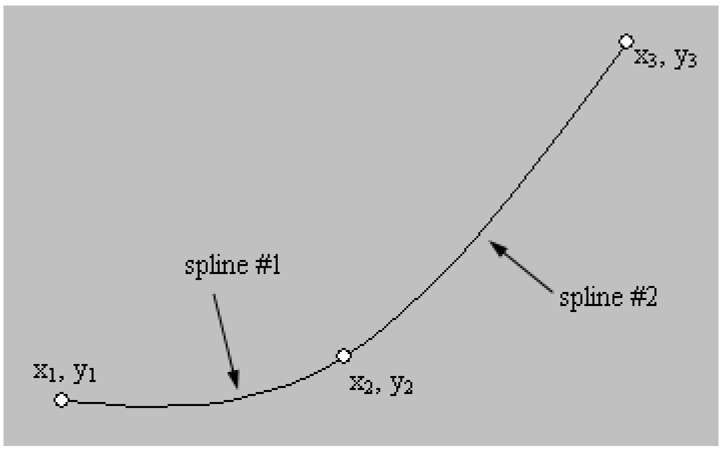
Physics Forums

Cubic Spline Tutorial

Cubic splines are a popular choice for curve fitting for ease of data interpolation, integration, differentiation, and they are normally very smooth. This tutorial will describe a computationally efficient method of constructing joined cubic splines through known data points. Consider the problem of constructing 2 cubic splines to fit 3 data points (x_1,y_1), (x_2,y_2), (x_3,y_3). This is the simplest case of cubic spline interpolation that will illustrate the methods used in more normal cases where several points are present. The key characteristics of cubic spline interpolation are:



1. The curves pass through all specified data points
2. 1st derivative continuity at interior points
3. 2nd derivative continuity at interior points
4. boundary conditions specified at the free ends

We begin with the equations of the two splines:

spline #1

$y = a_1(x-x_1)^3 + b_1(x-x_1)^2 + c_1(x-x_1) + d_1$

$y' = 3a_1(x-x_1)^2 + 2b_1(x-x_1) + c_1$

$y'' = 6a_1(x-x_1) + 2b_1$

spline #2

$y = a_2(x-x_2)^3 + b_2(x-x_2)^2 + c_2(x-x_2) + d_2$

$y' = 3a_2(x-x_2)^2 + 2b_2(x-x_2) + c_2$

$y'' = 6a_2(x-x_2) + 2b_2$

For now, we'll focus on spline #1. We start with the 2nd derivative. Imposing the compatibility constraints that $y'' = y_1''$ at $x = x_1$ and $y'' = y_2''$ at $x = x_2$, and calling $x_2-x_1 = h_1$:

$y_1'' = 6a_1(x_1-x_1) + 2b_1 = 0 + 2b_1 = 2b_1$ $\qquad b_1 = y_1''/2$

$y_2'' = 6a_1(x_2-x_1) + 2b_1 = 6a_1h_1 + y_1''$ $\qquad a_1 = (y_2''-y_1'')/6h_1$

This results in the following equation for the 2nd derivative:

$$y'' = (x-x_1)(y_2''-y_1'')/(x_2-x_1) + y_1''$$

which can be verified to be correct (i.e. $y'' = y_1''$ at $x = x_1$ and $y'' = y_2''$ at $x = x_2$). Next, apply the conditions that the spline pass though the points, in other words $y_1 = f(x_1)$ and $y_2 = f(x_2)$:

$y_1 = 0 + 0 + 0 + d_1$ $\qquad d_1 = y_1$

$y_2 = (x_2-x_1)^3(y_2''-y_1'')/6h_1 + y_1''(x_2-x_1)^2/2 + c_1(x_2-x_1) + y_1$

$y_2 = h_1^3(y_2''-y_1'')/6h_1 + y_1''h_1^2/2 + c_1h_1 + y_1$

$y_2 = h_1^2(y_2''-y_1'')/6 + y_1''h_1^2/2 + c_1h_1 + y_1$

$y_2-y_1 = y_2''h_1^2/6 - y_1''h_1^2/6 + y_1''h_1^2/2 + c_1h_1$

$(y_2-y_1)/h_1 = y_2''h_1/6 - y_1''h_1/6 + y_1''h_1/2 + c_1$

$(y_2-y_1)/h_1 = y_2''h_1/6 - y_1''h_1/6 + 3y_1''h_1/6 + c_1$

$(y_2-y_1)/h_1 = y_2''h_1/6 + y_1''h_1/3 + c_1$ $\qquad c_1 = (y_2-y_1)/h_1 – y_2''h_1/6 - y_1''h_1/3$

Finally, impose the compatibility condition that y_2' in spline 1 must equal y_2' in spline 2:

$3a_1(x_2-x_1)^2 + 2b_1(x_2-x_1) + c_1 = 3a_2(x_2-x_2)^2 + 2b_2(x_2-x_2) + c_2$
$3a_1h_1^2 + 2b_1h_1 + c_1 = c_2$
$h_1(y_2''-y_1'')/2 + y_1''h_1 + (y_2-y_1)/h_1 - y_2''h_1/6 - y_1''h_1/3 = (y_3-y_2)/h_2 - y_3''h_2/6 - y_2''h_2/3$
$h_1(y_2''-y_1'')/2 + y_1''h_1 - y_2''h_1/6 - y_1''h_1/3 + y_3''h_2/6 + y_2''h_2/3 = (y_3-y_2)/h_2 - (y_2-y_1)/h_1$
$3h_1(y_2''-y_1'') + 6y_1''h_1 - y_2''h_1 - 2y_1''h_1 + y_3''h_2 + 2y_2''h_2 = 6(y_3-y_2)/h_2 - 6(y_2-y_1)/h_1$
$3h_1y_2'' - 3h_1y_1'' + 6y_1''h_1 - y_2''h_1 - 2y_1''h_1 + y_3''h_2 + 2y_2''h_2 = 6(y_3-y_2)/h_2 - 6(y_2-y_1)/h_1$
$y_1''(6h_1 - 3h_1 - 2h_1) + y_2''(2h_1 + 2h_2) + y_3''h_2 = 6(y_3-y_2)/h_2 - 6(y_2-y_1)/h_1$

$h_1y_1'' + 2(h_1 + h_2)y_2'' + h_2y_3'' = 6[(y_3-y_2)/h_2 - (y_2-y_1)/h_1]$ (1) governing equation for cubic splines

Generalizing, this equation results in a tri-diagonal set of linear equations (Ax = b), where x represents the unknowns (2nd derivatives of the points), and b is the right hand side. Tri-diagonal sets of linear equations are efficiently solved with specialized algorithms.

$$\begin{bmatrix} ? & & & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & & \\ & h_2 & 2(h_2+h_3) & h_3 & \ddots & & \\ & & & \ddots & & & \\ & & & & \ddots & h_{n-2} & \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ & & & & & & ? \end{bmatrix} \begin{bmatrix} y_1'' \\ y_2'' \\ \vdots \\ \vdots \\ \vdots \\ y_{n-1}'' \\ y_n'' \end{bmatrix} = 6 \begin{bmatrix} ? \\ (y_3-y_2)/h_2 - (y_2-y_1)/h_1 \\ \vdots \\ \vdots \\ \vdots \\ (y_n-y_{n-1})/h_{n-1} - (y_{n-1}-y_{n-2})/h_{n-2} \\ ? \end{bmatrix}$$

If equal point spacing is used (i.e. $h_1 = h_2 = \ldots h_{n-1} = h$), even more simplification can be made:

$$\begin{bmatrix} ? & & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & \ddots & & \\ & & & \ddots & & & \\ & & & & \ddots & 1 & \\ & & & & 1 & 4 & 1 \\ & & & & & & ? \end{bmatrix} \begin{bmatrix} y_1'' \\ y_2'' \\ \vdots \\ \vdots \\ \vdots \\ y_{n-1}'' \\ y_n'' \end{bmatrix} = 6/(h*h) \begin{bmatrix} ? \\ y_3 - 2y_2 + y_1 \\ y_4 - 2y_3 + y_2 \\ \vdots \\ \vdots \\ y_n - 2y_{n-1} + y_{n-2} \\ ? \end{bmatrix}$$

The first and last equations represent the boundary conditions of the free ends of the spline that must be chosen. Often, so called ‘natural’ boundary conditions are used, where the 2nd derivative is set to zero. Natural boundary conditions result in total minimum curvature. Other boundary conditions can be used. For example:

1. Natural boundary conditions	$y_1'' = 0$	$y_n'' = 0$
2. Parabolic runout	$y_1'' = y_2''$	$y_{n-1}'' = y_n''$
3. Zero slope		
4. Specified 1st derivative		
5. Specified 2nd derivative	$s_1 = y_1''$	$s_n = y_n''$

