Санкт-Петербургский Национальный Исследовательский
Университет ИТМО
Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №1

Вариант: метод Гаусса

Выполнил:

студент группы P3231

Нестеров Иван Алексеевич

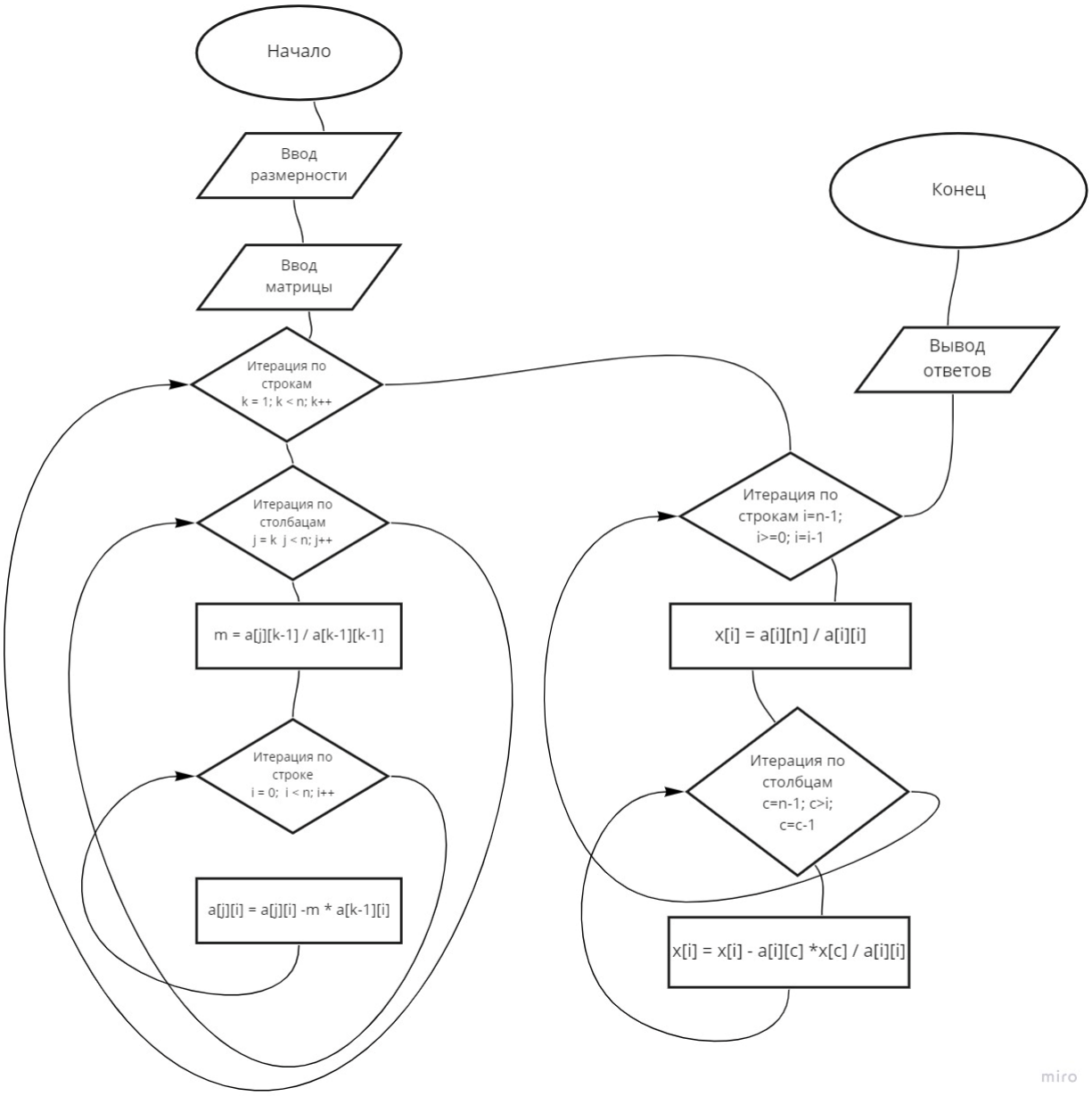
Преподаватель:

Перл Ольга Вячеславовна

г. Санкт-Петербург

2022 г.

