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 СЛУ методом Гаусса»**

**Выполнил**:

студент группы 3824Б1-ПМ1

Блеклов С.С.

**Проверила**:

Бусько П.В.

Нижний Новгород

2025

**Содержание**

[Введение 3](#_Toc182419046)

[Постановка задачи 4](#_Toc182419047)

[Руководство пользователя 5](#_Toc182419048)

[Описание программной реализации 6](#_Toc182419049)

[Результаты экспериментов 7](#_Toc182419050)

[Заключение 8](#_Toc182419051)

[Литература 9](#_Toc182419052)

[Приложение 10](#_Toc182419053)

# Введение

Решение систем линейных уравнений (СЛУ) — базовая задача вычислительной математики. Метод Гаусса, основанный на последовательном исключении переменных, позволяет находить точные решения для квадратных систем. В данной работе реализован улучшенный вариант метода Гаусса с выбором ведущего элемента, что повышает устойчивость алгоритма к ошибкам округления.

**Цель работы**:

* Разработка программы на C++ с использованием шаблонов и наследования.
* Анализ точности и эффективности алгоритма.

# Постановка задачи

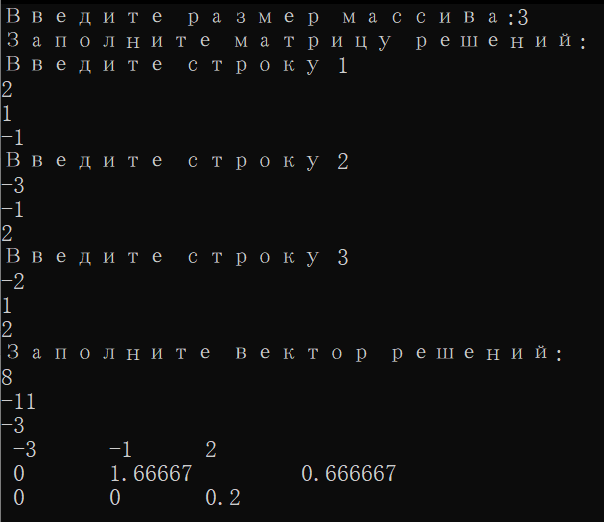
Основная задача: Реализация метода Гаусса с выбором ведущего элемента для решения СЛУ вида **Ax=b**.

Этапы работы:

1. **Реализация:**
   * Создание классов vector, matrix, SLU с использованием шаблонов.
   * Наследование классов для упрощения операций с матрицами и векторами.
   * Реализация прямого хода (исключение переменных) и обратной подстановки.
2. **Тестирование:**
   * Проверка корректности решения для заданных матриц.
   * Анализ устойчивости при работе с вырожденными системами.
3. **Анализ:**
   * Сравнение результатов с аналитическими решениями.

# Руководство пользователя

1. **Запуск программы:**
   * + Пользователь задает матрицу коэффициентов A и вектор правой части b в коде.
     + Пример входных данных:



(Приложение №1 – Пример входных данных)

1. **Выполнение:**
   * + Программа автоматически выполняет прямой ход метода Гаусса и обратную подстановку.
2. **Результат:**
   * + Вектор решений выводится в консоль:



(Приложение №2 – Пример вывода решений)

# Описание программной реализации

**Структура проекта:**

1. gauss.h — заголовочный файл с объявлениями классов.
2. main.cpp — пример использования классов для решения СЛУ.

**Классы:**

1. vector<T> (Приложение №3):
   * Хранит одномерный массив.
   * Методы:
     + Конструкторы: создание вектора заданного размера.
     + operator[]: доступ к элементам.
     + Перегрузка оператора вывода <<.
2. matrix<T> (наследует vector<vector<T>>) (Приложение №4):
   * Хранит двумерный массив.
   * Методы:
     + Конструктор: создание матрицы размером n×m.
     + operator[]: доступ к строкам матрицы.
3. SLU<T> (наследует matrix<T>) (Приложение №5):
   * Решает СЛУ методом Гаусса.
   * Приватные методы:
     + swap\_rows: перестановка строк для выбора ведущего элемента.
     + gaussian\_elimination: прямой ход метода.
     + back\_substitution: обратная подстановка.
   * Публичный метод: get\_solution — возвращает вектор решений.

**Алгоритмы:**

* Прямой ход:
  1. Поиск максимального элемента в столбце для минимизации погрешности.
  2. Перестановка строк.
  3. Исключение элементов ниже ведущего.
* Обратная подстановка:  
  Вычисление неизвестных, начиная с последней переменной.

