Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа по вычислительной математике №1

Метод простых итераций

Преподаватель: Малышева Татьяна Алексеевна

Выполнила: Голованова Дарья Владимировна

Группа: Р3222

Санкт-Петербург,
2022г

Решение системы линейных алгебраических уравнений
Метод простых итераций

Описание метода:

Метод простых итераций – один из классических методов решения систем линейных алгебраических уравнений. Он является итерационным. (В итерационных методах сначала задается некоторое начальное приближение. Далее с помощью определенного алгоритма проводится один цикл вычислений - итерация. В результате итерации находят новое приближение. Итерации проводятся до получения решения с требуемой точностью.) Суть метода заключается в последовательном приближении вектора корней к решению, причем каждое следующее приближение получается из предыдущего и является более точным, чем предыдущее. Точное решение СЛАУ – предел последовательности векторов $x^{(0)}, x^{(1)}, \ldots, x^{(k)}$. Для возможности решения СЛАУ методом простых итераций необходимо, что бы выполнялось одно из условий сходимости.

В данной работе я буду пользоваться проверкой на наличие диагонального преобладания:

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \,, \qquad i = 1, 2, \ldots, n$$

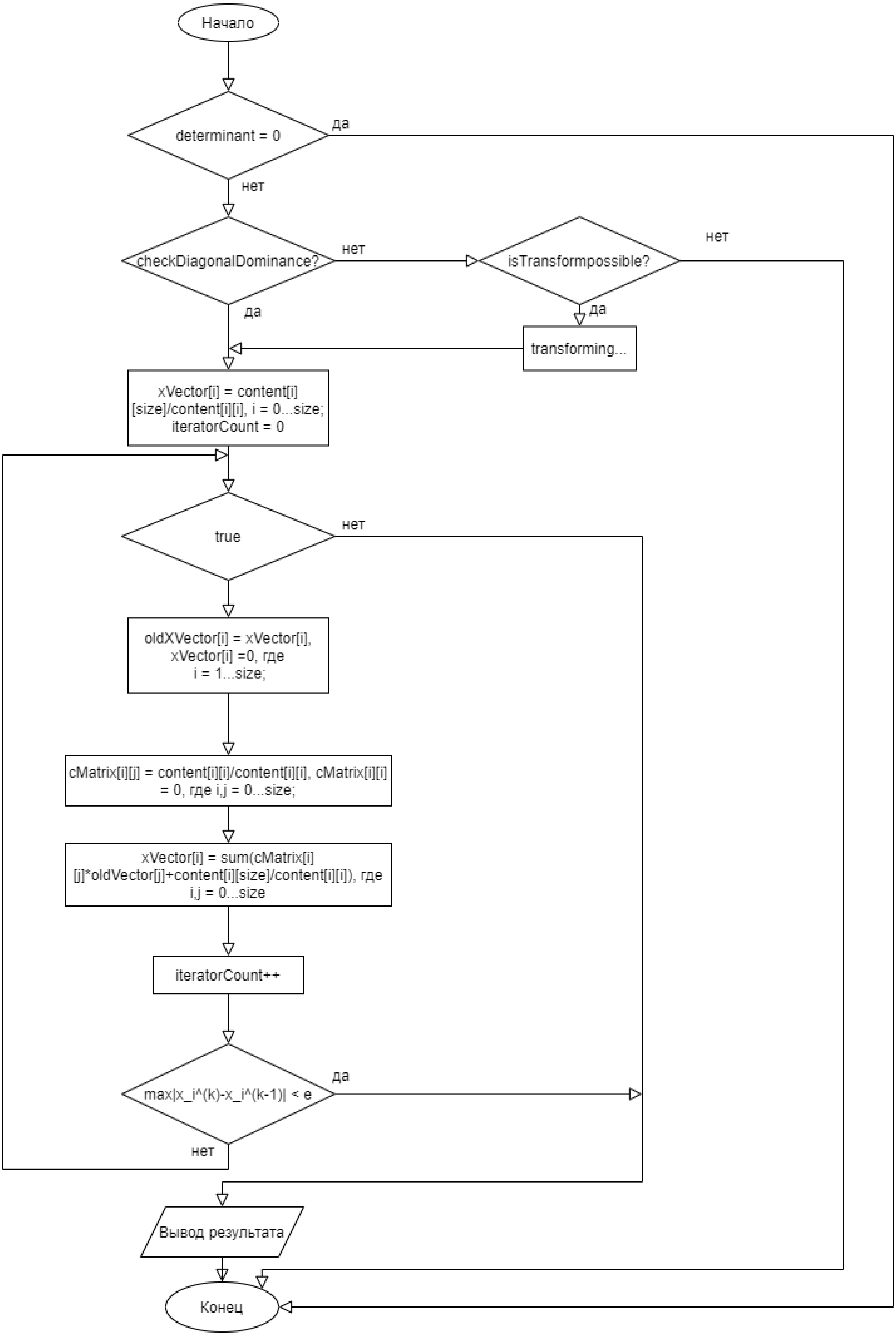
Как правило, за конечное число шагов (т.е. итераций) предел не достигается, поэтому используется такое понятие, как желаемая точность ε - некоторое положительное и достаточно малое число, и процесс вычислений (итераций) проводят до тех пор, пока не будет выполнено некоторое условие, называемое критерием окончания итерационного процесса.

