

Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1  
по «Вычислительная техника»

Выполнил:  
Студент Р3207  
Разинкин А.В.

Преподаватели:  
Рыбаков С.Д

г. Санкт-Петербург  
2024

## Цель работы

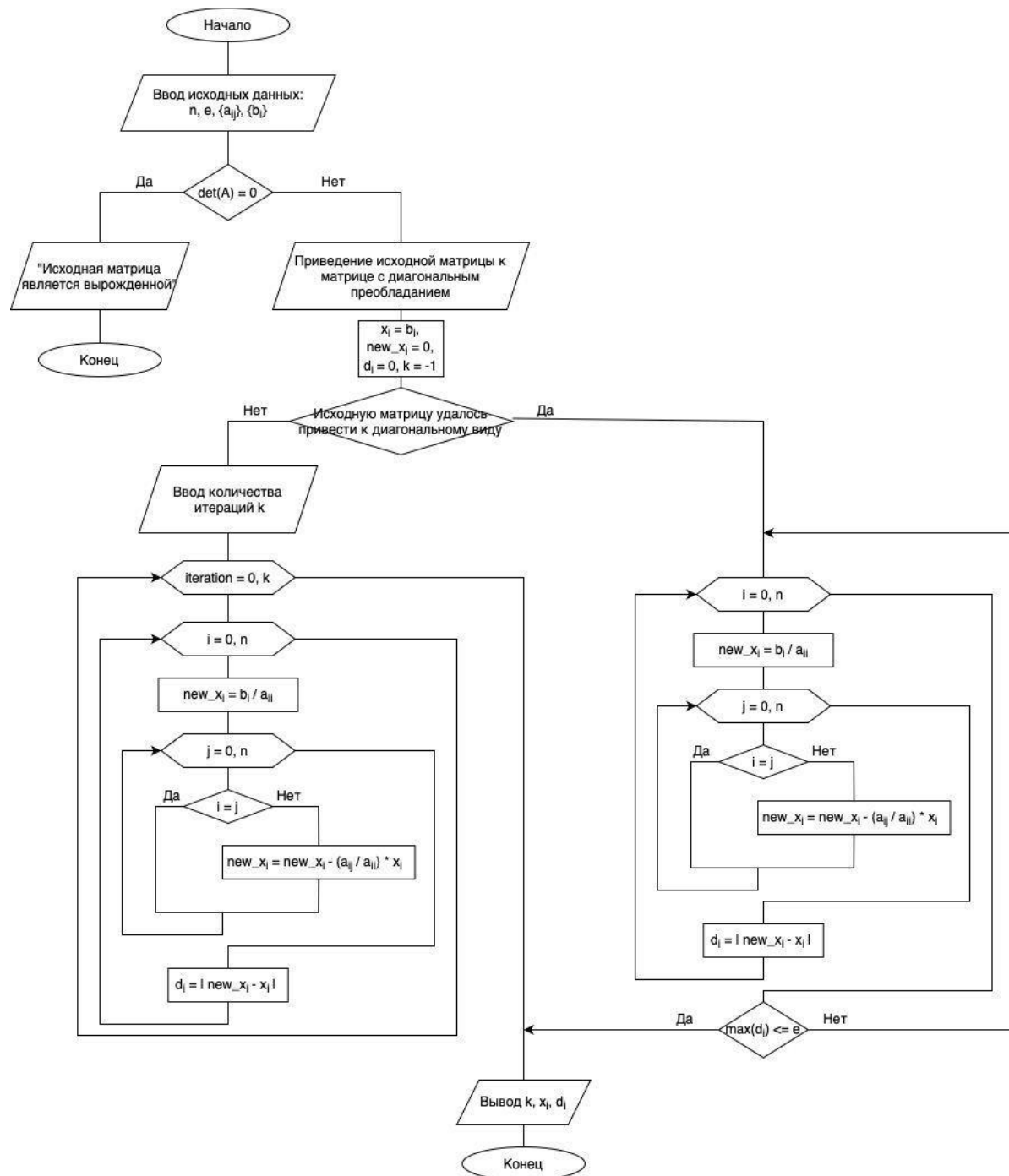
Разработать консольное приложение для решения систем линейных алгебраических уравнений методом простых итераций

