Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1
по «Вычислительной математике»

Вариант 9

Выполнил: Кривошейкин Сергей

Группа P3214

Преподаватель: Малышева Т. А.

Санкт-Петербург

2020

Цель работы: написать программу, решающую систему линейных алгебраических уравнений СЛАУ

Метод: Метод Гаусса.

Основан на приведении матрицы системы к треугольному виду так, чтобы ниже ее главной диагонали находились только нулевые элементы.

Прямым ходом метода Гаусса состоит в последовательном исключении неизвестных из уравнений системы.

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: начиная с последнего пошагово придем к первому.

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1,$$
$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2,$$
$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$
$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n.$$

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}\,, i,j = 2,3 \dots n$$
$$b_i^{(1)} = b_i - \frac{a_{i1}}{a_{11}} b_1\,, i = 2,3 \dots n$$

Для последующих элементов аналогично

$$x_n = {b_n^{(n-1)}} \Big/ {a_{nn}^{(n-`1)}}$$

Листинг программы:

```
public class Gauss {

  private int size;
  private double[][] matrix;
  private double[][] matrix2;

  public Gauss(int size, double[][] matrix) {
    this.size = size;
    this.matrix = matrix;
    this.matrix2 = matrix;
  }

  public double getA() {
    return det(matrix2);
  }

  private double[][] GetMinor(double[][] matrix, int row, int column) {
    int minorLength = matrix.length - 1;
    double[][] minor = new double[minorLength][minorLength];

    int dI = 0; //переменные для пропуска ненужных строк и столбцов
    int dJ = 0;
    for (int i = 0; i <= minorLength; i++) {
      dJ = 0;
      for (int j = 0; j <= minorLength; j++) {
```

```
            if (i == row) {
                dI = 1;
            } else {
                if (j == column) {
                    dJ = 1;
                } else {
                    minor[i - dI][j - dJ] = matrix[i][j];
                }
            }
        }
    }

    return minor;
}

public double det(double[][] matrix) {
    double Det = 0.0;
    if (matrix.length == 2) {
        Det = matrix[0][0] * matrix[1][1] - matrix[1][0] * matrix[0][1];
    } else {
        int koeff = 1;
        for (int i = 0; i < matrix.length; i++) {
            if (i % 2 == 1) {
                koeff = -1;
            } else {
                koeff = 1;
            }
            Det += koeff * matrix[0][i] * this.det(this.GetMinor(matrix, 0, i));
        }
    }
    return Det;
}

public boolean checkIfHasSolutions() {
    boolean f = true;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            if (matrix[i][j] != 0)
                f = false;
        }
        if (f)
            if (matrix[i][size] != 0)
                return false;
        f = true;
    }
    return true;
}

public double[] getSolutions() {
    double[] sol = new double[size];
    double b = 0;
    for (int i = size - 1; i >= 0; i--) {
        if (i == size - 1)
            if (matrix[i][size - 1] != 0)
                sol[i] = matrix[i][size] / matrix[i][size - 1];
            else sol[i] = 0;
        else {
```

