МИНЕСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., канд. техн. наук |  |  |  | О. М. Косогоров |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| Интерполяционные полиномы. |
| по курсу: ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | М023 |  |  |  | Д.А.Трегуб |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

**1)Задание:**

Построить различные виды интерполяции (линейная, канонический полином, Лагранжа, Ньютона) для таблицы значений из задания №1 своего варианта

**Вариант №8:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | -1 | -0,5 | 0 | 0,5 |
| y | -2 | 0 | 1 | 3,5 |

**2)Код на C++:**

#include <iostream>

#include <math.h>

#include <iomanip>

using namespace std;

class Interpolation{

private:

double \*x, \*y; // Линейная и вывод

double \*\*kn, \*\*kn\_ob, \*x\_k, \*\*y\_k; //Каноническая

double \*Lg; //Лагранж

double\*\* dy; //Ньютон

int leng;

public:

Interpolation(int n) {

//Линейная

leng = n;

x = new double[leng\*2];

y = new double[leng\*2];

//каноническая

x\_k = new double[leng];

y\_k = new double\* [leng];

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

y\_k[i] = new double[leng];

kn = new double\*[leng];

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

kn[i] = new double[leng];

kn\_ob = new double\* [leng];

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

kn\_ob[i] = new double[leng];

//Лагранжа

Lg = new double[leng\*2];

//Ньютона

dy = new double\* [leng];

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

dy[i] = new double[leng];

}

~Interpolation() {

delete x;

delete y;

delete x\_k;

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

delete[] y\_k;

delete y\_k;

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

delete[] kn;

delete kn;

delete kn\_ob;

delete Lg;

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

delete[] dy;

delete dy;

}

void print() {

cout <<"Интерполяция:" << endl << "x:";

for (int i = 0; i < leng \* 2 - 1; i++)

cout << setw(10) << x[i] << " ";

cout << endl << "y:";

for (int i = 0; i < leng \* 2 - 1; i++)

cout << setw(10) << y[i] << " ";

cout << endl << endl;

}

void enter() {

for (int i = 0; i < leng \* 2; i++) {

x[i] = 0; y[i] = 0;

}

cout << "Введите значения x: " << endl;

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

cin >> x\_k[i];

cout << "Введите значения y: " << endl;

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

cin >> y\_k[i][0];

for (int i = 0; i < leng; i++) {

x[i \* 2] = x\_k[i];

y[i \* 2] = y\_k[i][0];

}

for (int i = 0; i < leng \* 2; i++) {

if ((i % 2) != 0)

x[i] = (x[i - 1] + x[i + 1]) / 2;

}

}

void LinearInter() {

for (int i = 0; i < leng\*2; i++) {

if ((i % 2) != 0)

x[i] = (x[i - 1] + x[i + 1]) / 2;

}

for (int i = 1; i < leng\*2; i++) {

if ((i%2)!=0)

y[i] = y[i-1] + (y[i+1]-y[i-1]) / (x[i+1]-x[i-1]) \* (x[i]-x[i-1]);

}

}

void CanonInter() {

for (int i = 0; i < leng \* 2; i++) {

if ((i % 2) != 0)

x[i] = (x[i - 1] + x[i + 1]) / 2;

}

//Заполнение матрицы k

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

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

kn[i][j] = pow(x\_k[i], j);

//обратная матрица k

kn = invMatr(kn, leng);

//умножение матрицы

kn = prodMatr(kn, y\_k, leng);

for (int i = 0; i < leng \* 2; i++) {

if ((i % 2) != 0) {

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

y[i] =+ kn[j][0] \* pow(x[i], j);

}

}

}

void GetMatr(double\*\* mas, double\*\* p, int i, int j, int m) {

int ki, kj, di, dj;

di = 0;

for (ki = 0; ki < m - 1; ki++) { // проверка индекса строки

if (ki == i) di = 1;

dj = 0;

for (kj = 0; kj < m - 1; kj++) { // проверка индекса столбца

if (kj == j) dj = 1;

p[ki][kj] = mas[ki + di][kj + dj];

}

}

}

double Determinant(double\*\* mas, int m) {

int i, j, k, n;

double\*\* p, d;

p = new double\* [m];

for (i = 0; i < m; i++)

p[i] = new double[m];

j = 0; d = 0;

k = 1; //(-1) в степени i

n = m - 1;

if (m < 1) cout << "Определитель вычислить невозможно!";

if (m == 1) {

d = mas[0][0];

return(d);

