**Вариант 3**

I2302 Ciobanu Stanislav

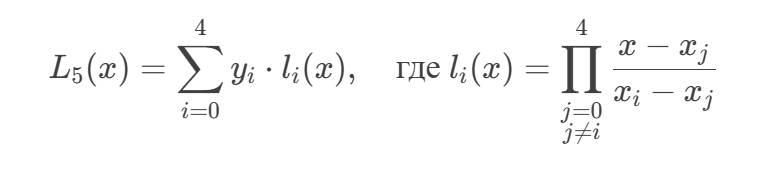
X: 0.259 0.841 1.562 2.304 2.856

Y: 0.018 -1.259 -1.748 -0.532 0.911

**Интерполяционный многочлен Лагранжа**

**Теория:**

Метод Лагранжа:



Вычисление базисных полиномов:

l0 = ((x - x1) (x - x2) / (x0 – x1) (x0 – x2)) \* y0

L(x) = y, L(x0) = y0

L(x) = l0 (x) + l1 (x) + …

**Решение:**

l0 = ((x - x1) (x - x2) (x – x3) (x – x4) / (x0 – x1) (x0 – x2) (x0 – x3) (x0 – x4)) \* y0 = 0.00447 \* (x - 0.841)\*(x - 1.562)\*(x - 2.304)\*(x - 2.856)

l1 = ((x – x0) (x - x2) (x – x3) (x – x4) / (x1 – x0) (x1 – x2) (x1 – x3) (x1 – x4)) \* y1 = 1.017 \* (x - 0.259)\*(x - 1.562)\*(x - 2.304)\*(x - 2.856)

l2 = ((x – x0) (x - x1) (x – x3) (x – x4) / (x2 – x0) (x2 – x1) (x2 – x3) (x2 – x4)) \* y2

= -1.942 \* (x - 0.259)\*(x - 0.841)\*(x - 2.304)\*(x - 2.856)

l3 = ((x – x0) (x - x1) (x – x2) (x – x4) / (x3 – x0) (x3 – x1) (x3 – x2) (x3 – x4)) \* y3 = 0.434 \* (x - 0.259)\*(x - 0.841)\*(x - 1.562)\*(x - 2.856)

l4 = ((x – x0) (x - x1) (x – x2) (x – x3) / (x4 – x0) (x4 – x1) (x4 – x2) (x4 – x3)) \* y4 = 0.2436 \* (x - 0.259)\*(x - 0.841)\*(x - 1.562)\*(x - 2.304)

L(x)= 0.00447⋅(x−0.841)(x−1.562)(x−2.304)(x−2.856) + 1.017⋅(x−0.259)(x−1.562)(x−2.304)(x−2.856)−1.942⋅(x−0.259)(x−0.841)(x−2.304)(x−2.856) + 0.434⋅(x−0.259)(x−0.841)(x−1.562)(x−2.856) + 0.2436⋅(x−0.259)(x−0.841)(x−1.562)(x−2.304).​

*L*(*x*)=−0.238*x*4+2.117*x*3−5.605*x*2+6.619*x*−2.523

**Интерполяционный многочлен Ньютона**

**Теория:**

Pn(x) =a0+ a1(x-x0) + a2(x-x0)(x-x1) + …+ an(x-x0)(x-x1)…(x-xn-1)

**Решение:**

Таблица разделённых разностей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Узлы (x) | Y | 1-я | 2-я | 3-я | 4-я |
| 0.259 | 0.018 | -2.1945 | 1,1637759 | 0.205591506 | -0.23787298 |
| 0.841 | -1.259 | -0.6781 | 1,58421053 | -0.412162333 |  |
| 1.562 | -1.748 | 1.6396 | 0.753709428 |  |  |
| 2.304 | -0.532 | 2.6149 |  |  |  |
| 2.856 | 0.911 |  |  |  |  |

Многочлен Ньютона

P(x)=

 0.018

−2.1945(x−0.259)

+1.1637759(x−0.259)(x−0.841)

+0.205591506(x−0.259)(x−0.841)(x−1.562)

−0.237872098(x−0.259)(x−0.841)(x−1.562)(x−2.304)​

Раскрытие скобок

1. Линейные члены:

−2.1945(*x*−0.259)=−2.1945*x*+0.568

2. Квадратичные члены:

(*x*−0.259)(*x*−0.841)=

x2−(0.259+0.841)x+(0.259⋅0.841)=x2−1.1x+0.218.*x*2−(0.259+0.841)*x*+(0.259⋅0.841)=*x*2−1.1*x*+0.218.

