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 группы 382003-1

Викулов М.А.

**Проверил**:

Ассистент каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2021

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 7](#_Toc26962566)

[Результаты экспериментов 8](#_Toc26962567)

[Заключение 9](#_Toc26962568)

[Приложение 10](#_Toc26962569)

# Постановка задачи

Цель моей работы заключалась в реализации метода Гаусса для действительной квадратной матрицы с выбором ведущего элемента на языке программирования С++. Также необходимо было реализовать 3 класса:

1. Шаблонный класс Вектор.
2. Класс Квадратная Матрица, являющаяся шаблоном класса Вектор от Вектора.
3. Класс СЛАУ, который включен в класс Квадратная Матрица и который имеет метод решение системы уравнений методом Гаусса.

Сложность реализованного алгоритма составляет **O(n3).**

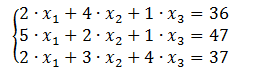
# Метод решения

Рассмотрим поэтапно алгоритм метода Гаусса.

1. Общий вид системы уравнений имеет вид в матричной форме: **А\*х = b**, где

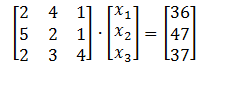
**А –** матрицы коэффициентов, **х –** столбец неизвестных и **y –** столбец свободных членов.

1. Рассмотрим все на следующем иллюстрирующем примере. Допустим, у нас есть следующая система уравнений:

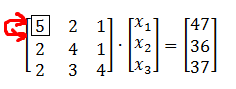


Эквивалентно в матричной

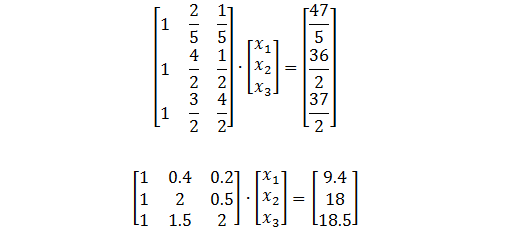
форме.



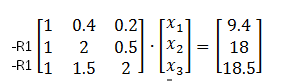
1. Выбираем строку с максимальным коэффициентом **ai1** и меняем её с первой строкой (в данном примере совпало с первой строчкой).

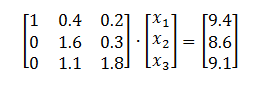


1. Нормируем уравнения (приводим коэффициенты перед **х1** к единицам).

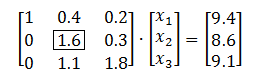


1. Вычитаем из 2 и 3 уравнений 1-е уравнение, тем самым закончив первый шаг приведения матрицы А коэффициентов к верхнетреугольному виду.

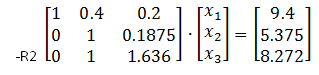




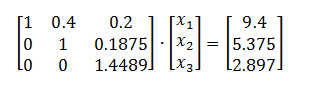
1. Выбираем строку с наибольшим коэффициентом при **ai2** (После первого шаге уравнение 1 уже не рассматривается, поскольку мы занулили коэффициент **a02**) и перемещаем ее на место 2 строки (в данном примере, вторая строка совпала со второй).



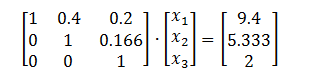
1. Нормируем уравнения 2 и 3 относительно коэффициента при **х2**.



1. Вычитаем из 3 уравнения 2, тем самым заканчивая 2-й шаг приведения матрицы А к верхнетреугольному виду.

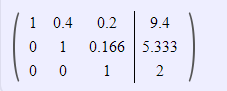


1. Нормируем уравнение 3 относительно коэффициента при **х3**.



1. Получаем верхнетреугольную матрицу А после применения элементарных преобразований.

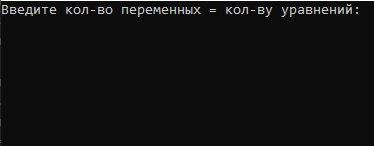
Верхнетреугольная Матрица системы



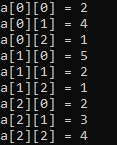
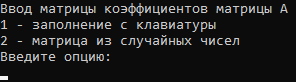
1. Далее начинается обратный ход метода Гаусса. Из нашей преобразованной матрицы находится *x3 = 2*.  Подставляем полученное значение в уравнения 2 и 1 и получаем:  
   https://prog-cpp.ru/wp-content/uploads/2016-12-28_09-36-09.png  
   Подставляя полученное значение **x2** = 5 в уравнение 1, найдем  
   https://prog-cpp.ru/wp-content/uploads/2016-12-28_09-36-22.png
2. Откуда вектор столбец решений: **X = (7 5 2)T**.

