МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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 интерполяции кубическими сплайнами»

Студент:
Чернова Анна Ивановна

Группа Р32301

Преподаватель:

Перл Ольга Вячеславовна

Санкт-Петербург, 2023

1. Описание реализованного метода, расчетные формулы

Отрезок, на котором определена функция, разбивается на участки, содержащие малое число экспериментальных точек. На каждом участке строится многочлен третьей степени,

$$S_i(x) = a_i + b_i * (x_i - x_{i-1}) + c_i * (x_i - x_{i-1})^2 + d_i * (x_i - x_{i-1})^3, \qquad x_{i-1} \le x \le x_i$$

который в узлах интерполяции принимает значения интерполируемой функции и непрерывен вместе со своими производными $S'(x), S''(x)$ на отрезке $[x_0; x_n]$.

$$a_i = S_i(x_i) = y_i$$

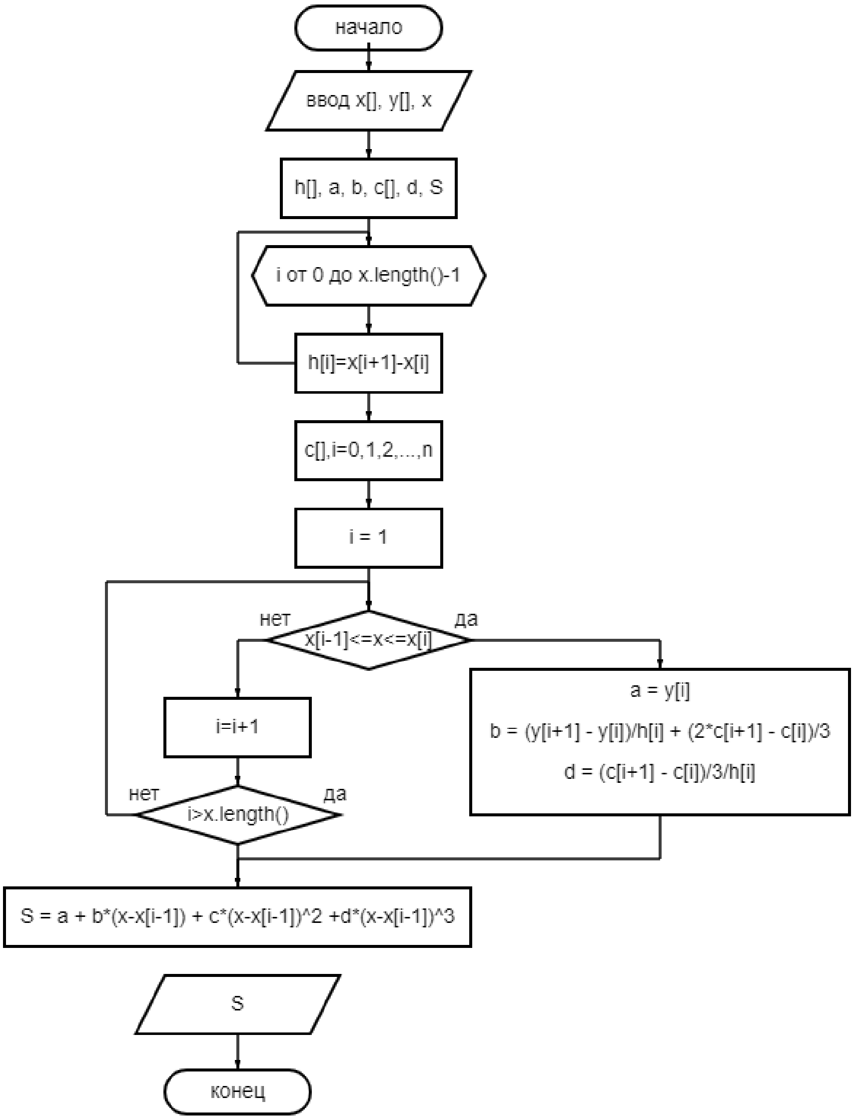
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ h_1 & \delta_1 & h_2 & 0 & 0 & 0 \\ 0 & h_2 & \delta_2 & h_3 & 0 & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & h_{n-1} & \delta_{n-1} & h_n \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \dots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ \varepsilon_1 \\ \varepsilon_2 \\ \dots \\ \varepsilon_{n-1} \\ 0 \end{bmatrix},$$