}

if (m == 2) {

d = mas[0][0] \* mas[1][1] - (mas[1][0] \* mas[0][1]);

return(d);

}

if (m > 2) {

for (i = 0; i < m; i++) {

GetMatr(mas, p, i, 0, m);

d = d + k \* mas[i][0] \* Determinant(p, n);

k = -k;

}

}

return(d);

}

double\*\* prodMatr(double\*\* a, double\*\* b, int n) {

double\*\* res;

double sum = 0;

res = new double\* [n];

int i, j;

res = new double\* [n];

for (i = 0; i < n; i++)

res[i] = new double[n];

for (i = 0; i < n; i++) {

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

res[i][j] = 0;

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

res[i][j] += a[i][k] \* b[k][j];

}

}

return res;

}

double\*\* invMatr(double\*\* a, int n) {

double\*\* q = new double\* [n];

double\*\* T = new double\* [n];

double\*\* h = new double\* [n];

double d;

for (int i = 0; i < n; i++) {

q[i] = new double[n];

T[i] = new double[n];

h[i] = new double[n];

}

d = 1 / Determinant(a, n);

//Транспонирование

for (int i = 0; i < n; i++) {

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

T[j][i] = a[i][j];

}

//Матрица алгербраических дополнений

cout << endl;

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

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

GetMatr(T, h, i, j, n);

q[i][j] = d \* Determinant(h, 2);

if ((i + j + 2) % 2 == 1)

q[i][j] = q[i][j] \* (-1);

}

return q;

}

void lagranInter() {

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

Lg[i] = Yn(y\_k, x\_k, x[i \* 2 + 1]);

int j = 1;

for (int i = 0; i < leng - 1; i++) {

y[j] = Lg[i];

j = j + 2;

}

}

double l(int indx, double\* X, double x) {

double l = 1;

for (int i = 0; i < leng; i++) {

if (i != indx)

l = l \* (x - X[i]) / (X[indx] - X[i]);

}

return l;

}

double Yn(double\*\* y\_k, double\* x\_k, double q) {

double Yn = 0;

for (int i = 0; i < leng - 1; i++) {

Yn += y\_k[i][0] \* l(i, x\_k, q);

}

return Yn;

}

void NewtonInter() {

for (int i = 0; i < leng; i++) {

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

dy[i][j] = 0;

}

//Первая конечная разность

for (int i = 0; i < leng - 1; i++)

dy[i][0] = y\_k[i + 1][0] - y\_k[i][0];

//Вторая конечная разность

for (int i = 1; i < leng; i++) {

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

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

}

for (int i = 0; i < leng - 1; i++)

y[i \* 2 + 1] = NewtonHelp(x[i \* 2 + 1]);

}

double NewtonHelp(double x0) {

int fact = 1;

double mult, sum, q, h;

mult = 1;

sum = y\_k[0][0];

h = 0.25;

q = (x0 - x\_k[0]) / h;

for (int i = 0; i < leng; i++) {

mult \*= (q - i);

fact \*= i + 1;

sum += mult \* dy[0][i] / (fact);

}

return sum;

}

};

int main() {

setlocale(LC\_ALL, "Russian");

Interpolation q(4);

cout << "Линейная интерполяция:" << endl;

q.enter();

q.LinearInter();

q.print();

cout << endl << "Интерполяция методом канонического полинома:" << endl;

q.enter();

q.CanonInter();

q.print();

cout << endl << "Интерполяция Лагранжа:" << endl;

q.enter();

q.lagranInter();

q.print();

cout << endl << "Интерполяция Ньютона:" << endl;

q.enter();

q.NewtonInter();

q.print();

cout << endl;

system("pause");

return 0;

