БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №1

Решение систем линейных алгебраических уравнений, нахождение определителя и обратной матрицы методом Гаусса

Выполнил:

Студент 2 курса 5 группы ФПМИ

Дунаев Виктор

Руководитель:

Радкевич Елена Владимировна

Минск, 2016 г.

**Оглавление**

Постановка задачи ………………………………………………………………………………………......…..3 Описание метода нахождения решений

системы алгебраических уравнений методом Гаусса…………………………………………..3

Нахождение определителя матрицы ……………………………………………………………………5

Нахождение обратной матрицы …………………………………………………………………………..5

Нахождение вектора невязки………………………………………………………………………………..5

Листинг программы ……………………………………………………………………………………………….5

Входные данные ………………………………………………………………………………………….……….13

Выходные данные …………………………………………………………………………………………….....13

**1. Постановка задачи**

Решить систему линейных алгебраических уравнений методом Гаусса с выбором максимального(главного) элемента по строке. Вычислить определитель матрицы. Найти ее обратную матрицу и провести проверку. Найти вектор невязки для оценки погрешности вычислений.

**2.Описание метода нахождения решений системы**

**алгебраических уравнений методом Гаусса с выбором главного элемента по строке**

Рассмотрим систему вида

При неизвестных xj, 1 ≤j≤n есть хотя бы один коэффициент, отличный от нуля. Без ограничения общности будем считать, что это а11. (Если это не так, рассматриваем уравнение, в котором коэффициенты при неизвестных xj не равны нулю и меняем его местами с 1ым уравнением.)

И пусть a11, не теряя общности, максимальный по модулю элемент среди всех элементов матрицы системы, иначе если aij максимальный по модулю элемент, то меняем i-ую строку с первой строкой местами, j-ый столбец с первым столбцом местами и отмечаем количество совершенных перестановок столбцов и строк.

Исключим неизвестную x1из всех уравнений системы, начиная со второго и ниже. Дляэтого ко второму уравнению прибавим первое уравнение, умноженное на (), к третьему прибавим первое, умноженное на (), и

т.д. к n-ому прибавим первое, умноженное на ().

Таким образом, мы перешли от системы (1) к равносильной ей системе (2):

В результате этого шага могут возникнуть следующие ситуации:

1. Среди уравнений системы (2) есть уравнение, у которого все коэффициенты при неизвестных равны нулю, а свободный член отличен от нуля. В этом случае система (2) несовместна, а, следовательно, несовместна система (1). Процесс решения закончен.
2. В системе (2) есть уравнение, у которого все коэффициенты при неизвестных и свободный член равны нулю. Это уравнение удаляем из системы.
3. Среди коэффициентов аij, есть хотя бы один, отличный от нуля.

Тогда без ограничения общности считаем, что а22 отличный от нуля и максимальный по модулю элемент, и исключаем x2 из всех уравнений, начиная с третьего, и т.д.

И так далее исключая последовательно неизвестные, через к шагов мы либо докажем, что система несовместна, либо получим систему:

Общие формулы для нахождения ,:

где k=;

i, j=.

После приведения системы к виду (3) можно приступать к обратному ходу метода гаусса, а именно к последовательному нахождению неизвестных переменных при движении от последнего уравнения к первому.

Для нахождения неизвестного xк воспользуемся формулой:

**3.Нахождение определителя матрицы**

Исходя из свойств определителя имеем формулу:

=

гдеалгебраическое дополнение к элементу .

Если, используя метод Гаусса, привести матрицу к верхне-треугольному виду, то = 0, если ij.

Тогда определитель матрицы можно найти по формуле:

**4.Нахождение обратной матрицы**

Для нахождения обратной матрицы достаточно решитьметодомГаусса n-линейных уравнений, у каждого из которых в столбце свободных членов на k-ом месте стоит 1, а на остальных - нули, т е. решить систему с расширенной матрицей вида:

AX = E => X = A-1

Для проверки правильности нахождения обратной матрицы воспользуемся формулой:

A\* A-1 = E.

**5.Нахождение вектора невязки**

Решаем систему AX = B, находим приближенное решение Х.

Вектор невязки – это вектор R = AX – B.

Чем ближе к нулю будут координаты вектора, тем точнее наше решение.

