Группа 4310

Нигамадянов Ф.М.

Маннанов Д.И.

**ЛАБОРАТОРНАЯ РАБОТА №6-7.**

**ИНТЕРПОЛЯЦИЯ И АППРОКСИМАЦИЯ ФУНКЦИЙ**

Цель работы: научиться строить интерполяционные и аппроксимационные многочлены по заданной системе точек с помощью ЭВМ.

Содержание работы:

1. Изучить принципы построения интерполяционной формулы Лагранжа, I и II интерполяционных формул Ньютона и аппроксимационного полинома.

2. На конкретном примере усвоить порядок построения указанных полиномов с помощью ЭВМ.

3. Составить программу на любом языке программирования, реализующую процесс построения указанных полиномов второго порядка для системы из трех равноотстоящих узловых точек.

4. Сделать вывод о точности построения полиномов.

5. Составить отчет о проделанной работе.

Задание (вариант №16).

1. Составить таблицу значений экспериментальной функции с точностью для равноотстоящей системы из трех узловых точек , на отрезке из области допустимых значений функции, где .

2. По сформированной системе точек построить интерполяционную формулу Лагранжа, I и II интерполяционные формулы Ньютона и аппроксимационный полином второго порядка.

3. Составить программу на любом языке программирования, реализующую процесс построения указанных полиномов для заданной системы точек.

Решение.

1. Данная функция определена на всем множестве действительных чисел. Рассмотрим отрезок . Таблица значений функции с точностью для равноотстоящей системы из трех узловых точек , на отрезке , где , имеет вид:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
|  | 0.0 | 0.5 | 1.0 |
|  | 5648.003 | 5640.949 | 5619.857 |

2. Интерполяционный полином Лагранжа.

Замечание. Так как данный полином строится для произвольной системы узловых точек, то запишем этот полином для равноотстоящих узловых точек:

,

где коэффициенты вычисляются так:

;

;

.

Тогда искомый многочлен Лагранжа второго порядка будет иметь вид:

, где .

I интерполяционная формула Ньютона второго порядка по заданной системе точек строится в виде:

.

Здесь величины , называются табличными разностями первого и второго порядков соответственно, .

II интерполяционная формула Ньютона второго порядка по заданной системе точек строится в виде:

.

Здесь величины и вводятся аналогично случаю, рассмотренному выше, .

Вычислим значения табличных разностей первого и второго порядков, необходимых для построения I и II интерполяционных формул Ньютона:

Таким образом, I и II интерполяционные формулы Ньютона второго порядка по заданной системе точек запишутся так:

,

.

Иначе

,

.

При построении аппроксимационного многочлена методом наименьших квадратов необходимо, чтобы сумма квадратов отклонений построенной функции от экспериментальной в узловых точках была минимальна. Будем строить функцию в виде многочлена второго порядка

.

Согласно алгоритму метода наименьших квадратов, для построения многочлена второй степени необходимо вычислить следующие суммы:

и решить систему линейных алгебраических уравнений 3-го порядка вида

(1)

относительно неизвестных коэффициентов . В данном случае система (1) будет выглядеть так

(2)

Для ее решения можно воспользоваться любым известным методом, например, методом Крамера. Для этого необходимо вычислить четыре определителя системы (2) вида:

Значения искомых коэффициентов вычисляются по формулам:

Искомый аппроксимационный многочлен второго порядка будет иметь вид:

Для построения программного кода выведем формулы вычисления определителей выше в следующем виде:

Для проверки правильности построения полиномов необходимо провести программно процесс табулирования четырех построенных полиномов и экспериментальной функции при с одинаковым шагом табулирования.

Графики этих функций представлены на рисунке 1. Они были построены в среде Wolfram Matematica. На рисунке 2 представлен код для построения этих графиков. Из рис. 1 видно, что искомые полиномы на отрезке практически совпадают с экспериментальной функцией и проходят через узловые точки.

Замечание. Аппроксимационный полином в общем случае не проходит через узловые точки и для системы из трех узловых точек может давать погрешность, превышающую погрешность построения остальных полиномов.

