Национальный исследовательский университет ИТМО
Факультет ПИиКТ

Лабораторная работа №1

по курсу «*Вычислительная математика*»
Вариант «Метод Гаусса»

Работу выполнил:
Глушков Даниил Григорьевич
Группа P3233

Преподаватель:
Перл О. В.

г. Санкт-Петербург
2022

1 Описание метода

Суть метода в преобразовании матрицы к треугольному виду так, чтобы на главной диагонали были единицы. После этого последовательно вычисляются корни уравнения.

При этом если появляется строка вида $\begin{pmatrix} 0 & 0 & \cdots & 0 & | & \lambda \end{pmatrix}$, где $\lambda \neq 0$, либо существуют пропорциональные или одинаковые строки, либо невозможно с помощью элементарных преобразований исключить из главной диагонали нули (т.е. существуют нулевые столбцы), то система несовместна и решений либо бесконечно много, либо не существует.

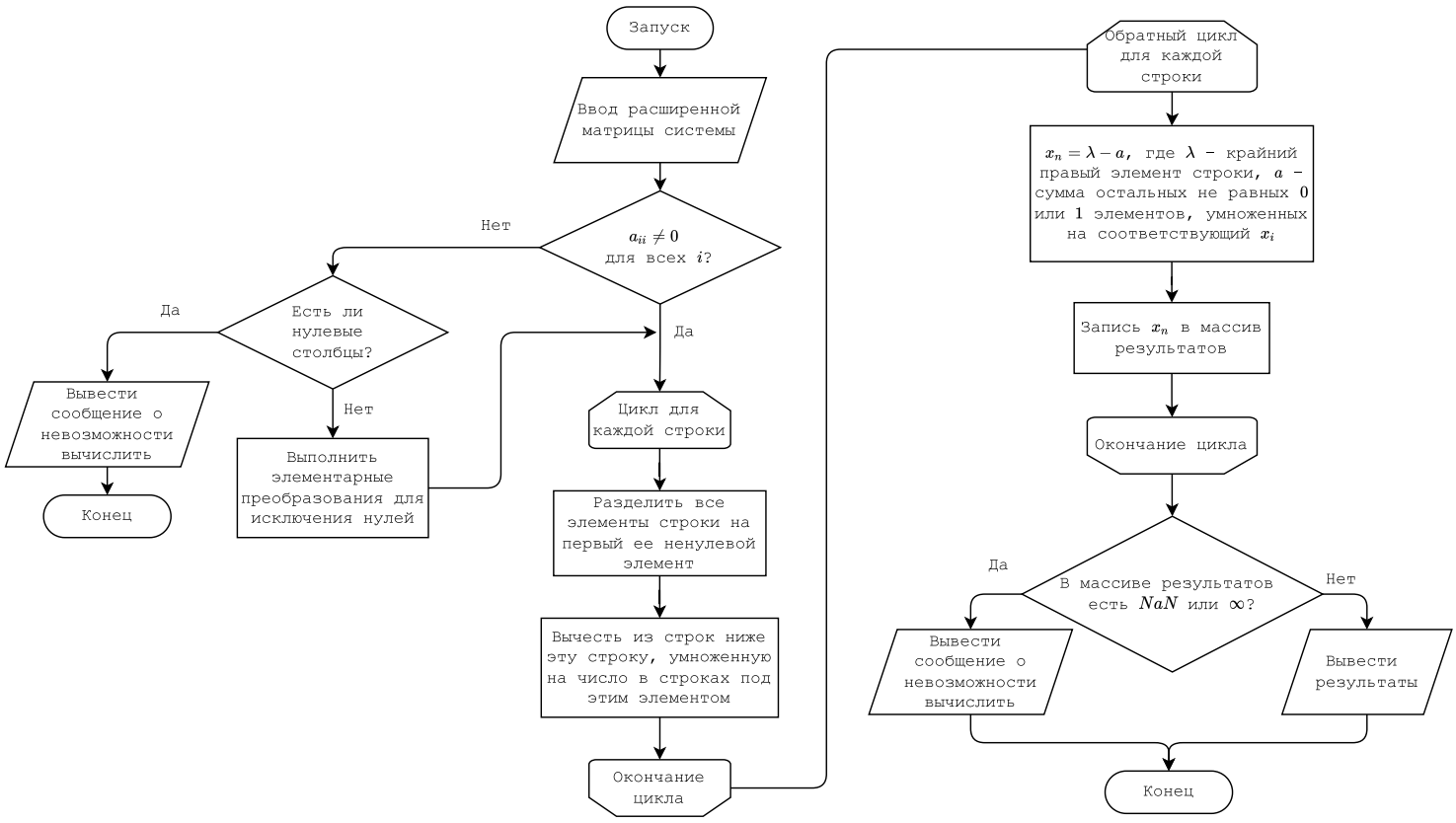
2 Расчетные формулы

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} (a_{ij} \cdot x_j)}{a_{ii}}$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \end{pmatrix} \sim \begin{pmatrix} 1 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} & \frac{b_1}{a_{11}} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{pmatrix} \sim \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22} - \frac{a_{21} \cdot a_{12}}{a_{11}} & \cdots & a_{2n} - \frac{a_{21} \cdot a_{1n}}{a_{11}} & b_2 - \frac{a_{21} \cdot b_1}{a_{11}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2} - \frac{a_{n1} \cdot a_{12}}{a_{11}} & \cdots & a_{nn} - \frac{a_{n1} \cdot a_{1n}}{a_{11}} & b_n - \frac{a_{n1} \cdot b_1}{a_{11}} \end{pmatrix}$$

3 Блок-схема



4 Листинг реализации

Полный код лабораторной доступен на:
g.mrcat-pixel.xyz/linear-eq-system-solver

```
package com.cat.math1.entities;

import com.cat.math1.exceptions.UnableToModifyMatrixException;

import java.util.Random;

public class Matrix {

    private final int size;
    private double[][] table;

    public Matrix(int size) {
        this.size = size;
        table = new double[size][size+1];
    }

    public void fillRandom() {
        for (int i = 0; i < size; i++)
            table[i] = new Random()
                    .doubles(size + 1, -20, 20)
                    .toArray();
    }

    public void setTable(double[][] table) {
        this.table = table;
    }

    public void print() {
        System.out.println("Size: " + size);
        for (double[] i : table) {
            for (int i2 = 0; i2 < size+1; i2++) {
                if (i2 == size) System.out.print("|");
                System.out.format("%15.4f ", i[i2]);
            }
            System.out.print('\n');
        }
    }

    public Results calcRes() {
        System.out.println("Checking if the main diagonal has zeroes on it...");

        if (hasZeroesOnMainDiagonal())
            try { modify(); }
            catch (UnableToModifyMatrixException e) {
                System.out.println("There is a column that's all zeroes.");
                return Results.returnInvalid();
            }
        else System.out.println("Zeroes not found, proceeding...");
```

```
        System.out.println("Triangulating the matrix...");
        triangulate();

        System.out.println("Calculating the results...");
        Results res = new Results(size);
        for (int i = size - 1; i >= 0; i--) {
            double subValue = 0;
            for (int ii = size - 1; ii > i; ii--)
                subValue += table[i][ii] * res.getAt(ii);

            res.setAt(i, table[i][size] - subValue);
        }
        return res;
    }

    public Results calcErr(Results n) {
        Results res = new Results(size);

        for (int i = 0; i < size; i++) {
            double temp = 0;
            for (int ii = 0; ii < size; ii++)
                temp += table[i][ii] * n.getAt(ii);

            temp = table[i][size] - temp;

            res.setAt(i, temp);
        }
        return res;
    }

    public Matrix returnDuplicate() {
        Matrix ret = new Matrix(size);
        ret.setTable(table);
        return ret;
    }

    private void normalizeRow(int row) {
        double del = table[row][row];
        for (int i = 0; i < size+1; i++)
            table[row][i] /= del;
    }

    private void subtractRows(int destination, int source, double multiplier) {
        for (int i = 0; i < size+1; i++)
            table[destination][i] = table[destination][i] - table[source][i]*multiplier;
    }

    private void addRows(int destination, int source) {
        for (int i = 0; i < size+1; i++)
            table[destination][i] = table[destination][i] + table[source][i];
    }

    private int findRowWithNoZeroInPlaceAtIndex(int index) throws UnableToModifyMatrixException {
        for (int i = 0; i < size; i++)
            if (table[i][index] != 0) return i;
        throw new UnableToModifyMatrixException();
    }
```

```
    private void cascadingSubtraction(int row) {
        for (int i = row + 1; i < size; i++)
            subtractRows(i, row, table[i][row]);
    }

    private void triangulate() {
        for (int i = 0; i < size; i++) {
            normalizeRow(i);
            cascadingSubtraction(i);
        }
    }

    private boolean hasZeroesOnMainDiagonal() {
        for (int i = 0; i < size; i++)
            if (table[i][i] == 0) return true;
        return false;
    }

    private void modify() throws UnableToModifyMatrixException {
        System.out.println("Zeroes found, attempting to modify the matrix...");

        for (int i = 0; i < size; i++)
            if (table[i][i] == 0)
                addRows(i, findRowWithNoZeroInPlaceAtIndex(i));
    }
}

package com.cat.math1.entities;

public class Results {

    private final int size;
    private final double[] table;

    public static Results returnInvalid() {
        Results res = new Results(1);
        res.setAt(0, Double.NaN);
        return res;
    }

    public Results(int size) {
        this.size = size;
        table = new double[size];
    }

    public boolean isValid() {
        for (double i : table)
            if (Double.isInfinite(i) || Double.isNaN(i)) return false;
        return true;
    }

    public void print() {
        if (!isValid()) System.out.println("No solution found.");
        else for (int i = 0; i < size; i++)
            System.out.format("x" + (i+1) + " = %15.4f%n", table[i]);
    }
```

```
    public void setAt(int id, double value) {
        table[id] = value;
    }

    public double getAt(int id) {
        return table[id];
    }
}
```