}

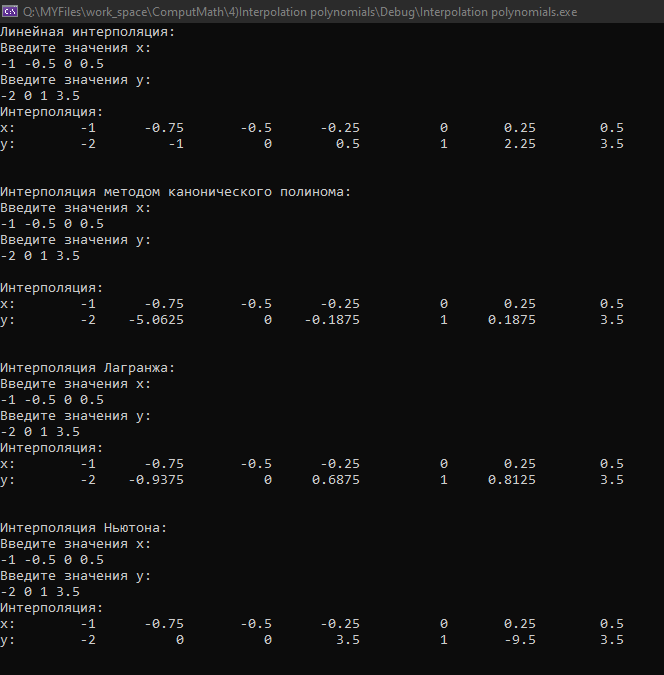
/\*

-1 -0.5 0 0.5

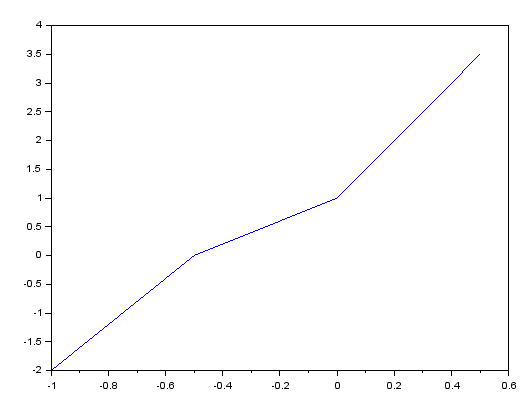
-2 0 1 3.5

\*/

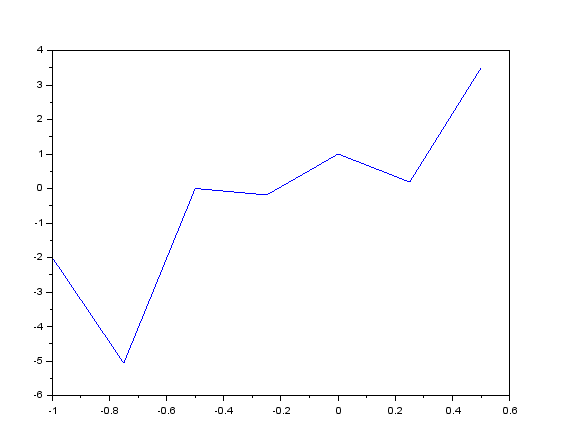
**3)Результат работы программы с++:**



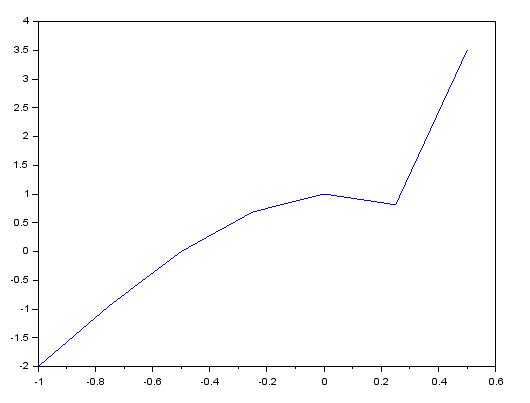
**4)Графики в scilab:**

Линейная интерполяция:  


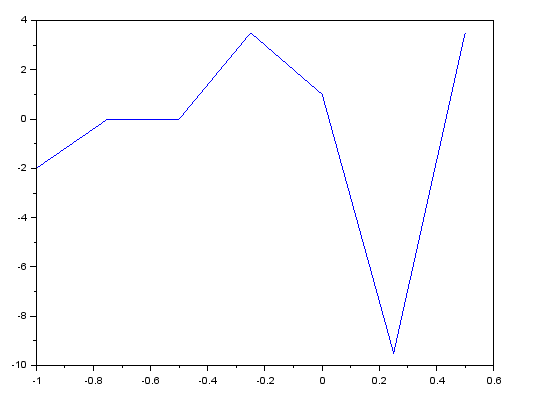
Интерполяция методом канонического полинома:



Интерполяция Лагранжа:



Интерполяция Ньютона:



**5)Вывод:** Выполняя данную лабораторную работу реализовал 4 метода интерполяции(Линейная, канонического полинома, Лагранжа и Ньютона) на С++, и построил графики по получившимся значениям в scilab.