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

**Ивлев А.Д.**

**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

1. Необходимо реализовать шаблонный класс вектор
2. Необходимо реализовать шаблонный класс матриц, который является наследником класса вектор от векторов
3. Необходимо реализовать метод Гаусса для действительной квадратной матрицы с выбором ведущего элемента. Метод Гаусса принимает правую часть (вектор значений) в качестве аргумента и возвращает ответ в виде вектора (вектор решений)

# Метод решения

Основной метод программы - метод Гаусса. Решение СЛАУ путём элементарных преобразований.

Алгоритм принимает квадратную матрицу с коэффициентами и вектор(столбец) значений, что соответствует СЛАУ в матричном виде.

Алгоритм на каждом шаге (номере столбца n начиная с 0 до n-1) выбирает максимальный по модулю элемент в данном столбце начиная с n позиции и его номер строки (m), в котором он находится. Далее алгоритм меняет n-ю строку с строкой m и меняет элементы n и m в векторе значений (если m не n). То есть максимальный элемент попадает на позицию n x n. Затем алгоритм делит на этот элемент всю строку n и число на n-ой позиции в векторе значений. Затем зануляет данный столбец используя элементарные преобразования (\*, -=) в СЛАУ.

Если решений нет (система не совместна, есть нулевая строка с ненулевым значением) или их бесконечно много (существуют зависимые переменные, максимальный элемент 0), то выводит ошибку. Иначе выводит вектор решений.

# Руководство пользователя

Пользователь запускает программу вводит размер матрицы m строк на n столбцов.

Затем выбирает создать случайную матрицу (1) или ввести её самому (2). Так же ему предлагается вывести введённую или созданную матрицу (1 вывести, 2 нет).

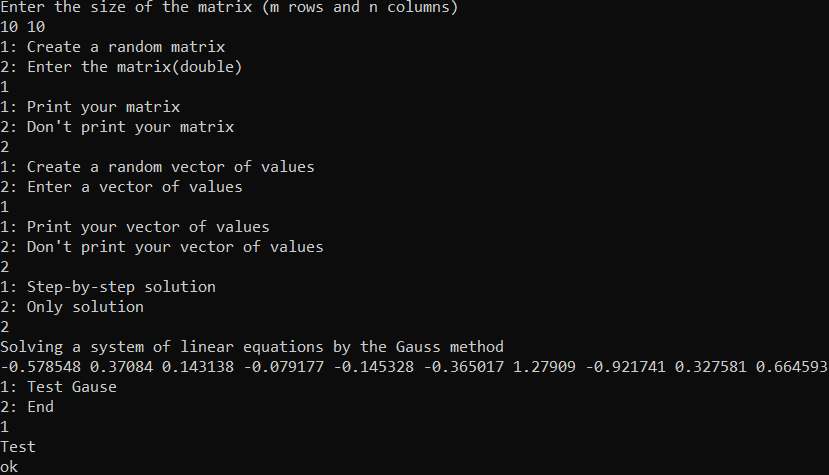
Далее пользователь выбирает создать случайный вектор значений (1) или ввести его самому (2). Так же ему предлагается вывести введённый или созданный вектор значений (1 вывести, 2 нет).

Далее пользователь выбирает вывести пошаговое решение (1) или только вектор решений (2).

Далее программа реализует метод Гаусса для данной матрицы и вектора значений. Выводит ошибку, если матрица не квадратная или СЛАУ не имеет единственного решения. Если решение есть выводит его в качестве вектора решений.

Так же пользователю предлагается проверить решение (1) или закончить программу.

Пример работы программы, где пользователь создал случайную матрицу 10 на 10 и случайный вектор значений.



# Описание программной реализации

Проект состоит из нескольких файлов:

* MainProgramm.cpp - главный файл с main, где подключенны Vector.h и Matrix.h. Реализуется меню для работы с программой.
* Vector.h - содержит шаблонный класс вектор.

