Министерство образования и науки 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 № 1

«Решение СЛАУ методом Гаусса»

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 методом Гаусса с выбором главного элемента, нахождения определителя матрицы системы и вычисления обратной матрицы. Исходные данные – матрица системы уравнений и столбец свободных членов должны читаться из файла, а результаты расчетов помещаться в файл. В случае, когда матрица системы вырождена, выдать об этом сообщение. В противном случае вывести решение системы, невязки, величину определителя, обратную матрицу. Подобрать тестовые примеры, предусматривающие различные ситуации (матрица вырожденная, невырожденная) и провести вычисления.

Краткое описание:

При решении СЛАУ методом Гаусса матрица в первую очередь приводится к треугольному виду путем вычитания i-строки из всех строк ниже нее для обнуления элементов i-столбца ниже главной диагонали. Для минимизации погрешностей из-за ошибок округления алгоритм реализован с выбором главного элемента, во время k-цикла среди чисел ниже a_{sk}, s = k, k+1, … , n находим наибольший по модулю и выводим его на главную диагональ. Если найденный элемент оказался равен нулю (меньше ε = 0.00001), значит матрица вырождена. После приведения матрицы к треугольному виду очень просто вычислить определитель по формуле:

$$\det A = \pm \prod_{k=1}^{n} a_{kk}^{(k)}$$

Для вычисления обратной матрицы используется уже ранее приведенная к треугольному виду матрица, которая и далее остается в неизменном состоянии. Решаем n систем со специальной правой частью и из решений строим по столбцам обратную матрицу. Для преобразования правой части используются числа $C_{pk} = a_{pk}^{(k)} / a_{kk}^{(k)}$, p > k, сохраненные ранее на месте нулей в треугольной матрице.

Была выбрана евклидова норма:

$$\|A\| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}$$

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

```
Приведенная матрица:
  7,000  10,000   8,000   7,000 | 32,000
  0,000  -0,571   3,143   3,000 |  5,571
  0,000   0,000   2,500   4,250 |  6,750
  0,000   0,000   0,000   0,100 |  0,100
```

```
Решение:
  1,000   1,000   1,000   1,000
Невязка:
  0,000   0,000   0,000   0,000
detA = 1,000
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
||A|| * ||invA|| = 1727,583
```

---

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

```
Приведенная матрица:
  7,000  10,000   8,000   7,000 | 31,990
  0,000  -0,571   3,143   3,000 |  5,570
  0,000   0,000   2,500   4,250 |  6,767
  0,000   0,000   0,000   0,100 |  0,121
Решение:
  2,360   0,180   0,650   1,210
Невязка:
  0,000   0,000   0,000   0,000
detA = 1,000
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
||A|| * ||invA|| = 1727,583
```

---

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

```
Приведенная матрица:
  7,000  10,000   8,000   7,000 | 31,900
  0,000  -0,571   3,143   3,000 |  5,557
  0,000   0,000   2,500   4,250 |  6,925
  0,000   0,000   0,000   0,100 |  0,310
Решение:
 14,600  -7,200  -2,500   3,100
Невязка:
  0,000   0,000   0,000   0,000
detA = 1,000
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
||A|| * ||invA|| = 1727,583
```

in.txt(4)

```
4
6 9 6 2 23
2 15 8 7 32
6 8 5 14 33
5 7 9 10 31
```

out.txt

```
Приведенная матрица:
  6,000   9,000   6,000   2,000 | 23,000
  0,000  12,000   6,000   6,333 | 24,333
  0,000   0,000   4,250   8,597 | 12,847
  0,000   0,000   0,000  13,539 | 13,539
Решение:
  1,000   1,000   1,000   1,000
Невязка:
  0,000   0,000   0,000   0,000
detA = -4143,000
Обратная матрица:
  0,178  -0,121   0,071  -0,050
  0,032   0,082   0,036  -0,113
 -0,031  -0,003  -0,149   0,218
 -0,083   0,007   0,074   0,009
||A|| * ||invA|| = 10,197
```

---

in.txt(5)

```
4
6 9 6 2 23,01
2 15 8 7 31,99
6 8 5 14 32,99
5 7 9 10 31,01
```

out.txt

```
Приведенная матрица:
  6,000   9,000   6,000   2,000 | 23,010
  0,000  12,000   6,000   6,333 | 24,320
  0,000   0,000   4,250   8,597 | 12,848
  0,000   0,000   0,000  13,539 | 13,518
Решение:
  1,002   0,998   1,003   0,998
Невязка:
  0,000   0,000   0,000   0,000
detA = -4143,000
Обратная матрица:
  0,178  -0,121   0,071  -0,050
  0,032   0,082   0,036  -0,113
 -0,031  -0,003  -0,149   0,218
 -0,083   0,007   0,074   0,009
||A|| * ||invA|| = 10,197
```

