Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Метод Жордана-Гаусса для решения систем линейных алгебраических уравнений»**

**Выполнила**:

студентка группы 3823Б1ПМ1-1

Акчурина А.Н.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2024

**Оглавление**

[Постановка задачи 3](#_Toc167785214)

[Метод решения 4](#_Toc167785215)

[Руководство пользователя 5](#_Toc167785216)

[Описание программной реализации: 6](#_Toc167785217)

[Подтверждение корректности. 9](#_Toc167785218)

[Заключение 11](#_Toc167785219)

[Список литературы 12](#_Toc167785220)

[Приложение 13](#_Toc167785221)

# **Постановка задачи**

Требовалось написать программу на языке С++, реализующую метод Гаусса для решения системы линейных уравнений.

В программе должны быть реализованы: наследование, исключение и шаблоны. Также запрещено использовать библиотеку std::vector<> из стандартной библиотеки.

# **Метод решения**

Метод Жордана – Гаусса предназначен для решения системы линейных уравнений. Данный метод является модификацией Гаусса.

Ход решения Жордана – Гаусса:

1. На первом шаге выбирается ненулевой ведущий элемент, в той строке и том столбце, в которых еще не выбирались ведущие элементы.
2. На втором шаге идет зануление остальных элементов в выбранном столбце путем элементарных преобразований.

Типы элементарных преобразований:

1. Транспозиция строк.
2. Транспозиция столбцов.
3. Прибавление к строке матрицы другой, умноженной на константу.

Путем данных преобразований получается приведенная матрица. В программе реализована сортировка по индексам элементов. Таким образом, Ведущие элементы в конечном счете будут идти по возрастанию индекса. Матрица будет иметь ступенчатый вид.

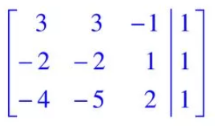
1. 

Рисунок 1. Матрица до применения метода Гаусса

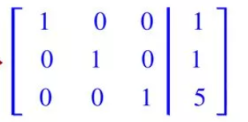
1. 

Рисунок 2. Матрица приведенного вида

# **Руководство пользователя**

При запуске программы на консоль выводится строка, запрашивающая сначала количество строк, и после его введения количество столбцов. При этом программа проверяет вводимые значения. При записи нулевого значения строки или столбца на консоль выводится строка, объявляющая ошибочность этих значений.

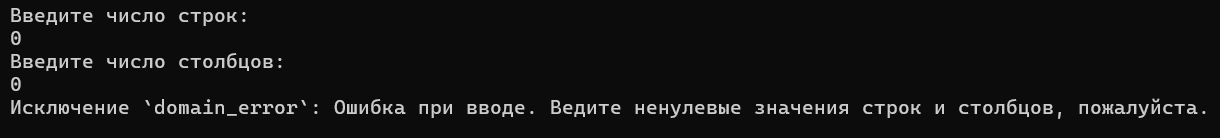


Рисунок 3. Вывод на консоль

# **Описание программной реализации:**

Программа начинает свою работу с функции main().

