Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно–вычислительных систем (КИБЭВС)

ИНТЕРПОЛЯЦИЯ И АППРОКСИМАЦИЯ ФУНКЦИЙ

Отчет по лабораторной работе №3

по дисциплине «Численные методы»

Выполнил

Студент гр. 728–2

\_\_\_\_\_\_\_ Д. Р. Геворгян

\_\_\_.12.2019

Принял

Доцент кафедры КИБЭВС

\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ С. С. Катаева  
 оценка подпись \_\_\_.12.2019

**1 Введение**

Необходимо для функции вычислить интерполяционный полином и аппроксимирующую функцию.

Порядок действий:

1. Вычислить пары точек , где , ,  – индивидуальная функция.
2. Запрограммировать вычисление интерполяционного полинома Лагранжа  2 степени по заданным точкам.
3. По методу наименьших квадратов рассчитать параметры функции , аппроксимирующей функцию . Построить в Excel график зависимости  от , вывести линии полиномиального (2 степени) тренда, уравнение тренда и коэффициент детерминации .

Взять произвольный узел , , промежуточную точку , , выбрать точку прогноза, и для всех 3 точек заполнить таблицу:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | абсол.  ошибка | относ.  ошибка |  | абсол.  ошибка | относ.  ошибка |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**2 Ход работы**

2.1 Интерполяционный полином Лагранжа

В общем виде интерполяционный многочлен в форме Лагранжа записывается в следующем виде:

где *n* – степень полинома;

*f(xi)* – значение интерполирующей функции *f(x)* в точке *xi*;

*li(x)* – базисные полиномы (множитель Лагранжа), которые определяются по формуле:

Так, например, интерполяционный многочлен в форме Лагранжа, проходящий через три заданные точки , будет записываться в следующем виде:

2.2 Аппроксимирующая функция по методу наименьших квадратов

Для расчета коэффициентов аппроксимирующей функции применяется метод наименьших квадратов.

Расширенная матрица системы уравнений:, решив которую получим искомые коэффициенты a, b,c.

Коэффициент детерминации (R2) — это доля дисперсии зависимой переменной, объясняемая рассматриваемой моделью. Более точно — это единица минус доля необъяснённой дисперсии (дисперсии случайной ошибки модели, или условной по признакам дисперсии зависимой переменной) в дисперсии зависимой переменной. В случае линейной зависимости R2 является квадратом так называемого множественного коэффициента корреляции между зависимой переменной и объясняющими переменными.

Коэффициент корреляции:

где:

Была написана консольная программа, реализующая данные методы.

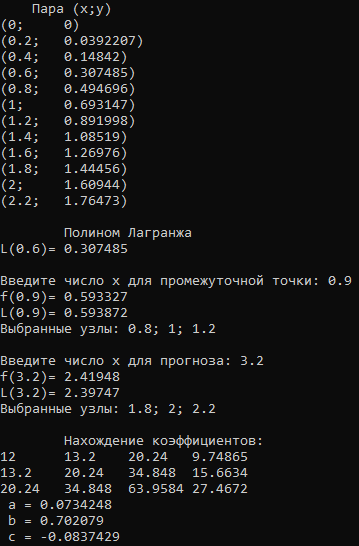


Рисунок 2.2.1 - Результат выполнения кода

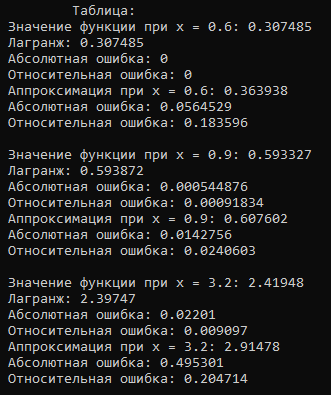


Рисунок 2.2.2 – Результат выполнения кода (Продолжение)

2.3 Результаты вычислений

Результаты вычислений представлены в таблице 2.3.1. На рисунке 2.3.1 представлен график функции.

Абсолютная ошибка – А.О.

Относительная ошибка – О.О.