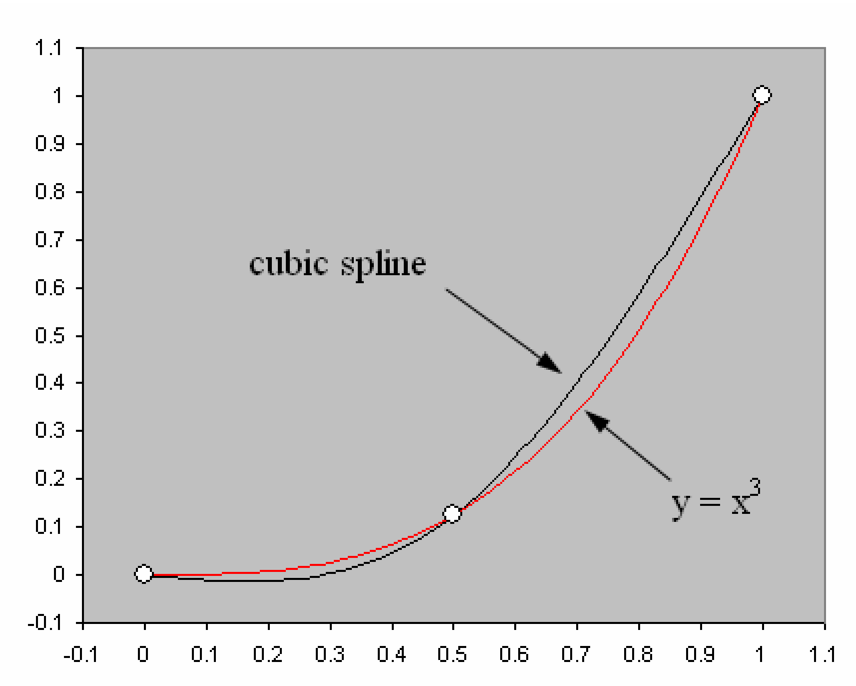
Example Problem #1:

Let's illustrate with a specific problem: fit 2 cubic splines to the function $y = x^3$ in the range of x = 0 to 1. Thus, $x_1 = 0$, $y_1 = 0$, $x_3 = 1$, $y_3 = 1$. We'll pick $x_2 = 0.5$ (thus $y_2 = 0.125$) and use natural boundary conditions. Because the only unknowns are the 2nd derivative at each point, we have a 3 x 3 matrix to solve. Also, since $(x_2\text{-}x_1) = h_1 = (x_3\text{-}x_2) = h_2 = 0.5$, we can used the simplified version (Note: . means zero):

$$\begin{bmatrix} 1 & . & . \\ 1 & 4 & 1 \\ . & . & 1 \end{bmatrix} \begin{bmatrix} y_1'' \\ y_2'' \\ y_3'' \end{bmatrix} = 6/(0.25)\begin{bmatrix} 0 \\ y_3 - 2y_2 + y_1 \\ 0 \end{bmatrix} = 24\begin{bmatrix} 0 \\ 1-2(0.125)+0 \\ 0 \end{bmatrix} = 24\begin{bmatrix} 0 \\ 0.75 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 18 \\ 0 \end{bmatrix}$$

The solution is $[y_1'', y_2'', y_3'']^T = [0, 4.5, 0]^T$. Note: in this case the solution is trivial, $y_2'' = 18/4$. From this we can calculate the coefficients of the cubic spline segments:

$a_1 = (y_2''-y_1'')/6h_1$	$b_1 = y_1''/2$	$c_1 = (y_2-y_1)/h_1 - y_2''h_1/6 - y_1''h_1/3$	$d_1 = y_1$
$a_2 = (y_3''-y_2'')/6h_2$	$b_2 = y_2''/2$	$c_2 = (y_3-y_2)/h_2 - y_3''h_2/6 - y_2''h_2/3$	$d_2 = y_2$
$a_1 = 1.5$	$b_1 = 0$	$c_1 = -0.125$	$d_1 = 0$
$a_2 = -1.5$	$b_2 = 2.25$	$c_2 = 0$	$d_2 = 0.125$



As can be seen in the plot, the cubic spline interpolation doesn't fit the function very well. Wait a minute. How can 2 cubic splines not fit a cubic polynomial very well? It should be a perfect fit, especially since it only takes 1 cubic spline to represent the cubic polynomial function $y = x^3$. The answer is that we forced the 2nd derivative of the spline to be zero at each free end. This works fine at x = 0 for the function $y = x^3$ because the 2nd derivative of this function is indeed 0 at x = 0. However, it isn't a good choice at x = 1 because the 2nd derivative of $y = x^3$ at x = 1 is 6x = 6. If the boundary condition of x_3 is changed to reflect a value of 6 instead of 0, the fit is perfect. This illustrates the importance of choosing appropriate boundary conditions for the problem at hand.

