МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Программная инженерия»

**ОТЧЕТ**

по лабораторной работе

**ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МЕТОДА ГАУССА**

**Выполнил:** студент группы

3822Б1ПР2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.А.Ермолаев

Подпись

**Проверил:** д.ф.-м.н., доц.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.А. Баркалов

Подпись

Нижний Новгород  
2023

**Содержание**

[1.ВВЕДЕНИЕ 3](#_Toc130390192)

[2.ПОСТАНОВКА ЗАДАЧИ 4](#_Toc130390193)

[3.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc130390194)

[3.1 Последовательное исключение 5](#_Toc130390195)

[3.2 Обратный ход: 6](#_Toc130390196)

[4.ОПИСАНИЕ ПРОГРАММЫ 7](#_Toc130390197)

[4.1 Пользовательские типы данных 7](#_Toc130390198)

[4.2 Остальные функции 7](#_Toc130390199)

[5.РЕЗУЛЬТАТЫ 8](#_Toc130390200)

[6.ЗАКЛЮЧЕНИЕ 9](#_Toc130390201)

[8.ЛИТЕРАТУРА 10](#_Toc130390202)

[9.ПРИЛОЖЕНИЕ 1 11](#_Toc130390203)

[10.ПРИЛОЖЕНИЕ 2 12](#_Toc130390204)

[11.ПРИЛОЖЕНИЕ 3 15](#_Toc130390205)

[12.ПРИЛОЖЕНИЕ 4 16](#_Toc130390206)

[13.ПРИЛОЖЕНИЕ 5 20](#_Toc130390207)

# 1.ВВЕДЕНИЕ

Что вы будете делать, если вас попросят решить простенькую систему с тремя неизвестными? Каждый выберет свой собственный и наиболее удобный лично для него подход. Существует масса способов решить систему линейных алгебраических уравнений (СЛАУ). Но почему предпочтение отдается именно методу Гаусса?

Рассмотрим другой способ решения СЛАУ– метод Крамера. Он заключается в подсчёте определителей размера , в каждом из таких определителей , где – количество уравнений в СЛАУ. При достаточно больших это очень затратно, преимущество метода Гаусса очевидно.

# 2.ПОСТАНОВКА ЗАДАЧИ

Реализовать решение систем линейных алгебраических уравнений методом Гаусса средствами языка C++. Посчитать точность вычислений с помощью подстановки корней в исходные уравнения. Найти и оценить максимальную погрешность.

# 3.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Пусть наша СЛАУ имеет вид:

Матричный вид:

## 3.1 Последовательное исключение

Исключения Гаусса основаны на идее последовательного исключения переменных по одной до тех пор, пока не останется только одно уравнение с одной переменной в левой части. Затем это уравнение решается относительно единственной переменной. Таким образом, систему уравнений приводят к треугольной (ступенчатой) форме. Для нормируют все уравнения, разделив его на коэффициент , где **i**– номер столбца.

Затем вычитают получившуюся после перестановки первую строку из остальных строк:

Получают новую систему уравнений, в которой заменены соответствующие коэффициенты.

После того, как указанные преобразования были совершены, первую строку и первый столбец мысленно вычёркивают и продолжают указанный процесс для всех последующих уравнений пока не останется уравнение с одной неизвестной:

## 3.2 Обратный ход:

Вычисляем корни уравнения в обратном порядке:

## 3.3 Дополнение алгоритма

В случае, когда на главной диагонали матрицы коэффициентов встречается ноль, мы найдём максимум в столбце под этим нулём и поменяем найденную строку и строку, содержащую ноль, местами.

# 4.ОПИСАНИЕ ПРОГРАММЫ

## 4.1 Пользовательские типы данных

class Matrix

{

private:

double\*\* matrix;

int rows, colomns;

public:

Matrix(const int \_rows = 1, const int \_colomns = 1);

Matrix(const Matrix& other);

~Matrix();

template <class Type> requires requires(Type t) { t \* 1.; }

Matrix operator\* (const Type coefficient);

Matrix operator+ (const Matrix& other);

Matrix operator- (Matrix B);

Matrix operator\* (Matrix& other);

Matrix& operator= (const Matrix& other);

double\* operator[] (const int index);

Matrix transposition();

int getRows();

int getColomns();

void swapRows(const int i, const int j);

void swapColomns(const int i, const int j);

void print();

};

Класс Matrix реализует поведение матрицы с поддержкой математических операций: сложение матриц, вычитание матриц, умножение матриц, транспонирование. В классе реализован доступ к размерам матрицы через getRows и getColomns. Присутствуют методы обмена местами строк и столбцов, а также метод вывода матрицы в консоль.

