Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1
Аппроксимация и интерполяция

по курсу «Вычислительной математики»

Выполнил:
Студент группы P3230
Пономаренко Алиса Валерьевна

Преподаватель:
Перл Ольга Вячеславовна

Санкт-Петербург
2024

Описание численного метода

Интерполяция методом Ньютона

Метод интерполяции Ньютона - это способ аппроксимации функции с использованием многочлена. Он базируется на интерполяционном полиноме Ньютона.

Пусть у нас есть набор данных $(x_0, y_0), (x_1, y_1), ...(x_n, y_n)$

Интерполяционный полином Ньютона для этого набора данных представляет собой сумму слагаемых, каждое из которых содержит произведение разностей между точками x. Коэффициенты этого полинома вычисляются с помощью разделенных разностей.

Разделенная разность для заданного набора точек определяется рекурсивно. Используя её, можно вычислить все необходимые коэффициенты для полинома.

Основное преимущество метода интерполяции Ньютона - его эффективность и способность адаптироваться к новым данным - не придется пересчитывать все коэффициенты заново при добавлении новых значений. Однако этот метод сильно подвержен шумам.

Существуют формулы для равноотстоящих и неравноотстоящих узлов, но первый - является частным случаем второго, поэтому будем организовывать его.

Рассчитывается полином так: Для каждого i-го слагаемого, начиная с 1 рассчитывается произведение i-1 разностей вида $x - x_k$, где k изменяется от 0 до i - 2, а также разделенная разность.

Разделенная разность - производная для дискретного набора точек, представляет собой отношение изменения значения функции к изменению аргумента между двумя соседними точками набора. Это позволяет оценить скорость изменения функции в определенных точках набора данных.

Получаем формулу для разделенной разности:

$$f(x_j;\ x_{j+1};\ \ldots;\ x_{j+k-1};\ x_{j+k}) = \frac{f(x_{j+1};\ \ldots;\ x_{j+k-1};\ x_{j+k}) - f(x_j;\ x_{j+1};\ \ldots;\ x_{j+k-1})}{x_{j+k} - x_j},$$

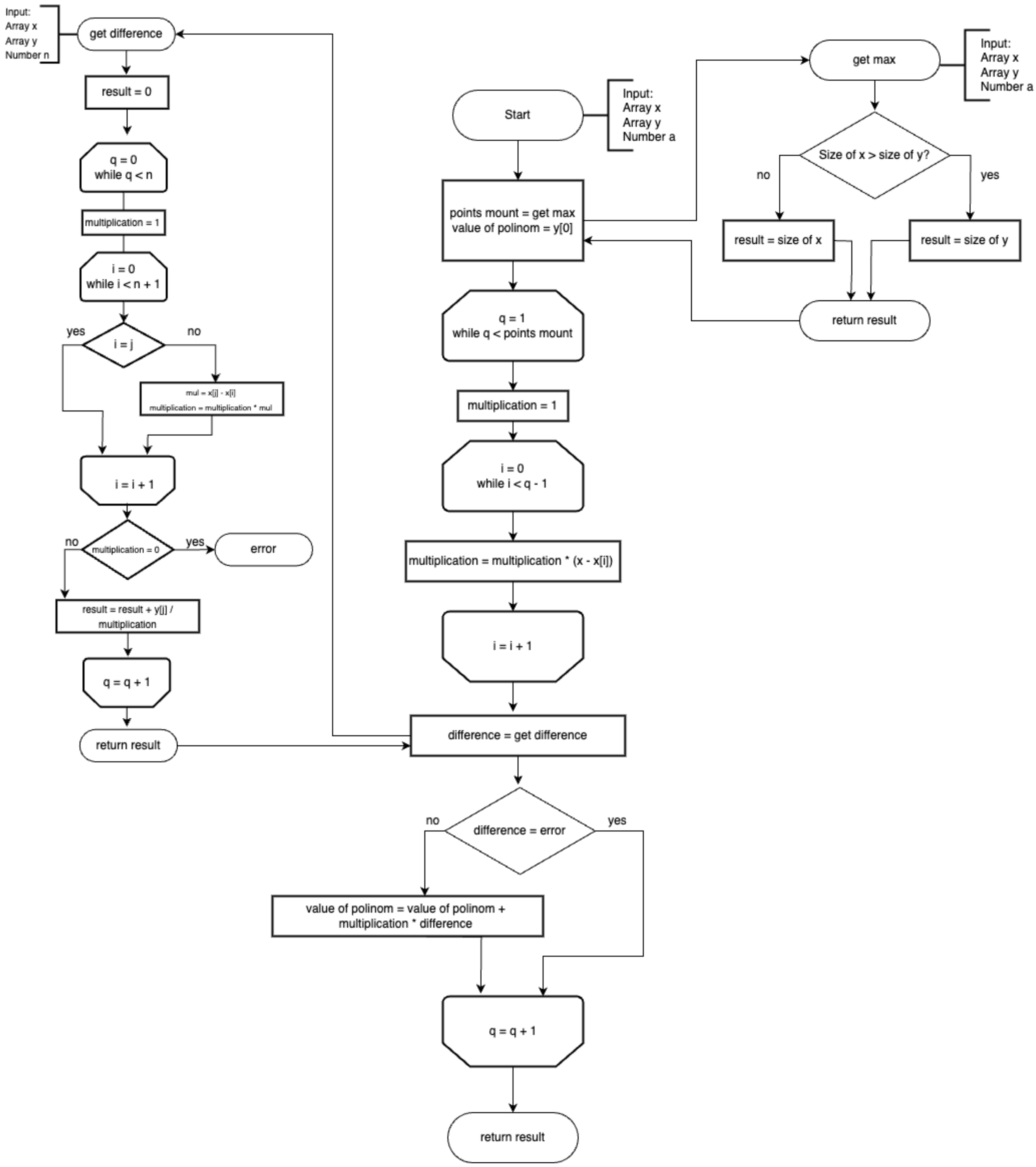
Итого формула полинома Ньютона:

$$P_n(x) = f(x_0) + (x - x_0)f(x_0; x_1) + (x - x_0)(x - x_1)f(x_0; x_1; x_2) + \ldots$$

$$+(x - x_0) \ldots (x - x_{n-1})f(x_0; \ldots; x_n),$$

где $f(x_0; \ldots; x_n)$ - разделенная разность

Блок-схема по составленному описанию метода



1 Код численного метода

```java
import java.util.List;

1 usage  Alice Ponomarenko *
class Result {

    /*
     * Complete the 'interpolate_by_newton' function below.
     *
     * The function is expected to return a DOUBLE.
     * The function accepts following parameters:
     *  1. DOUBLE_ARRAY x_axis
     *  2. DOUBLE_ARRAY y_axis
     *  3. DOUBLE x
     */

    1 usage  Alice Ponomarenko *
    public static double interpolate_by_newton(List<Double> x_axis, List<Double> y_axis, double x) {
        int pointsMount = getValidateInputCount(x_axis, y_axis);
        double valueOfPolinom = y_axis.get(0);
        for (int q = 1; q < pointsMount; q++) {
            double multiplication = 1.0d;
            for (int i = 0; i <= q - 1; i++) {
                multiplication *= (x - x_axis.get(i));
            }
            try {
                valueOfPolinom += getDifference(x_axis, y_axis, q) * multiplication;
            } catch (DivisionByZeroException exp) {
                return valueOfPolinom;
            }
        }
        return valueOfPolinom;
    }

    /**
     * @param x_axis Список аргументов
     * @param y_axis Список значений функции соответствующих аргументов
     * @param n      количество точек (т.е. для разделенной разности f(x0, x1, ... xn).
     * @return коэффициент полинома Ньютона - разделенную разность для n точек.
     */
    1 usage  Alice Ponomarenko *
    private static double getDifference(List<Double> x_axis, List<Double> y_axis, int n)
            throws DivisionByZeroException {
        double resultDifference = 0.0d;
        for (int j = 0; j < n + 1; j++) {
            double multiplication = 1.0d;
            for (int i = 0; i < n + 1; i++) {
```

```java
                if (i != j) {
                    double mul = x_axis.get(j) - x_axis.get(i);
                    multiplication *= (mul);
                }
            }
            if (multiplication != 0) {
                resultDifference += (y_axis.get(j) / multiplication);
            } else throw new DivisionByZeroException();

        }
        return resultDifference;
    }

    /**
     * @param x_axis array of x-values of points.
     * @param y_axis array of y-values of points.
     * @return minimum of count of arrays' values. It must have the same quantity.
     */
    1 usage   new *
    private static int getValidateInputCount(List<Double> x_axis, List<Double> y_axis) {
        return Math.min(x_axis.size(), y_axis.size());
    }
}

/**
 * Result will be calculated until the invalid point.
 */
3 usages   Alice Ponomarenko
class DivisionByZeroException extends RuntimeException {
    Alice Ponomarenko
    @Override
    public String getMessage() { return "Invalid input - two points with the same x coordinates"; }
}
```

