БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №1

**Метод Гаусса**

**Выполнил:**

Крючков Василий

2 курс 9 группа

**Преподаватель:**

Горбачева Ю.Н.

Минск, 2021

**Постановка задачи**

Написать и отладить программу, реализующую метод Гаусса с выбором главного элемента по столбцу для численного решения систем линейных алгебраических уравнений , вычисления обратной матрицы и её проверки, посчитать относительную погрешность. Предусмотреть сообщения, предупреждающие о невозможности решения указанных задач с заданной матрицей A порядка n.

**Краткие теоретические сведения**

Алгоритм Метода Гаусса с выбором главного элемента по столбцу:

На каждом k-м шаге находим наибольший по модулю элемент aj,k в k-м столбце,причем индекс j больше или равен индексу диагонального элемента в этом же столбце и если он будет не равен, то то меняем j-ую строчку со строкой в которой находится диагональный элемент принадлежавший k-ому столбцу. А далее, по алгоритму метода Гаусса проводим исключение переменной xk из матрицы, образованной после одной из перечисленных выше перестановок. И обратным ходом находим вектор решений.

Прямой ход метода Гаусса:

где m — номер уравнения, из которого исключается xk; k — номер неизвестного, которое исключается из оставшихся (n – k) уравнений, а также обозначает номер уравнения, с помощью которого исключается xk; i — номер столбца исходной матрицы; — главный (ведущий) элемент матрицы.

Обратный ход метода Гаусса:

Для нахождение обратной матрицы с помощь метода Гаусса нам необходимо решить ряд линейный систем , где – i-ый столбец единичной матрицы. Матрица [], где – столбец решения , и будет является обратной A матрицей.

**Листинг программы**

**Lab1.java**

import java.math.\*;

import java.util.Arrays;

import java.util.Formatter;

public class Lab1 {

public static double[] gaussSelectionByColumn(int matrixOrder,double [][] matrixA,double [] columnF){

//Прямой ход

double [] columnX = new double[matrixOrder];

int maxIndex;

double maxInColumn;

double temp;

for (int i=0; i<matrixOrder-1; i++) {

maxIndex=i;

maxInColumn=Math.abs(matrixA[i][i]);

for (int j=i; j<matrixOrder; j++){

if(maxInColumn<Math.abs(matrixA[j][i]))

{

maxInColumn=Math.abs(matrixA[j][i]);

maxIndex=j;

}

if(maxInColumn==0)

throw new ArithmeticException("Нулевой столбец - решение данной слау невозможно");

}

for (int k=i; k<matrixOrder; k++){

temp=matrixA[i][k];

matrixA[i][k]=matrixA[maxIndex][k];

matrixA[maxIndex][k]=temp;

}

temp=columnF[i];

columnF[i]=columnF[maxIndex];

columnF[maxIndex]=temp;

for (int j=i+1; j<matrixOrder; j++)

matrixA[i][j]= matrixA[i][j]/ matrixA[i][i];

columnF[i]= columnF[i]/matrixA[i][i];

matrixA[i][i]=1.0;

for (int j=i+1; j<matrixOrder; j++)

{

for (int k=i+1; k<matrixOrder; k++)

matrixA[j][k]= matrixA[j][k]+matrixA[i][k]\*matrixA[j][i]\*(-1.0);

columnF[j]= columnF[j]+ columnF[i]\*matrixA[j][i]\*(-1.0);

matrixA[j][i]=0;

}

}

columnF[matrixOrder-1]=columnF[matrixOrder-1]/matrixA[matrixOrder-1][matrixOrder-1];

matrixA[matrixOrder-1][matrixOrder-1]=1;

//Обратный ход

columnX[matrixOrder-1]=columnF[matrixOrder-1];

for (int i=matrixOrder-2; i>-1; i--) {

columnX[i]=columnF[i];

for (int k=i; k<matrixOrder-1; k++)

columnX[i]=columnX[i]+matrixA[i][k+1]\*columnX[k+1]\*(-1.0);

}

return columnX;

}

public static void main(String[] args) {

int matrixOrder = 10;

double [][] matrixA = new double[matrixOrder] [matrixOrder];

double [] columnX = new double[matrixOrder];

double [] exactСolumnX = new double[matrixOrder];

double [] columnF = new double[matrixOrder];

//Точное решение слау

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

exactСolumnX[i]=i+1;

