Федеральное государственное автономное образовательное
учреждение высшего образования Национальный
исследовательский университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №1
Вычислительная математика

Вариант: 1

Группа	P3208
Студент	Абдуллин И.Э.
Преподаватель	Машина Е.А.

1 Цель работы

1. Изучить прямые и итерационные методы решения систем линейных алгебраических уравнений
2. Выполнить программную реализацию методов

2 Описание используемого метода

В данной лабораторной работе использовался метод Гаусса для поиска решений СЛАУ.

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

Прямой ход метода Гаусса состоит в последовательном исключении неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается x_1 из всех последующих уравнений системы. Затем с помощью второго уравнения исключается x_2 из третьего и всех последующих уравнений и т.д. Этот процесс продолжается до тех пор, пока в левой части последнего (n-го) уравнения не останется лишь один член с неизвестным x_n , т. е. матрица системы будет приведена к треугольному виду.

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: решая последнее уравнение, находим единственное в этом уравнении неизвестное x_n . Далее, используя это значение, из предыдущего уравнения вычисляем x_{n-1} и т. д. Последним найдем x_1 из первого уравнения.

Метод имеет много различных вычислительных схем, но в каждой из них основным требованием является $\det A \neq 0$.

3 Расчетные формулы метода

1. Прямой и обратный ходы:

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_i^{(k)} \cdot a_j^{(k)}}{a_{ii}^{(k)}} \quad \text{для } j = i+1, i+2, \ldots, n$$

$$b_i^{(k+1)} = b_i^{(k)} - \frac{a_i^{(k)} \cdot b_j^{(k)}}{a_{ii}^{(k)}} \quad \text{для } j = i+1, i+2, \ldots, n$$

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j \right) \quad \text{для } i = n, n-1, \ldots, 1$$

2. Невязка

$$r = Ax^* - b$$

, где A - исходная матрица, x^* - вектор решений методом Гаусса, b - правая часть уравнения

4 Листинг программы

Программа написана на Java. Реализован класс MatrixExecutor.

```
public class GaussExecutor {
  private final DecimalFormat FORMAT = new DecimalFormat("#.##########");


  public void solve(double[][] matrix) {
    var m = rightMethod(matrix);
    if (m != null) {
      var vectors = backMethod(m);
      printTriangleMatrix(m);
      System.out.printf("> Determinant: %s\n\n", getDeterminate(m));
      printVectors(vectors, "> Vectors:");
```

```
      printVectors(getResidualVectors(matrix, vectors), "> ResidualVectors:");
    }
  }

  private double[][] rightMethod(double[][] m) {
    for (int i = 0; i < m.length; i++) {
      if (Double.compare(m[i][i], 0) == 0) {
        int swapRow = findNonZeroDiagonalElementRow(m, i);
        if (swapRow != -1) {
          swapRows(m, i, swapRow);
        } else {
          int rankCoefficients = calculateRank(m);
          System.out.println("> Determinant: 0");
          if (rankCoefficients < m.length) {
            System.out.println("> Vectors: inf solutions");
          } else {
            System.out.println("> Vectors: zero solutions");
          }
          return null;
        }
      }
      for (int j = i + 1; j < m.length; j++) {
        double factor = m[j][i] / m[i][i];
        for (int k = 0; k <= m.length; k++) {
          m[j][k] -= m[i][k] * factor;
        }
      }
    }
    return m;
  }

  private int findNonZeroDiagonalElementRow(double[][] m, int col) {
    for (int i = col + 1; i < m.length; i++) {
      if (m[i][col] != 0) {
        return i;
      }
    }
    return -1;
  }

  private void swapRows(double[][] m, int row1, int row2) {
    double[] temp = m[row1];
    m[row1] = m[row2];
    m[row2] = temp;
  }

  private double[] backMethod(double[][] m) {
    double[] solution = new double[m.length];
    for (int i = m.length - 1; i >= 0; i--) {
      solution[i] = m[i][m.length];
      for (int j = i + 1; j < m.length; j++) {
        solution[i] -= m[i][j] * solution[j];
      }
      double res = solution[i] / m[i][i];
      solution[i] = formatDouble(res, FORMAT);
    }
    return solution;
  }

  private void printVectors(double[] vs, String message) {
    System.out.println(message);
    for (int i = 0; i < vs.length; i++) {
      System.out.printf("x%s = %s\n", i + 1, vs[i]);
    }
    System.out.println();
  }

  private double getDeterminate(double[][] m) {
    double determinate = 1;
    for (int i = 0; i < m.length; i++) {
      determinate *= m[i][i];
    }
    return formatDouble(determinate, FORMAT);
  }

  private void printTriangleMatrix(double[][] m) {
```

```java
    System.out.println("> TriangleMatrix:");
    for (double[] doubles : m) {
      for (int j = 0; j <= m.length; j++) {
        if (j == m.length) {
          System.out.print("= " + formatDouble(doubles[j], FORMAT));
        } else {
          System.out.print(formatDouble(doubles[j], FORMAT) + " ");
        }
      }
      System.out.println();
    }
    System.out.println();
  }

  private double formatDouble(double value, DecimalFormat format) {
    String form = format.format(value).replace(',', '.');
    double n = Double.parseDouble(form);
    if (Math.abs(value - n) < 0.000000001) {
      return n;
    }
    return value;
  }

  private int calculateRank(double[][] matrix) {
    int rowCount = matrix.length;
    int colCount = matrix[0].length;

    int rank = 0;
    boolean[] rowMarked = new boolean[rowCount];

    for (int col = 0; col < colCount; col++) {
      boolean found = false;
      for (int row = 0; row < rowCount && !found; row++) {
        if (!rowMarked[row] && matrix[row][col] != 0) {
          rank++;
          rowMarked[row] = true;
          found = true;

          for (int k = row + 1; k < rowCount; k++) {
            double factor = matrix[k][col] / matrix[row][col];
            for (int j = col; j < colCount; j++) {
              matrix[k][j] -= matrix[row][j] * factor;
            }
          }
        }
      }
    }

    return rank;
  }

  private double[] getResidualVectors(double[][] m, double[] solutions) {
    var res = new double[solutions.length];
    for (int i = 0; i < m.length; i++) {
      double sum = 0;
      for (int j = 0; j < m.length; j++) {
        sum += m[i][j] * solutions[j];
      }
      double value = formatDouble(m[i][m.length] - sum, FORMAT);
      res[i] = value;
    }
    return res;
  }
}
```

