БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ЛАБОРАТОРНАЯ РАБОТА № 1

Вычислительные методы алгебры

**Кратное интерполирование**

Манец Мария

2 курс 1 группа

Преподаватель :

Будник А. М.

**I. Постановка задачи**

Для заданной функции f(x) построить многочлен Эрмита по значениям функции и её производной в заданных узлах интерполирования, вычислить её значения в контрольных точках и провести анализ результатов.

**Входные данные:**

Функция : , а=0.1

x\* =1.03(3), x\*\*=1.53(3), x\*\*\*=1.96(6)

|  |  |  |  |
| --- | --- | --- | --- |
|  | x0 | x1 | x2 |
| x | 1.0 | 1.5 | 2.0 |
| f(x) | 1.0291520691730114 | 1.3459143949774555 | 1.5572732940361786 |
| fI(x) | 0.7581002581272304 | 0.5118323885347391 | 0.3643734570006369 |

**Алгоритм решения и формулы**

Интерполирование с кратными узлами  — задача о построении многочлена минимальной степени, принимающего в некоторых точках заданные значения, а также заданные значения производных до некоторого порядка.

Показывается, что существует единственный многочлен \ P_n(x) степени \ n, удовлетворяющий условиям:

P_n^{(k)}(x_i)=f_{i,k}, i=1,\cdots,m; k=0,\cdots,n_i-1,где n_1+n_2+ \cdots +n_m=n+1.

Этот многочлен называют многочленом с кратными узлами, или Многочленом Эрмита. В общем виде:

P_n(x)=\sum_{i=1}^m\sum_{k=0}^{n_i-1}l_{i,k}(x)f_{i,k}, \ m — количество узлов и \ n_i — кратность узла \ x_i

l_{i,k}(x)=\left[\frac{1}{k!}\frac{\prod_{j=1}^m(x-x_j)^{n_j}}{(x-x_i)^{n_i}}\right]\sum_{s=0}^{n_i-k-1}c_s^i(x-x_i)^{k+s},где \ c_s^i  — коэффициенты ряда Тейлора  для функции \frac{(x-x_i)^{n_i}}{\prod_{j=1}^m(x-x_j)^{n_j}}=\sum_{s=0}^{\infty}c_s^i(x-x_i)^s.

Строится таблицу разделенных разностей:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X0 | F(x0) | F(x0,x0)=F’(x0) |  | … | .. | F(x0,x0,  X1,x1,x2,x2) |
| X0 | F(x0) | F(x0,x1)= |  | .. | .. |  |
| X1 | F(x1) | F(x1,x1)=F’(x1) |  | .. |  |  |
| X1 | F(x1) | F(x1,x2)= |  |  |  |  |
| X2 | F(x2) | F(x2,x2)=F’(x2) |  |  |  |  |
| X2 | F(x2) |  |  |  |  |  |

Далее строится H5(x)по формуле:

H5(x) = f(x0)+ (x-x0)F’(x0)+(x-x0)^2f(x0,x0,x1)+(x-x0)^2(x-x1)f(x0,x0,x1,x1)+

(x-x0)^2(x-x1)^2f(x0,x0,x1,x1,x2)+ (x-x0)^2(x-x1)^2(x-x2)f(x0,x0,x1,x1,x2,x2)

Погрешность вычисляется по формуле:

**Листинг**

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip>

**class** Interpolation {

**public**:

Interpolation(**const** std::vector<std::vector<**double**>> &function) :

divided\_diff\_(10, std::vector<**double**>(9)) {

**for** (**int** i = 0; i < 9; ++i) {

divided\_diff\_[0][i] = function[0][i / 3];

divided\_diff\_[1][i] = function[1][i / 3];

}

**for** (**int** i = 1; i < 9; ++i) {

**if** (i % 3 == 0) {

divided\_diff\_[2][i] = (divided\_diff\_[1][i] - divided\_diff\_[1][i - 1]) /

(divided\_diff\_[0][i] - divided\_diff\_[0][i - 1]);

} **else** {

divided\_diff\_[2][i] = function[2][i / 3];

}

}

**for** (**int** i = 2; i < 9; ++i) {

**if** (i % 3 != 2) {

divided\_diff\_[3][i] = (divided\_diff\_[2][i] - divided\_diff\_[2][i - 1]) /

(divided\_diff\_[0][i] - divided\_diff\_[0][i - 2]);

