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.

```java
public class GaussExecutor {

  public void solve(BigDecimal[][] matrix) {
    var m = rightMethod(matrix);
    if (m != null) {
      var vectors = backMethod(m);
      printTriangleMatrix(m, "> Triangle matrix:");
      System.out.printf("> Determinate: %.5f\n\n", getDeterminate(m).doubleValue());
      printVectors(vectors, "> Vectors:");
      printVectors(getResidualVectors(matrix, vectors), "> Residual Vectors:");
    }
```

```java
}

private BigDecimal[][] rightMethod(BigDecimal[][] m) {
  for (int i = 0; i < m.length; i++) {
    if (m[i][i].compareTo(BigDecimal.ZERO) == 0) {
      int swapRow = findNonZeroDiagonalElementRow(m, i);
      if (swapRow != -1) {
        swapRows(m, i, swapRow);
      } else {
        int rankCoefficients = calculateRank(m);
        System.out.println("> Determinate: 0");
        if (rankCoefficients < m.length) {
          System.out.println("> Vectors: SLAU has infinite solves");
        } else {
          System.out.println("> Vectors: SLAU hasn't solves");
        }
        return null;
      }
    }
    for (int j = i + 1; j < m.length; j++) {
      BigDecimal factor = m[j][i].divide(m[i][i], 20, RoundingMode.HALF_UP);
      for (int k = 0; k <= m.length; k++) {
        m[j][k] = m[j][k].subtract(m[i][k].multiply(factor));
      }
    }
    printTriangleMatrix(m, "> Remove i = " + (i + 1));
  }
  return m;
}

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

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

private BigDecimal[] backMethod(BigDecimal[][] m) {
  BigDecimal[] solution = new BigDecimal[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] = solution[i].subtract(m[i][j].multiply(solution[j]));
    }
    BigDecimal res = solution[i].divide(m[i][i], 20, RoundingMode.HALF_UP);
    solution[i] = res;
  }
  return solution;
}

private void printVectors(BigDecimal[] vs, String message) {
  System.out.println(message);
  for (int i = 0; i < vs.length; i++) {
    System.out.printf("x%s = %.5f\n", i + 1, vs[i].doubleValue());
  }
  System.out.println();
}

private BigDecimal getDeterminate(BigDecimal[][] m) {
  BigDecimal determinate = BigDecimal.ONE;
  for (int i = 0; i < m.length; i++) {
    determinate = determinate.multiply(m[i][i]);
  }
  return determinate;
}

private void printTriangleMatrix(BigDecimal[][] m, String mes) {
  System.out.println(mes);
```

```java
    for (BigDecimal[] doubles : m) {
      for (int j = 0; j <= m.length; j++) {
        if (j == m.length) {
          System.out.print("= " + String.format("%.5f", doubles[j].doubleValue()));
        } else {
          System.out.print(String.format("%.5f", doubles[j].doubleValue()) + " ");
        }
      }
      System.out.println();
    }
    System.out.println();
  }

  private int calculateRank(BigDecimal[][] 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].compareTo(BigDecimal.ZERO) != 0)) {
          rank++;
          rowMarked[row] = true;
          found = true;

          for (int k = row + 1; k < rowCount; k++) {
            BigDecimal factor = matrix[k][col].divide(matrix[row][col], 20, RoundingMode.HALF_UP);
            for (int j = col; j < colCount; j++) {
