«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

По лабораторной работе №4

По дисциплине «**Вычислительная математика**»

Автор: Назирджанов Некруз Фарходович P32211

Факультет: ПИиКТ

ITMO UNIVERSITY

Описание метода интерполяции полиномом Ньютона

Метод интерполяции полиномом Ньютона, используется для нахождения приближенного значения функции на интервале, по изначально полученным точкам.

Шаги

1. Задается набор исходных точек, в которых известны значения функции. Набор точек может быть равномерно распределенным или не очень равномерно распределенным.

2. Дальше вычисляются разделенные разности, которые представляют собой разности между значениями функции в узлах интерполяции.

   $f[x_i] = f(x_i)$

   $f[x_i,x_{i+1}] = (f[x_{i+1}] - f[x_i]) / (x_{i+1} - x_i)$

   $f[x_i,x_{i+1},x_{i+2}] = (f[x_{i+1},x_{i+2}] - f[x_i,x_{i+1}]) / (x_{i+2} - x_i)$

   $f[x_i,x_{i+1},...,x_j] = (f[x_{i+1},x_{i+2},...,x_j] - f[x_i,x_{i+1},...,x_{j-1}]) / (x_j - x_i)$

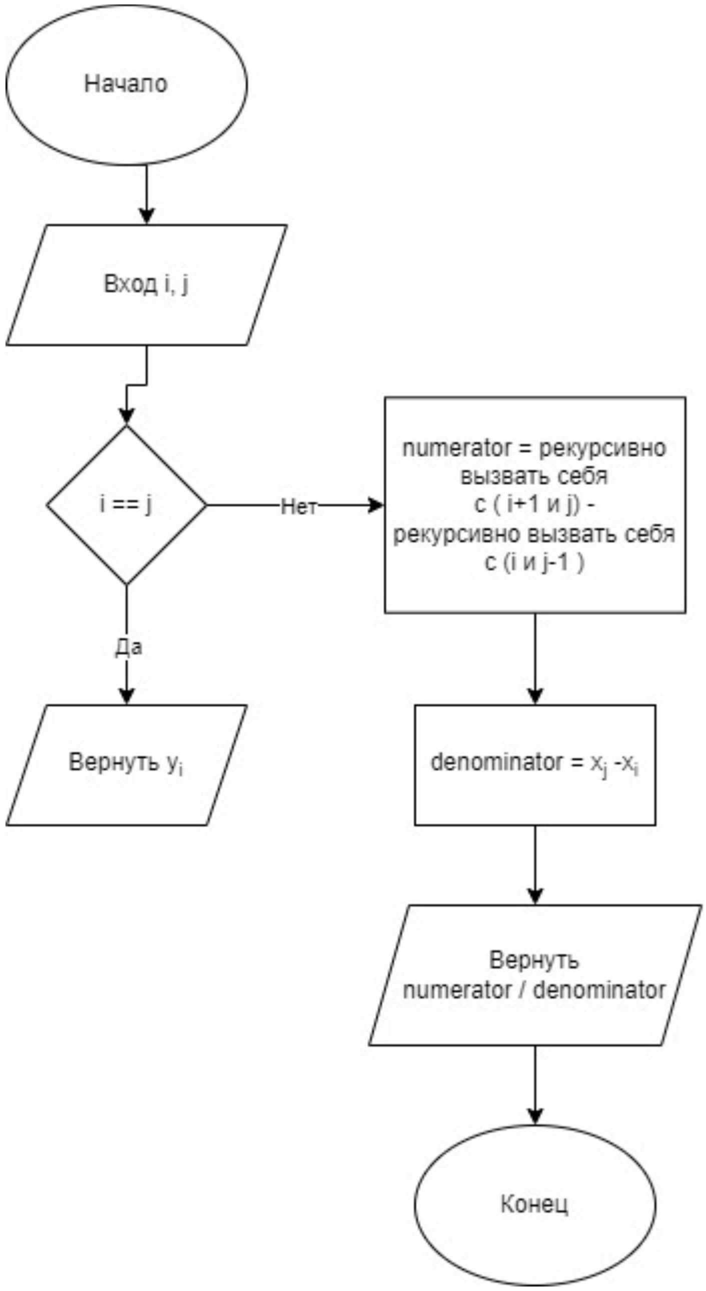
   Разделенные разности можно вычислить рекурсивно, начиная с порядка 0, который является значением функции f в точке x_i. Каждый следующий порядок вычисляется путем вычитания предыдущей разделенной разности из следующей и деления на разность соответствующих точек данных.

