МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №10
по дисциплине «Вычислительная математика»
Тема: Использование интерполяционной формулы в вычислении значения заданной функции

Студент гр. 7383 _____ Кирсанов А.Я.

Преподаватель _____ Сучков А.И.

Санкт-Петербург
2018

Цель работы.

В соответствии с вариантом, студентам необходимо разработать программу, обеспечивающую вычисление значения функции в заданных точках с использованием подходящих для каждого конкретного случая интерполяционных формул.

Основные теоретические положения.

Если значения функции $f_k = f(x_k)$ заданы в точках $x_k = x_0 + kh (k = 0,1,\ldots,n)$ с постоянным положительным шагом, то часто используется интерполяционный многочлен Ньютона для интерполяции вперед:

$$l_n(x) = l_n(x_0 + qh) = f_0 + q\frac{\Delta f_0}{1!} + \frac{q(q-1)}{2!}\Delta^2 f_0 + \ldots + \left(\prod_{i=0}^{n-1}(q-i)\right)\frac{\Delta^n f_0}{n!}, \quad (1)$$

где $q = \frac{x - x_0}{h}$, а конечные разности $\Delta^r f_0$, носящие названия нисходящих разностей, находят из соотношений:

$$\Delta f_k = f_{k+1} - f_k,$$

$$\Delta^r f_k = \Delta^r f_{k+1} - \Delta^{r-1} f_k = \sum_{j=0}^{r}(-1)^j C_r^j f_{k+r-j}.$$

Интерполяционный многочлен (1) удобно использовать при работе в начале таблицы значений функции и для экстраполяции левее точки x_0.

Интерполяционный многочлен с узлами $x_0, x_{-1},\ldots,x_{-n}$, где $x_{-k} = x_0 - kh$, имеет вид:

$$l_n(x) = l_n(x_0 + qh) = f_0 + q\frac{\nabla f_0}{1!} + \frac{q(q+1)}{2!}\nabla^2 f_0 + \ldots + \left(\prod_{i=0}^{n-1}(q+i)\right)\frac{\nabla^n f_0}{n!} \quad (2)$$

и называется интерполяционным многочленом Ньютона для интерполяции назад. Его удобно использовать при интерполяции в конце таблицы и для экстраполяции правее точки x_0. Входящие в выражение (2) значения конечных восходящих разностей находят из соотношений:

$$\nabla f_k = f_k - f_{k-1} = \nabla f_{k-1} \ ,$$

$$\dots$$

$$\nabla^r f_k = \nabla^{r-1} f_k - \nabla^{r-1} f_{k-1} = \Delta^r f_{k-r} \ .$$

Если при заданном x в таблице значений функции f с шагом h имеется достаточное число узлов с каждой стороны от x, то целесообразно узлы интерполяции $x_0, x_1, ..., x_n$ выбрать так, чтобы точка x оказалась как можно ближе к середине минимального отрезка, содержащего узлы. При этом обычно в качестве x_0 берется ближайший к x узел, затем за x_1 принимается ближайший к x узел, расположенный с противоположной от x стороны, чем x_0. Следующие узлы назначаются поочередно с разных сторон от x и должны быть расположены как можно ближе к x. Одной из возможных схем интерполяции в этом случае является схема Стирлинга с интерполяционным многочленом вида

$$l_n(x) = l_n(q) = f_0 + q\frac{\Delta f_{-1} + \Delta f_0}{2} + \frac{q^2}{2}\Delta^2 f_{-1} + \frac{q(q^2-1)}{3!}\frac{\Delta^3 f_{-2} + \Delta^3 f_{-1}}{2} + \frac{q^2(q^2-1)}{4!}\Delta^4 f_{-2} + \dots$$

$$\dots + \frac{q(q^2-1)(q^2-2^2)\dots\left[q^2-(n-1)^2\right]}{(2n-1)!}\frac{\Delta^{2n-1} f_{-n} + \Delta^{2n-1} f_{-(n-1)}}{2} + \qquad (3)$$

$$+ \frac{q^2(q^2-1)(q^2-2^2)\dots\left[q^2-(n-1)^2\right]}{(2n)!}\Delta^{2n} f_{-n}.$$

В этом выражении учитывается, что дано нечетное число $n+1=2m+1$ значений функции $f_k = f(x_0 + kh)$, где $k=0,\pm1,\pm2,\dots,\pm m$. Обычно эту формулу целесообразно использовать при $|q| \le 0.25$.