Блок схема:



Описание метода:

Реализованная программа состоит из трех блоков:

1. Пользовательский ввод данных. Есть возможность выбора из трех вариантов:
    a) Ввод количества переменных (размерности будущей матрицы) и всех последующих коэффициентов вручную. Выводятся многочисленные подсказки для пользователя с целью упрощения использования программы;
    b) Использование случайных данных. В таком случае программа автоматически сгенерирует случайную систему линейных уравнений;

c) Чтение готовой матрицы из файла – предварительно нужно указать лишь ее размерность. Данные должны быть записаны в виде таблицы чисел, где числа между собой в пределах строки отделены одним пробелом, а каждая строка в свою очередь завершается переводом каретки.

На всех этапах отслеживаются и предупреждаются возможности некорректного пользовательского ввода.

2. Непосредственно математическая часть программы – решение системы линейных уравнений методом Гаусса.
   a) Рекурсивно вычисляется определитель полученной матрицы. В случае если он равен нулю – решения системы не существует, пользователю демонстрируется этот вывод, и программа завершает работу. Если же определитель не равен нулю, то алгоритм переходит к следующему этапу.
   b) Выбирается строка, первый коэффициент которой не равен нулю. Далее все коэффициенты первого столбца, находящиеся ниже выбранного, обнуляются путём вычитания из каждой строки первой строки, помноженной на отношение первого элемента этих строк к первому элементу первой строки. Аналогичное действие происходит для каждого элемента строки до тех пор, пока матрица не будет приведена к треугольному виду, то есть до тех пор, пока ниже главной диагонали матрицы не будут находиться одни лишь нули. Этот этап называется прямым ходом и обозначен на блок-схеме в левой части.
   c) Далее, начиная с последней строки, значения переменных вычисляются одно за другим, путем подстановки известного значения в уравнений (для последней неизвестной значение становится известно по определению треугольной матрицы). Данный этап называется обратным ходом и обозначен на блок схеме слева.
   d) Необходимо также помнить об особенностях представления чисел с плавающей точкой в памяти компьютера. В связи с этим параллельно с обратным ходом считаются невязки для каждой переменной, путем сравнения данной нам правой части уравнения и реально получившейся левой части, когда в нее подставляются вычисленные значения.
3. Вывод невязок для каждой переменной и непосредственно значения каждой переменной. Завершение работы программы.
4. Кроме того, программу можно завершить в любой момент исполнения, нажав сочетание клавиш ctrl + D (или command + D для macOS), о чем программа предупреждает пользователя при запуске.

Преимущества метода:

1. Удобство и понятность в реализации;
2. Решение за конечное число операций;
3. Возможность однозначно определить совместность системы и найти решение;
4. Возможность параллельно посчитать еще и ранг матрицы.

Недостатки метода:

1. Неэффективность для матриц большого размера. Существуют асимптотически более быстрые алгоритмы. Применение различных модификаций метода позволяет посчитать ответ быстрее и сократить код.