## 4.2 Остальные функции

void initSystemOfEquations(Matrix& A, Matrix& b); - инициализация системы уравнений (если количество уравнений < 5 сработает ручной ввод, иначе случайное заполнение)

void printSystemofEquations(Matrix& A, Matrix& b); - вывод системы уравнений в консоль

void solveSystemOfEquations0(Matrix A, Matrix b, Matrix& x); - решение системы уравнений методом Гаусса

void solveSystemOfEquations1(Matrix A, Matrix b, Matrix& x); - решение системы уравнений методом Гаусса с перемещением нулей с главной диагонали

void saveSystemToFile(Matrix& A, Matrix& b, std::string file = "data.txt"); - сохранение системы уравнений в файл

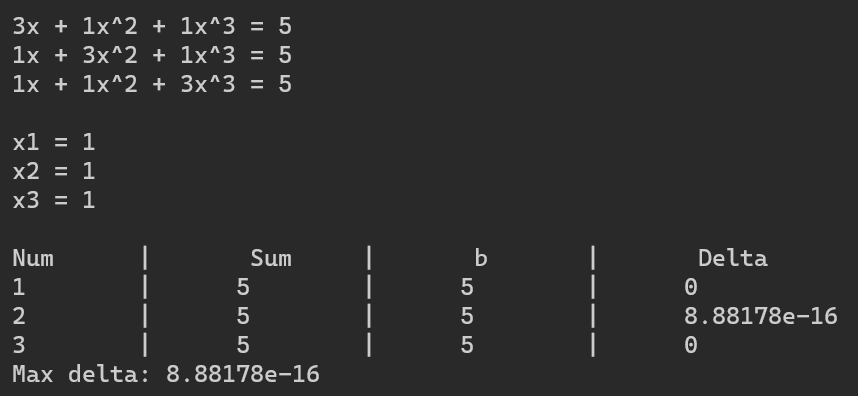
void loadSystemFromFile(Matrix& A, Matrix& b, std::string file = "data.txt"); - загрузка сохраненной системы уравнений из файла

void checkRoots(Matrix& A, Matrix& b, Matrix& x); - проверка корней методом постановки, вычисление и вывод в консоль погрешности

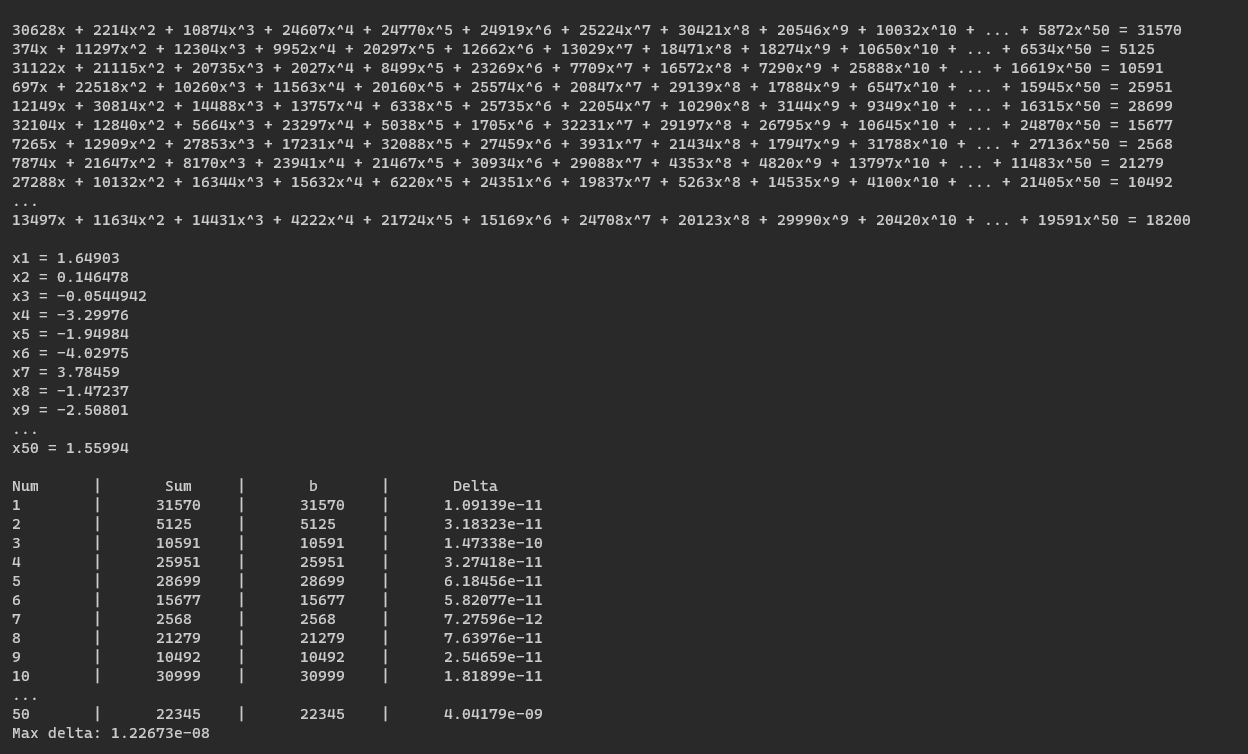
void printRoots(Matrix& x); - вывод корней в консоль