Таблица 2.3.1 – Результаты вычислений

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *x* | *f(x)* | *L(x)* | А.О | О.О. |  | А.О. | О.О. |
| = 0.6 | 0.307485 | 0.307485 | 0 | 0 | 0.363938 | 0.0564529 | 0.183596 |
| = 0.9 | 0.593327 | 0.593872 | 0.000544876 | 0.00091834 | 0.607602 | 0.0142756 | 0.0240603 |
| = 3.2 | 2.41948 | 2.39747 | 0.02201 | 0.009097 | 2.91478 | 0.495301 | 0.204714 |

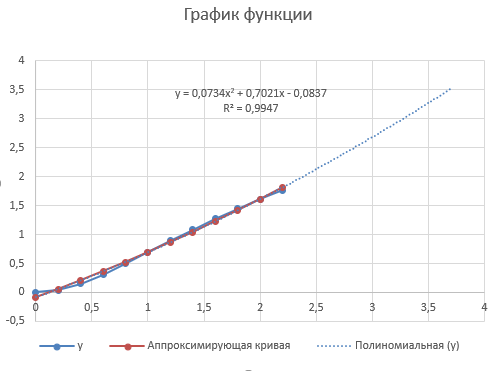


Рисунок 2.3.1 – График функции и аппроксимирующей кривой

**3 Заключение**

В результате выполнения данной лабораторной работы были получены навыки разработки программ, использующих в своей работе вычисление интерполяционного полином Лагранжа и вычисление аппроксимирующей функции по методу наименьших квадратов.

Отчет был написан согласно ОС ТУСУР.

Приложение А (код программы)

#include <iostream>

#include <cmath>

#include <math.h>

#include <vector>

using namespace std;

double func(double x)

{

return log(pow(x, 2) + 1); //функция из задания

}

vector <double> n = { 0,0.2,0.4,0.6,0.8,1.0,1.2,1.4,1.6,1.8,2.0,2.2 };

double a[3][4];

vector <double> tabl(12);

vector <double> koefL;

double Lagrange(double x, double i)

{

koefL.clear();

int l1 = 0;

if (11 - i >= 2 && i - int(i) == 0 && i <= 11)

{

l1 = i;

for (int k = i; k <= i + 2; k++)

koefL.push\_back(n[k]);

}

else

if (11 - i <= 2 && i - int(i) == 0 && i <= 11)

{

l1 = i - 2;

for (int k = i - 2; k <= i; k++)

koefL.push\_back(n[k]);

}

else

if (i - int(i) != 0 && i <= 11 && int(i) != 0 && int(i) != 11)

{

l1 = int(i - 1);

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

koefL.push\_back(n[k]);

}

else

if (i - int(i) != 0 && i <= 11 && int(i) == 0 && int(i) != 11)

{

l1 = int(i);

for (int k = int(i); k <= int(i + 2); k++)

koefL.push\_back(n[k]);

}

else

if (i - int(i) != 0 && i <= 11 && int(i) != 0 && int(i) == 11)

{

l1 = int(i - 2);

for (int k = int(i - 2); k <= int(i); k++)

koefL.push\_back(n[k]);

}

else

if (i > 11)

{

l1 = 9;

for (int k = 9; k <= 11; k++)

koefL.push\_back(n[k]);

}

double L = tabl[l1] \* (x - koefL[1])\*(x - koefL[2]) / (koefL[0] - koefL[1]) / (koefL[0] - koefL[2]) + tabl[l1 + 1] \* (x - koefL[0]) \* (x - koefL[2]) / (koefL[1] - koefL[0]) / (koefL[1] - koefL[2]) + tabl[l1 + 2] \* (x - koefL[0]) \* (x - koefL[1]) / (koefL[2] - koefL[0]) / (koefL[2] - koefL[1]);

return L;

}

int main() {

setlocale(LC\_ALL, "Russian");

cout << " Пара (x;y)" << endl;

for (int i = 0; i < 12; i++) {

tabl[i] = func(n[i]);

cout << "(" << n[i] << ";" << "\t" << tabl[i] << ")" << endl;

