Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра « »

Лабораторная работа №1

по дисциплине «Геометрическое моделирование»

Исследование интерполирующей кривой Лагранжа

Выполнил: студент гр. − qwinmen

Проверил: С. А.

Тамбов,

***Цели и задачи****.*

Реализовать математическую часть интерполяционного многочлена Лагранжа и выявить его недостатки.

***Решение задачи****.*

Интерполяционным многочленом Лагранжа называется многочлен

1

Этот многочлен удовлетворяет условиям . Тут узлы (или полюсы) интерполяции, заданные числа.

Интерполяционной формулой Лагранжа называется формула

2

Производя интерполирование функции по формуле Лагранжа, заменяют эту функцию полиномом совпадающим с ней в n+1 данных точках отрезка [a;b]. В остальных точках этого отрезка разность отлична от нуля и представляет собой погрешность метода.

Пример расчетов для точек

При n=2 формула 1 имеет вид

Подставляя в эту формулу заданные значения, находим

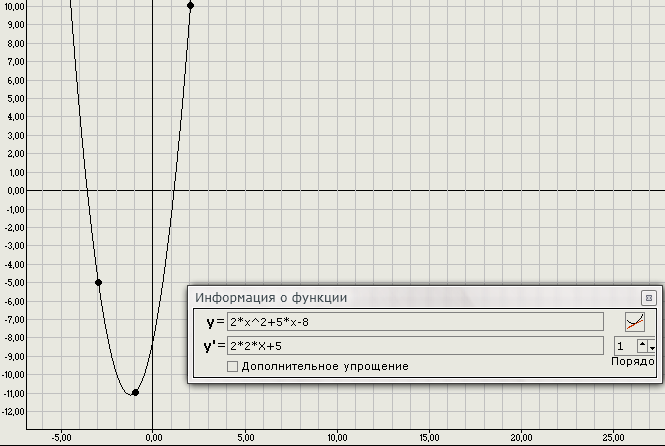
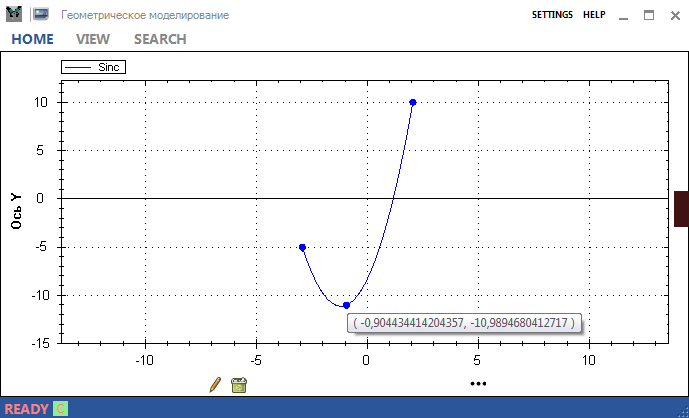
Итак, Теперь строим график функции (Рисунок 1) через специальное по:

Рисунок 1 График функции

Проверить полученный результат расчетов, а заодно и правильность кода программы, можно указав те же три точки для построения (Рисунок 2):

Рисунок 2 Тестирование программы по точкам

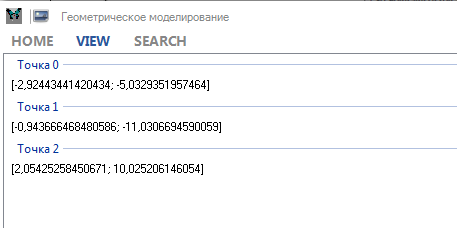
Поставленные точки выводятся в отдельную вкладку View и группируются по очередности добавления на холст (Рисунок 3):

Рисунок 3 Координаты точек на холсте

Полученный график идентичен эталону, а это подтверждает правильность работы матчасти программы. Расчеты сводятся к нескольким управляющим методам и сборочным (вспомогательным). Такой подход обеспечивает независимость обработки количества точек.

Построение (матчасть) начинается с события

/// <summary>

/// Точки сняты, рассчитать формулу

/// </summary>

**private** void btnBuild\_Click**(object** sender**,** EventArgs e**)**

**{**

\_pointArr **=** **new** PointD**[**\_lpointArr**.**Count**];**

\_lpointArr**.**CopyTo**(**\_pointArr**);**

\_num **=** \_pointArr**.**Length**;**

#region построить график

int n **=** \_num **-** 1**;**//n скобок на 1 дробьЧислитель

double**[]** res **=** **new** double**[**\_num**];**

// Интервал, где есть данные

double xmin **=** \_pointArr**[**0**].**X**;**//-100;

double xmax **=** \_pointArr**[**n**].**X**;**//100;

var text **=** ""**;**

**foreach** **(**PointD pointD **in** \_pointArr**)**

**{**

text **+=** "[" **+** pointD**.**X **+** "; " **+** pointD**.**Y **+** "] "**;**

**}**

MessageBox**.**Show**(**text**);**

// Заполняем список точек

//\_list.Add(xmin, \_pointArr[0].Y);

