СПБ НИУ ИТМО

«Университет информационных технологий,
механики и оптики»

Кафедра вычислительной техники

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Вариант: интерполирование многочленом Ньютона

Выполнил:

Ощепков А.А.
группа: P3202

Преподаватель:

Петрова М.М.

Санкт-Петербург 2017

Задание

Требуется написать программу, интерполирующую функцию, заданную таблично, другой функцией. Таблица должна строиться на основе выбранной пользователем функции.

Программно построить график, на котором одним цветом функция (sinx), а другим цветом полученный график в результате интерполяции. На графике должны быть отмечены сами точки (узлы) интерполяции.

Описание метода

Интерполяционный многочлен Ньютона позволяет интерполировать некоторую фунцию $f(x)$, заданную таблично. Для его построения необходимо ввести понятие "разделённая разность". Разделенные разности нулевого порядка совпадают со значениями функции в узлах. Разделенные разности первого порядка определяются через разделенные разности нулевого порядка:

$f(x_i, x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$

Разделенные разности второго порядка определяются через разделенные разности первого порядка:

$f(x_i, x_{i+1}, x_{i+2}) = \frac{f(x_{i+1}, x_{i+2}) - f(x_i, x_{i+1})}{x_{i+2} - x_i}$

И так далее.

Используя понятие разделенной разности, интерполяционный многочлен Ньютона можно записать в следующем виде:

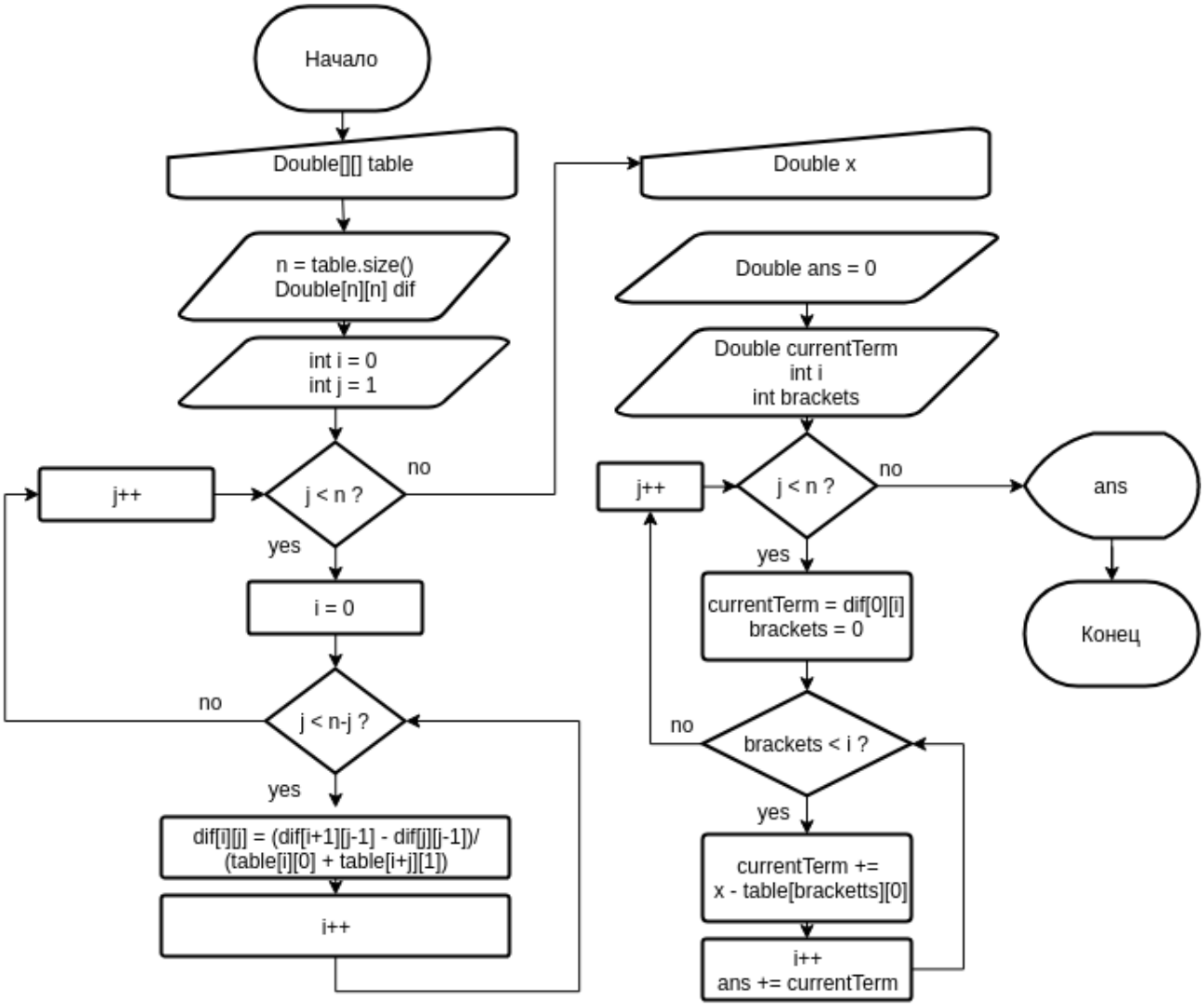
$P_n(x) = f(x_0) + f(x_0, x_1) \cdot (x - x_0) + f(x_0, x_1, x_2) \cdot (x - x_0) \cdot (x - x_1) + ... + f(x_0, x_1, ..., x_n) \cdot (x - x_0) \cdot (x - x_1) \cdot ... \cdot (x - x_{n-1})$

Текст программы

```
1  public class Approximation {
2      public static Function<Number,Number> getApproximateFunction(
       LinkedList<Pair<Double,Double>> table){
3          int n = table.size();
4          Double[][] dif = new Double[n][n];
5          for(int i = 0; i < n; i++){
6              dif[i][0] = table.get(i).getValue();
7          }
8          // Вычисляем конечные разности
9          for(int j = 1; j < n; j++)
10             for(int i = 0; i < n-j;i++){
11                 dif[i][j] = (dif[i+1][j-1] - dif[i][j-1])/
12                             ( -table.get(i).getKey() + table.get(i+j).
       getKey());
13             }
14         return new Function<Number, Number>() {
15             @Override
16             public Number apply(Number x) {
17                 // Многочлен Ньютона
18                 Double ans = 0.;
19                 Double currentTerm;
20                 for(int i = 0; i < n;i++) {
21                     currentTerm = dif[0][i];
22                     for (int brackets = 0; brackets < i; brackets++) {
```

```
23                          currentTerm *= (Double)x - table.get(brackets)
    .getKey();
24                      }
25                      ans += currentTerm;
26                  }
27
28                  return ans;
29              }
30          };
31      }
32  }
```

Блок-схема



Выводы

Мы взрослеем, когда осознаём простые факты, которые, вроде бы и слышали сотни раз. Во время выполнения этой лабораторной я понял, как сделать работу быстро: не торопясь. Вдумчиво, сосредоточенно, внимательно - качественно.

Интерполяция с помощью многочлена Ньютона удивила меня: я не ожидал увидеть столь близкой аппроксимации при таком малом дискретном наборе данных.