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

Варварин Е.М.

**Проверил**:

ассистент каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2019

**Содержание**

[Постановка задачи 3](#_Toc41942901)

[Метод решения 4](#_Toc41942902)

[Описание программной реализации 7](#_Toc41942903)

[Подтверждение корректности 9](#_Toc41942904)

[Заключение 10](#_Toc41942905)

[Приложение 11](#_Toc41942906)

# Постановка задачи

Целью данной лабораторной работы является реализация метода Гаусса для квадратной матрицы с выбором ведущего элемента.

Для этого:

Необходимо создать шаблонный класс вектор.

Далее реализовать шаблонный класс квадратная матрица, которая является шаблоном класса вектор от вектора.

Затем нужно реализовать шаблонный класс СЛАУ, который наследуется от матрицы и для него написать метод Гаусса.

# Метод решения

Реализован шаблонный класс MyVector, хранящий в себе массив заданного размера и его размер. Для него созданы:

* Конструктор по умолчанию, создающий вектор размером 2 со значением 0.
* Написан конструктор копирования.
* Перегружен оператор =.
* Перегружен оператор [ ].
* Метод fill позволяет задать значения для вектора.
* Метод print выводит значения вектора в консоль.
* Деструктор.

Реализован класс Matrix, который является шаблоном класса MyVector от MyVector. Для данного класса реализованы:

* Метод fill для задания значений матрицы.
* Конструктор по умолчанию.
* Метод print выводит матрицу в консоль.
* Реализован класс SoHAE (СЛАУ), являющийся наследником класса Matrix.
* В нем реализованы:
* Конструктор по умолчанию.
* Метод print, выводящий в консоль расширенную матрицу.

Метод gauss – принимает в качестве аргумента массив с элементами правой части СЛАУ. Метод приводит СЛАУ к треугольному виду, затем решает СЛАУ и выводит вектор значений СЛАУ.

* А также приватные методы swap и subtractingRows, необходимые для метода gauss.

Реализована шаблонная функция module, возвращающая модуль числа.

Руководство пользователя

При запуске программы пользователь должен ввести размер матрицы:

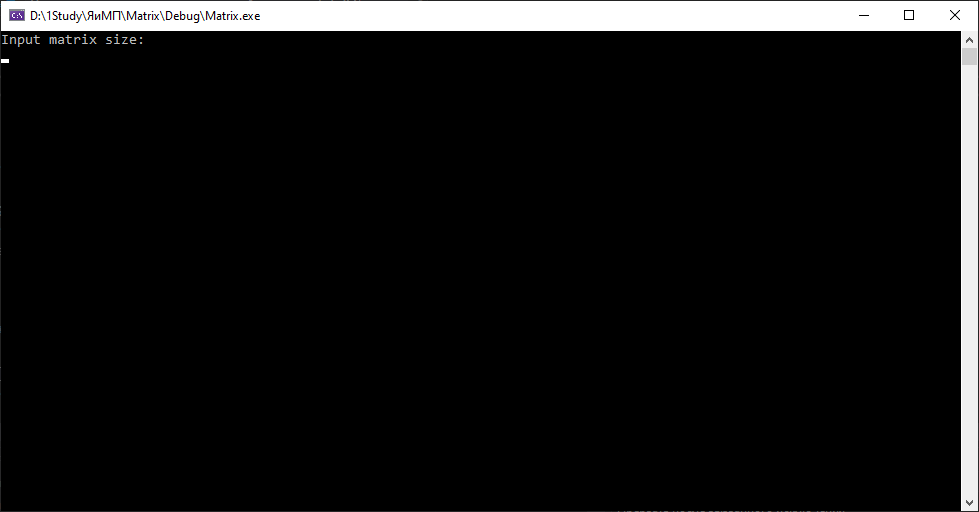


Рисунок 1. Ввод размера

Далее нужно ввести значения правой части

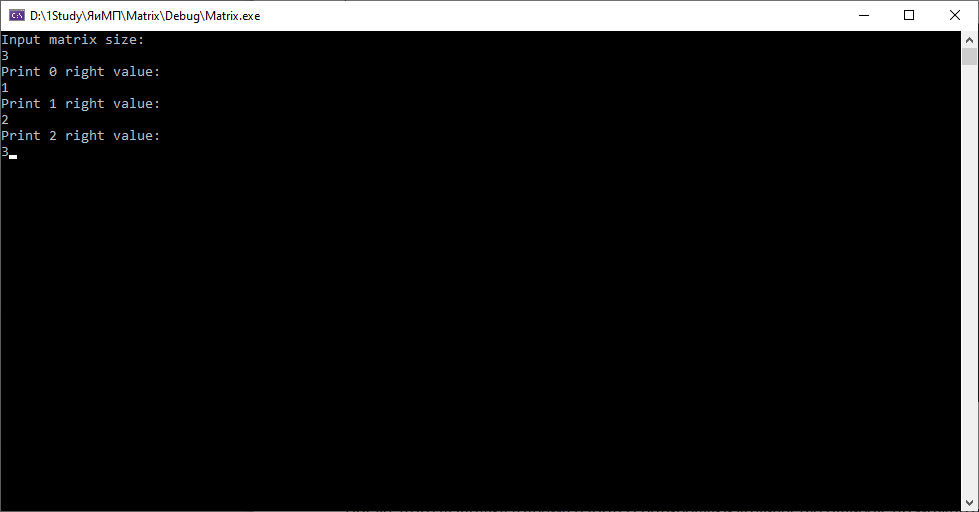


Рисунок 2. Ввод значений правой части

Затем построчно вводим элементы правой части

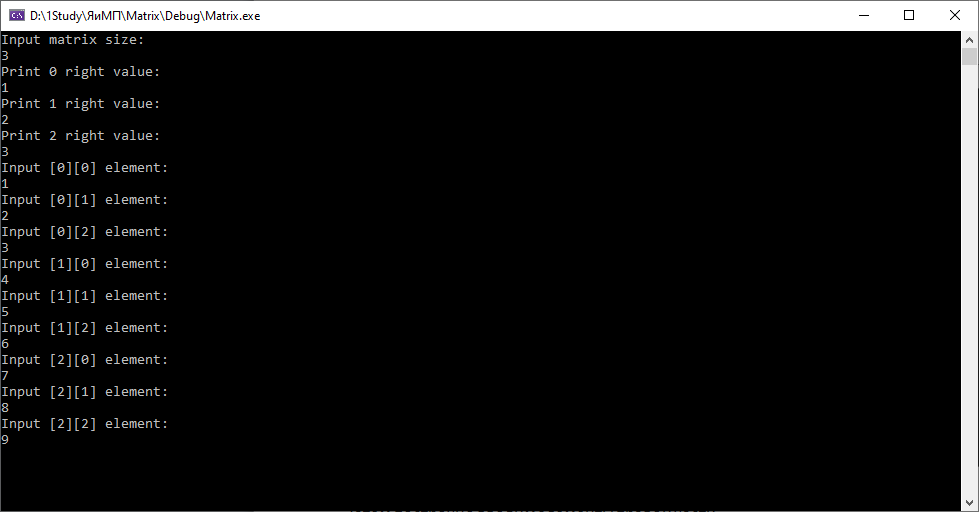


Рисунок 3. Ввод значений матрицы системы

После ввода всех данных в консоль выводится введенная СЛАУ, матрица, приведенная к ступенчатому виду, вектор решений системы - если матрица имеет единственное решение. В противном случае выводится сообщение об ошибке.

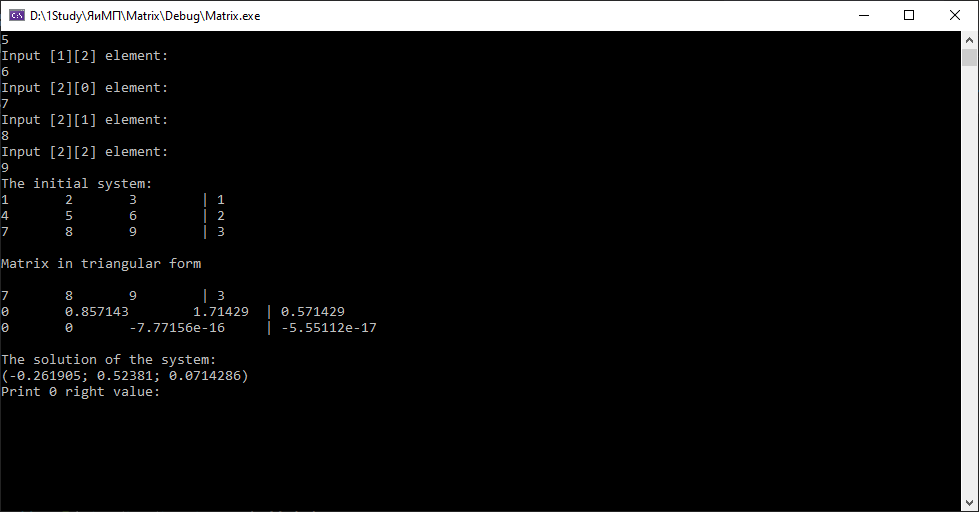


Рисунок 4. Результат выполнения программы

# Описание программной реализации

Программа состоит из 5 файлов.