где $h_i = x - x_{i-1} \;\; i \in [1, n]$, $\delta_i = 2 * (h_i + h_{i+1})$, $\varepsilon_i = 3 * (\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i})$, $i \in [1, n-1]$

$$b_i = \frac{a_i - a_{i-1}}{h_i} + \frac{2 * c_i - c_{i-1}}{3} * h_i = \frac{y_i - y_{i-1}}{h_i} + \frac{2 * c_i - c_{i-1}}{3} * h_i$$

$$d_i = \frac{c_i - c_{i-1}}{3 * h_i}$$

2. Блок-схема численного метода



3. Листинг реализованного метода

```java
public double[] calcA() {
    double[] a = new double[xList.size()];
    for (int i = 0; i < a.length; i++) {
        a[i] = yList.get(i);
    }
    return a;
}


public double[] calcB(double[] h, double[] a, double[] c, double[] d) {
    double[] b = new double[xList.size() - 1];
    for (int i = 0; i < b.length; i++) {
        b[i] = (a[i + 1] - a[i]) / h[i] + h[i] * c[i + 1] - h[i] * h[i] * d[i];
    }
    return b;
}

public double[] calcC(double[][] matrix, double[] freeTerm) {
    double[] cArg = new double[xList.size()];
    if (Determinate.calDet(matrix, matrix.length) != 0 &&
IterativeMethod.checkDiagonallyDominant(freeTerm.length, matrix, freeTerm)) {
        IterativeMethod.normalize(freeTerm.length, matrix, freeTerm);
        cArg = IterativeMethod.calcRoots(freeTerm.length, matrix, freeTerm, 0.00001,
freeTerm.clone());
    }
    return cArg;
}

public double[] calcD(double[] h, double[] c) {
    double[] d = new double[xList.size() - 1];
    for (int i = 1; i < d.length + 1; i++) {
        d[i - 1] = (c[i] - c[i - 1]) / 3 / h[i - 1];
    }
    return d;
}

public double[] calcH() {
    double[] h = new double[xList.size() - 1];
    for (int i = 0; i < h.length; i++) {
        h[i] = xList.get(i + 1) - xList.get(i);
    }
    return h;
}

public double[][] calcMatrix(double[] h) {
    double[][] matrix = new double[xList.size()][xList.size()];
    matrix[0][0] = 1;
    matrix[xList.size() - 1][xList.size() - 1] = 1;
    for (int i = 1; i < matrix.length - 1; i++) {
        matrix[i][i - 1] = h[i - 1];
        matrix[i][i + 1] = h[i];
        matrix[i][i] = (h[i] + h[i - 1]) * 2;
    }
    return matrix;
}

public double[] calcFreeTerm(double[] h) {
    double[] freeTerm = new double[xList.size()];
    freeTerm[0] = 0;
    freeTerm[xList.size() - 1] = 0;
    for (int i = 1; i < xList.size() - 1; i++) {
        freeTerm[i] = 3 * ((yList.get(i + 1) - yList.get(i)) / h[i] - (yList.get(i) -
yList.get(i - 1)) / h[i - 1]);
    }
    return freeTerm;
}
```

4. Примеры результата работы программы

Выберите функцию из списка. Введите число от 1 до 5

1.

x | 1.2 | 2.9 | 4.1 | 5.5 | 6.7 | 7.8 | 9.2 | 10.3 |
y | 7.4 | 9.5 | 11.1 | 12.9 | 14.6 | 17.3 | 18.2 | 20.7 |

2.

x | 0.1 | 0.4 | 0.5 | 0.8 | 1.0 | 1.8 | 2.9 | 3.8 |
y | 8.3 | 6.8 | 5.2 | 3.6 | 2.0 | 0.6 | 0.4 | 0.2 |

3.

x | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 |
y | 1.15 | 1.6 | 1.79 | 2.29 | 2.6 | 3.13 | 3.5 | 4.18 |

4.