1. Кубические члены:

(*x*2−1.1*x*+0.218)(*x*−1.562)=*x*3−2.662*x*2+1.936*x*−0.340

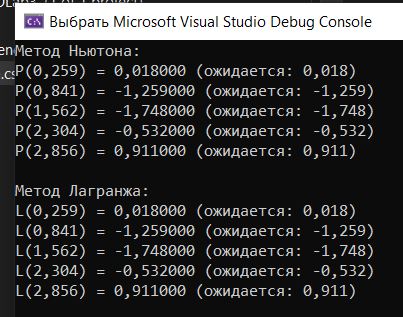
1. Четвёртая степень:  
   (x−0.259)(x−0.841)(x−1.562)(x−2.304)(*x*−0.259)(*x*−0.841)(*x*−1.562)(*x*−2.304)= −0.237872098*x*4+1.181*x*3−1.460*x*2+1.062*x*−0.187.
2. P(x)= −2.1945*x* + 0.568 + *x*2 − 1.1*x* + 0.218 + *x*3 − 2.662*x*2 + 1.936*x* − 0.340 − 0.237872098*x*4 + 1.181*x*3 − 1.460*x*2 + 1.062*x* − 0.187

*P*(*x*)=−0.238*x*4+2.117*x*3−5.605*x*2+6.619*x*−2.523 = *L*(*x*)

**Сравнение**

|  |  |  |
| --- | --- | --- |
| Критерий | Метод Ньютона | Метод Лагранжа |
| Способ получения | Построение через разделённые разности | Построение через базисные полиномы |
| Сложность | O(n2) | O(n2) |
| Удобство | Легко строить, а далее требуется лишь раскрыть скобки | Требуется провести много вычислений для каждого из базисных полиномов |

**Программа**



**Код**

using System;

namespace CombinedInterpolation

{

class Program

{

static void Main(string[] args)

{

double[] x = { 0.259, 0.841, 1.562, 2.304, 2.856 };

double[] y = { 0.018, -1.259, -1.748, -0.532, 0.911 };

double[,] divDiff = CalculateDividedDifferences(x, y);

// Передача массива y в методы

Console.WriteLine("Метод Ньютона:");

PrintResults(x, y, divDiff);

Console.WriteLine("\nМетод Лагранжа:");

PrintLagrangeResults(x, y);

}

// Добавляем параметр y в метод

static void PrintResults(double[] x, double[] y, double[,] divDiff)

{

foreach (var xi in x)

{

double result = NewtonInterpolate(x, divDiff, xi);

Console.WriteLine($"P({xi:F3}) = {result:F6} (ожидается: {y[Array.IndexOf(x, xi)]})");

}

}

// Остальной код без изменений

static double[,] CalculateDividedDifferences(double[] x, double[] y)

{

int n = x.Length;

double[,] divDiff = new double[n, n];

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

divDiff[i, 0] = y[i];

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

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

divDiff[i, j] = (divDiff[i + 1, j - 1] - divDiff[i, j - 1]) / (x[i + j] - x[i]);

return divDiff;

}

static double NewtonInterpolate(double[] x, double[,] divDiff, double targetX)

{

int n = x.Length;

double result = divDiff[0, 0];

double term;

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

{

term = divDiff[0, i];

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

term \*= (targetX - x[j]);

result += term;

}

return result;

}

static double L(int i, double[] x, double value)

{

double result = 1.0;

for (int j = 0; j < x.Length; j++)

{

if (j != i)

{

result \*= (value - x[j]) / (x[i] - x[j]);

}

}

return result;

}

static double LagrangeInterpolate(double[] x, double[] y, double value)

{

double result = 0.0;

for (int i = 0; i < x.Length; i++)

{

result += y[i] \* L(i, x, value);

}

return result;

}

static void PrintLagrangeResults(double[] x, double[] y)

{

foreach (var xi in x)

{

double result = LagrangeInterpolate(x, y, xi);

Console.WriteLine($"L({xi:F3}) = {result:F6} (ожидается: {y[Array.IndexOf(x, xi)]})");

}

}

}

}