МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

з об'эктно-орієнтованого програмування

(назва дисципліни)

на тему: розв’язання систем алгебраїчних рівнянь точними методами

Студента (ки, ів) 1-го курсу, групи ІП-91

Дзюбака Дем’яна Павловича

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- 2020 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра автоматизованих систем обробки інформації і управління

Дисципліна Об'єктно-орієнтоване програмування

Спеціальність "Інженерія програмного забезпечення"

Курс 1 Група ІП-91 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Дзюбака Дем'яна Павловича |

(прізвище, ім’я, по батькові)

|  |  |
| --- | --- |
| 1. Тема роботи | Розв’язання систем лінійних алгебраїчних рівнянь методами Гауса, Жордана-Гауса та обертань |
|  | |
|  | |

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

|  |  |
| --- | --- |
| 3. Вихідні дані до роботи | Перелік знайдених значень змінних до заданої системи лінійних алгебраїчних рівнянь |
|  | |
|  | |
|  | |

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

|  |
| --- |
| Прийняття СЛАР шляхами зчитування з файлу, автоматичної генерації та введення |
| вручну. ирішення заданої СЛАР методами Гауса, Жордана-Гауса та обертань. Вивантаження |
| знайдених значень змінних заданої системи у файл. |
|  |

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. | Аналіз предметної області |  |  |
| 4. | Проектування архітектури програмної системи |  |  |
| 5. | Розробка сценарію роботи програми |  |  |
| 6. | Узгодження з керівником інтерфейсу користувача |  |  |
| 7. | Розробка програмного забезпечення |  |  |
| 8. | Узгодження з керівником плану тестування |  |  |
| 9. | Тестування програми |  |  |
| 10. | Підготовка пояснювальної записки |  |  |
| 11. | Здача курсової роботи на перевірку |  |  |
| 12. | Захист курсової роботи |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Дзюбак Дем’ян Павлович |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| Керівник |  |  | Головченко Максим Миколайович |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020\_ р.

# Анотація

Пояснювальна записка до курсової роботи: 62 сторінок, 23 рисунки, 2 таблиць, 3 посилання.

Об’єкт дослідження: задача розв’язання системи лінійних алгебраїчних рівнянь.

Мета роботи: дослідження методів розв’язання систем лінійних рівнянь.

Вивчено методи вирішення систем лінійних алгебраїчних рівнянь. Приведені змістовні постановки задач, їх індивідуальні математичні моделі, а також описано детальний процес розв’язання кожної з них.

Виконана програмна реалізація методів вирішення СЛАР : Гауса, Жордана-Гауса, обертань.

СИСТЕМА ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ, СЛАР, МАТРИЦІ, МЕТОД ГАУСА, МЕТОД ЖОРДАНА-ГАУСА, МЕТОД ОБЕРТАНЬ.

**ЗМІСТ**

**ВСТУП**

Курсова робота призначена для вирішення СЛАР методами Гаусса, Жордана-Гаусса та обертання автоматично.

Ціль роботи – розробити програмне забезпечення типу десктоп, що прийматиме матрицю коефіцієнтів та стовбець вільних членів СЛАР та коректно вирішуватиме її.

Сутність вирішуваної задачі – розробка інтерфейсу, що дозволятиме вводити дані та отримувати результат, який зв’язаний з програмною маделлю, що обраховуватиме СЛАР за отриманими даними та повертатиме результат.

# ПОСТАНОВКА ЗАДАЧІ

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

а) метод Гаусса;

б) метод Жордана-Гауса;

в) метод обертання;

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

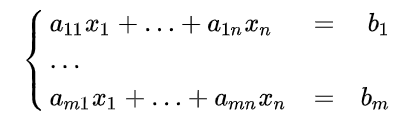
де – матриця коефіцієнтів, – вектор шуканих значень (рішення системи),  
 – вектор вільних членів.Програмне забезпечення повинно обробляти матрицю коефіцієнтів та стовпець вільних членів для СЛАР розмірність яких знаходиться в межах від 2 до 8.

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.

# ТЕОРЕТИЧНІ ВІДОМОСТІ

Метод Гаусса:

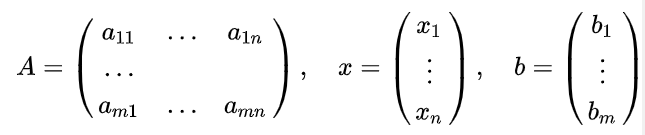
Нехай вихідна система виглядає наступним чином:



Її можна записати в матричному вигляді:

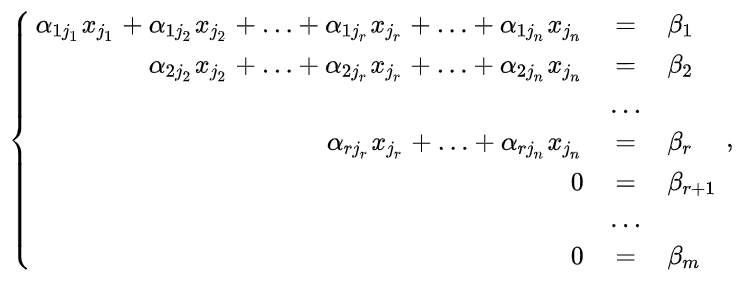


де



Матриця A називається головного датчика системи, b - стовпцем вільних членів.

Тоді, відповідно до властивості елементарних перетворень над рядками, основну матрицю цієї системи можна привести до східчастого увазі (ці ж перетворення потрібно застосовувати до стовбця вільних членів):



де 

При цьому будемо вважати, що базисний мінор (ненульовий мінор максимального порядку) основної матриці знаходиться у верхньому лівому куті, тобто в нього входять тільки коефіцієнти при змінних .

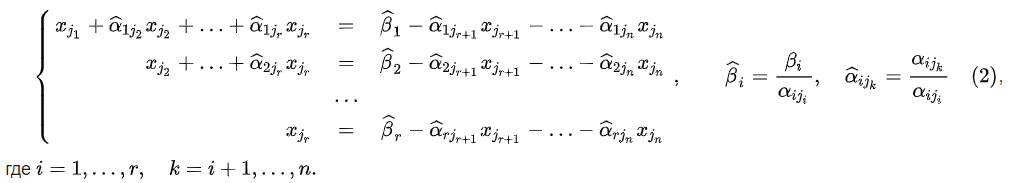
Тоді змінні  називаються головними змінними. Всі інші називаються вільними.

Якщо хоча б одне число , то розглянута система несумісна, т . е. у неї немає жодного рішення.

Нехай  для будь-яких i> r.

Перенесемо вільні змінні за знаки рівності і поділимо кожне з рівнянь системи на свій коефіцієнт при самому лівому i - номер рядка):



{\displaystyle i=1,\ldots ,r,\quad k=i+1,\ldots ,n.}

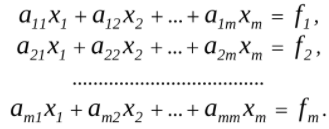
Якщо вільним змінним системи (2) надавати всі можливі значення і вирішувати нову систему щодо головних невідомих знизу вгору (тобто від нижнього рівняння до верхнього), то ми отримаємо всі рішення цієї СЛАР. Так як ця система отримана шляхом елементарних перетворень над вихідною системою (1), то по теоремі про еквівалентність при елементарних перетвореннях системи (1) і (2) еквівалентні, тобто безлічі їх рішень збігаються.

*Метод Жордана-Гаусса:*

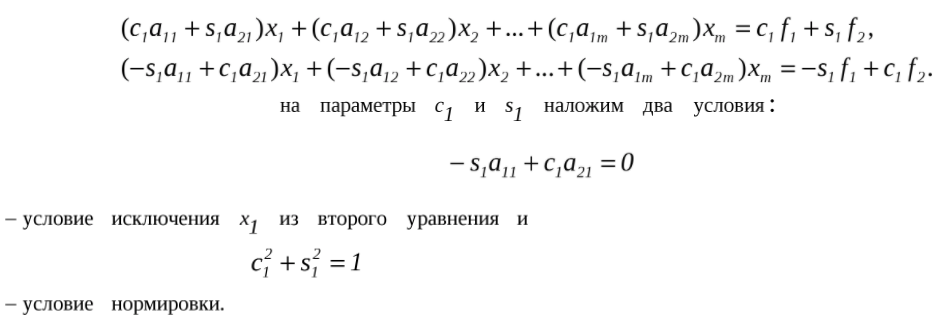
1. Вибирають перший зліва стовпець матриці, в якому є хоч одне відмінне від нуля значення.
2. Якщо саме верхнє число в цьому стовпці нуль, то змінюють всю перший рядок матриці з іншого рядком матриці, де в цій колонці немає нуля.
3. Всі елементи першого рядка ділять на верхній елемент вибраного стовпця.
4. З решти рядків віднімають перший рядок, помножену на перший елемент відповідного рядка, з метою отримати першим елементом кожного рядка (крім першої) нуль.
5. Далі проводять таку ж процедуру з матрицею, що виходить з вихідної матриці після викреслювання першого рядка і першого стовпця.
6. Після повторення цієї процедури n-1 раз отримують верхню трикутну матрицю
7. Вираховують із передостанній рядки останній рядок, помножену на відповідний коефіцієнт, з тим, щоб в передостанньому рядку залишилася тільки 1 на головній діагоналі.
8. Повторюють попередній крок для наступних рядків. В результаті отримують одиничну матрицю і рішення на місці вільного вектора (з ним необхідно проводити всі ті ж перетворення).

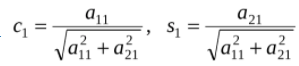
*Метод обертань:*

Як і в методі Гаусса, мета прямого ходу перетворень в цьому методі-приведення системи до треугольно виду послідовним обнуленням поддіагональних елементів спочатку першого стовпчика, потім другого і т.д.

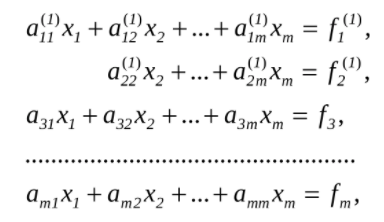


Помножити перше рівняння вихідної системи (1) на з1, друге на s1 і скласти їх; отриманим рівнянням замінимо перше рівняння системи. Потім перше рівняння вихідної системи множимо на -s1, друге на c1 і результатом їх складання замінимо друге рівняння. Таким чином, перші два рівняння (1) замінюються рівняннями

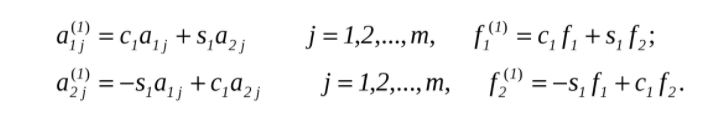


Звідси Ці числа можна інтерпретувати як косинус і синус деяких кута (звідси назва метод обертання, кожен крок такого перетворення можна розглядати як обертання Розширене матриці системи в площині обнуляє індексу).

В результаті перетворень отримаємо систему

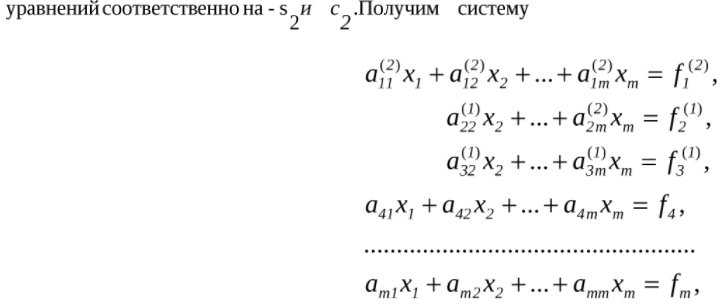


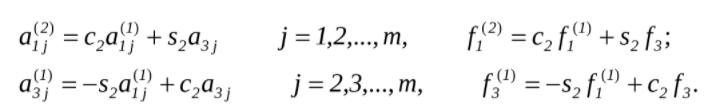
де

Далі перше рівняння системи замінюється новим, отриманим складання результатів множення першого і третього рівнянь відповідно на



а третє-рівнянням, отримане при складання результатів множення тих же

де

Виконавши перетворення m-1 раз, прийшовши до системи

Вид отриманої системи такий же, як після першого етапу перетворень методом Гаусса. Ця система має наступну властивість: довжина будь-якого вектора-стовпця (евклідова норма) Розширене матриці залишається такою ж, як у вихідній матриці. Отже, при виконанні перетворень не спостерігається зростання елементів.

Далі за цим же алгоритмом перетворюється матриця

і т.д.

В результаті m-1 етапів прямого ходу система буде приведена до треугольно виду.

Знаходження невідомих НЕ відрізняється від зворотного ходу методу Гаусса.

1. **ОПИС АЛГОРИТМІВ**

Перелік всіх основних змінних та їхнє призначення наведено в

таблиці 3.1.

Таблиця 3.1 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
|  | Точність з якою необхідно знайти корені розв’язків  () |
| Matrix | Розширена матриця системи |
| result | Масив знайдений значень змінних системи |
| permittingElement | Головний елемент(у методі Жордана-Гауса) |
|  |  |

## Загальний алгоритм

1. ПОЧАТОК
2. Зчитати розмірність системи.
3. Зчитати матрицю системи та стовпець вільних членів:
   1. Зчитати матрицю системи:
      1. Цикл проходу по всіх рядках матриці системи (:
         1. Цикл проходу всіх стовпцях матриці системи   
            (:
            1. ЯКЩО поточний елемент матриці – вірно записане число, ТО записати його в відповідну комірку *matrix*. ІНАКШЕ видати повідомлення про помилку та перейти до пункту 8.
   2. Зчитати стовпець вільних членів:
      1. Цикл проходу по всіх елементах стовпця вільного членів:
         1. Якщо поточний елемент стовпцю вільних членів – вірно записане число, ТО записати його в відповідну комірку *free*. ІНАКШЕ видати повідомлення про помилку та перейти до пункту 8.
4. ЯКЩО обраний метод Гауса, ТО обробити дані згідно алгоритму методу Гауса (пункт 3.2)
5. ЯКЩО обраний метод Жордана-Гауса, ТО обробити дані згідно алгоритму методу Жордана-Гауса (пункт 3.3)
6. ЯКЩО обраний метод обертань, ТО обробити дані згідно алгоритму методу обертань(пункт 3.4).
7. ЯКЩО обраний метод сходиться для вхідних даних, ТО:
   1. ЯКЩО обрана система на дві невідомих, ТО побудувати та вивести графік системи.
   2. Вивести рішення системи.
   3. Записати систему та її рішення у файл.
8. КІНЕЦЬ

## Алгоритм методу Гауса

1. ПОЧАТОК
2. Шляхом елементарних перетворень знайти ранг розширеної та нерозширеної матриць:
   1. ЯКЩО ранг розширеної матриці більший за ранг основної ТО :

Вивести повідомлення : «Матриця несумісна, розв’язків не має»,

Перейти до пункту 7

ІНАКШЕ ЯКЩО ранг розширеної матриці менший за ранг основної ТО :

Вивести повідомлення : «Матриця сумісна і має безліч розв’язків», перейти до пункту 7.

ІНАКШЕ

продовжити

1. Приведення матриці до канонічного (діагонального) вигляду (**stableMatrix**):
   * 1. Поділити елементи першого рядку на його перший елемент
     2. ЦИКЛ проходу по всіх рядках матриці (*Matrix*):
        1. ЦИКЛ проходу по всіх елементах поточного рядка у матриці (*Matrix*):
        2. Відняти відповідні елементи першого рядка, домножені на значення першого елемента в поточному рядку від елементів останнього.
2. Зворотній хід:
   1. Цикл проходу по всіх рядках матриці(*Matrix*), починаючи з останнього:
      1. Розділити поточний рядок на перше ненульове значення.
      2. Отримане значення в стовбці вільних членів помножити на усі елементи наступного по циклу рядка, крім першого ненульового
3. Ініціювати масив чисел (*result*) довжиною кількості рядків матриці(*Matrix*).
4. Цикл проходу по індексам масиву (*result*) :
   1. Присвоїти значення відповідного елемента по індексу стовпця вільних членів елементу масиву (*result*)
5. Кінець

Метод Жордана-Гауса :

1. ПОЧАТОК
2. Шляхом елементарних перетворень знайти ранг розширеної та нерозширеної матриць:
   1. ЯКЩО ранг розширеної матриці більший за ранг основної ТО :

Вивести повідомлення : «Матриця несумісна, розв’язків не має»,

Перейти до пункту 4.

