

Section 5.6 Cubic Spline Interpolation

It is assumed that you have read the file **splinedefs.doc**.

We have seen that the Lagrange form of the interpolating polynomial to a set of distinct points

$S = \{(x_i, f_i) \mid i = 0, 1, \dots, n\}$ is given by $P_n(x) = \sum_{j=0}^n L_j(x) f_j$ where the (Lagrange) basis functions

$$L_j(x) \equiv \prod_{\substack{i=0 \\ i \neq j}}^n \left(\frac{x - x_i}{x_j - x_i} \right) \text{ and satisfies } L_j(x_k) = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases}. \text{ Unfortunately the Lagrange basis functions are}$$

cumbersome to use and adjoining or deleting a point from the data set requires recalculation of all basis functions. If we use the Divided Difference basis functions

$$\left\{ 1, (x - x_0), (x - x_0)(x - x_1), (x - x_0)(x - x_1)(x - x_2), \dots, \prod_{j=0}^{n-1} (x - x_j) \right\}$$

we gain some flexibility in adjoining or deleting a point from the data set. These basis functions are somewhat simpler than the Lagrange basis functions, but in either case there is a major draw back.

The Lagrange and divided difference basis functions are “**global**” in the sense that they are defined over the interval $[\min\{x_i\}, \max\{x_i\}]$; in fact, they are defined over the entire real line. In addition if the data set is large, the polynomial wiggle problem may become an issue.

We have seen that an alternative to generating a single polynomial through the data set S is to use piecewise polynomial approximations. Such piecewise interpolation models are continuous, but not differentiable at all points of the set S . Another approach is to use Hermite data; that is, ordered triples of x -coordinates, y -coordinates, and the slope at the (x, y) -pair. This approach delays the on set of the Runge phenomena, but does not eliminate it.

An approach that will generate a piecewise polynomial with more than one continuous derivative is called **spline interpolation.**

Goal

Show how to construct cubic splines. These are piecewise cubic polynomials between successive data points that have **TWO continuous derivatives** at every point in $[\min\{x_i\}, \max\{x_i\}]$.

Requirement: the data set $S = \{(x_i, f_i) \mid i = 0, 1, \dots, n\}$ is such that $x_0 < x_1 < x_2 < \dots < x_n$. (That is, the data must be ordered.)

We will construct a cubic polynomial over each interval $[x_i, x_{i+1}]$ so that the cubic pieces over successive intervals are continuous at the shared point and have a first and second derivative that is continuous at the shared point.

Example

Let $S = \{(1,2), (3,1), (4,0)\}$. We want to construct a cubic polynomial on $[1, 3]$ that interpolates $(1,2)$ and $(3,1)$ and a second cubic polynomial on $[3, 4]$ that interpolates $(3,1)$ and $(4,0)$. At the shared point $(3,1)$ the two cubics must have the same derivative value and the same second derivative value.

Cubic polynomial on $[1, 3]$: $S_1(x) = d_1(x-1)^3 + c_1(x-1)^2 + b_1(x-1) + a_1$

Cubic polynomial on $[3, 4]$: $S_2(x) = d_2(x-3)^3 + c_2(x-3)^2 + b_2(x-3) + a_2$

<<Note there are **8 coefficients** to be determined.>>

Interpolation Conditions:

$$S_1(1) = 2 \rightarrow a_1 = 2$$

$$S_1(3) = 1 \rightarrow 8d_1 + 4c_1 + 2b_1 + a_1 = 1$$

$$S_2(3) = 1 \rightarrow a_2 = 1$$

$$S_2(4) = 0 \rightarrow d_2 + c_2 + b_2 + a_2 = 0$$

First Derivative Conditions at the shared point $(3,1)$:

$$S'_1(x) = 3d_1(x-1)^2 + 2c_1(x-1) + b_1$$

$$S'_2(x) = 3d_2(x-3)^2 + 2c_2(x-3) + b_2$$

$$\text{then } S'_1(3) = S'_2(3) \rightarrow 12d_1 + 4c_1 + b_1 = b_2$$

Second Derivative Conditions at the shared point $(3,1)$:

$$S''_1(x) = 6d_1(x-1) + 2c_1$$

$$S''_2(x) = 6d_2(x-3) + 2c_2$$

$$\text{then } S''_1(3) = S''_2(3) \rightarrow 12d_1 + 2c_1 = 2c_2$$

Counting the number of equations to be satisfied we get SIX. But we have EIGHT coefficients. Hence there are TWO free parameters. So there are infinitely many cubic pieces that could be used to obtain a "cubic spline interpolant" to data set S .

