

# Cubic Spline Interpolation, Least Squares Curve Fitting, Use of Software

Mike Renfro

March 26, 2008

# Review of Previous Lecture

# Cubic Spline Interpolation, Least Squares Curve Fitting, Use of Software

## Cubic Spline Interpolation

- Basics

- Piecewise Cubic Constraint Equations

- Lagrangian Option to Reduce Number of Equations

## Least-Squares Curve Fitting

- Linear Regression

- Linear Regression Example

- Nonlinear Regression

## Use of Software

- Excel

- MATLAB

# Part I

## Review of Previous Lecture

## Review of Previous Lecture

- Relevance of Curve Fitting, Relevance of Interpolation
- Spline Interpolation: step function, linear, quadratic

## Part II

# Cubic Spline Interpolation, Least-Squares Curve Fitting, Use of Software

# Basics of Cubic Spline Interpolation

A third-order (cubic) polynomial spline will be continuous, and also have continuous first and second derivatives. If you use cubic spline interpolation to position an object, you'll be assured of having a continuous position, velocity, and acceleration on the part.

# Piecewise Cubic Polynomial

We have  $n$  intervals and  $n + 1$  data points  $(x_0, f(x_0)) \cdots (x_n, f(x_n))$ . The  $i$ th interval, containing  $x$  values  $x_{i-1}$  and  $x_i$ , has a polynomial function given by

$$f_i(x) = a_i + b_i x + c_i x^2 + d_i x^3$$

The four coefficients in this particular polynomial are part of a set of  $4n$  coefficients when we consider all  $n$  intervals and polynomials. This means we'll need  $4n$  independent equations to solve for them.

# Piecewise Cubic Constraint Equations

- The function value at any interior knot  $x_i$  must be  $f(x_i)$  regardless of if it's calculated as  $f_i(x_i)$  or  $f_{i+1}(x_i)$ . This gives  $2n - 2$  conditions.
- The first and last functions  $f_1(x)$  and  $f_n(x)$  must pass through the first and last endpoints  $(x_0, f(x_0))$  and  $(x_n, f(x_n))$ . This gives 2 more conditions.
- The first derivative at any interior knot must be continuous.  $f'_i(x_i) = f'_{i+1}(x_i)$ . This gives  $n - 1$  conditions.
- The second derivative at any interior knot must also be continuous.  $f''_i(x_i) = f''_{i+1}(x_i)$ . Another  $n - 1$  conditions show up here.
- 2 conditions remain, and we have some options. One common pair of conditions is to assume that the curvature at the endpoints  $x_0$  and  $x_n$  is zero.

## Lagrangian Option to Reduce Number of Equations

Equations 5.92–5.95 on p.394–395 leads to the development of a more efficient way of calculating the equation coefficients. Rather than solving  $4n$  equations, we can solve  $n - 1$  related equations. Use this method if there are a large number of points, but for a limited number of points, MATLAB should handle the systems of equations without problems.

# Least-Squares Curve Fitting

Experimental data always has a finite amount of error included in it, due to both accumulated instrument inaccuracies and also imperfections in the physical system being measured. Even data describing a linear system won't all fall on a single straight line. Least-squares curve fitting is a method to find parameters that fit the error-laden data as best we can.

# Linear Regression

*Linear regression* is the method of finding the slope and  $y$  intercept of a line that best fits a set of data, and is the most common least-squares method used in mechanical engineering.

- Consider a series of data points  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ .
- A linear function that attempts to fit this data would be  $y = a_0 + a_1x$ , where  $a_0$  and  $a_1$  are constants not yet determined.
- After plugging each known value  $x_i$  into the equation, we'll find some amount of error between the  $y_i$  points in the original data the the predicted  $y$  value from the linear model. This error  $e_i$  is found with the relationship  $e_i = y_i - a_0 - a_1x_i$ .

## Linear Regression (continued)

The error  $e_i$  at each point may be positive or negative, large or small. One way to quantify how good a fit a particular line is would be to:

- Treat negative and positive errors the same.
- Penalize lines that have large errors at one or more points.
- Don't encourage lines that fit a few points exactly if they fit several other points poorly.