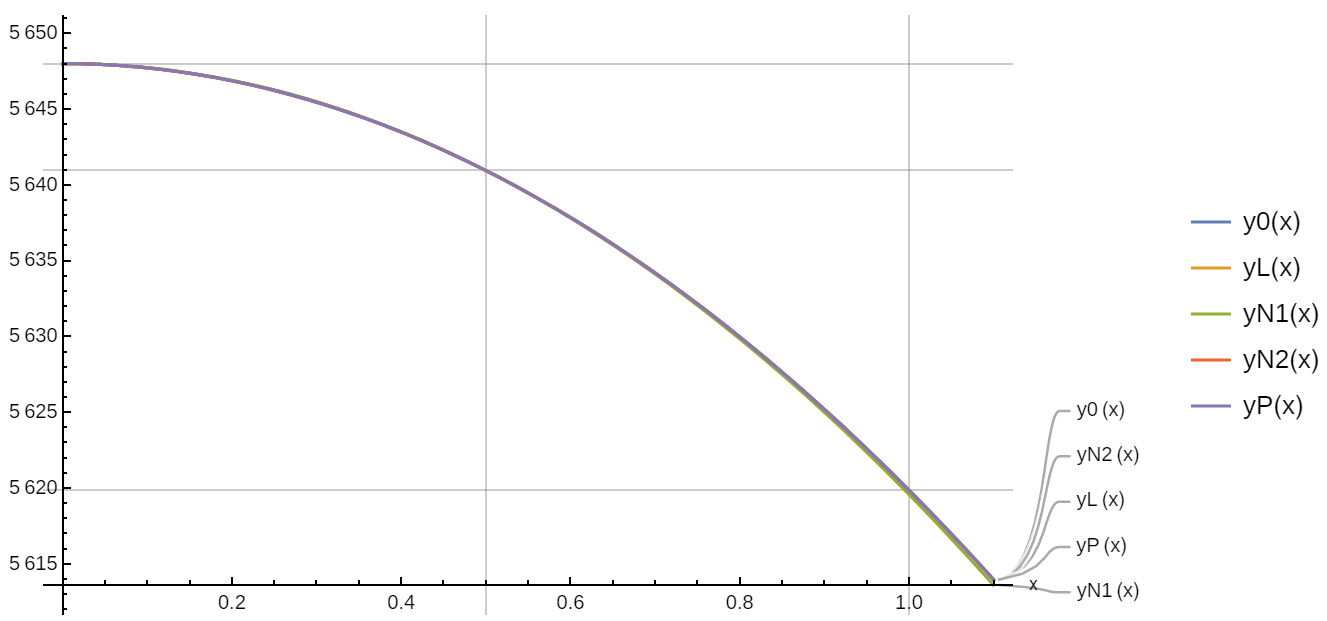


Рис. 1

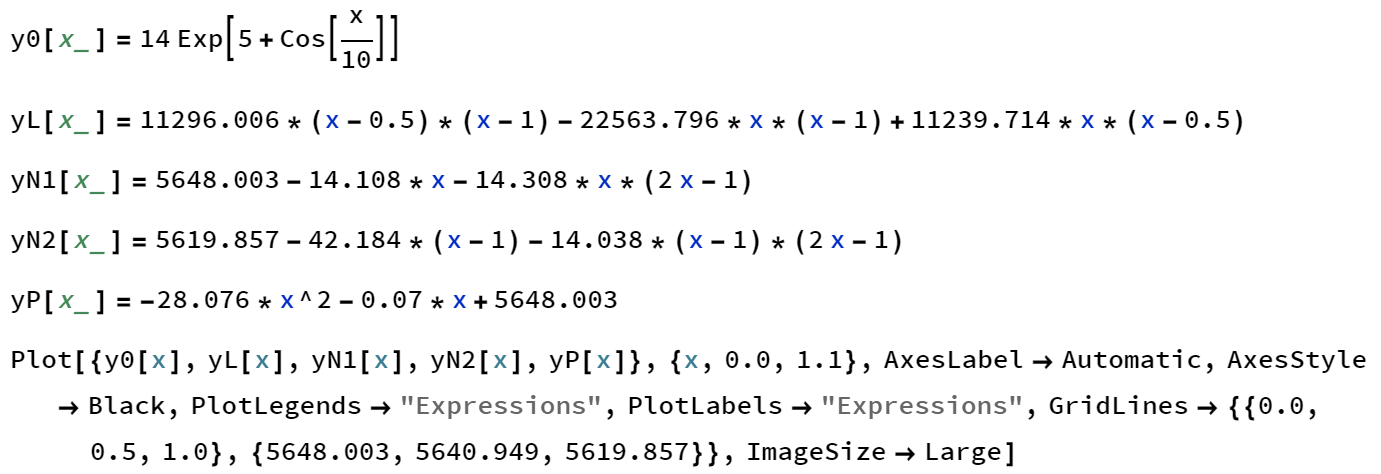


Рис. 2

Листинг программы:

#include <iostream>

#include "locale.h"

#include <math.h>

#include <iomanip>

using namespace std;

const double x0 = -4.7, xn = -3.5; // Границы отрезка

const double x[] = { x0, x0 + (xn - x0) / 2.0, xn }; // xi (узловые точки)

double y[3]; // yi (узловые точки)

double a[3]; // коэффициенты P2

const int n = 12; // Кол-во шагов

const double hx = (xn - x0) / n; // Для построения ИФЛ и АМ

const double ht = ((xn - x0) / n) / ((xn - x0) / 2.0); // Для I и II ИФН

double f(double x) // Исходная функция

{

return 14.0 \* exp(5.0 + cos(x/10.0));