Включает методы и поля:

* + size\_t size - размер вектора
  + T\* data - содержимое вектора типа T
  + size\_t get\_size - возвращает размер вектора
  + void rand\_set\_data\_double - случайно заполняет вектор числами типа double
  + MyVec(size\_t \_size = 0) - конструктор с параметром
  + MyVec(const MyVec& sec) - конструктор копирования
  + T& operator [](size\_t i) const - перегруженный оператор []
  + MyVec& operator = (const MyVec& sec) - перегруженный оператор =
  + MyVec operator \* (double sec) - перегруженный оператор \* на число типа double
  + MyVec& operator -= (const MyVec& sec) - перегруженный оператор -=
  + MyVec& operator /= (T sec) - перегруженный оператор /= с числом типа T
  + bool check\_size(const MyVec& sec) - проверка соответствия размеров двух векторов
  + void re\_size(size\_t sec\_size) - изменение размера вектора
  + void swap(size\_t i, size\_t j) - меняет i и j элементы вектора
  + ~MyVec() - деструктор
  + Так же перегружены операторы ввода и вывода
* Matrix.h - содержит шаблонный класс матрицы и метод Гаусса для квадратных матриц типа double.

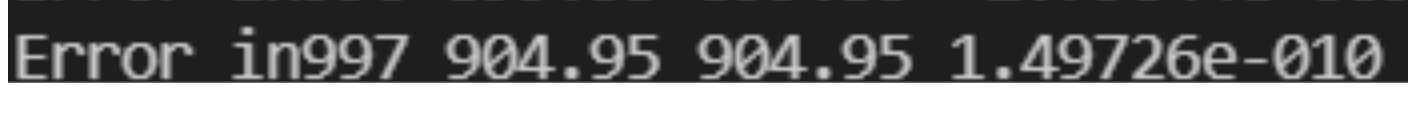
Включает методы и поля:

* + size\_t size\_column - размер столбцов
  + Matrix(size\_t \_size\_line, size\_t \_size\_column) : MyVec<MyVec<Type>> {\_size\_line} - конструктор с параметром
  + void rand\_set\_matrix\_double() - случайно заполняет матрицу числами типа double
  + size\_t max\_in\_column\_i\_line(size\_t column, size\_t start) - находит номер строки с максимальным элементом в данном столбце
  + MyVec<Type> Gauss(MyVec<Type> X, int StepByStep) - метод Гаусса принимает вектор значений (и int StepByStep, если 1, то решает вместе с выводом преобразований после каждого шага), возвращает вектор решений
  + void test(MyVec<Type> & X, MyVec<Type> & \_X) - тест метода Гаусса. Вызывается для начальной матрицы. Принимает сначала вектор решений, затем вектор значений.
  + Matrix operator \* (Type sec) - перегруженный оператор \* на число типа Type
  + Matrix& operator /= (Type sec) - перегруженный оператор /= с числом типа Type
  + Так же перегружены операторы ввода и вывода