One easy way to achieve these goals is to square the individual errors and add them all up:

$$S = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - a_0 - a_1 x_i)^2$$

## Linear Regression (continued)

We can treat the sum of the squares of the error  $S$  as a function of  $a_0$  and  $a_1$ . An optimal pair of  $a_0$  and  $a_1$  values would minimize  $S$ .

- How do we find these optimal  $a_0$  and  $a_1$  values?
- Minimize  $S$  by finding values of  $a_0$  and  $a_1$  where the derivatives of  $S$  with respect to  $a_0$  and  $a_1$  are zero simultaneously.

$$\frac{\delta S}{\delta a_0} = -2 \sum_{i=1}^n (y_i - a_0 - a_1 x_i) = 0$$

$$\frac{\delta S}{\delta a_1} = -2 \sum_{i=1}^n x_i (y_i - a_0 - a_1 x_i) = 0$$

# Linear Regression (continued)

Reformat the previous two equations as

$$\sum_{i=1}^n y_i - \sum_{i=1}^n a_0 - a_1 \sum_{i=1}^n x_i = 0$$
$$\sum_{i=1}^n x_i y_i - a_0 \sum_{i=1}^n x_i - a_1 \sum_{i=1}^n x_i^2 = 0$$

## Linear Regression (continued)

Since  $\sum_{i=1}^n a_0 = na_0$ , these two equations can be reformatted as two simultaneous linear algebraic equations with  $a_0$  and  $a_1$  as the unknowns:

$$\begin{aligned} na_0 + \left( \sum_{i=1}^n x_i \right) a_1 &= \sum_{i=1}^n y_i \\ \left( \sum_{i=1}^n x_i \right) a_0 + \left( \sum_{i=1}^n x_i^2 \right) a_1 &= \sum_{i=1}^n x_i y_i \end{aligned}$$

# Linear Regression (continued)

In matrix form:

$$\begin{bmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{Bmatrix}$$

## Linear Regression (continued)

Solve the matrix form of the equations with Cramer's rule:

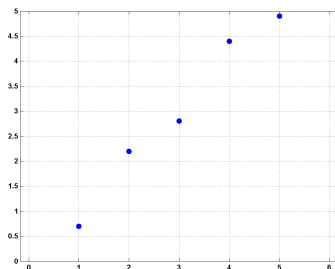
$$a_0 = \frac{\begin{vmatrix} \sum_{i=1}^n y_i & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i y_i & \sum_{i=1}^n x_i^2 \end{vmatrix}}{\begin{vmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{vmatrix}} = \frac{\sum_{i=1}^n y_i \sum_{i=1}^n x_i^2 - \sum_{i=1}^n x_i \sum_{i=1}^n x_i y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}$$

$$a_1 = \frac{\begin{vmatrix} n & \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i y_i \end{vmatrix}}{\begin{vmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{vmatrix}} = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}$$

# Example 5.13

Use linear regression to fit the following data points

$i$	1	2	3	4	5
$x_i$	1	2	3	4	5
$y_i$	0.7	2.2	2.8	4.4	4.9



## Example 5.13 (continued)

For this problem:

$$n = 5$$

$$\sum_{i=1}^n x_i = 1 + 2 + 3 + 4 + 5 = 15$$

$$\sum_{i=1}^n y_i = 0.7 + 2.2 + 2.8 + 4.4 + 4.9 = 15.0$$

$$\sum_{i=1}^n x_i^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$$

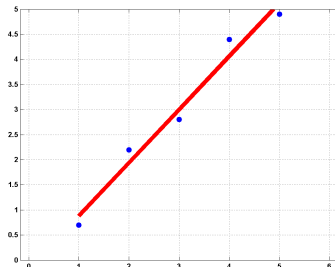
$$\sum_{i=1}^n x_i y_i = 1(0.7) + 2(2.2) + 3(2.8) + 4(4.4) + 5(4.9) = 55.6$$

## Example 5.13 (continued)

$$a_0 = \frac{15(55) - 15(55.6)}{5(55) - 15^2} = -0.18$$

$$a_1 = \frac{5(55.6) - 15(15.0)}{5(55) - 15^2} = 1.06$$

So  $y = -0.18 + 1.06x$ :



# Nonlinear Regression

Higher-order polynomials can be used as regression functions with a similar method. The  $S$  function becomes a function of  $m + 1$  variables, and we solve  $m + 1$  equations for these unknowns. The basic procedure also applies for other nonlinear functions.

