Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

Лабораторная работа №4

“Численные методы и методы оптимизации”

Тема:”Интерполяционные многочлены Лагранжа и Ньютона”

Проверил: Верлан Игорь

Выполнил: Чобану Артём

Группа: i1902

Кишинев 2021

**Код программы можно получить и по ссылке:**

https://github.com/ArtiomCiobanu/NumericalCalculation\_Labs

**Задание:**

Построить интерполяционные многочлены:

1. Лагранжа
2. Ньютона

**Решение:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | 0 | 0.25 | 0.5 | 1 | 1.5 | 2 |
| Y | 1 | 0.890625 | 1.125 | 3 | 7.375 | 15 |

1. **Интерполяционный многочлен Лагранжа**

Код программы:

//Интерполируемая функция

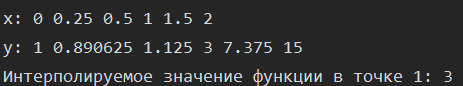
private static double F(double x)

=> Math.Pow(x, 3) + 2 \* Math.Pow(x, 2) - x + 1;  
  
private static void Main()  
{  
 double[] x = {0, 0.25, 0.5, 1, 1.5, 2};  
 double[] y = new double[x.Length];

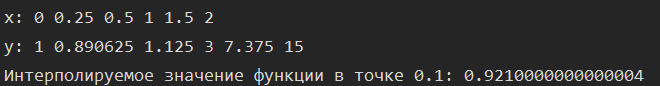
//= {1, 0.890625, 1.125, 3, 7.375, 15};  
 for (int i = 0; i < x.Length; i++)  
 {  
 y[i] = F(x[i]);  
 }  
  
 Console.Write("x: ");  
 x.Display();  
 Console.Write("y: ");  
 y.Display();  
  
 double xToInterpolate = 1;  
 double result = 0;  
 for (int i = 0; i < x.Length; i++)  
 {  
 var numerator = CountNumerator(i, xToInterpolate, x);  
 var denomenator = CountDenomenator(i, x);  
  
 result += y[i] \* numerator / denomenator;  
 }  
 Console.WriteLine($"Интерполируемое значение функции в точке {xToInterpolate}: {result}");  
}

private static double CountNumerator(int i, double xToInterpolate, double[] x)  
{  
 double result = 1;  
 for (int j = 0; j < x.Length; j++)  
 {  
 if (j != i)  
 {  
 result \*= xToInterpolate - x[j];  
 }  
 }  
  
 return result;  
}  
  
private static double CountDenomenator(int i, double[] x)  
{  
 double result = 1;  
  
 for (int j = 0; j < x.Length; j++)  
 {  
 if (j != i)  
 {  
 result \*= x[i] - x[j];  
 }  
 }  
  
 return result;  
}

Результат в одной из данных точек:



Проверка в любой другой точке:



1. Интерполяционный многочлен Ньютона

private static void Newton(

double[] x,

double[] y,

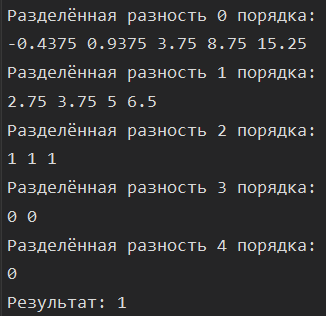
double xToInterpolate)  
{  
 double result = y[0];  
  
 double[] previousGradeDifferences = y;  
 for (int grade = 0; grade < x.Length - 1; grade++)  
 {  
 double current = 0;  
  
 double[] currentGradeDifferences =

new double[x.Length - grade - 1];  
  
 Console.WriteLine($"Разделённая разность {grade} порядка: ");  
 for (int i = 0; i < previousGradeDifferences.Length - 1; i++)  
 {  
 var difference = CountDividedDifference(x, previousGradeDifferences, i, grade);  
 Console.Write(difference + " ");  
  
 currentGradeDifferences[i] = difference;  
 }  
  
 current = currentGradeDifferences[0];  
  
 for (int i = 0; i < x.Length - grade; i++)  
 {  
 current \*= xToInterpolate - x[i];  
 }  
  
 result += current;  
  
 previousGradeDifferences = currentGradeDifferences;  
  
 Console.WriteLine();  
 }  
  
 Console.WriteLine($"Результат: {result}");  
}  
  
private static double CountDividedDifference(

double[] x, double[] y,

int i, int grade)  
{  
 var right = i + grade + 1;  
 return (y[i + 1] - y[i]) / (x[right] - x[i]);  
}

Вывод программы (при x = 1):



При x = 0.1:

