9 Approximation by Spline Functions

By experimentation in a wind tunnel, an airfoil is constructed by trial and error so that it has certain desired characteristics. The cross section of the airfoil is then drawn as a curve on coordinate paper (see Figure 9.1). To study this airfoil by analytical methods or to manufacture it, it is essential to have a formula for this curve. To arrive at such a formula, one first obtains the coordinates of a finite set of points on the curve. Then a smooth curve called a *cubic interpolating spline* can be constructed to match these data points. This chapter discusses general polynomial spline functions and how they can be used in various numerical problems such as the data-fitting problem just described.

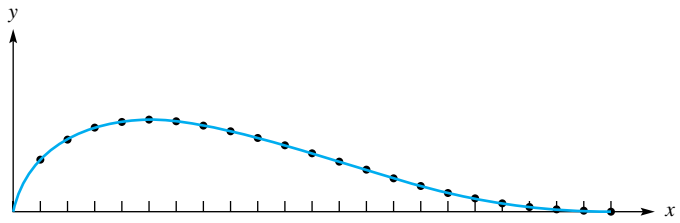

FIGURE 9.1
Airfoil cross section

9.1 First-Degree and Second-Degree Splines

The history of spline functions is rooted in the work of draftsmen, who often needed to draw a gently turning curve between points on a drawing. This process is called *fairing* and can be accomplished with a number of ad hoc devices, such as the *French curve*, made of plastic and presenting a number of curves of different curvature for the draftsman to select. Long strips of wood were also used, being made to pass through the *control points* by weights laid on the draftsman's table and attached to the strips. The weights were called *ducks* and the strips of wood were called *splines*, even as early as 1891. The elastic nature of the wooden strips allowed them to bend only a little while still passing through the prescribed points. The wood was, in effect, solving a differential equation and minimizing the strain energy. The latter is known to be a simple function of the curvature. The mathematical theory of these curves owes much to the early investigators, particularly Isaac Schoenberg in the 1940s and 1950s. Other important names associated with the early development of the subject (i.e., prior to 1964) are Garrett Birkhoff, C. de Boor, J. H. Ahlberg, E. N. Nilson,

H. Garabedian, R. S. Johnson, F. Landis, A. Whitney, J. L. Walsh, and J. C. Holladay. The first book giving a systematic exposition of spline theory was the book by Ahlberg, Nilson, and Walsh [1967].

First-Degree Spline

A **spline function** is a function that consists of polynomial pieces joined together with certain smoothness conditions. A simple example is the **polygonal** function (or spline of degree 1), whose pieces are linear polynomials joined together to achieve continuity, as in Figure 9.2. The points $t_0, t_1, \ldots, t_n$ at which the function changes its character are termed **knots** in the theory of splines. Thus, the spline function shown in Figure 9.2 has eight knots.

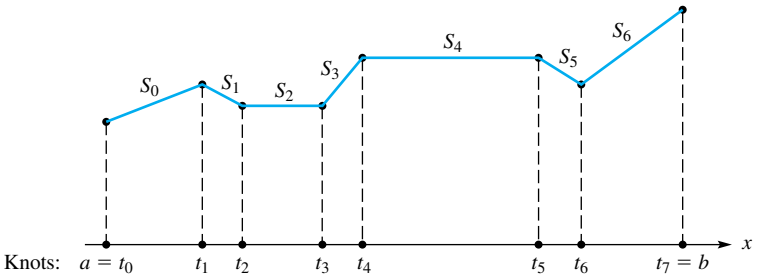

FIGURE 9.2 First-degree spline function

Such a function appears somewhat complicated when defined in explicit terms. We are forced to write

$$S(x) = \begin{cases} S_0(x) & x \in [t_0, t_1] \\ S_1(x) & x \in [t_1, t_2] \\ \vdots & \vdots \\ S_{n-1}(x) & x \in [t_{n-1}, t_n] \end{cases} \tag{1}$$

where

$$S_i(x) = a_i x + b_i \tag{2}$$

because each piece of $S(x)$ is a linear polynomial. Such a function $S(x)$ is **piecewise linear**. If the knots $t_0, t_1, \ldots, t_n$ were given and if the coefficients $a_0, b_0, a_1, b_1, \ldots, a_{n-1}, b_{n-1}$ were all known, then the evaluation of $S(x)$ at a specific x would proceed by first determining the interval that contains x and then using the appropriate linear function for that interval.