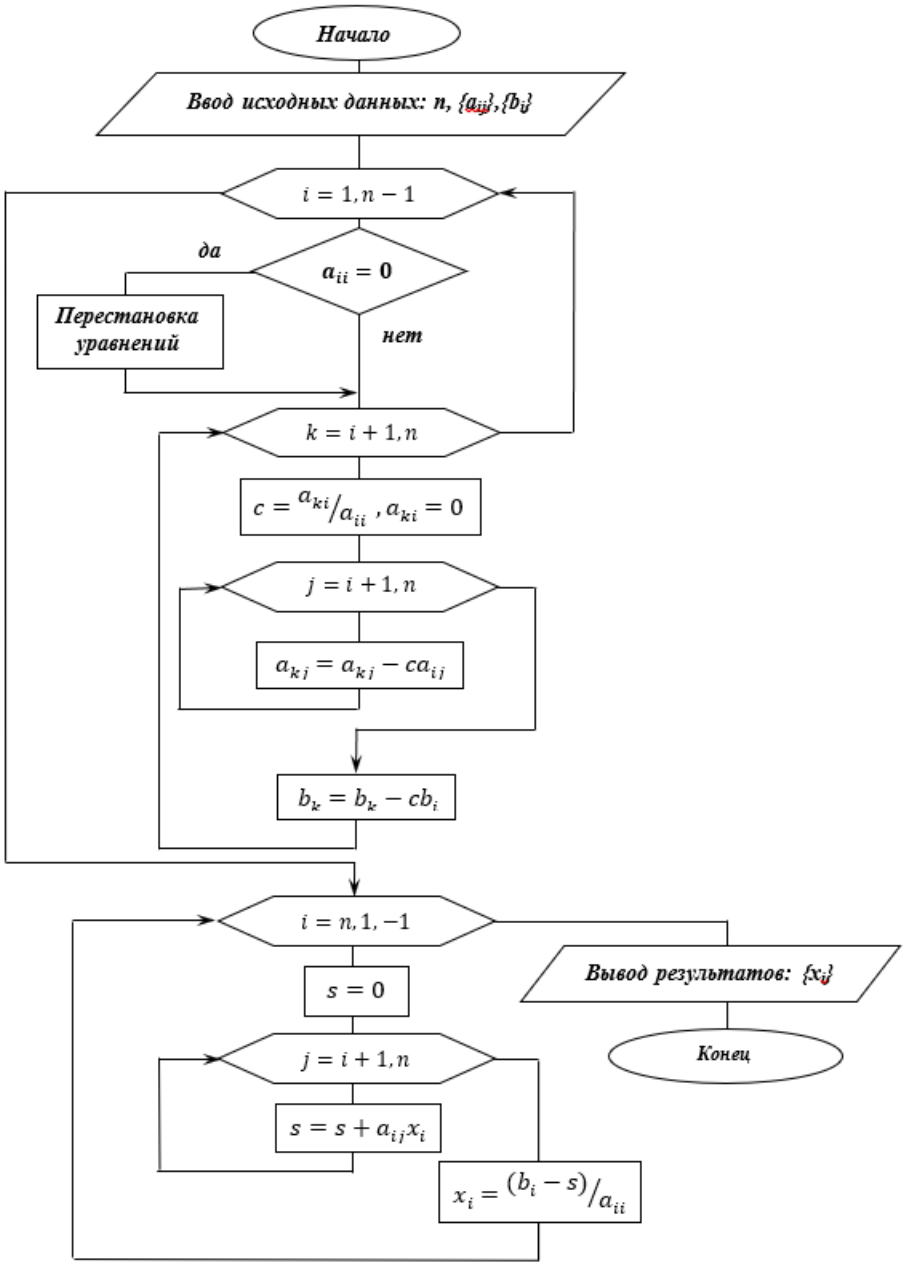
```
            for (int j = 0; j < size; j++) {
                if (matrix[i][j] != 0) {
                    for (int k = j + 1; k < size; k++)
                        b += (matrix[i][k] * sol[k]);
                    sol[i] = (matrix[i][size] - b) / matrix[i][j];
                    break;
                }
            }
            b = 0;
        }
    }
    return sol;
}

public double[] getError(double[] sol) {
    double[] err = new double[size];
    for (int i = 0; i < size; i++)
        err[i] = 0;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++)
            err[i] += matrix[i][j] * sol[j];
        err[i] -= matrix[i][size];
    }
    return err;
}

public double[][] getTriangular() {
    double number;

    for (int k = 0; k < size - 1; k++) {
        number = matrix[k][k];
        if ((k != 0) && (number == 0)) {
            continue;
        }
        for (int i = 0; i < size + 1; i++)
            matrix[k][i] /= number;
        for (int i = k + 1; i < size; i++) {
            number = matrix[i][k];
            for (int j = 0; j < size + 1; j++) {
                matrix[i][j] -= matrix[k][j] * number;
            }
        }
        if (!checkIfHasSolutions()) {
            return null;
        }
    }
    return matrix;
}
}
```

Блок-схема:



Примеры и результаты работы программы:

1) Matrix:

1,00 2,00 3,00 | 3,00
3,00 5,00 7,00 | 0,00
1,00 3,00 4,00 | 1,00

Det = 1.0

Triangular:

1,00 2,00 3,00 | 3,00
0,00 1,00 2,00 | 9,00
0,00 0,00 -1,00 |-11,00

Solution:
x[0] = -4,000

x[1] = -13,000
x[2] = 11,000

Errors:
[0] = 0.0
[1] = 0.0
[2] = 0.0

2) Matrix:

7,00	8,00	9,00	2,00	6,00	7,00
5,00	2,00	3,00	6,00	9,00	3,00
5,00	8,00	9,00	2,00	1,00	4,00
8,00	5,00	2,00	9,00	6,00	3,00
9,00	5,00	1,00	7,00	5,00	3,00

Det = 1156.0

Triangular:

1,00	1,14	1,29	0,29	0,86	1,00
0,00	1,00	0,92	-1,23	-1,27	0,54
0,00	0,00	1,00	7,33	-0,83	-4,83
0,00	0,00	0,00	1,00	-0,29	-0,71
0,00	0,00	0,00	0,00	-2,81	-2,55

Solution:
x[0] = -0,775
x[1] = 1,870
x[2] = -0,789
x[3] = -0,448
x[4] = 0,910

Errors:
[0] = 0.0
[1] = 0.0
[2] = 0.0
[3] = 0.0
[4] = 0.0

Вывод: в ходе выполнения данной лабораторной работы я научился производить матричные вычисления и написал программу по решению СЛАУ методом Гаусса.