

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

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по дисциплине «Вычислительная математика»

Вариант 7

Студент:

Кулагин Вячеслав

Группа:

P3209

Преподаватель:

Наумова Надежда

Александровна

Санкт-Петербург
2025

Оглавление

Задание	Ошибка! Закладка не определена.
Реализация запросов на SQL	Ошибка! Закладка не определена.
Ответы на вопросы, представленные в задании	Ошибка! Закладка не определена.
Планы выполнения запросов	Ошибка! Закладка не определена.
Вывод	Ошибка! Закладка не определена.

Цель работы

Создать программу для решения СЛАУ размерностью не более 20 методом простых итераций.

Описание метода

Метод простых итераций используется для приближенного решения СЛАУ. Он заключается в последовательном уточнении значений неизвестных, начиная с некоторого начального приближения.

На каждом шаге вычисляются новые значения переменных, используя предыдущее приближение. Этот процесс продолжается до тех пор, пока изменения в значениях переменных не станут достаточно малы.

Основные формулы

$$x_i^{(k+1)} = \frac{b_i}{a_{ii}} - \sum_{\substack{j=1 \\ j \neq i}}^n \frac{a_{ij}}{a_{ii}} x_j^k, \quad i = 1, 2, \dots, n$$

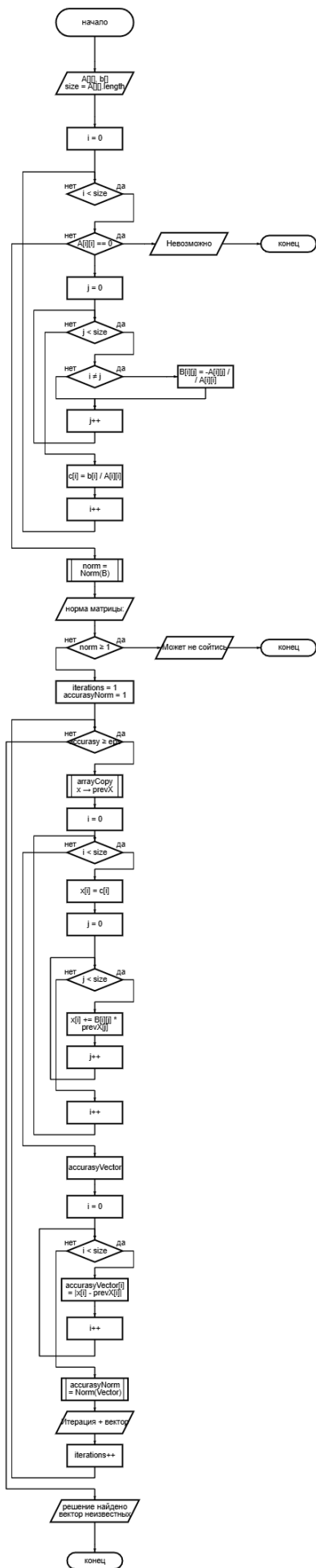
Основная рабочая формула:

Достаточное условие сходимости итерационного процесса:

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

Достаточное условие сходимости метода по норме: $\|C\| < 1$

Блок-схема



Листинг метода

```
import java.util.Arrays;

public class Solver {
    private static double epsilon = 1e-6;

    public static void solve(double[][] A, double[] b) {
        int size = A.length;
        double[][] B = new double[size][size];
        double[] c = new double[size];

        for (int i = 0; i < size; i++) {
            if (A[i][i] == 0) {
                System.out.println("Невозможно решить: на диагонали нулевой элемент");
                return;
            }
            for (int j = 0; j < size; j++) {
                if (i != j) {
                    B[i][j] = -A[i][j] / A[i][i];
                }
            }
            c[i] = b[i] / A[i][i];
        }

        double norm = calculateNorm(B);
        System.out.println("Норма матрицы B: " + norm);
        if (norm >= 1) {
            System.out.println("Метод может не сходиться, так как норма B >= 1.");
            return;
        }

        double[] x = new double[size];
        double[] prevX = new double[size];
        int iterations = 1;
        double accuracyNorm = 1;

        while (accuracyNorm >= epsilon) {
            System.arraycopy(x, 0, prevX, 0, size);

            for (int i = 0; i < size; i++) {
                x[i] = c[i];
                for (int j = 0; j < size; j++) {
                    x[i] += B[i][j] * prevX[j];
                }
            }

            double[] accuracyVector = new double[size];
            for (int i = 0; i < size; i++) {
                accuracyVector[i] = Math.abs(x[i] - prevX[i]);
            }
            accuracyNorm = calculateNorm(accuracyVector);
            System.out.println("Итерация " + iterations + ", Вектор погрешности: " + Arrays.toString(accuracyVector));

            iterations++;
        }
    }
}
```

```

        System.out.println("Решение найдено за " + (iterations - 1) + " итераций");
        System.out.println("Вектор неизвестных x: " + Arrays.toString(x));
    }

    private static double calculateNorm(double[][] matrix) {
        double maxSum = 0;
        for (double[] row : matrix) {
            double sum = 0;
            for (double val : row) {
                sum += Math.abs(val);
            }
            maxSum = Math.max(maxSum, sum);
        }
        return maxSum;
    }

    private static double calculateNorm(double[] vector) {
        double max = 0;
        for (double v : vector) {
            max = Math.max(max, Math.abs(v));
        }
        return max;
    }

    public static void setEpsilon(double epsilon) {
        Solver.epsilon = epsilon;
    }
}