1. В функции main(): пользователь вводит количество строк и столбцов матрицы.
2. Функция check\_input проверяет введенные значения. Если они оба равны нулю, она выдает сообщение об ошибке.
3. В функции main: пользователь вводит количество строк и столбцов матрицы.Она вызывает check\_input, чтобы проверить введенные значения. Если они оба равны нулю, она выдает сообщение об ошибке затем создает матрицу A с указанными размерами и заполняет ее псевдослучайными числами в диапазоне от 0 до 100. Она создает вектор v1 с одним элементом, установленным в 1. Вычисляет v, которая является результатом умножения матрицы A на вектор v1. Следом использует объект GaussSolver для решения системы линейных уравнений, представленной матрицей A и вектором v. Результат хранится в a. Наконец, она выводит решения, хранящиеся в a.
4. Класс GaussSolver: шаблонный класс, параметризованный типом элемента T. Он предоставляет метод solve, который решает системы линейных уравнений методом Гаусса. Метод comp: Вспомогательный метод, который сравнивает число типа double с нулем с учетом некоторого допуска (малого положительного числа). Используется для проверки, является ли элемент матрицы близким к нулю. Метод solve: принимает матрицу A и вектор v в качестве параметров. Составляет расширенную матрицу M путем добавления столбца из вектора v. Выполняет прямой ход и обратный ход алгоритма Гаусса для преобразования M в ступенчатую форму. Определяет ранг матрицы M и находит базисные решения системы линейных уравнений. Возвращает вектор решений, содержащий базисные решения и свободные переменные (если есть). Использование метода solve: создает объект GaussSolver<double> для работы с системами линейных уравнений с вещественными коэффициентами. Вызывает метод solve для решения системы линейных уравнений. Выводит расширенную матрицу и ранг результата. Возвращает вектор решений, содержащий базисные решения и свободные переменные.
5. Класс MatrixT<T>: шаблонный класс, параметризованный типом элемента T. Хранит матричные данные во внутреннем члене-данных data типа VectorT<Vector>, который представляет собой массив векторов.

Конструкторы:

* Конструктор по умолчанию создает матрицу с нулевым количеством строк и столбцов.
* Конструктор с двумя параметрами создает матрицу с указанным количеством строк и столбцов.
* Конструктор с одним параметром Vector создает матрицу с указанным вектором в качестве единственного столбца.

Геттеры getN() и getM(): возвращают количество строк и столбцов в матрице соответственно.

Операторы ():

* + Операторы () перегружены для доступа к элементам матрицы.
  + Оператор () с двумя параметрами возвращает ссылку на элемент на пересечении указанной строки и столбца.
  + Оператор () с тремя параметрами (константный) возвращает ссылку на элемент на пересечении указанной строки и столбца, но он константный.

Оператор []: оператор [] перегружен для доступа к строкам матрицы. Возвращает ссылку на вектор, представляющий указанную строку.

Оператор <<: перегружен для вставки матрицы в поток вывода. Выводит матрицу в формате, где каждая строка представлена своим вектором.

Оператор \*: перегружен для умножения матрицы на вектор. возвращает новый вектор, который является результатом умножения матрицы на предоставленный вектор. Выбрасывает исключение, если количество столбцов в матрице не совпадает с размером вектора.

Метод Transposition(): транспонирует матрицу.

Псевдоним Matrix: определяет псевдоним Matrix для класса MatrixT<double>.

6. Класс VectorT<T>: хранит координаты вектора в динамическом массиве coords.

Конструкторы:

* Конструктор по умолчанию создает вектор нулевого размера.
* Конструктор с одним параметром размера n создает вектор указанного размера и инициализирует его нулями.
* Конструктор копирования создает новый вектор со значениями, скопированными из предоставленного вектора.
* Конструктор массива создает новый вектор, инициализированный значениями из предоставленного массива arr.
* Конструктор преобразования типа создает новый вектор, преобразуя значения из вектора с другим типом элемента.

# **Подтверждение корректности.**

Проводился эксперимент по выявлению точности. Эксперимент заключался в проверке значений решения матрицы на различных размерах. Были выбраны такие размеры матрицы как 2\*2, 3\*3, 4\*4, 1\*1. Данные вектора решений сверялись с правильными данными, которые были получены из решений матриц идентичных размеров и состоящих из таких же элементов строк и столбцов.

Данные программы:

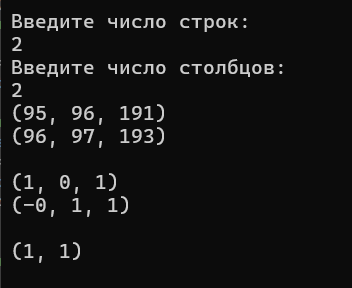


Рисунок 4. Решение матрицы 2\*2 программой.



Рисунок 5. Решение матрицы 2\*2 из официальных данных.

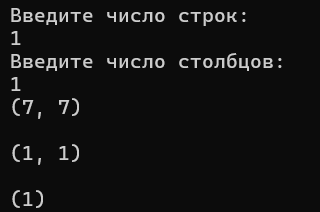


Рисунок 6. решение матрицы 1\*1 программой.



