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 381903-2

Розанов Д.И.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Проверил**:

ассистент каф. МОСТ,

Волокитин В.Д.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_

\_

Нижний   Новгород  
2020

Содержание

[1 Постановка задачи 3](#_Toc40625756)

[2 Методы Решения 4](#_Toc40625757)

[3 Руководство пользователя 6](#_Toc40625758)

[4 Описание программной реализации 9](#_Toc40625759)

[5 Подтверждение корректности 10](#_Toc40625760)

[6 Заключение 11](#_Toc40625761)

[7 Приложение 12](#_Toc40625762)

# Постановка задачи

Основной целью данной работы является реализация алгоритма метода Гаусса.  
Для этого необходимо реализовать:

1. шаблонный класс вектор,
2. класс матрица, которая является шаблоном класса вектор от вектора,
3. класс СЛАУ, наследуемый от класса матрица

Также необходимо проверить корректность работы алгоритма.

# Методы Решения

Реализован шаблонный класс Vector, элемент которого - размер вектора и шаблонный массив v.

Для данного класса написан конструктор по умолчанию, конструктор инициализатор и конструктор копирования. Конструктор по умолчанию создает вектор размером 10. В конструктор-инциализатор передается размер вектора и значение, которым его заполнить.

Пользователь может заполнить созданный вектор самостоятельно.

Реализован класс Matrix – наследник Vector, являющийся шаблоном вектора от вектора. Для данного класса написан конструктор по умолчанию, который вызывает конструктор-инциализатор вектора, передавая в качестве параметров размер вектора и сам вектор.  
Также для решения задачи необходим конструктор копирования, вызывающий конструктор копирования вектора.

Пользователь может заполнить созданную матрицу самостоятельно.

Реализован класс Linear\_equation\_system, наследуемый от класса Matrix.

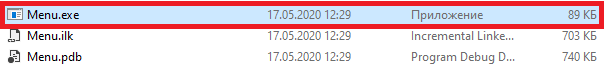
Для данного класса написан конструктор по умолчанию, принимающий размер квадратной матрицы и матрицу для преобразования методом Гаусса \_m: вызывается конструктор копирования класса Matrix для переданной матрицы \_m.

Также написан метод method\_Gauss, в который передается правое значение системы линейных уравнений (). Чтобы не изменить начальную матрицу, создается переменная такая же матрица. Далее, в каждой строке ищется ведущий элемент, меняются местами строки относительно ведущего элемента, каждая строка делится на свой ведущий элемент. Зануляются столбцы, находящиеся под ведущим элементом каждой строки путем вычитания строки, умноженной на нужный элемент, и зануляются столбцы, находящиеся над ведущим элементом. Полученная преобразованная матрица печатается.

Метод Check: проверяет корректность работы метода Гаусса. Умножением начальной матрицы А на полученный вектор х находится вектор b (. Полученное значение b должно совпасть с начальным вектором.

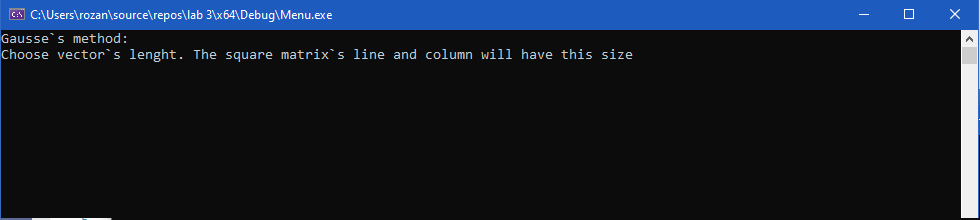
# Руководство пользователя

Для использования программы по нахождению решений системы линейных уравнений необходимо открыть файл Menu.exe. (рис.1.).



1. Скриншот нахождения программы “Menu.exe”.

