НИУ ИТМО
Факультет программной инженерии и компьютерной техники

Лабораторная работа № 1

по дисциплине
«Вычислительная математика**»**

Вариант «Метод Гаусса с выбором главного элемента»

Выполнила
Рыжова Евгения Романовна
Группа P32312
Преподаватель
Перл Ольга Вячеславовна

г. Санкт-Петербург
2023

Оглавление

Описание метода

Пусть задана матрица A и столбец её свободных членов B.

Сначала матрица приводится к треугольному виду (в моей реализации к верхне-треугольному).

- Для этого для шага с номером N в N-ом выбирается максимальный элемент в строках с N-ой по последнюю. Если этот максимальный элемент находится не в N-ой строке, а, допустим, в K-ой, K-ая и N-ая строка меняются местами. Каждый такой обмен меняет знак определителя матрицы.

- Затем для расширенной матрицы A|B из всех строк ниже получившейся N-ой вычитается N-ая строчка, помноженная на множитель, подобранный для каждой отдельной строки так, чтобы в N-ом столбце данной строки получился 0.

- Таким образом, после N-ого шага во всех строках ниже N-ой элементы N-ого столбца будут равны 0.

Выполняем n шагов, где n — число строк/столбцов в матрице A.

В результате мы получим матрицу, эквивалентную исходной, определитель которой может отличаться от определителя исходной только знаком. Если была дана матрица, определитель которой не равен нулю, после данного алгоритма мы получим треугольную матрицу. Если нет — мы получим матрицу, ниже главной диагонали которой находятся только нули. В любом случае, для вычисления определителя данной матрицы достаточно посчитать произведение элементов на главной диагонали и помножить на количество обменов строк, которые мы произвели.

Если определитель оказался равным нулю, система Ax = B имеет либо более одного решения, либо ни одного решения.

Иначе последовательно вычислим единственное решение матрицы из полученной верхне-треугольной матрицы).

- Для шага с номером N вычислим переменную x_N. Считаем, что все переменные с большими чем N номерами уже вычислены.

- Для этого рассмотрим N-ую строчку матрицы A|B. Так как матрица верхне-треугольная, все элементы столбцов до N-ого равны 0.

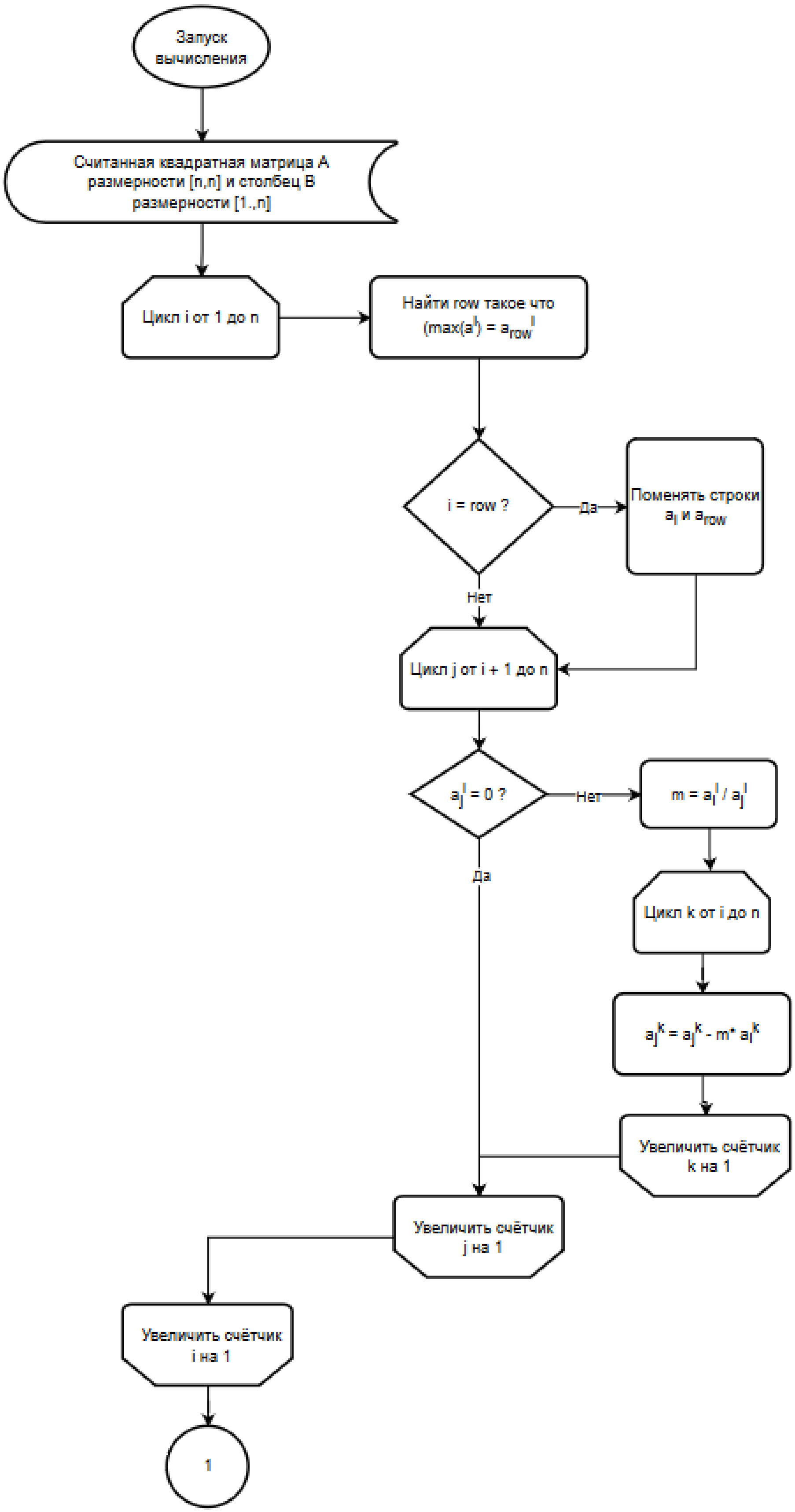
- Вычислим следующую величину: вычтем из последнего элемента рассматриваемой строчки (то есть N-ого свободного члена) M-ый элемент рассматриваемой строчки, умноженный на значение M-ой переменной, для всех переменных с номерами, большими N

