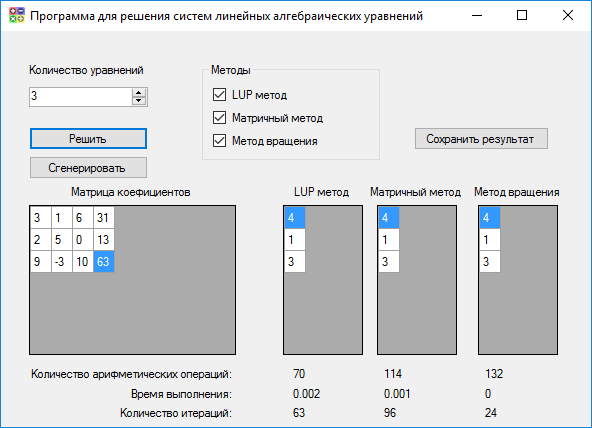


Рисунок 7.4 Результат роботи программи

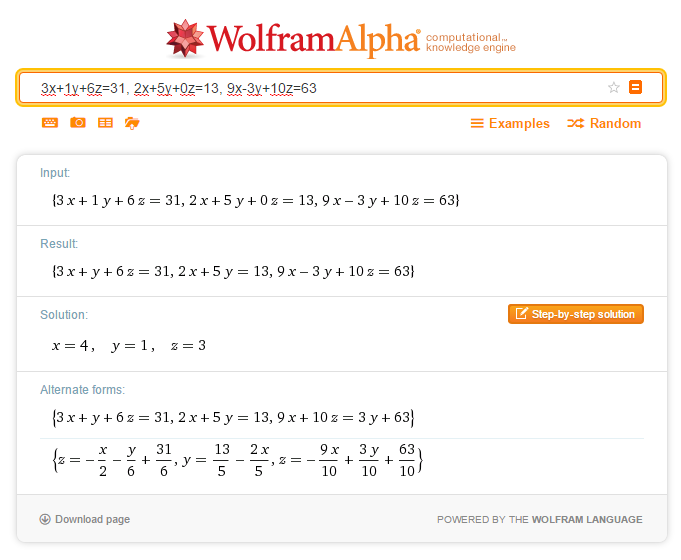


Рисунок 7.5 Результат роботи сторонньої программи

**Висновки**

На етапі пошуку та вивчення теорії пов’заної з моєю темою було досліджено алгоритми розв’язку систем лінійних алгебраїчних рівнянь а саме: методи LUP, Матричний та Обертання які являються точними методами для вирішення систем, тобто дають результат з точністю до округлення дійсних чисел.

На етапі проектування була створена структура matrix що дозволяє ефективно зберігати та обробляти матриці чисел. Ця структура використовувалась для передачі та повернення значень функціями, виконання дій над рядками та матрицями вцілому, для зберігання матриць коефіцієнтів та вільних членів та для пошуку обернених матриць.

Було розроблено реалізації вищевказаних методів. Результатом розробки є освоєння роботи з матрицями, системами лінійних рівнянь та закріплення знань здобутих за курс предмету “Основи програмування”

Окрім реалізації саме алгоритму задачі був розроблений користувацький інтерфейс що значно спрощує використання програми, а саме: введення виведення данних в программу, візуалізація результатів та взяємодію з програмою. Також користувацький інтерфейс дозволяє уникнути роботу з консольним вікном що дозволяє користуватися моєю программою значно ширшій аудиторії.

Також була реалізована можливість збереження результатів роботи программи у файл, а саме збереження вхідних данних (у вигляді системи рівнянь а не матриці коефіцієнтів) збереження результатів роботи методів, а також їх швидкодію кількість арифметичних операцій виконаних методами та кількість ітерацій.

Було проведено тестування програми, під час якого було визначено, що программа правильно обробляє всі виключні випадки та підраховує правильний результат на всіх коректних вхідних наборах данних. Також було проведено аналіз швидкодії программи при якому підтвердилась очікувана складність роботи программи.

Результатом курсової роботи є програмний продукт що може використовуватись в навчальних цілях, в економіці, математиці та інформатиці з метою оптимізації деяких процессів, та безпосерелнього вирішення СЛАР.

**Перелік посилань**

1. Онлайн посилання та ВЕБ-ресурси:

1.1 Вікіпедія – вільна енциклопедія (http://uk.wikipedia.org/)

1.2 - Образовательный портал математического направления (http://yukhym.com/)

1.3 - Сайт для студентів спеціальності інформатика (http://www.mathros.net.ua)

Додаток А Технічне завдання

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Головченко Максим Миколайович

«17» березня 2016 р.

Виконавець:

Студент Булатов Дмитро Єгорович

«17» березня 2016 р..

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: «Розв'язання СЛАР точними методами»

з дисципліни:

«Основи програмування»

Київ 2016

1.1 Мета: Метою курсової роботи є розробка комплексу програм для розв'язання СЛАР точними методами.

1.2 Найменування та галузь застосування об'єкта розробки:Дана робота присвячена розробці комплексу програм призначених для розв'язання СЛАР точними методами.

1.3 Підстава для проведення робіт:Підставою для розробки програмного забезпечення є навчальний план спеціальності 6.050301 «Програмна інженерія», робоча програма дисципліни „Основи програмування”, індивідуальне завдання.

1.4 Дата початку роботи: «14» Березня 2016 р.

1.5 Дата закінчення роботи: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2016 р.

1.6 Призначення розробки: Функціональним призначенням програми є автоматизація наступних процесів: точне вирішення СЛАР за допомогою LUP-методу, матричного методу та методу обертання, графічна інтерпретація процесів та збереження результатів у вигляді текстового файлу. Програма може експлуатуватися у математиці, економіці та інформатиці для точного вирішення СЛАР.

1.7 Вимоги до програми та програмної документації: Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ИСО 5807 - 85 ГОСТ на розробку програмних документів, схем алгоритмів програм, даних та систем.

ГОСТ 19.781 - 74 - Вимоги до розробки програмного забезпечення.

