Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Лабораторная работа №1
Вариант: метод Гаусса

Выполнил:
Воробьев Кирилл
P3231

Преподаватель:
Перл Ольга Вячеславовна

Санкт-Петербург, 2022г

Описание метода

Решение СЛАУ методом Гаусса включает в себя два этапа: прямой и обратный ход. На первом этапе исходная матрица приводится к треугольной форме с помощью элементарных преобразований (либо же устанавливается , что система несовместна). Для этого мы обнуляем ненулевые элементы , лежащие под главной диагональю путем вычитания из столбца правее лежащего столбца, умноженного на найденный коэффициент. Если на какой-то из итераций среди элементов столбца не нашелся ненулевой элемент, то мы переходим к следующему столбцу. На втором этапе мы выражаем решение системы в численном виде. Эта процедура начинается с последнего уравнения, из которого выражается соответствующая переменная и подставляется в предыдущие уравнения.

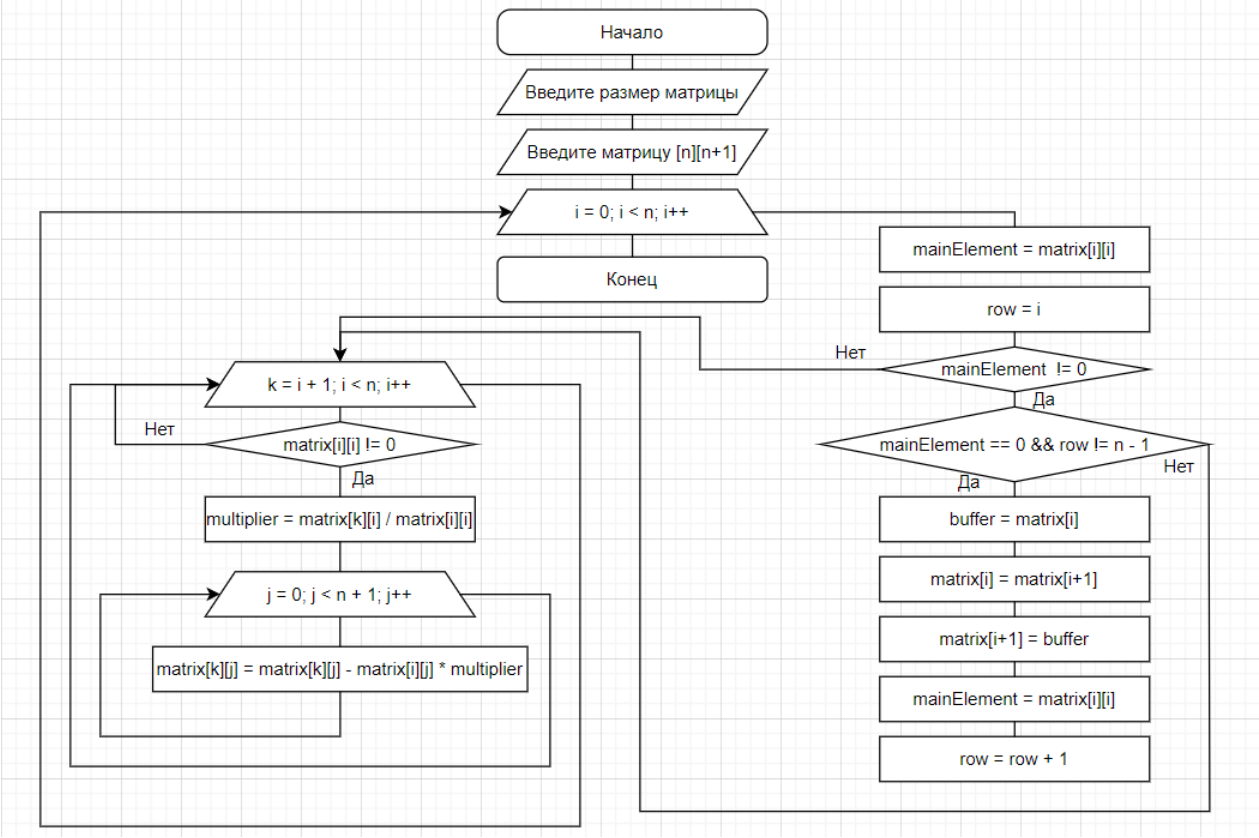
Расчетные формулы метода

Преобразованная система:

$$\begin{cases} \alpha_{1j_1}x_{j_1} + \alpha_{1j_2}x_{j_2} + \ldots + \alpha_{1j_r}x_{j_r} + \ldots + \alpha_{1j_n}x_{j_n} = \beta_1 \\ \alpha_{2j_2}x_{j_2} + \ldots + \alpha_{2j_r}x_{j_r} + \ldots + \alpha_{2j_n}x_{j_n} = \beta_2 \\ \ldots \\ \alpha_{rj_r}x_{j_r} + \ldots + \alpha_{rj_n}x_{j_n} = \beta_r \\ 0 = \beta_{r+1} \\ \ldots \\ 0 = \beta_m \end{cases},$$

где $\alpha_{1j_1}, \ldots, \alpha_{rj_r} \neq 0.$

Блок-схема метода



Метод приведения матрицы к треугольной

```
public Matrix gaussMethod(Matrix matrix) {
    Matrix triangleMatrix = Util.cloneMatrix(matrix);
    for (int i = 0; i < triangleMatrix.getSize(); i++) {
        triangleMatrix = swapRows(triangleMatrix, i);
        triangleMatrix = nullingColumn(triangleMatrix, i);
    }
    Util.validateMatrix(triangleMatrix);
    return triangleMatrix;
}
```