Постановка задачи.

В ходе работы нужно разработать программу на одном из языков программирования, обеспечивающую решение одного из вариантов.

Выполнение работы.

В работе написана программа, в которой интерполировалась функция, заданная значениями в одиннадцати узлах. Узлы представлены в табл. 1.

Таблица 1 – Узлы интерполяции.

Номер узла i	Значение x_i	Значение y_i
0	0.3120	-0.3060
1	0.4990	-0.0760
2	0.6870	0.0180
3	0.8740	0.0150
4	1.0620	-0.0440
5	1.2490	-0.1210
6	1.4370	-0.1770
7	1.6240	-0.1720
8	1.8120	-0.0650
9	1.9990	0.1800
10	2.1870	0.6080

В головной программе реализованы интерполяционные формулы (1), (2) и (3). Исходный код программы представлен в приложении А.

В точках $x_1 = 1.3020, x_2 = 1.5690, x_3 = 1.8350$ вычислены значения функции (см. табл. 1) с помощью формул (1), (3), (2) соответственно.

Значения, полученные в программе:

$$f_{x_1} = -0.14036;$$

$$f_{x_3} = -0.0431499;$$

$$f_{x_2} = -0.170309.$$

Выводы.

В данной работе были реализованы функции для вычисления значений функции, заданной с помощью набора равноотстоящих узлов с помощью интерполяционных многочленов Ньютона для интерполяции вперед и назад, и с

помощью схемы Стирлинга. Для вычисления значения функции в точке $x_1 = 1.3020$ использовался многочлен Ньютона для интерполяции вперед. Интерполяция вперед используется в тех случаях, когда требуется вычислить значение функции в точке, находящейся близко к левой границе узлов. Для вычисления значения функции в точке $x_3 = 1.8350$ использовался многочлен Ньютона для интерполяции назад. Интерполяция назад используется, когда требуется найти значение функции в точке, лежащей близко к правой границе узлов. Для вычисления значения функции в точке $x_2 = 1.5690$ использовалась схема Стирлинга. Схема Стирлинга используется в тех случаях, когда требуется вычислить значение функции в точке, лежащей близко к центру таблицы узлов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <cmath>
using namespace std;
double X[11] = {0.3120,  0.4990, 0.6870,  0.8740, 1.0620, 1.2490, 1.4370,
1.6240, 1.8120, 1.9990, 2.1870};
double Y[11] = {-0.3060, -0.0760, 0.0180, 0.0150, -0.0440, -0.1210, -
0.1770, -0.1720, -0.0650, 0.1800, 0.6080};

double F1(int r, int k)
{
    return (r == 0)?(Y[k+1] - Y[k]):(F1(r-1,k+1)-F1(r-1,k));
}
double F2(int r, int k)
{
    return (r == 0)?(Y[k] - Y[k-1]):(F2(r-1,k)-F2(r-1,k-1));
}

double forward_int(double x)
{
     double q = (x-X[0])/0.1875;
     double res = Y[0];
     double p = 1;
     for(int i=0; i<11;i++){
         p*=(q-i)/(i+1);
         res+=p*F1(i,0);
     }
     return res;
}

double back_int(double x)
{
     double q = (x-X[10])/0.1875;
     double res = Y[10];
     double p = 1;
     for(int i=0; i<11;i++){
         p*=(q+i)/(i+1);
         res+=p*F2(i,10);
     }
     return res;
}
```

```
double Stirling(double x)
{
    double q = (-x+X[7])/0.1875;
    double res = Y[7];
    double p = 1;
    for(int i=1; i<4;i++){
        p*=(pow(q,2)-pow(i-1,2))/(2*i-1);
        res+=p*(F1(2*i-1,7-i)-F1(2*i-1,8-i))/2/q;
        p=p/(2*i);
        res+=p*F1(2*i, 7-i);
    }
    return res;
}

int main()
{
    cout << forward_int(1.3020) << " " << back_int(1.8350) << " " <<
Stirling(1.5690) << endl;
    return 0;
}
```