ГОСТ 19.101-77 (СТ СЭВ 1626 - 79) - Держстандарт на розробку програмної документації, видів програм та програмних документів.

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 - 95 - Розробка технічної документації.

1.8 Стадії та етапи розробки:

1. Аналіз методів вирішення поставленої задачі (до 03.04.2016 р.)

2. Розробка сценарію роботи програми (до 17.04.2016 р.)

3. Розробка алгоритмічного забезпечення (до 24.04.2016 р.)

4. Розробка програмного забезпечення (до 01.05.2016 р.)

5. Розробка інтерфейсу, планування, тестування розробленої програми (до 05.06.2016 р.)

6. Розробка пояснювальної записки (до 12.06.2016 р.).

7. Захист курсової роботи (до 09.06.2016 р.).

1.9 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 її оцінювання.

Додаток Б Тексти програмного коду

Project2

*Тексти програмного коду програмного забезпечення*

<*Розв’язання СЛАР точними методами*>

(Вид носія даних)

*CD-RW*

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

*20 арк , 60 Кб*

*студента групи ІП-51 курсу*

*Булатова Дмитра Єгоровича*

# LUPMethod.h

#pragma once

#include "matrix.h"

#include "LUPMethod.h"

matrix LUP(matrix SLAR, matrix fr, int n);

# matrix.h

#pragma once

#include <vector>//Нужен для хранения матрицы

#include <math.h>//Нужен для вычисления корня в методе вращения

extern int operations, iterations;// Об'явлены в MyForm.h

#define inc operations++//Макрос для увеличения количества операций

#define iter iterations++//Макрос для увеличения количества итераций

typedef double matrixtype;//Можно изменить тип данных в котором будут храниться елементы в матрице

struct matrix//Пользовательская структура для хранения и обработки матриц

{

std::vector < std::vector < matrixtype > > m;//m содержит в себе значение всех ячеек матрицы

int h, w;//Высота Ширина

void makeidendity();//Делает матрицу единичной

void swaprows(int a, int b);//Меняет местами строки

void tr();//Транспонирует матрицу

void simplerows(int a, int b, int i);//Избавляеться от 1 переменной действиями с матрицей

void simplerows(int a, int b, int i, matrix &f);//Этот вариант также делает соответственные преобразование с матрицей свободных коефициентов

void transformrows(int r1, int r2, int st, matrix &fr);//Изменяет строки в соответствии с методов вращения

void multiplyrow(int j, matrixtype koef);//Умножает строку на число

double det();//Вычисляет детерминант

matrix(int dh, int dw);//Конструктор (инициализирует m,и сохраняет высоту и ширину)

std::vector < matrixtype > &operator[](int a);//Позволяет обращаться к елементам матрицы через [][] без обращения к m

matrix operator\*(matrix g);//Умножение на матрицу

matrix operator\*(matrixtype a);//Умножение на число

matrix operator/(matrixtype a);//Деление на число

matrix inverse();//Вычисление обратной матрицы (Асимптотика O(N\*N\*N))

};

# MatrixMethod.h

#pragma once

#include "matrix.h"

matrix MatrixMethod(matrix SLAR, matrix fr);

# MyForm.h

#pragma once

#include <time.h>

#include <fstream>

#include "LUPMethod.h"

#include "MatrixMethod.h"

#include "RotationMethod.h"

const int WCONST = 5;

const int minrand = 0;

const int maxrand = 100;

int operations, iterations;

int op1, op2, op3;//Кількість операцій кожного із методів (1-LUP;2-Матричний;3-Обертання)

float t1, t2, t3;//Час роботи кожного із методів в с (1-LUP;2-Матричний;3-Обертання)

int it1, it2, it3;//Кількість ітерацій кожного із методів (1-LUP;2-Матричний;3-Обертання)

matrix SLARmatrix = matrix(1, 1), fr = matrix(1, 1);

namespace Project2 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::NumericUpDown^ Number;

protected:

protected:

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::GroupBox^ groupBox1;

private: System::Windows::Forms::CheckBox^ RotationCheck;

private: System::Windows::Forms::CheckBox^ MatrixCheck;

private: System::Windows::Forms::CheckBox^ LUPCheck;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::DataGridView^ SLAR;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::DataGridView^ LUPSol;

private: System::Windows::Forms::DataGridView^ MatrixSol;

private: System::Windows::Forms::DataGridView^ RotationSol;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::DataVisualization::Charting::Chart^ graph;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::Label^ label8;

private: System::Windows::Forms::Label^ label9;

private: System::Windows::Forms::Label^ label10;

private: System::Windows::Forms::Label^ label11;

private: System::Windows::Forms::Label^ label12;

private: System::Windows::Forms::Label^ label13;

private: System::Windows::Forms::Label^ label14;

private: System::Windows::Forms::Label^ label15;

private: System::Windows::Forms::Label^ label16;

private: System::Windows::Forms::Label^ label17;

private: System::Windows::Forms::Label^ label18;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

System::Windows::Forms::DataVisualization::Charting::ChartArea^ chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());

System::Windows::Forms::DataVisualization::Charting::Legend^ legend1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Legend());

System::Windows::Forms::DataVisualization::Charting::Series^ series1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

