Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа №4

По дисциплине “Вычислительная математика”

Вариант – “Метод Ньютона”

Группа: P32312

Выполнил: Обляшевский С.А.

Преподаватель:

Перл О. В.

Санкт-Петербург, 2023

Описание метода, расчетные формулы:

Интерполяция - нахождение неизвестных промежуточных значений некоторой функции, по имеющемуся дискретному набору её известных значений, определенным способом.

В данной лабораторной работе я рассмотрел 2 полиномиальных интерполяционных метода:

1) Метод Ньютона.
2) Метод Лагранжа.

Метод ньютона:

В основе лежит интерполяционный многочлен Ньютона степени n, составляющийся из n+1 точек. Если мы имеет несколько точек $x_1, x_2, x_3 \dots x_{n+1}$ и соответствующие им значения функции $f(x_1), f(x_2) \dots f(x_{n+1})$, то многочлен выглядит следующим образом:

$$P_n(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + b_3(x - x_0)(x - x_1)(x - x_2) + \dots + b_n(x - x_0)\dots(x - x_n). \quad (1)$$

Чтобы найти коэффициенты b, последовательно подставим в формулу $x_0, x_1 \dots x_{(n+1)}$ и получим следующую треугольную систему уравнений:

$$\begin{cases} y_0 = b_0, \\ y_1 = b_0 + b_1(x_1 - x_0), \\ y_2 = b_0 + b_1(x_2 - x_0) + b_2(x_2 - x_0)(x_2 - x_1), \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots, \\ y_{n+1} = b_0 + b_1(x_{n+1} - x_0) + \dots + b_n(x_{n+1} - x_0) \cdot \dots \cdot (x_{n+1} - x_n) \end{cases}$$

Из которой можно легко последовательно вычислить все коэффициенты b.

Метод Лагранжа:

Данный метод также является полиномиальным, в его основе лежит интерполяционный многочлен Лагранжа степени n, также строящийся из n+1 точки. Он имеет следующий вид:

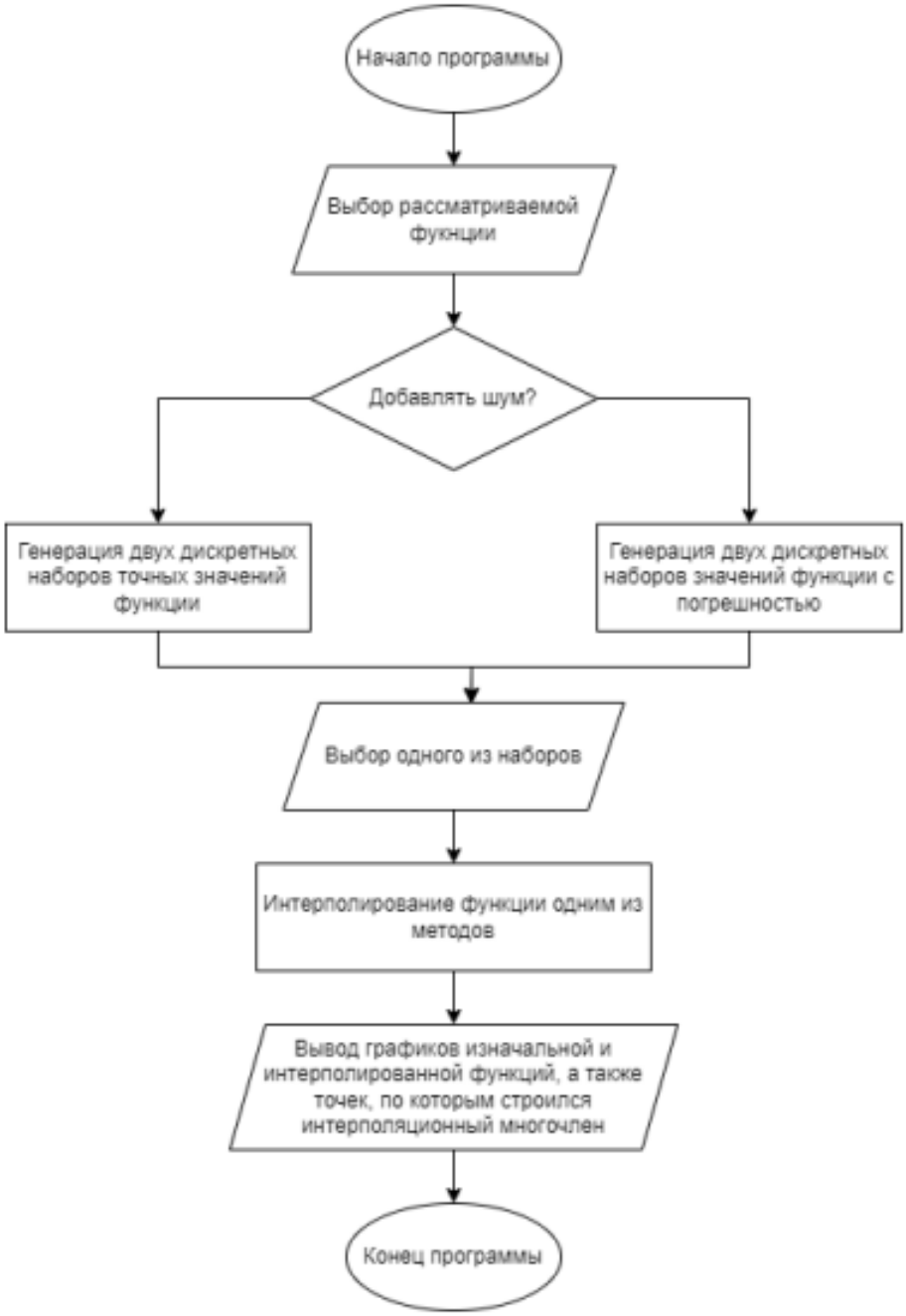
$$L(x) = \sum_{i=0}^{n} y_i l_i(x)$$

Где y_i – значение функции в I точке, а l_i – базисный полином, строящийся следующим образом:

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} = \frac{x - x_0}{x_i - x_0} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n}$$

Видно, что для любых точек $x_0 \dots x_{(n+1)}$ $l_i(x_j) = \begin{cases} 0, i \neq j \\ 1, i = j \end{cases}$

Блок – схема:


Начало программы
Выбор рассматриваемой фукнции
Добавлять шум?
Генерация двух дискретных наборов точных значений функции
Генерация двух дискретных наборов значений функции с погрешностью
Выбор одного из наборов
Интерполирование функции одним из методов
Вывод графиков изначальной и интерполированной функций, а также точек, по которым строился интерполяционный многочлен
Конец программы

Листинг численного метода:

Метод Ньютона:

```
@Override
public List<Dot> interpolate(List<Dot> startDots){
    int count = startDots.size();
    double[] coefficients = computeCoefficients(startDots);
    double start = startDots.get(0).getX();
    double end = startDots.get(startDots.size() - 1).getX();
    List<Dot> interpolatedDots = new ArrayList<>();
    for (double i = start; i <= end; i += 0.01){
        double y = coefficients[count - 1];
        for (int j = count - 2; j >= 0; j--) {
            y = y * (i - startDots.get(j).getX()) + coefficients[j];
        }
        interpolatedDots.add(new Dot(i, y));
    }
    return interpolatedDots;
}

@Override
protected double[] computeCoefficients(List<Dot> startDots) {
    int count = startDots.size();
    double[] coefficients = new double[count];
    for (int i = 0; i < count; i++) {
        coefficients[i] = startDots.get(i).getY();
    }
    for (int j = 1; j < count; j++) {
        for (int i = count - 1; i >= j; i--) {
            coefficients[i] = (coefficients[i] - coefficients[i - 1]) /
(startDots.get(i).getX() - startDots.get(i - j).getX());
        }
    }
    return coefficients;
}
```

Метод Лагранжа:

