Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра « »

Лабораторная работа №3

по дисциплине «Геометрическое моделирование»

Построение кривой Эрмита

Выполнил: студент гр. −31 Д. Ю.

Проверил: С. А.

Тамбов,

***Цели и задачи****.*

Исследовать алгоритм Эрмита и определить характерные особенности в применении векторов кривизны.

***Решение задачи****.*

Для построения отрезка необходимы четыре точки – это минимум в алгоритме. При этом две точки являются направляющими векторами (рисунок 1).

Рисунок Пример кривой Эрмита

Формула записывается следующим образом (две точки, два вектора):

Для n-количества точек формулу P(t) вычисляют n-1 раз:

\_ \_ \_

Вот участки кода, которые вычисляют значения точек и векторов в программе:

/// <summary>

/// Построить

/// </summary>

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

**{**

//проверить, все ли точки имеют вектор//если нет, нестроить

**if(**\_lpointArr**.**Count**%**2**!=**0**)** **return;**

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

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

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

var res **=** **new** PointD**[**\_num**];**

\_list**.**Clear**();**

#region построить график

var tmpArrP **=** **new** PointD**[**4**];**

//Note: количество отрезков = \_lpointArr.Count/4

**for** **(**int o **=** 2**,** iter **=** 0**;** o **<** \_lpointArr**.**Count**;** o**+=**2**,** iter**+=**2**)**

**{**//построить отрезок точка-вектор и точка-вектор:

tmpArrP**[**0**]** **=** **new** PointD**(**\_pointArr**[**iter**].**X**,** \_pointArr**[**iter**].**Y**);** //P1:0//2//4//..

tmpArrP**[**1**]** **=** **new** PointD**(**\_pointArr**[**iter **+** 1**].**X**,** \_pointArr**[**iter **+** 1**].**Y**);** //R1:1//3//5//..

tmpArrP**[**2**]** **=** **new** PointD**(**\_pointArr**[**iter **+** 2**].**X**,** \_pointArr**[**iter **+** 2**].**Y**);** //P4:2//4//6//..

tmpArrP**[**3**]** **=** **new** PointD**(**\_pointArr**[**iter **+** 3**].**X**,** \_pointArr**[**iter **+** 3**].**Y**);** //R4:3//5//6//..

**for** **(**double t **=** 0.01**;** t **<=** 1**;** t **+=** 0.01**)**

**{**

**for** **(**int i **=** 0**;** i **<** 4**;** i**++)**

**{**//для i-точки вычислить

//vector() OR point()

**if(**i**%**2**==**0**)** res**[**i**]** **=** Точка**(**i**,** t**,** tmpArrP**[**i**]);**

**else**

**{**//Note: vector = R[n] - P[n-1]

var vector **=** **new** PointD**(**tmpArrP**[**i**].**X **-** tmpArrP**[**i **-** 1**].**X**,**

tmpArrP**[**i**].**Y **-** tmpArrP**[**i **-** 1**].**Y**);**

res**[**i**]** **=** Вектор**(**i**,** t**,** vector**);**

**}**

**}**

var tmp **=** **new** PointD**();**

**foreach** **(**var pointD **in** res**)**

**{**

tmp**.**X **+=** pointD**.**X**;**

tmp**.**Y **+=** pointD**.**Y**;**

**}**

// добавим в список точку P(t)

\_list**.**Add**(**tmp**.**X**,** tmp**.**Y**);**

**}**

**}**

#endregion

DrawGraph**();**

**}**

/// <summary>

/// Вычислить P\*(t)

/// </summary>

**private** PointD Точка**(**int i**,** double t**,** PointD point**)**

**{**

**if** **(**i **==** 0**)**//p1:

**return** **new** PointD**(**point**.**X**\*(**2.0**\***t**\***t**\***t **-** 3.0**\***t**\***t **+** 1.0**),** point**.**Y**\*(**2.0**\***t**\***t**\***t **-** 3.0**\***t**\***t **+** 1.0**));**

//p4:

**return** **new** PointD**(**point**.**X**\*(-**2.0**\***t**\***t**\***t **+** 3.0**\***t**\***t**),** point**.**Y**\*(-**2.0**\***t**\***t**\***t **+** 3.0**\***t**\***t**));**

**}**

/// <summary>

/// Вычислить R\*(t)

/// </summary>

**private** PointD Вектор**(**int i**,** double t**,** PointD point**)**

**{**

**if** **(**i **==** 1**)**//r1

**return** **new** PointD**(**point**.**X **\*** **(**t **\*** t **\*** t **-** 2.0 **\*** t **\*** t **+** t**),** point**.**Y **\*** **(**t **\*** t **\*** t **-** 2.0 **\*** t **\*** t **+** t**));**

//r4

**return** **new** PointD**(**point**.**X **\*** **(**t **\*** t **\*** t **-** **(**t **\*** t**)),** point**.**Y **\*** **(**t **\*** t **\*** t **-** **(**t **\*** t**)));**

**}**

Алгоритм кода следующий: из массива точек берем первые четыре, прогоняем в цикле с шагом 0.01 по t, решаем формулу P(t) и вносим результат в массив на вывод. После этого смещаемся на 2 ячейки в массиве с точками и работаем по точкам 2, 3, 4, 5. Самое главное - не допустить момента, когда точке нехватает её вектора.

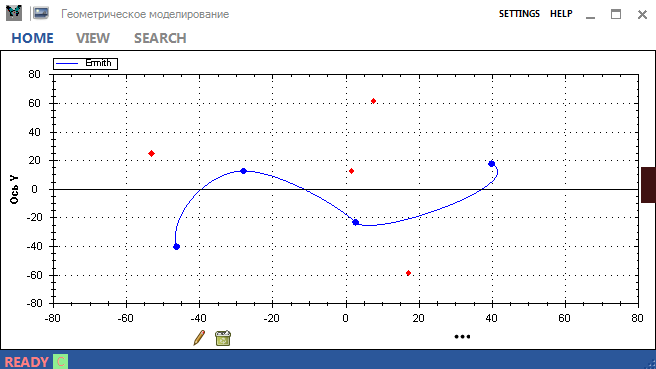
Пример работы с программой приведен на рисунке 2:

Рисунок Пример из 4 точек

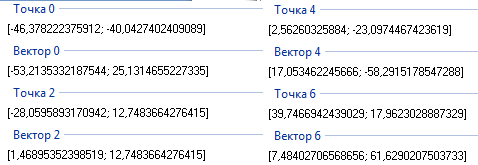
Соответственно, координаты точек на рисунке 3:

Рисунок Координаты точек кривой

***Вывод.***

Алгоритм достаточно прост в изучении, имеет широкие возможности для построения сложных фигур. Вектора кривизны позволяют весьма тонко задать изгиб нужного участка кривой, характерных минусов не выявил.