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**

**«Решение СЛАУ методом Гаусса»**

**Выполнил**:

студентка группы 3824Б1ПМ1

Фирсова А.М

**Проверила**:

Бусько П.В.

Нижний Новгород

2024

**Содержание**

[Введение 3](#_Toc199022372)

[Постановка задачи 4](#_Toc199022373)

[Руководство пользователя 5](#_Toc199022374)

[Описание программной реализации 7](#_Toc199022375)

[Результаты экспериментов 10](#_Toc199022376)

[Заключение 12](#_Toc199022377)

[Литература 13](#_Toc199022378)

[Приложение 14](#_Toc199022379)

# Введение

В вычислительной математике и инженерных расчетах часто возникают задачи, требующие решения систем линейных уравнений. Такие системы появляются при моделировании физических процессов (например, распределение тепла в материалах, расчет электрических цепей), оптимизации экономических моделей, обработке данных и даже в компьютерной графике (например, при расчете освещения или анимации).

Однако точное аналитическое решение СЛАУ возможно лишь для небольших и хорошо обусловленных систем. В реальных задачах, где матрицы могут быть большими или плохо обусловленными, на первый план выходят численные методы, обеспечивающие устойчивость и точность вычислений. Одним из таких методов является метод Гаусса с выбором ведущего элемента, который минимизирует влияние ошибок округления и повышает надежность расчетов.

# Постановка задачи

Разработать программу на C++, реализующую метод Гаусса с выбором ведущего элемента для решения СЛАУ. Программа должна:

* Поддерживать работу с квадратными матрицами произвольного размера;
* Обеспечивать устойчивость вычислений за счет выбора ведущего элемента;
* Проверять систему на вырожденность и корректно обрабатывать ошибки;
* Предоставлять интерактивный интерфейс для ввода данных и вывода результатов.

# Руководство пользователя

**Описание работы программы.**

Программа реализует метод Гаусса с выбором ведущего элемента для решения систем линейных алгебраических уравнений (СЛАУ) вида: A\*X=B, где:

* A — квадратная матрица коэффициентов размера n × n,
* X — вектор неизвестных,
* B — вектор правых частей.

**Действия пользователя.**

1. Запуск программы

Пользователь запускает программу. На экране появляется приглашение ввести размер матрицы.

2. Ввод размера матрицы

Пользователь вводит целое число n (размер матрицы n×n).

3. Ввод матрицы коэффициентов

Программа запрашивает построчный ввод элементов матрицы A. Пользователь вводит элементы каждой строки через пробел.

4. Ввод вектора правых частей

Программа запрашивает вектор B. Пользователь вводит n чисел через пробел.

5. Получение решения

Программа вычисляет решение и выводит его.

6. Продолжение работы (опционально)

Программа спрашивает, нужно ли решить систему с другим вектором B. Если пользователь вводит y, программа запрашивает новый вектор B и повторяет вычисления. Если вводит n, программа завершает работу.

7. Обработка ошибок

Если матрица вырождена, программа выдает ошибку.

8. Завершение работы

После ввода n программа закрывается.

Пример вывода в консоль(*рис.1*):

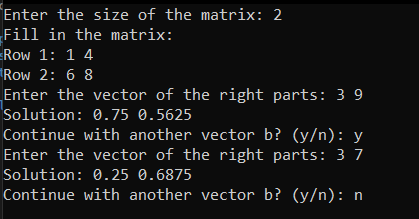


Рисунок 1. Пример вывода в консоль.

# Описание программной реализации

1. Файл ‘’Gauss.h’.

В нем находятся заголовочные файлы, классовая структура.

* Заголовочные файлы.
* <iostream> - ввод-вывод;
* <cmath> - математические функции.
* Классовая структура.
* Vector<T>: работа с одномерными массивами.
* Matrix<T>:представление квадратной матрицы (наследуется от Vector<Vector<T>>).
* SLAU<T>: решение СЛАУ (наследуется от Matrix<T>).
* Класс Vector.

Назначение: Работа с одномерными массивами (векторами) произвольного типа T.

Поля:

size\_t size - размер вектора.

T\* data - указатель на динамический массив данных.

Методы:

* Конструкторы:

Vector(size\_t n = 1) - создает вектор размером n, заполненный нулями.