**6. Листинг программы**

**Lab1VMA.java**

**Основной класс программы**

package lab1vma;

import java.io.\*;

import java.text.NumberFormat;

public class Lab1VMA {

public static void main(String[] args) {

Input I = new Input("InputFile.txt");

FindBestElements F = new FindBestElements();

OneSplittingScheme S = new OneSplittingScheme();

FindResults C = new FindResults(); Output O = new Output();

int stop =5;double det;double []nevVekt; double []X ; double [][]EMatr;

int n;int m;double [][]matrix;int[] A; double [][]copyMatrix;double [][]resultMatrix;

NumberFormat formatter = NumberFormat.getNumberInstance();

formatter.setMaximumFractionDigits(stop);

try

{ n=I.sizeRead();

m=I.sizeRead(); }

catch (IOException e)

{

System.out.println("Ошибка чтения!");

n=1;m=1;

}

matrix=I.createMatrix(n, m);

copyMatrix = I.copyMatrix(matrix, n, m);

A = new int[n];

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

{

A[i]=i;

}

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

{

matrix=F.findBest(matrix, i, A, n);

matrix = S.split(matrix, i, n);

}

X = S.findVectorX(matrix, n);

System.out.println("Вектор X: ");

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

{

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

{

if(A[j]==i)

{

System.out.print(formatter.format(X[A[j]])+" ");

}}

}

System.out.println();

System.out.println("Вектор невязки: ");

nevVekt = S.check(copyMatrix, n, X, A);

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

{

System.out.print(nevVekt[i]+" ");

}

System.out.println();

resultMatrix = C.findInverseMatrix(copyMatrix, n);

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

O.outMatrix(resultMatrix, n, n,10);

det = Math.pow(-1, F.kol\_vo);

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

{

det = det\*X[i];

}

System.out.println("Определитель = "+ det);

EMatr = C.checkInverseMatrix(copyMatrix, resultMatrix, n);

System.out.println("Проверка обратной матрицы: A \* A^(-1) = E");

O.outMatrix(EMatr, n, n,20); } }

**Input.java**

**Класс для получения нужной информации из исходного файла**

package lab1vma;

import java.io.\*;

import java.util.StringTokenizer;

public class Input {

String a;

int b;

FileReader fr;

BufferedReader br;

String nameFile;

**//Функция инициализации файла**

Input(String name){

try

{

fr = new FileReader(name);

br = new BufferedReader(fr);

nameFile=name;

}

catch (IOException e)

{

System.out.println("Ошибка чтения!");

}

}

**//Функция чтение строки**

public String stringRead() throws IOException{

a=new String();

try

{

a=br.readLine();

}

catch (IOException e)

{

System.out.println("Ошибка чтения с клавиатуры!");

}

return a;

}

**//Функция чтения размерности**

public int sizeRead() throws IOException{

b=0;

try

{

String line = br.readLine();

b = Integer.parseInt(line);

}

catch (NumberFormatException e)

{

System.out.println("Не целое число");

}

catch (IOException e)

{

System.out.println("Ошибка чтения с клавиатуры");

}

return b;

}

**//Функция создания матрицы**

public double[][] createMatrix ( int n, int m){

double [][]M;

M=new double[n][m];

StringTokenizer st;

try{

st = new StringTokenizer(stringRead()," ,;");

}

catch (IOException e)

{

System.out.println("Ошибка чтения!");

st=new StringTokenizer(""," ,;");

}

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

{

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

{

if(st.hasMoreTokens())

{

M[i][j]=Double.parseDouble(st.nextToken());

}

else

{

M[i][j]=0;

}

}

if(i<n-1)

{

try{

st = new StringTokenizer(stringRead()," ,;");

}

catch (IOException e)

{

System.out.println("Ошибка чтения!");

st=new StringTokenizer(""," ,.;");

}

}

}

return M;

}

**//Функция копирования матрицы**

public double[][] copyMatrix (double[][] M, int n, int m){

double[][] N = new double[n][m];

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

{

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

{

N[i][j]=M[i][j];

}

}

return N;

}

}

**FindBestElements.java**

**Класс для нахождения максимального(главного) элемента**

package lab1vma;

public class FindBestElements {

int kol\_vo;

**//Констурктор класса**

FindBestElements(){

kol\_vo=0;

}

**//Функция поиска главного элемента по строке**

public double[][] findBest(double[][]M,int str,int[] A,int n){

double best=M[str][str]; int num=str;

for(int i=str;i<n;i++)

{

if(Math.abs(best) < Math.abs(M[str][i]))

{

best=M[str][i]; num = i;

}

}

double temp;

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

{

temp = M[i][str];

M[i][str] = M[i][num];

M[i][num] = temp;

}

int temp2;

if(A[str]!=A[num])

{

kol\_vo++;

}

temp2=A[str];