## Задание

Должно быть реализовано:

- Ввод данных при помощи клавиатуры или файла – по усмотрению пользователя.
- Ввод размерности матрицы и точности вычислений.
- Проверка диагонального преобладания (в случае, если диагональное преобладание в исходной матрице отсутствует, сделать перестановку строк. В случае невозможности достижения диагонального преобладания – выводить соответствующее сообщение.
- Вывод вектора неизвестных:  $x_1, x_2, \dots, x_n$ .
- Вывод количества итераций, за которое было найдено решение.
- Вывод вектора погрешностей:  $|x_i^{(k)} - x_i^{(k-1)}|$

## Блок-схема реализованного алгоритма



## Реализация (код) численного метода

```
// метод для приведения матрицы к матрице с диагональным преобладанием
public void toConvergence(Data data) {
    List<List<Double>> A = data.getA();
    List<Double> B = data.getB();
    List<List<Integer>> rowPositions = new ArrayList<>();
    for (int i = 0; i < A.size(); i++)
        rowPositions.add(new ArrayList<>());

    for (int i = 0; i < A.size(); i++) {
        List<Double> row = A.get(i);

        double max = Math.abs(row.get(0));
        List<Integer> positions = new ArrayList<>();
        positions.add(0);
        double sum = 0;

        for (int j = 1; j < row.size(); j++) {
            double current = Math.abs(row.get(j));
            if (current > max) {
                sum += max * positions.size();
                max = current;
                positions.clear();
                positions.add(j);
            } else if (current == max) {
                positions.add(j);
            } else {
                sum += current;
            }
        }

        if (max < sum + max * (positions.size() - 1)) {
            System.out.print("
            Не удалось привести исходную матрицу к матрице с
диагональным преобладанием.
            Предупреждение: при продолжении решения конечный ответ
может не сойтись."");
            InputReader inputReader = new InputReader();
            data.setIterations(inputReader.readPositiveInt("Введите кол-во
итераций: "));
            return;
        }

        for (int position : positions)
            rowPositions.get(position).add(i);
    }

    int[] permutation = new int[A.size()];
    List<Integer> placedRows = new ArrayList<>();
    Arrays.fill(permutation, -1);

    for (int i = 0; i < A.size(); i++) {
        List<Integer> rowPosition = rowPositions.get(i);
```

```

        if (rowPosition.size() == 1) {
            permutation[i] = rowPosition.get(0);
            placedRows.add(rowPosition.get(0));
        }
    }

    for (int i = 0; i < A.size(); i++) {
        List<Integer> rowPosition = rowPositions.get(i);
        if (rowPosition.size() == 2) {
            if (!placedRows.contains(rowPosition.get(0))) {
                permutation[i] = rowPosition.get(0);
                placedRows.add(rowPosition.get(0));
            } else if (!placedRows.contains(rowPosition.get(1))) {
                permutation[i] = rowPosition.get(1);
                placedRows.add(rowPosition.get(1));
            } else {
                System.out.print("""
                    Не удалось привести исходную матрицу к матрице с
диагональным преобладанием.
                    Предупреждение: при продолжении решения конечный
ответ может не сойтись.""");
                InputReader inputReader = new InputReader();
                data.setIterations(inputReader.readPositiveInt("Введите кол-
во итераций: "));
                return;
            }
        }
    }

    List<List<Double>> new_A = new ArrayList<>();
    List<Double> new_B = new ArrayList<>();

    for (int i : permutation) {
        if (i == -1) {
            System.out.print("""
                Не удалось привести исходную матрицу к матрице с
диагональным преобладанием.
                Предупреждение: при продолжении решения конечный
ответ может не сойтись.""");
            InputReader inputReader = new InputReader();
            data.setIterations(inputReader.readPositiveInt("Введите кол-во
итераций: "));
            return;
        }
        new_A.add(A.get(i));
        new_B.add(B.get(i));
    }

    data.setA(new_A);
    data.setB(new_B);
}

// метод поиска решения с заявленной точностью
public void iterate(Data data) {