Существует несколько таких критериев: критерий по относительным разностям, критерий по невязке, и критерий по абсолютным отклонениям.

В данной работе я буду пользоваться критерием по абсолютным отклонениям, так как это наиболее простой способ. Суть критерия - сравнение между собой соответствующих неизвестных по двум соседним итерациям (k) и (k-1).
Итерационный процесс продолжается до тех пор, пока не будет выполнено соотношение:

$$\max|x_i^{(k)} - x_i^{(k-1)}| < \varepsilon$$

Блок схема метода:



Листинг программы (только численный метод):
Matrix.java

```
public String start() {
        oldXVector = new double[size];
        xVector = new double[size];
        maxValuesIndex = new int[size];
        double[][] tmp = makeTMPMatrixForCheck();
        DeterminantCalc determinantCalc = new DeterminantCalc(tmp);


        if (determinantCalc.determinant().compareTo(BigDecimal.ZERO) == 0) {
            System.out.println("Детерминант равен 0. \nЭта система несовместна");
            exit(0);

        }
        ;
        if (checkDiagonalDominance()) {
            description = "Есть диагональное преобладание...\n";
            solve();
        } else {
            description = "В этой матрице нет диагонального преобладания! \n";
            if (isTransformPossible()) {
                description += "Трансформирую матрицу...\n";
                transform();
                description += show() + "\n";
                solve();
            } else description += "Трансформация невозможна\nЭто задание невозможно решить
итерационным методом";
        }
        return description;
    }

    private void solve() {
        initXVector();

        int iteratorCount = 0;
        double[][] cmatrix = makeCmatrix();

        while (!isTheEnd()) {
            iteratorCount++;
            for (int i = 0; i < size; i++) {
                oldXVector[i] = xVector[i];
                xVector[i] = 0;
            }
            for (int i = 0; i < size; i++) {
                for (int j = 0; j < size; j++) {
                    xVector[i] += cmatrix[i][j] * oldXVector[j];
                }
                xVector[i] += content[i][size] / content[i][i];
            }
        }

        description += "Решение: \n";
        for (int i = 0; i < size; i++) {
            description += "x" + (i + 1) + " =  " + xVector[i] + "\n";
        }
        description += "Погрешности: \n";
        for (int i = 0; i < size; i++)
            description += "dx[" + i + "]=  " + Math.abs(xVector[i] - oldXVector[i]) +
"\n";
```

```
        description += "\nКоличество итераций: " + iteratorCount;

    }

    private void initXVector() {
        for (int i = 0; i < size; i++)
            xVector[i] = content[i][size] / content[i][i];
    }

    private boolean isTheEnd() {
        for (int i = 0; i < size; i++) {
            if (abs(xVector[i] - oldXVector[i]) < accuracy)
                return true;
        }
        return false;
    }

    private double[][] makeCmatrix() {
        double[][] fcmatrix = new double[size][size];
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                fcmatrix[i][j] = -doubleRound(content[i][j] / content[i][i], 4);
            }
            fcmatrix[i][i] = 0;
        }
        return fcmatrix;
    }

    private double[][] makeTMPMatrixForCheck() {
        double[][] tmpMatrix = new double[size][size];
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                tmpMatrix[i][j] = content[i][j];
            }
        }
        return tmpMatrix;
    }

    private double doubleRound(double value, int places) {
        double scale = Math.pow(10, places);
        return Math.round(value * scale) / scale;
    }

    private boolean checkDiagonalDominance() {
        boolean isOK = true;
        for (int i = 0; i < size; i++) {
            int lineSum = 0;
            for (int j = 0; j < size; j++) {
                if (i != j)
                    lineSum += abs(content[i][j]);
            }
            isOK &= (abs(content[i][i]) >= lineSum);
        }
        return isOK;
    }

    private boolean isTransformPossible() {
        double maxValue;
        int indexMax;
        for (int i = 0; i < size; i++) {
            maxValue = Math.abs(content[i][0]);
```

```
            indexMax = 0;
            for (int j = 1; j < size; j++) {
                if (Math.abs(content[i][j]) > maxValue) {
                    maxValue = Math.abs(content[i][j]);
                    indexMax = j;
                }
            }
            maxValuesIndex[i] = indexMax;
        }
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                if (i != j) {
                    if (maxValuesIndex[i] == maxValuesIndex[j]) {
                        return false;
                    }
                }
            }
        }
        return true;
    }

    private void transform() {
        double[][] tmpContent = new double[size][size];

        for (int i = 0; i < size; i++) {
            int lineIndex = maxValuesIndex[i];
            tmpContent[lineIndex] = content[i];
        }
        content = tmpContent;
    }
```