System::Windows::Forms::DataVisualization::Charting::Series^ series2 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));

this->Number = (gcnew System::Windows::Forms::NumericUpDown());

this->label1 = (gcnew System::Windows::Forms::Label());

this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());

this->RotationCheck = (gcnew System::Windows::Forms::CheckBox());

this->MatrixCheck = (gcnew System::Windows::Forms::CheckBox());

this->LUPCheck = (gcnew System::Windows::Forms::CheckBox());

this->button1 = (gcnew System::Windows::Forms::Button());

this->SLAR = (gcnew System::Windows::Forms::DataGridView());

this->label2 = (gcnew System::Windows::Forms::Label());

this->LUPSol = (gcnew System::Windows::Forms::DataGridView());

this->MatrixSol = (gcnew System::Windows::Forms::DataGridView());

this->RotationSol = (gcnew System::Windows::Forms::DataGridView());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label6 = (gcnew System::Windows::Forms::Label());

this->graph = (gcnew System::Windows::Forms::DataVisualization::Charting::Chart());

this->button2 = (gcnew System::Windows::Forms::Button());

this->button3 = (gcnew System::Windows::Forms::Button());

this->label7 = (gcnew System::Windows::Forms::Label());

this->label8 = (gcnew System::Windows::Forms::Label());

this->label9 = (gcnew System::Windows::Forms::Label());

this->label10 = (gcnew System::Windows::Forms::Label());

this->label11 = (gcnew System::Windows::Forms::Label());

this->label12 = (gcnew System::Windows::Forms::Label());

this->label13 = (gcnew System::Windows::Forms::Label());

this->label14 = (gcnew System::Windows::Forms::Label());

this->label15 = (gcnew System::Windows::Forms::Label());

this->label16 = (gcnew System::Windows::Forms::Label());

this->label17 = (gcnew System::Windows::Forms::Label());

this->label18 = (gcnew System::Windows::Forms::Label());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->Number))->BeginInit();

this->groupBox1->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->SLAR))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->LUPSol))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->MatrixSol))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->RotationSol))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->graph))->BeginInit();

this->SuspendLayout();

//

// Number

//

this->Number->Location = System::Drawing::Point(28, 56);

this->Number->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 500, 0, 0, 0 });

this->Number->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

this->Number->Name = L"Number";

this->Number->Size = System::Drawing::Size(119, 20);

this->Number->TabIndex = 0;

this->Number->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

this->Number->ValueChanged += gcnew System::EventHandler(this, &MyForm::numericUpDown1\_ValueChanged);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(25, 32);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(122, 13);

this->label1->TabIndex = 1;

this->label1->Text = L"Количество уравнений";

//

// groupBox1

//

this->groupBox1->Controls->Add(this->RotationCheck);

this->groupBox1->Controls->Add(this->MatrixCheck);

this->groupBox1->Controls->Add(this->LUPCheck);

this->groupBox1->Location = System::Drawing::Point(201, 32);

this->groupBox1->Name = L"groupBox1";

this->groupBox1->Size = System::Drawing::Size(178, 98);

this->groupBox1->TabIndex = 2;

this->groupBox1->TabStop = false;

this->groupBox1->Text = L"Методы";

//

// RotationCheck

//

this->RotationCheck->AutoSize = true;

this->RotationCheck->Checked = true;

this->RotationCheck->CheckState = System::Windows::Forms::CheckState::Checked;

this->RotationCheck->Location = System::Drawing::Point(11, 70);

this->RotationCheck->Name = L"RotationCheck";

this->RotationCheck->Size = System::Drawing::Size(112, 17);

this->RotationCheck->TabIndex = 2;

this->RotationCheck->Text = L"Метод вращения";

this->RotationCheck->UseVisualStyleBackColor = true;

//

// MatrixCheck

//

this->MatrixCheck->AutoSize = true;

this->MatrixCheck->Checked = true;

this->MatrixCheck->CheckState = System::Windows::Forms::CheckState::Checked;

this->MatrixCheck->Location = System::Drawing::Point(11, 47);

this->MatrixCheck->Name = L"MatrixCheck";

this->MatrixCheck->Size = System::Drawing::Size(117, 17);

this->MatrixCheck->TabIndex = 1;

this->MatrixCheck->Text = L"Матричный метод";

this->MatrixCheck->UseVisualStyleBackColor = true;

//

// LUPCheck

//

this->LUPCheck->AutoSize = true;

this->LUPCheck->Checked = true;

this->LUPCheck->CheckState = System::Windows::Forms::CheckState::Checked;

this->LUPCheck->Location = System::Drawing::Point(11, 24);

this->LUPCheck->Name = L"LUPCheck";

this->LUPCheck->Size = System::Drawing::Size(81, 17);

this->LUPCheck->TabIndex = 0;

this->LUPCheck->Text = L"LUP метод";

this->LUPCheck->UseVisualStyleBackColor = true;

this->LUPCheck->CheckedChanged += gcnew System::EventHandler(this, &MyForm::checkBox1\_CheckedChanged);

//

// button1

//

this->button1->Location = System::Drawing::Point(28, 96);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(119, 23);

this->button1->TabIndex = 4;

this->button1->Text = L"Решить";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// SLAR

//

this->SLAR->AccessibleName = L"";

this->SLAR->AllowUserToAddRows = false;

this->SLAR->AllowUserToDeleteRows = false;

this->SLAR->AutoSizeColumnsMode = System::Windows::Forms::DataGridViewAutoSizeColumnsMode::DisplayedCells;

this->SLAR->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->SLAR->ColumnHeadersVisible = false;

this->SLAR->Location = System::Drawing::Point(28, 174);

this->SLAR->Name = L"SLAR";

this->SLAR->RowHeadersVisible = false;

this->SLAR->Size = System::Drawing::Size(207, 150);

this->SLAR->TabIndex = 5;

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(67, 154);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(127, 13);

this->label2->TabIndex = 6;

this->label2->Text = L"Матрица коефициентов";

//

// LUPSol

//

this->LUPSol->AllowUserToAddRows = false;

this->LUPSol->AllowUserToDeleteRows = false;

this->LUPSol->AllowUserToResizeColumns = false;

this->LUPSol->AllowUserToResizeRows = false;

this->LUPSol->AutoSizeColumnsMode = System::Windows::Forms::DataGridViewAutoSizeColumnsMode::DisplayedCells;

this->LUPSol->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->LUPSol->ColumnHeadersVisible = false;

this->LUPSol->Location = System::Drawing::Point(282, 174);

this->LUPSol->Name = L"LUPSol";

this->LUPSol->RowHeadersVisible = false;

this->LUPSol->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->LUPSol->Size = System::Drawing::Size(80, 150);

this->LUPSol->TabIndex = 7;

this->LUPSol->Visible = false;