**for** **(**double x **=** xmin**;** x **<=** xmax**;** x **+=**0.01**)**

**{**

//\_num //Количество дробей == количеству точек

**for** **(**int i **=** 0**;** i **<** \_num**;** i**++)**

**{**

//за один такт вычислить 1 дробь

//1)составить скобки в дроби

double resLocDrob **=** СобратьЧислитель**(**n**,** x**,** i**)/**СобратьЗнаменатель**(**n**,** x**,** i**);**

//Итог в i-дроби есть число:

res**[**i**]** **=** \_pointArr**[**i**].**Y **\*** resLocDrob**;**

**}**

double tmp **=** 0.0**;**

**foreach** **(**double re **in** res**)**

**{**//просуммировать итог(результат n-дробей)

tmp **+=** re**;**

**}**

//SetPointCurve(x,tmp);

// добавим в список точку

\_list**.**Add**(**x**,** tmp**);**

**}**

DrawGraph**();**

#endregion

**}**

Расчет числителя и знаменателя дроби обрабатываются двумя методами для упрощения структуры программы

/// <summary>

/// Собрать числитель дроби

/// </summary>

/// <param name="n">Количество скобок</param>

/// <param name="x">Точка для которой ищем У</param>

/// <param name="i">Номер дроби для генератора</param>

**private** double СобратьЧислитель**(**int n**,** double x**,** int i**)**

**{**

double temp **=** 0.0**;**

//Общий вид: (x-generator)\*(x-generator)

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

**{**

//собрать скобку

**if** **(**temp **==** 0.0**)**

temp **=** **(**x **-** generator**(**i**,** j**));**

**else** temp **=** temp **\*** **(**x **-** generator**(**i**,** j**));**

**}**

**return** temp**;**

**}**

/// <summary>

/// Собрать знаменатель дроби

/// </summary>

/// <param name="n">Количество скобок</param>

/// <param name="x">Точка для которой ищем У</param>

/// <param name="i">Номер дроби для генератора</param>

**private** double СобратьЗнаменатель**(**int n**,** double x**,** int i**)**

**{**

double temp **=** 0.0**;**

//Общий вид: (xi-generator)\*(xi-generator)

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

**{**

//собрать скобку

**if** **(**temp **==** 0.0**)**

temp **=** **(**\_pointArr**[**i**].**X **-** generator**(**i**,** j**));**

**else** temp **=** temp **\*** **(**\_pointArr**[**i**].**X **-** generator**(**i**,** j**));**

**}**

**return** temp**;**

**}**

**private** int \_gindex**;**

/// <summary>

/// Определить какой Х использовать

/// </summary>

/// <param name="i">Порядковый номер дроби</param>

/// <param name="j">Текущая скобка номер</param>

/// <returns></returns>

**private** double generator**(**int i**,** int j**)**

**{**

**if** **(**j **==** 0**)**

\_gindex **=** i **+** 1**;**

**else** \_gindex**++;**

**if** **(**\_gindex **>=** \_pointArr**.**Length**)**

\_gindex **=** 0**;**

**return** \_pointArr**[**\_gindex**].**X**;**

**}**

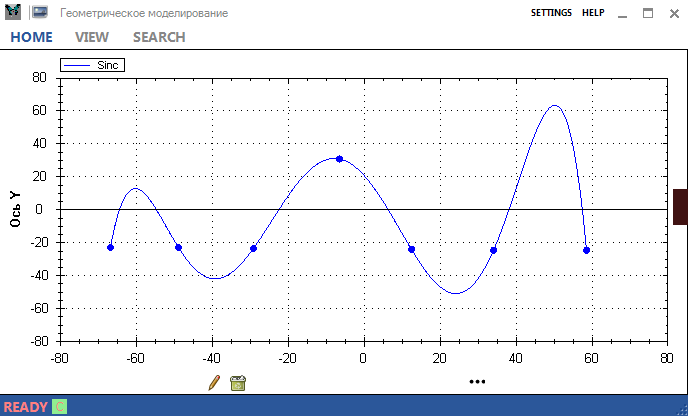
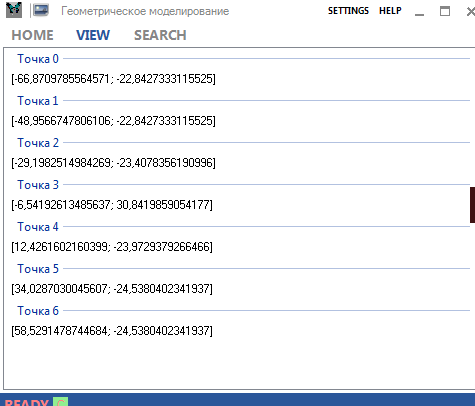
***Тест на семь точек***

Рисунок 4 Построение через семь точек

Рисунок 5 Координаты точек

***Вывод.***

Подход хорош для простеньких задач на построение графиков без сложных переходов, задачи на отрисовку сложных фигур становятся проблемой, так как при наличии более трех точек в графике возникают бесконтрольные всплески, что в конечном виде портит всю задачу.