Пример и результат работы программы:

Пример 1. Чтение данных из файла

```
0.000001
5
100 1 2 3 4 5
6 200 7 8 9 10
11 12 300 13 14 15
16 17 18 400 19 20
21 22 23 24 500 25
```

(файл ex3.txt)

```
Вы ввели: Размер:  5
Точность:  1.0E-6
-------------------
 100,000000    1,000000    2,000000    3,000000    4,000000    5,000000
   6,000000  200,000000    7,000000    8,000000    9,000000   10,000000
  11,000000   12,000000  300,000000   13,000000   14,000000   15,000000
  16,000000   17,000000   18,000000  400,000000   19,000000   20,000000
  21,000000   22,000000   23,000000   24,000000  500,000000   25,000000

Есть диагональное преобладание...
Решение:
x1 =  0.04575101490817983
x2 =  0.04353753997031654
x3 =  0.04277507580377513
x4 =  0.042392234720268124
x5 =  0.0421604477300665
Погрешности:
dx[0]=  4.941830519591961E-7
dx[1]=  6.925797493492758E-7
dx[2]=  7.486072747947548E-7
dx[3]=  7.750691397648279E-7
dx[4]=  7.906188003101167E-7

Количество итераций: 6
```

Пример 2. Генерация матрицы

```
Вы ввели: Размер:  5
Точность:  0.001
-------------------
 109,215634    9,709695   11,612753   -2,329650  -10,153253  -12,381824
   9,023547  109,765431   13,558540   -8,291822   -9,206058   -2,771522
  -4,098496   10,284243   77,189900    1,406165   -1,333769  -14,747044
   5,478520   -8,177476   11,448165   87,820142   -0,168747    5,052801
   3,751183   -6,494862   -2,917798   11,720425   87,584260   -5,815024

Есть диагональное преобладание...
Решение:
x1 =  -0.09846951822543878
x2 =  0.006888141163281878
x3 =  -0.1999161127136295
x4 =  0.08950928434041967
x5 =  -0.07998728124533808
Погрешности:
dx[0]=  0.0027031745328357643
dx[1]=  4.5051219474778054E-4
dx[2]=  0.0040147429592946315
dx[3]=  0.0024632865100119583
dx[4]=  0.0025123251668007496

Количество итераций: 2
```

Пример 3. Чтение данных с клавиатуры.

```
Введите размер матрицы... (Размер матрицы - целое число [1;20])
3
Введите точность...
0.01
Введитее матрицу построчно, разделяя элементы строки пробелами:
2 2 10 14
10 1 1 12
2 10 1 13
Вы ввели: Размер:  3
Точность:  0.01
-------------------
   2,000000    2,000000   10,000000   14,000000
  10,000000    1,000000    1,000000   12,000000
   2,000000   10,000000    1,000000   13,000000

В этой матрице нет диагонального преобладания!
Трансформирую матрицу...
Размер:  3
Точность:  0.01
-------------------
  10,000000    1,000000    1,000000   12,000000
   2,000000   10,000000    1,000000   13,000000
   2,000000    2,000000   10,000000   14,000000
```

```
Решение:
x1 =  1.0015
x2 =  1.0019200000000001
x3 =  1.0024
Погрешности:
dx[0]=  0.006900000000000128
dx[1]=  0.008520000000000083
dx[2]=  0.010800000000000032

Количество итераций: 4
```

Вывод:

В данной работе реализован один из методов решения СЛАУ – метод простых итераций.

Этот метод дает худшую сходимость, чем другой итерационный метод – метод Гаусса-Зейделя, но приводит к менее объемным вычислениям. Общий недостаток итерационных методов – зависимость от начального приближения.

Погрешности в итерационных методах не накапливаются, поскольку точность вычислений в каждой итерации определяется лишь результатами предыдущей.

Также следует сказать про прямые методы решения СЛАУ.

Метод Гаусса - один из самых распространенных прямых методов. Он заключается в последовательном исключении переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе ступенчатого вида, из которого последовательно, начиная с последних по номеру переменных, находятся все остальные.

Преимущества данного метода - наличие множества модификаций, сравнительная простота, и большая универсальность, по сравнению с итерационными.

Но данный метод менее эффективен при решении матриц больших размеров, так как алгоритмическая сложность данного метода O(n^3). Таким образом, для больших систем метод Гаусса практически не применим.