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 Гаусса»**

**Выполнил**:

студент/ка группы 3821Б1ПМ2

Василевский А.П.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

**Содержание**

[Постановка задачи 3](#_Toc104834997)

[Метод решения 4](#_Toc104834998)

[Руководство пользователя 6](#_Toc104834999)

[Описание программной реализации 9](#_Toc104835000)

[Заключение 12](#_Toc104835001)

# Постановка задачи

Задачами лабораторной работы являются реализация метода Гаусса с прямым и обратным ходом, а также проверка решённой системы линейных уравнений на языке программирования Си++.

# Метод решения

Метода Гаусса заключается в приведении матрицы к ступенчатому виду с помощью элементарных преобразований: перестановки строк, вычитания и складывайте строк, умножение и деление строк на числа (кроме 0) этот этап называется прямым ходом. Далее при помощи обратного хода находятся решения системы линейных уравнений.

Прямой ход метода Гаусса реализован следующим образом. Имеются два счётчика: для позиции строки и столбца. На каждой итерации находится наибольший по модулю элемент в столбце начиная с некоторой строки. После чего строка с наибольшим элементом и та, на которой мы находимся меняются местами. При помощи эквивалентных преобразований все элементы столбца ниже выбранной строчки становятся равными нулю. Таким образом мы получаем “угловой элемент” после чего номер выбранной нами колонки и столбца увеличиваются на 1. Если максимальный по модулю элемент оказался равным нулю, то и все элементы ниже выбранного нами равны нулю. Значит никаких преобразований делать не нужно, следует просто увеличить номер строки и перейти на следующую итерацию.

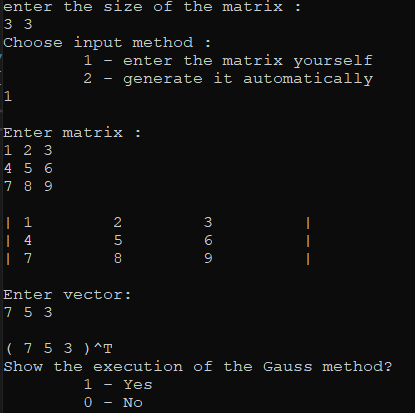
После того как матрица была приведена к лестничному виду приступаем к обратному ходу. При обратном ходе возможны 3 случая: система несовместна, система имеет ровно одно решение и система имеет бесконечно много решений.

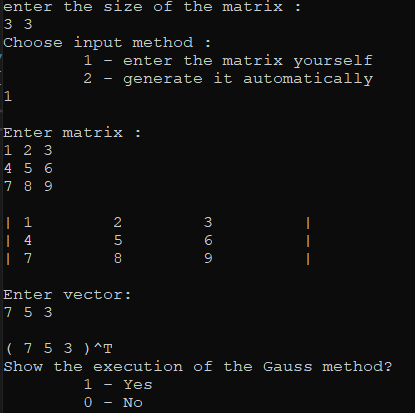
Система линейных уравнений не будет иметь решений если существует строка в матрице состоящая из нулей, а элемент из столбца свободных членов соответствующий этой строке не равен 0.

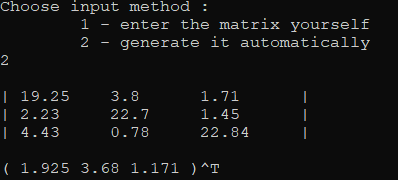
Система линейных уравнений будет иметь одно решение если количество ненулевых строк и столбцов в матрице равно между собой. В таком случае нужно выражать “угловые элементы” каждой строчки начиная с конца через уже известные.

Система линейных уравнений будет иметь бесконечно много решений если количество ненулевых строк меньше чем количество столбцов в матрице. В таком случае нужно выражать базисные (угловые элементы) каждой строчки через оставшиеся свободные элементы, которые равны . Таким образом у нас получаются разные решения в зависимости от выбранных . В программной реализации все свободные элементы считаются равными нулю, таким образом мы получаем одно из возможных решений.

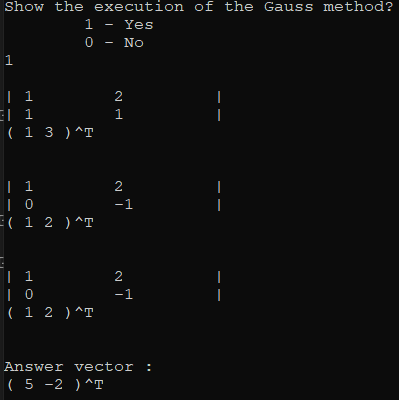
# Руководство пользователя

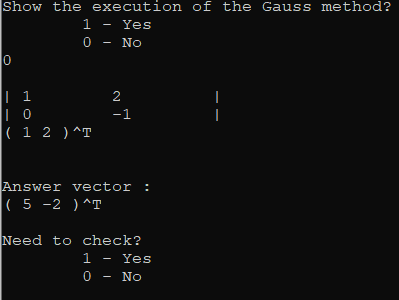
При запуске программы нужно ввести размер матрицы как два числа и выбрать способ задания матрицы: ввести вручную или сгенерировать автоматически.