5 Результаты работы программы

5.1 Пример 1: случайная матрица

```
Welcome to the matrix solver. To see the list of commands, type "h". To quit, type "q".
---------------------------------------
complab1>inpr
---------------------------------------
Input matrix size (1..20):5
---------------------------------------
Generating random matrix...
The matrix is:
Size: 5
        19,1349        -14,0541         12,9946        -11,0795        -17,8138 |         17,3372
         5,4627         -8,9850          1,4155         -9,0807         -8,2865 |        -16,4823
        -8,9059         -6,3785         -4,3423         -0,6345          9,1449 |         -5,4966
         6,0612        -14,0731        -12,5273          1,2375         -3,3995 |         -6,9637
         5,1084         -0,8272         -4,8357        -12,0661        -17,7437 |         -9,9264
---------------------------------------
Confirm computation (y/n):y
Checking if the main diagonal has zeroes on it...
Zeroes not found, proceeding...
Triangulating the matrix...
Calculating the results...
The results are:
x1 =        -15,6130
x2 =         -7,4260
x3 =          7,3597
x4 =         15,8740
x5 =        -16,3898
The residual error is:
x1 =          0,0000
x2 =         -0,0000
x3 =         -0,0000
x4 =          0,0000
x5 =          0,0000
---------------------------------------
complab1>
```

5.2 Пример 2: Ручной ввод матрицы, которую можно проверить вручную

```
complab1>inpc
---------------------------------------
Input matrix size (1..20):4
---------------------------------------
Please input 4*5 doubles (using comma as a decimal separator), separated by spaces
and/or newlines:
0 1 0 0 45
1 0 0 0 24
0 0 1 2 45
0 0 1 0 23
The matrix you inputted is:
Size: 4
         0,0000          1,0000          0,0000          0,0000 |         45,0000
         1,0000          0,0000          0,0000          0,0000 |         24,0000
         0,0000          0,0000          1,0000          2,0000 |         45,0000
```

```
         0,0000          0,0000          1,0000          0,0000 |        23,0000
----------------------------------------
Confirm computation (y/n):y
Checking if the main diagonal has zeroes on it...
Zeroes found, attempting to modify the matrix...
Triangulating the matrix...
Calculating the results...
The results are:
x1 =         24,0000
x2 =         45,0000
x3 =         23,0000
x4 =         11,0000
The residual error is:
x1 =          0,0000
x2 =          0,0000
x3 =          0,0000
x4 =          0,0000
----------------------------------------
complab1>
```

5.3 Пример 3: несовместная матрица (нулевой столбец)

```
complab1>inpc
----------------------------------------
Input matrix size (1..20):3
----------------------------------------
Please input 3*4 doubles (using comma as a decimal separator), separated by spaces and/or newlines:
0 43 23 98
0 4 43 2
0 57 0 23
The matrix is:
Size: 3
         0,0000         43,0000         23,0000 |        98,0000
         0,0000          4,0000         43,0000 |         2,0000
         0,0000         57,0000          0,0000 |        23,0000
----------------------------------------
Confirm computation (y/n):y
Checking if the main diagonal has zeroes on it...
Zeroes found, attempting to modify the matrix...
There is a column that's all zeroes.
The results are:
No solution found.
----------------------------------------
complab1>
```

5.4 Пример 4: несовместная матрица (строка вида $\begin{pmatrix} 0 & 0 & \cdots & 0 & | & \lambda \end{pmatrix}$, где $\lambda \neq 0$)

```
complab1>inpc
----------------------------------------
Input matrix size (1..20):3
----------------------------------------
Please input 3*4 doubles (using comma as a decimal separator), separated by spaces and/or newlines:
1 2,4 56 23,4
```

```
12,4 2 65 34
0 0 0 23
The matrix is:
Size: 3
         1,0000          2,4000         56,0000 |        23,4000
        12,4000          2,0000         65,0000 |        34,0000
         0,0000          0,0000          0,0000 |        23,0000
---------------------------------------
Confirm computation (y/n):y
Checking if the main diagonal has zeroes on it...
Zeroes found, attempting to modify the matrix...
Triangulating the matrix...
Calculating the results...
The results are:
No solution found.
---------------------------------------
complab1>
```

6 Вывод

Метод Гаусса – достаточно универсальный метод решения СЛАУ.

Его основное достоинство в том, что он может быть применен для любой имеющей решение системы, у которой нет на главной диагонали нулей (но на практике это ограничение тривиально обойти элементарными преобразованиями над матрицей)

Тем не менее, у него существуют недостатки:

1. При вычислении на компьютере накапливается погрешность из-за ограничений операций с числами с плавающей точкой;
2. Проблемы с памятью и временем исполнения: Матрицу надо постоянно держать в памяти, и сложность алгоритма $O(n^3)$.