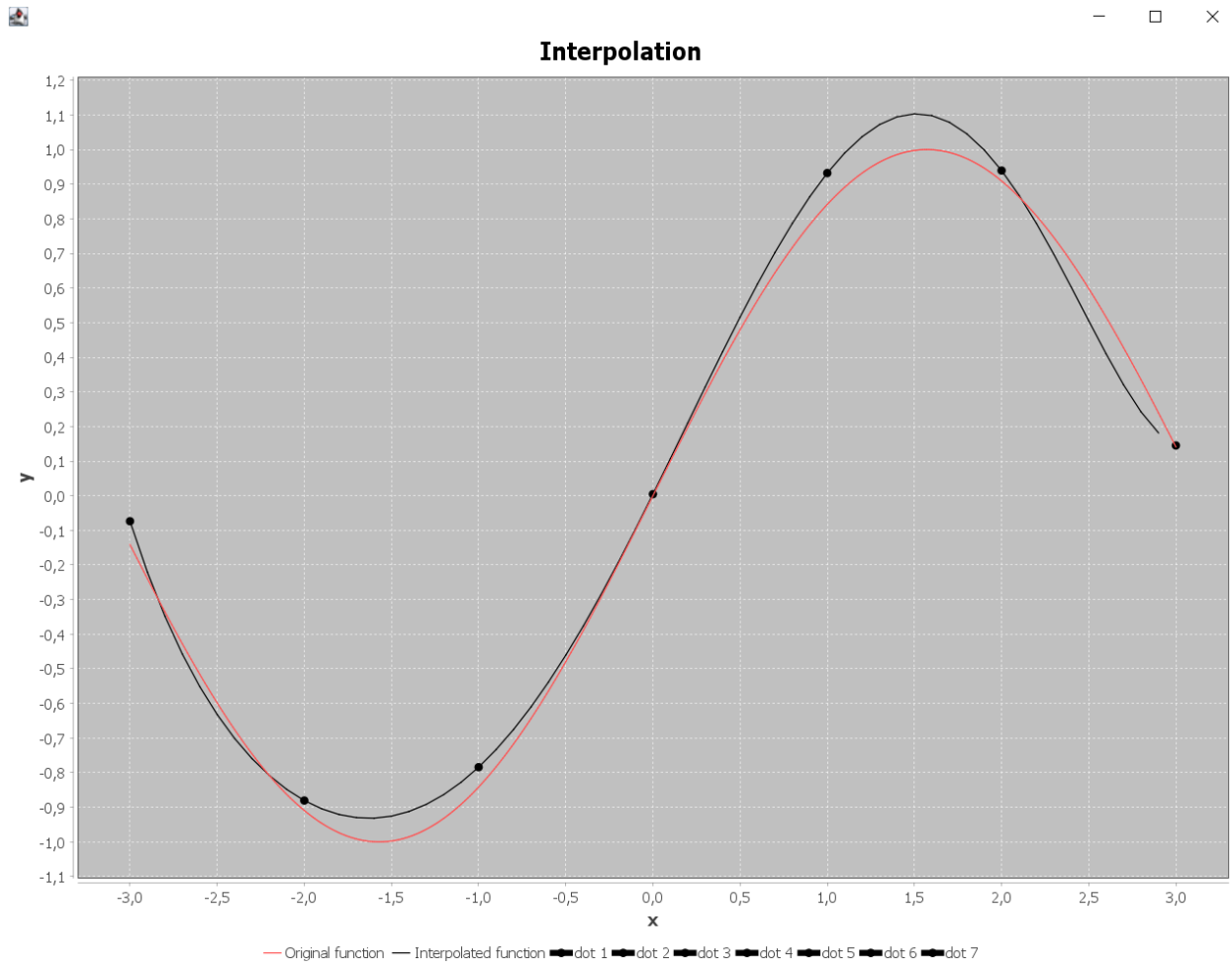
```
@Override
public List<Dot> interpolate(List<Dot> startDots) {
    List<Dot> interpolatedDots = new ArrayList<>();
    for (double i = startDots.get(0).getX(); i <
startDots.get(startDots.size() - 1).getX(); i += 0.1){
        double y = 0;
        for (int j = 0; j < startDots.size(); j++){
            double l = 1;
            for (int k = 0; k < startDots.size(); k++){
                if (j != k){
                    l *= (i -
startDots.get(k).getX())/(startDots.get(j).getX() - startDots.get(k).getX());
                }
            }
            y += l * startDots.get(j).getY();
        }
        interpolatedDots.add(new Dot(i, y));
    }
    return interpolatedDots;
}
```

Примеры и рез-ты работы:

Функция с шумом:

```
Выберите функцию:
1. y = x^2 - x + 1
2. y = e^x
3. y = sin(x)
1
Добавить шум?
1 - Да,
2 - Нет.
1
Выберите список начальных точек:
1) [(-3.0, -0.07361446521752764), (-2.0, -0.88048339300107), (-1.0, -0.7841897129979353), (0.0, 0.004631088990925647), (1.0, 0.9320942142310893), (2.0,
 0.9393763009592246), (3.0, 0.14515989627554288)]
2) [(0.0, 0.05788792739667535), (1.0, 0.8868614119219271), (2.0, 0.9214204687637355), (3.0, 0.18339875784321968), (4.0, -0.7104533293819758), (5.0,
 -0.8957165582462988)]
1
```

