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

**Отчёт по лабораторной работе**

**«Решение системы линейных уравнений методом Гаусса»**

**Выполнил**:

Студент группы 3824Б1ПМ1

Кутьин Артём Павлович

**Проверила**:

Бусько П.В.

Нижний Новгород

2025

**Содержание**

[Введение 3](#_Toc199104579)

[Постановка задачи 4](#_Toc199104580)

[Руководство пользователя 5](#_Toc199104581)

[Описание программной реализации 6](#_Toc199104582)

[Результаты экспериментов 10](#_Toc199104583)

[Заключение 12](#_Toc199104584)

[Литература 12](#_Toc199104585)

[Приложения 13](#_Toc199104586)

# Введение

Метод Гаусса – это алгоритм решения систем линейных алгебраических уравнений (СЛАУ), названный в честь немецкого математика Карла Фридриха Гаусса. Принцип метода прост – система приводится к верхне-правому или нижне-левому треугольному виду путём последовательного исключения переменных.

# Постановка задачи

Основная задача лабораторной работы – реализовать метод Гаусса для квадратной матрицы с выбором ведущего элемента.

Также, необходимо было реализовать следующие классы:

1. Шаблонный класс вектор
2. Класс квадратной матрицы, который является шаблоном класса вектор от вектора
3. Класс СЛАУ, который является наследником класса квадратной матрицы. Класс СЛАУ и будет содержать метод Гаусса, выводящий вектор.

# Руководство пользователя

Основная концепция программы – решение системы линейных уравнений методом Жордана-Гаусса (для квадратной матрицы это дважды использование метода Гаусса, то есть обнуление элементов и под главной диагональю, и над главной диагональю. В программировании для обоих методов используются одинаковые конструкции).

Решение системы линейных уравнений, то есть решение системы

где А – матрица коэффициентов, b – вектор значений. x – совокупность векторов, удовлетворяющих равенству.

Программа состоит из 3 этапов:

1. Ввод размерности
2. Ввод матрицы A и вектора b
3. Вывод полученных результатов

На первом этапе пользователь вводит размерность (то есть длину вектора и размер матрицы).

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1 – Ввод размерности

Далее пользователь вводит значение вектора b (через пробел) и матрицы A (по строкам через пробел)

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 2 – Ввод матрицы A и вектора b

После ввода данных программа выводит значение вектора x (подробнее в описании программной реализации)

Изображение выглядит как Шрифт, текст, снимок экрана, Графика

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3 – Вывод вектора

Также, программа может вывести обратную к А матрицу при введении «0» в консоль:

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4 – Вывод обратной матрицы

# Описание программной реализации

Проект состоит из одного основного файла – «Лабораторная 2. Метод Гаусса.cpp»:

В начале подключаются библиотеки: “iostream” и “locale.h” для подключения русского языка.

Классы:

1. Шаблонный класс Vector

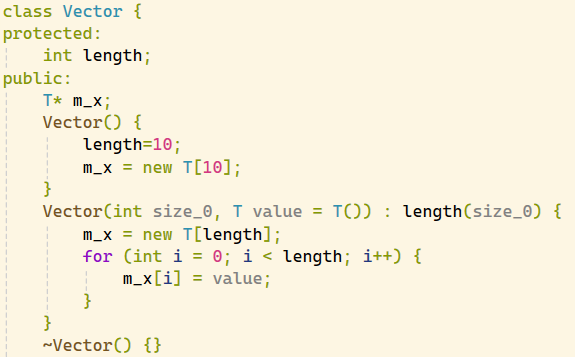


Рис. 5 – Конструктор по умолчанию, конструктор и деструктор (Класс Vector)

Название класса отражает его суть – создаётся массив чисел (вектор), у которого задана длина. Для класса Vector написаны перегрузки операторов (см. Приложение 1) =, +, -, \*, \*=, /, /=, [] для типа данных Vector, а также перегрузка оператора = для константы (написан он для создания нулевого вектора длины length, но если вектор приравнять к ненулевому значению C, то получится вектор длины length, где каждый элемент равен C).

Также, созданы две функции для ввода и вывода вектора, каждая из которых принимает длину вектора в виде аргумента.

Изображение выглядит как текст, Шрифт, рукописный текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 6 – Функции ввода и вывода вектора

1. Класс Квадратная матрица – Sq\_Matrix (шаблон класса вектор от вектора)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 7 – Конструктор по умолчанию, конструктор и деструктор (Класс Sq\_Matrix)

Для класса Sq\_Matrix написаны две перегрузки оператора = для константы (каждый элемент матрицы равен константе) и матрицы:

Sq\_Matrix<T>& operator =(T x) {

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

matrix[i] = x;

}

return \*this;

}

Sq\_Matrix<T>& operator =(const Sq\_Matrix<T> A) {

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

matrix[i] = A.matrix[i];

}

n = A.n;

return \*this;

}

Для класса Sq\_Matrix написаны функции ввода и вывода матрицы (см. Приложение 2), а также функция swap, принимающая номера 2 строк, которая меняет эти строки местами.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 8 – Функция swap (Класс Sq\_Matrix)

1. Класс СЛАУ (SLAE – system of linear algebraic equations)

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 9 – Конструктор и деструктор (Класс SLAE)

Класс, в котором происходят все преобразования матриц и строк. В классе SLAY содержится функция Gauss (см. Приложение 3), принимающая в аргумент по условию задачи вектор. В функции Gauss вводятся матрица A и вектор значения b для решения уравнения. Возвращаемся к нашему уравнению

По условию система линейных уравнений совместна, и матрица A – квадратная. Пусть n – размерность. По условию задачи вывод функции метода Гаусса должен быть вектор. То есть у системы 1 решение, а это и есть критерий равенства ранга матрицы и количества её строк и столбцов. Все строки вводимой матрицы – линейно независимы, значит определитель матрицы отличен от нуля, и, следовательно, существует обратная матрица A-1. Наше единственное решение x находится по формуле

Для нахождения обратной матрицы «на бумаге» мы должны сделать преобразования в матрице размером n на 2\*n:

Для решения вручную мы делаем преобразования всей строки. Но, чтобы аналогично преобразовать в коде, нам необходимо создать единичную матрицу E, и делать равносильные преобразования, как в матрице А, которую мы приводим к единичной.