5 Примеры и результаты работы программы

1. Система неопределена
2. Система неопределена, но все векторы различны
3. Система несовместна
4. Система определена

input.txt:

```
3
1 1 1 = 1
1 1 1 = 1
1 1 1 = 1

3
1 2 3 = 4
3 6 9 = 12
1 1 1 = 1

4
3 4 9 2 = 1
1 1 1 1 = 5
1 343 322 4 = 2
1 1 1 1 = 3

5
1 3 8 3 8 = 9
1 12 3 343 343 = 12
737 745 38 282 3 = 3
23 77 32 838 33 = 82
9 74 73 7333 9 = 23
```

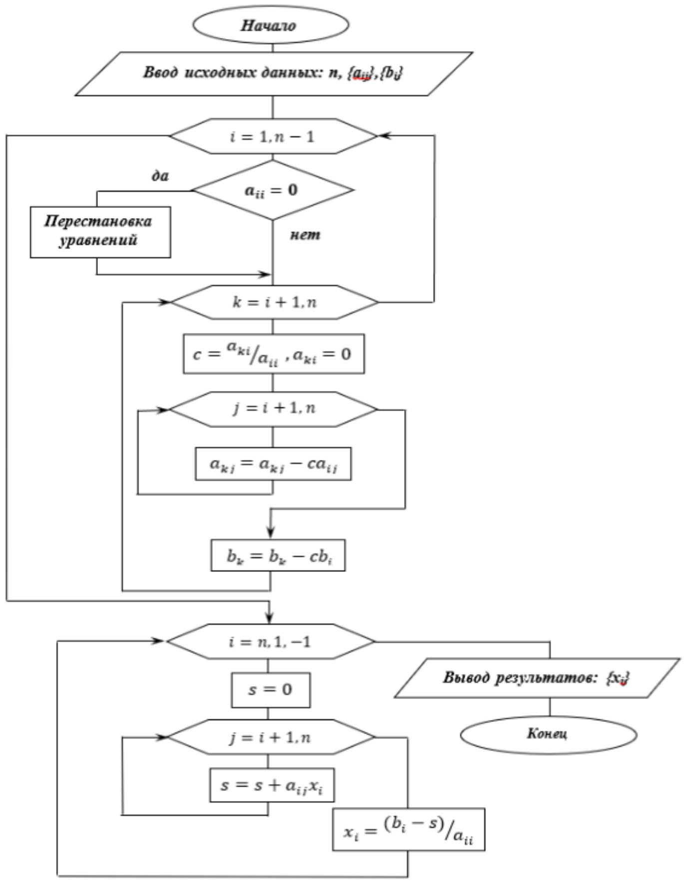
output:

```
----------------TEST #1----------------

> Исходная СЛАУ:
1.0 1.0 1.0 = 1.0
1.0 1.0 1.0 = 1.0
1.0 1.0 1.0 = 1.0

> Remove i = 1
1.0 1.0 1.0 = 1.0
0.0 0.0 0.0 = 0.0
0.0 0.0 0.0 = 0.0

> Детерминант: 0
> Векторы неизвестных: СЛАУ имеет бесконечное количество решений

----------------TEST #2----------------

> Исходная СЛАУ:
1.0 2.0 3.0 = 4.0
3.0 6.0 9.0 = 12.0
1.0 1.0 1.0 = 1.0

> Remove i = 1
1.0 2.0 3.0 = 4.0
0.0 0.0 0.0 = 0.0
0.0 -1.0 -2.0 = -3.0

> Remove i = 2
1.0 2.0 3.0 = 4.0
0.0 -1.0 -2.0 = -3.0
0.0 0.0 0.0 = 0.0

> Детерминант: 0
> Векторы неизвестных: СЛАУ имеет бесконечное количество решений
```

```
----------------TEST #3----------------

> Исходная СЛАУ:
3.0 4.0 9.0 2.0 = 1.0
1.0 1.0 1.0 1.0 = 5.0
1.0 343.0 322.0 4.0 = 2.0
1.0 1.0 1.0 1.0 = 3.0

> Remove i = 1
3.0 4.0 9.0 2.0 = 1.0
0.0 -0.3333333333 -2.0 0.3333333333 = 4.6666666667
0.0 341.6666666667 319.0 3.3333333333 = 1.6666666667
0.0 -0.3333333333 -2.0 0.3333333333 = 2.6666666667

> Remove i = 2
3.0 4.0 9.0 2.0 = 1.0
0.0 -0.3333333333 -2.0 0.3333333333 = 4.6666666667
0.0 0.0 -1731.0 345.0 = 4785.0
0.0 0.0 0.0 0.0 = -2.0

> Remove i = 3
3.0 4.0 9.0 2.0 = 1.0
0.0 -0.3333333333 -2.0 0.3333333333 = 4.6666666667
0.0 0.0 -1731.0 345.0 = 4785.0
0.0 0.0 0.0 0.0 = -2.0

> Детерминант: 0
> Векторы неизвестных: СЛАУ не имеет решений

----------------TEST #4----------------

> Исходная СЛАУ:
1.0 3.0 8.0 3.0 8.0 = 9.0
1.0 12.0 3.0 343.0 343.0 = 12.0
737.0 745.0 38.0 282.0 3.0 = 3.0
23.0 77.0 32.0 838.0 33.0 = 82.0
9.0 74.0 73.0 7333.0 9.0 = 23.0