# Подтверждение корректности

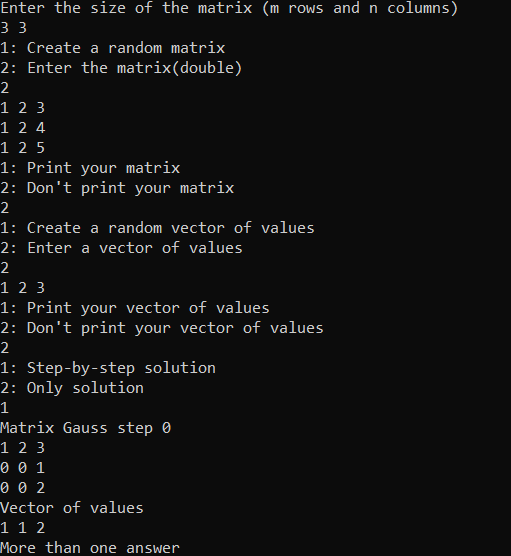
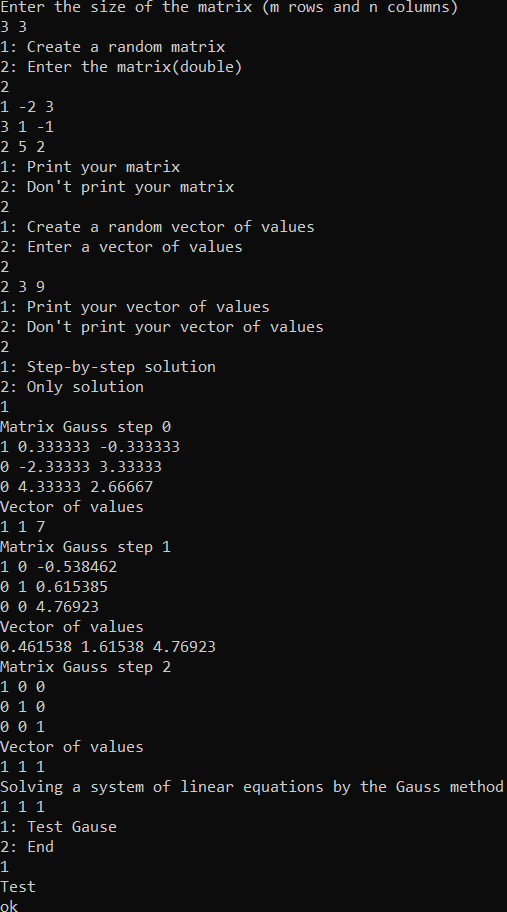
Для подтверждения корректности в программе используется метод матрицы test. Производит умножение исходной матрицы на результат метода Гаусса для сравнения с исходным вектором значений. Если всё правильно и погрешность вычислений меньше 10^-8 выводит ok, иначе выводит номер в векторе решений, где ошибка, элементы вектора решений и вектора значений под данным номером и разницу между ними.

Пример вывода ошибки в 997 строке, где разница между элементами < 10^-8.



# Результаты экспериментов

Пример работы программы с решением по шагам:



В первом случае, программа корректно выполнила алгоритм шаг за шагом и вывела решение. Во втором случае уже после первого шага программа выявила, что решений больше одного, так как существуют зависимые переменные (В данном случае первая переменная зависит от второй).

Во всех случаях программа отработала корректно. Решения были проверены мной вручную и с помощью сервиса <https://matworld.ru/calculator/gauss-method-online.php>.

# Заключение

Мне удалось реализовать шаблонный класс вектор, класс матрицы (наследование от вектора векторов) и метод Гаусса (для матриц типа double, точность не меньше, чем 10^-8.

# Приложение

Смотреть весь код в файлах MainProgramm.cpp, Vector.h, Matrix.h.

Пример реализации метода Гаусса.

MyVec<Type> Gauss(MyVec<Type> X, int StepByStep) // For type double, With an accuracy of 10^-8.

//If StepByStep != 1 step-by-step solution

{

Matrix<Type> Copy = \*this;

if ((this->size == size\_column) && (this->size == X.get\_size()))

{

size\_t MaxValuei;

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

{

MaxValuei = max\_in\_column\_i\_line(i, i);

Type MaxV = this->data[MaxValuei][i];

if (MaxValuei > i)

{

this->swap(MaxValuei, i);

X.swap(MaxValuei, i);

}

MaxValuei = i;

if (abs(MaxV) != 0)

{

this->data[MaxValuei] /= MaxV;

X[MaxValuei] /= MaxV;

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

{

if (j != MaxValuei)

{

X[j] -= (X[MaxValuei]\*this->data[j][i]);

this->data[j] -= (this->data[MaxValuei]\*this->data[j][i]);

}

}

if (StepByStep != 2)

{

cout << "Matrix Gauss step " << i << endl;

cout << \*this;

cout << "Vector of values" << endl;

cout << X;

}

}

else

{

cout << "More than one answer" << endl;

system("pause");

exit(1);

}

}

for (size\_t i = 0; i < X.get\_size(); i++)

{

Type a = X[i];

if (abs(a) < 0.00000001)

{

X[i] = 0;

}

}

\*this = Copy;

return X;

}

else

{

cout << "Error size" << endl;

system("pause");

exit(0);

}

}