}

void uzlovye\_tochki() // Расчет узловых точек

{

y[0] = f(x[0]);

y[1] = f(x[1]);

y[2] = f(x[2]);

cout << "-----------------------------------------------------" << endl << setprecision(7);

cout << "| xi |" << setw(10) << "x0 = " << x[0] << setw(4) << "|" << setw(8) << "x1 = " << x[1]

<< setw(4) << "|" << setw(10) << "x2 = " << x[2] << setw(4) << "|" << endl;

cout << "-----------------------------------------------------" << endl;

cout << "| yi |" << setw(11) << y[0] << setw(4) << "|" << setw(11) << y[1]

<< setw(4) << "|" << setw(11) << y[2] << setw(4) << "|" << endl << endl;

}

double L2(double \_x) // Интерполяционный полином Лагранжа

{

double a[3];

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

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

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

double result = a[0] \* (\_x - x[1]) \* (\_x - x[2]) + a[1] \* (\_x - x[0]) \* (\_x - x[2]) +

a[2] \* (\_x - x[0]) \* (\_x - x[1]);

return result;

}

double N1(double t) // I интерполяционная формула Ньютона

{

double dy[] = { y[1] - y[0], y[2] - y[1] };

double result = y[0] + t \* dy[0] + t \* (t - 1) / 2 \* (dy[1] - dy[0]);

return result;

}

double N2(double t\_) // II интерполяционная формула Ньютона

{

double dy[] = { y[1] - y[0], y[2] - y[1] };

double result = y[2] + t\_ \* dy[1] + t\_ \* (t\_ + 1) / 2 \* (dy[1] - dy[0]);

return result;

}

void koefs\_P2()

{

double sumx[5], sumy, sumxy, sumxxy; // sumx[5], 5 - просто для удобства пользования нумерацией, 0 не используем

double det[4]; // Определители

sumx[1] = x[0] + x[1] + x[2];

sumx[2] = pow(x[0], 2.0) + pow(x[1], 2.0) + pow(x[2], 2.0);

sumx[3] = pow(x[0], 3.0) + pow(x[1], 3.0) + pow(x[2], 3.0);

sumx[4] = pow(x[0], 4.0) + pow(x[1], 4.0) + pow(x[2], 4.0);

sumy = y[0] + y[1] + y[2];

sumxy = x[0] \* y[0] + x[1] \* y[1] + x[2] \* y[2];

sumxxy = pow(x[0], 2.0) \* y[0] + pow(x[1], 2.0) \* y[1] + pow(x[2], 2.0) \* y[2];

det[0] = 3 \* sumx[2] \* sumx[4] + sumx[1] \* sumx[3] \* sumx[2] + sumx[2] \* sumx[1] \* sumx[3] - sumx[2] \* sumx[2] \* sumx[2] - sumx[3] \* sumx[3] \* 3 - sumx[4] \* sumx[1] \* sumx[1];

det[1] = sumy \* sumx[2] \* sumx[4] + sumx[1] \* sumx[3] \* sumxxy + sumx[2] \* sumxy \* sumx[3] - sumxxy \* sumx[2] \* sumx[2] - sumx[3] \* sumx[3] \* sumy - sumx[4] \* sumxy \* sumx[1];

det[2] = 3 \* sumxy \* sumx[4] + sumy \* sumx[3] \* sumx[2] + sumx[2] \* sumx[1] \* sumxxy - sumx[2] \* sumxy \* sumx[2] - sumxxy \* sumx[3] \* 3 - sumx[4] \* sumx[1] \* sumy;

det[3] = 3 \* sumx[2] \* sumxxy + sumx[1] \* sumxy \* sumx[2] + sumy \* sumx[1] \* sumx[3] - sumx[2] \* sumx[2] \* sumy - sumx[3] \* sumxy \* 3 - sumxxy \* sumx[1] \* sumx[1];

a[0] = det[1] / det[0];

a[1] = det[2] / det[0];

a[2] = det[3] / det[0];

}

double P2(double \_x) // Аппроксимационный формула

{

double result = a[2] \* pow(\_x, 2.0) + a[1] \* \_x + a[0];

return result;

}

void tab\_func()

{

double \_x = x[0], t1 = (x[0] - x[0]) / ((xn - x0) / 2.0), t2 = (x[0] - x[2]) / ((xn - x0) / 2.0);

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

cout << setprecision(3) << setw(2) << i << "|"

<< setw(7) << \_x << setw(2) << "|"