//Заполняем матрицу A случайными числами из диапазона от -100 до 100

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

for (int j=0; j<matrixOrder; j++)

matrixA[i][j] = 100-(Math.random()\*200);

//Вектор столбец f

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

for(int j = 0; j < matrixOrder; j++)

columnF[i] += exactСolumnX[j] \* matrixA[i][j];

//Метод

try {

columnX=gaussSelectionByColumn(matrixOrder,Arrays.stream(matrixA).map(double[]::clone).toArray(double[][]::new),columnF);

}catch(ArithmeticException e){

System.out.println(e.getMessage());

System.exit(0);

}

//Погрешность

double normXX=Math.abs(exactСolumnX[0]-columnX[0]);

double normX = Math.abs(exactСolumnX[0]);

for (int i=1; i<matrixOrder; i++) {

if(Math.abs(exactСolumnX[i]-columnX[i])>normXX)

normXX=Math.abs(exactСolumnX[i]-columnX[i]);

if(Math.abs(exactСolumnX[i])>normX)

normX=Math.abs(exactСolumnX[i]);

}

double accuracy=normXX/normX\*100;

//Нахождение обратной

double [][] inverseMatrixA = new double[matrixOrder][matrixOrder];

double [][] unitMatrix = new double[matrixOrder][matrixOrder];

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

for (int j=0; j<matrixOrder; j++) {

if(i==j)

unitMatrix[i][j]=1;

else

unitMatrix[i][j]=0;

}

double [] tempColumn = new double[matrixOrder];

for (int i=0; i<matrixOrder; i++) {

tempColumn=unitMatrix[i];

try {

tempColumn=gaussSelectionByColumn(matrixOrder,Arrays.stream(matrixA).map(double[]::clone).toArray(double[][]::new),tempColumn);

}catch(ArithmeticException e){

System.out.println(e.getMessage());

System.exit(0);

}

for (int j=0; j<matrixOrder; j++)

inverseMatrixA[j][i]=tempColumn[j];

}

//Перемножение матрицы и обратной

double[][] mulMatr=new double[matrixOrder][matrixOrder];

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

for (int j=0; j<matrixOrder; j++)

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

mulMatr[i][j] += matrixA[i][k] \* inverseMatrixA[k][j];

//Ввывод результатов

System.out.println("Матрица А и столбец f");

for (int i=0; i<matrixOrder; i++) {

Formatter f = new Formatter();

for (int j=0; j<matrixOrder; j++) {

f.format("%8.4f ", matrixA[i][j]);

}

f.format("| %8.4f%n", columnF[i]);

System.out.print(f);

}

System.out.println();

System.out.println("Вектор точного решения Вектор приближенного решения");

Formatter f = new Formatter();

for (int j=0; j<matrixOrder; j++)

f.format("%19.1f %40.16f%n", exactСolumnX[j], columnX[j]);

System.out.print(f);

System.out.print("Относительная погрешность =");

f = new Formatter();

f.format("%19.16f%%%n", accuracy);

System.out.print(f);

System.out.println("Обратная матрица А");

for (int i=0; i<matrixOrder; i++) {

f = new Formatter();

for (int j=0; j<matrixOrder; j++) {

f.format("%8.4f ", inverseMatrixA[i][j]);

}

System.out.println(f);

}

System.out.println("Произведение А\*А^(-1)");

for (int i=0; i<matrixOrder; i++) {

f = new Formatter();

for (int j=0; j<matrixOrder; j++) {

f.format("%12.4e ", mulMatr[i][j]);

}

System.out.println(f);

}

//Оценка точности решения системы линейных алгебраических уравнений

System.out.println("Порядок матрицы Относительная погрешность ");

f = new Formatter();

final int k=6+1;

for (int i=k; i<=101+k; i+=10) {

matrixOrder = i;

matrixA = new double[matrixOrder] [matrixOrder];

columnX = new double[matrixOrder];

exactСolumnX = new double[matrixOrder];

columnF = new double[matrixOrder];

for (int i1=0; i1<matrixOrder; i1++)

exactСolumnX[i1]=i1+1;

for (int i1=0; i1<matrixOrder; i1++)

for (int j=0; j<matrixOrder; j++)

matrixA[i1][j] = 100-(Math.random()\*200);

for(int i1 = 0; i1 < matrixOrder; i1++)

for(int j = 0; j < matrixOrder; j++)

columnF[i1] += exactСolumnX[j] \* matrixA[i1][j];

try {

columnX=gaussSelectionByColumn(matrixOrder,Arrays.stream(matrixA).map(double[]::clone).toArray(double[][]::new),columnF);