A[str]=A[num];

A[num]=temp2;

return M;

}}

**OneSplittingScheme.java**

**Класс для нахождения вектора решения и вектора невязки**

package lab1vma;

public class OneSplittingScheme {

**//Схема единственного деления**

public double[][] split(double[][]M, int str,int n){

double splitter = M[str][str];

for(int j=str;j<n+1;j++)

{

M[str][j]=M[str][j]/splitter;

}

for(int i=str+1;i<n;i++)

{

double temp = M[i][str];

for(int j=str;j<n+1;j++)

{

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

} }

return M;

}

**//Функция поиска вектора решения**

public double[] findVectorX (double[][] M,int n){

double[] X = new double[n];

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

{

X[i]=M[i][n];

for(int j=n-1;j>i;j--)

{

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

}

}

return X;

}

**//Функция поиска вектора невязки**

public double[] check (double[][] M, int n,double[] X,int []A)

{

double [][] res = new double[n][n];double []vektorNev = new double [n];

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

{

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

{

res[i][A[j]]=X[j]\*M[i][A[j]];

}

}

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

{

vektorNev[i]=M[i][n];

}

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

{

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

{

vektorNev[i]=vektorNev[i]-res[i][j];

}

}

return vektorNev;

}

}

**VectorWork.java**

**Класс для работы с векторами**

package lab1vma;

public class VectorWork {

int n;

double[] M;

**//Конструктор класса**

VectorWork(double[] matr, int size){

n=size;

M = new double[n];

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

{

M[i]=matr[i];

}

}

**//Функция построения(поиска) вектора**

public VectorWork findVector (double[][]matrRes,double[][]matr,

int string, int size, VectorWork[] vc){

VectorWork res = new VectorWork(matrRes[string],size);

for(int j=n-1;j>string;j--)

{

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

{

res.M[i]=res.M[i]-vc[j].M[i]\*matr[string][j];

}

}

return res;

}

**//Функция печати векторов**

public void printVector(){

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

{

System.out.print(M[i]+" ");

}

System.out.println();

}

**//Функция построения матрицы по векторам**

public double[][] createMatrixFromVectors(VectorWork[] vc, int size){

double [][]m = new double[size][size];

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

{

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

{

m[i][j]=vc[i].M[j];

}

}

return m;

}}

**FindResults.java**

**Класс для поиска обратной матрицы**

package lab1vma;

public class FindResults {

**//Функция поиска обратной матрицы**

public double[][] findInverseMatrix(double [][]M, int n){

double [][]workMatr = new double[n][n];

double [][]resultMatr = new double[n][n];

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

{

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

{

workMatr[i][j]=M[i][j];

resultMatr[i][j]=0;

if(i==j)

{

resultMatr[i][j]=1;

} }

}

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

{

double delitel = workMatr[i][i];

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

{

workMatr[i][j]=workMatr[i][j]/delitel;

resultMatr[i][j]/=delitel;

}

for(int g=i+1;g<n;g++)

{

double mnogitel = workMatr[g][i];

for(int f=0;f<n;f++)

{

workMatr[g][f]=workMatr[g][f]-workMatr[i][f]\*mnogitel;

resultMatr[g][f]=resultMatr[g][f]-resultMatr[i][f]\*mnogitel;

}

}

}

VectorWork[] vc = new VectorWork[n];

VectorWork f = new VectorWork(resultMatr[0],n);

for(int str = n-1;str>=0;str--)

{

vc[str]=f.findVector(resultMatr, workMatr, str, n, vc);

}

resultMatr=f.createMatrixFromVectors(vc, n);

return resultMatr;

}

**//Функция проверки обратной матрицы по формуле A \* A^(-1) = E**

public double[][] checkInverseMatrix(double[][]A,double[][]B,int n){

double[][]res = new double[n][n];

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

{

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

{

res[i][j]=0;

}

}

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

{

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

{

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

{

res[i][j]+=A[i][k]\*B[k][j];

}

} }

return res;

}}

**Output.java**

**Класс для вывода результатов**

package lab1vma;

import java.text.NumberFormat;

public class Output {

int stop;

NumberFormat formatter = NumberFormat.getNumberInstance();

**//Функция вывода матрицы**

public void outMatrix(double [][]matrix,int n,int m,int size){

stop=size;

formatter.setMaximumFractionDigits(stop);

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

{

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

{

System.out.print(formatter.format(matrix[i][j])+ "\t");

}

System.out.println();

} }}

**7.Входные данные**

**8.Выходные данные**

