Министерство просвещения Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий
Кафедра прикладной математики

Отчет защищен с оценкой___________________
Преподаватель Кантор С.А
«___» ____________ 2021 г.

Отчет
по лабораторной работе № 2

«Решение СЛАУ методом квадратного корня»

по дисциплине
«Вычислительная математика»

Студент группы ПИ-81 (Б): И. А. Песняк

Преподаватель: доцент, к.ф-м.н., Кантор С. А.

Барнаул 2021

Задание:

Составить программу для решения системы линейных алгебраических уравнений с симметричной матрицей методом квадратного корня. Предусмотреть в программе возможность вычисления обратной матрицы. Исходные данные - матрица системы уравнений и столбец свободных членов должны читаться из файла, а результаты отчетов помещаться в файл. Для Ваших тестовых примеров сравнить результаты расчетов, полученные по методу Гаусса и методу квадратного корня.

Исследовать зависимость числа обусловленности матрицы Гильберта от числа n для n = 2,3,...,7.

Краткое описание:

Метод квадратного корня используется для разложения симметричной матрицы $\boldsymbol{A}$:

$$\boldsymbol{S^T D S} = \boldsymbol{A}$$

где S – верхняя треугольная матрица, S^T – транспонированная матрица S, D – матрица в которой, $\forall i, j$: $\mathrm{d}[i][j] = \pm 1$ при $i = j$, $\mathrm{d}[i][j] = 0$ при $i \neq j$.

Вычисление элементов матрицы S происходит построчно сверху вниз, в строке слева направо. С помощью выведенных расчетных формул находим:

$$s_{ii} = \sqrt{\left| a_{ii} - \sum_{l=1}^{i-1} s_{li}^2 d_{ll} \right|}$$

$$d_{ii} = sign\left(a_{ii} - \sum_{l=1}^{i-1} s_{li}^2 d_{ll} \right),$$

в частности,

$$s_{11} = \sqrt{a_{11}},$$

$$d_{11} = sign(a_{11}),$$

а при $i < j$:

$$s_{ij} = \frac{a_{ij} - \sum_{l=1}^{i-1} s_{li} s_{lj} d_{ll}}{s_{ii} d_{ii}}.$$

Если при некотором i оказалось, что $s_{ii} = 0$, то вычисления можно продолжить найдя некоторое k, при котором $a_{kk} \neq a_{ii}$, а затем переставить в матрице A столбцы k и i и строчки с теми же индексами, в матрице s достаточно переставить соответствующие столбцы. В правой части также нужно поменять местами i-ый и k-ый элементы.

Чтобы решить СЛАУ с помощью полученного разложение необходимо последовательно вычислить: $\boldsymbol{S^T z = b, Dy = z, Sx = y}$**.** Тот же порядок применяется и при вычислениях по столбцам обратной матрицы.

Тестовые примеры:

in.txt(1)

```
4
5 7 6 5 23
7 10 8 7 32
6 8 10 9 33
5 7 9 10 31
```

out.txt

```
Матрица S:
 2,236  3,130  2,683  2,236
 0,000  0,447 -0,894  0,000
 0,000  0,000  1,414  2,121
 0,000  0,000  0,000  0,707
Матрица D:
 1  0  0  0
 0  1  0  0
 0  0  1  0
 0  0  0  1
x1 = 1,000   x2 = 1,000   x3 = 1,000   x4 = 1,000
detA = 1,00
Обратная Матрица:
 67,998 -40,999 -17,000  10,000
-40,999  24,999  10,000  -6,000
-17,000  10,000   5,000  -3,000
 10,000  -6,000  -3,000   2,000
```

---

in.txt(2)

```
4
5 7 6 5 23,01
7 10 8 7 31,99
6 8 10 9 32,99
5 7 9 10 31,01
```

out.txt

```
Матрица S:
 2,236  3,130  2,683  2,236
 0,000  0,447 -0,894  0,000
 0,000  0,000  1,414  2,121
 0,000  0,000  0,000  0,707
Матрица D:
 1  0  0  0
 0  1  0  0
 0  0  1  0
 0  0  0  1
x1 = 2,360   x2 = 0,180   x3 = 0,650   x4 = 1,210
detA = 1,00
Обратная Матрица:
 67,998 -40,999 -17,000  10,000
-40,999  24,999  10,000  -6,000
-17,000  10,000   5,000  -3,000
 10,000  -6,000  -3,000   2,000
```