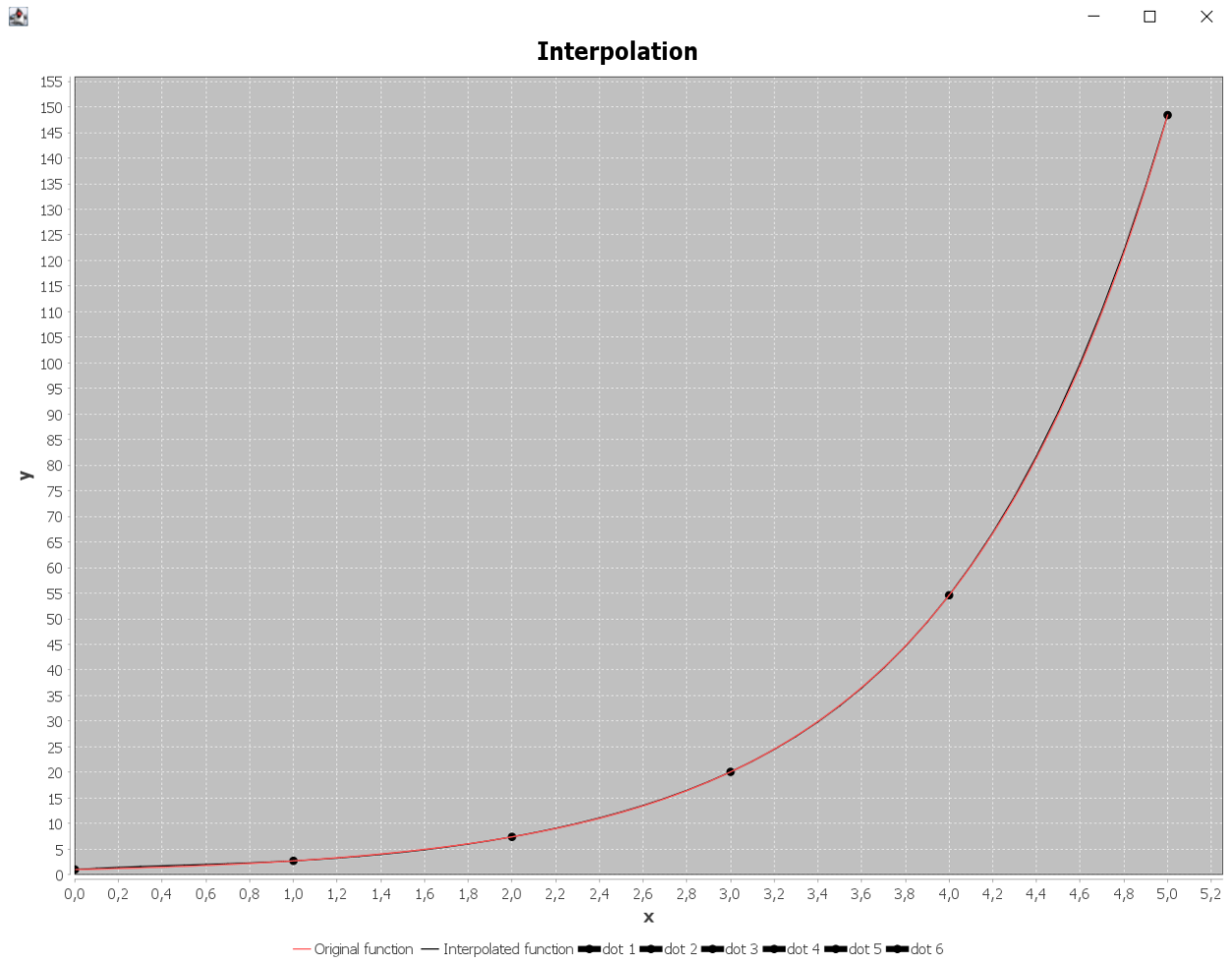
Результат:



Функция без шума:

```
1. y = x^2 - x + 1
2. y = e^x
3. y = sin(x)
2
Добавить шум?
1 - Да,
2 - Нет.
2
Выберите список начальных точек:
1) [(-3.0, 0.04978706836786395), (-2.0, 0.1353352832366127), (-1.0, 0.36787944117144233), (0.0, 1.0), (1.0, 2.718281828459045), (2.0, 7.3890560989306495),
 (3.0, 20.085536923187664)]
2) [(0.0, 1.0), (1.0, 2.718281828459045), (2.0, 7.3890560989306495), (3.0, 20.085536923187664), (4.0, 54.59815003314423), (5.0, 148.41315910257657)]
2
```

Результат:



Вывод:

Я рассмотрел интерполирование функций на примере полиномиальных методов Лагранжа и Ньютона.

Эти методы являются классическими, и главное их достоинство, что им не нужно большое кол-во точек для построения функций кривых (например, параболы). Можно отметить такие недостатки, как возрастающая сложность при увеличении кол-ва точек, а также большое влияние шума на результат работы методов.

Есть у методов Ньютона и Лагранжа также различия, например: при добавлении новых точек полином Лагранжа нужно будет считать заново, а полином Ньютона – нет, но при этом полином Лагранжа легче реализовать, в чем я убедился на собственном опыте.