Университет ИТМО
Кафедра Вычислительной Техники

Лабораторная работа № 1

по предмету Вычислительная Математика

Метод Простых итераций

Выполнил:

Тарасов А.С.

Преподаватель:

Перл О. В.

Санкт-Петербург – 2020

Решение системы линейных алгебраических уравнений
Метод просты итераций

Описание метода:

Метод простых итераций – один из классических методов решения систем линейных алгебраических уравнений. 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 $x^{(0)}, x^{(1)}, \dots, x^{(k)}$. 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:

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

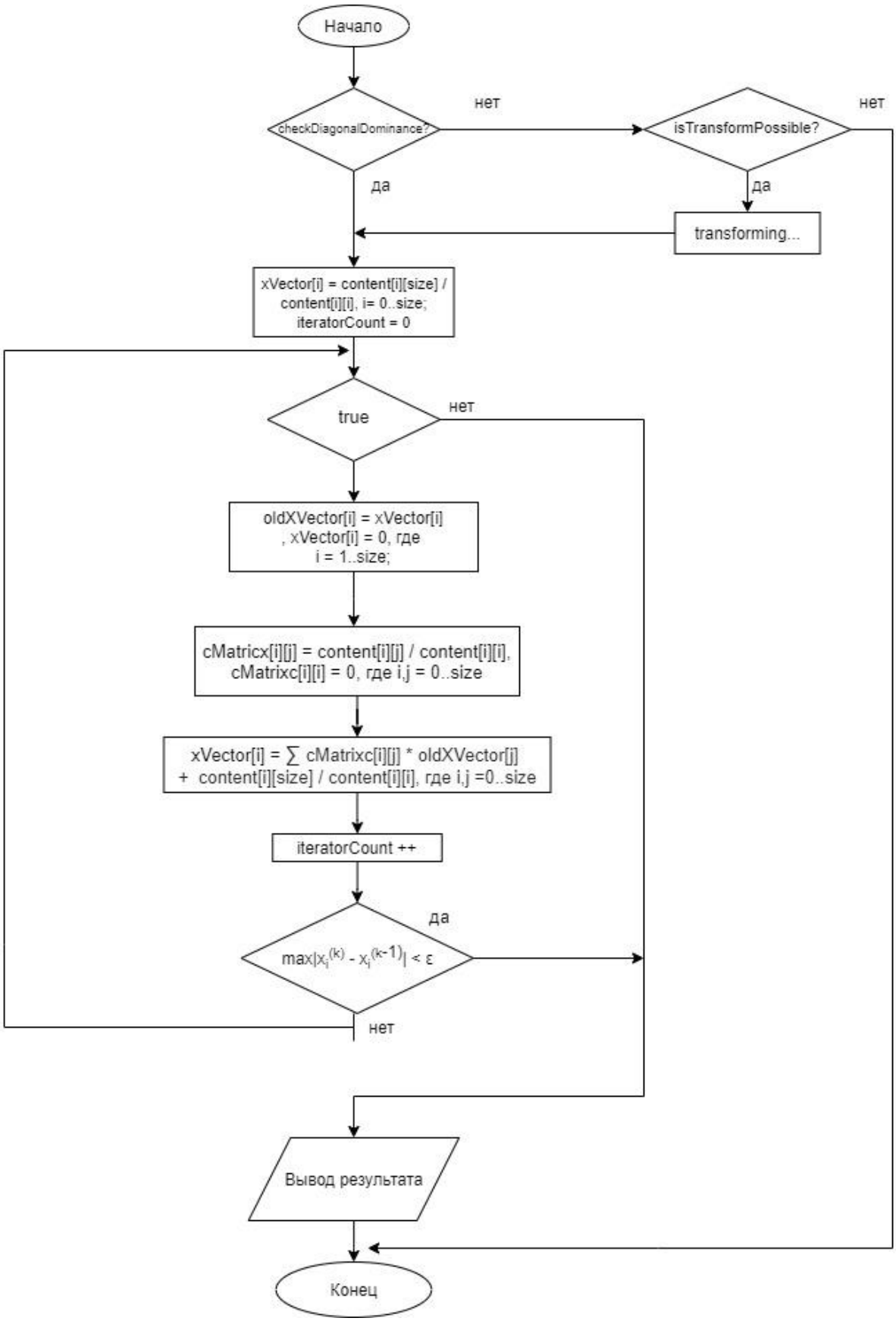
Как правило, за конечное число шагов (т.е. итераций) предел не достигается, поэтому используется такое понятие, как желаемая точность ε - некоторое положительное и достаточно малое число, и процесс вычислений (итераций) проводят до тех пор, пока не будет выполнено некоторое условие, называемое критерием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

