Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет Программной инженерии и компьютерной техники

Кафедра информатики и прикладной математики

**Дисциплина «Вычислительная математика»**

**Лабораторная работа №3**

**Интерполяционный многочлен Ньютона**

Выполнил:

Съестов Дмитрий Вячеславович

Группа Р3217

Преподаватель:

Калёнова Ольга Вячеславовна

Санкт-Петербург, 2017 г.

**Описание метода**

Интерполяционная формула Ньютона – формула, применяющая для полиноминального интерполирования.

Пусть функция *{\displaystyle f}f(x)* задана на множестве *X* и{\displaystyle X,}XX фиксированы попарно различные точки *x0, x1, …, xn* ∈ *X*.{\displaystyle x\_{0},\;\ldots ,\;x\_{n}\in X.}xx

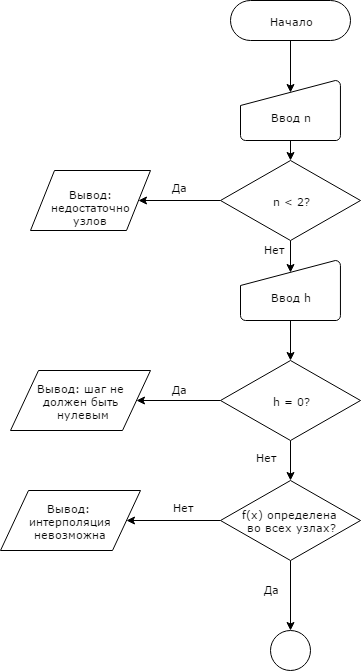
Тогда разделённой разностью нулевого порядка функции *{\displaystyle f}f* в точке *xj*{\displaystyle x\_{j}} называют значение *f(xj)* {\displaystyle f(x\_{j}),}f(а разделённую разность порядка{\displaystyle k} *k* для системы точек {\displaystyle (x\_{j},\;x\_{j+1},\;\ldots ,\;x\_{j+k})}определяют рекурсивно через разделённые разности порядка (*k* – 1) {\displaystyle (k-1)}по формуле

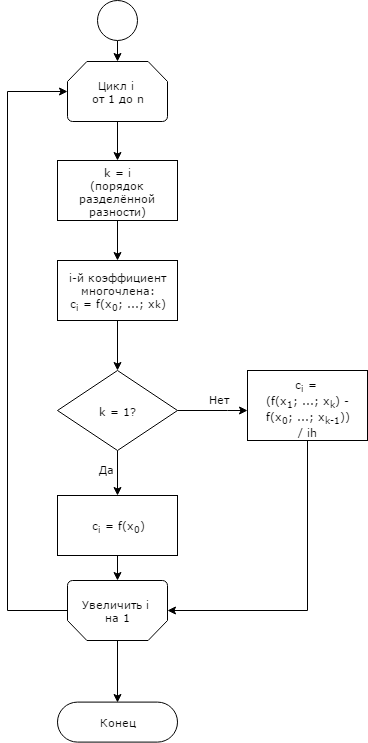
В частности,

С помощью разделённых разностей можно записать интерполяционный многочлен Ньютона «вперёд»:

И интерполяционный многочлен Ньютона «назад»:

**Блок-схема**

****

****

**Листинг программы**

//Разделённая разность (k - порядок)  
private static double difference(Function f, double x, double h, int k)  
{  
 Assert.GreaterOrEqual(k, 0);  
    if (k == 0) return f(x);  
    return (difference(f, x + h, h, k - 1) - difference(f, x, h, k - 1)) / (h \* k);  
}  
  
public static Polynomial Interpolate(Function f, double x0, double step, int nodeCount)  
{  
 if (f == null) throw new ArgumentNullException("Функция для интерполяции не задана.");  
    if (nodeCount <= 2) throw new ArgumentException("Для интерполяции необходимо как минимум два узла.");  
    if (step == 0) throw new ArgumentException("Шаг не должен быть равен нулю.");