ІНАКШЕ ЯКЩО ранг розширеної матриці менший за ранг основної ТО :

Вивести повідомлення : «Матриця сумісна і має безліч розв’язків», перейти до пункту 4.

ІНАКШЕ:

Продовжити.

1. Приведеня основної матриці до одиничного:
   1. Цикл(1) проходу по всіх рядках матриці(*Matrix*):
      1. Ініціювати чисельну змінну (*permittingElement*) .
      2. Надати їй значення елементу поточного рядка з індексом стовпця поточного рядка(елемент головної діагоналі) .
      3. Цикл проходу по всіх елементах поточного рядка матриці(*Matrix*):
         1. Розділити елемент поточного індекса по стовбцю на (*permittingElement*).
      4. Надати змінній (*permittingElement*) значення елементу головної діагоналі поточного рядка.
      5. Цикл проходу по елементах рядка заданої матриці (*Matrix*)(починаючи з індексу, на один більший за поточний індекс рядка):
         1. Цикл(2) походу по стовбцях матриці (*Matrix*):
            1. Якщо індекс циклу(2) не дорівнює індексу циклу(1).

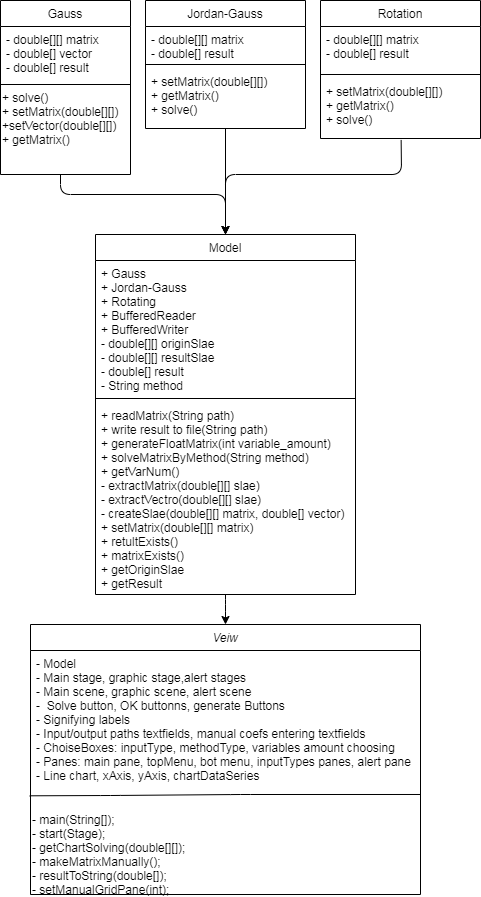
Помножити (*permittingElement*) на елемент матриці (*Matrix*)по індексам :[поточний індекс циклу по елементах рядка][поточний індекс циклу(2)] та відняти від отриманого результату добуток елементів по індексам :[поточний індекс циклу(1)][ поточний індекс циклу по елементах рядка], [поточний індекс циклу(2)][ поточний індекс циклу(1)].

Елементу матриці (*Matrix*) по індексам: [поточний індекс циклу по елементах рядка][почтовий індекс циклу(2)] присвоїти отримане значення.

* + 1. Онулювати значення ствобця поточного індекса циклу(1) крім елементу головної діагоналі.
    2. Ініціювати чисельний масив(*result*) довжиною кількості змінних системи.
    3. Цикл проходу по масиву(*result*):
       1. Присвоїти значення відповідного елемента по індексу стовпця вільних членів елементу масиву (*result*).

1. КІНЕЦЬ.

**4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЧЕННЯ**

****

(рис. )

# 4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

| **№ п/п** | **Назва классу** | **Назва методу** | **Призначення методу** | **Опис вхідних параметрів** | **Опис вихідних параметрів** | **Заголовний файл** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | View | main | Завантаження заданих аргументів у javafx при запуску програми через термінал | String[] args | - | View.java |
| 2 | View | start | Метод, з якого программа javafx починає працювати | Stage stage | - | View.java |
| 3 | View | getChartSolving | При заданні системи з 2-ма змінними метод молює графіки її функцій | double[][] matrix | Вікно з графіком заданої системи | View.java |
| 4 | View | makeMatrixManually | Генерує матрицю з коефіцієнтів, введених у відповідні текстові поля | - | - | View.java |
| 5 | View | resultToString | Конвертує результат розв’язання матриці у строковий тип | double[][] result | String | View.java |
| 6 | View | setManualGridPane | Створює структуровані текстові поля для введення матриці вручну | int length | - | View.java |
| 7 | View | getLabeledMatrix | Створює лей бел, на якому відображається задана згенерована матриця | double[][] matrix | Label | View.java |
| 8 | Model | readMatrix | При обранні вхідного типу «Зчитувати з файлу» зчитує матрицю з заданого файлу | String inPath |  | Model.java |
| 9 | Model | writeResultToFile | Записує результат вирішення системи в заданий файл | String outPath |  | Model.java |
| 10 | Model | generateFloatMatrix | Генерує матрицю СЛАР заданої довжини з елементами типу double | int varNum |  | Model.java |
| 11 | Model | generateIntMatrix | Генерує матрицю СЛАР заданої довжини з округленими елементами типу double | int varNum |  | Model.java |
| 12 | Model | solveMatrixByMethod | Вирішує СЛАР заданим методом | String method |  | Model.java |
| 13 | Model | cloneMatrix | Клонує задану матрицю | double[][] matrix |  | Model.java |
| 14 | Model | extractMatrix | З заданої слар повертає матрицю коефіцієнтів | double[][] slae | double[][] | Model.java |
| 15 | Model | еxtractVector | З заданої СЛАР повертає маси вільних членів | double[][] slae | double[] | Model.java |
| 16 | Model | makeSlae | З заданих матриці коефіцієнтв та масиву вільних членів повертає СЛАР | double[][] matrix, double[] vector | double[][] | Model.java |
| 17 | Model | setMatrix | Записує задану матрицю, як вирішувану | double[][] matrix | - | Model.java |
| 18 | Model | resultExists | З'ясувати чи матриця обрахована(чи існує результат) | - | boolean | Model.java |
| 19 | Model | matrixExists | З'ясувати чи існує матриця(яку програма обраховуватиме) | - | boolean | Model.java |
| 20 | Model | getOriginSlae | Повертає задану користувачем матрицю | - | double[][] | Model.java |
| 21 | Model | getResult | Повертає отриманий в ході обчислень результат | - | double[] | Model.java |
| 22 | Gauss | setMatrix | Записує задану матрицю коефіцієнтів, як вирішувану | double[][] matrix | - | Gauss.java |
| 23 | Gauss | setVector | Записує заданий вектор, як вирішуваний | double[] vector | - | Gauss.java |
| 24 | Gauss | getMatrix | Повертає матрицю коефіцієнтів | - | double[][] | Gauss.java |
| 25 | Gauss | solve | Вирішує СЛАР та повертає масив знайдених значень змінних | - | double[] | Gauss.java |
| 26 | Jordan  Gauss | setMatrix | Записує задану СЛАР, як вирішувану | double[][] matrix | - | JordanGauss  .java |
| 27 | Jordan  Gauss | solve | Вирішує СЛАР та повертає масив знайдених значень змінних | - | double[] | JordanGauss  .java |
| 28 | Rotation | setMatrix | Записує задану СЛАР, як вирішувану | double[][] matrix |  | Rotation.java |
| 29 | Rotation | getMatrix | Повертає СЛАР | - | double[][] | Rotation.java |
| 30 | Rotation | solve | Вирішує СЛАР та повертає масив знайдених значень змінних | - | double[] | Rotation.java |
| 31 | Rotation | solvable | Перевіряє слар на можливість розв’язання | - | boolean | Rotation.java |

