Университет ИТМО

Факультет программной
инженерии и компьютерной техники
Вычислительная математика

Лабораторная работа № 4

Интерполяция по равноотстоящим узлам
(Метод Ньютона)

Преподаватель: Перл Ольга Вячеславовна
Выполнил: Геллер Л. А.
Группа: P3230

Санкт-Петербург, 2021

Описание метода, расчетные формулы

Решение нелинейных уравнений

1) Задана таблица [2 x n] с n равноотстоящими друг от друга точками и значениями некоторой функции f(x)ы в них
2) Заполняем таблицу конечных разностей (y_i = значение функции в i-ой точке в таблице):

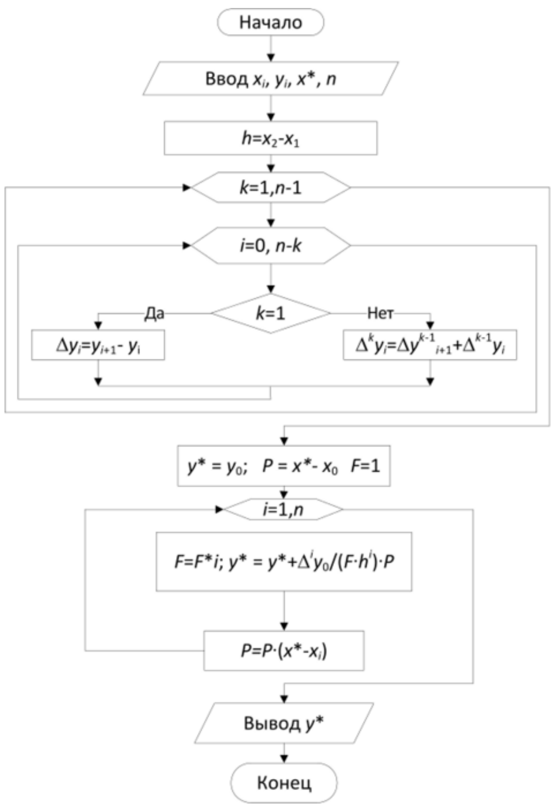
	0	1	2	3	…	n-3	n-2	n-1	n
y	y_0	y_1	y_2	y_3	…	y_{n-3}	y_{n-2}	y_{n-1}	y_n
Δy	$y_1 - y_0$	$y_2 - y_1$	$y_3 - y_2$	$y_4 - y_3$	…	$y_{n-2} - y_{n-3}$	$y_{n-1} - y_{n-2}$	$y_n - y_{n-1}$	
$\Delta^2 y$	$\Delta y_1 - \Delta y_0$	$\Delta y_2 - \Delta y_1$	$\Delta y_3 - \Delta y_2$	$\Delta y_4 - \Delta y_3$	…	$\Delta y_{n-2} - \Delta y_{n-3}$	$\Delta y_{n-1} - \Delta y_{n-2}$		
$\Delta^3 y$	$\Delta^2 y_1 - \Delta^2 y_0$	$\Delta^2 y_2 - \Delta^2 y_1$	$\Delta^2 y_3 - \Delta^2 y_2$	$\Delta^2 y_4 - \Delta^2 y_3$	…	$\Delta^2 y_{n-2} - \Delta^2 y_{n-3}$			
…	…	…	…	…	…				
$\Delta^n y$	$\Delta^{n-1} y_1 - \Delta^{n-1} y_0$								

3) Полином Ньютона строится по формуле:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \\ + a_3(x - x_0)(x - x_1)(x - x_2) + \ldots + a_n(x - x_0)\ldots(x - x_{n-1}).$$, где

a_i – коэффициенты полинома, вычисляются $a_i = \dfrac{\Delta^i(y_0)}{i!h^i}$. , h – расстояние между двумя точками