}

cout << "\n\t" << "Полином Лагранжа" << endl; //Полином Лагранжа

vector <double> Lag;

Lag.push\_back(Lagrange(n[3], 3));

vector <double> yfin;

yfin.push\_back(tabl[3]);

cout << "L(" << n[3] << ")= " << Lag[0] << endl;

cout << "\nВведите число х для промежуточной точки: ";

vector <double> xfin;

xfin.push\_back(n[3]);

double xp;

int sr = -1;

do

{

cin >> xp;

if (xp >= 0 && xp <= 2.2)

{

xfin.push\_back(xp);

for (int i = 0; i < 12; i++)

if (xp > n[i])

sr++;

else

if (xp < n[i])

{

sr += 0.5;

break;

}

else

{

sr += 1;

break;

}

yfin.push\_back(func(xp));

cout << "f(" << xp << ")= " << yfin[1] << endl;

Lag.push\_back(Lagrange(xp, sr));

cout << "L(" << xp << ")= " << Lag[1] << endl;

cout << "Выбранные узлы: " << koefL[0] << "; " << koefL[1] << "; " << koefL[2] << endl;

break;

}

} while (true);

cout << "\nВведите число х для прогноза: ";

do

{

cin >> xp;

if (3.2 >= xp && xp > 2.2) {

xfin.push\_back(xp);

yfin.push\_back(func(xp));

cout << "f(" << xp << ")= " << yfin[2] << endl;

Lag.push\_back(Lagrange(xp, 13));

cout << "L(" << xp << ")= " << Lag[2] << endl;

cout << "Выбранные узлы: " << koefL[0] << "; " << koefL[1] << "; " << koefL[2] << endl;

break;

}

} while (true);

double d, x[3], x0;

cout << "\n\t" << "Нахождение коэффициентов: " << endl; //Нахождение коэффициентов по методу Гаусса

for (int i = 0; i <= 11; i++) {

a[0][0] += pow(n[i], 0); //c

a[0][1] += n[i]; //b

a[0][2] += pow(n[i], 2); //a

a[0][3] += tabl[i]; //f(x)

a[1][0] += n[i];

a[1][1] += pow(n[i], 2);

a[1][2] += pow(n[i], 3);

a[1][3] += tabl[i] \* n[i];

a[2][0] += pow(n[i], 2);

a[2][1] += pow(n[i], 3);

a[2][2] += pow(n[i], 4);

a[2][3] += tabl[i] \* pow(n[i], 2);

}

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 4; j++)

cout << a[i][j] << "\t";

cout << endl;

}

for (int k = 0; k < 3; k++) {

for (int i = k + 1; i < 3; i++) {

d = -(a[i][k] / a[k][k]);

for (int j = k; j < 4; j++) {

a[i][j] = a[i][j] + a[k][j] \* d;

}

}

}

x[2] = a[2][3] / a[2][2];

x[1] = (a[1][3] - x[2] \* a[1][2]) / a[1][1];

x[0] = (a[0][3] - a[0][2] \* x[2] - a[0][1] \* x[1]) / a[0][0];

cout << " a = " << x[2] << endl;

cout << " b = " << x[1] << endl;

cout << " c = " << x[0] << endl;

cout << "\n\t" << " Таблица:" << endl;

for (int u = 0; u < 3; u++)

{

cout << " Значение функции при х = " << xfin[u] << ": " << yfin[u] << endl;

cout << " Лагранж: " << Lag[u] << endl;

cout << " Абсолютная ошибка: " << fabs(yfin[u] - Lag[u]) << endl;

cout << " Относительная ошибка: " << fabs(fabs(yfin[u] - Lag[u]) / yfin[u]) << endl;

x0 = x[2] \* pow(xfin[u], 2) + x[1] \* xfin[u] + x[0];

cout << " Аппроксимация при x = " << xfin[u] << ": " << x0 << endl;

cout << " Абсолютная ошибка: " << fabs(yfin[u] - x0) << endl;

cout << " Относительная ошибка: " << fabs(fabs(yfin[u] - x0) / yfin[u]) << endl << endl;

}

system("pause");

return 0;

}