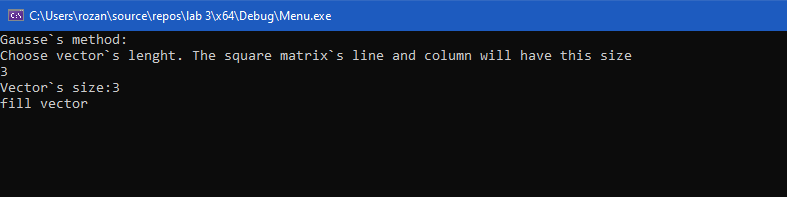
При открытии программы откроется консоль с описанием дальнейших действий (рис.2):



1. Описание дальнейших действий.1

Необходимо выбрать размер квадратной матрицы

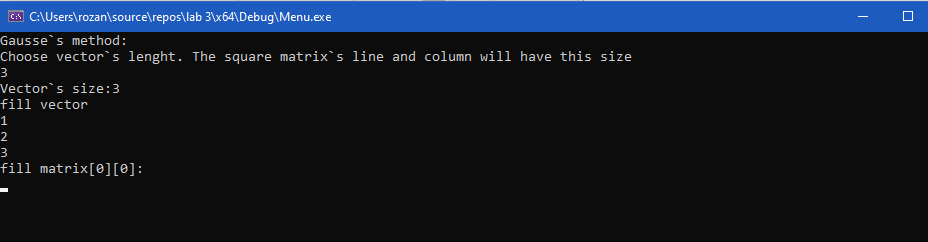
Пусть выбрано число 3 (рис.3):



1. Описание дальнейших действий.2

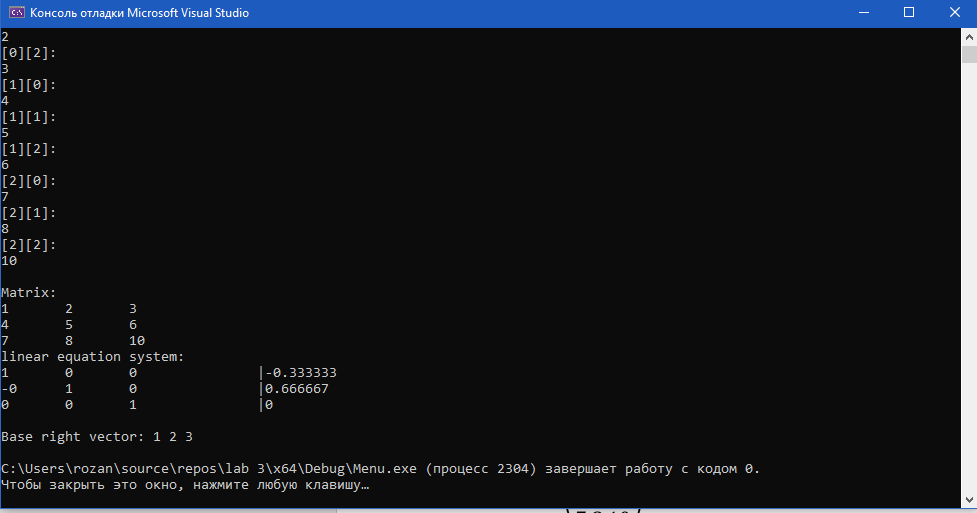
Необходимо заполнить вектор b : .

Пусть вектор (рис 4.):



1. Описание дальнейших действий.3

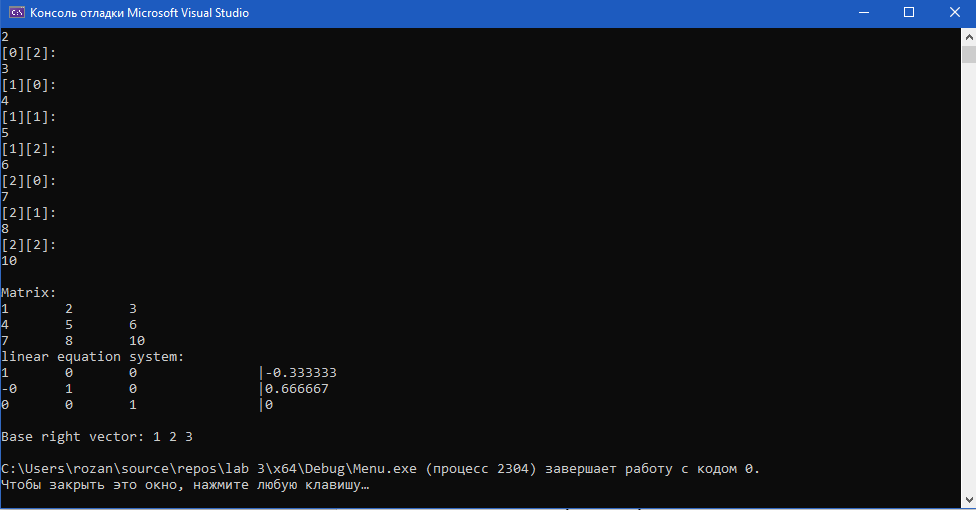
Необходимо заполнить матрицу A: .   
Пусть (рис. 5)



1. Описание дальнейших действий.4

Программа выводит начальную матрицу и систему линейных уравнений, полученную в итоге преобразований.