# Результаты экспериментов

**Тестовая система:**

**2x+y−z=8**

**−3x−y+2z=−11**

**−2x+y+2z=−3**

**Результаты:**

* Полученное решение: x=2.0, y=3.0, z=−1.0
* Решение совпадает с аналитическим.

**Анализ устойчивости:**

* Выбор ведущего элемента предотвратил деление на ноль.
* Алгоритм устойчив для матриц с малыми диагональными элементами.

# Заключение

Работа продемонстрировала эффективность метода Гаусса в сочетании с современными возможностями C++. Использование шаблонов и ООП-подхода не только обеспечило модульность кода, но и сделало его легко адаптируемым под новые требования.

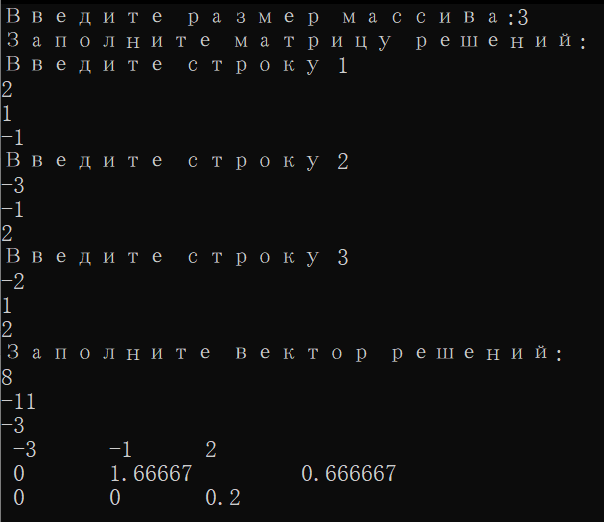
Возникли проблемы с созданием файла реализации функций из файла gauss.h из-за использования шаблонов, появлялась ошибка LNK2019, которую получилось решить только удалением файла реализации и переписыванием кода в заголовочный файл.

# Литература

1. Бахвалов Н.С. Численные методы.
2. Кнут Д.Э. Искусство программирования. Том 2.
3. Документация C++.

# Приложение

Приложение №1:



Приложение №2:



Приложение №3:

template <class T>

class vector {

protected:

int size;

public:

T\* array;

vector() {

size = 1;

array = new T[1];

array[0] = 0;

}

vector(int \_size) {

size = \_size;

array = new T[\_size];

for (int i = 0; i < size; i++) { array[i] = 0; }

}

void set() {

for (int i = 0; i < size; i++) { std::cin >> array[i]; }

}

T& operator[](int n) {

return array[n];

}

friend std::ostream& operator<<(std::ostream& os, const vector a) {

for (int i = 0; i < a.size; i++) {

os << " " << a.array[i] << "\t";

}

os << "\n";

return os;

}

};

Приложение №4:

template <class T>

class matrix :public vector<vector<T>> {

public:

int str, stlb;

matrix(int n, int m) {

str = n;

stlb = m;

this->array = new vector<T>[n];

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

this->array[i] = vector<T>(m);

}

}

friend std::ostream& operator<<(std::ostream& os, const matrix a) {

for (int i = 0; i < a.str; i++) {

os << a.array[i];

}

os << "\n";

return os;

}

vector<T>& operator[](int n) {

return this->array[n];

}

};

Приложение №5:

template <class T>

class SLU : public matrix<T> {

private:

using matrix<T>::array; // SLU dont see array without "using"

using matrix<T>::str; // SLU dont see str without "using"

vector<T> b;

vector<T> x;

void swap\_rows(int row1, int row2) {

vector<T> temp = array[row1];

array[row1] = array[row2];

array[row2] = temp;

T temp\_b = b[row1];

b[row1] = b[row2];

b[row2] = temp\_b;

}

void gaussian\_elimination() {

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

// Searching vedushiy element

int max\_row = k;

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

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

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

max\_row = i;

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

}

}

if (max\_row != k) {

swap\_rows(k, max\_row);

}

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

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

for (int j = k; j < str; ++j) {

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

}

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

}

}

}

void back\_substitution() {

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

x[i] = b[i];

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

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

}

x[i] /= array[i][i];

}

}

public:

SLU(matrix<T>& A, vector<T>& \_b) : matrix<T>(A), b(\_b) {

x = vector<T>(A.str);

gaussian\_elimination();

back\_substitution();

}

vector<T> get\_solution() const {

return x;

}

};