If the function S defined by Equation (1) is continuous, we call it a **first-degree spline**. It is characterized by the following three properties.

■ **DEFINITION 1**

SPLINE OF DEGREE 1

A function S is called a **spline of degree 1** if:

1. The domain of S is an interval $[a, b]$.
2. S is continuous on $[a, b]$.
3. There is a partitioning of the interval $a = t_0 < t_1 < \cdots < t_n = b$ such that S is a linear polynomial on each subinterval $[t_i, t_{i+1}]$.

Outside the interval $[a, b]$, $S(x)$ is usually defined to be the same function on the left of a as it is on the leftmost subinterval $[t_0, t_1]$ and the same on the right of b as it is on the rightmost subinterval $[t_{n-1}, t_n]$, namely, $S(x) = S_0(x)$ when $x < a$ and $S(x) = S_{n-1}(x)$ when $x > b$.

Continuity of a function f at a point s can be defined by the condition

$$\lim_{x \to s^+} f(x) = \lim_{x \to s^-} f(x) = f(s)$$

Here, $\lim_{x \to s^+}$ means that the limit is taken over x values that converge to s from above s; that is, $(x - s)$ is positive for all x values. Similarly, $\lim_{x \to s^-}$ means that the x values converge to s from below.

EXAMPLE 1 Determine whether this function is a first-degree spline function:

$$S(x) = \begin{cases} x & x \in [-1, 0] \\ 1 - x & x \in (0, 1) \\ 2x - 2 & x \in [1, 2] \end{cases}$$

Solution The function is obviously piecewise linear but is not a spline of degree 1 because it is discontinuous at $x = 0$. Notice that $\lim_{x \to 0^+} S(x) = \lim_{x \to 0}(1 - x) = 1$, whereas $\lim_{x \to 0^-} S(x) = \lim_{x \to 0} x = 0$. ■

The spline functions of degree 1 can be used for interpolation. Suppose the following table of function values is given:

x	t_0	t_1	$\cdots$	t_n
y	y_0	y_1	$\cdots$	y_n

There is no loss of generality in supposing that $t_0 < t_1 < \cdots < t_n$ because this is only a matter of labeling the knots.

The table can be represented by a set of $n + 1$ points in the plane, (t_0, y_0), (t_1, y_1), $\ldots$, (t_n, y_n), and these points have distinct abscissas. Therefore, we can draw a polygonal line through the points without ever drawing a *vertical* segment. This polygonal line is the graph of a function, and this function is obviously a spline of degree 1. What are the equations of the individual line segments that make up this graph?

By referring to Figure 9.3 and using the point-slope form of a line, we obtain

$$S_i(x) = y_i + m_i(x - t_i) \tag{3}$$

on the interval $[t_i, t_{i+1}]$, where m_i is the slope of the line and is therefore given by the formula

$$m_i = \frac{y_{i+1} - y_i}{t_{i+1} - t_i}$$

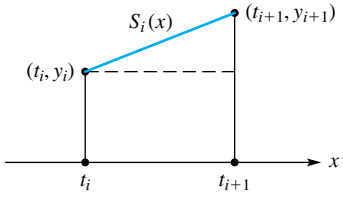

FIGURE 9.3 First-degree spline: linear $S_i(x)$

Notice that the function S that we are creating has $2n$ parameters in it: the n coefficients a_i and the n constants b_i in Equation (2). On the other hand, exactly $2n$ conditions are being imposed, since each constituent function S_i must interpolate the data at the ends of its subinterval. Thus, the number of parameters equals the number of conditions. For the higher-degree splines, we shall encounter a mismatch in these two numbers; the spline of degree k will have $k - 1$ free parameters for us to use as we wish in the problem of interpolating at the knots.