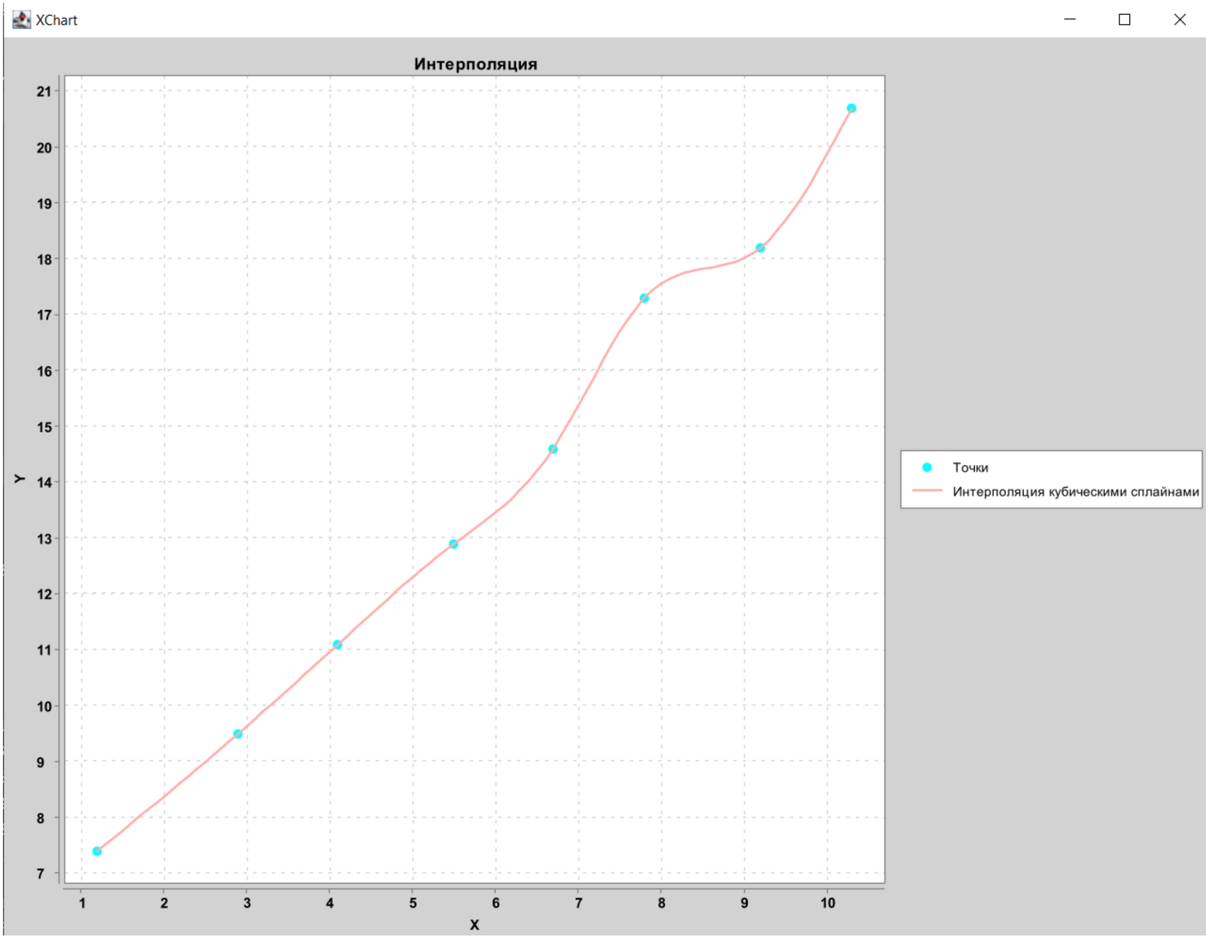
x | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 | 6.0 | 7.0 | 8.0 |
y | 2.0 | 6.0 | 7.0 | 13.0 | 26.0 | 42.0 | 64.0 | 110.0 |