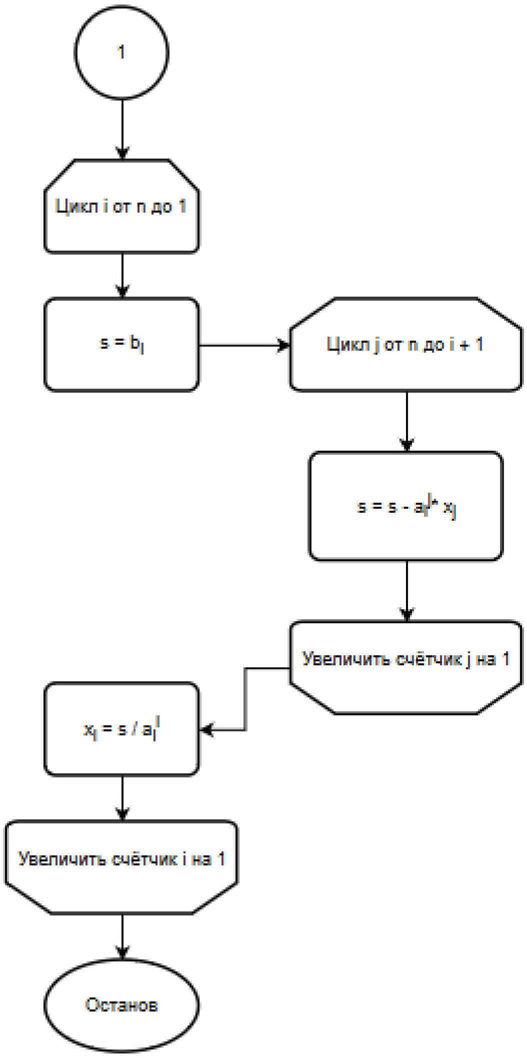
- Иными словами, перенесём все переменные с большими чем N номерами в правую часть уравнения и, пользуясь тем, что значения этих переменных уже вычислены, вычислим правую часть.

- Таким образом мы получим уравнение вида $c * x_N = d$, из которого легко вычислить значение x_N

В результате данного алгоритма мы последовательно вычислим все значения переменных, начиная с x_n и заканчивая x_1.