3. После этого строится полином Ньютона, который является суммой разделенных разностей, умноженных на соответствующие разности между аргументом интерполяции и узлами интерполяции.
   $P_n(x) = f[x_0] + f[x_0,x_1] (x - x_0) + f[x_0,x_1,x_2] (x - x_0) (x-x_1) + ... + f[x_0,x_1,...,x_n] (x - x_0) (x-x_1)...(x-x_{n-1})$

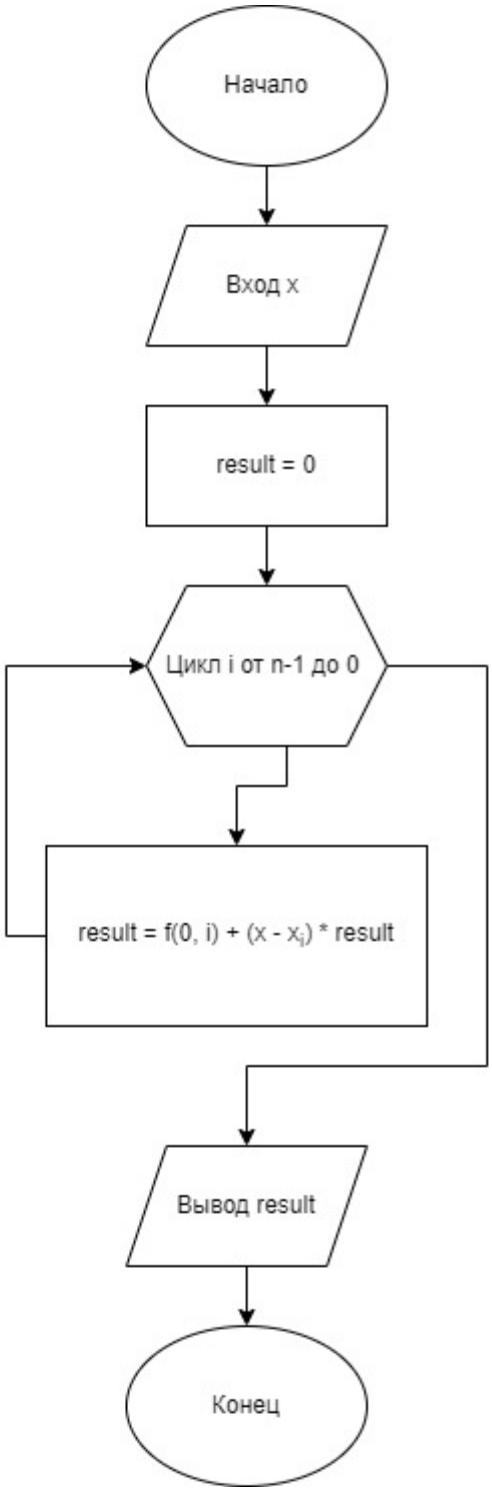
4. После построения полинома Ньютона можно использовать его для приближенного нахождения значений функции в любых точках внутри интервала.