*Преобразование матрицы A в единичную.*

Во-первых, сделаем так, чтобы на диагональных элементах матрицы стояли ненулевые элементы. Для этого пишем цикл, в котором 2 строки будут меняться местами, если на главной диагонали стоит нулевой элемент. Заметим, что те же строки мы меняем и в единичной матрице E.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 10 – Создание единичной матрицы, ставим на главную диагональ ненулевые элементы

Во-вторых, производим классический метод Гаусса-Жордана, «зануляя» по столбцам все элементы вне главной диагонали.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рис.11 – Метод Гаусса-Жордана для матрицы A

Рассмотрим на примере двух столбцов как происходит зануление первого.

После аналогичных изменений во всех столбцах матрицы A мы получаем диагональную матрицу

Далее изменять матрицу A нет необходимости. Чтобы получить обратную матрицу A-1 остаётся i-ую строку матрицы S, преобразованную из единичной E поделить на di.

Изображение выглядит как текст, Шрифт, линия, белый

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 12 – Получение из диагональной матрицы (A на этом шаге) обратную (E после этого шага)

Итак, получаем путь получения обратной матрицы:

После нахождения обратной матрицы A-1 остаётся умножить её на столбец b и вывести полученный вектор в консоль:

Изображение выглядит как текст, Шрифт, рукописный текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 13 – Нахождение и вывод x, умножение A-1 на b

# Результаты экспериментов

Приведём пример и покажем, как работает программа на данных:

Находим значение x:

# Заключение

В процессе лабораторной работы мною была изучена работа шаблонных классов в c++, наследование классов и перегрузка операторов разных классов.

Также, я вспомнил использование метода Гаусса для решения системы линейных уравнений, повторил нахождение обратной матрицы и перемножение матрицы на столбец.

# Литература

1. Керниган Б., Ритчи Д., Фьюэр А. Язык программирования СИ //М.: Финансы и статистика. – 1992.
2. Гантмахер Ф.Р. Теория матриц. – М. Наука, 1988. С. 552.

# Приложения

Приложение 1. Перегрузки операторов в классе Vector

Vector<T>& operator=(const Vector<T> A) {

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

m\_x[i] = A.m\_x[i];

}

return \*this;

}

Vector<T>& operator=(T x) {

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

m\_x[i] = x;

}

return \*this;

}

Vector<T>& operator +(const Vector<T> A) {

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

m\_x[i] += A.m\_x[i];

}

return \*this;

}

Vector<T>& operator -(const Vector<T> A) {

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

m\_x[i] -= A.m\_x[i];

}

return \*this;

}

Vector<T>& operator \*(T x) {

//Vector<T> a(length);

//a.data = data;

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

m\_x[i] \*= x;

}

return \*this;

}

Vector<T>& operator \*=(T x) {

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

m\_x[i] \*= x;

}

return \*this;

}

Vector<T>& operator /(T x) {

//Vector<T> a(length);

//a.data = data;

if (x != 0) {

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

m\_x[i] /= x;

}

}

return \*this;

}

Vector<T>& operator /=(T x) {

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

if (m\_x[i] != 0) {

m\_x[i] /= x;

}

}

return \*this;

}

T& operator[](int index) { return m\_x[index]; }

Приложение 2. Функции ввода и вывода матрицы

void input\_matrix() {

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

matrix[i].input(n);

}

cout << endl;

}

void output\_matrix() {

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

matrix[i].output(n);

cout << endl;

}

}

Приложение 3. Функция Гаусса

void Gauss(Vector<T> b) {

Sq\_Matrix<T> A(order);

Vector<T> answer(order);

answer = 0;

cout << "Введите матрицу A размером и рангом " << order << ":" << endl;

A.input\_matrix();

//b.input(order);

Sq\_Matrix<T> E(order);

E = 0;

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

E.matrix[i][i] = 1;

}

for (int k = 0;k < order;k++) {

if (A.matrix[k][k] == 0) {

for (int j = k+1;j < order;j++) {

if (A.matrix[j][k] != 0) {

A.swap(k, j);

E.swap(k, j);

break;

}

}

}

}

//A.output\_matrix();

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

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

if ((i != j) && (A.matrix[j][i]!=0)) {

T lead\_el = A.matrix[i][i];

T koef = A.matrix[j][i];

A.matrix[i] \*= koef;

E.matrix[i] \*= koef;

A.matrix[j] \*= lead\_el;

E.matrix[j] \*= lead\_el;

A.matrix[j] = A.matrix[j] - A.matrix[i];

E.matrix[j] = E.matrix[j] - E.matrix[i];

A.matrix[i] /= koef;

E.matrix[i] /= koef;

}

}

}

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

E.matrix[i] /= A.matrix[i][i];

}

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

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

answer[i] += (E.matrix[i][j]\*b[j]);

}

}

cout << "Уравнение A \* x = b;" << endl<<"x = (";

answer.output(order);

int decision;

cout << ")"<<endl<<endl;

cout << "Введите 0, чтобы вывести обратную к А матрицу" << endl;

cin >> decision;

if (decision == 0) {

cout << endl;

E.output\_matrix();

}

}