# 5.РЕЗУЛЬТАТЫ

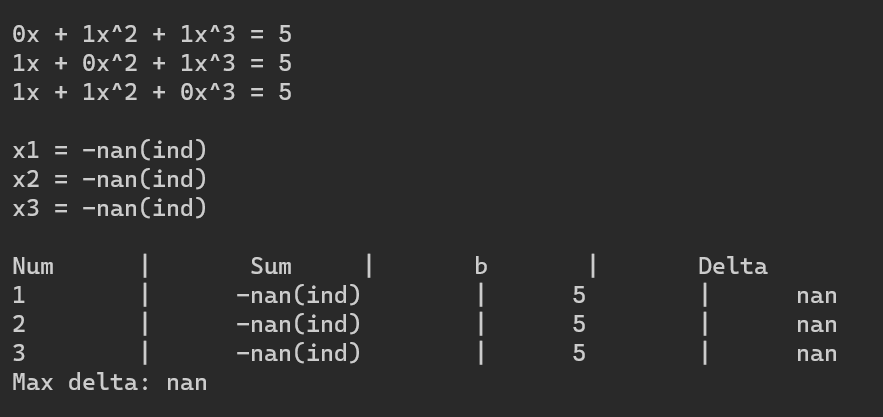
На заданной системе небольшого размера (3 уравнения) максимальная погрешность программы составляет: 8.88178e-16



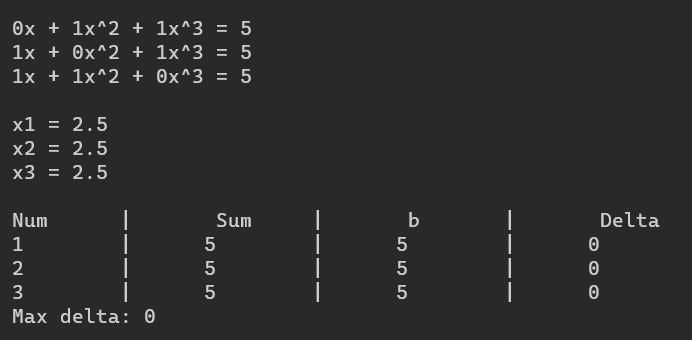
На системе большого размера (50 уравнений), уравнения которой заполнены случайными числами максимальная погрешность: 1.22673e-08



В случае, когда на главной диагонали встречается ноль, решение без дополнения ведёт себя некорректно:



В свою очередь, решение с дополнением может решить подобные системы.



# 6.ЗАКЛЮЧЕНИЕ

Было реализовано решение систем линейных алгебраических уравнений методом Гаусса средствами языка C++. Посчитана точность вычислений с помощью подстановки корней в исходные уравнения. Была найдена и оценена максимальная погрешность.