* Главный файл – matrix.cpp

В этом файле подключены все заголовочные файлы и вызван метод Гаусса.

* ClassVector.h

Содержит реализацию класса MyVector вместе с его методами:

template < typename T >

class MyVector {

public:

MyVector() {…}

MyVector(int \_size, T value = T()) {…}

MyVector(int \_size, T\* values) {…}

MyVector& operator = (const MyVector& other) {…}

MyVector(const MyVector& other) {…}

void fill() {…}

~MyVector() {…}

void print() {…}

T& operator[](int i) {…}

int getSize() {…}

protected:

T\* m\_x;

int size;

};

* Matrix.h

Содержит реализацию класса Matrix:

template <typename T>

class Matrix : public MyVector<MyVector <T> > {

public:

Matrix(int n): MyVector<MyVector <T>>(n, MyVector <T>(n)) {}

void print() {…}

void fill() {…}

};

* SoHAE.h

Содержит реализацию класса SoHAE:

template <typename T>

class SoHAE : public Matrix<T> {

public:

SoHAE(int n): Matrix<T>(n) {…};

void print(T\* rv) {}

MyVector <T> gauss(T\* rv) {…}

private:

void swap(int str1, int str2, T\* rv) {}

void subtractingRows(int mainRow, int sideRow, T\* rv, int index) {}

};

* module.h

Содержит функцию module:

template <typename T>

T module(T a) {

if (a >= 0) {

return a;

}

else {

return -a;

}

}

# Подтверждение корректности

Для подтверждения корректности необходимо подставить вектор решений x системы в правую часть СЛАУ, после чего будет получен ранее введенный вектор b.

# Заключение

В ходе лабораторной работы был реализован метод Гаусса для решения СЛАУ.

Метод Гаусса корректно работает и решает систему линейных уравнений.

# Приложение

MyVector <T> gauss(T\* rv) {

//Print advanced SoHAE

cout << "The initial system: " << endl;

print(rv);

//saving the original matrix

Matrix<T> oM(this->size);

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

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

oM[j][i] = this->m\_x[j][i];

}

}

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

int count = 0;

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

if (this->m\_x[i][j] == 0) {

count++;

}

else {

break;

}

}

//All elements in row 0

if (count == this->size) {

if (rv[i] == 0) {

// 0 = 0

cout << "Can't solve: the equation not depends on the variable" << endl;

exit(1);

}

else if (rv[i] != 0) {

// 0 != 0

cout << "No solution: the system of equations is inconsistent. 0\*x = B, B != 0" << endl;

exit(2);

}

}

}

int index = 0; // Calculates how many of the first coefficients of the side row are zero

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

// Choice main element

T maxValue = 0;

int strMaxValue = 0;

// Search max value

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

if (maxValue <= module(this->m\_x[i + j][j])) {

maxValue = module(this->m\_x[i + j][j]);

strMaxValue = i + j;

}

}

if (maxValue == 0) {

cout << "Division by 0 when searching for solutions" << endl;

exit(3);

}

//print(rv);

swap(strMaxValue, j, rv);

//Subtracts one row multiplied by a number from all the others

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

//Subtracts each row element from the corresponding column element

subtractingRows(j, i + j, rv, index);

}

//print(rv);

index++;

}

cout << "Matrix in triangular form" << endl << endl;

print(rv);

T\* sol = new T[this->size];

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

T tmp = 0;

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

tmp += this->m\_x[i][j] \* sol[j];

}

sol[i] = (rv[i] - tmp) / this->m\_x[i][i];

}

//returns the values of the original matrix

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

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

this->m\_x[j][i] = oM[j][i];

}

}

cout << "The solution of the system: " << endl;

MyVector<T>(this->size, sol).print();

return MyVector<T>(this->size, sol);

}