5.

x | -1.0 | 0.0 | 1.0 | 2.0 |
y | 0.5 | 1.0 | 2.0 | 4.0 |

1

Коэффициенты функции рассчитаны методом интерполяции кубическими сплайнами
1.2 <= x < 2.9 | 9.5 | 1.2896604503424738 | 0.04797029355477801 | 0.009405939912701571 |
2.9 <= x < 4.1 | 11.1 | 1.3631196461903146 | 0.013247744293837536 | -0.009645152572483467 |
4.1 <= x < 5.5 | 12.9 | 1.1123499572291438 | -0.192371366952428 | -0.04895693124911084 |
5.5 <= x < 6.7 | 14.6 | 2.256157860428978 | 1.1455496756791037 | 0.3716447340643143 |
6.7 <= x < 7.8 | 17.3 | 1.5912081524962427 | -1.750052976997571 | -0.8774553492959624 |
7.8 <= x < 9.2 | 18.2 | 1.1962389876729032 | 1.467935607944244 | 0.7661877583194802 |
9.2 <= x < 10.3 | 20.7 | 2.810970328973493 | 0.0 | -0.4448289721043158 |



Выберите функцию из списка. Введите число от 1 до 5

1.

x | 1.2 | 2.9 | 4.1 | 5.5 | 6.7 | 7.8 | 9.2 | 10.3 |
y | 7.4 | 9.5 | 11.1 | 12.9 | 14.6 | 17.3 | 18.2 | 20.7 |

2.

x | 0.1 | 0.4 | 0.5 | 0.8 | 1.0 | 1.8 | 2.9 | 3.8 |
y | 8.3 | 6.8 | 5.2 | 3.6 | 2.0 | 0.6 | 0.4 | 0.2 |

3.

x | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 |
y | 1.15 | 1.6 | 1.79 | 2.29 | 2.6 | 3.13 | 3.5 | 4.18 |

4.

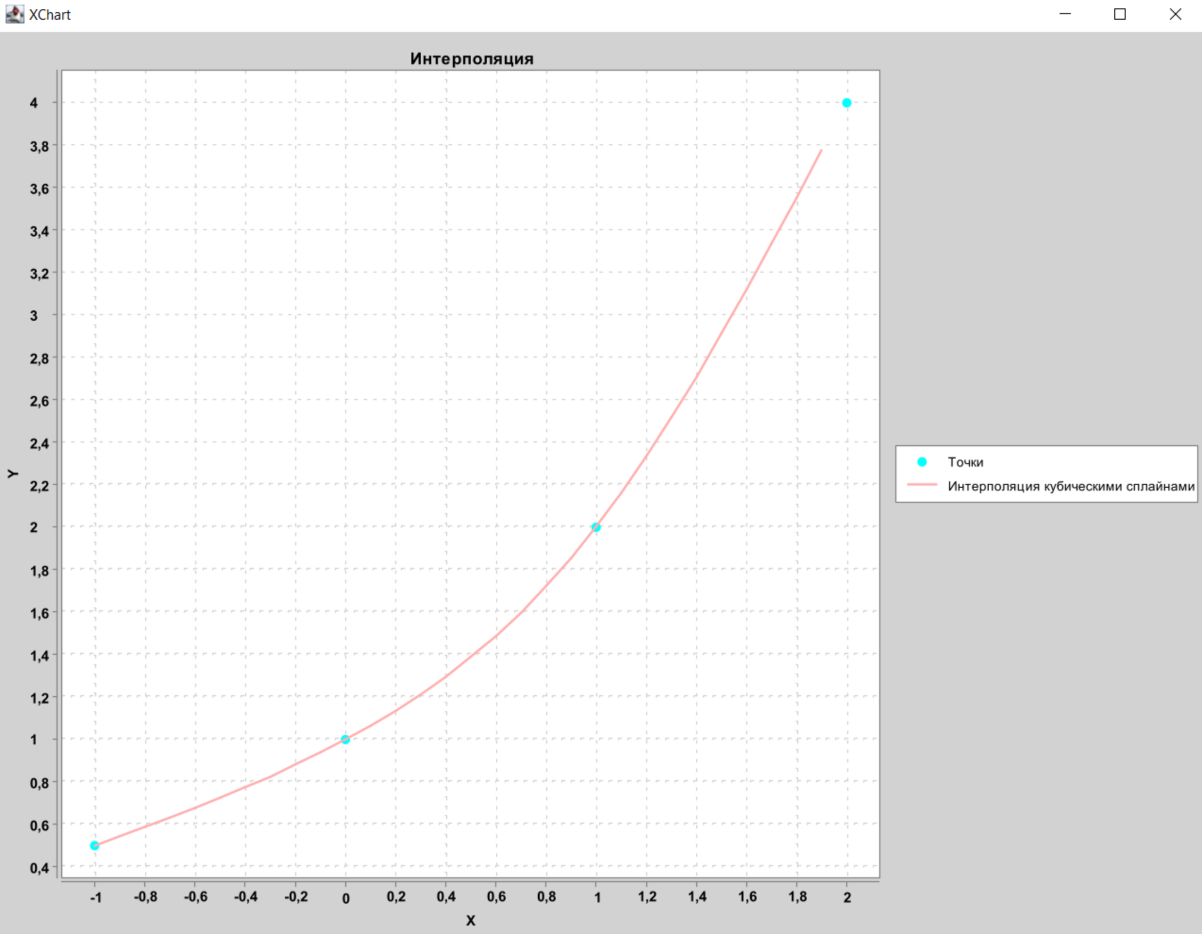
x | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 | 6.0 | 7.0 | 8.0 |
y | 2.0 | 6.0 | 7.0 | 13.0 | 26.0 | 42.0 | 64.0 | 110.0 |