Блок-схема


Запуск вычисления
Считанная квадратная матрица A размерности [n,n] и столбец B размерности [1.,n]
Цикл i от 1 до n
Найти row такое что $(max(a^l) = a_{row}^l$
i = row ?
Да
Поменять строки a_l и a_{row}
Нет
Цикл j от i + 1 до n
$a_j^l = 0$?
Нет
$m = a_l^l / a_j^l$
Цикл k от i до n
$a_j^k = a_j^k - m^* a_l^k$
Увеличить счётчик k на 1
Да
Увеличить счётчик j на 1
Увеличить счётчик i на 1
1


1
Цикл i от n до 1
s = b_i
Цикл j от n до i + 1
s = s - a_i^j * x_j
Увеличить счётчик j на 1
x_i = s / a_i^i
Увеличить счётчик i на 1
Останов

Примеры и результаты работы

Пример 1. Верхне-треугольная целочисленная матрица

```
Original matrix:

1.0  2.0  3.0  4.0  5.0  |  10.0

0.0  2.0  6.0  7.0  8.0  |  11.0

0.0  0.0  3.0  9.0  10.0  |  12.0

0.0  0.0  0.0  4.0  11.0  |  2.0

0.0  0.0  0.0  0.0  5.0  |  15.0


Transformed matrix:

1.0  2.0  3.0  4.0  5.0  |  10.0

0.0  2.0  6.0  7.0  8.0  |  11.0

0.0  0.0  3.0  9.0  10.0  |  12.0

0.0  0.0  0.0  4.0  11.0  |  2.0

0.0  0.0  0.0  0.0  5.0  |  15.0


Matrix determinant = 120.0


Variables:

[36.5, -31.125, 17.25, -7.75, 3.0]


Errors:

[0.0, 0.0, 0.0, 0.0, 0.0]


Execution time: 56 ms
```

Пример 2. Не верхне-треугольная целочисленная матрица

```
Original matrix:

1.0  2.0  3.0  4.0  5.0  |  10.0

1.0  2.0  6.0  7.0  8.0  |  11.0

7.0  16.0  3.0  9.0  10.0  |  12.0

14.0  19.0  27.0  4.0  11.0  |  2.0
```

```
21.0  18.0  42.0  3.0  5.0  |  15.0


Transformed matrix:

21.0  18.0  42.0  3.0  5.0  |  10.0

0.0  10.0  -11.0  8.0  8.333333333333334  |  10.523809523809524

0.0  0.0  6.699999999999999  -3.5999999999999996  1.833333333333334  |  7.463945578231294

0.0  0.0  0.0  8.767590618336886  5.371002132196161  |  -17.88990470388582

0.0  0.0  0.0  0.0  0.6461575875486378  |  19.286014220863443


Matrix determinant = -7970.999999999994


Variables:

[55.64831903404948, -27.33178358283473, -17.973913685408935, -20.324790375314603,
29.847230137821043]


Errors:

[2.8421709430404007E-14, 5.329070518200751E-15, -1.687538997430238E-14, -1.4210854715202004E-14,
0.0]


Execution time: 40 ms
```

Пример 3. Не верхне-треугольная рациональная матрица

```
Original matrix:

5.12  3.71  4.22  7.23  |  11.04

8.32  7.21  11.33  8.19  |  23.387

2.11  12.86  9.351  4.84  |  2.14

43.43  6.2  8.0  2.53  |  6.3


Transformed matrix:

43.43  6.2  8.0  2.53  |  11.04

0.0  12.5587796454064  8.962328574717938  4.717082661754548  |  21.27203799217131

0.0  0.0  5.499757663067411  5.443361847342581  |  -8.596841919102383

0.0  0.0  0.0  4.6736824076572905  |  1.7515582281257067
```

```
Matrix determinant = 14019.741969567998


Variables:

[0.1698877923706499, 2.933237448419653, -1.9340585416089173, 0.37477048617081477]


Errors:

[0.0, -3.552713678800501E-15, -1.7763568394002505E-15, -2.220446049250313E-16]


Execution time: 47 ms
```