Блок-схема метода

Раздельные разности



Вычисление полинома

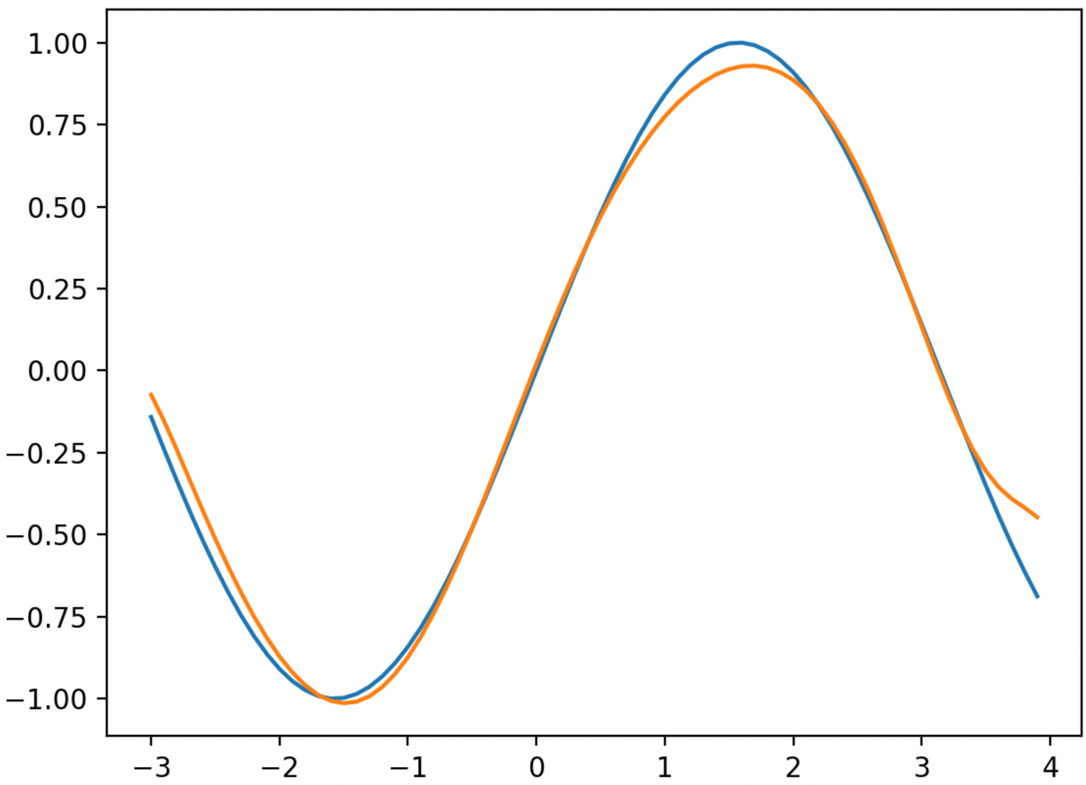


Реализация

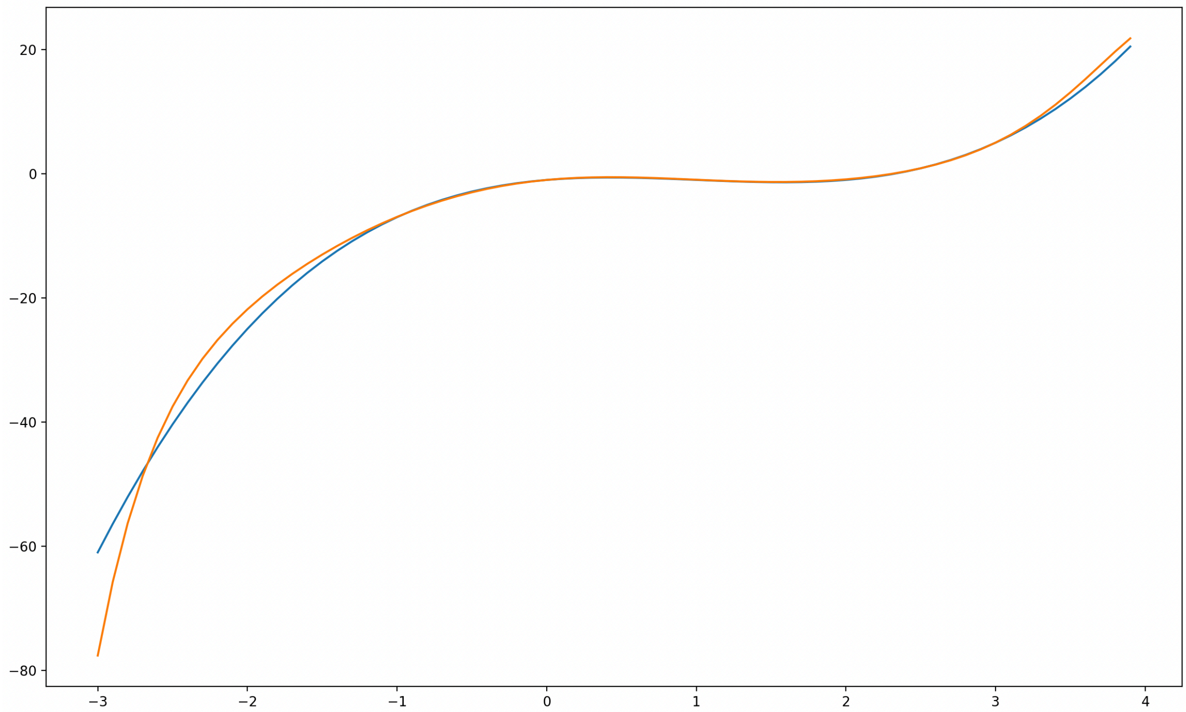
```
using  namespace  std;
#include "NewtonMethod.h"
vector<double> NewtonMethod::interpolate_by_newton(vector<double> x_axis, vector<double> y_axis) {
    int n = x_axis.size();
    vector<double> ans;
    for (int i = 0; i < n; i++) {
        if (i == 0) {
            ans.push_back(y_axis[ i]);
        } else {
            double d = diff(x_axis, y_axis, i + 1);
            ans.push_back(d);
        }
    }
    return ans;
}

double NewtonMethod::diff(vector<double> &x_axis, vector<double> &y_axis, int n) {
    double diveded_diff = 0;
    for (int j = 0; j < n; j++) {
        double buf = y_axis[j];
        for (int i = 0; i < n; i++) {
            if (i == j) continue;
            buf /= (x_axis[j] - x_axis[i]);
        }
        diveded_diff += buf;
    }
    return diveded_diff;
}

double NewtonMethod::calculate_in_point(vector<double> &coefficients, vector<double> &x_axis, double x)
{
    double ans = 0;
    for (int i = 0; i < coefficients.size(); i++) {
        double buf = coefficients[i];
        for (int j = 0; j < i; j++) {
            buf *= (x - x_axis[j]);
        }
        ans += buf;
    }
    return ans;

}
```

Примеры и результаты работы программы

1) **sin(x)**



2)i^3 - 3*i^2 + 2i - 1



Вывод

Метод интерполяции полиномом Ньютона основан на разделенных разностях, его лучше использовать, если требуется вычислить значения функции внутри диапазона заданных точек. Этот метод эффективнее, когда есть много точек

Метод интерполяции полиномом Лагранжа использует коэффициенты Лагранжа, этот метод более простой, особенно если необходимо построить интерполяционную функцию на основе небольшого количества точек или если точки расположены неравномерно. Кроме того, этот метод менее чувствителен к ошибкам округления. Этот метод может быть менее эффективен, чем метод Ньютона

Интерполяция кубическими сплайнами использует кусочно-полиномиальную функцию, его лучше использовать, если нужно построить гладкую кривую через заданные точки, особенно если точки расположены по кривой или сильно изменяющейся форме в таком случае он будет лучше метода Ньютона и Лагранжа. Кубические сплайны обеспечивают более гладкую интерполяцию между точками и обычно дают лучшие результаты. Но, при большом количестве точек, метод может быть слишком затратным.