The form of Equation (3) is better than that of Equation (2) for the practical evaluation of $S(x)$ because some of the quantities $x - t_i$ must be computed in any case simply to determine which subinterval contains x. If $t_0 \leqq x \leqq t_n$ then the interval $[t_i, t_{i+1}]$ containing x is characterized by the fact that $x - t_i$ is the first of the quantities $x - t_{n-1}, x - t_{n-2}, \ldots, x - t_0$ that is *nonnegative*.

The following is a function procedure that utilizes $n + 1$ table values (t_i, y_i) in linear arrays (t_i) and (y_i), assuming that $a = t_0 < t_1 < \cdots < t_n = b$. Given an x value, the routine returns $S(x)$ using Equations (1) and (3). If $x < t_0$, then $S(x) = y_0 + m_0(x - t_0)$; if $x > t_n$, then $S(x) = y_{n-1} + m_{n-1}(x - t_{n-1})$.

```
real function Spline1(n, (t_i), (y_i), x)
integer i, n;   real x;   real array (t_i)_{0:n}, (y_i)_{0:n}
for i = n − 1 to 0 step −1 do
    if x − t_i ≧ 0 then exit loop
end for
Spline1 ← y_i + (x − t_i)[(y_{i+1} − y_i)/(t_{i+1} − t_i)]
end function Spline1
```

Modulus of Continuity

To assess the **goodness of fit** when we interpolate a function with a first-degree spline, it is useful to have something called the *modulus of continuity* of a function f. Suppose f is defined on an interval $[a, b]$. The **modulus of continuity** of f is

$$\omega(f; h) = \sup\{|f(u) - f(v)|: a \leqq u \leqq v \leqq b, |u - v| \leqq h\}$$

Here, sup is the **supremum**, which is the least upper bound of the given set of real numbers. The quantity $\omega(f; h)$ measures how much f can change over a small interval of width h. If f is continuous on $[a, b]$, then it is uniformly continuous, and $\omega(f; h)$ will tend to zero as h tends to zero. If f is not continuous, $\omega(f; h)$ will not tend to zero. If f is differentiable on (a, b) (in addition to being continuous on $[a, b]$) and if $f'(x)$ is bounded on (a, b), then the Mean Value Theorem can be used to get an estimate of the modulus of continuity: If u and v are as described in the definition of $\omega(f; h)$, then

$$|f(u) - f(v)| = |f'(c)(u - v)| \leqq M_1|u - v| \leqq M_1 h$$

Here, M_1 denotes the maximum of $|f'(x)|$ as x runs over (a, b). For example, if $f(x) = x^3$ and $[a, b] = [1, 4]$, then we find that $\omega(f; h) \leqq 48h$.

■ **THEOREM 1**

FIRST-DEGREE POLYNOMIAL ACCURACY THEOREM

If p is the first-degree polynomial that interpolates a function f at the endpoints of an interval $[a, b]$, then with $h = b - a$, we have

$$|f(x) - p(x)| \leqq \omega(f; h) \qquad (a \leqq x \leqq b)$$

Proof The linear function p is given explicitly by the formula

$$p(x) = \left(\frac{x-a}{b-a}\right) f(b) + \left(\frac{b-x}{b-a}\right) f(a)$$

Hence,

$$f(x) - p(x) = \left(\frac{x-a}{b-a}\right)[f(x) - f(b)] + \left(\frac{b-x}{b-a}\right)[f(x) - f(a)]$$

Then we have

$$\begin{aligned} |f(x) - p(x)| &\leqq \left(\frac{x-a}{b-a}\right)|f(x) - f(b)| + \left(\frac{b-x}{b-a}\right)|f(x) - f(a)| \\ &\leqq \left(\frac{x-a}{b-a}\right)\omega(f; h) + \left(\frac{b-x}{b-a}\right)\omega(f; h) \\ &= \left[\left(\frac{x-a}{b-a}\right) + \left(\frac{b-x}{b-a}\right)\right]\omega(f; h) = \omega(f; h) \end{aligned}$$

■

From this basic result, one can easily prove the following one, simply by applying the basic inequality to each subinterval.