Пример 4. Не-верхнетреугольная разреженная матрица.

```
Original matrix:

0.001  0.002  0.003  0.004  |  0.234

0.005  0.006  0.007  0.008  |  0.021

0.017  0.013  0.018  0.029  |  0.123

0.061  0.067  0.063  0.064  |  0.012


Transformed matrix:

0.061  0.067  0.063  0.064  |  0.234

0.0  9.01639344262295E-4  0.0019672131147540984  0.0029508196721311475  |  0.0018196721311475386

0.0  0.0  0.012818181818181828  0.029727272727272745  |  0.06923427719821161

0.0  0.0  0.0  -5.957446808510648E-4  |  0.0032101151028950123


Matrix determinant = -4.20000000000001E-10


Variables:

[12.309484777517556, -19.396721311475382, 17.897775175644, -5.3884074941451905]


Errors:

[-2.7755575615628914E-17, 0.0, 4.163336342344337E-17, 0.0]


Execution time: 377 ms
```

Пример 5. Работа с матрицей, у которой более 1 решения / нет решения.

```
Original matrix:

1.0  2.0  3.0  4.0  5.0  |  7.0

0.0  2.0  9.0  11.0  2.0  |  0.321

0.1  32.0  1.0  0.0  0.0  |  0.0

0.5  1.0  1.5  2.0  2.5  |  8.0

2.0  6.0  18.0  9.11  9.23  |  1.21


Transformed matrix:

2.0  6.0  18.0  9.11  9.23  |  7.0

0.0  31.7  0.09999999999999998  -0.4555  -0.4615  |  0.321

0.0  0.0  8.993690851735016  11.028738170347003  2.0291167192429023  |  -0.3702523659305994

0.0  0.0  0.0  6.784412486846721  1.7234233602244824  |  6.131623991581901

0.0  0.0  0.0  0.0  0.0  |  -5.5925640126271485


Matrix determinant = -0.0


This equation has either multiple solutions or no solution and can't be solved by this program.

Execution time: 32 ms
```

Листинг реализации численного метода

```java
void toUpperTriangularStep(int step) {
    int rowIdx = matrix.getMaxColElementRow(step, step);
    if (rowIdx != step) {
        matrix.swapRows(step, rowIdx);
        detMultiplier *= -1;
    }
    for (int j = step + 1; j < matrix.getRows(); j++) {
        matrix.subRowWithMultiplier(j, step, step);
    }
}

void transformToUpperTriangular() {
    for (int i = 0; i < matrix.getCols(); i++) {
        toUpperTriangularStep(i);
        System.out.println(matrix);
    }
}

double calcDet() {
    if (!matrix.isUpperTriangular()) {
        transformToUpperTriangular();
        if (!matrix.isUpperTriangular()) {
            throw new NonSolvableEquationException();
        }
    }
    return matrix.calcDetForUpperTriangular() * detMultiplier;
}

void findVariableStep(int step) {
    double sum = matrix.getFreeValues()[step];
    for (int i = matrix.getCols() - 1; i > step; i--) {
        sum -= matrix.getVariables()[i] * matrix.getValues()[step][i];
    }
    double value = sum / matrix.getValues()[step][step];
    matrix.setVariable(step, value);
}

void solve() {
    if (matrix.isASolvableEquation()) {
        for (int i = matrix.getCols() - 1; i >= 0; i--) {
            findVariableStep(i);
        }
    } else {
        throw new NonSolvableEquationException();
    }
}
```

Вывод

Метод Гаусса (как с выбором данного элемента, так и без) в любом случае позволяет найти решение (если для данного уравнения существует единственное решение).

Так как мы вычисляем значения элементов матрицы и переменных последовательно, ошибка накапливается.

Метод работает за $O(N^3)$ по времени.

Принципиальное отличие от метода Гаусса в том, что при преобразовании матрицы к треугольному виду выбирается не произвольная строка с ненулевым значением элемента данного столбца, а строка с максимальным значением в данном столбце из ещё не преобразованных. Так как в итоге происходит деление на максимальный элемент в строке, получаются меньшие числа. Поэтому ошибка в методе Гаусса с выбором главного элемента меньше, чем без главного элемента.

Если сравнивать с итерационными методами, то методы Гаусса могут проигрывать по точности (если уравнение решается итерационным методом, то решение можно сделать сколь угодно точным), но выигрывать тем, что всегда известно, что решение (если оно существует и единственно) точно найдётся за конечное число шагов.