Example Problem #2:

As a final illustration, we will show how to enforce a slope at either end. Recall the equation of the 1st derivative:

At $x = x_1$:

$y_1' = 3a_1(x_1\text{-}x_1)^2 + 2b_1(x_1\text{-}x_1) + c_1 = 0 + 0 + c_1 = c_1 = (y_2\text{-}y_1)/h_1 - y_2''h_1/6 - y_1''h_1/3$

$(2h_1)y_1'' + (h_1)y_2'' = 6[(y_2\text{-}y_1)/h_1 - y_1']$

At $x = x_3$:

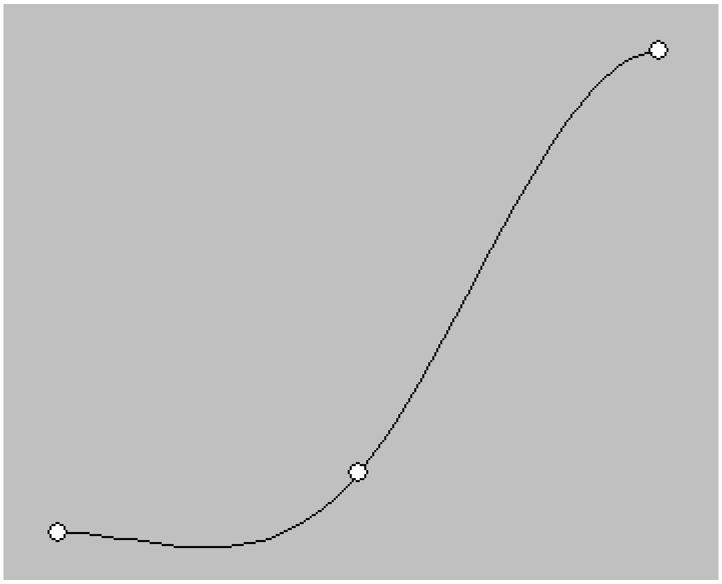
$y_3' = 3a_2(x_3-x_2)^2 + 2b_2(x_3-x_2) + c_2 = 3a_2h_2^2 + 2b_2h_2 + c_2$
$y_3' = 3h_2^2(y_3''-y_2'')/6h_2 + 2h_2y_2''/2 + (y_3-y_2)/h_2 - y_3''h_2/6 - y_2''h_2/3$
$y_3' = 3h_2(y_3''-y_2'')/6 + h_2y_2'' + (y_3-y_2)/h_2 - y_3''h_2/6 - y_2''h_2/3$
$h_2y_3''/2 - h_2y_2''/2 + h_2y_2'' - y_3''h_2/6 - y_2''h_2/3 = y_3' - (y_3-y_2)/h_2$
$3h_2y_3'' - 3h_2y_2'' + 6h_2y_2'' - y_3''h_2 - 2y_2''h_2 = 6(y_3' - (y_3-y_2)/h_2)$
$y_3''(3h_2 - h_2) - y_2''(6h_2 - 3h_2 - 2h_2) = 6(y_3' - (y_3-y_2)/h_2)$
$(h_2)y_2'' + (2h_2)y_3'' = 6[y_3' - (y_3-y_2)/h_2]$

If we were to force the slope to be zero at both ends, the matrix equation would be:

$$\begin{bmatrix} 2h_1 & h_1 & . \\ h_1 & 2(h_1+h_2) & h_2 \\ . & h_2 & 2h_2 \end{bmatrix} \begin{bmatrix} y_1'' \\ y_2'' \\ y_3'' \end{bmatrix} = 6\begin{bmatrix} (y_2-y_1)/h_1 - 0 \\ (y_3-y_2)/h_2 - (y_2-y_1)/h_1 \\ 0-(y_3-y_2)/h_2 \end{bmatrix}$$

Using the simplified version, we have:

$$\begin{bmatrix} 2 & 1 & . \\ 1 & 4 & 1 \\ . & 1 & 2 \end{bmatrix} \begin{bmatrix} y_1'' \\ y_2'' \\ y_3'' \end{bmatrix} = 6/(0.25)\begin{bmatrix} y_2-y_1 \\ y_3-2y_2+y_1 \\ -(y_3-y_2) \end{bmatrix} = 24\begin{bmatrix} 0.125-0 \\ 1-2(0.125)+0 \\ -(1-0.125) \end{bmatrix} = 24\begin{bmatrix} 0.125 \\ 0.75 \\ -0.875 \end{bmatrix} = \begin{bmatrix} 3 \\ 18 \\ -21 \end{bmatrix}$$



The solution is $[y_1'', y_2'', y_3'']^T = [-3, 9, -15]^T$, from which we can calculate the spline segment coefficients and plot the result.

Conclusion:

We have demonstrated a method of formulating cubic splines to interpolate a given set of points and shown how to implement various free end boundary conditions.

Discussion:

The formulation described here is by no means the only one - there are other formulations of cubic splines. One possibility is to set up the matrix equations to directly calculate the spline segment coefficients, but it requires a matrix of dimension 4*(n-1), which is much more computationally intensive than the method shown here. If it is desired to not choose the free end boundary conditions, the splines on either end can be fit to the 3 points instead of 2, or the method illustrated here can be used with the boundary conditions determined by fitting splines to 4 points on either end. These are just a few of the possible techniques for cubic spline interpolation.