■ **THEOREM 2**

FIRST-DEGREE SPLINE ACCURACY THEOREM

Let p be a first-degree spline having knots $a = x_0 < x_1 < \cdots < x_n = b$. If p interpolates a function f at these knots, then with $h = \max_i (x_i - x_{i-1})$, we have

$$|f(x) - p(x)| \leqq \omega(f; h) \qquad (a \leqq x \leqq b)$$

If f' or f'' exist and are continuous, then more can be said, namely,

$$|f(x) - p(x)| \leqq M_1 \frac{h}{2} \qquad (a \leqq x \leqq b)$$

$$|f(x) - p(x)| \leqq M_2 \frac{h^2}{8} \qquad (a \leqq x \leqq b)$$

In these estimates, M_1 is the maximum value of $|f'(x)|$ on the interval, and M_2 is the maximum of $|f''(x)|$.

The first theorem tells us that if more knots are inserted in such a way that the maximum spacing h goes to zero, then the corresponding first-degree spline will converge uniformly to f. Recall that this type of result is conspicuously lacking in the polynomial interpolation theory. In that situation, raising the degree and making the nodes fill up the interval will not necessarily ensure that convergence takes place for an arbitrary continuous function. (See Section 4.2.)

Second-Degree Splines

Splines of degree higher than 1 are more complicated. We now take up the quadratic splines. Let's use the letter Q to remind ourselves that we are considering piecewise quadratic functions. A function Q is a **second-degree spline** if it has the following properties.

■ **DEFINITION 2** **SPLINE OF DEGREE 2**

A function Q is called a **spline of degree 2** if:

1. The domain of Q is an interval $[a, b]$.
2. Q and Q' are continuous on $[a, b]$.
3. There are points t_i (called **knots**) such that $a = t_0 < t_1 < \cdots < t_n = b$ and Q is a polynomial of degree at most 2 on each subinterval $[t_i, t_{i+1}]$.

In brief, a quadratic spline is a continuously differentiable piecewise quadratic function, where *quadratic* includes all linear combinations of the basic functions $x \mapsto 1, x, x^2$.

EXAMPLE 2 Determine whether the following function is a quadratic spline:

$$Q(x) = \begin{cases} x^2 & (-10 \leqq x \leqq 0) \\ -x^2 & (0 \leqq x \leqq 1) \\ 1 - 2x & (1 \leqq x \leqq 20) \end{cases}$$

Solution The function is obviously piecewise quadratic. Whether Q and Q' are continuous at the interior knots can be determined as follows:

$$\begin{aligned}
\lim_{x\to 0^-} Q(x) &= \lim_{x\to 0^-} x^2 = 0 & \lim_{x\to 0^+} Q(x) &= \lim_{x\to 0^+} (-x^2) = 0 \\
\lim_{x\to 1^-} Q(x) &= \lim_{x\to 1^-} (-x^2) = -1 & \lim_{x\to 1^+} Q(x) &= \lim_{x\to 1^+} (1 - 2x) = -1 \\
\lim_{x\to 0^-} Q'(x) &= \lim_{x\to 0^-} 2x = 0 & \lim_{x\to 0^+} Q'(x) &= \lim_{x\to 0^+} (-2x) = 0 \\
\lim_{x\to 1^-} Q'(x) &= \lim_{x\to 1^-} (-2x) = -2 & \lim_{x\to 1^+} Q'(x) &= \lim_{x\to 1^+} (-2) = -2
\end{aligned}$$

Consequently, $Q(x)$ is a quadratic spline. ■

Interpolating Quadratic Spline $Q(x)$

Quadratic splines are not used in applications as often as are natural cubic splines, which are developed in the next section. However, the derivations of interpolating quadratic and cubic splines are similar enough that an understanding of the simpler second-degree spline theory will allow one to grasp easily the more complicated third-degree spline theory. We want to emphasize that quadratic splines are rarely used for interpolation, and the discussion here is provided only as preparation for the study of higher-order splines, which are used in many applications.