Рисунок 7. решение матрицы 1\*1 из официальных данных.

# **Заключение**

В данной программе был реализован метод Жордана – Гаусса. Для применения этого метода в коде были прописаны класс вектор (т.к. библиотечный файл был недоступен для использования) и класс матриц. Были реализованы: наследование, шаблоны и исключения. Также написанный код прошел проверку на корректность при сравнении данных, полученных программой и правильных данных.

# **Список литературы**

1. Метод Жордана-Гаусса: [Электронный ресурс] // Справочник от автора 24. URL: <https://spravochnick.ru/matematika/metod_zhordana-gaussa/> (Дата обращения: 26.05.2024)
2. Линейная алгебра : учеб. пособие / Н.В. Гредасова, М.А. Корешникова, Н.И. Желонкина [и др.] ; Мин-во науки и высш. образования РФ.— Екатеринбург : Изд-во Урал. ун-та, 2019.— 88 с.

# **Приложение**

class GaussSolver {

public:

bool comp(double& b) const {

double epsilone=0.;

return abs(b)==epsilone;

}

std::vector<Vector> solve(const Matrix& A, const Vector& v){//метод Гаусса

Matrix M(A.getN(), A.getM() + 1);//создает матрицу n\*(m+1)

for (int i = 0; i < M.getN(); i++) {

M[i][M.getM() - 1] = v[i];

for (int j = 0; j < A.getM(); j++) M[i][j] = A[i][j];

}

std::cout << M << std::endl;

int n = M.getN(), m = M.getM();

int rank = 0;

VectorT<int> indexVed(n);

for (int i = 0; i < n; i++) indexVed[i] = 100000000;//вектор хранит индексы ведущих переменных в каждой строке

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

T Ved;

int indexV = 0;

while ((indexV < m - 1) && (M[i][indexV]==0)) indexV++;

if (indexV < m - 1) {

Ved = M[i][indexV];//обновляет ранг матрицы

rank++;

indexVed[i] = indexV;//сохраняет индекс ведущей переменной

for (int t = 0; t < m; t++) M[i][t] = (double)M[i][t] / Ved;

Ved = 1;

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

if ((t != i) && (M[t][indexV]!=0)) {//элементарное преобразование строк Гаусса

double tmp = (double)M[t][indexV] / Ved;

for (int j = 0; j < m; j++) {

M[t][j] -= (M[i][j] \* tmp);

}

}

}

}

}

for (int i = 1; i < n; i++) {//цикл для сортировки строк по возрастанию индексов ведущих элементов

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

if (indexVed[j] < indexVed[j - 1]) {

Vector tmp(n);

tmp = M[j - 1];

M[j - 1] = M[j];

M[j] = tmp;

int tmp2 = indexVed[j - 1];

indexVed[j - 1] = indexVed[j];

indexVed[j] = tmp2;

}

}

}

bool Flag = true;//проверка на существование решения

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

int j = 0;

bool flag = false;

while (j < m - 1) {//проверяет есть ли хотя бы олин ненулевой элемент в строке

if (!comp(M[i][j])) {

flag = true;

break;

}

j++;

}

if ((flag == false)&&(!comp(M[i][m-1]))) Flag = false;

}

std::cout << M << std::endl;

if (Flag == true) {

std::vector<Vector> ans(m - rank);//создание вектора решений

for (int i = 0; i < m - rank; i++) ans[i] = Vector(m - 1);//заполнение вектора решений базовыми переменными

int p = 1;

for (int i = 0; i < m - 1; i++) {

bool flag = true;

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

if (indexVed[j] == i) {

flag = false;

ans[0][j] = M[j][m - 1];

break;

}

}

if (flag == true) {//заполнение вектора решений свободными переменными

for (int j = 0; j < n; j++) ans[p][j] = -M[j][i];

ans[p][i] = 1;

p++;

}

}

return ans;

}

std::vector<Vector> ans;

return ans;

}

};