Метод для замены строк матрицы (если число на гл. диагонали равняется 0)

```
public Matrix swapRows(Matrix matrix, int numberOfRow) {
    Matrix cloneMatrix = Util.cloneMatrix(matrix);
    double[][] cloneMatrixArray = cloneMatrix.getMatrix();
    int row = numberOfRow;
    double[] bufferRow;
    double element = cloneMatrixArray[numberOfRow][numberOfRow];
    if (element == 0) {
        while (element == 0 && row != matrix.getSize() - 1) {
            bufferRow = cloneMatrixArray[numberOfRow];
            cloneMatrixArray[numberOfRow] = cloneMatrixArray[row + 1];
            cloneMatrixArray[row + 1] = bufferRow;
            element = cloneMatrixArray[numberOfRow][numberOfRow];
            ++row;
        }
    }
    return cloneMatrix;
}
```

Метод для обнуления чисел матрицы в столбце под главной диагональю

```
private Matrix nullingColumn(Matrix matrix, int numberOfColumn) {
    Matrix cloneMatrix = Util.cloneMatrix(matrix);
    double[][] cloneMatrixArray = cloneMatrix.getMatrix();
    for (int i = numberOfColumn + 1; i < cloneMatrix.getSize(); i++) {
        if (cloneMatrixArray[numberOfColumn][numberOfColumn] == 0) break;
        double multiplier = cloneMatrixArray[i][numberOfColumn] / cloneMatrixArray[numberOfColumn][numberOfColumn];
        for (int j = 0; j < cloneMatrix.getSize() + 1; j++) {
            cloneMatrixArray[i][j] = cloneMatrixArray[i][j] - cloneMatrixArray[numberOfColumn][j] * multiplier;
        }
    }
    return cloneMatrix;
}
```

Метод для поиска неизвестных

```java
public double[] getResults(Matrix matrix) {
    double[] results = new double[matrix.getSize()];
    for (int i = 0; i < matrix.getSize(); i++) {
        results[matrix.getSize() - i - 1] = getResult(matrix, results, matrix.getSize() - i - 1);
    }
    return results;
}

public double getResult(Matrix matrix, double[] results, int i) {
    double result = matrix.getMatrix()[i][matrix.getSize()];
    for (int k = 0; k < matrix.getSize(); k++) {
        if (k != i) {
            result = result - matrix.getMatrix()[i][k] * results[k];
        }
    }
    result = result / matrix.getMatrix()[i][i];
    return result;
}
```

Метод для вычисления невязки

```java
public double[] getResiduals(Matrix matrix, double[] result) {
    double[] residuals = new double[matrix.getSize()];
    for (int i = 0; i < matrix.getSize(); i++) {
        residuals[i] = matrix.getMatrix()[i][matrix.getSize()];
        for (int j = 0; j < matrix.getSize(); j++) {
            residuals[i] = residuals[i] - matrix.getMatrix()[i][j] * result[j];
        }
    }
    return residuals;
}
```

Примеры

1.
Матрица: 5
3,1601 7,3882 7,9593 9,8979 7,8327 8,5208
2,2464 0,2789 5,4173 9,7930 4,4924 8,2840
8,1368 2,9691 7,0697 4,8244 2,0869 6,9279
0,5591 9,2576 4,9432 5,5291 9,7641 3,9967
0,2843 4,0477 2,7607 9,7874 4,0686 7,8918

Определитель: -3752,1125

Ответ:
x[1]: 0.4796
x[2]: 0.038
x[3]: -0.1452
x[4]: 0.8414
x[5]: -0.0571

Невязки:
residual[1]: -2.220446049250313E-16
residual[2]: -1.3877787807814457E-15
residual[3]: 4.85722573273506E-16
residual[4]: 2.220446049250313E-16
residual[5]: 5.828670879282072E-16

2.
Матрица: 3
1 2 3 5
5 2 8 6
8 1 4 7

Определитель: 55

Ответ:
x[1]: 0.7273
x[2]: 2.7091
x[3]: -0.3818

Невязки:
residual[1]: 0.0
residual[2]: -8.881784197001252E-16
residual[3]: -1.7763568394002505E-15

3.

Матрица: 4
9,2766 9,5721 9,0137 7,9403 1,0074
6,8025 3,1972 7,7958 3,6074 7,2185
5,8551 3,4218 3,2577 8,0695 3,6753
6,4790 5,4668 2,6689 0,5473 9,9221

Определитель: -18045,6915

Ответ:
x[1]: 3.1585
x[2]: -1.4345
x[3]: -0.8283
x[4]: -0.8936

Невязки:
residual[1]: -8.881784197001252E-16
residual[2]: -2.6645352591003757E-15
residual[3]: 8.881784197001252E-16
residual[4]: 1.4432899320127035E-15

Вывод

Прямые методы используют конечные формулы для вычисления неизвестных. Они дают решение за конечное число арифметических операций. Эти методы сравнительно просты и наиболее универсальны.

Недостатки прямых методов:
- Требование хранить в оперативной памяти всю матрицу, следовательно при больших n расходуется память.
- Накапливается погрешность в процессе решения, поскольку вычисления на любом этапе используют результаты предыдущих вычислений.