21 вопрос

Интерполяционные формулы Ньютона — формулы вычислительной математики, применяющиеся для полиномиального интерполирования.

Если узлы интерполяции равноотстоящие и упорядочены по величине, так что $x_{i+1} - x_i = h = const$, т.е. $x_i = x_0 + ih$, то интерполяционный многочлен можно записать в форме Ньютона.

Интерполяционные полиномы в форме Ньютона удобно использовать, если точка интерполирования находится вблизи начала (**прямая формула Ньютона**) или конца таблицы (**обратная формула Ньютона**).

Короткая форма интерполяционной формулы Ньютона

В случае равноудалённых центров интерполяции, находящихся на единичном расстоянии друг от друга, справедлива формула:

$$\boldsymbol{P_n(x) = \sum_{m=0}^{n} C_k^m \sum_{k=0}^{m} (-1)^{m-k} C_m^k f(k)}$$

Где $\boldsymbol{C_k^m}$ **–** обобщенные на область действительных чисел биноминальные коэффициенты

Существуют две формулы Ньютона для случая равноотстоящих узлов интерполирования, которые называются соответственно первой и второй интерполяционными формулами Ньютона и имеют вид:

1. Прямая (или первая) интерполяционная формула Ньютона, применяется для интерполирования вперёд:

$$P_{1,n}(x) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \ldots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_0$$

Где $q = \frac{x-x_0}{h}$, $y_i = f_i$, а выражения $\Delta^k y_0$ – конечные разности.

2. Обратная (или вторая) интерполяционная формула Ньютона, применяется для интерполирования назад:

$$P_{2,n}(x) = y_n + q\Delta y_{n-1} + \frac{q(q+1)}{2!}\Delta^2 y_{n-2} + \ldots + \frac{q(q+1)\ldots(q+n-1)}{n!}\Delta^n y_0$$

где $q = \frac{x-x_n}{h}$

Погрешность:

Следующая формула представляет собой оценку погрешности интерполяции в общем случае:

$$|f(x) - L_n(x)| \le \frac{M_{n+1}}{(n+1)!}|(x-x_0)\cdot\ldots\cdot(x-x_n)|, \tag{11}$$

где $M_{n+1} = \max\limits_{[a;b]}|f^{(n+1)}(x)|$, а производная $f^{(n+1)}(x)$ непрерывна на отрезке

$[x_0;x_n]$

Код:

```
public class nutonbombom {
	public static void main(String[] args) {
		double x = 2.5;
		double step = 1;
		int n = 3;
		double[] MasX = new double[] { 0, 1, 2, 3 };
		double[] MasY = new double[] { 0, 1, 8, 27 };
		double Res = Newton(x, n, MasX, MasY, step);
		System.out.print("result = "+Res);
	}

	static public double Newton(double x, int n, double[] MasX, double[] MasY, double step)
{

		double[][] mas = new double[n + 2][ n + 1];
		for (int i = 0; i < 2; i++) {
			for (int j = 0; j < n + 1; j++) {
				if (i == 0) mas[i][j] = MasX[j];
				else if (i == 1) mas[i][j] = MasY[j];
			}
		}
		int m = n;
		for (int i = 2; i < n + 2; i++){
			for (int j = 0; j < m; j++){
				mas[i][j] = mas[i - 1][j + 1] - mas[i - 1][j];
			}
			m--;
		}

		double[] dy0 = new double[n + 1];

		for (int i = 0; i < n + 1; i++) {
			dy0[i] = mas[i + 1][0];
		}

		double res = dy0[0];
		double[] xn = new double[n];
		xn[0] = x - mas[0][0];

		for (int i = 1; i < n; i++) {
			double ans = xn[i - 1] * (x - mas[0][i]);
			xn[i] = ans;
			ans = 0;
		}

		int m1 = n + 1;
		int fact = 1;

		for (int i = 1; i < m1; i++) {
			fact = fact * i;
			res = res + (dy0[i] * xn[i - 1]) / (fact * Math.pow(step, i));
		}
		return res;

	}
}
```