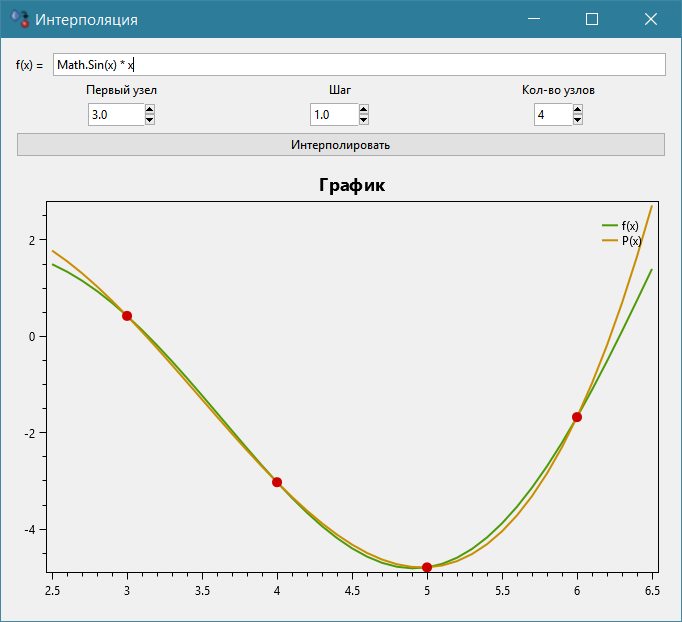
//Проверка, что функция определена во всех узлах  
    for (int i = 0; i < nodeCount; i++)  
    {  
     double yn = f(x0 + step \* i);  
        if (double.IsNaN(yn) || double.IsInfinity(yn))  
            throw new ArithmeticException("Функция должна быть определена во всех узлах.");  
   }

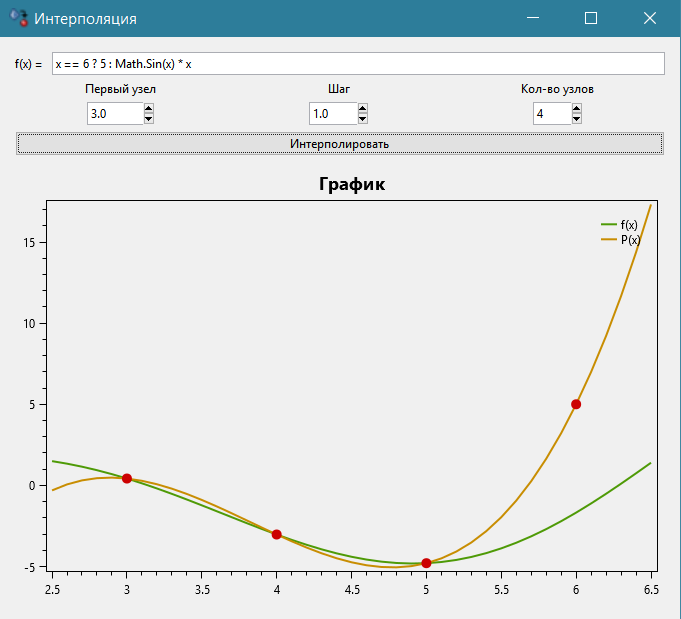
    var coefficients = new double[nodeCount];  
  
    for (int i = 0; i < nodeCount; i++)  
     coefficients[i] = difference(f, x0, step, i);  
  
    var polynomial = new Polynomial(coefficients, x0, step);

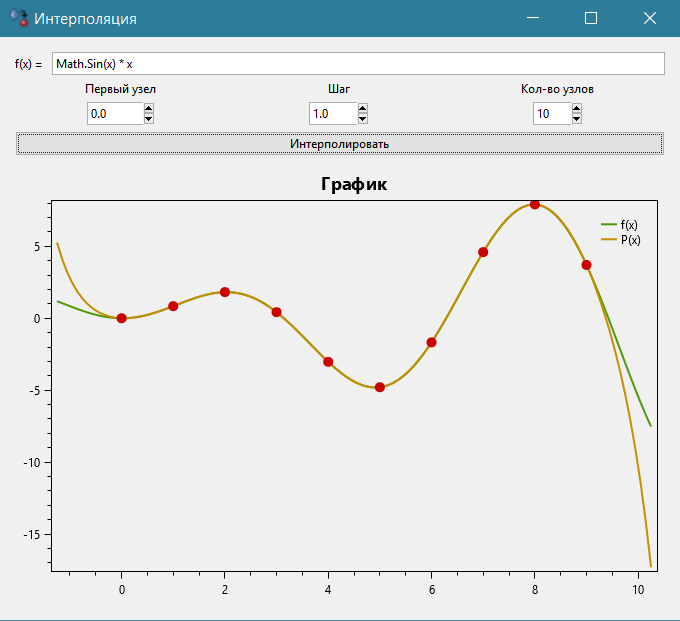
//Проверка условия интерполяции

    for (int i = 0; i < nodeCount; i++)  
    {  
     double xn = x0 + step \* i;  
       Assert.AreEqual(f(xn), polynomial.ValueAt(xn), 1E-5);  
    }  
  
    return polynomial;  
}

**Тестовые данные**

****

****

****

**Вывод**

Интерполяция даёт более точный результат на отрезке, чем аппроксимация.

С другой стороны, за пределами отрезка интерполяция может сильно отличаться от функции, в то время как аппроксимация позволяет достаточно точно экстраполировать её.

По сравнению с многочленом Лагранжа многочлен Ньютона удобен для вычислений тем, что при увеличении числа узлов интерполяции не обязательно перестраивать весь полином заново.

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