---

in.txt(3)

```
4
5 7 6 5 23,1
7 10 8 7 31,9
6 8 10 9 32,9
5 7 9 10 31,1
```

out.txt

```
Матрица S:
 2,236  3,130  2,683  2,236
 0,000  0,447 -0,894  0,000
 0,000  0,000  1,414  2,121
 0,000  0,000  0,000  0,707
Матрица D:
 1  0  0  0
 0  1  0  0
 0  0  1  0
 0  0  0  1
x1 = 14,600   x2 = -7,200   x3 = -2,500   x4 = 3,100
detA = 1,00
Обратная Матрица:
 67,998 -40,999 -17,000  10,000
-40,999  24,999  10,000  -6,000
-17,000  10,000   5,000  -3,000
 10,000  -6,000  -3,000   2,000
```

in.txt(4)

```
3
1 2 3 3
2 4 6 5
3 6 7 8
```

out.txt

```
На диагонали матрицы s получен 0.
Произведена перестановка строк и столбцов 1 и 2
На диагонали матрицы s получен 0.
Продолжить вычисления не является возможным.
```

---

in.txt(5)

```
3
 25 -10 3 1
-10   4 2 3
  3   2 7 8
```

out.txt

```
На диагонали матрицы s получен 0.
Произведена перестановка строк и столбцов 1 и 2
Матрица S:
 5,000  0,600 -2,000
 0,000  2,577  1,242
 0,000  0,000  1,242
Матрица D:
 1  0  0
 0  1  0
 0  0 -1
x1 = 0,016   x3 = 1,062   x2 = 0,258
detA = -256,00
Обратная Матрица:
 -0,094  -0,297   0,125
 -0,297  -0,648   0,312
  0,125   0,313   0,000
```

---

in.txt(6)

```
4
1 2 3 4 10
2 7 5 1 15
3 5 8 6 22
4 1 6 6 17
```

out.txt

```
Матрица S:
 1,000  2,000  3,000  4,000
 0,000  1,732 -0,577 -4,041
 0,000  0,000  1,155  7,217
 0,000  0,000  0,000  5,074
Матрица D:
 1  0  0  0
 0  1  0  0
 0  0 -1  0
 0  0  0  1
x1 = 1,000   x2 = 1,000   x3 = 1,000   x4 = 1,000
detA = -103,00
Обратная Матрица:
  0,136   0,388  -0,709   0,553
  0,388   0,252  -0,311   0,010
 -0,709  -0,311   0,767  -0,243
  0,553   0,010  -0,243   0,039
```

---

in.txt(7)

```
2
-4 4 7,3
4 8 8,9
```

out.txt

```
Матрица S:
 2,000 -2,000
 0,000  3,464
Матрица D:
-1  0
 0  1
x1 = -0,475   x2 = 1,350
detA = -48,00
Обратная Матрица:
 -0,167   0,083
  0,083   0,083
```

```
Размер матрицы Гильберта = 2
число обусловленности с Евклидовой нормой = 19,33
число обусловленности с нормой, подчиненной векторной l1, = 27,00
Размер матрицы Гильберта = 3
число обусловленности с Евклидовой нормой = 526,16
число обусловленности с нормой, подчиненной векторной l1, = 748,00
Размер матрицы Гильберта = 4
число обусловленности с Евклидовой нормой = 15613,46
число обусловленности с нормой, подчиненной векторной l1, = 28374,39
Размер матрицы Гильберта = 5
число обусловленности с Евклидовой нормой = 480334,51
число обусловленности с нормой, подчиненной векторной l1, = 942644,80
Размер матрицы Гильберта = 6
число обусловленности с Евклидовой нормой = 14626653,57
число обусловленности с нормой, подчиненной векторной l1, = 28124505,37
Размер матрицы Гильберта = 7
число обусловленности с Евклидовой нормой = 307950093,73
число обусловленности с нормой, подчиненной векторной l1, = 628621052,90
```

Полученные данные достаточно хорошо согласуются с числами, полученными в статье «Исследование свойств матрицы Гильберта и причин ее плохой обусловленнсоти».

m	2	3	4	5	6
$\|\|H_m\|\|$	1,27	1,41	1,5	1,57	1,62
$\|\|T_m\|\|$	15,2	3,72E+2	1,03E+4	3,04E+5	9,24E+6
C_m	1,93E+1	5,24E+2	1,55E+4	4,77E+5	1,5E+7

, где C_m – Евклидова норма.

