Санкт-Петербургский *Национальный Исследовательский Университет ИТМО*
Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №4

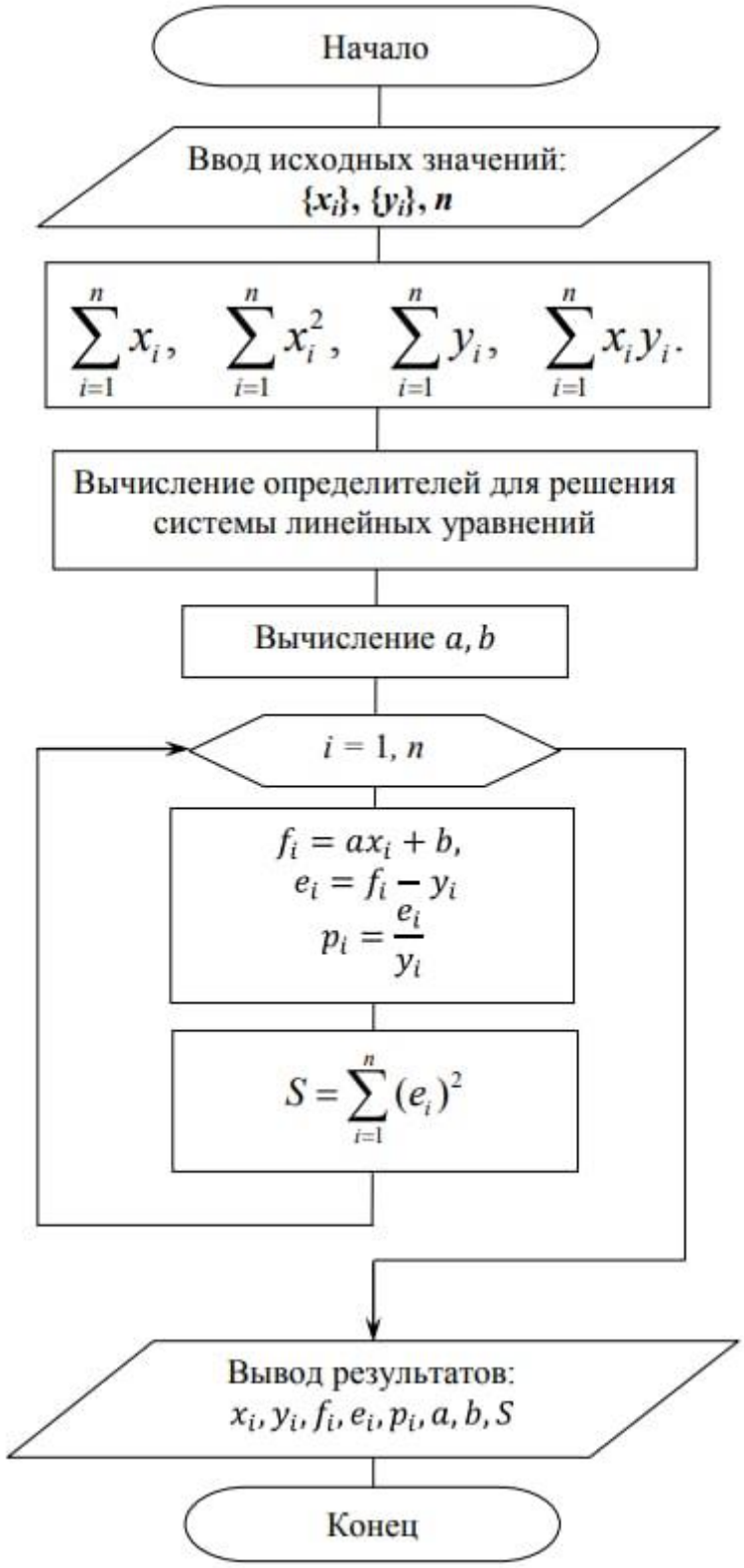
Вариант: аппроксимация методом наименьших квадратов

Выполнил:
студент группы P3231
Нестеров Иван Алексеевич

Преподаватель:
Перл Ольга Вячеславовна

г. Санкт-Петербург
2022 г.

Блок-схема указанного метода:



Описание метода:

Представим, что есть некоторый набор точек с координатами x и y. Можно представить, что есть некоторая функция f(x), выражающая зависимость