//

// MatrixSol

//

this->MatrixSol->AutoSizeColumnsMode = System::Windows::Forms::DataGridViewAutoSizeColumnsMode::DisplayedCells;

this->MatrixSol->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->MatrixSol->ColumnHeadersVisible = false;

this->MatrixSol->Location = System::Drawing::Point(376, 174);

this->MatrixSol->Name = L"MatrixSol";

this->MatrixSol->RowHeadersVisible = false;

this->MatrixSol->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->MatrixSol->Size = System::Drawing::Size(80, 150);

this->MatrixSol->TabIndex = 8;

this->MatrixSol->Visible = false;

//

// RotationSol

//

this->RotationSol->AutoSizeColumnsMode = System::Windows::Forms::DataGridViewAutoSizeColumnsMode::DisplayedCells;

this->RotationSol->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->RotationSol->ColumnHeadersVisible = false;

this->RotationSol->Location = System::Drawing::Point(477, 174);

this->RotationSol->Name = L"RotationSol";

this->RotationSol->RowHeadersVisible = false;

this->RotationSol->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->RotationSol->Size = System::Drawing::Size(80, 150);

this->RotationSol->TabIndex = 9;

this->RotationSol->Visible = false;

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(290, 154);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(62, 13);

this->label3->TabIndex = 10;

this->label3->Text = L"LUP метод";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(366, 154);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(98, 13);

this->label4->TabIndex = 11;

this->label4->Text = L"Матричный метод";

//

// label5

//

this->label5->AutoSize = true;

this->label5->Location = System::Drawing::Point(470, 154);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(93, 13);

this->label5->TabIndex = 12;

this->label5->Text = L"Метод вращения";

//

// label6

//

this->label6->AutoSize = true;

this->label6->Location = System::Drawing::Point(416, 63);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(0, 13);

this->label6->TabIndex = 13;

//

// graph

//

chartArea1->CursorX->AutoScroll = false;

chartArea1->CursorY->AutoScroll = false;

chartArea1->Name = L"ChartArea1";

this->graph->ChartAreas->Add(chartArea1);

this->graph->Enabled = false;

legend1->Name = L"Legend1";

this->graph->Legends->Add(legend1);

this->graph->Location = System::Drawing::Point(591, 32);

this->graph->Name = L"graph";

series1->ChartArea = L"ChartArea1";

series1->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;

series1->Legend = L"Legend1";

series1->Name = L"Перше рівняня";

series2->ChartArea = L"ChartArea1";

series2->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;

series2->Legend = L"Legend1";

series2->Name = L"Друге рівняня";

this->graph->Series->Add(series1);

this->graph->Series->Add(series2);

this->graph->Size = System::Drawing::Size(527, 337);

this->graph->TabIndex = 14;

this->graph->Text = L"chart1";

this->graph->Click += gcnew System::EventHandler(this, &MyForm::graph\_Click);

//

// button2

//

this->button2->Location = System::Drawing::Point(28, 125);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(119, 23);

this->button2->TabIndex = 15;

this->button2->Text = L"Сгенерировать";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// button3

//

this->button3->Location = System::Drawing::Point(413, 96);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(135, 23);

this->button3->TabIndex = 16;

this->button3->Text = L"Сохранить результат";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3\_Click);

//

// label7

//

this->label7->AutoSize = true;

this->label7->Location = System::Drawing::Point(27, 336);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(208, 13);

this->label7->TabIndex = 17;

this->label7->Text = L"Количество арифметических операций:";

//

// label8

//

this->label8->AutoSize = true;

this->label8->Location = System::Drawing::Point(289, 336);

this->label8->Name = L"label8";

this->label8->Size = System::Drawing::Size(0, 13);

this->label8->TabIndex = 18;

//

// label9

//

this->label9->AutoSize = true;

this->label9->Location = System::Drawing::Point(380, 336);

this->label9->Name = L"label9";

this->label9->Size = System::Drawing::Size(0, 13);

this->label9->TabIndex = 19;

//

// label10

//

this->label10->AutoSize = true;

this->label10->Location = System::Drawing::Point(481, 336);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(0, 13);

this->label10->TabIndex = 20;

//

// label11

//

this->label11->AutoSize = true;

this->label11->Location = System::Drawing::Point(481, 356);

this->label11->Name = L"label11";

this->label11->Size = System::Drawing::Size(0, 13);

this->label11->TabIndex = 24;

//

// label12

//

this->label12->AutoSize = true;

this->label12->Location = System::Drawing::Point(380, 356);

this->label12->Name = L"label12";

this->label12->Size = System::Drawing::Size(0, 13);

this->label12->TabIndex = 23;

//

// label13

//

this->label13->AutoSize = true;

this->label13->Location = System::Drawing::Point(289, 356);

this->label13->Name = L"label13";

this->label13->Size = System::Drawing::Size(0, 13);

this->label13->TabIndex = 22;

//

// label14

//

this->label14->AutoSize = true;

this->label14->Location = System::Drawing::Point(127, 356);

this->label14->Name = L"label14";

this->label14->Size = System::Drawing::Size(108, 13);

this->label14->TabIndex = 21;

this->label14->Text = L"Время выполнения:";

//

// label15

//

this->label15->AutoSize = true;

this->label15->Location = System::Drawing::Point(481, 375);

this->label15->Name = L"label15";

this->label15->Size = System::Drawing::Size(0, 13);

this->label15->TabIndex = 28;

//

// label16

//

this->label16->AutoSize = true;

this->label16->Location = System::Drawing::Point(380, 375);

this->label16->Name = L"label16";

this->label16->Size = System::Drawing::Size(0, 13);

this->label16->TabIndex = 27;

//

// label17

//

this->label17->AutoSize = true;

this->label17->Location = System::Drawing::Point(289, 375);

this->label17->Name = L"label17";

this->label17->Size = System::Drawing::Size(0, 13);

this->label17->TabIndex = 26;