Если было введено число 1, то нужно ввести матрицу, а после и столбец свободных членов.

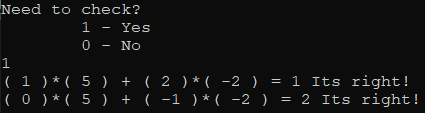
Если было введено число 2 (или любое другое кроме 1), то матрица и столбец свободных членов будут созданы автоматически, причём у матрицы будет диагональное преобладание.

После создания матрицы и столбца свободных членов будет предоставлен выбор: показывать ход метода Гаусса или нет.

 Если было введено число 1, то каждая итерация прямого хода будет выводиться на экран, а после будет выведен ответ.

Если было введено число 2 (или любое другое кроме 1), то будет выведен результат прямого хода метода Гаусса и ответ.

После получения ответа будет выведено сообщение с предложением проверить ответ. Если было введено 1 то произойдёт проверка ответа, в противном случае программа просто завершит своё выполнение.



# Описание программной реализации

Шаблонный класс Vector (template <typename T> class Vector) имеет следующие поля: int size – размер вектора, T\* array – массив элементов T вектора, имеющий длину size.

В классе Vector реализованы следующие методы:

Vector(int i = 1) – конструктор класса, выделяет память для массива array и устанавливает размер size.

~Vector() – деструктор класса, освобождает память выделенную массиву array и устанавливает size = 0.

void resize(int size) – изменяет размер вектора, копируя элементы (устанавливает значение size и перевыделяет память для массива array).

T& operator [](int index) – перегрузка оператора [], возвращает ссылку на array[index].

Vector<T>& operator = (Vector<T>& vector) – перегрузка оператора =.

int GetVectorSize() – возвращает размер вектора (size).

void SwapElem(int FirstId, int SecondId) – меняет элементы вектора местами.

void ShowVector() – выводит вектор в консоль.

void SetVectorValue() – метод для ввода значений вектора.

void GenerateRandomVector() – задаёт вектору случайные значения.

Шаблонный класс Matrix который наследуется от Vector<Vector<T>> (template<typename T> class Matrix :public Vector<Vector<T>>).

Matrix(int ColumnSize = 1, int StringSize = 1) :Vector< Vector<T> >(ColumnSize) – создаёт матрицу размером ColumnSize на StringSize являющуюся “вектором векторов”.

~Matrix() – удаляет матрицу.

int IndexMaxElemOnColumn(int ColumnNumber, int first = 0) – возвращает номер наибольшего по модулю элемента в столбце начиная с first.

void SetElemOnColumn(int ColumnNumber, T elem, int first = 0) – устанавливает элемент (elem) в столбце(ColumnNumber) начиная с first.

Vector<T>& operator [](int index) - перегрузка оператора [], возвращает ссылку на this->array[index].

void SwapString(int FirstString, int SecondString) – меняет местами строки в матрице.

void ShowMartrix() – выводит матрицу в консоль.

void SetMatrixValue() - метод для ввода значений матрицы.

void GenerateMatrixValue() - задаёт матрице случайные значения.

int GetColumnSize() – возвращает количество строк.

int GetStringSize() – возвращает количество элементов в строке.

Шаблонный класс SLU (template <typename T> class SLU) имеет следующие поля: Matrix<T> matrix - матрица, Vector<T> vector - столбец свободных членов; Vector<T> answer – вектор ответов; T precision – переменная, отвечающая за точность (если число по модулю меньше чем precision, то оно считается равным 0); int \_ShowIteration = 0 – переменная отвечающая за необходимость вывода итераций метода гаусса.

SLU(Matrix<T>& \_matrix, Vector<T>& \_vector, T precision = 0.000001) : matrix(\_matrix.GetColumnSize(), \_matrix.GetStringSize()), vector(\_vector.GetVectorSize()), answer(\_matrix.GetStringSize()) – конструктор системы линейных уравнений.

void Swap(int FirstString, int SecondString) – меняет местами строки матрицы и соответствующие элементы из столбца свободных членов.

void ShowIteration() – устанавливает значение \_ShowIteration равным 1.

void DontShowIteration() – устанавливает значение \_ShowIteration равным 0.

Vector<T>& GAUSS() – решает систему линейных уравнений и возвращает веткор.

void ForwardCourse() – метод реализующий прямой ход метода Гаусса. Если в процессе выполнения произойдёт деление на 0 бросит ошибку "ERROR (деление на 0)"

void ReverseCourse() – реализует обратный ход метода Гаусса. Если система линейных элементов несовместна, то бросит ошибку "ERROR 000....00 | not 0 (несовместная система уравнений)"

void Check() – метод реализующий проверку ответа, если ответ неверный, то бросит ошибку "ERROR NOT RIGHT RESULT!".

# Заключение

В ходе лабораторной работы был реализованы классы вектор, матрица и система линейных уравнений, содержащая реализацию метода Гаусса на языке программирования Си++. Был описан алгоритм работы метода Гаусса