# Руководство пользователя

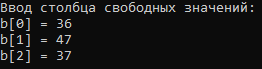
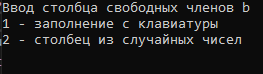
1. Необходимо ввести количество неизвестных (количество уравнений) в системе с клавиатуры.



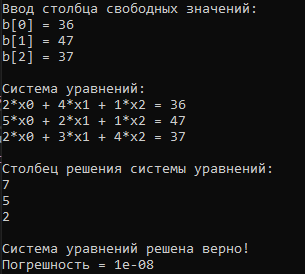
1. После этого пользователю выдается на выбор заполнение матрицы А коэффициентов, где введя 1 – заполнение с клавиатуры, а 2 – заполнение псевдослучайными числами (в данном случае, подразумевается ввод с клавиатуры).



1. Аналогично предлагается заполнение столбца b свободных членов теми же двумя способами.



1. На финальном шаге пользователю выводятся система уравнений в более привычном виде и решение в виде вектора-столбца, после чего выполняется проверка на корректность решения с последующим выводом об ее успешности.



# Описание программной реализации

В моем проекте содержатся 2 файла:

1. **Main.cpp**, в котором содержатся меню для пользователя, объявления классов и проверка корректности алгоритма.
2. Заголовочный файл **VectorsMatrixGauss.h**, в котором описана реализация соответствующих классов.

* Для шаблонного класса **Vector** реализованы следующие перегрузки операторов и описание методов:

1. Присваивание матрицы матрицу.
2. Сложение(вычитание) векторов
3. Умножение(деление) на число
4. Взятие i-й координаты, i – порядок координаты (индекс элемента в массиве)
5. = числу (для последующей перегрузки оператора умножения)
6. Ввод / Вывод
7. Метод типа bool, который проверяет корректность алгоритма

* Для шаблонного класса **Matrix,** являющимся наследником

**<Vector<Vector <T>>** (шаблон класса вектор векторов), реализованы:

1. Перегрузка оператора присваивания матриц
2. Перегрузка оператора умножения матрицы на вектор
3. Перегрузка операторов Ввода / Вывода

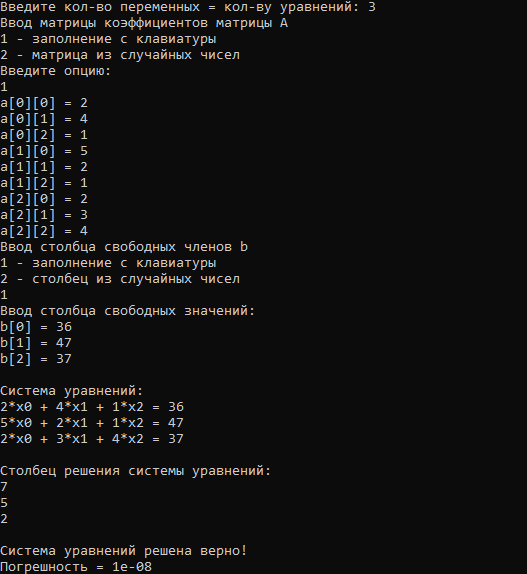
* Для класса **GaussMethod**:

1. Метод вывода системы уравнений в обычной записи
2. Метод **Solve,** содержащий в себе реализацию алгоритма метода Гаусса и возвращающий вектор – столбец решений.

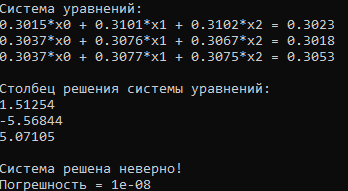
# Подтверждение корректности

Для подтверждения корректности в программе проведем несколько экспериментов с входными данными.

