

[articles](#) [Q&A](#) [forums](#) [lounge](#)

Search for articles, questions, tips



Curve Fitting using Lagrange Interpolation

**Fady Aladdin**, 12 Sep 2008

CPOL

Rate:

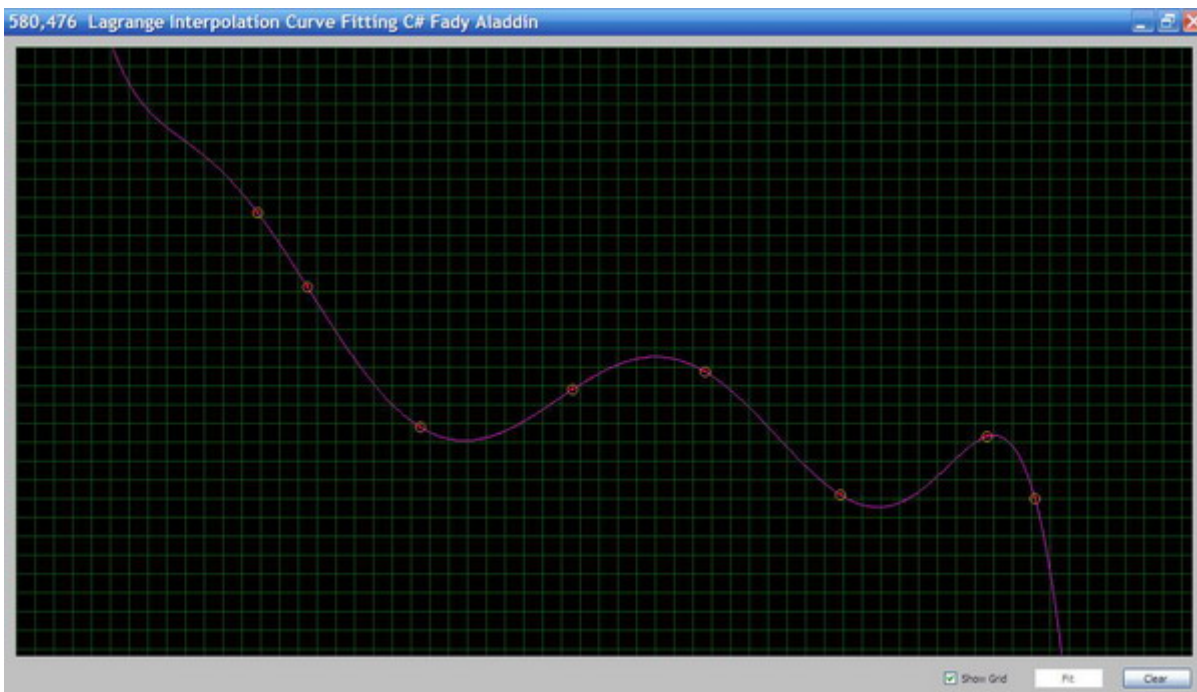


4.55 (21 votes)

Fitting a set of points P to a curve using Lagrange Interpolation Polynomial.



Is your email address OK? You are signed up for our newsletters but your email address is either unconfirmed, or has not been reconfirmed in a long time. Please [click here to have a confirmation email sent](#) so we can confirm your email address and start sending you newsletters again. Alternatively, you can [update your subscriptions](#).

[Download source code - 20.9 KB](#)

Introduction

In this article, I will explain curve fitting using the Lagrange interpolation polynomial. Curve fitting is used in a wide spectrum in engineering applications such as cars and air crafts surface design. The main problem is, given a set of points in the plan, we want to fit them in a smooth curve that passes through these points. The order of the curve $f(x)$ depends on the number of points given. For example, if we have two points, the function will be of the first order, and the curve will be the line that passes through these two points, while if you have three points, the function will be of the second order $f(x) = x^2$. Let's first explain the Lagrange polynomial, then we will proceed to the algorithm and the implementation. In this article, I am using C# for coding.

Background

Lagrange Polynomial

An interpolation on two points, (x_0, y_0) and (x_1, y_1) , results in a linear equation or a straight line. The standard form of a linear equation is given by $y = mx + c$, where m is the gradient of the line and c is the y -intercept.

[Hide](#) [Copy Code](#)

$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

$$c = y_0 - mx_0$$

which results in:

[Hide](#) [Copy Code](#)

$$y = \left(\frac{y_1 - y_0}{x_1 - x_0} \right) * x + (x_1 y_0 - x_0 y_1) / (x_1 - x_0)$$

The linear equation is rewritten so that the two interpolated points, (x_0, y_0) and (x_1, y_1) , are directly represented.

With this in mind, the linear equation is rewritten as:

[Hide](#) [Copy Code](#)

$$P_1(x) = a_0(x - x_1) + a_1(x - x_0)$$

where a_0 and a_1 are constants. The points x_0 and x_1 in the factors of the above equation are called the *centers*. Applying the equation at (x_0, y_0) , we obtain:

[Hide](#) [Copy Code](#)

$$y_0 = a_0(x_0 - x_1) + a_1(x_0 - x_0)$$

or

$$a_0 = y_0 / (x_0 - x_1)$$

At (x_1, y_1) , we get:

[Hide](#) [Copy Code](#)

$$y_1 = a_0(x_1 - x_1) + a_1(x_1 - x_0), \text{ or } a_1 = y_1 / (x_1 - x_0)$$

Therefore, the linear equation becomes:

[Hide](#) [Copy Code](#)

$$P_1(x) = y_0 (x - x_1) / (x_0 - x_1) + y_1 (x - x_0) / (x_1 - x_0)$$

The quadratic form of the Lagrange polynomial interpolates three points, (x_0, y_0) , (x_1, y_1) , and (x_2, y_2) . The polynomial has the form:

[Hide](#) [Copy Code](#)

$$P_2(x) = a_0(x - x_1)(x - x_2) + a_1(x - x_0)(x - x_2) + a_2(x - x_0)(x - x_1)$$

with centers at x_0, x_1 , and x_2 . At (x_0, y_0) :

[Hide](#) [Copy Code](#)

$$y_0 = a_0(x_0 - x_1)(x_0 - x_2) + a_1(x_0 - x_0)(x_0 - x_2) + a_2(x_0 - x_0)(x_0 - x_1),$$

or

$$a_0 = y_0 / ((x_0 - x_1)(x_0 - x_2))$$

$$a_1 = y_1 / ((x_1 - x_0)(x_1 - x_2))$$

$$a_2 = y_2 / ((x_2 - x_0)(x_2 - x_1))$$

$$P_2(x) = y_0 (x - x_1)(x - x_2) / ((x_0 - x_1)(x_0 - x_2)) + y_1 (x - x_0)(x - x_2) / ((x_1 - x_0)(x_1 - x_2)) + y_2 (x - x_0)(x - x_1) / ((x_2 - x_0)(x_2 - x_1))$$

In general, a Lagrange polynomial of degree n is a polynomial that is produced from an interpolation over a set of points, (x_i, y_i) for

$i = 0, 1, \dots, n$, as follows:

[Hide](#) [Copy Code](#)

$$P_n(x) = y_0L_0(x) + y_1L_1(x) + \dots + y_nL_n(x)$$

Using the Code

The Algorithm

[Hide](#) [Copy Code](#)

```
Given the interpolating points (xi , yi ) for i =0, 1, . . . ,n;
for i = 0 to n
    //the cumulative multiplication from k = 1 (and k not equal i) to n
    Evaluate Li (x) = ∏nk=1,k != i (x-xk ) / (xi-xk ) ;
endfor
Evaluate Pn(x) = y0L0(x) + y1L1(x) + ... + ynLn(x);
```

Download the source code from the top of this page to view the code.

Points of Interest

Numerical computing is a very interesting field in software development. This article is related to that field ... And, I will post more articles soon about Computational Geometry ...such as Convex Hull algorithm and Triangulation.

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

Share

EMAIL

About the Author



Fady Aladdin No Biography provided

Software Developer

Egypt

You may also be interested in...

[Interpolation from Polynomial to Natural Splines](#)

[#COBOLrocks TechCasts: New tricks for COBOL devs](#)

[CCurveDlg - Curve Interpolation](#)

[ES6 Arrow functions - The new fat and concise syntax in JavaScript](#)

[C# Cubic Spline Interpolation](#)

[Strong Authentication and the Road to FIDO](#)

Comments and Discussions

[First](#)
[Prev](#)
[Next](#)

A compact version **gustavtr** 21-Feb-12 5:24

Re: A compact version **Member 11943593** 30-Aug-15 3:18

My vote of 5 **MarkDaniel** 6-Jan-12 6:21

how to get all pixels coordinate from curve fitting using Lagrange interpolationan on an image **MrKyaw** 25-Oct-11 10:18

Radius of curve **MisterLister** 26-Aug-10 3:48




need help plz **d_e_e_o** 6-Aug-10 22:33

Corrected an Overflow error **SBGTrading** 22-Jul-10 23:05

Contact **icae** 27-Apr-09 23:42

Re: Contact **Fady Aladdin** 28-Apr-09 1:19

Area under the curve

Dorisvaldo 17-Sep-08 9:06**piecewise bezier** **Sarath.** 15-Sep-08 15:54Re: piecewise bezier **catbertfromgilead** 16-Sep-08 3:31**I don't mean to quibble but this looks like interpolation** **MicroImaging** 12-Sep-08 5:15Re: I don't mean to quibble but this looks like interpolation **axelriet** 12-Sep-08 23:30Re: I don't mean to quibble but this looks like interpolation **MicroImaging** 13-Sep-08 3:20**Awesome! Just what I've been looking for!** **dybs** 7-Sep-08 1:42Re: Awesome! Just what I've been looking for! **Fady Aladdin** 11-Sep-08 20:36Re: Awesome! Just what I've been looking for! **dybs** 12-Sep-08 0:32Re: Awesome! Just what I've been looking for! **axelriet** 12-Sep-08 23:26**Computational Geometry** **abombardier** 3-Sep-08 8:37Re: Computational Geometry **axelriet** 12-Sep-08 23:10Re: Computational Geometry **abombardier** 13-Sep-08 3:26**getting back values** **KJJ2** 3-Sep-08 4:21[Refresh](#)

1

[General](#)
[News](#)
[Suggestion](#)
[Question](#)
[Bug](#)
[Answer](#)
[Joke](#)
[Praise](#)
[Rant](#)
[Admin](#)

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Terms of Use](#) | [Mobile](#)

Web01 | 2.8.160621.1 | Last Updated 12 Sep 2008

請選取語言 ▼

Layout: [fixed](#) | [fluid](#)Article Copyright 2008 by Fady Aladdin
Everything else Copyright © [CodeProject](#), 1999-2016