> Remove i = 1
1.0 3.0 8.0 3.0 8.0 = 9.0
0.0 9.0 -5.0 340.0 335.0 = 3.0
0.0 -1466.0 -5858.0 -1929.0 -5893.0 = -6630.0
0.0 8.0 -152.0 769.0 -151.0 = -125.0
0.0 47.0 1.0 7306.0 -63.0 = -58.0

> Remove i = 2
1.0 3.0 8.0 3.0 8.0 = 9.0
0.0 9.0 -5.0 340.0 335.0 = 3.0
0.0 0.0 -6672.4444444444 53453.2222222222 48674.7777777778 = -6141.3333333333
0.0 0.0 -147.5555555556 466.7777777778 -448.7777777778 = -127.6666666667
0.0 0.0 27.1111111111 5530.4444444444 -1812.4444444444 = -73.6666666667

> Remove i = 3
1.0 3.0 8.0 3.0 8.0 = 9.0
0.0 9.0 -5.0 340.0 335.0 = 3.0
0.0 0.0 -6672.4444444444 53453.2222222222 48674.7777777778 = -6141.3333333333
0.0 0.0 0.0 -715.2957436888 -1525.179977353 = 8.1438086991
0.0 0.0 -0.0 5747.6326517019 -1614.671751149 = -98.6197961766

> Remove i = 4
```

```
1.0 3.0 8.0 3.0 8.0 = 9.0
0.0 9.0 -5.0 340.0 335.0 = 3.0
0.0 0.0 -6672.4444444444 53453.2222222222 48674.7777777778 = -6141.3333333333
0.0 0.0 0.0 -715.2957436888 -1525.179977353 = 8.1438086991
0.0 0.0 -0.0 0.0 -13869.9861646646 = -33.1816592923

> Remove i = 5
1.0 3.0 8.0 3.0 8.0 = 9.0
0.0 9.0 -5.0 340.0 335.0 = 3.0
0.0 0.0 -6672.4444444444 53453.2222222222 48674.7777777778 = -6141.3333333333
0.0 0.0 0.0 -715.2957436888 -1525.179977353 = 8.1438086991
0.0 0.0 -0.0 0.0 -13869.9861646646 = -33.1816592923

> Треугольная матрица:
1.0 3.0 8.0 3.0 8.0 = 9.0
0.0 9.0 -5.0 340.0 335.0 = 3.0
0.0 0.0 -6672.4444444444 53453.2222222222 48674.7777777778 = -6141.3333333333
0.0 0.0 0.0 -715.2957436888 -1525.179977353 = 8.1438086991
0.0 0.0 -0.0 0.0 -13869.9861646646 = -33.1816592923

> Детерминант: -5.957844235039999E11

> Векторы неизвестных:
x1 = -1.3602050921
x2 = 1.3147562183
x3 = 0.8057820663
x4 = -0.0164862588
x5 = 0.0023923354

> Векторы невязок:
x1 = 0.0
x2 = -2.0E-10
x3 = -3.324E-7
x4 = 1.3E-9
x5 = -3.93E-7
```

6 Блок-схема алгоритма



7 Вывод

В ходе выполнения лабораторной работы, я вспомнил метод Гаусса для нахождения решений СЛАУ, а также написал реализацию этого алгоритма на языке Java