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**:

студентка группы 381903-2

Волкова Арианна Андреевна

**Проверил**:

ассистент каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2020

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 7](#_Toc26962564)

[Описание программной реализации 9](#_Toc26962565)

[Подтверждение корректности 10](#_Toc26962566)

[Заключение 11](#_Toc26962568)

[Приложение 12](#_Toc26962569)

# Постановка задачи

Реализовать метод Гаусса для действительной квадратной матрицы с выбором ведущего элемента:  
- Необходимо реализовать шаблонный класс вектор  
- Необходимо реализовать класс квадратная матрица, которая является шаблоном класса вектор от вектора  
- Необходимо реализовать класс СЛАУ, который является наследником квадратной матрицы и у него есть метод Гаусса. Метод Гаусса принимает правую часть аргумента и возвращает ответ в виде вектора.

# Метод решения

Реализован шаблонный класс вектор:

template<typename T> class vvector

{

public:

vvector() : vvector(0) { }

vvector(int size) : vvector(new T[size], size) { }

vvector(int size, T obj) : vvector(size)

{

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

arr[i] = obj;

}

vvector(T\* arr, int size)

{

this->arr = arr;

this->size = size;

}

T& operator +(vvector a)

{

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

arr[i] += a[i];

}

T& operator -(vvector a)

{

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

arr[i] -= a[i];

}

T& operator \*(int a)

{

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

arr[i] = arr[i] \* a;

}

T& operator[](int index)

{

return arr[index];

}

int size;

protected:

T\* arr;

};

Реализован класс матрица, который наследован от вектора и определен как «вектор от вектора»

template<typename T> class matrix\_ : public vvector<vvector<T>>

{

public:

matrix\_(int sizeX, int sizeY) : size\_x(sizeX), vvector<vvector<T>>(sizeY)

{

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

this->arr[i] = vvector<T>(sizeX);

}

int sizeX()

{

return this->size\_x;

}

int sizeY()

{

return this->size;

}

private:

const int size\_x;

};

Реализован класс СЛАУ, который наследован от класса матрица и содержит метод Гаусса

class SLAU : public matrix\_<double>

{

public:

SLAU(int sizeX, int sizeY) : matrix\_<double>(sizeX, sizeY) { }

vvector<double> gauss(vvector<double> ex)

{

porydok(ex);

int size\_X = this->sizeX();

matrix\_print();

vvector<double> arr = vvector<double>(size\_X, 0);

vvector<vvector<double>>\* \_mat = (vvector<vvector<double>>\*)this;

for (int i = sizeY() - 1; i >= 0; i--)

{

double ans = ex[i];

double sum = getSum(\_mat->operator[](i), arr);

double val = getValue(\_mat->operator[](i));

arr[i] = round((ans - sum) / val, 5);

}

return arr;

}

double& operator[](element index)

{

vvector<double>& \_y = this->arr[index.y];

double& \_x = \_y.operator[](index.x);

return \_x;

}

void matrix\_print();

private:

void porydok(vvector<double> ex)

{

int i = 1;

int size = this->sizeY();

while (i < size)

{

habr(ex);

int zc\_1 = zero\_in\_start(this->arr[i - 1]);

int zc\_2 = zero\_in\_start(this->arr[i]);

if (zc\_1 == zc\_2) revert(i - 1, i, ex);

else i++;

matrix\_print();

}

habr(ex);

}

static double round(double n, int count)

{

double st = pow(count, 10);

return std::round(n \* st) / st;

}

static int zero\_in\_start(vvector<double>& arr)

{

int length = arr.size;

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

if (arr[i] != 0)

return i + 1;

return length;

}

void habr(vvector<double> ex);

void revert(int arr1\_index, int arr2\_index, vvector<double> ex);

static double getSum(vvector<double> vals, vvector<double> arr)

{

int length = arr.size;

double sum = 0;

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

if (arr[i] != 0)

sum += arr[i] \* vals[i];

return sum;

}

static double getValue(vvector<double> vec)

{

int length = vec.size;

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

if (vec[i] != 0)

return vec[i];

return 0;

}

};

# Руководство пользователя

При запуске программы в консоль требуется ввести размер матрицы (рис.1)



рис. 1

Далее ввод коэффициентов уравнений (рис. 2)

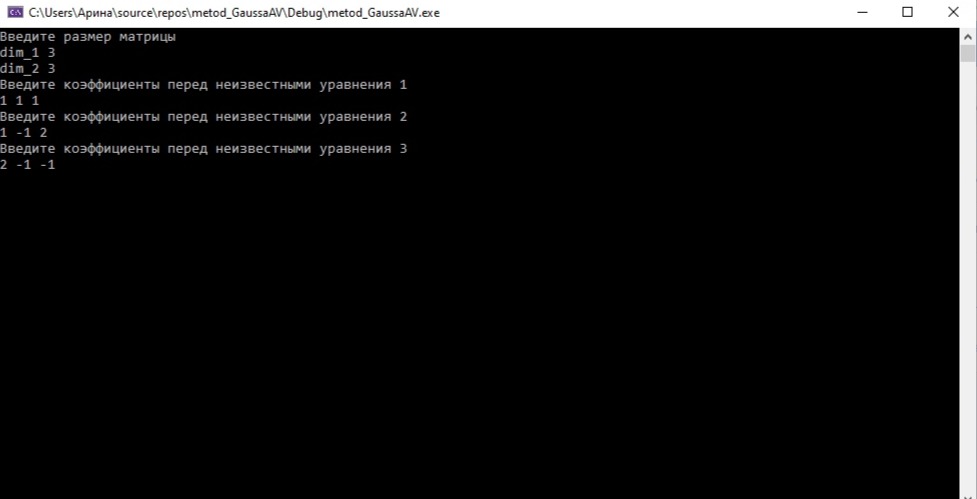


рис. 2

И ввод столбца свободных членов (рис. 3)

