БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №2

Решение систем линейных алгебраических уравнений методом квадратного корня

Выполнил:

Студент 2 курса 5 группы ФПМИ

Дунаев Виктор

Руководитель:

Радкевич Елена Владимировна

Минск, 2016 г.

**Оглавление**

Постановка задачи …………………………………………………………………….……………………......…..3

Метод квадратного корня……………………………………………………………………………….………..3

Применение к решению заданной системы ………………………………………..………………5

Вычисление определителя ……………………………………………………………………………..……..5

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

Входные данные ……………………………………………………………………………………….…….……….11

Выходные данные ……………………………………………………………….……………………………….....11

1. **Постановка задачи**

Дана СЛАУ, где расширенная матрица этой системы имеет следующий вид:

**А b**

0.4997 -0.0658 0.0132 0.0263 0.0921 -2.8141

0.0684 0.7824 0.0000 -0.0526 0.0526 2.4104

0.0395 0.0000 0.6286 -0.1841 0.1052 2.2828

-0.0789 0.1657 0.0000 0.6181 -0.0263 -1.6332

0.3288 0.0000 0.1184 0.0132 0.7364 1.8936

1. Найти решение системы при помощи метода квадратного корня
2. Вычислить определитель матрицы при помощи метода квадратного корня
3. Вывести на печать:

А) Исходную матрицу;

Б) Матрицу A(t)\*A;

В) Вектор A(t)\*B;

Г) Матрицу S;

Д) Вектора X и Y;

Е) Определитель.

**2. Описание метода нахождения решений системы**

**алгебраических уравнений методом квадратного корня**

Метод квадратного корня:



Метод квадратного корня можно применить в случае специального вида матрицы системы, а именно , если , то . Если  является эрмитовой (симметрической), то существуют такие верхняя треугольная матрица  с вещественными положительными элементами на главной диагонали и диагональная матрица  с элементами  на диагонали, что справедливо разложение ,

, , , 

Если  (матрица  положительно определена), то  .

Приравнивая элементы матриц  и , получим систему уравнений для определения ,  вида

, , .

Полагая , получим

, ;

;

При  получим

, , .

Для нахождения всех ненулевых элементов матриц  и  эти соотношения применяются рекуррентно.

. ; ; , .



Если  и , то формулы принимают вид

; , ;

; ;

; (1)

Если разложение вида  получено, то решение исходной системы сводится к решению двух систем с треугольными матрицами

; если  и , то .

Алгоритм решения последней системы: прямой ход состоит в последовательном нахождении  по соотношениям (1) и  по следующим рекуррентным формулам

, , .

Обратный ход состоит в вычислении  по следующим формулам

, , .

**3. Применение к решению заданной системы.**

Матрица  не является симметрической, поэтому домножим исходную систему  слева на  и обозначим , . Получим систему вида , где , . Так как , то . Решая эту систему, получим решение , которое также будет решением исходной системы.

**4. Вычисление определителя.**

.

**5. Листинг программы**

**Lab2\_Koren.java**

**Основной класс программы**

public class Lab2\_Koren {

public static void main(String[] args) {

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

Output out = new Output();

Matrix ma = new Matrix();

Umnozit mul = new Umnozit();

Vector v = new Vector();

int n = IF.intTake();

int m = IF.intTake();

double[][] afull = IF.create2XMatrix(n, m);

double[] b = new double[n];

double[][] a = new double[n][n];

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

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

a[i][j]=afull[i][j];}

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

b[i]=afull[i][m-1];}

System.out.println("Source matrix A: ");

out.Output(a, n, n);

System.out.println("Source vector B: ");

out.OutputVector(b,n);

b = mul.doItWithVect(ma.transpored(a), b);

a = mul.doIt(ma.transpored(a), a);

System.out.println("Matrix A(t)\*A: ");

out.Output(a, n, n);

System.out.println("Vector A(t)\*B: ");

out.OutputVector(b,n);

double[] d = null;

double[][]S = ma.findS(a);

double[] Y = ma.findY(S, b);

double[]X = ma.findX(S, Y);

System.out.println("Matrix S: ");

out.Output(S, n, n);

System.out.println("Vector Y: ");

out.OutputVector(Y,n);

System.out.println("Vector X: ");

out.OutputVector(X,n);

System.out.println();

double det = ma.findDet(S);

System.out.println("Determinant: "+ det);

}}

**InputFile.java**

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

public class InputFile {

String a;

int b;

FileReader fr;

BufferedReader br;

String nameFile;

**//Конструктор класса**

InputFile(String name)

{

try

{

fr = new FileReader(name);

br = new BufferedReader(fr);

nameFile=name;

}

catch (IOException e)

{System.out.println("Ошибка чтения");}

}

**//Чтение строки**

public String stringTake() throws IOException

{

a=new String();

try

{

a=br.readLine();

}

catch (IOException e)

{

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

}

return a;

}

**//Чтение числа**

public int intTake()// 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;}

**//Создание матрицы**

public double[][] create2XMatrix ( int n, int m)

{

double [][]M;

M=new double[n][m];

StringTokenizer st;

try{

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

}

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]=Float.parseFloat(st.nextToken());

}

else

{

M[i][j]=0;

}

}

if(i<n-1)

{

try{

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

}

catch (IOException e)

{

System.out.println("Ошибка чтения");

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

}

}

}

return M;

}}

**Matrix.java**

**Класс для работы с исходной матрицей**

public class Matrix {

double[] d;

**//Транспонирование матрицы**

double[][] transpored(double[][]a){

double[][] ta = new double[a.length][a.length];

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

for(int j=0;j<a.length;j++)

ta[i][j]=a[j][i];

return ta;

}

**//Вычисление sign**

double sign(double element)

{

if(element<0)

{

return -1;