Formal Definition of a Cubic Spline Interpolant

Let f be a function defined on the interval $[a, b]$, and let

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

be the $n + 1$ distinct points at which f is to be interpolated. Recall that the x_i divide $[a, b]$ into n subintervals, referred to as a partition of $[a, b]$.

A CUBIC SPLINE INTERPOLANT of f relative to the partition

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

is a function s which satisfies:

(1) on each subinterval $[x_j, x_{j+1}]$, $j = 0, 1, 2, \dots, n - 1$, s coincides with the cubic polynomial

$$s(x) = s_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3;$$

(2) s interpolates f at $x_0, x_1, x_2, \dots, x_n$;

(3) s is continuous on $[a, b]$;

(4) s' is continuous on $[a, b]$;

(5) s'' is continuous on $[a, b]$.

The interpolatory cubic spline consists of n cubic pieces, each with four coefficients, so **there are a total of $4n$ unknowns**.

Interpolation at x_0 through x_n provides **$n + 1$ equations**.

Continuity of the spline and its first two derivatives at the $n - 1$ interior knots contribute **$3(n - 1)$ equations**.

Thus we have a total of $4n - 2$ equations to determine $4n$ unknowns. To determine the spline **we need two more equations to be specified** in a way that gives us a nonsingular linear system to solve for the spline coefficients.

We discuss three types of additional constraints which are called **boundary conditions**.

- For a natural cubic spline we have boundary conditions **$S''(x_0) = 0$ and $S''(x_n) = 0$** .
- For a clamped cubic spline we have boundary conditions **$S'(x_0) = \alpha$ and $S'(x_n) = \beta$** where α and β are specified values.
- For a Not-A-Knot cubic spline we have boundary conditions **S''' be continuous at $x = x_1$ and $x = x_{n-1}$** .

Fortunately in each of these cases the resulting linear systems of linear equations can be solved efficiently. Employing some algebra we can show that in each case the coefficient matrix of the system of equations is tridiagonal and strictly diagonally dominant. Because of the clever algebra needed to get the coefficient matrix to have these properties the linear system is constructed in such a way that **we solve for the c 's in the cubic expressions**

$a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$ and then solve for the b's and d's. It happens that the a's are just the function values from the data.

The other issue is to evaluate the spline S at various points in order to interpolate or graph the spline. Here there is a two step process; first determine which cubic piece to use and then evaluate the corresponding cubic polynomial.

YOU DO NOT WANT TO DO NOT THIS TYPE OF COMPUTATION BY HAND!

For this course we have three commands that incorporate the calculation (and display of the coefficients), evaluation of the spline, and graphing the spline.

nspline The natural cubic spline to the data in vectors x and y is computed. The x-values must be in increasing order. The output is a table of coefficients for the piecewise cubic polynomials. Sketching and evaluation options are available.

Use in the form `====> nspline(x,y) <====`

cspline The clamped cubic spline to the data in vectors x and y is computed. The x-values must be in increasing order. The output is a table of coefficients for the piecewise cubic polynomials. Sketching and evaluation options are available.

Use in the form `====> cspline(x,y) <====`

nakspline The Not-A-Knot cubic spline to the data in vectors x and y is computed. The x-values must be in increasing order. The output is a table of coefficients for the piecewise cubic polynomials. Sketching and evaluation options are available.

Use in the form `====> nakspline(x,y) <====`

Example: Take the data set $\{(1,2), (3,1), (4,0), (7,3)\}$ and compute each of the three types of splines.