Число обусловленности матрицы Гильберта ***n*n*** возрастает как $O(\frac{(1+\sqrt{2})^{4n}}{\sqrt{n}})$.

Код программы:

```
package cm;
import java.io.*;
import java.util.Scanner;


import static java.lang.Math.*;
public class Main {
    //вычисление евклидовой нормы матрицы
    static double euclideanNorm(double[][] matrix, int n){
        double norm1 = 0;
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                norm1 += matrix[i][j] * matrix[i][j];
            }
        }
        return sqrt(norm1);
    }

    //вычисление матричной нормы, подчиненной векторной l1
    static double m_norm(double[][] matrix, int n){
        double max = Double.MIN_VALUE;
        for(int i = 0; i < n; i++) {
            double s1 = 0;
            for (int j = 0; j < n; j++) {
                s1 += abs(matrix[i][j]);
            }
            if(s1 > max)
                max = s1;
        }
        return max;
    }

    //перестановка столбцов матрцы
    static void swapColumns(double[][] matrix, int c1, int c2, int n){
       for(int i = 0; i < n; i++){
           double t = matrix[i][c1];
           matrix[i][c1] = matrix[i][c2];
           matrix[i][c2] = t;
       }
    }

    //перестановка строк матрицы
    static void swapRows(double[][] matrix, int r1, int r2, int n){
        for(int i = 0; i < n; i++){
            double[] t = matrix[r1];
            matrix[r1] = matrix[r2];
            matrix[r2] = t;
        }
    }

    //решение системы с верхней треугольной матрицей
    static void solveSystem(double[][] s, double[] d, double[] x, double[] b,
int n){
        double z[] = new double[n];
        for (int i = 0; i < n; i++) {
            double sum = 0;
            for (int j = 0; j < i; j++) {
                sum += z[j] * s[j][i];
            }
            z[i] = (b[i] - sum) / s[i][i];
        }
        double y[] = new double[n];
```

```
        for (int i = 0; i < n; i++) {
            y[i] = z[i] * d[i];
        }

        for (int i = n - 1; i >= 0; i--) {
            double sum = 0;
            for (int j = i + 1; j < n; j++) {
                sum += x[j] * s[i][j];
            }
            x[i] = (y[i] - sum) / s[i][i];
        }
    }

    //вычиление чисел обусловленности гильбертвоых матриц
    static double hilbert(){
        for(int size = 2; size <= 7; size++){
            double matrix[][] = new double[size][size];
            for(int i = 0; i < size; i++){
                for(int j = 0; j < size; j++){
                    matrix[i][j] = 1.0f / (i + j + 1);
                }
            }
            double s[][] = new double[size][size];
            double d[] = new double[size];
            squareRootDecomposition(matrix, s, d,  null, 0.00001f, null,
size);
            double invMatrix[][] = new double[size][size];
            getInverseMatrix(s, invMatrix, null, d, size);

            System.out.println("Размер матрицы Гильберта = " + size);
            System.out.printf("число обусловленности с Евклидовой нормой =
%.2f \n",euclideanNorm(matrix, size)*euclideanNorm(invMatrix, size));
            System.out.printf("число обусловленности с нормой, подчиненной
векторной l1, = %.2f \n", m_norm(matrix, size) * m_norm(invMatrix, size));
        }
        return 0;
    }

    static boolean squareRootDecomposition(double[][] A, double[][] s,
double[] d, double[] b, double e, int[] row, int n){
        boolean isDecompositionPossible = true;
        for(int i = 0; i < n; i++){
            //вычисляем диагональный элемент
            double sum = 0;
            for(int j = 0; j < i; j++){
                sum += s[j][i] * s[j][i] * d[j];
            }
            d[i] = signum(A[i][i] - sum);
            s[i][i] = (double)sqrt(abs(A[i][i] - sum));

            //если на диагонали оказался ноль
            if (abs(s[i][i]) < e){
                System.out.println("На диагонали матрицы s получен 0.");
                int k;
                boolean isAnotherExist = false;
                for(k = i + 1; k < n; k++)
                    if(abs(A[k][k] - A[i][i]) > e){
                        isAnotherExist = true;
                        break;
                    }
                if(isAnotherExist){
                    swapColumns(A, i, k, n);
                    swapRows(A, i, k, n);
                    swapColumns(s, i, k, n);
```

```