Листинги математической части программы:

```java
public static double[] gaussSolution(double[][] matrix) {
    try {
        int dimension = matrix.length;
        double[][] copyOfMatrix = new double[dimension][dimension + 1];
        for (int i = 0; i < dimension; i++) {
            for (int j = 0; j < dimension + 1; j++) {
                copyOfMatrix[i][j] = matrix[i][j];
            }
        }
        for (int k = 0; k < dimension; k++) { // k is number of line
            for (int i = 0; i < dimension + 1; i++) // i is number of column
                copyOfMatrix[k][i] = copyOfMatrix[k][i] / matrix[k][k]; // divide k-number line to first item != 0
            for (int i = k + 1; i < dimension; i++) { // i lines which are follows for the k line
                double coefficient = copyOfMatrix[i][k] / copyOfMatrix[k][k];
                for (int j = 0; j < dimension + 1; j++) { // j is column which are follows for the k column
                    copyOfMatrix[i][j] = copyOfMatrix[i][j] - copyOfMatrix[k][j] * coefficient; // zeroes
                }
            }
            for (int i = 0; i < dimension; i++) { // update initial matrix
                for (int j = 0; j < dimension + 1; j++) {
                    if (new Double(copyOfMatrix[i][j]).equals((0.0))
                            || (copyOfMatrix[i][j] * (-1) == 0)) {
                        matrix[i][j] = 0;
                    } else matrix[i][j] = copyOfMatrix[i][j];
                }
            }
        }
```

```java
        System.out.println("Your augmented matrix, reduced to triangular form:");
        printMatrix(matrix);
        double determinant = 1;
        for (int i = 0; i < matrix.length; i++) {
            determinant *= matrix[i][i];
        }
        if (determinant == 0) {
            throw new NoSolutionException();
        }
        //Обратный ход (Зануление верхнего правого угла)
        for (int k = dimension - 1; k > -1; k--) { // k is number of line
            for (int i = dimension; i > -1; i--) { // i is number of column
                copyOfMatrix[k][i] = copyOfMatrix[k][i] / matrix[k][k];
            }
            for (int i = k - 1; i > -1; i--) { // i lines which are follows for the k line
                double coefficient = copyOfMatrix[i][k] / copyOfMatrix[k][k];
                for (int j = dimension; j > -1; j--) // j is column which are follows for the k column
                    copyOfMatrix[i][j] = copyOfMatrix[i][j] - copyOfMatrix[k][j] * coefficient;
            }
            // save clean answers
            double[] answers = new double[dimension];
            for (int i = 0; i < dimension; i++) {
                answers[i] = copyOfMatrix[i][dimension];
            }
            double[] residuals = new double[answers.length];
            for (int i = matrix.length - 1;  i >= 0; i--) {
                double b = matrix[i][matrix[0].length - 1];
```

```
            double b = matrix[i][matrix[0].length - 1];
            double sum = 0;
            for (int j = 0; j < matrix[0].length - 1; j++) {
                sum += answers[j] * matrix[i][j];
            }
            residuals[i] = b - sum;
        }
        System.out.println("\u001B[34m" + "\nResiduals:");
        for (int index = 0; index < residuals.length; index++) {
            System.out.println("For x(" + index + 1 + "): " + residuals[index]);
        }
        System.out.println("\u001B[0m");
        return answers;
    }
} catch (NoSolutionException noSolutionException) {
    System.err.println(noSolutionException.getMessage());
    System.exit( status: 0);
}
return null;
```