5.

x | -1.0 | 0.0 | 1.0 | 2.0 |
y | 0.5 | 1.0 | 2.0 | 4.0 |

5

Коэффициенты функции рассчитаны методом интерполяции кубическими сплайнами
-1.0 <= x < 0.0 | 1.0 | 0.6333332061767578 | 0.19999980926513672 | 0.0666666030883789 |
0.0 <= x < 1.0 | 2.0 | 1.5333328247070312 | 0.6999993324279785 | 0.16666650772094727 |
1.0 <= x < 2.0 | 4.0 | 2.233333110809326 | 0.0 | -0.23333311080932617 |



4. Вывод

Интерполяция кубическими сплайнами - один из способов кусочно-полиноминальной интерполяции, когда весь отрезок разбивают на частичные отрезки и на каждом из частичных отрезков приближенно заменяют исходную функцию многочленом невысокой (в данном случае, третьей степени), в отличие от формул Ньютона и Лагранжа, где отрезок не разбивается.

Интерполяцию кубическими сплайнами рационально применять, если f(x) - периодическая или тригонометрическая функция. Что касается других методов интерполяции, а именно формул Ньютона и Лагранжа, то формулу Лагранжа можно применять для таблиц с различными расстояниями между узлами, а формулы Ньютона – только для таблиц с равноотстоящими узлами.

Формулы Ньютона имеют следующее преимущество перед формулой Лагранжа: добавление в таблицу узлов интерполяции при использовании формулы Лагранжа ведет к необходимости пересчета каждого коэффициента заново, тогда как при использовании формулы Ньютона достаточно добавить к существующему многочлену только одно слагаемое. Кроме того, по сравнению с этими методами большую точность интерполяции можно получить применением методов сплайн–интерполяции. Что касается сравнения с методом аппроксимации, то следует обратить внимание на разницу в постановке задач аппроксимации и интерполяции: интерполирующая функция должна принадлежать к определенному классу в точках $x_i(i = 0, 1, \ldots, n)$ принимать те же значения, что и исходная функция, для аппроксимирующей функции это требование обязательным не является, но должен выполняться критерий наилучшего приближения