              matrix[k][j] = matrix[k][j].subtract(matrix[row][j].multiply(factor));
            }
          }
        }
      }
    }

    return rank;
  }

  private BigDecimal[] getResidualVectors(BigDecimal[][] m, BigDecimal[] solutions) {
    var res = new BigDecimal[solutions.length];
    for (int i = 0; i < m.length; i++) {
      BigDecimal sum = BigDecimal.ZERO;
      for (int j = 0; j < m.length; j++) {
        sum = sum.add(m[i][j].multiply(solutions[j]));
      }
      res[i] = m[i][m.length].subtract(sum);
    }
    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
```

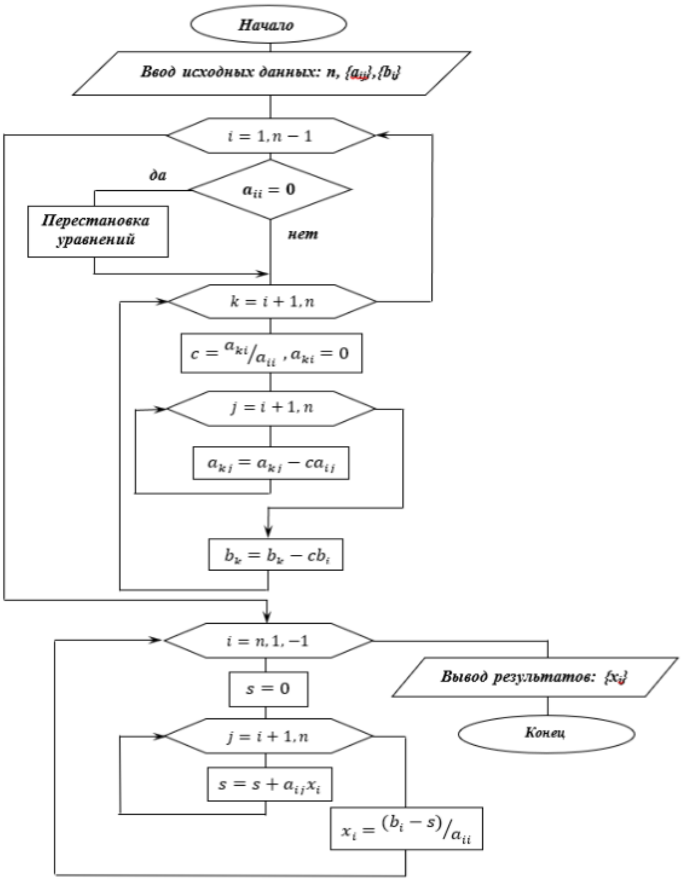
output:

```
----------------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:
1.000 3.000 8.000 3.000 8.000 = 9.000
1.000 12.000 3.000 343.000 343.000 = 12.000
737.000 745.000 38.000 282.000 3.000 = 3.000
23.000 77.000 32.000 838.000 33.000 = 82.000
9.000 74.000 73.000 7333.000 9.000 = 23.000

> Remove i = 1
1,00000 3,00000 8,00000 3,00000 8,00000 = 9,00000
0,00000 9,00000 -5,00000 340,00000 335,00000 = 3,00000
0,00000 -1466,00000 -5858,00000 -1929,00000 -5893,00000 = -6630,00000
0,00000 8,00000 -152,00000 769,00000 -151,00000 = -125,00000
0,00000 47,00000 1,00000 7306,00000 -63,00000 = -58,00000

> Remove i = 2
1,00000 3,00000 8,00000 3,00000 8,00000 = 9,00000
0,00000 9,00000 -5,00000 340,00000 335,00000 = 3,00000
0,00000 0,00000 -6672,44444 53453,22222 48674,77778 = -6141,33333
0,00000 -0,00000 -147,55556 466,77778 -448,77778 = -127,66667
0,00000 0,00000 27,11111 5530,44444 -1812,44444 = -73,66667

> Remove i = 3
1,00000 3,00000 8,00000 3,00000 8,00000 = 9,00000
0,00000 9,00000 -5,00000 340,00000 335,00000 = 3,00000
0,00000 0,00000 -6672,44444 53453,22222 48674,77778 = -6141,33333
0,00000 -0,00000 -0,00000 -715,29574 -1525,17998 = 8,14381
0,00000 0,00000 0,00000 5747,63265 -1614,67175 = -98,61980

> Remove i = 4
1,00000 3,00000 8,00000 3,00000 8,00000 = 9,00000
0,00000 9,00000 -5,00000 340,00000 335,00000 = 3,00000
0,00000 0,00000 -6672,44444 53453,22222 48674,77778 = -6141,33333
0,00000 -0,00000 -0,00000 -715,29574 -1525,17998 = 8,14381
0,00000 -0,00000 0,00000 -0,00000 -13869,98616 = -33,18166
```

```
> Remove i = 5
1,00000 3,00000 8,00000 3,00000 8,00000 = 9,00000
0,00000 9,00000 -5,00000 340,00000 335,00000 = 3,00000
0,00000 0,00000 -6672,44444 53453,22222 48674,77778 = -6141,33333
0,00000 -0,00000 -0,00000 -715,29574 -1525,17998 = 8,14381
0,00000 -0,00000 0,00000 -0,00000 -13869,98616 = -33,18166

> Треугольная матрица:
1,00000 3,00000 8,00000 3,00000 8,00000 = 9,00000
0,00000 9,00000 -5,00000 340,00000 335,00000 = 3,00000
0,00000 0,00000 -6672,44444 53453,22222 48674,77778 = -6141,33333
0,00000 -0,00000 -0,00000 -715,29574 -1525,17998 = 8,14381
0,00000 -0,00000 0,00000 -0,00000 -13869,98616 = -33,18166

> Детерминант: -595784423504,00000

> Векторы неизвестных:
x1 = -1,36021
x2 = 1,31476
x3 = 0,80578
x4 = -0,01649
x5 = 0,00239

> Векторы невязок:
x1 = 0,00000
x2 = 0,00000
x3 = -0,00000
x4 = 0,00000
x5 = -0,00000
```

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. Недостаток этого метода в том, что он требует больших затрат на память, а преимущество – точность вычислений.