Существует несколько таки 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:

$$\max|xi(k) - xi(k-1)| < \varepsilon$$

Блок схема метода:



Листинг программы(только численный метод):
Matricx.java

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


    if (checkDiagonalDominance()) {
        description = "Diagonal dominance is detected...\n";
        solve();
    } else{
        description = "! there is no diagonal dominance in this matricx !\n";
        if(isTransformPossible()) {
            description += "transforming matricx...\n";
            transform();
            description += show() + "\n";
            solve();
        }else description += "transforming is impossible\nIt's impossible to solve this
task by iteration method";
    }
    return description;
}

private void solve(){
    initXVector();

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

    while (!isTheEnd()) {
        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];
        }
        iteratorCount++;
        if (iteratorCount % 3 == 0) xCircleTest = xVector[0];
        if (iteratorCount % 3 != 0)
            if (xCircleTest == xVector[0]) {
                description += "\n CIRCLE\n";
                break;
            }
    }

    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 + "]=  " + (xVector[i] - oldXVector[i]) + "\n";

    description += "\niter: " + iteratorCount;

}
```

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

private boolean isTheEnd() {
    double max = -999999;
    for (int i = 0; i < size; i++) {
        if (abs(xVector[i] - oldXVector[i]) > max)
            max = abs(xVector[i] - oldXVector[i]);
    }
    return max < accuracy;
}

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

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)

```
Укажите способ установки данных:  k - клавиатура     f - файл     g - сгенерировать данные
f
введите путь к файлу...
ex3.txt

Вы ввели:size:  5
accuracy:  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

Diagonal dominance is detected...
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

iter: 6
```

Пример 2. Генерация матрицы

```
WELCOME!
Укажите способ установки данных:  k - клавиатура     f - файл     g - сгенерировать данные
g
введите размер матрицы...
5
введите точность...
0.001
Вы ввели:size:  5
accuracy:  0.001
--------------------
  81,404181    9,849545   -8,256942    7,830792    6,391446    6,665675
   5,338675   51,882975    2,791658   -1,481073    5,492548    2,060427
  -5,811786    6,903963   62,625736    2,062562  -11,396365   -2,553963
   1,717232   -5,016157   12,009142   86,018041  -10,938467   -8,143359
 -13,281184   -4,857419   10,540205   11,351796  100,541046   -8,266915

Diagonal dominance is detected...
Решение:
x1 =  0.08649290230686717
x2 =  0.036171850954510115
x3 =  -0.043448175748363946
x4 =  -0.09511780888750784
x5 =  -0.05385126348498845
Погрешности:
dx[0]=  2.4767147927162725E-4
dx[1]=  9.588253430384275E-5
dx[2]=  -2.245275984247147E-4
dx[3]=  -2.75360025921170 26E-4
dx[4]=  7.790250746669142E-5

iter: 4
```

Пример 3. Чтение данных с клавиатуры.

```
Укажите способ установки данных:  k - клавиатура     f - файл     g - сгенерировать данные
k
введите размер матрицы...
4
введите точность...
0.001
Введитее матрицу построчно, разделяя элементы строки пробелами
10 1 1 1 12
2 2 2 11 14
1 9 2 2 1
2 2 14 1 1
Вы ввели:size:  4
accuracy:  0.001
--------------------
  10,000000    1,000000    1,000000    1,000000   12,000000
   2,000000    2,000000    2,000000   11,000000   14,000000
   1,000000    9,000000    2,000000    2,000000    1,000000
   2,000000    2,000000   14,000000    1,000000    1,000000

! there is no diagonal dominance in this matrix !
transforming matrix...
size:  4
accuracy:  0.001
--------------------
  10,000000    1,000000    1,000000    1,000000   12,000000
   1,000000    9,000000    2,000000    2,000000    1,000000
   2,000000    2,000000   14,000000    1,000000    1,000000
   2,000000    2,000000    2,000000   11,000000   14,000000

Решение:
x1 =  1.123808366872266
x2 =  -0.23553072079892684
x3 =  -0.13635009351170102
x4 =  1.1364440827883773
```

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.