```

```

List<List<Double>> A = data.getA();
List<Double> B = data.getB();
int n = data.getA().size();
double[] previousApproximation = new double[n];
for (int i = 0; i < n; i++)
    previousApproximation[i] = data.getB().get(i);

double[] newApproximation = new double[n];
int iterationCounter = 0;

InputReader inputReader = new InputReader();
if (data.getIterations() == -1)
    data.setIterations(inputReader.readPositiveInt("Введите максимальное кол-во итераций: "));

while (true) {

    if (iterationCounter != 0 && iterationCounter == data.getIterations()) {
        int c = inputReader.readPositiveInt("Введите кол-во символов после запятой: ");
        System.out.println("Было проведено " + iterationCounter + " итераций. Но ответ так и не найден.");
        for (int i = 0; i < n; i++) {
            System.out.println("x" + (i + 1) + "=" + String.format("%. " + c + "f", newApproximation[i]) + "; Отклонение составляет: " + String.format("%. " + c + "f", Math.abs(newApproximation[i] - previousApproximation[i])));
        }
        break;
    }

    for (int i = 0; i < n; i++) {
        double newValue = B.get(i) / A.get(i).get(i);
        for (int j = 0; j < n; j++) {
            if (i != j) {
                newValue -= (A.get(i).get(j) / A.get(i).get(i)) * previousApproximation[j];
                if (Double.isNaN(newValue) || Double.isInfinite(newValue)) {
                    System.out.println("Данная СЛАУ не обладает сходящимся решением.");
                    System.exit(1);
                }
            }
        }
        newApproximation[i] = newValue;
    }

    if (getMaxDeviation(previousApproximation, newApproximation) <= data.getAccuracy()
        || (data.getIterations() != -1 && data.getIterations() == iterationCounter)) {
        int c = inputReader.readPositiveInt("Введите кол-во символов после запятой: ");
        System.out.println("Было проведено " + iterationCounter + " итераций.");
        for (int i = 0; i < n; i++) {

```

```

        System.out.println("x" + (i + 1) + "=" + String.format("%. " +
c + "f", newApproximation[i]) + "; Отклонение составляет: " +
String.format("%. " + c + "f", Math.abs(newApproximation[i] -
previousApproximation[i])));
    }
    break;
}
for (int i = 0; i < n; i++)
    previousApproximation[i] = newApproximation[i];
iterationCounter++;
}
}

// метод поиска максимального отклонения следующего приближения
private double getMaxDeviation(double[] previousApproximation, double[]
newApproximation) {
    double maxDeviation = 0;

    for (int i = 0; i < previousApproximation.length; i++) {
        double deviation = Math.abs(previousApproximation[i] -
newApproximation[i]);
        if (deviation > maxDeviation)
            maxDeviation = deviation;
    }

    return maxDeviation;
}

```



Ссылка на GitHub с основной реализацией

[https://github.com/DecafMangoITMO/ITMO/tree/main/Computational%20Mathematics/  
MethodOfSimpleIterations](https://github.com/DecafMangoITMO/ITMO/tree/main/Computational%20Mathematics/MethodOfSimpleIterations)

## Пример работы программы

Для выхода из программы напишите exit.

Введите размерность матрицы: 3

Введите точность: 0.01

Введите коэффициенты построчно.

Например, если ваш имеет вид:

a11 a12 | b1

a21 a22 | b2

Ввод будет следующим:

a11 a12 b1

a21 a22 b2

2 2 10 14

10 1 1 12

2 10 1 13

Введите максимальное кол-во итераций: 100

Введите кол-во символов после запятой: 3

Было проведено 7 итераций.

x1=1.000; Отклонение составляет: 0.002

x2=1.001; Отклонение составляет: 0.002

x3=1.001; Отклонение составляет: 0.003

## Вывод

В ходе реализации данной лабораторной работы я ознакомился с работой алгоритма простых итераций, предназначенного для решения совместных определенных систем линейных алгебраических уравнений (СЛАУ). Данный алгоритм относится к виду итерационных: решение системы (если оно существует) достигается путем приближения за счет конечного числа итераций.