//

// label18

//

this->label18->AutoSize = true;

this->label18->Location = System::Drawing::Point(116, 375);

this->label18->Name = L"label18";

this->label18->Size = System::Drawing::Size(119, 13);

this->label18->TabIndex = 25;

this->label18->Text = L"Количество итераций:";

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(590, 396);

this->Controls->Add(this->label15);

this->Controls->Add(this->label16);

this->Controls->Add(this->label17);

this->Controls->Add(this->label18);

this->Controls->Add(this->label11);

this->Controls->Add(this->label12);

this->Controls->Add(this->label13);

this->Controls->Add(this->label14);

this->Controls->Add(this->label10);

this->Controls->Add(this->label9);

this->Controls->Add(this->label8);

this->Controls->Add(this->label7);

this->Controls->Add(this->button3);

this->Controls->Add(this->button2);

this->Controls->Add(this->graph);

this->Controls->Add(this->label6);

this->Controls->Add(this->label5);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->RotationSol);

this->Controls->Add(this->MatrixSol);

this->Controls->Add(this->LUPSol);

this->Controls->Add(this->label2);

this->Controls->Add(this->SLAR);

this->Controls->Add(this->button1);

this->Controls->Add(this->groupBox1);

this->Controls->Add(this->label1);

this->Controls->Add(this->Number);

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->Name = L"MyForm";

this->Text = L"Программа для решения систем линейных алгебраических уравнений";

this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this, &MyForm::MyForm\_FormClosing);

this->Load += gcnew System::EventHandler(this, &MyForm::MyForm\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->Number))->EndInit();

this->groupBox1->ResumeLayout(false);

this->groupBox1->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->SLAR))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->LUPSol))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->MatrixSol))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->RotationSol))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->graph))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

srand(time(NULL));

SLAR->ColumnCount = 2; // начальное число столбцов и строк

SLAR->RowCount = 1;

SLAR->AutoResizeColumns();// выравнивание ячеек

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

int n = (int)Number->Value;

bool badinput = false;

SLARmatrix = matrix(n, n);

fr = matrix(n, 1);

matrix solcopy = matrix(1, 1);

try // Пытаемся считать таблицу

{

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

SLARmatrix[i][j] = Convert::ToDouble(SLAR->Rows[i]->Cells[j]->Value);

for (int i = 0; i < n; i++)

fr[i][0] = Convert::ToDouble(SLAR->Rows[i]->Cells[n]->Value);

}

catch (FormatException^ e)

{

badinput = true; //Сюда попадаем если в таблицу введено не число

}

if ((SLARmatrix.det() == 0) || (badinput) || (!((LUPCheck->Checked) || (MatrixCheck->Checked) || (RotationCheck->Checked))))//Проверка ошибок

{

if (badinput)

label6->Text = L"Ошибка ввода";

else

if (!((LUPCheck->Checked) || (MatrixCheck->Checked) || (RotationCheck->Checked)))

label6->Text = L"Не выбран ни один метод";

else

label6->Text = L"Система не совместима";

// Дальше очистка старых данных

LUPSol->Visible = 0;

MatrixSol->Visible = 0;

RotationSol->Visible = 0;

Width = 606;

graph->Enabled = false;

label8->Text = "";

label9->Text = "";

label10->Text = "";

label11->Text = "";

label12->Text = "";

label13->Text = "";

label15->Text = "";

label16->Text = "";

label17->Text = "";

}

else

{

label6->Text = L"";//Очищаем сообщение об ошибке

if (LUPCheck->Checked)

{

iterations = operations = 0;//Обнуляем счетчики

time\_t start = clock();//Начинаем считать время выполнения

matrix LUPS = LUP(SLARmatrix, fr, n);//Вызов решения

t1 = (clock() - start) / 1000.0;//Сохраняем время выполнения

op1 = operations;//Сохраняем количество операций

it1 = iterations;//Сохраняем количество итераций

label8->Text = Convert::ToString(op1);//Вывод количества операций

label13->Text = Convert::ToString(t1);//Вывод времени работы

label17->Text = Convert::ToString(it1);//Вывод количества итераций

solcopy = LUPS;//Сохраняем копию ответа (нужна для графика)

//Далее вывод ответа

LUPSol->Visible = true;

LUPSol->RowCount = n;

LUPSol->ColumnCount = 1;

for (int i = 0; i < n; i++)

LUPSol->Rows[i]->Cells[0]->Value = LUPS[i][0];

LUPSol->AutoResizeColumns();//Автоматически изменить ширину колонок

}

else

{

//Если метод не выбран убираем таблицу с ответом и очищаем количество операций\время работы

LUPSol->Visible = false;

label8->Text = "";

label13->Text = "";

label17->Text = "";

}

//Остальные методы работают аналогично

if (MatrixCheck->Checked)

{

iterations = operations = 0;

time\_t start = clock();

matrix MatrixS = MatrixMethod(SLARmatrix, fr);

t2 = (clock() - start) / 1000.0;

op2 = operations;

it2 = iterations;

label9->Text = Convert::ToString(op2);

label12->Text = Convert::ToString(t2);

label16->Text = Convert::ToString(it2);

solcopy = MatrixS;

MatrixSol->Visible = true;

MatrixSol->RowCount = n;

MatrixSol->ColumnCount = 1;

for (int i = 0; i < n; i++)

MatrixSol->Rows[i]->Cells[0]->Value = MatrixS[i][0];

MatrixSol->AutoResizeColumns();

}

else

{

MatrixSol->Visible = false;

label9->Text = "";

label12->Text = "";

label16->Text = "";

}

if (RotationCheck->Checked)

{

iterations = operations = 0;

time\_t start = clock();

matrix RotationS = RotationMethod(SLARmatrix, fr, n);

t3 = (clock() - start) / 1000.0;

op3 = operations;

it3 = iterations;

label10->Text = Convert::ToString(op3);

label11->Text = Convert::ToString(t3);

label15->Text = Convert::ToString(it3);

solcopy = RotationS;

RotationSol->Visible = true;

RotationSol->RowCount = n;

RotationSol->ColumnCount = 1;

for (int i = 0; i < n; i++)