The natural spline:

The coefficients of the cubic pieces

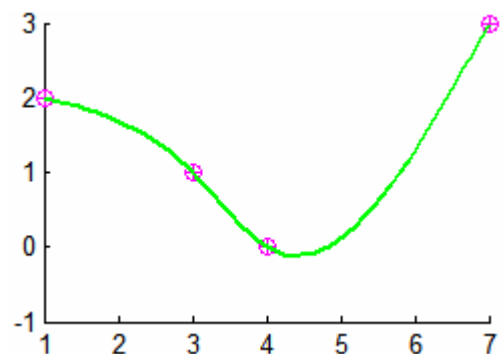
$D*(x-x(i))^3 + C*(x-x(i))^2 + B*(x-x(i)) + A$ are:

i	x	D	C	B	A
1	1	-3/47	0	-23/94	2
2	3	37/94	-18/47	-95/94	1
3	4	-25/282	75/94	-28/47	0

$$S_1(x) = \frac{-3}{47}(x-1)^3 + 0(x-1)^2 + \frac{-23}{94}(x-1) + 2$$

$$S_2(x) = \frac{37}{94}(x-3)^3 + \frac{-18}{47}(x-3)^2 + \frac{-95}{94}(x-3) + 1$$

$$S_3(x) = \frac{-25}{282}(x-4)^3 + \frac{75}{94}(x-4)^2 + \frac{-28}{47}(x-4)$$



The clamped spline with the slope at (1, 2) is 3 and the slope at (4,0) is -2.

The coefficients of the cubic pieces

$D*(x-x(i))^3 + C*(x-x(i))^2 + B*(x-x(i)) + A$ are:

i	x	D	C	B	A
1	1	23/42	-239/84	3	2
2	3	31/84	37/84	-38/21	1
3	4	-107/252	65/42	5/28	0

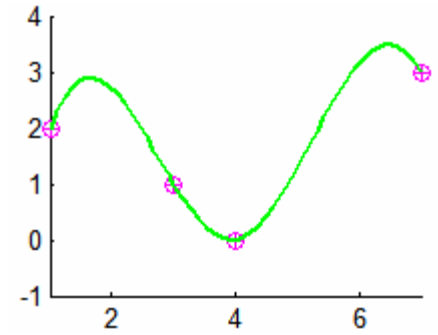
Initial slope = 3

Final slope = -2

$$S_1(x) = \frac{23}{42}(x-1)^3 + \frac{-239}{84}(x-1)^2 + 3(x-1) + 2$$

$$S_2(x) = \frac{31}{84}(x-3)^3 + \frac{37}{84}(x-3)^2 + \frac{-38}{21}(x-3) + 1$$

$$S_3(x) = \frac{-107}{252}(x-4)^3 + \frac{65}{42}(x-4)^2 + \frac{5}{28}(x-4)$$



The Not_A-Knot spline.

The coefficients of the cubic pieces

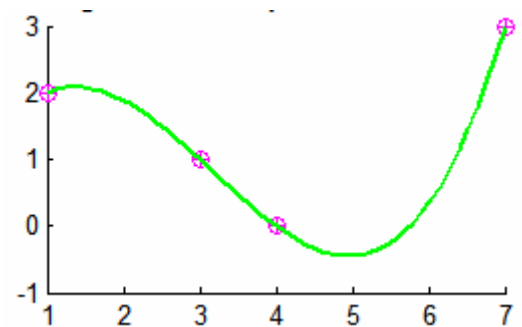
$D*(x-x(i))^3 + C*(x-x(i))^2 + B*(x-x(i)) + A$ are:

i	x	D	C	B	A
1	1	1/9	-13/18	1/2	2
2	3	1/9	-1/18	-19/18	1
3	4	1/9	5/18	-5/6	0

$$S_1(x) = \frac{1}{9}(x-1)^3 + \frac{-13}{18}(x-1)^2 + \frac{1}{2}(x-1) + 2$$

$$S_2(x) = \frac{1}{9}(x-3)^3 + \frac{-1}{18}(x-3)^2 + \frac{-19}{18}(x-3) + 1$$

$$S_3(x) = \frac{-1}{9}(x-4)^3 + \frac{5}{18}(x-4)^2 + \frac{-5}{6}(x-4)$$



A property of clamped splines.

The clamped cubic spline satisfies an interesting property related to the curvature of a function. For a general function, the curvature at a point is defined by

$$\kappa(x) = \frac{|f''(x)|}{(1 + [f'(x)]^2)^{3/2}},$$

which is commonly linearized to $\kappa(x) \approx |f''(x)|$. The quantity $\int_a^b [f''(x)]^2 dx$ can therefore be viewed as a crude measure of the total curvature over an interval.

