Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1 по "Вычислительной математике"

Решение системы линейных алгебраических уравнений СЛАУ

Вариант 4

Метод простых итераций

Выполнил:	Дьяконов Михаил Павлович
Группа:	P3211
Преподаватель:	Малышева Татьяна Алексеевна

Университет ИТМО, 2021

Описание метода

Метод простых итераций - один из простейших численных методов решения уравнений, относится к категории итерационных методов. Задается некоторое начальное приближение. Далее с помощью алгоритма проводится один цикл вычислений - итерация. В результате итерации находят новое приближение. Итерации проводятся до получения решения с требуемой точностью. Достаточным условием сходимости итерационного процесса к решению системы является выполнение условия преобладания диагональных элементов:

$$|a_{ii}| \leq \sum_{j \neq i} |a_{ij}|, i = 1...n$$

При этом хотя бы для одного уравнения неравенство должно выполняться строго. Далее выражаем неизвестные x_1, x_2, x_3 и т.д. соответственно из первого, второго и третьего и т.д. уравнений системы. Система в сокращенном виде будет выглядеть так:

$$x_i = \sum_{j=1}^{n} c_{ij} x_j + d_i, i = 1...n$$

Рабочая формула метода:

$$x_i^{(k+1)} = \frac{b_i}{a_{ii}} - \sum_{j \neq i}^{n} a_{ij} x_j^k, i = 1...n$$

, где k - номер итерации.

За начальное (нулевое) приближение выбирают вектор свободных членов: $x^{(0)} = D$ или нулевой вектор. Следующее приближение: $\vec{x}^{(1)} = c\vec{x}^{(0)} + \vec{d}$, $\vec{x}^{(2)} = c\vec{x}^{(1)} + \vec{d}$ ...