RotationSol->Rows[i]->Cells[0]->Value = RotationS[i][0];

RotationSol->AutoResizeColumns();

}

else

{

RotationSol->Visible = false;

label10->Text = "";

label11->Text = "";

label15->Text = "";

}

if (n == 2)

{

//Рисуем график

Width = 1153;//Изменяем ширину чтобы поместить график

graph->Enabled = true;//Включаем график

double xans = solcopy[0][0], yans = solcopy[1][0];//Координаты точек пересечения

graph->ChartAreas[0]->AxisY->ScaleView->Zoom(yans - WCONST, yans + WCONST);//Изменяем масштаб для наглядности

graph->ChartAreas[0]->AxisX->ScaleView->Zoom(xans - WCONST, xans + WCONST);

graph->Series[0]->Points->Clear();//Очищаем старые графики

graph->Series[1]->Points->Clear();

if (SLARmatrix[0][1] != 0)

{

//Обычный отрезок по двум точкам

graph->Series[0]->Points->AddXY(xans - WCONST, (fr[0][0] - SLARmatrix[0][0] \* (xans + WCONST)) / SLARmatrix[0][1]);

graph->Series[0]->Points->AddXY(xans + WCONST, (fr[0][0] - SLARmatrix[0][0] \* (xans - WCONST)) / SLARmatrix[0][1]);

}

else

{

//Особый случай для вертикальной прямой

graph->Series[0]->Points->AddXY(xans, yans - WCONST);

graph->Series[0]->Points->AddXY(xans, yans + WCONST);

}

if (SLARmatrix[1][1] != 0)

{

//Обычный отрезок по двум точкам

graph->Series[1]->Points->AddXY(xans - WCONST, (fr[1][0] - SLARmatrix[1][0] \* (xans + WCONST)) / SLARmatrix[1][1]);

graph->Series[1]->Points->AddXY(xans + WCONST, (fr[1][0] - SLARmatrix[1][0] \* (xans - WCONST)) / SLARmatrix[1][1]);

}

else

{

//Особый случай для вертикальной прямой

graph->Series[1]->Points->AddXY(xans, yans - WCONST);

graph->Series[1]->Points->AddXY(xans, yans + WCONST);

}

}

else

{

//Если график не нужен

Width = 606;//Возвращаем исходную ширину окна

graph->Enabled = false;//Выключаем график

}

}

}

private: System::Void numericUpDown1\_ValueChanged(System::Object^ sender, System::EventArgs^ e) {

//При изменении количества уравнений обновляем таблицу с входными данными

SLAR->RowCount = 0;//Эта строка очищает старые значения

SLAR->ColumnCount = (int)Number->Value + 1;//Устанавливает новый размер таблицы

SLAR->RowCount = (int)Number->Value;

SLAR->AutoResizeColumns();//Автоматическое изменение ширины ячеек

}

private: System::Void checkBox1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void graph\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

int n = (int)Number->Value;

for (int i = 0; i < n; i++)

for (int j = 0; j <= n; j++)

SLAR->Rows[i]->Cells[j]->Value = (rand() % ((maxrand - minrand) \* 10)) / 10.0 + minrand;//Забиваем таблицу случайными значениями

}

private: System::Void MyForm\_FormClosing(System::Object^ sender, System::Windows::Forms::FormClosingEventArgs^ e) {

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

//Сохранение результатов

if (!((LUPSol->Visible) || (MatrixSol->Visible) || (RotationSol->Visible)))

{

//Нечего сохранять выводим ошибку

label6->Text = "Нет решения для сохранения";

return;

}

std::ofstream fout;

fout.open("output.txt", std::ofstream::out);//Открываем файл для вывода

//Вывод условия

fout << "Условие:\n";

for (int i = 0; i < SLARmatrix.h; i++)

{

for (int j = 0; j < SLARmatrix.w; j++)

{

if ((SLARmatrix[i][j] >= 0) && (j != 0)) fout << "+";

fout << SLARmatrix[i][j] << "\*x" << j;

}

fout << "=" << fr[i][0] << "\n";

}

//Дальше вывод результатов

if (LUPSol->Visible)//Если таблица с решениями LUP метода то у нас есть решение для сохранения

{

fout << "LUP метод: операцій=" << op1 << " час=" << t1 << " кількість ітераій=" << it1 << "\n";//Вывод количества операций, итераций и время выполнения

for (int i = 0; i < LUPSol->Rows->Count; i++)

fout << "x" << i << "=" << Convert::ToDouble(LUPSol->Rows[i]->Cells[0]->Value) << " ";//Вывод ответов в формате xi=число

fout << "\n";

}

//Данные других методов выводяться аналогично

if (MatrixSol->Visible)

{

fout << "Матричний метод: операцій=" << op2 << " час=" << t2 << " кількість ітераій=" << it2 << "\n";;

for (int i = 0; i < MatrixSol->Rows->Count; i++)

fout << "x" << i << "=" << Convert::ToDouble(MatrixSol->Rows[i]->Cells[0]->Value) << " ";

fout << "\n";

}

if (RotationSol->Visible)

{

fout << "Метод поворота: операцій=" << op3 << " час=" << t3 << " кількість ітераій=" << it3 << "\n";;

for (int i = 0; i < RotationSol->Rows->Count; i++)

fout << "x" << i << "=" << Convert::ToDouble(RotationSol->Rows[i]->Cells[0]->Value) << " ";

fout << "\n";

}

MessageBox::Show("Результат сохранен в файл output.txt", "Данные сохранены");//Вывод всплывающего окна с сообщением об успешном сохранении

fout.close();//Закрываем файл

}

};

}

# RotationMethod.h

#pragma once

#include "matrix.h"

matrix RotationMethod(matrix SLAR, matrix fr, int n);

# LUPMethod.cpp

#include "matrix.h"

matrix LUP(matrix SLAR, matrix fr, int n)//LUP метод

{

matrix C = SLAR;//Создаем копию матрицы коефициентов

for (int i = 0; i < n; i++)