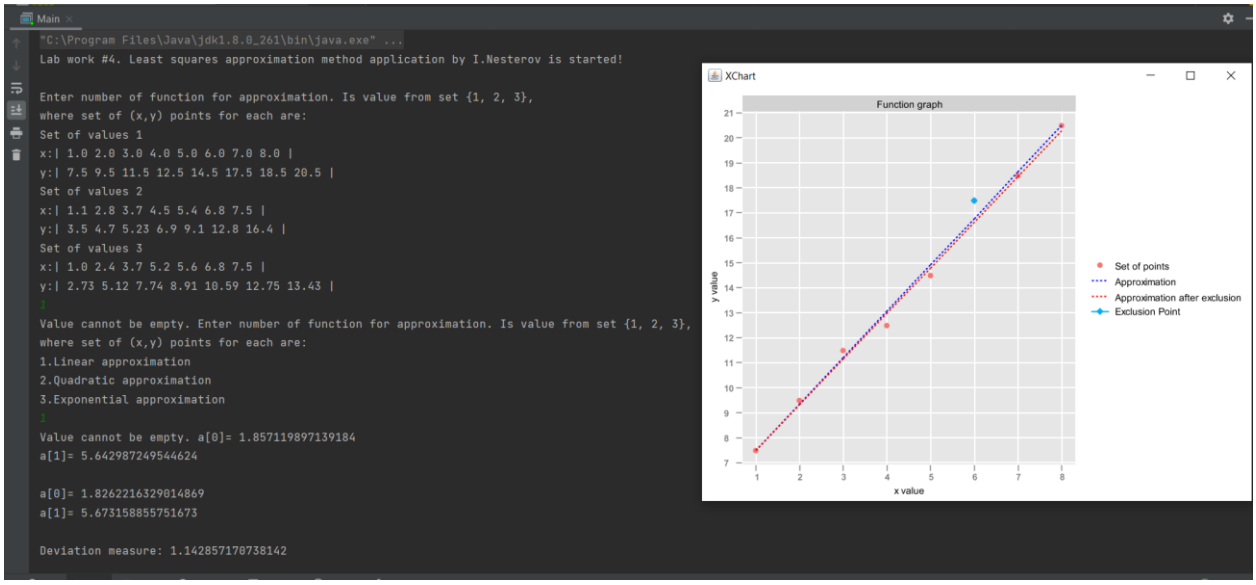
между x и y из ранее определённого набора. Но эта функция нам неизвестна. Однако можно все же "придумать" такую функцию, то есть аппроксимировать зависимость этой самой функцией. Разумеется, идеально подобную функцию подобрать нельзя. Однако представляется возможным подобрать такую, что набор точек, принадлежащей этой функции будет схож с набором данных точек. Метод же наименьших квадратов здесь выступает одним из вариантов «минимизации» этой разницы. В сущности, речь идет о решение переопределенной системы уравнений $f_i(x) = y_i$ и стремлении максимально «сблизить» значения левой и правой части такого уравнения, то есть сделать так, чтобы $\sum_i e_i^2 = \sum_i (y_i - f_i(x))^2 \to \min_x$. Вышеописанная же аппроксимация в коде была реализована на базе нескольких различных по своей природе функций. Программа выводит требуемые графики, используя для их построения средства встроенной в Java SE графической библиотеки AWT а так же сторонней библиотеки XChart.

Листинг вычислительного метода:.

```
private static void approximationRunner(Function function, Approximation
approximation) {
    Function firstApproximation = approximation.approximation(function,
approximation);
    Point max = findPointWithMaxDifference(function, firstApproximation);
    Function functionAfterExclusion = functionAfterExclusion(function, max);
    Function approximationAfterExclusion =
approximation.approximation(functionAfterExclusion, approximation);
    IOManipulator.printDeviation(getDeviation(getDifference(function,
firstApproximation)));
    Graphics.drawGraphic(function, firstApproximation,
approximationAfterExclusion, max);
}
private static List<Double> getDifference(Function function, Function
approximation) {
    List<Double> functionPoints = function.getPoints()
            .stream()
            .map(Point::getY)
            .collect(Collectors.toList());
    double[] newY = approximation.getArrayY();
    List<Double> difference = new ArrayList<>();
    for (int i = 0; i < newY.length; i++) {
        difference.add(newY[i] - functionPoints.get(i));
    }
    return difference;
}
public static Point findPointWithMaxDifference(Function function, Function
approximation) {
    double max = 0;
    double x = 0;
    List<Point> points = function.getPoints();
    double[] newY = approximation.getArrayY();
    for (int i = 0; i < points.size(); i++) {
        double y = points.get(i).getY();
        if (Math.abs(newY[i] - y) > max) {
            x = points.get(i).getX();
            max = Math.abs(newY[i] - y);
        }
```

```
    }
    double finalX = x;
    return points.stream()
            .filter(point -> (point.getX() == finalX))
            .findFirst()
            .get();
}
private static double getDeviation(List<Double> difference) {
    return difference.stream()
            .mapToDouble(dig -> dig * dig)
            .sum();
}
private static Function functionAfterExclusion(Function function, Point
point) {
    List<Point> pointsAfterExclusion = function.getPoints().stream()
            .filter(entity -> entity.getX() != point.getX())
            .collect(Collectors.toList());
    return new Function(pointsAfterExclusion);
}
```

Демонстрация работы программы:



Вывод:

Реализованы методы аппроксимации при помощи линейной, экспоненциальной и квадратичной функции, на примере графиков, которые демонстрирует программа, показана незначительная разница в получаемых ими результатами, а так же показана визуально и численно точность аппроксимации методом наименьших квадратов.