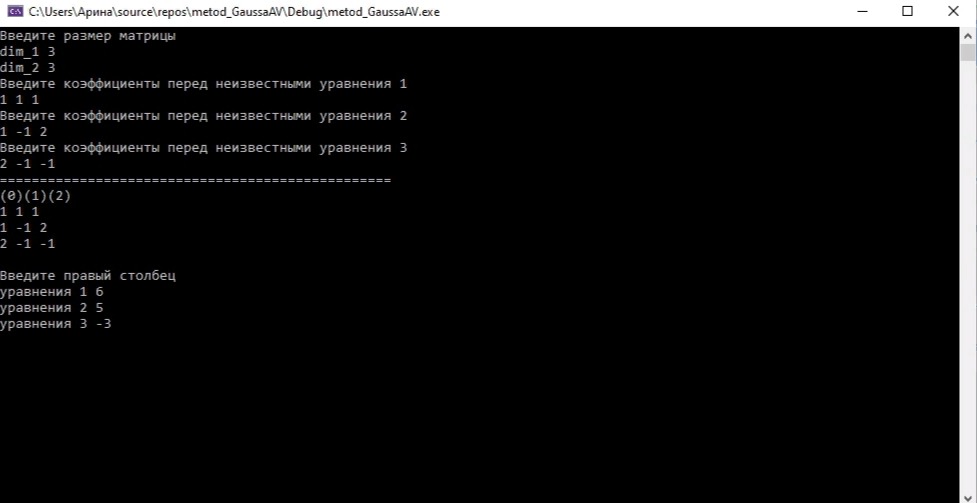


рис. 3

После мы можем отследить поэтапное приведение матрицы к треугольному виду и вывод ответа (рис. 4)

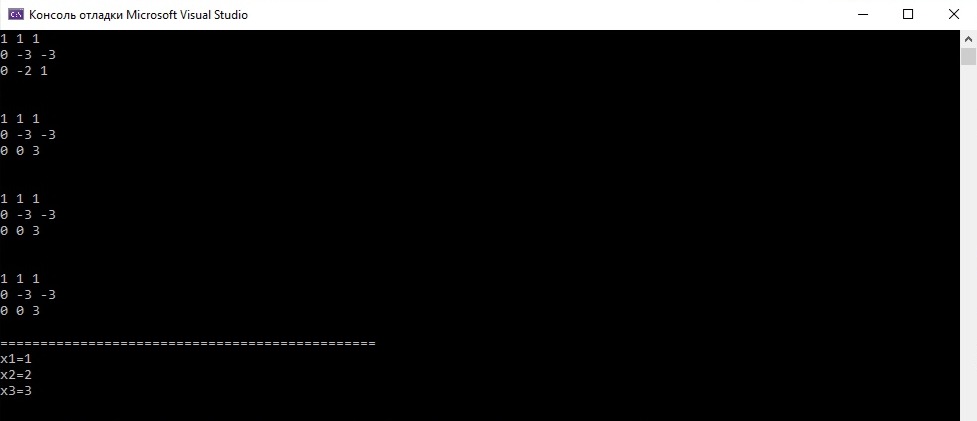


рис. 4

Далее можно заполнить новый столбец свободных членов для той же матрицы или выйти из программы.

# Описание программной реализации

Файлы:

-metod\_GaussaAV.cpp

Содержится подключение заголовочных файлов, инициализация переменных, вызов метода Гаусса.

-vvector.h

Содержится структура, которая возвращает конкретный элемент матрицы; функция swap, которая меняет строки и элементы местами; реализация класса вектор и его методов.

matrix\_.h

Содержится реализация класса матрица

-SLAU.cpp

Содержится функция печати матрицы, функция сортировки.

-SLAU.h

Содержится реализация класса СЛАУ и его методов, в котором прописан метод Гаусса.

# Подтверждение корректности

Для подтверждения корректности можно ввести уже посчитанную матрицу и сравнить с полученным ответом.

# 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

vvector<double> gauss(vvector<double> ex) //метод Гаусса

{

porydok(ex);

int size\_X = this->sizeX();

matrix\_print();

vvector<double> arr = vvector<double>(size\_X, 0);

vvector<vvector<double>>\* \_mat = (vvector<vvector<double>>\*)this;

for (int i = sizeY() - 1; i >= 0; i--)

{

double ans = ex[i];

double sum = getSum(\_mat->operator[](i), arr);

double val = getValue(\_mat->operator[](i));

arr[i] = round((ans - sum) / val, 5);

}

return arr;