Vector(const Vector& other) - копирующий конструктор.

* Деструктор: Освобождает память.

Операторы:

Operator [] - доступ к элементам по индексу.

Operator = - копирование векторов.

Operator << и operator >> - ввод/вывод.

* Класс Matrix.

Назначение: Представление квадратной матрицы как вектора векторов.

Наследование: public Vector<Vector<T>>.

Методы:

* Конструктор: Matrix(size\_t n = 1) - создает матрицу n×n, заполненную нулями.
* swapRows(size\_t i, size\_t j) - переставляет строки местами.
* Класс SLAU
* Назначение класса

Класс SLAU<T> предназначен для решения систем линейных алгебраических уравнений (СЛАУ).

* Методы класса
* Конструктор:

Создает систему уравнений с квадратной матрицей размером n×n

Наследует функциональность класса Matrix<T>

* Основной метод solve():

Решает систему уравнений методом Гаусса с выбором ведущего элемента.

* Алгоритм работы:

Создает копию матрицы коэффициентов.

Выполняет прямой ход метода Гаусса:

Выбор ведущего элемента в столбце;

Перестановка строк для устойчивости;

Обнуление элементов ниже главной диагонали.

Выполняет обратный ход:

Последовательное вычисление неизвестных;

Возвращает вектор решений;

Параметры:

b - вектор правых частей.

Возвращаемое значение:

Вектор решений X.

* Исключения:

std::runtime\_error при вырожденной матрице.

std::runtime\_error при нулевом диагональном элементе.

* Особенности реализации

Выбор ведущего элемента:

Для повышения устойчивости алгоритма;

Предотвращает деление на малые числа.

Проверка на вырожденность:

Пороговая величина ε = 1e-10;

Контролирует точность вычислений.

* Вывод

Класс SLAU<T> представляет собой надежную реализацию метода Гаусса для решения систем линейных уравнений. Благодаря выбору ведущего элемента и контролю точности, он обеспечивает устойчивое решение для широкого класса задач. Класс может быть использован как основа для более сложных вычислительных алгоритмов.

# Результаты экспериментов

Для проверки корректности работы программы были рассмотрены три примера систем линейных уравнений (СЛАУ) с ручным расчетом и программным решением. Во всех случаях результаты совпали с ожидаемыми значениями.

**Пример 1. Простая система 2×2**

Исходная система:

2x + y = 5

x – y = 1

Ручной расчет (метод Гаусса):

Прямой ход:

Строка 2 = Строка 2 - 0.5 × Строка 1 → 0x - 1.5y = -1.5

Обратный ход:

y = (-1.5) / (-1.5) = 1

x = (5 - 1×1) / 2 = 2

Решение программы (рис.2):

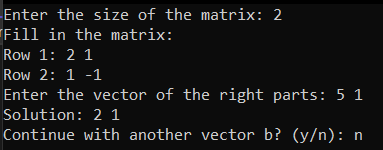


Рисунок 2. Решение программы.

Проверка:

2\*2 + 1\*1 = 5

2\*1 – 1\*1 = 1

**Пример 2. Система 3×3**

Исходная система:

x + 2y + 3z = 14

2x + 4y + z = 13

3x + y + 2z = 11

Ручной расчет:

Прямой ход (с выбором ведущего элемента):

Перестановка строк 1 и 3 (ведущий элемент 3)

Обнуление под диагональю

Обратный ход:

z = 2, y = 3, x = 1

Решение программы (рис.3):

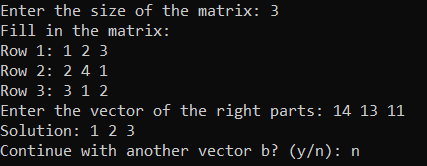


Рисунок 3. Решение программы.

Проверка:

1\*1 + 2\*3 + 3\*2 = 14

2\*1 + 4\*3 + 1\*2 = 13

3\*1 + 1\*3 + 2\*2 = 11

**Пример 3. Вырожденная система**

Исходная система:

x + y = 2

2x + 2y = 4

Ручной анализ:

Определитель матрицы равен 0 → система вырождена.

Второе уравнение является кратным первому → бесконечное число решений.