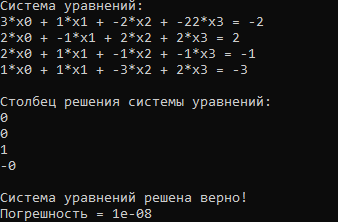
1. Проверим в нашей программе прошлый иллюстрирующий пример, который был приведен в качестве рассмотрения работы алгоритма.



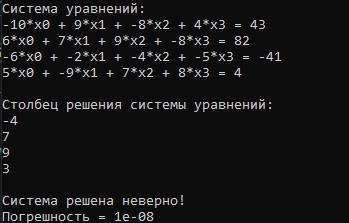
1. Возьмем случайные матрицы А размером 3х3 и вектор b размером = 3.



1. Или размером 4х4



1. И еще одну систему 4х4



И если брать матрицы размером больше 4 со случайными числами, то в большинстве случаев мы будем получать верный математический ответ (подставить вручную и подсчитать), но не программный. Все это происходит из-за арифметических действий с длинными десятичными дробями (вычитание, деление, умножение на длинную дробь). В таком случае, в крайних разрядах будет накапливаться пусть и очень маленькое значение, но кардинально меняющее работы программы (именно поэтому во втором и последнем примерах решение вывелось неверным).

Пример данной проблемы. Сравниваются 2 значения вектора свободных членов (первый – введенный пользователем, второй – полученный в результате умножения матрицы на столбец X при проверке).

≠



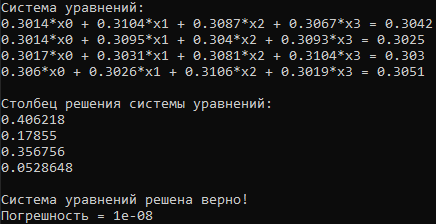
В моем случае, это и являлось причиной «неверно» решенной системы уравнений.

В таком случае я заменил метод **Is\_equal**, который сравнивает поэлементно оператором ==, на функцию **checking**, принимающую в качестве 2 вектора и работающую следующим образом:

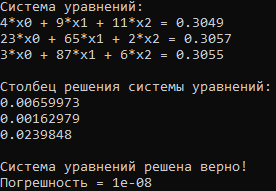
1. Считаю за 0 eps = 10-9 – погрешность.
2. И сравниваю модуль разности точного столбца решения и полученного в моей программе (если <, то считается верным решением, в противном случае, неверным).

Проведем после модификации еще несколько экспериментов для того, чтобы удостовериться в корректности работы данного алгоритма.

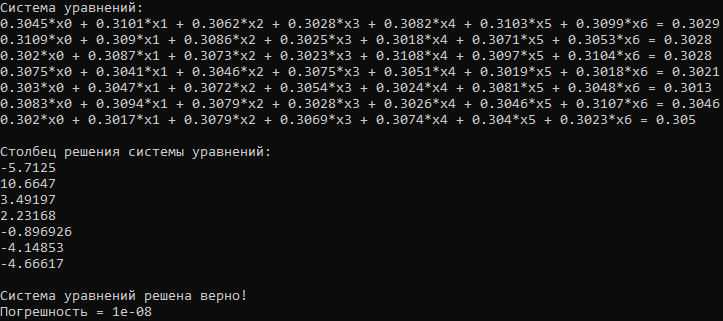
1. Матрица 4х4 с псевдослучайными числами.



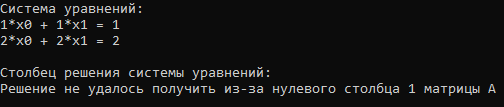
1. Матрица 3х3 с введенной матрицей А и случайным столбцом.



1. Система из 7 уравнений со случайными числами.



1. Система 2х2 из линейно зависимых строк.



# Результаты экспериментов

Исходя из этих экспериментов, мы можем видеть, что реализованный мною алгоритм решения с.л.у. методом Гаусса вполне корректен. Также я проверил работу алгоритма на разных объемах входных данных и зафиксировал примерное время выполнения.

|  |  |
| --- | --- |
| Количество уравнений | Время работы(сек) |
| 10 | 0.1 |
| 100 | 9.365 |
| 1000 | 933.701 |

# Заключение

Исходя из данных экспериментов, можно сделать вывод, что эффективность данного алгоритма O(n3). Также для достижения большей точности необходимо уменьшать погрешность в вычислениях