                    double t = b[i];
                    b[i] = b[k];
                    b[k] = t;
                    int t2 = row[i];
                    row[i] = row[k];
                    row[k] = t2;
                    System.out.println("Произведена перестановка строк и 
столбцов " + i + " и " + k);
                    //рестарт итерации
                    i--; continue;
                }
                else {
                    isDecompositionPossible = false;
                    System.out.println("Продолжить вычисления не является 
возможным.");
                }
            }
            //дальше по столбцам строки
            for(int k = i + 1; k < n; k++){
                double sum2 = 0;
                for(int j = 0; j < i; j++){
                    sum2 += s[j][i] * s[j][k]*d[j];
                }
                s[i][k] = (A[i][k] - sum2)/ (s[i][i]* d[i]);
            }
        }
        return isDecompositionPossible;
    }

    //получение обратной матрицы
    static void getInverseMatrix(double[][] s, double[][] inverseMatrix, 
int[] row, double d[], int n){
        for(int i = 0; i < n; i++){
            double right[] = new double[n];
            right[i] = 1;
            double column[] = new double[n];
            solveSystem(s, d, column, right, n);
            for(int j = 0; j < n; j++)
                if(row != null)
                    inverseMatrix[row[j]][row[i]] = column[j];
                else
                    inverseMatrix[j][i] = column[j];
        }
    }



    public static void main(String[] args) throws IOException {
        hilbert();
        OutputStream os = System.out;
        PrintStream fos = new PrintStream(new FileOutputStream("out.txt"));
        System.setOut(fos);
        boolean isDecompositionPossible = true;
        Scanner fin = new Scanner(new File("in.txt"));
        int n = fin.nextInt();

        final double e = 0.00001f;
        double[][] A = new double[n][n];
        double[][] originalMatrix = new double[n][n];
        for (int i = 0; i < n; i++) originalMatrix[i] = A[i].clone();
        double b[] = new double[n];
        double originalB[] = b.clone();
        double x[] = new double[n];
```

```
        //запоминаем перестановки строк
        //i - номер строки в текущей матрице,
        //row[i] - номер строки в исходной матрице
        int row[] = new int[n];
        for(int i = 0; i < n; i++)
            row[i] = i;

        //чтение матрицы из файла
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                A[i][j] = fin.nextDouble();
            }
            b[i] = fin.nextDouble();
        }

        double d[] = new double[n];
        double s[][] = new double[n][n];

        isDecompositionPossible = squareRootDecomposition(A, s, d, b, e, row,
n);

        if(isDecompositionPossible) {
            System.out.println("Матрица S: ");
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    System.out.printf("%6.3f ", s[i][j]);
                }
                System.out.println();
            }
            System.out.println("Матрица D: ");
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    if (i == j)
                        System.out.printf("%2.0f ", d[i]);
                    else
                        System.out.printf("%2.0f ", 0.0);
                }
                System.out.println();
            }

            /*
                Решение системы
             */
            solveSystem(s, d, x, b, n);

            for (int i = 0; i < n; i++) {
                System.out.printf("x%d = %.3f   ", row[i] + 1, x[i]);
            }
            System.out.println();

            /*
                Нахождение определителя
             */
            double det = 1;
            for(int i = 0; i < n; i++){
                det *= s[i][i] * (s[i][i] * d[i]);
            }
            System.out.printf("detA = %.2f \n", det);

            /*
                Нахождение обратной матрицы
             */
            double inverseMatrix[][] = new double[n][n];
            getInverseMatrix(s, inverseMatrix, row, d, n);
```

```
            System.out.println("Обратная Матрица: ");
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    System.out.printf("%7.3f ", inverseMatrix[i][j]);
                }
                System.out.println();
            }
        }
    }
}
```