```

Результат работы

Чтение из файла:

Чтение данных из файла: in.txt

Введите необходимую точность (степень 10, не больше -1, не меньше -15), по умолчанию: -6

Введите размерность матрицы (n <= 20): Введите коэффициенты матрицы построчно (без свободных членов!):

Введенная матрица коэффициентов:

2.0 2.0 10.0

10.0 1.0 1.0

2.0 10.0 1.0

Введите свободные члены:

Попытка преобразовать матрицу...

Новая матрица с диагональным преобладанием:

10.0 1.0 1.0

2.0 10.0 1.0

2.0 2.0 10.0

Норма матрицы B: 0.4

Итерация 1, Вектор погрешности: [1.2, 1.3, 1.4]

Итерация 2, Вектор погрешности: [0.27000000000000013, 0.38, 0.5]
Итерация 3, Вектор погрешности: [0.08799999999999997, 0.10399999999999998,
0.13000000000000012]
Итерация 4, Вектор погрешности: [0.023399999999999865, 0.03059999999999996,
0.03840000000000001]
Итерация 5, Вектор погрешности: [0.006900000000000128, 0.008520000000000083,
0.010800000000000032]
Итерация 6, Вектор погрешности: [0.0019320000000001558, 0.0024600000000002398,
0.003083999999999756]
Итерация 7, Вектор погрешности: [5.544000000000066E-4, 6.948000000003285E-4,
8.784000000000569E-4]
Решение найдено за 7 итераций
Вектор неизвестных x: [1.0001224, 1.0001548000000002, 1.0001944]

Process finished with exit code 0

Чтение из консоли:

Введите необходимую точность (степень 10, не больше -1, не меньше -15), по
умолчанию: -6

-6

Введите размерность матрицы ($n \leq 20$): 3

Введите коэффициенты матрицы построчно (без свободных членов!):

2 2 10

10 1 1

2 10 1

Введенная матрица коэффициентов:

2.0 2.0 10.0

10.0 1.0 1.0

2.0 10.0 1.0

Введите свободные члены:

14 12 13

Попытка преобразовать матрицу...

Новая матрица с диагональным преобладанием:

10.0 1.0 1.0

2.0 10.0 1.0

2.0 2.0 10.0

Норма матрицы B: 0.4

Итерация 1, Вектор погрешности: [1.2, 1.3, 1.4]

Итерация 2, Вектор погрешности: [0.27000000000000013, 0.38, 0.5]

Итерация 3, Вектор погрешности: [0.08799999999999997, 0.10399999999999998,
0.13000000000000012]

Итерация 4, Вектор погрешности: [0.023399999999999865, 0.03059999999999996,
0.03840000000000001]
Итерация 5, Вектор погрешности: [0.006900000000000128, 0.008520000000000083,
0.010800000000000032]
Итерация 6, Вектор погрешности: [0.0019320000000001558, 0.0024600000000002398,
0.003083999999999756]
Итерация 7, Вектор погрешности: [5.54400000000066E-4, 6.948000000003285E-4,
8.784000000000569E-4]
Итерация 8, Вектор погрешности: [1.573200000001634E-4, 1.9872000000020762E-4,
2.498400000001677E-4]
Итерация 9, Вектор погрешности: [4.4856000000148555E-5, 5.644800000004224E-5,
7.120800000004479E-5]
Итерация 10, Вектор погрешности: [1.2765600000186339E-5, 1.60920000000786E-5,
2.0260800000060364E-5]
Итерация 11, Вектор погрешности: [3.635280000136021E-6, 4.579200000076611E-6,
5.771520000030783E-6]
Итерация 12, Вектор погрешности: [1.0350720001106595E-6, 1.3042080000413847E-6,
1.6428960000203219E-6]
Итерация 13, Вектор погрешности: [2.9471040008388627E-7, 3.713039999908574E-7,
4.678560000526133E-7]
Решение найдено за 13 итераций
Вектор неизвестных x: [1.0000000653184, 1.000000082296, 1.00000010368]

Вывод

Проведя эту работу, я научился работать с методом простых итераций, узнал и реализовал как с его помощью можно решать СЛАУ хоть приблизительно, но все же достаточно точно.