}

return 1; }

**//Вычисление матрицы S**

double[][] findS(double[][]A){

int n = A.length;

d = new double[n];

double[][] S=new double[n][n];

d[0]=sign(A[0][0]);

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

{

double sum = A[i][i];

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

sum -= S[k][i] \* S[k][i] \* d[k];

//System.out.println("sum = "+sum);

}

d[i]=sign(A[i][i]-Math.abs(sum));

//System.out.println("d"+(i+1) + " = "+d[i]);

S[i][i] = Math.sqrt(sum/d[i]);

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

if (j > i)

{

sum = A[i][j];

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

sum -= S[k][j] \* S[k][i] \* d[k];

S[i][j] = sum / (d[i]\*S[i][i]);

}

else if (j < i)

S[i][j] = 0;

}

return S; }

**//Нахождение X**

double[] findY (double[][] S, double[] b)

{

int n = S.length;

double[] y =new double[n];

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

{

double sum = b[i];

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

sum -= S[k][i] \* y[k];

//System.out.println("y"+(i+1) + " = " + sum + " / " + S[i][i]);

y[i] = sum / S[i][i];

}

return y;

}

**//Нахождение Y**

double[] findX (double[][] S, double[] y)

{

int n = S.length;

double[] x = new double[n];

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

{

double sum = y[i];

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

sum -= S[i][k] \* x[k]\*d[i];

x[i] = sum / (S[i][i] \* d[i]);

}

return x;

}

**//Поиск определителя**

double findDet (double[][] S)

{

int n = S.length;

double det = 1;

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

det \*= d[i] \* S[i][i];

return det;

}}

**Vector.java**

**Класс для работы с векторами**

public class Vector {

**//Инициализация вектора**

double[] VectorR (double[][]a,int n,double[] x){

double[] b = new double[n];

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

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

b[i]+=(a[i][j]\*x[j]);

}

b[i]-=a[i][n-1];

}

return b;

} }

**Umnozit.java**

**Класс для умножения матриц на матрицы и вектора**

public class Umnozit {

**//Умножение матрицы на матрицу**

double[][] doIt(double[][]mA, double[][]mB){

int m = mA.length;

int n = mB[0].length;

int o = mB.length;

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

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

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

for (int k = 0; k < o; k++) {

res[i][j] += mA[i][k] \* mB[k][j];

}}

}

return res;

}

**//Умножение матрицы на вектор**

double[] doItWithVect(double[][]mA, double[]mB){

int m = mA.length;

double[] res = new double[m];

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

//System.out.println("res"+i + " = ");

for (int k = 0; k < m; k++) {

//System.out.println(mA[i][k] + " \* " + mB[k] + " + ");

res[i] += mA[i][k] \* mB[k];

}

}

return res;

}}

**Output.java**

**Класс для вывода результатов**

public class Output {

**//Вывод матрицы**

void Output(double[][]ma1,int n1, int m1){

NumberFormat formatter = NumberFormat.getNumberInstance();

formatter.setMaximumFractionDigits(5);

for(int i = 0; i < n1; ++i) {

for(int j = 0; j < m1; ++j) {

if(ma1[i][j]==0) System.out.print("0.0000 ");

else System.out.print(formatter.format(ma1[i][j]) + " ");

//System.out.print(ma1[i][j] + " ");

}

System.out.println();

}

System.out.println();

}

**//Вывод X**

void OutputX(double[]x, int n){

NumberFormat formatter = NumberFormat.getNumberInstance();

formatter.setMaximumFractionDigits(5);

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

System.out.print("x"+(j+1)+" = "+formatter.format(x[j]) + "\n");

}}

**//Вывод вектора**

void OutputVector(double[]x, int n){

NumberFormat formatter = NumberFormat.getNumberInstance();

formatter.setMaximumFractionDigits(5);

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

System.out.print(formatter.format(x[j]) + "\n");

}}}

**6. Входные данные**

**А b**

0.4997 -0.0658 0.0132 0.0263 0.0921 -2.8141

0.0684 0.7824 0.0000 -0.0526 0.0526 2.4104

0.0395 0.0000 0.6286 -0.1841 0.1052 2.2828

-0.0789 0.1657 0.0000 0.6181 -0.0263 -1.6332

0.3288 0.0000 0.1184 0.0132 0.7364 1.8936

**7. Выходные данные**

Source matrix A:

0,4997 -0,0658 0,0132 0,0263 0,0921

0,0684 0,7824 0.0000 -0,0526 0,0526

0,0395 0.0000 0,6286 -0,1841 0,1052

-0,0789 0,1657 0.0000 0,6181 -0,0263

0,3288 0.0000 0,1184 0,0132 0,7364

Source vector B:

-2,8141

2,4104

2,2828

-1,6332

1,8936

Matrix A(t)\*A:

0,37027 0,00756 0,07036 -0,04216 0,29798

0,00756 0,64394 -0,00087 0,05953 0,03074

0,07036 -0,00087 0,40933 -0,11382 0,15453

-0,04216 0,05953 -0,11382 0,41957 -0,02625

0,29798 0,03074 0,15453 -0,02625 0,56529

Vector A(t)\*B:

-0,39969

1,80044

1,62202

-1,60555

1,54516

Matrix S:

0,6085 0,01243 0,11562 -0,06928 0,48969

0.0000 0,80236 -0,00287 0,07527 0,03072

0.0000 0.0000 0,62925 -0,1678 0,15575

0.0000 0.0000 0.0000 0,61721 0,05104

0.0000 0.0000 0.0000 0.0000 0,54561

Vector Y: Vector X:

-0,65684 -6,0052

2,25411 3,00027

2,7087 2,00029

-2,2135 -3,99978

2,72843 5,00072

Determinant: 0.10345886940169502