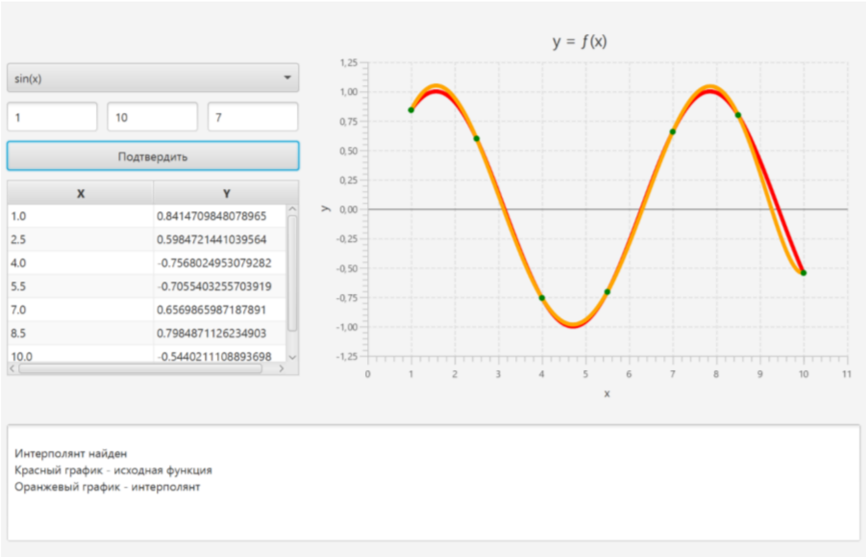
Блок-схема численного метода

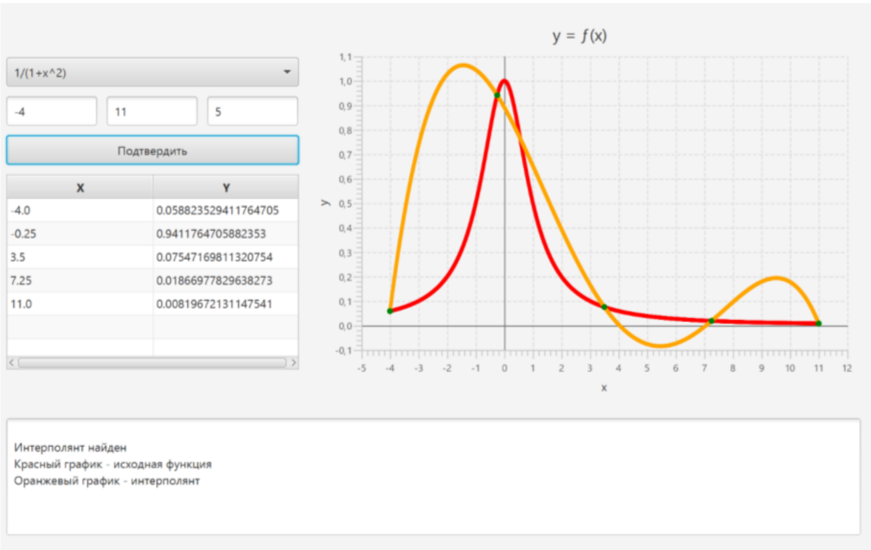


Листинг реализованного численного метода программы

```java
public DoubleFunction<Double> interpolate() {
    int n = values.size();
    double[][] finiteDifferences = new double[n][n];
    double[] a = new double[n];
    for (int i=0; i<n; i++) finiteDifferences[0][i] = values.get(i);
    for (int i=1; i<n; i++)
        for (int j=0; j<n-i; j++)
            finiteDifferences[i][j] = finiteDifferences[i-1][j+1] - finiteDifferences[i-1][j];

    for (int i=0; i<n; i++) a[i] = finiteDifferences[i][0]/(Math.pow(h, i) * factorial(i));

    return  x -> {
        double result = a[0];
        for (int i=1; i<n; i++) {
            double iterResult = a[i];
            for (int j=0; j<i; j++)  iterResult *= (x - points.get(j));
            result += iterResult;
        }
        return result;
    };
}

private double factorial(int bound) {
    int j, fact = 1;
    for(j=1; j<=bound; j++)
        fact = fact*j;
    return fact;
}
```

Примеры и результаты работы программы на разных данных





Вывод

Аппроксимация функции – нахождение функции, близкой исходной по опытным данным (значениям на наборе точек):

1 способ – аппроксимация интерполяционным многочленом (Лагранжа или Ньютона), интерполянт будет проходить через все точки опытных данных

2 способ – с помощью аппроксимирующего многочлена n-ой степени (методом наименьших квадратов), полученная функция не будет обязательно проходить через заданные узлы

Сравнение методов аппроксимации:

1) Интерполирование многочленом Лагранжа

   При добавлении нового узла необходимо пересчитать формулу заново, но метод применим к данным, где не все точки – равноудалённые

   Многочлен Лагранжа: $L_n(x) = \sum_{i=0}^{n}\left(y_i \prod_{j=0, j\neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)} \right).$

2) Интерполирование многочленом Ньютона

   Увеличение числа узлов на единицу требует добавления только одного слагаемого, применимо только если расстояния между двумя последовательными точками одинаковы

3) Интерполирование кубическими сплайнами

   Сплайн – функция, область определения которой разбита на конечное число отрезков, на каждом из которых она совпадает с некоторым полиномом (кубический сплайн -> степень полинома не превышает 3)

   Следовательно, степень многочлена не будет зависеть от количества точек

4) Аппроксимация методом наименьших квадратов

   Подходит когда не требуется точное прохождение аппроксимирующей функции через исходные точки. Необходим подбор общего вида формулы, дает большую погрешность на сложных функциях