We will now prove that, in this measure, any smooth interpolating function which satisfies clamped boundary conditions must have a total curvature at least as large as that of the clamped cubic spline. This is sometimes referred to as the minimum curvature property of the clamped cubic spline.

Error in Cubic Spline Interpolation:

We conclude the discussion of cubic spline interpolation with a theorem on the error associated with the clamped cubic spline. For a proof of this result, see de Boor [2], Hall and Meyer [3] or Schultz [4]. An error bound for the not-a-knot cubic spline, also of fourth-order, can be found in de Boor [5] or Beatson [6].

Theorem:

Let f be continuous, with four continuous derivatives, on the interval $[a, b]$, and let s be the clamped cubic spline interpolant of f relative to the partition $a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$.

Then

$$\max_{x \in [a, b]} |f(x) - s(x)| \leq \frac{5}{384} h^4 \max_{x \in [a, b]} |f^{(4)}(x)|,$$

where $h = \max_{0 \leq i \leq n-1} (x_{i+1} - x_i)$.

Other results:

The error theory for spline approximation is more difficult than that for ordinary polynomial interpolation and that for piecewise polynomial interpolation. The details are more intricate due to the complexity of the governing equations.

Theorem

If $f(x)$ is in $C^4[a, b]$ and $S(x)$ is a cubic spline that interpolates the data set $\{(x_i, f(x_i)) \mid i = 0, 1, 2, \dots, n\}$ where $a = x_0 < x_1 < \dots < x_n = b$ with $\max_{i=0,1,\dots,n-1} \{(x_{i+1} - x_i)\} \leq h$, then

$$\max_{x \in [a, b]} |f(x) - S(x)| \leq \frac{5}{384} h^4 \max_{x \in [a, b]} |f^{(4)}(x)|.$$

In addition there exist constant C_k , $1 \leq k \leq 3$, such that

$$\max_{x \in [a, b]} |f^{(k)}(x) - S^{(k)}(x)| \leq C_k h^{4-k} \max_{x \in [a, b]} |f^{(4)}(x)|.$$

Comment: Cubic spline interpolation is no more accurate, in terms of the exponent on h , than ordinary piecewise polynomial interpolation, but the constant on the right of the inequality is smaller.

The advantage of cubic spline interpolation lies in the smoothness of the approximation; that is, cubic spline interpolants are twice continuously differentiable.

Example Consider the 'serpentine curve' $f(x) = \frac{x}{0.25 + x^2}$ over $[-2, 2]$.

Let $x = [-2, -1, -0.5, -0.25, 0, 0.25, 0.5, 1, 2]$ be the x-coordinates of sample points. Enter this vector of data in to MATLAB. **Compute the corresponding y-coordinates** in MATLAB using command $y = x./(0.25+x.^2)$; Record the data set.