# 8.ЛИТЕРАТУРА

1. **Интернет-ресурс.** <https://ru.wikipedia.org/wiki/Метод_Гаусса> (22.03.2023)

# 9.ПРИЛОЖЕНИЕ 1

Matrix.h

#pragma once

#include <cassert>

#include <iostream>

class Matrix

{

private:

double\*\* matrix;

int rows, colomns;

public:

Matrix(const int \_rows = 1, const int \_colomns = 1);

Matrix(const Matrix& other);

~Matrix();

template <class Type> requires requires(Type t) { t \* 1.; }

Matrix operator\* (const Type coefficient);

Matrix operator+ (const Matrix& other);

Matrix operator- (Matrix B);

Matrix operator\* (Matrix& other);

Matrix& operator= (const Matrix& other);

double\* operator[] (const int index);

Matrix transposition();

int getRows();

int getColomns();

void swapRows(const int i, const int j);

void swapColomns(const int i, const int j);

void print();

};

# 10.ПРИЛОЖЕНИЕ 2

Matrix.cpp:

#include "Matrix.h"

Matrix::Matrix(const int \_rows, const int \_colomns)

{

rows = \_rows;

colomns = \_colomns;

matrix = new double\* [rows];

for (int i = 0; i < rows; i++)

{

matrix[i] = new double[colomns] { 0 };

}

}

Matrix::Matrix(const Matrix& other)

{

this->rows = other.rows;

this->colomns = other.colomns;

this->matrix = new double\* [this->rows];

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

{

this->matrix[i] = new double[this->colomns];

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

{

this->matrix[i][j] = other.matrix[i][j];

}

}

}

Matrix::~Matrix()

{

for (int i = 0; i < rows; i++)

delete[] matrix[i];

delete[] matrix;

}

Matrix Matrix::operator+ (const Matrix& other)

{

Matrix result(this->rows, this->colomns);

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

{

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

{

result.matrix[i][j] = this->matrix[i][j] + other.matrix[i][j];

}

}

return result;

}

template <class Type> requires requires(Type t) { t \* 1.; }

Matrix Matrix::operator\* (const Type coefficient)

{

Matrix result(this->rows, this->colomns);

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

{

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

{

result.matrix[i][j] = this->matrix[i][j] \* coefficient;

}

}

return result;

}

Matrix Matrix::operator- (Matrix B)

{

return \*this + B \* -1;

}

Matrix Matrix::operator\* (Matrix& other)

{

Matrix result(this->rows, other.colomns);

for (int i = 0; i < result.rows; i++)

{

for (int j = 0; j < result.colomns; j++)

{

result[i][j] = 0;

for (int i1 = 0; i1 < this->colomns; i1++)

{

result[i][j] += this->matrix[i][i1] \* other[i1][j];

}

}

}

return result;

}

Matrix& Matrix::operator= (const Matrix& other)

{

if (this == &other)

return \*this;

if (this->matrix)

{

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

delete[] matrix[i];

delete[] matrix;

}

this->rows = other.rows;

this->colomns = other.colomns;

this->matrix = new double\* [this->rows];

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

{

this->matrix[i] = new double[this->colomns];

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

{

this->matrix[i][j] = other.matrix[i][j];

}

}

return \*this;

}

Matrix Matrix::transposition()

{

Matrix result(this->colomns, this->rows);

for (int i = 0; i < result.rows; i++)

{

for (int j = 0; j < result.colomns; j++)

{

result[i][j] = this->matrix[j][i];

}

}

return result;

}

double\* Matrix::operator[] (const int index)

{

return matrix[index];

}

int Matrix::getRows()

{

return rows;

}

int Matrix::getColomns()

{

return colomns;

}

void Matrix::swapRows(const int i, const int j)

{

double\* tmp = matrix[i];

matrix[i] = matrix[j];

matrix[j] = tmp;

}

void Matrix::swapColomns(const int i, const int j)

{

for (int k = 0; k < rows; k++)

{

std::swap(matrix[k][i], matrix[k][j]);

}

}

void Matrix::print()

{

std::cout << "Matrix: " << std::endl;

for (int i = 0; i < std::min(rows, 10); i++)