1 Тексти програмного коду

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення*

Розв’язання СЛАР точними методами

(Вид носія даних)

*CD-RW*

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

*арк,Кб*

*студента І курсу групи ІП-91*

*Дзюбака Дем’яна Павловича*

public class View extends Application {

Model model;

Stage st, alertStage, resultStage, chartStage;

Scene scene, alertScene, resultScene, chartScene;

BorderPane layout, alertLayout;

HBox topMenu, manChosePane, botMenu, genChoosePane;

VBox readPane, manPane, genPane, alertPane, resultPane, chartPane;

AnchorPane buttonPane;

GridPane manFieldsPane;

Button solveB, genGenDoubleB, manApplyBб, okB, genGenIntB, okRB;

TextField readFileF, manCoefsF[][], manFreeValsF[], outputFileF;

Label inputL, methodL, manVarNumL, readWarnL, genVarNumL, outFileL, enterPathL, generateMatrixL, generatedMatrixL, enterValidOutputPathL, resultL, resultWrittenL;

String outputPath = "E://CourseWorkOutput/output.txt";

ChoiceBox<String> inputTypeList, methodList;

ChoiceBox<Integer> manVarNumCB, genVarNumCB;

LineChart<Number,Number> chart;

NumberAxis xAxis, yAxis;

XYChart.Series<Number, Number> e1;

XYChart.Series<Number, Number> e2;

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage stage) throws Exception {

model = new Model();

st = stage;

st.setTitle("SLAE solver");

st.setWidth(580);

st.setHeight(500);

//general menu

inputL = new Label("Choose input type : ");

methodL = new Label(" Choose solving method : ");

inputTypeList = new ChoiceBox<>();

inputTypeList.getItems().addAll("Generate randomly", "Read from file", "Enter manually");

inputTypeList.setValue("Generate randomly");

inputTypeList.getSelectionModel().selectedItemProperty().addListener((v, oldVal, newVal) -> {

if (newVal == "Read from file") {

layout.setCenter(readPane);

scene.setRoot(layout);

} else if (newVal == "Generate randomly") {

layout.setCenter(genPane);

scene.setRoot(layout);

} else if (newVal == "Enter manually") {

layout.setCenter(manPane);

scene.setRoot(layout);

}

});

methodList = new ChoiceBox<>();

methodList.getItems().addAll("Gauss", "Jordan-Gauss", "Rotation");

methodList.setValue("Gauss");

outFileL = new Label("Enter output file path :");

outputFileF = new TextField();

outputFileF.setMinSize(300, 10d);

outputFileF.setPromptText("default : " + outputPath);

topMenu = new HBox(10);

topMenu.getChildren().addAll(inputL, inputTypeList, methodL, methodList);

topMenu.setPadding(new Insets(10, 10, 10, 10));

solveB = new Button("Solve");

solveB.setOnAction(actionEvent -> {

if(!outputFileF.getText().trim().isEmpty())

if(new File(outputFileF.getText()).isFile())

outputPath = outputFileF.getText();

else {

alertPane.getChildren().remove(0);

alertPane.getChildren().add(0, enterValidOutputPathL);

alertStage.show();

}

if (inputTypeList.getValue() == null) {

alertStage.show();

} else {

if (inputTypeList.getValue() == "Read from file") {

if (!readFileF.getText().trim().isEmpty() & new File(readFileF.getText()).exists()) {

model.readMatrix(readFileF.getText());

if (model.getVarNum() == 2)

getChartSolving(model.getOriginSlae());

model.solveMatrixByMethod(methodList.getValue());

} else {

alertPane.getChildren().remove(0);

alertPane.getChildren().add(0, enterPathL);

alertStage.show();

}

} else if (inputTypeList.getValue() == "Generate randomly") {

if (model.matrixExists()) {

if (model.getVarNum() == 2)

getChartSolving(model.getOriginSlae());

model.solveMatrixByMethod(methodList.getValue());

} else {

alertPane.getChildren().remove(0);

alertPane.getChildren().add(0, generateMatrixL);

alertStage.show();

}

} else if (inputTypeList.getValue() == "Enter manually") {

makeMatrixManually();

if (model.getVarNum() == 2)

getChartSolving(model.getOriginSlae());

model.solveMatrixByMethod(methodList.getValue());

}

if (model.resultExists()) {

model.writeResultToFile(outputPath);

resultWrittenL.setText("Result was written to file : " + outputPath);

resultL.setText(resultToString(model.getResult()));

resultStage.show();

}

}

});

botMenu = new HBox(10);

botMenu.getChildren().addAll(outFileL, outputFileF, solveB);

buttonPane = new AnchorPane();

buttonPane.getChildren().addAll(botMenu);

AnchorPane.setRightAnchor(botMenu, 10d);

AnchorPane.setBottomAnchor(botMenu, 10d);

layout = new BorderPane();

layout.setTop(topMenu);

layout.setBottom(buttonPane);

scene = new Scene(layout);

//result box;

resultStage = new Stage();

resultStage.setTitle("Alert box");

resultStage.setHeight(500);

resultStage.setWidth(500);

resultStage.initModality(Modality.APPLICATION\_MODAL);

resultL = new Label();

resultWrittenL = new Label("Result written to file : " + outputPath);

okRB = new Button("OK");

okRB.setOnAction(ae -> {

resultStage.close();

});

resultPane = new VBox();

resultPane.getChildren().addAll(resultWrittenL, resultL, okRB);

VBox.setMargin(okRB, new Insets(10, 50, 10, 50));

resultScene = new Scene(resultPane);

resultStage.setScene(resultScene);

// alertBox

alertStage = new Stage();

alertStage.initModality(Modality.APPLICATION\_MODAL);

alertStage.setTitle("Alert box");

alertStage.setHeight(130);

alertStage.setWidth(200);

enterPathL = new Label("Enter valid input file path!");

enterPathL.setPadding(new Insets(10, 10, 10, 10));

enterValidOutputPathL = new Label("Enter valid output file path!");

generateMatrixL = new Label("Generate matrix firstly!");

generateMatrixL.setPadding(new Insets(10, 10, 10, 10));

okB = new Button("OK");

okB.setOnAction(ae -> {

alertStage.close();

});

alertPane = new VBox();

alertPane.getChildren().add(0, enterPathL);

alertPane.getChildren().add(1, okB);

VBox.setMargin(okB, new Insets(10, 50, 10, 10));

alertLayout = new BorderPane();

alertLayout.setCenter(alertPane);

alertScene = new Scene(alertLayout);

alertStage.setScene(alertScene);

// "generate" pane

genVarNumL = new Label("Set variables amount : ");

genVarNumCB = new ChoiceBox<>();

genVarNumCB.getItems().addAll(2, 3, 4, 5, 6, 7, 8);

genVarNumCB.setValue(2);

genGenDoubleB = new Button("Generate float num");

genGenDoubleB.setOnAction(ae -> {

model.generateFloatMatrix(genVarNumCB.getValue());

genPane.getChildren().remove(generatedMatrixL);

generatedMatrixL = getLabeledMatrix(model.getOriginSlae());

genPane.getChildren().add(generatedMatrixL);

layout.setCenter(genPane);

});

genGenIntB = new Button("Generate integer num");

genGenIntB.setOnAction(ae -> {

model.generateIntMatrix(genVarNumCB.getValue());

genPane.getChildren().remove(generatedMatrixL);

generatedMatrixL = getLabeledMatrix(model.getOriginSlae());

genPane.getChildren().add(generatedMatrixL);

layout.setCenter(genPane);

