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 5](#_Toc40625758)

[4 Описание программной реализации 7](#_Toc40625759)

[5 Подтверждение корректности 8](#_Toc40625760)

[6 Заключение 9](#_Toc40625761)

[7 Приложение 10](#_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 класс Vector, элемент которого - массив a размером nsize, созданный функцией rand().

Для данного класса написан конструктор по умолчанию, заполняющий вектор нулевыми элементами.

Реализован класс Matrix, являющийся шаблоном вектора от вектора. Элементы данного класса – 2 вектора, являющиеся шаблоном от вектора, (один создан для проверки правильности алгоритма метода Гаусса, являющийся копией другого вектора) и размер квадратной матрицы.

Для данного класса написан конструктор по умолчанию, принимающий размер квадратной матрицы. Вектор, являющийся шаблоном от вектора, заполняется векторами, элементы которого случайные числа

Реализован класс Linear\_equation\_system, наследуемый от класса Matrix. Элементы данного класса – 2 вектора (один создан для проверки правильности алгоритма метода Гаусса, являющийся копией другого вектора).

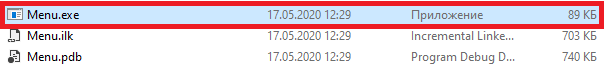
Для данного класса написан конструктор по умолчанию, принимающий размер квадратной матрицы: вызывается конструктор по умолчанию класса Matrix и конструктор по умолчанию класса Vector, создается массив размера nsize, заполненный случайными числами, вектор, являющийся элементом данного класса, заполняется этими числами.

Также написан метод method\_Gauss. Ищется в каждой строке ведущий элемент, меняются местами строки относительно ведущего элемента, каждая строка делится на свой ведущий элемент. Далее зануляются столбцы, находящиеся под ведущим элементом каждой строки путем вычитания строки, умноженной на нужный элемент, и занулятся столбцы, находящиеся над ведущим элементом.

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

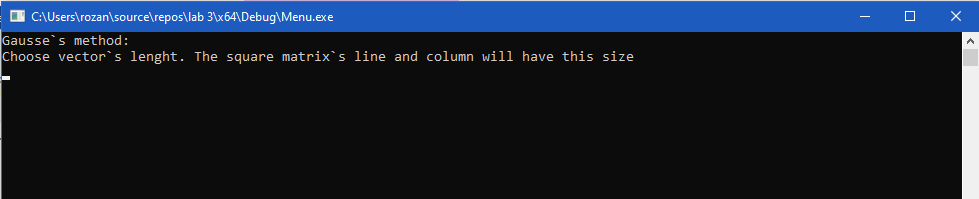
# Руководство пользователя

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



1. Скриншот нахождения программы “Menu.exe”.

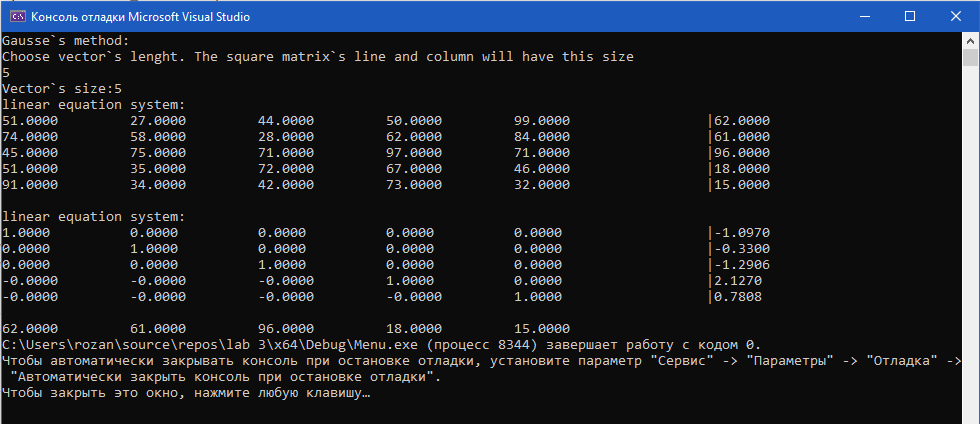
При открытии программы откроется консоль с описанием дальнейших действий (рис.2):



1. Описание дальнейших действий.1

Необходимо выбрать размер квадратной матрицы

Пусть выбрано число 5 (рис.3):



1. Описание дальнейших действий.2

Программа выводит начальную систему линейных уравнений, систему линейных уравнений, полученную в итоге преобразований, вектор b, полученный умножением ().

# 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 4 файлов .h:

1. В файле “ Func\_massive\_for\_creating\_vector.h” реализованы следующие функции: шаблонная функция create\_random\_massive для создания массива случайных числе размера nsize, шаблонная функция create\_null\_massive для создания массива нулевых элементов размером nsize и функция print\_massive, печатающая элементы массива.
2. В файле “Vector.h” реализован шаблонный класс Vector, элементы которого 2 массива размером nsize, один из которых создан функцией create\_random\_massive, другой копия 1го вектора. Для данного класса написан конструктор по умолчанию, конструктор-инциализатор, деструктор. Также написаны следующие методы: Getsize, print\_coord. Перегружен оператор присваивания, бинарный +, -, \*, /,

К этому файлу подключен “Func\_massive\_for\_creating\_vector.h”, для заполнения элементов вектора.

1. В файле “Square\_matrix.h” реализован шаблонный класс Matrix, элементы которого 2 вектора, являющийся шаблоном от вектора (2ой вектора является копией 1го), и размер квадратной матрицы. Для данного класса написан конструктор по умолчанию. Написан метод print\_Matrix.

К этому файлу подключен “Func\_massive\_for\_creating\_vector.h”, для заполнения элементов вектора.

1. В файле “linear\_equation\_system.h” Реализован класс Linear\_equation\_system, наследуемый от класса Matrix. Элемент данного класса – вектор. Для данного написан конструктор по умолчанию. Написан метод method\_Gauss, print\_linear\_equation\_system, Check.

К этому файлу подключен “Func\_massive\_for\_creating\_vector.h”, “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()

{

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

{

T max = this->m[i][i];

int \_j = i;

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

{

if (this->m[j][i] > max)

{

max = this->m[j][i];

\_j = j;

}

}

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

T \_tmpl;

\_tmpm = this->m[i];

\_tmpl = this->v[i];

this->m[i] = this->m[\_j];

this->v[i] = this->v[\_j];

this->m[\_j] = \_tmpm;

this->v[\_j] = \_tmpl;

this->v[i] = this->v[i] / this->m[i][i];

T tmp = this->m[i][i];

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

{

this->m[i][k] = this->m[i][k] / tmp;

}

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

{

T tmp = this->m[z][i];

this->v[z] = this->v[z] - this->v[i] \* tmp;

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

{

this->m[z][c] = this->m[z][c] - this->m[i][c] \* tmp;

}

}

}

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

{

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

{

this->v[k] = this->v[k] - this->v[z] \* this->m[k][z];

this->m[k][z] = this->m[k][z] - this->m[k][z] \* this->m[z][z];

}

}

}