{

for (int j = 0; j < std::min(colomns, 10); j++)

std::cout << matrix[i][j] << "\t";

if (colomns > 10) std::cout << "...\t";

std::cout << std::endl;

}

if (rows > 10) std::cout << "...\t" << std::endl;

std::cout << std::endl;

};

# 11.ПРИЛОЖЕНИЕ 3

Other Functions.h:

#pragma once

#include "Matrix.h"

#include <fstream>

void initSystemOfEquations(Matrix& A, Matrix& b);

void printSystemofEquations(Matrix& A, Matrix& b);

void solveSystemOfEquations0(Matrix A, Matrix b, Matrix& x);

void solveSystemOfEquations1(Matrix A, Matrix b, Matrix& x);

void saveSystemToFile(Matrix& A, Matrix& b, std::string file = "data.txt");

void loadSystemFromFile(Matrix& A, Matrix& b, std::string file = "data.txt");

void checkRoots(Matrix& A, Matrix& b, Matrix& x);

void printRoots(Matrix& x);

# 12.ПРИЛОЖЕНИЕ 4

Other Functions.cpp:

#include "Other Functions.h"

void initSystemOfEquations(Matrix& A, Matrix& b)

{

int N, M;

std::cout << "Number of equations: ";

std::cin >> N;

std::cout << "Number of variables: ";

std::cin >> M;

std::cout << std::endl;

A = Matrix(N, M);

b = Matrix(N);

if (A.getRows() <= 4 && A.getColomns() <= 5)

{

for (int i = 0; i < A.getRows(); i++)

{

std::cout << "Enter a row " << i + 1 << " of the matrix A: ";

for (int j = 0; j < A.getColomns(); j++)

std::cin >> A[i][j];

std::cout << "Enter a b" << i + 1 << ": ";

std::cin >> b[i][0];

}

std::cout << std::endl;

}

else

for (int i = 0; i < A.getRows(); i++)

{

for (int j = 0; j < A.getColomns(); j++)

A[i][j] = rand();

b[i][0] = rand();

}

}

void printSystemofEquations(Matrix& A, Matrix& b)

{

for (int i = 0; i < std::min(A.getRows(), 10); i++)

{

if (i == 9 && A.getRows() > 10)

{

std::cout << "..." << std::endl;

i++;

}

for (int j = 0; j < std::min(A.getColomns(), 10); j++)

{

std::cout << A[i][j] << "x";

if (j != 0)

std::cout << "^" << j + 1;

std::cout << " ";

if (j != A.getColomns() - 1)

std::cout << "+ ";

if (j == 9 && A.getColomns() > 10)

std::cout << "... + " << A[i][A.getColomns()-1] << "x^" << A.getColomns() << " ";

}

std::cout << "= " << b[i][0] << std::endl;

}

}

void solveSystemOfEquations0(Matrix A, Matrix b, Matrix& x)

{

double sum;

for (int j = 0; j < A.getColomns(); j++)

{

for (int i = j + 1; i < A.getRows(); i++)

{

double coefficient = A[i][j] / A[j][j];

for (int k = j + 1; k < A.getColomns(); k++)

A[i][k] -= A[j][k] \* coefficient;

b[i][0] -= b[j][0] \* coefficient;

}

}

for (int i = A.getRows() - 1; i >= 0; i--)

{

sum = 0;

for (int j = i + 1; j < A.getColomns(); j++)

{

sum += A[i][j] \* x[j][0];

}

x[i][0] = (b[i][0] - sum) / A[i][i];

}

}

void solveSystemOfEquations1(Matrix A, Matrix b, Matrix& x)

{

double sum;

for (int j = 0; j < A.getColomns(); j++)

{

if (A[j][j] == 0)

{

int max\_index = j;

double max\_element = 0;

for (int i = j + 1; i < A.getRows(); i++)

{

if (abs(A[i][j]) > max\_element)

{

max\_index = i;

max\_element = A[i][j];

}

}

A.swapRows(max\_index, j);

b.swapRows(max\_index, j);

}

for (int i = j + 1; i < A.getRows(); i++)

{

double coefficient = A[i][j] / A[j][j];

for (int k = j + 1; k < A.getColomns(); k++)

