МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з індивідуального розрахункового завдання №11

З предмету «Числові методи»

Виконав

Студент групи КН-36а

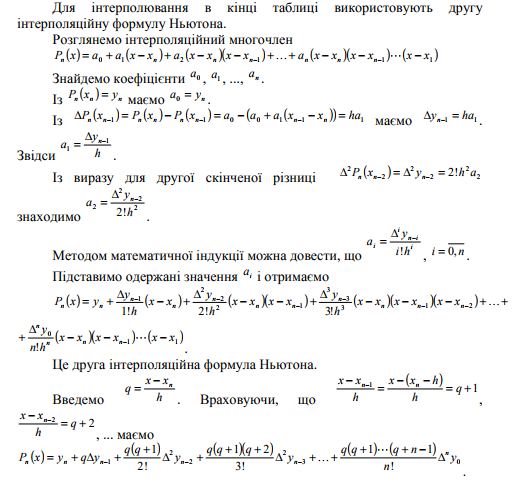
Рубан Ю.Д.

Перевірив:

Гужва В.О.

Харків - 2017

Завдання: знайти інтерполяційний поліном Ньютона по точкам за другою формулою



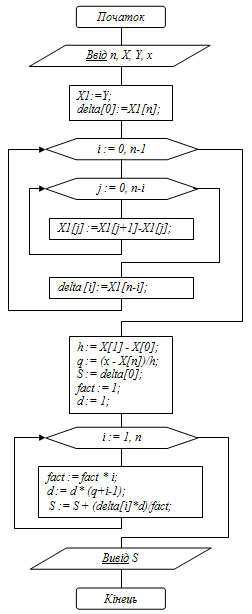


Рисунок 1 – Блок схема другої інтерполяційної формули методом Ньютона

Ручне рішення

|  |  |
| --- | --- |
| 7 | 2 |
| 10 | 4 |
| 13 | 14 |
| 16 | 20 |
| 19 | 10 |

2 10 6 -10

8 -4 -16

-12 -12

0

h = x(i+1) - x(i) = 3

q = (x - xn)/h = (x - 19)/3

f(x) = 10 + ((q + 0) \* -10/1!) + ((q + 0)(q + 1) \* -16/2!) + ((q + 0)(q + 1)(q + 2) \* -12/3!) + ((q + 0)(q + 1)(q + 2)(q + 3) \* 0/4!)

x=11

f(11) = 7.03704

Фрагмент коду програми:

#include"Newton\_interpolation\_polynomal.h"

Newton\_interpolation\_polynomal::Newton\_interpolation\_polynomal(vector<vector<double>>matrix, int size)

{

deltaY.resize(size-1);

int delta0 = size-1;

for (int i = delta0,k=0; k<size-1; i--,k++)

{

deltaY[k].resize(i);

}

for (int i = 0; i < delta0; i++)

{

for (int j = 0; j < deltaY.size()-i; j++)

{

if (i == 0)

deltaY[i][j] = matrix[j + 1][1] - matrix[j][1];

else

deltaY[i][j] = deltaY[i - 1][j + 1] - deltaY[i - 1][j];

}

}

ma = matrix;

Algorithms::show(deltaY);

}

double Newton\_interpolation\_polynomal::do\_algorithm(double x)

{

double y = ma[ma.size() - 1][1];

double qi = 1;

double h = ma[1][0] - ma[0][0];

int size = ma.size();

for (int i = 0, k = size - 1; i < size-1; i++, k--)

{

for (int j = 0; j <= i; j++)

{

qi \*= ((x - ma[size - 1][0]) / h) + j;

}

y += (qi / Algorithms::fact(i+1))\*deltaY[i][k-1];

qi = 1;

}

return y;

}

string Newton\_interpolation\_polynomal::get\_polynomal()

{

if(polynomal!="")

return polynomal;

else

{

polynomal += "h = x(i+1) - x(i) = " + toString(ma[1][0] - ma[0][0])+"\n";

polynomal += "q = (x - xn)/h = (x - " + toString(ma[ma.size() - 1][0]) + ")/" + toString(ma[1][0] - ma[0][0])+"\n";

polynomal += "f(x) = ";

polynomal += toString(ma[ma.size() - 1][1]);

polynomal += " + ";

double y = ma[ma.size() - 1][1];

double h = ma[1][0] - ma[0][0];

int size = ma.size();

for (int i = 0, k = size - 1; i < size - 1; i++, k--)

{

polynomal += "(";

for (int j = 0; j <= i; j++)

{

polynomal += "(q + " + toString(j) + ")";

}

polynomal += " \* " + toString(deltaY[i][k - 1]);

polynomal += "/" + toString(i+1) + "!) + ";

}

polynomal.erase(polynomal.end() - 2);

return polynomal;

}

}

function<double(double)> der(function<double(double)>& func, double x, double eps)

{

return [&func, &eps](double x) {double a = func(x + eps);

double b = func(x);return ((a - b) / eps); };

}

double Newton\_interpolation\_polynomal::R(double x,function<double(double)>&func)

{

int n = deltaY.size();

int size = ma.size();

double avg=0;

for (int i = 0; i < size; i++)

{

avg += ma[i][0];

}

avg = avg / size;

double h = ma[1][0] - ma[0][0];

double qi = 1;

double f = 1;

for (int j = 0; j < n; j++)

{

qi \*= (((x - ma[size - 1][0]) / h) + j);

f \*= j + 1;

}

qi = qi\*pow(h, n)/f;

function<double(double)>\*derevate = new function<double(double)>[n];

derevate[0] = der(func, avg, 0.000001);

for (int i = 1; i < n; i++)

{

derevate[i] = der(derevate[i-1], avg, 0.000001);

}

return qi\*derevate[n-1](avg);

}

Результат виконання програми

n=5

x,y =

7 2

10 4

13 14

16 20

19 10

xi = 11

f(xi) = 7.03704

Висновок:

Результати програми і ручного рішення збігаються

***Список використаних джерел***

1) Гончаров В. Л., Теория интерполирования и приближения функций, 2 изд., М., 1954;

2) Самарский А.А. Гулин А.В. Численные методы 1989г.

3) Костомаров Д.П., Фаворский А.П. Вводные лекции по численным методам