Реакция программы:

‘The matrix is degenerate’

# Заключение

В ходе выполнения лабораторной работы был разработан программный модуль для решения систем линейных алгебраических уравнений методом Гаусса. Реализация включает полный цикл обработки - от ввода данных до анализа результатов.

Разработанный модуль демонстрирует хорошую точность вычислений и может служить основой для создания более сложных вычислительных систем. Программа представляет практическую ценность как для учебных целей, так и для решения прикладных задач.

# Литература

1. Самарский А.А., Гулин А.В. Численные методы., 2018.
2. Мейерс С. Эффективный и современный C++ (2-е изд.). М.: ДМК Пресс, 2020.

# Приложение

#include <iostream>

#include <cmath>

template <typename T>

class Vector {

protected:

size\_t size;

public:

T\* data;

Vector(size\_t n = 1) : size(n), data(new T[n]()) {}

Vector(const Vector& other) : size(other.size), data(new T[other.size]) {

for (size\_t i = 0; i < size; ++i)

data[i] = other.data[i];

}

~Vector() { delete[] data; }

T& operator[](size\_t index) { return data[index]; }

const T& operator[](size\_t index) const { return data[index]; }

Vector& operator=(const Vector& other) {

if (this != &other) {

delete[] data;

size = other.size;

data = new T[size];

for (size\_t i = 0; i < size; ++i)

data[i] = other.data[i];

}

return \*this;

}

friend std::ostream& operator<<(std::ostream& os, const Vector& v) {

for (size\_t i = 0; i < v.size; ++i)

os << v.data[i] << " ";

return os;

}

friend std::istream& operator>>(std::istream& is, Vector& v) {

for (size\_t i = 0; i < v.size; ++i)

is >> v.data[i];

return is;

}

};

template <typename T>

class Matrix : public Vector<Vector<T>> {

public:

Matrix(size\_t n = 1) : Vector<Vector<T>>(n) {

for (size\_t i = 0; i < n; ++i)

this->data[i] = Vector<T>(n);

}

void swapRows(size\_t i, size\_t j) {

std::swap(this->data[i], this->data[j]);

}

};

template <typename T>

class SLAU : public Matrix<T> {

public:

using Matrix<T>::swapRows;

using Matrix<T>::data;

SLAU(size\_t n) : Matrix<T>(n) {}

Vector<T> solve(Vector<T> b) const {

const T eps = 1e-10;

size\_t n = this->size;

Matrix<T> temp(\*this);

Vector<T> x(n);

for (size\_t k = 0; k < n; ++k) {

size\_t max\_row = k;

T max\_val = std::abs(temp[k][k]);

for (size\_t i = k + 1; i < n; ++i) {

if (std::abs(temp[i][k]) > max\_val) {

max\_val = std::abs(temp[i][k]);

max\_row = i;

}

}

if (max\_val < eps) {

throw std::runtime\_error("The matrix is degenerate");

}

if (max\_row != k) {

temp.swapRows(k, max\_row);

std::swap(b[k], b[max\_row]);

}

for (size\_t i = k + 1; i < n; ++i) {

T factor = temp[i][k] / temp[k][k];

for (size\_t j = k; j < n; ++j)

temp[i][j] -= factor \* temp[k][j];

b[i] -= factor \* b[k];

}

}

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

if (std::abs(temp[i][i]) < eps) {

throw std::runtime\_error("The zero element on the ghosts diagonal");

}

x[i] = b[i];

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

x[i] -= temp[i][j] \* x[j];

x[i] /= temp[i][i];

}

return x;

}

};

#include "Gauss.h"

int main() {

size\_t n;

std::cout << "Enter the size of the matrix: ";

std::cin >> n;

SLAU<double> system(n);

std::cout << "Fill in the matrix:\n";

for (size\_t i = 0; i < n; ++i) {

std::cout << "Row " << (i + 1) << ": ";

for (size\_t j = 0; j < n; ++j)

std::cin >> system[i][j];

}

char choice;

do {

Vector<double> b(n);

std::cout << "Enter the vector of the right parts: ";

std::cin >> b;

Vector<double> x = system.solve(b);

std::cout << "Solution: " << x << "\n";

std::cout << "Continue with another vector b? (y/n): ";

std::cin >> choice;

} while (tolower(choice) == 'y');

return 0;

}