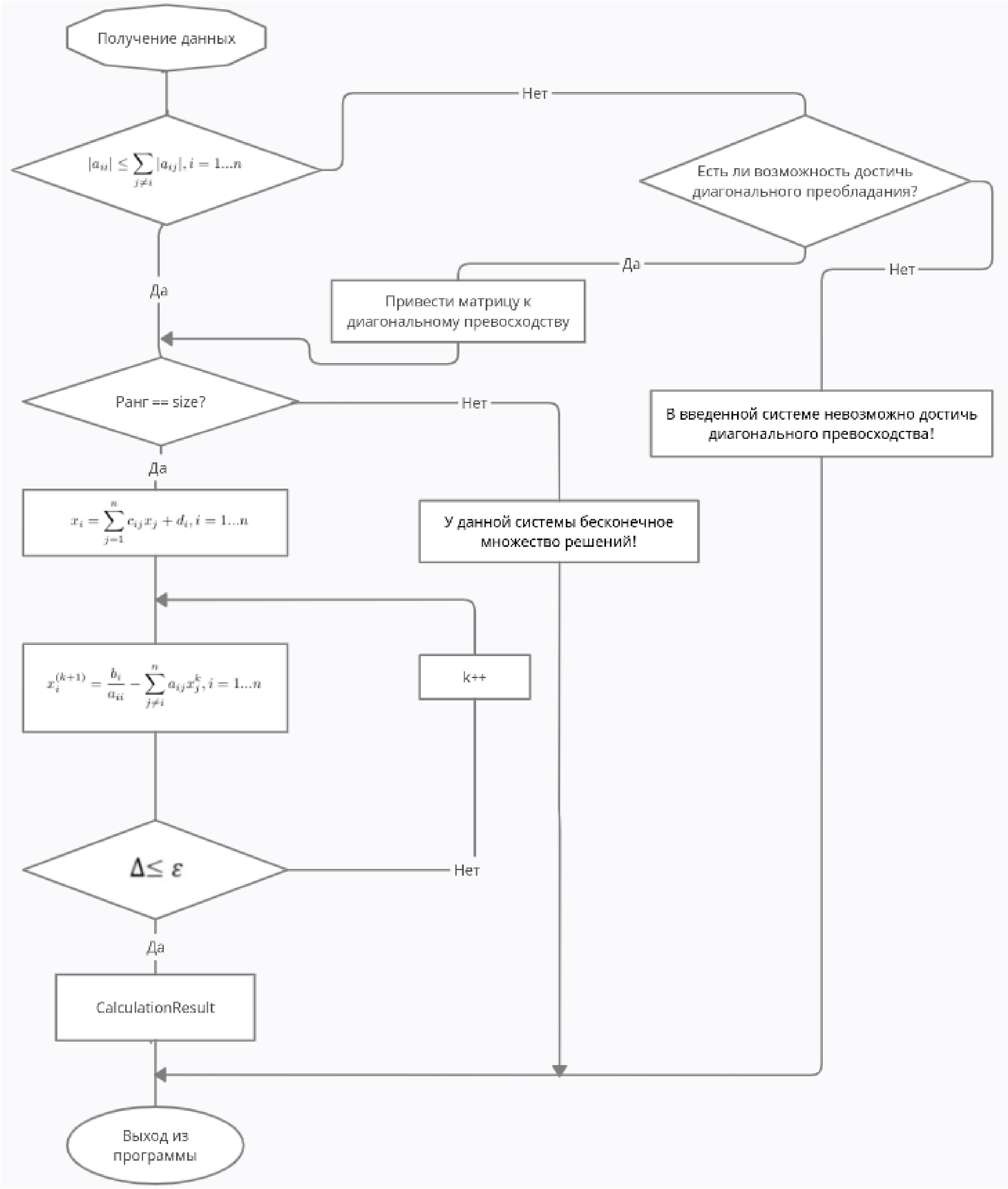
Листинг программы

SimpleIterationCalculator.java:

```java
public CalculationResult calculate() {
    prepareMatrix();
    prepareApproximations();
    double[] faultVector = new double[matrix.getKoefs().length];
    int iterationCount = 0;
    do {
        iterationCount++;
        double[] nextXVector = new double[approximationVector.length];
        IntStream.range(0, nextXVector.length).forEach(i -> {
            nextXVector[i] = IntStream.range(0, nextXVector.length)
                    .mapToDouble(j -> matrix.getKoefs()[i][j] * approximationVector[j]).sum() +
                        matrix.getFreeMembers()[i];
            faultVector[i] = Math.abs(nextXVector[i] - approximationVector[i]);
        });
        approximationVector = nextXVector;
    } while (DoubleStream.of(faultVector).max().orElse(0) > neededAccuracy);
    return new CalculationResult(approximationVector, faultVector, iterationCount);
}

private void prepareMatrix() {
    double[][] koefs = matrix.getKoefs();
    double[] freeMembers = matrix.getFreeMembers();
    for (int i = 0; i < koefs.length; i++) {
        double mainKoef = koefs[i][i];
        for (int j = 0; j < koefs[i].length; j++) {
            koefs[i][j] = -koefs[i][j] / mainKoef;
        }
        freeMembers[i] = freeMembers[i] / mainKoef;
        koefs[i][i] = 0;
    }
}
```

Блок-схема численного метода



Примеры

Чтение из файла:

Чтение из файла test2.txt ...
Введите точность (от 0.000001 до 1.0): 0.001
Вектор неизвестных: [0.99995256, -1.99987658, 2.99982052, -3.9999663, 4.99960141, -6.00005577]
Вектор погрешностей: [5.6373E-4, 1.145E-5, 3.3509E-4, 7.1554E-4, 4.6596E-4, 7.5354E-4]
Количество итераций: 9

Чтение из консоли:

Введите размерность матрицы: 3
Введите матрицу:
2 2 10 14
10 1 1 12
2 10 1 13
Введите точность (от 0.000001 до 1.0): 0.01
Вектор неизвестных: [0.999568, 0.99946, 0.999316]
Вектор погрешностей: [0.001932, 0.00246, 0.003084]
Количество итераций: 5

Генерация случайной матрицы:

Генерирование вектора неизвестных с длиной 12...
Ожидаемый вектор неизвестных: [5.0, 2.0, 3.0, -5.0, 8.0, 9.0, -2.0, 4.0, -9.0, 3.0, -4.0, 2.0]
Генерирование матрицы 12x12...
[3897.0, 135.0, -267.0, -483.0, 484.0, 62.0, -495.0, -305.0, 81.0, 45.0, -218.0, 30.0] * X = 25907.0
[485.0, 3418.0, -267.0, -135.0, 260.0, 148.0, 250.0, 423.0, -83.0, 270.0, 4.0, 178.0] * X = 15636.0
[404.0, -114.0, 2611.0, 275.0, -172.0, 128.0, 417.0, 16.0, -55.0, -85.0, -414.0, 291.0] * X = 9734.0
[2.0, -173.0, 408.0, 3530.0, -392.0, 66.0, -370.0, -140.0, 188.0, -454.0, -354.0, -336.0] * X = -21434.0
[244.0, 308.0, 33.0, 234.0, 3204.0, 111.0, 318.0, 131.0, -289.0, 50.0, 476.0, 162.0] * X = 28455.0
[301.0, -245.0, 300.0, 300.0, 293.0, 4427.0, 428.0, 244.0, 111.0, 352.0, -41.0, -368.0] * X = 42207.0
[-313.0, 363.0, -265.0, -185.0, 236.0, -98.0, 4041.0, -333.0, -438.0, 354.0, -295.0, 271.0] * X = -2391.0
[126.0, -108.0, -366.0, -376.0, 496.0, -17.0, 258.0, 4376.0, 454.0, 494.0, -191.0, -497.0] * X = 19165.0
[-473.0, -71.0, 273.0, 71.0, 282.0, -18.0, 344.0, 320.0, 3605.0, 123.0, -260.0, 460.0] * X = -29473.0
[-156.0, 165.0, 73.0, 437.0, -358.0, -410.0, -197.0, -376.0, 486.0, 4278.0, 244.0, 490.0] * X = -1616.0
[136.0, 219.0, 332.0, -215.0, 335.0, -381.0, 209.0, -216.0, -115.0, -477.0, 3297.0, 293.0] * X = -11840.0
[-170.0, 395.0, -91.0, 340.0, 137.0, -223.0, -20.0, 287.0, -302.0, 466.0, -218.0, 2946.0] * X = 9124.0
Используется точность = 0.001
Вектор неизвестных: [5.0000186, 2.0000683, 3.000026, -5.0000144, 8.0000507, 8.9999701, -1.9999846, 3.9999852, -9.0000037, 2.9999206, -3.9998947, 2.0000111]
Вектор погрешностей: [2.186E-4, 4.421E-4, 3.2E-6, 3.315E-4, 9.91E-5, 8.1E-5, 1.39E-4, 2.639E-4, 1.94E-4, 3.929E-4, 2.39E-5, 2.77E-5]
Количество итераций: 8

Вывод

В результате выполнения данной лабораторной работы я узнал, что итерационные методы просты для реализации на ЭВМ. Также они не требуют хранения в памяти машины всей матрицы системы, а только нескольких векторов с n компонентами.