---

in.txt(6)

```
4
6 9 6 2 23,1
2 15 8 7 31,9
6 8 5 14 32,9
5 7 9 10 31,1
```

out.txt

```
Приведенная матрица:
  6,000   9,000   6,000   2,000 | 23,100
  0,000  12,000   6,000   6,333 | 24,200
  0,000   0,000   4,250   8,597 | 12,858
  0,000   0,000   0,000  13,539 | 13,329
Решение:
  1,018   0,980   1,034   0,985
Невязка:
  0,000   0,000   0,000   0,000
detA = -4143,000
Обратная матрица:
  0,178  -0,121   0,071  -0,050
  0,032   0,082   0,036  -0,113
 -0,031  -0,003  -0,149   0,218
 -0,083   0,007   0,074   0,009
||A|| * ||invA|| = 10,197
```

---

in.txt(7)

```
3
1 2 3 2
2 4 6 4
1 1 1 5
```

out.txt

```
Приведенная матрица:
  2,000   4,000   6,000 |  4,000
  0,000  -1,000  -2,000 |  3,000
  0,000   0,000   0,000 |  0,000
Матрица вырождена, detA = 0
```

---

in.txt(8)

```
3
2 4 9 7
1 8 5 2
11 9 7 2
```

out.txt

```
Приведенная матрица:
 11,000   9,000   7,000 |  2,000
  0,000   7,182   4,364 |  1,818
  0,000   0,000   6,291 |  6,038
Решение:
 -0,159  -0,330   0,960
Невязка:
  0,000   0,000   0,000
detA = -497,000
Обратная матрица:
 -0,022  -0,107   0,105
 -0,097   0,171   0,002
  0,159  -0,052  -0,024
||A|| * ||invA|| = 5,702
```

in.txt(9)

```
2
4 3,19 7,19
5 4    9
```

out.txt

```
Приведенная матрица:
  5,000   4,000 |  9,000
  0,000  -0,010 | -0,010
Решение:
  1,000   1,000
Невязка:
  0,000   0,000
detA = 0,050
Обратная матрица:
 80,000 -63,800
-100,000  80,000
||A|| * ||invA|| = 1057,776
```

---

in.txt(10)

```
2
4 3,19 7,3
5 4 8,9
```

out.txt

```
Приведенная матрица:
  5,000   4,000 |  8,900
  0,000  -0,010 |  0,180
Решение:
 16,180 -18,000
Невязка:
  0,000   0,000
detA = 0,050
Обратная матрица:
 80,000 -63,800
-100,000  80,000
||A|| * ||invA|| = 1057,776
```

Пояснения: в тестах 1-3 (а также 9-10) незначительные изменения правой части СЛАУ влекут за собой сильное непропорциональное изменение корней. Связано это с большим значением числа обусловленности матрицы $\|A\|\|A^{-1}\| = 1727$. Для сравнения в тестах 4-6 берется похожая система, но ее число обусловленности равно 10, поэтому система более устойчивая по правой части.

Вывод: при проведении расчетов на вычислительных машинах не стоит забывать об особенностях машинной арифметики: о точности и допустимых значениях, об ошибках округления и об отсутствии ассоциативности сложения. Следует грамотно подходить к порядку вычислений, находить уязвимые места и стараться компенсировать их алгоритмическими и математическими манипуляциями.

Код программы:

```
package com.company;
import java.io.*;
import java.util.Scanner;

import static java.lang.Math.abs;
import static java.lang.Math.sqrt;

public class Main {
    public static void main(String[] args) throws FileNotFoundException,
IOException {
        OutputStream os = System.out;
        PrintStream fos = new PrintStream(new FileOutputStream("out.txt"));
        System.setOut(fos);
        boolean isMatrixDegenerate = false;
        Scanner fin = new Scanner(new File("in.txt"));
        int n = fin.nextInt();

        final float e = 0.00001f;
        float[][] matrix = new float[n][n];
        float[][] originalMatrix = new float[n][n];
        for(int i = 0; i < n; i++) originalMatrix[i] = matrix[i].clone();
        float b[] = new float[n];
        float originalB[] = b.clone();
        float x[] = new float[n];
        int detSign = 1;
        //чтение матрицы из файла
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = fin.nextFloat();
            }
            b[i] = fin.nextFloat();
        }

        //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;


        /*
            прямой ход
         */

        //приведение к треугольному виду
        //по строкам
        for (int i = 0; i < n - 1; i++) {
            //поиск главного элемента
            float max = matrix[i][i];
            int imax = i;
            for (int j = i; j < n; j++)
                if (abs(matrix[j][i]) > max) {
                    max = abs(matrix[j][i]);
                    imax = j;
                }
            //если строчка с максимальным элементом не текущая
            if (imax != i) {
                float[] t1 = matrix[imax];
                matrix[imax] = matrix[i];
                matrix[i] = t1;
                float t2 = b[imax];
```

```
                b[imax]= b[i];
                b[i] = t2;
                int t3 = row[i];
                row[i] = row[imax];
                row[imax] = t3;
                detSign *= -1;
            }
            //число на главной диагонали и числа ниже - нули
            //матрица вырождена, определитель равен 0
            if(abs(matrix[i][i]) < e){
                isMatrixDegenerate = true;
                break;
            }


            //по оставшимся строкам
            for (int j = i + 1; j < n; j++) {
                float koef = matrix[j][i] / matrix[i][i];
                //сохраняем значения коэффициента для вычисления обратной
матрицы
                matrix[j][i] = koef;
                //по столбцам
                for (int k = i + 1; k < n; k++) {
                    matrix[j][k] -= matrix[i][k] * koef;
                }
                b[j] -= b[i] * koef;

            }
        }
        //правый нижний элемент оказался равен нулю
        if(matrix[n-1][n-1] == 0)
            isMatrixDegenerate = true;


        //вывод получившейся матрицы
        System.out.println("Приведенная матрица: ");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if(i <= j)
                    System.out.printf("%7.3f ", matrix[i][j]);
                else
                    System.out.printf("%7.3f ", 0.0);
            }
            System.out.printf("|%7.3f\n", b[i]);
        }


        if(!isMatrixDegenerate) {
            /*
                Обратный ход
             */
            for (int i = n - 1; i >= 0; --i) {

                float tempb = b[i];
                for (int j = n - 1; j > i; --j) {
                    tempb -= matrix[i][j] * x[j];
                }
                x[i] = tempb / matrix[i][i];
            }

            System.out.println("Решение:");
            for (float __ : x) {
                System.out.printf("%7.3f ", __);
            }
```

```
            System.out.println();
             /*
                Вычисление невязки
                Подставляем x в исходную матрицу
             */
            float discrepancy[] = new float[n];
            for(int i = 0; i < n; i++){
                float left = 0;
                for(int j = 0; j < n; j++){
                    left += originalMatrix[i][j] * x[j];
                }
                discrepancy[i] = originalB[i] - left;
            }

            System.out.println("Невязка: ");
            for(float __: discrepancy){
                System.out.printf("%7.3f ", __);
            }
            System.out.println();

            //вычисление определителя
            double det = 1;
            for (int i = 0; i < n; i++){
                det *= matrix[i][i];
            }
            det *= detSign;
            System.out.printf("detA = %.3f\n", det);

            /*
                Вычисление обратной матрицы
             */

            float inverseMatrix[][] = new float[n][n];
            for(int q = 0; q < n; q++){
                float newB[] = new float[n];
                newB[q] = 1;
                //исправляем правую часть
                //начинаем с q, т.к. выше в правой части нули
                for(int j = q; j < n - 1; j++){
                    for(int i = j + 1; i < n; i++){
                        newB[i] -= newB[j] * matrix[i][j];
                    }
                }

                /*
                    Обратный ход
                */
                float[] tempX = new float[n];
                for (int i = n - 1; i >= 0; --i) {
                    float tempb = newB[i];
                    for (int j = n - 1; j > i; --j) {
                        tempb -= matrix[i][j] * tempX[j];
                    }
                    tempX[i] = tempb / matrix[i][i];
                }
                for(int i = 0; i < n; i++)
                    inverseMatrix[i][row[q]] = tempX[i];

            }
            //вывод обратной матрицы
            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();
            }

            //Вычисление евклидовых норм
            double norm1 = 0;
            double norm2 = 0;
            for(int i = 0; i < n; i++){
                for(int j = 0; j < n; j++){
                    norm1 += matrix[i][j] * matrix[i][j];
                    norm2 += inverseMatrix[i][j] * inverseMatrix[i][j];
                }
            }
            norm1 = sqrt(norm1);
            norm2 = sqrt(norm2);
            System.out.printf("||A|| * ||invA|| = %.3f", norm1 * norm2);
        }
        else {
            System.out.println("Матрица вырождена, detA = 0");
        }

    }
}
```