A[i][k] -= A[j][k] \* coefficient;

b[i][0] -= b[j][0] \* coefficient;

}

}

for (int i = A.getRows() - 1; i >= 0; i--)

{

sum = 0;

for (int j = i + 1; j < A.getColomns(); j++)

{

sum += A[i][j] \* x[j][0];

}

x[i][0] = (b[i][0] - sum) / A[i][i];

}

}

void saveSystemToFile(Matrix& A, Matrix& b, std::string file)

{

std::ofstream out(file);

out << A.getRows() << " " << A.getColomns() << std::endl << std::endl;

for (int i = 0; i < A.getRows(); i++)

{

for (int j = 0; j < A.getColomns(); j++)

{

out << A[i][j] << " ";

}

out << std::endl;

}

out << std::endl;

for (int i = 0; i < b.getRows(); i++)

{

out << b[i][0] << " ";

}

out.close();

}

void loadSystemFromFile(Matrix& A, Matrix& b, std::string file)

{

try {

std::ifstream in(file);

in.exceptions(std::ifstream::failbit);

int N, M;

in >> N >> M;

A = Matrix(N, M);

b = Matrix(N);

for (int i = 0; i < A.getRows(); i++)

for (int j = 0; j < A.getColomns(); j++)

in >> A[i][j];

for (int i = 0; i < b.getRows(); i++)

in >> b[i][0];

in.close();

}

catch (std::ios\_base::failure&) {

std::cerr << "Couldn't open the file! Enter The System manually.\n";

initSystemOfEquations(A, b);

return;

}

}

void checkRoots(Matrix& A, Matrix& b, Matrix& x)

{

double sum = 0, maxDelta = 0, delta = 0;

std::cout << "Num \t | \t Sum \t | \t b \t | \t Delta" << std::endl;

for (int i = 0; i < A.getRows(); i++)

{

sum = 0;

for (int j = 0; j < A.getColomns(); j++)

{

sum += A[i][j] \* x[j][0];

}

delta = abs(b[i][0] - sum);

maxDelta = std::max(delta, maxDelta);

if (i < std::min(A.getRows() - 1, 10))

std::cout << i + 1 << "\t | \t" << sum << "\t | \t" << b[i][0] << "\t | \t" << delta << std::endl;

}

if (A.getRows() > 10)

std::cout << "..." << std::endl;

std::cout << A.getRows() << "\t | \t" << sum << "\t | \t" << b[A.getRows()-1][0] << "\t | \t" << delta << std::endl;

std::cout << "Max delta: " << maxDelta << std::endl << std::endl;

}

void printRoots(Matrix& x)

{

for (int i = 0; i < std::min(x.getRows(), 10); i++)

{

if (i == 9 && x.getRows() > 10)

std::cout << "..." << std::endl << "x" << x.getRows() << " = " << x[x.getRows()-1][0] << std::endl;

else

std::cout << "x" << i + 1 << " = " << x[i][0] << std::endl;

}

}

# 13.ПРИЛОЖЕНИЕ 5

Source.cpp:

#include <iostream>

#include <stdlib.h>

#include <time.h>

#include "Matrix.h"

#include "Other Functions.h"

int main()

{

srand((unsigned)time(nullptr));

int op;

Matrix A, b;

while (true)

{

std::cout << "Load a saved System? (1/0): ";

std::cin >> op;

switch (op)

{

case 1:

loadSystemFromFile(A, b);

break;

default:

initSystemOfEquations(A, b);

break;

}

Matrix x(A.getRows());

std::cout << std::endl;

printSystemofEquations(A, b);

std::cout << std::endl;

//solveSystemOfEquations0(A, b, x);

//printRoots(x);

//std::cout << std::endl;

//checkRoots(A, b, x);

solveSystemOfEquations1(A, b, x);

printRoots(x);

std::cout << std::endl;

checkRoots(A, b, x);

std::cout << "Save the System? (1/0): ";

std::cin >> op;

if (op) saveSystemToFile(A, b);

std::cout << "Repeat again? (1/0): ";

std::cin >> op;

if (op == 0)

break;

if (op == 1)

system("cls");

}

system("cls");

std::cout << "Created by @Yermolaev Vladislav" << std::endl;

return 0;

}