{

//Ищем строку с наибольшим i'м коефициентом

double OpV = 0;

int Op = -1;

for (int j = i; j < n; j++,iter)

if (fabs(C[j][i]) > OpV)

{

OpV = fabs(C[j][i]);

Op = j;

}

if (OpV == 0)

{

return matrix(n, 1);

}

C.swaprows(Op, i);//Меняем местами строку с наибольшим i'м коефициентом и i'ю

fr.swaprows(Op, i);

for (int j = i + 1; j < n; j++) {

C[j][i] /= C[i][i]; inc;

for (int k = i + 1; k < n; k++)

{

C[j][k] -= C[j][i] \* C[i][k]; inc; inc; iter;//Преобразовываем матрицу чтобы получить LU разложение

}

}

}

matrix L = matrix(n, n), U = matrix(n, n);

L.makeidendity();//Делаем матрицу L единичной

for (int i = 0; i<n; i++)

for (int j = 0; j < i; j++,iter)

L[i][j] = C[i][j];//Переносим нижнюю часть матрицы C в матрицу L

for (int i = 0; i<n; i++)

for (int j = 0; j <= i; j++)

U[j][i] = C[j][i];//Переносим верхнюю часть матрицы C в матрицу U

//На данный момент матрицы L и U уже являються LU разложением матрицы коефициентов

matrix y = matrix(n, 1);//Создаем промежуточную матрицу y которая нужна для решения уравнения Ux=y

for (int i = 0; i < n; i++)

for (int j = i + 1; j < n; j++)

{

L.simplerows(i, j, i, fr);//Преобразованиями матриц избавляемся от переменных тем самым решая уравнение Ly=b

}

for (int i = 0; i<n; i++,iter)

y[i][0] = fr[i][0] / L[i][i];//После приведения L в вид матрицы у которой все елементы кроме елементов главной диагонали нулевые можем записать ответ в y

matrix x = matrix(n, 1);//Аналогично решаем ищем x который и будет ответом

matrix U1 = U;

for (int i = n - 1; i >= 0; i--)

for (int j = i - 1; j >= 0; j--)

U.simplerows(i, j, i, y);

for (int i = 0; i < n; i++)

{

x[i][0] = y[i][0] / U[i][i]; inc; iter;

}

return x;

}

# matrix.cpp

#pragma once

#include <vector>

#include <math.h>

#include "matrix.h"

//Макрос для увеличения количества операций

typedef double matrixtype;

void matrix::makeidendity()//Делает матрицу единичной

{

if (h != w) throw std::exception();//Если матрица не квадратная выкидываем ошибку

for (int i = 0; i<h; i++)

{

for (int j = 0; j < w; j++,iter)

m[i][j] = 0;

m[i][i] = 1;

}

}

void matrix::swaprows(int a, int b)//Меняет местами строки

{

if (a == b) return;//Если на вход даны номера одной и той же строки ничего делать не требуеться

matrixtype t;//Временная переменная

for (int i = 0; i<w; i++)

{

t = m[a][i];

m[a][i] = m[b][i];

m[b][i] = t;

iter;

}

}

void matrix::tr()//Транспонирует матрицу

{

if (w == h)//Для моих задач не требуеться транспонировать не квадратную матрицу

{

matrixtype t;//Временная переменная

for (int i = 0; i<w; i++)

for (int j = i + 1; j<h; j++)

{

t = m[i][j];

m[i][j] = m[j][i];

m[j][i] = t;

iter;

}

}

}

void matrix::simplerows(int a, int b, int i)//Избавляеться от 1 переменной действиями с матрицей

{

if (m[a][i] != 0)

{

matrixtype koef = -m[b][i] / m[a][i]; inc;//Вычисление коефициента с которым a'я строка прибавиться к b'той

for (int i = 0; i < w; i++)//Прибавление строк

{

m[b][i] += m[a][i] \* koef; inc; inc; iter;

}

}

}

void matrix::simplerows(int a, int b, int i, matrix &f)//Аналогично предыдущему методу

{

if (m[a][i] != 0)

{

matrixtype koef = -m[b][i] / m[a][i]; inc;

for (int i = 0; i < w; i++)

{

m[b][i] += m[a][i] \* koef; inc; inc; iter;

}

f[b][0] += f[a][0] \* koef; inc; inc;//Обновляем значение матрицы свободных членов

}

}

void matrix::multiplyrow(int j, matrixtype koef)//Умножение строки на число

{

for (int i = 0; i < w; i++)

{

m[j][i] \*= koef; inc; iter;

}

}

void matrix::transformrows(int r1, int r2, int st, matrix &fr)//Преобразование используемое в методе вращения

{

if (m[r2][st] != 0)

{

double c = m[r1][st] / sqrt(m[r1][st] \* m[r1][st] + m[r2][st] \* m[r2][st]), s = m[r2][st] / sqrt(m[r1][st] \* m[r1][st] + m[r2][st] \* m[r2][st]); operations += 10;//Формулы из теории

for (int i = 0; i < w; i++)

{

matrixtype t1 = m[r1][i], t2 = m[r2][i];//Сохраняем старые значения для коректного вычисления новых

m[r1][i] = c\*t1 + s\*t2; inc; inc; inc;//Обновляем i'й елемент в строках r1

m[r2][i] = -s\*t1 + c\*t2; inc; inc; inc;//r2

iter;

}

matrixtype t1 = fr[r1][0], t2 = fr[r2][0];//Те же действия для матрицы свободных членов

fr[r1][0] = c\*t1 + s\*t2; inc; inc; inc;

fr[r2][0] = -s\*t1 + c\*t2; inc; inc; inc;

}

}

double matrix::det()//Вычисление детерминанта

{

if (w == h)//Данный метод коректно работает только на квадратных матрицах

{

int koef = 1;

matrix copy = \*this;//Копируем матрицу чтобы исходная матрица не изменилась после выполнения функции

int n = copy.w;

for (int i = 0; i < n - 1; i++)//Делаем матрицу треугольной

{

//Ищем строку с наибольшым i'м коефициентом (чтобы избежать деления на 0)

int Op = -1;

matrixtype OpV = 0;

for (int j = i; j<n; j++)

if (fabs(copy.m[j][i])>OpV)