});

generatedMatrixL = new Label();

genChoosePane = new HBox();

genChoosePane.setPadding(new Insets(10));

genChoosePane.getChildren().addAll(genVarNumL, genVarNumCB, genGenDoubleB, genGenIntB);

genPane = new VBox();

genPane.getChildren().addAll(genChoosePane, generatedMatrixL);

layout.setCenter(genPane);

// "readPane"

readWarnL = new Label("WARNINGS :\n - coefficients in file have to be separated by spaces\n" +

" - amount of columns has to be one more bigger than amount of rows");

readWarnL.setPadding(new Insets(10));

readFileF = new TextField();

readFileF.setPromptText("Enter input SLAE matrix file path : ");

readFileF.setMaxSize(1000, 10d);

readPane = new VBox();

readPane.setPadding(new Insets(3, 3, 3, 3));

readPane.getChildren().addAll(readWarnL, readFileF);

//"manual" choose pane

manVarNumL = new Label("Choose variables amount : ");

manVarNumCB = new ChoiceBox<>();

manVarNumCB.getItems().addAll(2, 3, 4, 5, 6, 7, 8);

manVarNumCB.setValue(2);

manVarNumCB.getSelectionModel().selectedItemProperty().addListener((v, oldval, newVal) -> {

manPane.getChildren().remove(manFieldsPane);

manFieldsPane = new GridPane();

manFieldsPane.setPadding(new Insets(10));

manFieldsPane.setHgap(5);

manFieldsPane.setVgap(10);

setManualGridPane(newVal);

manPane.getChildren().add(manFieldsPane);

layout.setCenter(manPane);

});

manChosePane = new HBox(10);

manChosePane.setPadding(new Insets(10, 10, 10, 10));

manChosePane.getChildren().addAll(manVarNumL, manVarNumCB);

//"manual" elements pane

manFieldsPane = new GridPane();

manFieldsPane.setPadding(new Insets(10));

manFieldsPane.setHgap(5);

manFieldsPane.setVgap(10);

setManualGridPane(manVarNumCB.getValue());

manPane = new VBox();

manPane.setPadding(new Insets(10));

manPane.getChildren().addAll(manChosePane, manFieldsPane);

st.setScene(scene);

st.show();

}

public void getChartSolving(double[][] matrix) {

if (matrix.length != 2)

throw new RuntimeException("Matrix has less or more than two variables to solve it by chart");

else {

double e1x1 = matrix[0][2]/matrix[0][0]; //e1x2 = 0

double e1x2 = matrix[0][2] / matrix[0][1]; //e1x1 = 0

double e2x1 = matrix[1][2]/ matrix[1][0]; // e2x2 = 0

double e2x2 = matrix[1][2]/ matrix[1][1]; // e2x1 = 0

xAxis = new NumberAxis();

xAxis.setLabel("X");

yAxis = new NumberAxis();

yAxis.setLabel("Y");

chart = new LineChart<Number,Number>(xAxis, yAxis);

e1 = new XYChart.Series<>();

e1.setName(matrix[0][0] + " x1 + " + matrix[0][1]+" x2 = " + matrix[0][2]);

e2 = new XYChart.Series<>();

e2.setName(matrix[1][0] + " x1 + " + matrix[1][1]+" x2 = " + matrix[1][2]);

e1.getData().add(new XYChart.Data<>(e1x1, 0));

e1.getData().add(new XYChart.Data<>(0, e1x2));

e2.getData().add(new XYChart.Data<>(e2x1, 0));

e2.getData().add(new XYChart.Data<>(0, e2x2));

chart.setTitle("Chart solving : ");

chart.getData().addAll(e1,e2);

chartPane = new VBox();

chartPane.getChildren().add(chart);

chartScene = new Scene(chartPane);

chartStage = new Stage();

chartStage.setTitle("Chart solving");

chartStage.setWidth(500);

chartStage.setHeight(500);

chartStage.setScene(chartScene);

chartStage.show();

}

}

public void makeMatrixManually() {

int varNum = manVarNumCB.getValue();

double[][] matrix = new double[varNum][varNum + 1];

int counter = 0;

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

for (int j = 0; j < varNum + 1; j++) {

if (j < varNum)

matrix[i][j] = Double.parseDouble(manCoefsF[i][j].getText());

else

matrix[i][j] = Double.parseDouble(manFreeValsF[i].getText());

}

model.setMatrix(matrix);

}

public String resultToString(double[] result) {

String res = "";

int index;

for (int i = 0; i < result.length; i++) {

index = i + 1;

res += String.format("x" + index + " = " + " %.2f \n", result[i]);

}

return res;

}

public void setManualGridPane(int length) {

int width = length + 1;

manCoefsF = new TextField[length][length];

manFreeValsF = new TextField[length];

for (int i = 0; i < length; i++) {

for (int j = 0; j < width - 1; j++) {

manCoefsF[i][j] = new TextField("0");

manCoefsF[i][j].setMaxSize(30, 30);

int finalI = i;

int finalJ = j;

manCoefsF[i][j].textProperty().addListener((observable, oldValue, newValue) -> {

if (!newValue.matches("\\d\*")) {

manCoefsF[finalI][finalJ].setText(newValue.replaceAll("[^\\d]", ""));

}

});

manFieldsPane.getChildren().add(manCoefsF[i][j]);

GridPane.setConstraints(manCoefsF[i][j], j, i);

if (j + 2 == width) {

manFreeValsF[i] = new TextField("0");

manFreeValsF[i].setMaxSize(30, 30);

manFieldsPane.getChildren().add(manFreeValsF[i]);

GridPane.setConstraints(manFreeValsF[i], j + 1, i);

GridPane.setMargin(manFreeValsF[i], new Insets(0, 10, 0, 10));

}

}

}

}

public static Label getLabeledMatrix(double[][] matrix) {

Label label = new Label("");

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix[0].length; j++)

if (j < matrix[0].length - 1) {

int index = j + 1;

if (j < matrix[0].length - 2)

label.setText(label.getText() + String.format(" %.2f x" + index + " + ", matrix[i][j]));

else

label.setText(label.getText() + String.format(" %.2f x" + index, matrix[i][j]));

} else

label.setText(label.getText() + String.format(" = %.2f " + "%n", matrix[i][j]));

}

return label;

}

}

package Model;

import java.io.\*;

public class Model {

BufferedReader br;

BufferedWriter bw;

Gauss gauss = new Gauss();

JordanGauss jordanGauss = new JordanGauss();

Rotation rotation = new Rotation();

private double[][] originSlae;

private double[][] resultSlae;

private double[] result;

public void readMatrix(String inPath) {

File f = new File(inPath);

if (!f.exists()) {

throw new RuntimeException("File not found");

}

String[] temp;

double[][] matrix ;

int width, length;

try {

br = new BufferedReader(new FileReader(inPath));

temp = br.readLine().split(" ");

width = temp.length;

length = temp.length-1;

matrix = new double[length][width];

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

matrix[0][i] = Double.parseDouble(temp[i]);

while (br.ready()) {

temp = br.readLine().split(" ");

for (int i = 1, j = 0; i < matrix.length; j++, i++)

matrix[i][j] = Double.parseDouble(temp[j]);

}

this.originSlae = matrix;

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

public void writeResultToFile(String outPath) {

try {

bw = new BufferedWriter(new FileWriter(outPath));

bw.write("Input slae :\n");

for (int i = 0; i < originSlae.length; i++)

for (int j = 0; j < originSlae[0].length; j++) {

int index = j+1;

if(j < originSlae[0].length-1){

if (j < originSlae[0].length-2)

bw.write(String.format(" %.2f x" + index + " + ", originSlae[i][j]));

else

bw.write(String.format(" %.2f x" + index, originSlae[i][j]));

} else

bw.write(String.format(" = %.2f " + "%n", originSlae[i][j]));

}

bw.write("\nResult slae : \n");

for (int i = 0; i < resultSlae.length; i++)

for (int j = 0; j < resultSlae[0].length; j++) {

int index = j+1;

if(j < resultSlae[0].length-1){

if (j < resultSlae[0].length-2)

bw.write(String.format(" %.2f x" + index + " + ", resultSlae[i][j]));

else

bw.write(String.format(" %.2f x" + index, resultSlae[i][j]));