Примеры работы программы:

```
Gaussian linear equation solver launched!
Type ctrl + D (or cmd + D for macOS) to exit.
Choose type of data entering. Print only a number:
1. Enter data manually;
2. Generate random data;
3. Read data from file;
1
Your system of linear equations looks like this:
a(11) * x(1) + a(12) * x(2) + ... + a(1n) * x(n) = b(1)
a(21) * x(1) + a(22) * x(2) + ... + a(2n) * x(n) = b(2)
.....   ....   .....   ....   ...   .....   ....   ....
a(31) * x(1) + a(32) * x(2) + ... + a(3n) * x(n) = b(n)
Enter number of unknown variables: 2
You need to enter a(i j) values now. Let's try!
Enter an coefficient a(1)(1)
Enter value: 1
Enter an coefficient a(1)(2)
Enter value: 1
Enter an coefficient b(1)
Enter value: 10
Enter an coefficient a(2)(1)
Enter value: 1
Enter an coefficient a(2)(2)
Enter value: 1
Enter an coefficient b(2)
Enter value: 20
```

```
Your augmented matrix:
1.0000      1.0000      10.0000
1.0000      1.0000      20.0000


There are no solutions of this system! Determinant of this matrix is zero.

Process finished with exit code 0
```

```
Gaussian linear equation solver launched!
Type ctrl + D (or cmd + D for macOS) to exit.
Choose type of data entering. Print only a number:
1. Enter data manually;
2. Generate random data;
3. Read data from file;
2
Your system of linear equations looks like this:
a(11) * x(1) + a(12) * x(2) + ... + a(1n) * x(n) = b(1)
a(21) * x(1) + a(22) * x(2) + ... + a(2n) * x(n) = b(2)
.....   ....   .....   ....   ...   .....   ....   ....
a(31) * x(1) + a(32) * x(2) + ... + a(3n) * x(n) = b(n)
Enter number of unknown variables: 11
Random matrix generating...

Your augmented matrix:
72.5965     66.5520     74.2944     49.6321     94.5665     90.7456     56.5875     80.5868     88.7066     47.1856     81.5715     6.5285
67.7314     74.4816     3.6209      38.7698     88.2968     31.1370     96.2368     87.4803     47.1017     74.5469     99.3144     96.5754
78.9115     63.9977     34.2270     12.9981     10.9764     52.4208     43.2018     90.3476     51.8479     52.8369     84.5556     62.0626
44.2623     39.7512     68.5851     57.7470     91.5399     73.1964     50.2525     77.7287     74.8632     13.6876     4.2783      85.5705
14.8304     30.0470     95.8541     26.0084     94.8398     88.8697     25.9085     26.6687     56.1485     54.3453     73.7065     95.9208
94.8569     20.2145     81.5390     86.3518     39.6060     41.8345     26.3502     53.6014     65.4347     35.6397     1.3814      3.6753
4.0310      65.7276     87.2500     88.3610     43.5175     42.8668     26.3604     77.9944     71.9517     19.9138     27.1373     13.5846
7.8770      79.4422     15.2987     16.1795     78.5482     74.3302     18.2286     10.3607     51.8658     0.1181      15.4868     0.8831
1.5860      49.1377     13.5875     26.7324     87.0361     20.7345     6.4031      5.2470      58.3488     39.3426     63.8952     70.4602
56.1393     62.7404     51.3507     4.4434      60.5844     46.5046     18.8944     58.1874     59.0227     67.4986     16.7758     59.4374
43.4807     99.1715     85.7737     8.8988      19.9385     74.5106     44.0339     73.3645     9.6865      68.6811     41.7567     94.5631
```

```
Your augmented matrix, reduced to triangular form:
1.0000      0.9167      1.0234      0.6837      1.3026      1.2500      0.7795      1.1101      1.2219      0.6500      1.1236      0.0899
0.0000      1.0000      -5.3024     -0.6083     0.0055      -4.3203     3.5063      0.9923      -2.8782     2.4636      1.8733      7.3032
0.0000      0.0000      1.0000      0.5071      1.0110      0.9063      -0.1206     -0.1215     0.7556      -0.2435     -0.1269     -1.2769
0.0000      0.0000      0.0000      1.0000      0.8490      -0.1630     1.2030      1.8230      0.2365      -0.4854     -2.3862     6.4249
0.0000      0.0000      0.0000      0.0000      1.0000      0.4642      -1.0954     -2.7793     0.6245      -0.3703     2.1333      -12.9423
0.0000      0.0000      0.0000      0.0000      0.0000      1.0000      -1.4147     -1.0090     1.3365      -2.5952     -0.8523     -11.7211
0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      1.0000      1.9619      -0.2222     0.1136      -1.5752     8.9792
0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      1.0000      2.2388      -1.6478     5.1783      -23.2144
0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      1.0000      -0.7406     3.3232      -14.8512
0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      1.0000      1.9451      6.4649
0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000      1.0000      0.4637

Residuals:
For x(01): 59.78424712725966
For x(11): -120.24475301811809
For x(21): 32.330046893231966
For x(31): 51.5734077658222
For x(41): -43.00393428525282
For x(51): 24.235159775031224
For x(61): 45.97974064142547
For x(71): 45.86570546505622
For x(81): 4.119852390031845
For x(91): 0.0
For x(101): 0.0
```

```
Values of variables for your system of linear equations (residuals are included):
x(1) = -0.4311027068109853
x(2) = 6.434589296288595
x(3) = -1.2180126600088791
x(4) = 7.531347683282803
x(5) = -13.931453916780846
x(6) = -11.3259190701242
x(7) = 9.709650488094123
x(8) = -25.615628772785804
x(9) = -16.39214637672079
x(10) = 5.56290168610779
x(11) = 0.46370397141842506

Process finished with exit code 0
```

Вывод: проделанная работа позволила мне изучить алгоритм Гаусса подробнее (описал свое понимание выше), столкнуться с рядом проблем по ходу написания алгоритма, что привело к еще более подробному изучения метода. Опробовал различные синтаксические конструкции языка java для написания алгоритма. Постарался реализовать удобный и дружелюбный к пользователю функционал. Реализовал удобное визуальное восприятие информации.