}catch(ArithmeticException e){

System.out.println(e.getMessage());

System.exit(0);

}

//Погрешность

normXX=Math.abs(exactСolumnX[0]-columnX[0]);

normX = Math.abs(exactСolumnX[0]);

for (int i1=1; i1<matrixOrder; i1++) {

if(Math.abs(exactСolumnX[i1]-columnX[i1])>normXX)

normXX=Math.abs(exactСolumnX[i1]-columnX[i1]);

if(Math.abs(exactСolumnX[i1])>normX)

normX=Math.abs(exactСolumnX[i1]);

}

accuracy=normXX/normX\*100;

f.format("%5d%36.14f%%%n",i, accuracy);

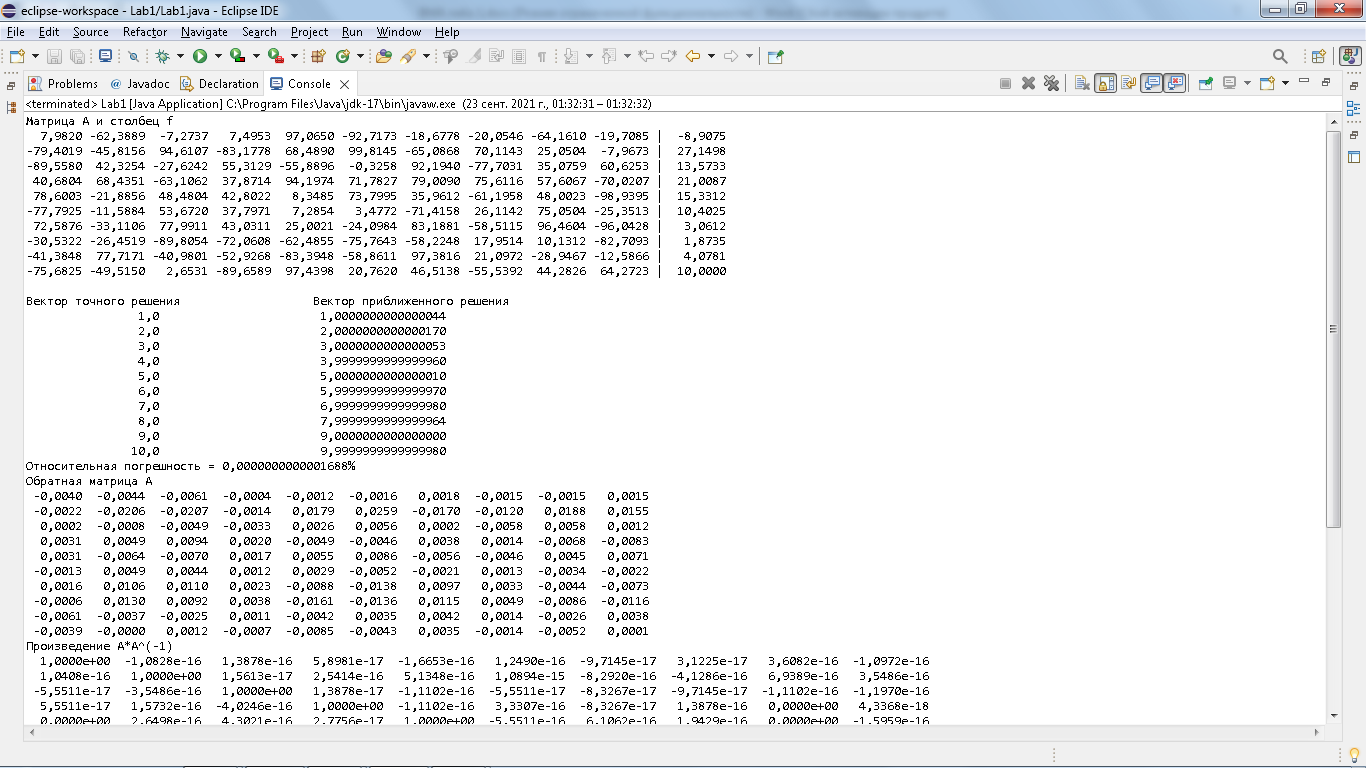
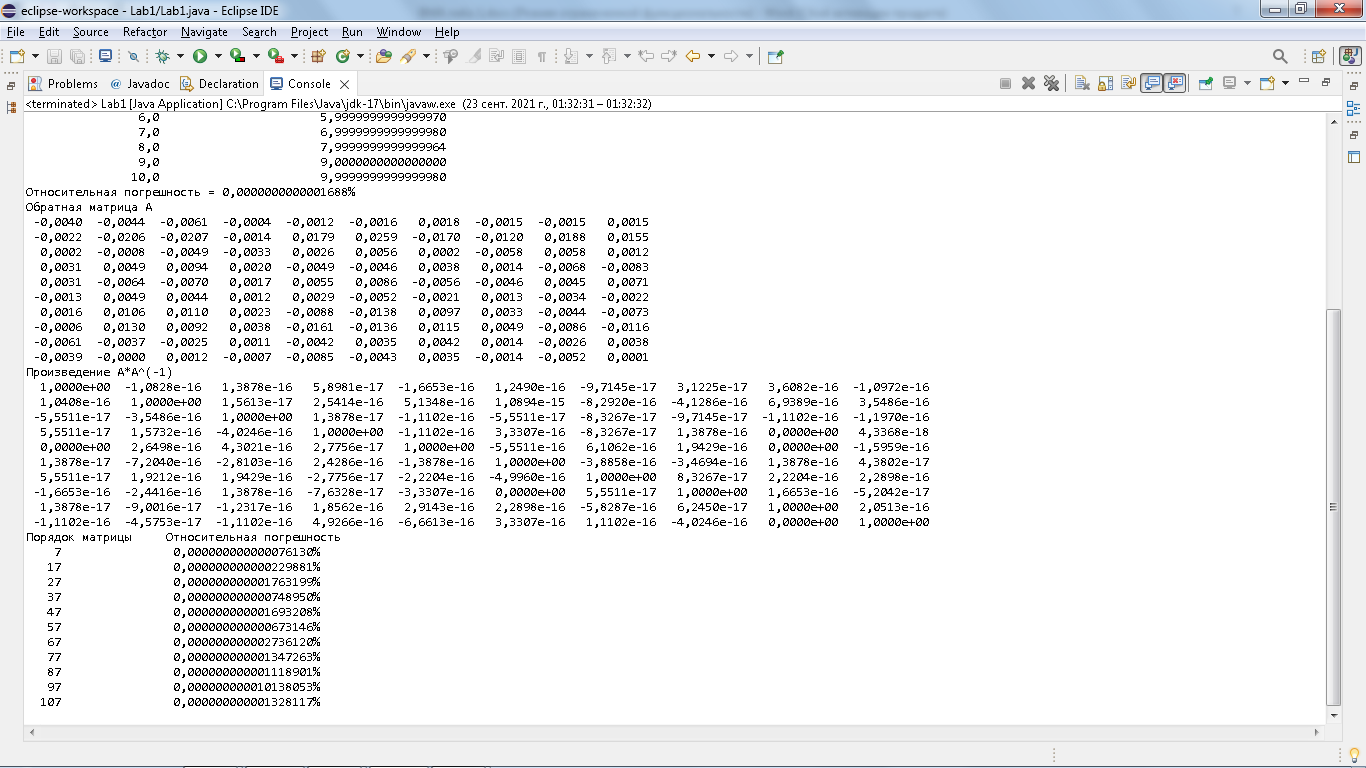
}

System.out.print(f);

}

}

**Результаты**

|  |  |
| --- | --- |
| Порядок  матрицы | Относительная погрешность вида |
| 7 | 0,000000000000076130% |
| 17 | 0,000000000000229881% |
| 27 | 0,000000000000748950% |
| 37 | 0,000000000000748950% |
| 47 | 0,000000000001693208% |
| 57 | 0,000000000000673146% |
| 67 | 0,000000000002736120% |
| 77 | 0,000000000001347263% |
| 87 | 0,000000000001118901% |
| 97 | 0,000000000010138053% |
| 107 | 0,000000000001328117% |

**Выводы**

Метод Гаусса с выбором главного элемента по столбцу позволяет при вычислениях часто избегать случаи деления на очень малые величины, обеспечивает низкую погрешность вычислений и, как следствие, относительную погрешность метода. При увеличение порядка матрицы A наблюдается рост относительной погрешности, что было предсказуемо, так как при увеличении порядка матрицы увеличивается и количество арифметических операций производимыми над коэффициентами.