{

OpV = fabs(copy.m[j][i]);

Op = j;

iter;

}

if (OpV == 0) return 0;//Если наибольний i'й коефициент равен нулю значит матрица вырождения и детерминант равен 0

{

copy.swaprows(i, Op);//Меняем строки местами

if (i != Op) koef \*= -1;//Если строки были поменяны местами значит нужно будет домножить результат на -1 (Замене 2х строк местами знак детерминанта меняеться на противоположный)

}

for (int j = i + 1; j < n; j++)

copy.simplerows(i, j, i);//Убираем i'ю переменную со следуйщих строк

}

//Если алгоритм дошел сюда то матрица верхнетреугольная

double ans = 1;

for (int i = 0; i < n; i++)

{

ans \*= copy.m[i][i]; inc; iter;//Для вичисления детерминанта треугольной матрицы достаточно перемножить числа на главной диагонали

}

inc; return ans\*koef;//Возвращаем найденый детерминант на тот коефициент который получили при сведении матрицы к треугольной

}

return 0;

}

matrix::matrix(int dh, int dw)//Конструктор

{

h = dh;//Сохраняем высоту

w = dw;//и ширину

m.resize(h);//Измением размер вектора векторов до нужных значений

for (int i = 0; i < h; i++,iter)

m[i].resize(w);

}

std::vector < matrixtype > &matrix::operator[](int a)//Позволяет обращаться к елементам матрицы через [][] без обращения к m

{

return m[a];

}

matrix matrix::operator\*(matrix g)//Умножает матрицу на матрицу

{

if (w != g.h) throw std::exception();//Если матрицы нельзя перемножить (ширина первой не равна высоте второй) выкидывает ошибку

matrix sol = matrix(h, g.w);//Создаем матрицу для хранения ответа

for (int i = 0; i < h; i++)

for (int j = 0; j < g.w; j++)

{

sol[i][j] = 0;

for (int r = 0; r < w; r++)

{

sol[i][j] += m[i][r] \* g[r][j]; inc; inc; iter;//Считаем по формуле перемноженую матрицу

}

}

return sol;

}

matrix matrix::operator\*(matrixtype a)//Умножаем матрицу на число

{

matrix sol = matrix(h, w);

for (int i = 0; i < h; i++)

for (int j = 0; j < w; j++)

{

sol[i][j] = m[i][j] \* a; inc; iter;//Умножаем каждый елемент

}

return sol;

}

matrix matrix::operator/(matrixtype a)//Делаит матрицу на число

{

matrix sol = matrix(h, w);

for (int i = 0; i < h; i++)

for (int j = 0; j < w; j++)

{

sol[i][j] = m[i][j] / a; inc; iter;//Делит каждый елемент

}

return sol;

}

matrix matrix::inverse()//Ищет обратную матрицу

{

matrix inv = matrix(h, 2 \* w);//Создает матрицу с присоединенной матрицей

for (int i = 0; i < w; i++)

{

for (int j = 0; j < h; j++)

{

inv[i][j] = m[i][j];//Копирует данные из данной матрицы в левую часть созданой

inv[i][j + w] = 0;//Обнуляет правую часть

iter;

}

inv[i][i + w] = 1;//Заполняет единицами главную диагональ правой части

}

for (int i = 0; i < h; i++)

{

//Ищем строку с наибольшим i'м коефициентом

double OpV = 0;

int Op = -1;

for (int j = i; j < h; j++)

if (fabs(inv[j][i]) > OpV)

{

OpV = fabs(inv[j][i]);

Op = j;

}

inv.swaprows(Op, i);//Меняем местами строку с наибольшим i'м коефициентом и i'ю

for (int j = i + 1; j < h; j++)

inv.simplerows(i, j, i);//Преобразованиями матриц делаем 1 коефициент в матрице нулевым на каждом шаге

}

//В данном месте программы матрица уже являеться верхнетреугольной

for (int i = h - 1; i > 0; i--)

for (int j = 0; j < i; j++)

inv.simplerows(i, j, i);//Дальше преобразованиями удаляем переменные

//Теперь все елементы левой части матрицы кроме главной диагонали нулевые

for (int i = 0; i < h; i++)

inv.multiplyrow(i, 1 / inv[i][i]);//Домножаем строки матрицы на такие коефициенты чтобы матрица полученая слева была единичной

matrix ans = matrix(h, w);

for (int i = 0; i < w; i++)

for (int j = 0; j < h; j++,iter)

ans[i][j] = inv[i][j + w];//Переносим присоединенную матрицу в матрицу-ответ

return ans;//Возвращаем ответ

}

# MatrixMethod.cpp

#include "matrix.h"

#include "MatrixMethod.h"

matrix MatrixMethod(matrix SLAR, matrix fr)//Матричный метод

{

return SLAR.inverse()\*fr;//Обратная матрица умноженая на матрицу свободных членов дает в результате матрицу x'ов что являеться ответом

}

# MyForm.cpp

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main(array<String^>^ arg) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Project2::MyForm form;

Application::Run(%form);

}

# RotationMethod.cpp

#include "matrix.h"

#include "RotationMethod.h"

matrix RotationMethod(matrix SLAR, matrix fr, int n)

{

for (int i = 0; i<n; i++)

{

for (int j = i + 1; j < n; j++)

SLAR.transformrows(i, j, i, fr);//Прямой ход метода поворота

}

for (int i = n - 1; i >= 0; i--)

{

for (int j = i - 1; j >= 0; j--)

SLAR.simplerows(i, j, i, fr);//Обратный ход метода поворота

}

matrix sol = matrix(n, 1);

for (int i = 0; i < n; i++)

{

sol[i][0] = fr[i][0] / SLAR[i][i]; inc; iter;//Находим ответ с упрощенной матрицы

}

return sol;

}