## Excel: Advantages and Capabilities

Excel does make a variety of least-squares curve fits easy, not just the linear least-squares method that we covered. Excel refers to any of these types of functions as *trendlines*.

Trendline functions available in Excel:

- Linear:  $y = mx + b$
- Polynomial (2nd–6th degree):  
$$y = b + c_1x + c_2x^2 + c_3x^3 + \cdots + c_6x^6$$
- Logarithmic:  $y = c \ln x + b$
- Exponential:  $y = ce^{bx}$
- Power:  $y = cx^b$

Excel can also smooth noisy data by applying a *moving average*, where the current interpolated point is calculated by averaging the last few data points. This doesn't yield any explicit interpolating equations, but reduces fluctuations in the data.

## Excel: Disadvantages and Limitations

Spline fits are not possible in Excel without installing some third-party add-ons, or writing the formulas to calculate the spline coefficients yourself.

## Excel: Adding Trendlines for Curve Fitting

Given: an  $x - y$  plot of data in Excel

- Right-click on the data series in the chart, and select **Add trendline**.
- On the Add Trendline dialog, select the **Type** tab, and pick the type of function you want to fit to the data. If you select Polynomial or Moving Average, also select the order of the fit function.
- Select the **Options** tab, and click a checkmark in the **Display equation on chart** and **Display R-squared value on chart** boxes.
- Click the **OK** button, and move the equation and  $R^2$  text box somewhere you can easily read them.

## Excel: Warnings Against Misuse

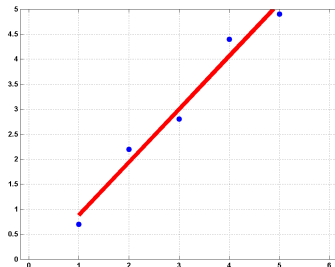
Normally, a  $R^2$  value near 1.0 indicates a very good fit of the data, indicating that you have an accurate interpolating function.

However, performing fits on small numbers of data points may lead to misleadingly high  $R^2$  values. Examine the data above the linear fit graph for an example.

Also, examine the data above the polynomial fit graph for another example of a misleading fit.

# Linear Regression in MATLAB

```
x=1:5;  
y=[0.7 2.2 2.8 4.4 4.9];  
plot(x,y,'bo');  
a=polyfit(x,y,1);  
hold on;  
plot(a(1)*x+a(2),'r');
```



## New MATLAB Material

- `polyfit` is a function that calculates the coefficients of a least-squares polynomial fit function. The 1 in the third argument makes a first-order (or linear) polynomial. The coefficients calculated by `polyfit` are given in order of their term power (constant term first, then linear term, quadratic, etc.)
- `hold on` keeps the existing plot data in place as you plot a new line. Without it, the red linear fit line would erase the blue data points. `hold off` can be used to return the figure window to default behavior.

# Linear and Cubic Interpolation in MATLAB

```
>> x=[2 3 6.5 8 12];  
>> y=[14 20 17 16 23];  
>> xx=[7 7.5];  
>> yy=interp1(x,y,xx,'linear')  
yy =  
    16.6667    16.3333  
>> yy=interp1(x,y,xx,'cubic')  
yy =  
    16.5768    16.1773
```

Notes/corrections to Rao: the function is `interp1` (with a number 1), not `interp1` (with a lowercase L). The  $x$  values to be interpolated at can be one point, or several. The 'cubic' interpolation method uses a different equation now, and the behavior Rao describes is available with the 'v5cubic' method.