} else

bw.write(String.format(" = %.2f " + "%n", resultSlae[i][j]));

}

bw.write("\nResult :\n");

for (int i = 0; i < result.length; i++) {

int index = i + 1;

bw.write(String.format("x" + index + " = " + " %.3f \n", result[i]));

}

} catch (IOException e) {

e.printStackTrace();

} finally{

try {

bw.flush();

bw.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

public void generateFloatMatrix(int varNum) {

double[][] matrix = new double[varNum][varNum + 1];

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

for (int j = 0; j < varNum + 1; j++)

matrix[i][j] = Math.random() \* 10;

this.originSlae = matrix;

}

public void generateIntMatrix(int varNum) {

double[][] matrix = new double[varNum][varNum + 1];

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

for (int j = 0; j < varNum + 1; j++)

matrix[i][j] = Math.round(Math.random() \* 10);

this.originSlae = matrix;

}

public void solveMatrixByMethod(String method) {

if (method == "Gauss") {

gauss.setMatrix(extractMatrix(cloneMatrix(originSlae)));

gauss.setVector(extractVector(cloneMatrix(originSlae)));

result = gauss.solve();

resultSlae = makeSlae(gauss.getMatrix(), result);

} else if (method == "Jordan-Gauss") {

jordanGauss.setMatrix(cloneMatrix(originSlae));

result = jordanGauss.solve();

resultSlae = jordanGauss.getMatrix();

} else if (method == "Rotation") {

rotation.setMatrix(cloneMatrix(originSlae));

result = rotation.solve();

resultSlae = rotation.getMatrix();

}

}

private double[][] cloneMatrix(double[][] matrix) {

double[][] result = new double[matrix.length][matrix[0].length];

for (int i = 0; i < matrix.length; i++)

for (int j = 0; j < matrix[0].length; j++)

result[i][j] = matrix[i][j];

return result;

}

public void showMatrix(double[][]matrix) {

for (int i = 0; i < matrix.length; i++) {

System.out.print("|");

for (int j = 0; j < matrix[0].length; j++) {

if (j < matrix[0].length - 1)

System.out.print(String.format(" %.3f |", matrix[i][j]));

else

System.out.println(String.format(" %.3f |", matrix[i][j]));

}

}

}

public int getVarNum() {

return originSlae.length;

}

private double[][] extractMatrix(double[][] slae) {

double[][] matrix = new double[slae.length][slae.length];

for (int i = 0; i < slae.length; i++)

for (int j = 0; j < slae.length; j++)

matrix[i][j] = slae[i][j];

return matrix;

}

private double[] extractVector(double[][] slae) {

double[] vector = new double[slae.length];

for (int i = 0; i < slae.length; i++)

vector[i] = slae[i][slae.length];

return vector;

}

private double[][] makeSlae(double[][] matrix, double[] resVector) {

double[][] slae = new double[matrix.length][matrix.length + 1];

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix.length+1; j++) {

if(j < matrix.length)

slae[i][j] = matrix[i][j];

else

slae[i][j] = resVector[i];

}

}

return slae;

}

public void createSlae(int varNum) {

double[][] slae = new double[varNum][varNum + 1];

this.originSlae = slae;

}

public void setMatrix(double[][] matrix) {

this.originSlae = matrix;

}

public boolean resultExists() {

return result != null;

}

public boolean matrixExists() {

return originSlae != null;

}

public double[][] getOriginSlae() {

return originSlae;

}

public double[][] getResultSlae() {

return resultSlae;

}

public double[] getResult() {

return result;

}

}

package Model;

import java.io.\*;

public class Model {

BufferedReader br;

BufferedWriter bw;

Gauss gauss = new Gauss();

JordanGauss jordanGauss = new JordanGauss();

Rotation rotation = new Rotation();

private double[][] originSlae;

private double[][] resultSlae;

private double[] result;

public void readMatrix(String inPath) {

File f = new File(inPath);

if (!f.exists()) {

throw new RuntimeException("File not found");

}

String[] temp;

double[][] matrix ;

int width, length;

try {

br = new BufferedReader(new FileReader(inPath));

temp = br.readLine().split(" ");

width = temp.length;

length = temp.length-1;

matrix = new double[length][width];

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

matrix[0][i] = Double.parseDouble(temp[i]);

while (br.ready()) {

temp = br.readLine().split(" ");

for (int i = 1, j = 0; i < matrix.length; j++, i++)

matrix[i][j] = Double.parseDouble(temp[j]);

}

this.originSlae = matrix;

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

public void writeResultToFile(String outPath) {

try {

bw = new BufferedWriter(new FileWriter(outPath));

bw.write("Input slae :\n");

for (int i = 0; i < originSlae.length; i++)

for (int j = 0; j < originSlae[0].length; j++) {

int index = j+1;

if(j < originSlae[0].length-1){

if (j < originSlae[0].length-2)

bw.write(String.format(" %.2f x" + index + " + ", originSlae[i][j]));

else

bw.write(String.format(" %.2f x" + index, originSlae[i][j]));

} else

bw.write(String.format(" = %.2f " + "%n", originSlae[i][j]));

}

bw.write("\nResult slae : \n");

for (int i = 0; i < resultSlae.length; i++)

for (int j = 0; j < resultSlae[0].length; j++) {

int index = j+1;

if(j < resultSlae[0].length-1){

if (j < resultSlae[0].length-2)

bw.write(String.format(" %.2f x" + index + " + ", resultSlae[i][j]));

else

bw.write(String.format(" %.2f x" + index, resultSlae[i][j]));

} else

bw.write(String.format(" = %.2f " + "%n", resultSlae[i][j]));

}

bw.write("\nResult :\n");

for (int i = 0; i < result.length; i++) {

int index = i + 1;

bw.write(String.format("x" + index + " = " + " %.3f \n", result[i]));

}

} catch (IOException e) {

e.printStackTrace();

} finally{

try {

bw.flush();

bw.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

public void generateFloatMatrix(int varNum) {

double[][] matrix = new double[varNum][varNum + 1];

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

for (int j = 0; j < varNum + 1; j++)

matrix[i][j] = Math.random() \* 10;

this.originSlae = matrix;

}

public void generateIntMatrix(int varNum) {

double[][] matrix = new double[varNum][varNum + 1];

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

for (int j = 0; j < varNum + 1; j++)

matrix[i][j] = Math.round(Math.random() \* 10);

this.originSlae = matrix;

}

public void solveMatrixByMethod(String method) {

if (method == "Gauss") {

gauss.setMatrix(extractMatrix(cloneMatrix(originSlae)));

gauss.setVector(extractVector(cloneMatrix(originSlae)));

result = gauss.solve();

resultSlae = makeSlae(gauss.getMatrix(), result);

} else if (method == "Jordan-Gauss") {

jordanGauss.setMatrix(cloneMatrix(originSlae));

result = jordanGauss.solve();

resultSlae = jordanGauss.getMatrix();

} else if (method == "Rotation") {

rotation.setMatrix(cloneMatrix(originSlae));

result = rotation.solve();

resultSlae = rotation.getMatrix();

}

}

private double[][] cloneMatrix(double[][] matrix) {

double[][] result = new double[matrix.length][matrix[0].length];

for (int i = 0; i < matrix.length; i++)

for (int j = 0; j < matrix[0].length; j++)

result[i][j] = matrix[i][j];

return result;

}

public void showMatrix(double[][]matrix) {

for (int i = 0; i < matrix.length; i++) {

System.out.print("|");

for (int j = 0; j < matrix[0].length; j++) {

if (j < matrix[0].length - 1)

System.out.print(String.format(" %.3f |", matrix[i][j]));

else

System.out.println(String.format(" %.3f |", matrix[i][j]));

}

}

}

public int getVarNum() {

return originSlae.length;

}

private double[][] extractMatrix(double[][] slae) {

double[][] matrix = new double[slae.length][slae.length];

for (int i = 0; i < slae.length; i++)

for (int j = 0; j < slae.length; j++)

matrix[i][j] = slae[i][j];

return matrix;

}

private double[] extractVector(double[][] slae) {

double[] vector = new double[slae.length];

for (int i = 0; i < slae.length; i++)

vector[i] = slae[i][slae.length];

return vector;

}

private double[][] makeSlae(double[][] matrix, double[] resVector) {

double[][] slae = new double[matrix.length][matrix.length + 1];

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix.length+1; j++) {

if(j < matrix.length)

slae[i][j] = matrix[i][j];

else

slae[i][j] = resVector[i];

}

}

return slae;

}

public void createSlae(int varNum) {

double[][] slae = new double[varNum][varNum + 1];

this.originSlae = slae;

}

public void setMatrix(double[][] matrix) {

this.originSlae = matrix;

}

public boolean resultExists() {

return result != null;

}

public boolean matrixExists() {

return originSlae != null;

}

public double[][] getOriginSlae() {

return originSlae;

}

public double[][] getResultSlae() {

return resultSlae;

}

public double[] getResult() {

return result;

}

}

package Model;

public class Gauss {

private static final double EPSILON = 0.00001;

private double[][] matrix;

private double[] vector;

private double[] result;

public void setMatrix(double[][] matrix) {

this.matrix = matrix;

}

public void setVector(double[] vector) {

this.vector = vector;

}

public double[][] getMatrix() {

return matrix;

}

public double[] solve() {

int n = matrix.length;

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

int max = i;

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

if (Math.abs(matrix[j][i]) > Math.abs(matrix[max][j]))

max = j;

double[] temp = matrix[i];

matrix[i] = matrix[max];

matrix[max] = temp;

double v = vector[i];

vector[i] = vector[max];

vector[max] = v;

if (Math.abs(matrix[i][i]) <= EPSILON)

throw new RuntimeException("Matrix is singular");

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

double alfa = matrix[k][i] / matrix[i][i];

vector[k] -= alfa \* vector[i];

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

matrix[k][j] -= alfa\*matrix[i][j];

}

}

}

double[] x = new double[n];

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

double sum = 0;

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

sum += matrix[i][j] \* x[j];

}

x[i] = (vector[i] - sum)/matrix[i][i];

}

this.result = result;

return x;

}

}

package Model;

public class JordanGauss {

private double[][] matrix;

public void setMatrix(double[][] matrix) {

this.matrix = matrix;

}

public double[][] getMatrix() {

return matrix;

}

public double[] solve() {

int n = matrix.length;

if ((matrix[0].length - matrix.length) != 1)

throw new RuntimeException("Coefficient matrix is not square!");

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

int permElIndex = i;

double permittingElement = matrix[i][i];

for (int j = 0; j < n+1; j++) //making permitting element equal to 1

matrix[i][j] /= permittingElement;

permittingElement = matrix[i][i];

for (int j = i+1; j < matrix[0].length; j++)

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

if (k != i)

matrix[k][j] = permittingElement\*matrix[k][j] - matrix[i][j]\*matrix[k][i];

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

if (i != j)

matrix[j][i] = 0;

}

double[] x = new double[n];

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

x[j] = matrix[j][n];

return x;

}

public void showMatrix(double[][] matrix){

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix[0].length; j++)

System.out.print(matrix[i][j] + " ");

System.out.println();

}

}

}

package Model;

public class Rotation {

private static double[][] matrix;

private double[] result;

public void setMatrix(double[][] matrix) {

this.matrix = matrix;

}

public double[][] getMatrix() {

return matrix;

}

private static void print(double[][] m, boolean solution) {

System.out.println("m.length = " + m.length);

for (int i = 0; i < m.length; i++) {

for (int j = 0; j < m[i].length; j++) {

if(j==m.length)

System.out.print("| ");

if(solution) {

System.out.print("x" + (i+1) + ": ");

System.out.print(m[i][j] + " ");

}

else

System.out.print(m[i][j] + " ");

}

System.out.print("\n");

}

}

private static boolean solvable() {

boolean state=false;

for(int i=0; i<matrix.length-1; i++)

if(matrix[i][i]!=0.0)

state=true;

return state;

}

public static double[] solve() {

double[][] sol=new double[matrix.length][1];

double[] temp\_first=new double[matrix[0].length];

double[] temp\_second=new double[matrix[0].length];

double[] result = new double[matrix.length];

if(check(matrix)) {

for (int i = 0; i < matrix.length - 1; i++) {

for (int j = i + 1; j < matrix.length; j++) {

double divisor = Math.sqrt(Math.pow(matrix[i][i], 2) + Math.pow(matrix[j][i], 2));

double C = matrix[i][i] / divisor;

double S = matrix[j][i] / divisor;

for (int el = 0; el < matrix[i].length; el++) {

temp\_first[el] = matrix[i][el] \* C + matrix[j][el] \* S;

temp\_second[el] = matrix[i][el] \* (-S) + matrix[j][el] \* C;

}

for (int el = 0; el < matrix[i].length; el++) {

matrix[i][el] = temp\_first[el];

matrix[j][el] = temp\_second[el];

}

}

}

double det = 1;

for (int i = 0; i < matrix.length; i++)

det \*= matrix[i][i];

System.out.println("DET = " + det);

}

//This peace will solve matrix with a simple backtrace.

print(matrix,false);

if(!solvable()) {

System.out.print("I don't think we can really solve this system. Too many solutions.");

return null;

}

for(int i=0; i<matrix.length; i++)

sol[i][0]=0.0;

for(int i=matrix.length-1; i>-1; i--) {

double previous=0;

for (int th = i; th < matrix[i].length - 1; th++)

previous+=matrix[i][th] \* sol[th][0];

sol[i][0] = (matrix[i][matrix[i].length - 1] - previous) / matrix[i][i];

}

System.out.print("END:\n");

print(matrix,false);

print(sol,true);

return result;

}

private static boolean check(double[][] matrix) {

double sum=0.0;

for(int i=0; i<matrix.length; i++) {

sum=0.0;

for (int j = 0; j < matrix.length; j++)

if (i != j)

sum+= matrix[i][j];

if (Math.abs(matrix[i][i])-Math.abs(sum)<0){

System.out.println(Math.abs(matrix[i][i]));

return false;

}

}

return true;

}

}

# 5 ІНСТРУКЦІЯ КОРИСТУВАЧА

## Робота з програмою

Після запуску программи, відкривається її головне вікно (Рисунок 5.1).

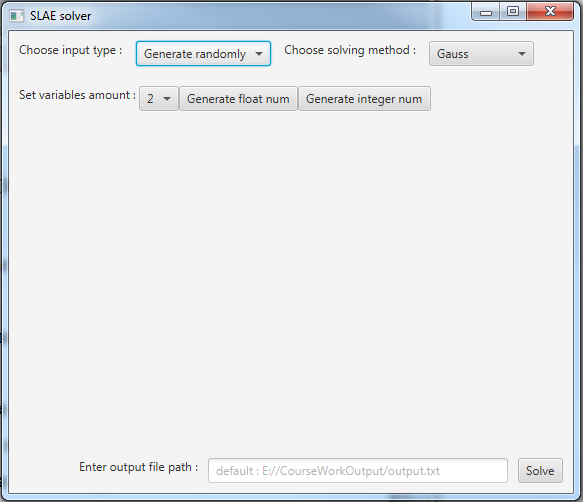


Рисунок 5.1 – Головне вікно програми

Користувач може згенерувати матрицю дійсних, або цілих чисел заданого розміру, або змінити його у Дроп-даун меню після лейбла «Set variables amount:» (рисунок 5.2):