<< setw(8) << t1 << setw(2) << "|"

<< setw(9) << t2 << setw(2) << "|"

<< setprecision(8)

<< setw(15) << f(\_x) << setw(2) << "|"

<< setw(15) << L2(\_x) << setw(2) << "|"

<< setw(15) << abs(f(\_x) - L2(\_x)) << setw(4) << "|"

<< setw(15) << N1(t1) << setw(2) << "|"

<< setw(15) << abs(f(\_x) - N1(t1)) << setw(4) << "|"

<< setw(15) << N2(t2) << setw(2) << "|"

<< setw(15) << abs(f(\_x) - N2(t2)) << setw(4) << "|"

<< setw(15) << P2(\_x) << setw(2) << "|"

<< setw(15) << abs(f(\_x) - P2(\_x)) << setw(4) << "|"

<< endl;

\_x += hx;

t1 += ht;

t2 += ht;

}

}

void show\_table()

{

cout << "--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n";

cout << " i| xi | ti | t'i | f(xi) | L2(xi) | |(f(xi)-L2(xi))| | N1(xi) | |(f(xi)-N1(xi))| | N2(xi) | |(f(xi)-N2(xi))| | P2(xi) | |(f(xi)-P2(xi))| |\n";

cout << "--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n";

}

int main()

{

setlocale(0, "");

cout << "Лабораторная работа по Вычислительной математике No6-7\n" << endl;

cout << "Выполнили студенты Маннанов и Нигамадянов, гр. 4310\n" <<

endl;

cout << "ИНТЕРПОЛЯЦИЯ И АППРОКСИМАЦИЯ ФУНКЦИЙ" << endl;

cout << "Вариант No16:" << endl;

cout << "14 \* e^(5 + cos(x/10)), x принадлежит [" << x0 << ", " << xn << "]" << endl;

uzlovye\_tochki();

koefs\_P2();

show\_table();

tab\_func();

system("pause");

return 0;

}

Работа программы:

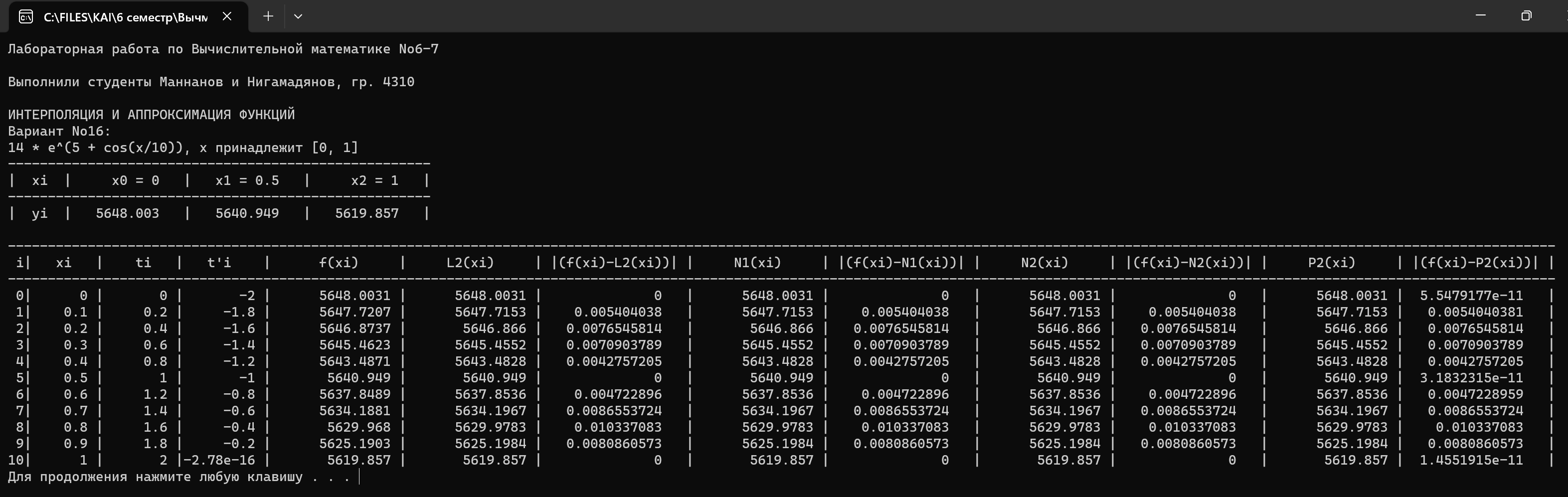


Рис. 3

Вывод: в результате с помощью разработанной программы мы можем говорить о точности построенных полиномов для функции с точностью для равноотстоящей системы из трех узловых точек на отрезке из области допустимых значений функции, где .