2 Примеры работы программы

1. Входные данные: (целые значение)

1 2 3

3 4 5

4

Выходные данные:

6.0

2. Входные данные: (дробные значение)

2.2 3.7 3.4

3 4.6 2.2

3.9

Выходные данные:

6.7777777777777715

3. Входные данные: (единственное значение)

1

2

3

Выходные данные:

2.0

4. Входные данные: (разное количество значений по x и y)

2 3

2 3 4

6

Выходные данные:

6.0

5. Входные данные: (значение аргумента искомой точки есть в массиве данных точек)

1 2 3

3 4 5

2

Выходные данные:

4.0

3 Выводы

Результаты запуска реализованного метода на различных данных
1. Метод правильно отрабатывает как и с целыми входными данными, так и с числами с дробной частью, потому что в коде вычисления производятся в типе double, а в java есть приведение типов из double в int.

2. Крайний случай с единственной входной точкой - получим полином степени 1, графико которого будет прямая и значение в искомой точке так де будет посчитано верно.

3. Разное количество значений по x и y - программа обрабатывает эту ситуацию и берет за общее количество элементов - меньшее по количеству.

4. Значение аргумента искомой точки есть в массиве данных точек - значение будет рассчитано до момента встречи одинаковых аргументов. Этот случай является частным, потому что обнулется одно из выражений $x - x_i$, что приводит к делению на 0 при вычислении полинома.

Сравнение с другими методами интеполяции:

Метод Лагранжа
Принцип: Использует полиномы Лагранжа для интерполяции функции. Преимущества: Прост в реализации, обеспечивает глобальную интерполяцию. Недостатки: Требует вычисления всех базисных полиномов, что может быть вычислительно затратно при большом количестве точек.

Сплайн-интерполяция
Принцип: Использует кусочно-полиномиальные функции (сплайны) для приближения функции на интервале. Преимущества: Обеспечивает гладкость интерполяции, предотвращает осцилляции. Недостатки: Более сложный в реализации, требует определения дополнительных условий на границах интервалов.

Кубическая интерполяция
Принцип: Использует кубические полиномы для интерполяции между точками. Преимущества: Предоставляет гладкую и дифференцируемую интерполяцию. Недостатки: Может привести к осцилляциям на некоторых интервалах, требует решения системы уравнений для определения коэффициентов полиномов.

Сравнение с методом интерполяции Ньютона:
Преимущества метода Ньютона: Простота в вычислениях, высокая точность интерполяции, возможность добавления новых точек без пересчета всего полинома. Недостатки метода Ньютона: При добавлении новых точек необходимо пересчитывать

разделенные разности, возможны численные ошибки из-за подсчета больших разделенных разностей.

Анализ применимости метода:

Метод интерполяции Ньютона идеально подходит в следующих случаях:
Малое количество точек (при интерполяции набора данных с небольшим числом точек метод Ньютона предоставляет высокую точность приближения с минимальными вычислительными затратами).

Динамически изменяющиеся наборы данных (если набор данных часто изменяется (добавление новых точек), метод Ньютона позволяет добавлять новые точки с минимальными вычислительными затратами).

Метод интерполяции Ньютона может быть менее подходящим в следующих случаях:
Большое количество точек (при интерполяции набора данных с большим числом точек метод Ньютона может столкнуться с проблемами из-за больших разделенных разностей и ошибок округления).

Высокие требования к скорости работы (в ситуациях, где требуется максимальная скорость выполнения вычислений, более простые методы интерполяции или специализированные алгоритмы могут быть более предпочтительными).

Алгоритмическая сложность метода: $O(N^3)$

Анализ численных ошибок метода интерполяции Ньютона:
1. Ошибки округления
2. Погрешности в данных
3. Сложность полинома

Вариации по уменьшению численных ошибки:
1. Использование методов стабилизации: дополнительные методы, такие как интерполяция сглаженных кривых или сплайн-интерполяция, могут помочь уменьшить влияние шума и погрешностей в данных.