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

Филатов Андрей Александрович

**Проверил**:

ассистент каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2020

**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

Написать на языке С++ программу, реализующую метод Гаусса на шаблонной матрице.

# Метод решения

Для решения данной задачи используется метод Гаусса с выбором ведущего элемента и шаблонная матрица. Шаблонная матрица принимает и обрабатывает любые типы данных (По правилам для них установленным).  
Метод Гаусса с выбором ведущего элемента отличается от оригинального метода, тем что из самого левого столбца выбирается наибольший элемент и вектор с ним переставляется так, чтобы он стал ведущим. Затем из оставшихся векторов вычитается ведущий, умноженный на коэффициент.

# Руководство пользователя

Для использования необходимо с помощью include подключить gauss.h к программе, после чего все классы можно использовать с различными типами данных.

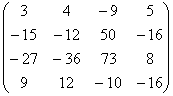
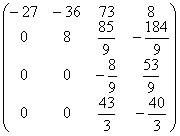
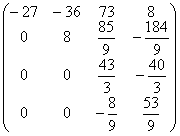
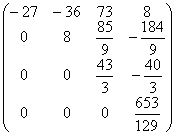
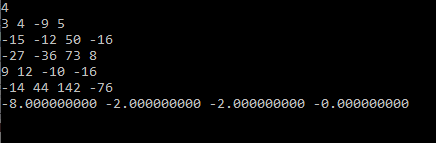
Пример их использования main.cpp.

# Описание программной реализации

Программа состоит из двух файлов. Первый файл gauss.h содержит реализация классов: vector, matrix и gauss.  
Класс vector является шаблонным вектором с функциями ввода, вывода, случайного заполнения сложения с другим объектом класса и домножения на константу.  
Класс matrix наследуется от вектора векторов также имеет ввод, вывод, случайное заполнение.  
Класс gauss имеет единственный метод solve, который исполняет метод гаусса с выбором ведущего элемента.

Файл main.cpp является примером взаимодействия с классами.

# Подтверждение корректности

Корректность алгоритма подтверждается запуском на известном примере, описанном в книге Синютина С.А. “Вычислительная математика”.  
*A*=  , b=  .  
*A=*  *b=*  *.*  
*A=*  *b=*  .  
*A=*  *b=*  .  
Отсюда x4 = 0, x3 = -86/42 = -2, x2 = -2, x1 = -8  
  
Результат работы программы совпадает с ручным подсчётом.

# Результаты экспериментов



Тесты проводились на ноутбуке Aser Intel Core i5-7200U CPU 2.50 GHz. Матрица заполнялась произвольными числами типа double.

# Заключение

Метод Гаусса позволяет решать системы уравнений. Он используется во многих задачах, но его ручной подсчёт для больших матриц слишком сложен, поэтому данный алгоритм играет большую роль в алгебре.

# Приложение

template <typename Type>

class Vector

{

public:

Vector<Type>()

{

size = 0;

T = nullptr;

}

Vector<Type>(size\_t size)

{

this->size = size;

T = new Type[size];

}

Vector<Type>(Vector& B) : Vector<Type>(B.size)

{

size = B.size;

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

{

T[i] = B[i];

}

}

size\_t get\_size()

{

return size;

}

void Random()

{

srand(clock());

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

{

T[i] = (double)rand() / (double)rand();

}

}

Type& operator[](size\_t i)

{

if (i > size)

{

cerr << "Index out of range";

exit(0);

}

return T[i];

}

Vector<Type>& operator=(Vector<Type> B)

{

swap(T, B.T);

swap(size, B.size);

return \*this;

}

Vector<Type> operator-(Vector<Type>& B)

{

if (size != B.size)

{

cerr << "Wrong size";

exit(0);

}

Vector C(size);

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

{

C[i] = T[i] - B[i];

}

return C;

}

Vector<Type> operator\*(Type b)

{

Vector C(size);

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

{

C[i] = T[i] \* b;

}

return C;

}

template <typename Type> friend istream& operator>>(istream& in, Vector<Type>& A);

template <typename Type> friend ostream& operator<<(ostream& out, Vector<Type>& A);

~Vector()

{

delete[](T);

size = 0;

}

protected:

size\_t size;

Type\* T;

};

template <typename Type>

istream& operator>>(istream& in, Vector<Type>& A)

{

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

{

in >> A[i];

}

return in;

};

template <typename Type>

ostream& operator<<(ostream& out, Vector<Type>& A)

{

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

{

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

}

return out;

};

template<typename Type>

class Matrix : public Vector<Vector<Type>>

{

public:

Matrix<Type>()

{

}

Matrix<Type>(size\_t size) : Vector<Vector<Type>>(size)

{

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

{

T[i] = Vector<Type>(size);

}

}

Matrix<Type>(Matrix& B) : Matrix(B.size)

{

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

{

T[i] = B[i];

}

}

void Random()

{

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

{

T[i].Random();

}

}

Matrix<Type>& operator=(Matrix B)

{

swap(size, B.size);

swap(T, B.T);

return \*this;

}

template <typename Type> friend istream& operator>>(istream& in, Matrix<Type>& A);

template <typename Type> friend ostream& operator<<(ostream& out, Matrix<Type>& A);

~Matrix()

{

}

};

template <typename Type>

istream& operator>>(istream& in, Matrix<Type>& A)

{

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

{

in >> A[i];

}

return in;

};

template <typename Type>

ostream& operator<<(ostream& out, Matrix<Type>& A)

{

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

{

out << A[i] << endl;

}

return out;

};

template <typename Type>

class Gauss

{

public:

Gauss<Type>(Matrix<Type>& M)

{

this->M = M;

}

Vector<Type> Solve(Vector<Type>& V)

{

size\_t size = V.get\_size();

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

{

size\_t max = i;

for (size\_t j = i + 1; j < size; j++)

{

if (M[j][i] - M[max][i] > e)

{

max = j;

}

}

if (max != i)

{

swap(M[i], M[max]);

}

if (abs(M[max][i]) < e)

{

cerr << "Invalid matrix";

exit(0);

}

for (size\_t j = i + 1; j < size; j++)

{

Type temp = M[j][i] / M[i][i];

M[j] = M[j] - (M[i] \* temp);

V[j] -= V[i] \* temp;

}

}

Vector<Type> G(size);

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

{

G[i] = V[i];

for (size\_t j = i + 1; j < size; j++)

{

G[i] -= G[j] \* M[i][j];

}

G[i] /= M[i][i];

}

return G;

}

private:

Matrix<Type> M;

};