Чтобы проверить, правильно ли преобразовалась матрица и верно ли найден x, можно вызвать метод Check, который распечатает значение вектора b, полученный в результате умножения начальной матрицы на вектор х () (рис. 6):



1. Вывод вектора b методом Check.

# Описание программной реализации

Программная реализация алгоритмов сортировки была выполнена на языке С++ в среде Microsoft Visual Studio 2019.

В проекте Menu 1 файл .cpp и 3 файлов .h:

1. В файле “Vector.h” реализован шаблонный класс Vector, элементы которого размер вектора, шаблонный массив v. Для данного класса написан конструктор по умолчанию, конструктор-инциализатор, конструктор-копирования, деструктор. Также написаны следующие методы: fill\_v, print\_coord. Перегружен оператор присваивания и оператор [].
2. В файле “Square\_matrix.h” реализован шаблонный класс Matrix, наследуемый от Vector. Для данного класса написан конструктор по умолчанию и конструктор копирования. Написан метод print\_m, fill\_m.

К этому файлу подключен “Vector.h”, для заполнения элементов вектора.

1. В файле “linear\_equation\_system.h” Реализован класс Linear\_equation\_system, наследуемый от класса Matrix. Для данного написан конструктор по умолчанию.

Написан метод method\_Gauss, Check.

К этому файлу подключен “Vector.h”, “Square\_matrix.h”.

# Подтверждение корректности

Для подтверждения корректности алгоритма метода Гаусса вызывается функция Check из класса Linear\_equation\_system: выводится на экран вектор b, полученный умножением (). В случае корректности работы алгоритма вектор b совпадет с начальным вектором.

# Заключение

В ходе лабораторной работы был разработан алгоритм метода Гаусса для решений системы линейных уравнений

Метод Гаусса корректно работает и решает систему линейных уравнений.

# Приложение

void method\_Gauss(Vector<T>& v\_r)

{

Vector<Vector<T>> tmp\_m = \*this;

for (int i = 0; i < this->size; i++)

{

T max = tmp\_m[i][i];

int \_j = i;

for (int j = i + 1; j < this->size; j++)

{

if (tmp\_m[i][j] > max)

{

max = tmp\_m[j][i];

\_j = j;

}

}

Vector<T> \_tmpm(this->size);

T \_tmpl;

\_tmpm = tmp\_m[i];

\_tmpl = v\_r[i];

tmp\_m[i] = tmp\_m[\_j];

v\_r[i] = v\_r[\_j];

tmp\_m[\_j] = \_tmpm;

v\_r[\_j] = \_tmpl;

v\_r[i] = v\_r[i] / tmp\_m[i][i];

T tmp = tmp\_m[i][i];

for (int k = 0; k < this->size; k++)

{

tmp\_m[i][k] = tmp\_m[i][k] / tmp;

}

for (int z = i + 1; z < this->size; z++)

{

T tmp = tmp\_m[z][i];

v\_r[z] = v\_r[z] - v\_r[i] \* tmp;

for (int c = 0; c < this->size; c++)

{

tmp\_m[z][c] = tmp\_m[z][c] - tmp\_m[i][c] \* tmp;

}

}

}

for (int z = this->size - 1; z > 0; z--)

{

for (int k = 0; (k < this->size - 1) && (k != z); k++)

{

v\_r[k] = v\_r[k] - v\_r[z] \* tmp\_m[k][z];

tmp\_m[k][z] = tmp\_m[k][z] - tmp\_m[k][z] \* tmp\_m[z][z];

}

}

std::cout << "linear equation system: " << std::endl;

for (int i = 0; i < this->size; i++)

{

Vector<T> tmp1(this->size);

tmp1 = tmp\_m[i];

for (int j = 0; j < this->size; j++)

{

std::cout << tmp1[j] << "\t";

}

std::cout << "\t|" << v\_r[i];

std::cout << std::endl;

}

}