} **else** {

divided\_diff\_[3][i] = function[3][i / 3] / 2;

}

}

**for** (**int** i = 4; i < 10; ++i) {

**for** (**int** j = i - 1; j < 9; ++j) {

divided\_diff\_[i][j] = (divided\_diff\_[i - 1][j] - divided\_diff\_[i - 1][j - 1]) /

(divided\_diff\_[0][j] - divided\_diff\_[0][j - i + 1]);

}

}

}

**double** value(**double** point) {

**double** res = 0;

**double** k = 1;

**for** (**int** i = 1; i < 10; ++i) {

res += k \* divided\_diff\_[i][i - 1];

k \*= (point - divided\_diff\_[0][i - 1]);

}

**return** res;

}

**private**:

std::vector<std::vector<**double**>> divided\_diff\_;

};

**double** func(**double** a, **double** x) {

**return** a \* exp(x) + (1 - a) \* sin(x);

}

**double** grad(**double** a, **double** x) {

**return** a \* exp(x) + (1 - a) \* cos(x);

}

**double** ges(**double** a, **double** x) {

**return** a \* exp(x) - (1 - a) \* sin(x);

}

**int** factorial(**int** x){

**if** (x==1 || x==0)

**return** 1;

**return** x\*factorial(x-1);

}

**double** A(**double** x){

**return** (x-1.0)\*(x-1.0)\*(x-1.5)\*(x-1.5)\*(x-2.0)\*(x-2.0);

}

**double** expectAc(**double** x){

**return** (grad(0.1,x)\*A(x))/factorial(6);

}

**int** main() {

**double** a = 0.1;

std::vector<std::vector<**double**>> function{{1.0, 1.5, 2.0},

{func(a, 1.0), func(a, 1.5), func(a, 2.0)},

{grad(a, 1.0), grad(a, 1.5), grad(a, 2.0)},

{ges(a, 1.0), ges(a, 1.5), ges(a, 2.0)}};

Interpolation interpolation(function);

**double** x1 = 1.033333333333333;

**double** x2= 1.5333333333333333;

**double** x3 = 1.906666666666667;

std::cout <<"x\*: "<<std::endl;

std::cout<<"resalt: " <<std::setprecision(20) << interpolation.value(x1)<<std::endl;

std::cout<<"expected: " <<std::setprecision(20) << expectAc(x1)<<std::endl;

std::cout<<"true: "<<std::setprecision(20) << interpolation.value(x1)- func(a, x1)<<std::endl;

std::cout <<"x\*\*: "<<std::endl;

std::cout<<"resalt: " <<std::setprecision(20) << interpolation.value(x2)<<std::endl;

std::cout<<"expected: " <<std::setprecision(20) << expectAc(x2)<<std::endl;

std::cout<<"true: "<<std::setprecision(20) << interpolation.value(x2)- func(a, x2)<<std::endl;

std::cout <<"x\*\*\*: "<<std::endl;

std::cout<<"resalt: " <<std::setprecision(20) << interpolation.value(x3)<<std::endl;

std::cout<<"expected: " <<std::setprecision(20) << expectAc(x3)<<std::endl;

std::cout<<"true: "<<std::setprecision(20) << interpolation.value(x3)- func(a, x3)<<std::endl;

**return** 0;

}

**Результаты и вывод**

**Выходные данные:**

x\*:

resalt: 1.0541510874082078431

expected: 2.3296002433561139164e-07

true: 5.0197623835401827819e-12

x\*\*:

resalt: 1.3627281527281485118

expected: 4.7517349857670526328e-08

true: 8.073541835074138362e-13

x\*\*\*:

resalt: 1.52277302781457613

expected: 6.1915214678527565388e-07

true: 5.5832671819189272355e-11

**Вывод**:

Несмотря на то, что интерполирование проводилось только по 6 узлам (3 точкам) результаты по точности сравнимые с результатами, полученными с помощью многочленов Лагранжа и Ньютона (разница только в два порядка). Такой хороший результат объясним тем, что значение первой производной в этих узлах даёт информацию о поведении функции в этих окрестностях, в то время как при интерполировании многочленами Ньютона и Лагранжа известна информация о поведении функции только в самих узлах.