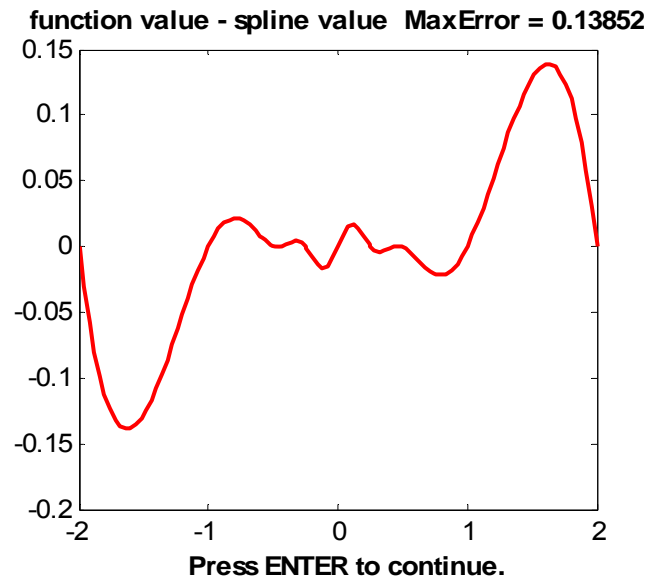
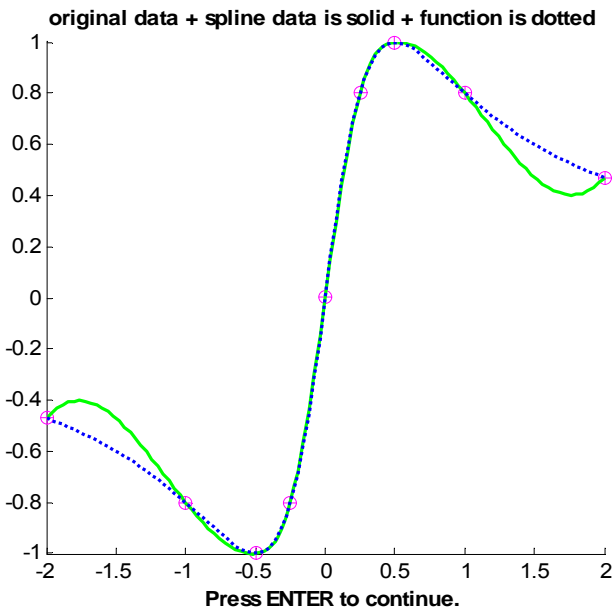
Use **nakspline** command in MATLAB to generate a cubic spline $s(x)$ approximation to $f(x)$. Display the coefficients of the piecewise cubic polynomials, generate a graph of the spline model with $f(x)$ superimposed, display a graph of the error, and approximate $\|f - s\|_\infty$. Use the spline model $s(x)$ to determine x so that $s(x) = 0.9$.

<<< Not-A-Knot Cubic Splines with Plotting >>>

The coefficients of the cubic pieces

$D*(x-x(i))^3 + C*(x-x(i))^2 + B*(x-x(i)) + A$ are:

	i	x	D	C	B	A
ans =	1	-2	0.60518	-1.56	0.62541	-0.47059
	2	-1	0.60518	0.25553	-0.67906	-0.8
	3	-0.5	7.6612	1.1633	0.030353	-1
	4	-0.25	-9.2122	6.9092	2.0485	-0.8
	5	0	-9.2122	-2.2204e-016	3.7758	0
	6	0.25	7.6612	-6.9092	2.0485	0.8
	7	0.5	0.60518	-1.1633	0.030353	1
	8	1	0.60518	-0.25553	-0.67906	0.8



Use the spline model $s(x)$ to determine x so that $s(x) = 0.9$. From the graph we see that x will be between 0.5 and 1 so we need to solve $0.9 = 0.60518(x-0.5)^3 - 1.1633(x-0.5)^2 + 0.030353(x-0.5) + 1$

BISECTION Method:

$f(x) = -0.9 + 0.60518(x-0.5)^3 - 1.1633(x-0.5)^2 + 0.030353(x-0.5) + 1$ on interval $[0.5, 1]$ with tol = 0.001

Bisect will use 10 steps to complete the approximation.

Left EndPoint	Right EndPoint	Midpoints	Function Values
0.5000000000000000	1.0000000000000000	0.7500000000000000	0.044337937500000
0.7500000000000000	1.0000000000000000	0.8750000000000000	-0.020292898437500
0.7500000000000000	0.8750000000000000	0.8125000000000000	0.014350424804687
0.8125000000000000	0.8750000000000000	0.8437500000000000	-0.002444666381836
0.8125000000000000	0.8437500000000000	0.8281250000000000	0.006091447555542
0.8281250000000000	0.8437500000000000	0.8359375000000000	0.001857166955948
0.8359375000000000	0.8437500000000000	0.8398437500000000	-0.000285413835287
0.8359375000000000	0.8398437500000000	0.8378906250000000	0.000787974056572
0.8378906250000000	0.8398437500000000	0.8388671875000000	0.000251802793849
0.8388671875000000	0.8398437500000000	0.8393554687500000	-0.000016675061274

All data displayed. Press ENTER to continue.

Press enter for more.

The interval of length \leq tolerance is

8.388671875000000e-001 8.398437500000000e-001

Its midpoint is the approximate root:

r =

8.393554687500000e-001

The value of f at the approximate root is:

fm =

-1.667506127400742e-005

BISECTION METHOD is over.