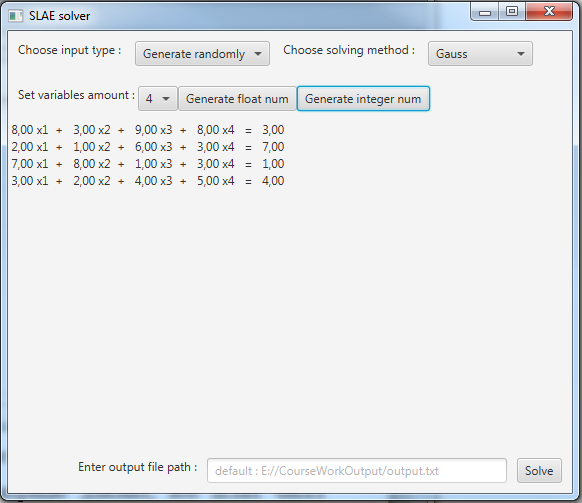
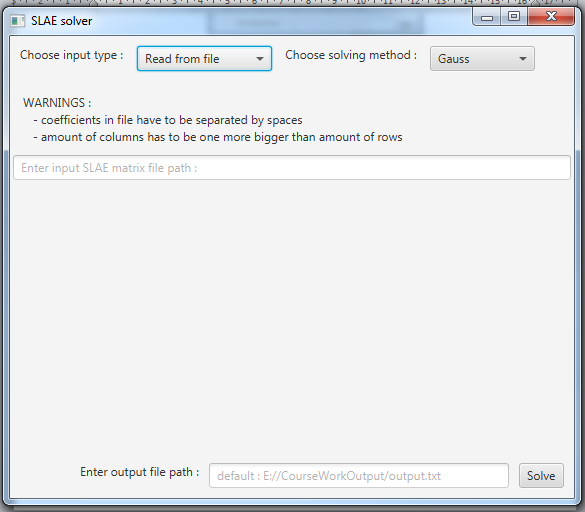


Рисунок 5.2 – Вибір необхідного розміру системи та її генерація

Також користувач може обрати тип введення вхідних даних та метод, яким буде вирішено СЛАР (рисунок 6.3)



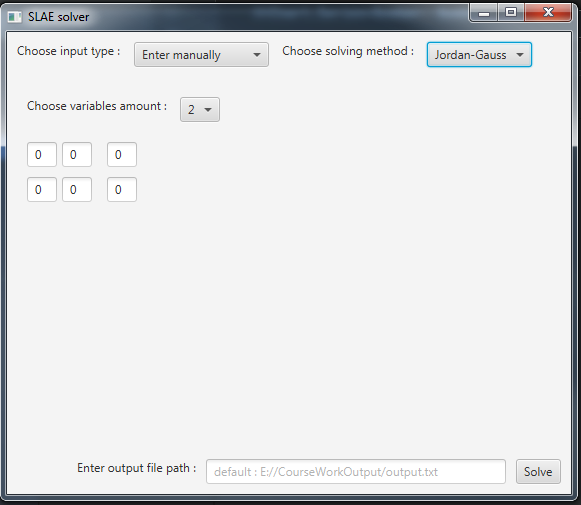


Рисунок 6.3 – Обрання типу введення вхідних даних та методу вирішення СЛАР

Після обрання типу введення, методу вирішення та в залежності від типу введення кількість змінних користувач може ввести шлях до файлу,у який хоче записати отриманий результат. Після цього натискається кнопка «Solve». Далі на екран виводиться вікно із шляхом до аутпут-файлу та результатом(Рисунок 6.4).

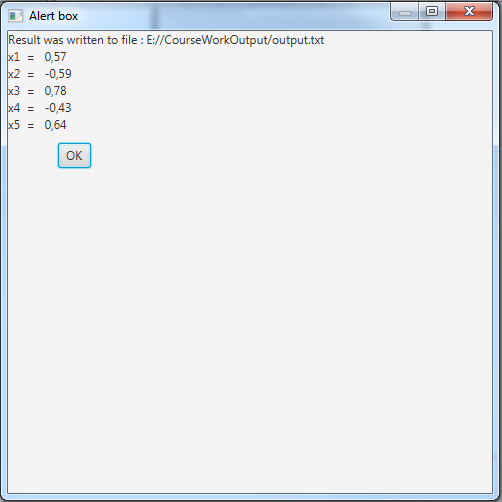


Рисунок 6.4 – Результати вирішення СЛАР

## 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 з точністю не більше, ніж 2 знака після коми (якщо точність більша, то програма автоматично округлить їх за математичними правилами до 2-х знаків після коми).

Результатом виконання програми є розв’язок зданої СЛАР, який видається у вигляді таблиці кожне число якої записане з точністю до 2-х знаків після коми або повідомлення, що дана система не має розв’язків або не сходиться для обраного методу.

## Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 6.1.

Таблиця 6.1 – Системні вимоги програмного забезпечення

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Операційна система | Windows® XP/Windows Vista/Windows 7/ Windows 8/Windows 10 (з останніми обновленнями) | Windows 7/ Windows 8/Windows 10  (з останніми обновленнями) |
| Процесор | Intel® Pentium® ІІІ  1.0 GHz або  AMD Athlon™ 1.0 GHz | Intel® Pentium® D або AMD Athlon™ 64 X2 |
| Оперативна пам'ять | 256 MB RAM (для Windows® XP) / 1 GB RAM (для Windows Vista/Windows 7/  Windows 8/Windows 10) | 2 GB RAM |
| Відеоадаптер | Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог) | |
| Дисплей | 800х600 | 1024х768 або краще |

Продовження таблиці 6.1

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Прилади введення | Клавіатура, комп’ютерна миша | |
| Додаткове програмне забезпечення | Microsoft .Net Framework 4.5.2 або вище | |

**ПЕРЕЛІК ПОСИЛАНЬ**

Метод Гаусса: <https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0>

Метод Жордана-Гаусса: <https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0_%E2%80%94_%D0%96%D0%BE%D1%80%D0%B4%D0%B0%D0%BD%D0%B0>

Метод обертань: <https://works.doklad.ru/view/qYgaul2bTZI.html>

**ДОДАТКИ:**

Технічне завдання

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Головченко Максим Михайлович

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_ р.

Виконавець:

Студент Дзюбак Дем'ян Павлович

«19» березня 2020 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: Розв'язання СЛАР точними методами

з дисципліни:

«Об'єктно-орієнтоване програмування»

Київ 2020

* 1. *Мета*: Метою курсової роботи є розробка програмного забезпечення, що буде знаходити рішення для заданої Системи Лінійних Алгебраїчних Рівнянь(СЛАР) методами Гаусса, Жордана-Гауса та обертання.
  2. *Дата початку роботи*: «19» березня 2020 р.
  3. *Дата закінчення роботи*: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2020 р.
  4. *Вимоги до програмного забезпечення*.

1. Функціональні вимоги:

* Генерувати, зчитувати з файлу або мати можливість приймати матрицю коефіцієнтів СЛАР шляхом введення вручну в вигляді Ax = b де – матриця коефіцієнтів, – вектор шуканих значень (рішення системи), b – вектор вільних членів;
* Обробляти матрицю коефіцієнтів та стовпець вільних членів для СЛАР розмірність яких знаходиться в межах від 2 до 8;
* Можливість обирати метод вирішення СЛАУ з заданих;
* Завантажувати результат роботи в задану папку або файл;
* Виводити у вікні кількість операцій додавання , віднімання, множення та ділення.
* Виводити у вікні графічне вирішення системи 2х2

1. Нефункціональні вимоги:

* Тип – десктоп;
* Мати графічний інтерфейс, що дозволяє вводити відомі коефіцієнти та вільні члени СЛАР в окремі комірки в відповідних місцях(відомі коефіцієнти біля відповідних невідомих, вільні члени після знаку «=») і отримувати результат її розв’язку;
* 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 - 95 - Розробка технічної документації.

* 1. *Стадії та етапи розробки*:

1. Об'єктно-орієнтований аналіз предметної області задачі (до\_\_.\_\_.2020 р.)
2. Об'єктно-орієнтоване проектування архітектури програмної системи (до \_\_.\_\_.2020 р.)
3. Розробка програмного забезпечення (до \_\_.\_\_.2020 р.)
4. Тестування розробленої програми (до \_\_.\_\_.2020 р.)
5. Розробка пояснювальної записки (до \_\_.\_\_.2020 р.).
6. Захист курсової роботи (до \_\_.\_\_.2020 р.).
   1. *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.