Proceeding now to the interpolation problem, suppose that a table of values has been given:

x	t_0	t_1	t_2	$\cdots$	t_n
y	y_0	y_1	y_2	$\cdots$	y_n

We shall assume that the points $t_0, t_1, \ldots, t_n$, which we think of as the **nodes** for the interpolation problem, are also the knots for the spline function to be constructed. Later, another quadratic spline interpolant is discussed in which the nodes for interpolation are different from the knots.

A quadratic spline, as just described, consists of n separate quadratic functions $x \mapsto a_i x^2 + b_i x + c_i$, one for each subinterval created by the $n + 1$ knots. Thus, we start with $3n$ coefficients. On each subinterval $[t_i, t_{i+1}]$, the quadratic spline function Q_i must satisfy the interpolation conditions $Q_i(t_i) = y_i$ and $Q_i(t_{i+1}) = y_{i+1}$. Since there are n such subintervals, this imposes $2n$ conditions. The continuity of Q does *not* add any additional conditions. (Why?) However, the continuity of Q' at each of the interior knots gives $n - 1$ more conditions. Thus, we have $2n + n - 1 = 3n - 1$ conditions, or *one* condition short of the $3n$ conditions required. There are a variety of ways to impose this additional condition; for example, $Q'(t_0) = 0$ or $Q_0'' = 0$.

We now derive the equations for the interpolating quadratic spline, $Q(x)$. The value of $Q'(t_0)$ is prescribed as the additional condition. We seek a piecewise quadratic function

$$Q(x) = \begin{cases} Q_0(x) & (t_0 \leqq x \leqq t_1) \\ Q_1(x) & (t_1 \leqq x \leqq t_2) \\ \quad\vdots & \quad\vdots \\ Q_{n-1}(x) & (t_{n-1} \leqq x \leqq t_n) \end{cases} \tag{4}$$

which is continuously differentiable on the entire interval $[t_0, t_n]$ and which interpolates the table; that is, $Q(t_i) = y_i$ for $0 \leqq i \leqq n$.

Since Q' is continuous, we can put $z_i \equiv Q'(t_i)$. At present, we do not know the correct values of z_i; nevertheless, the following must be the formula for Q_i:

$$Q_i(x) = \frac{z_{i+1} - z_i}{2(t_{i+1} - t_i)}(x - t_i)^2 + z_i(x - t_i) + y_i \tag{5}$$

To see that this is correct, verify that $Q_i(t_i) = y_i$, $Q_i'(t_i) = z_i$, and $Q_i'(t_{i+1}) = z_{i+1}$. These three conditions define the function Q_i uniquely on $[t_i, t_{i+1}]$ as given in Equation (5).

Now, for the quadratic spline function Q to be continuous and to interpolate the table of data, it is necessary and sufficient that $Q_i(t_{i+1}) = y_{i+1}$ for $i = 0, 1, \ldots, n - 1$ in Equation (5). When this equation is written out in detail and simplified, the result is

$$z_{i+1} = -z_i + 2\left(\frac{y_{i+1} - y_i}{t_{i+1} - t_i}\right) \qquad (0 \leqq i \leqq n - 1) \tag{6}$$

This equation can be used to obtain the vector $[z_0, z_1, \ldots, z_n]^T$, starting with an arbitrary value for z_0. We summarize with an algorithm:

■ **ALGORITHM 1** ***Quadratic Spline Interpolation at the Knots***

1. Determine $[z_0, z_1, \ldots, z_n]^T$ by selecting z_0 arbitrarily and computing $z_1, z_2, \ldots, z_n$ recursively by Formula (6).
2. The quadratic spline interpolating function Q is given by Formulas (4) and (5).

EXAMPLE 3 For the five data points (0, 8), (1, 12), (3, 2), (4, 6), (8, 0), construct the linear spline S and the quadratic spline Q.

Solution Figure 9.4 illustrates graphically these two low order spline curves. They fit better than the interpolating polynomials in Figure 4.6 (p. 154) with regard to reduced oscillations. ■

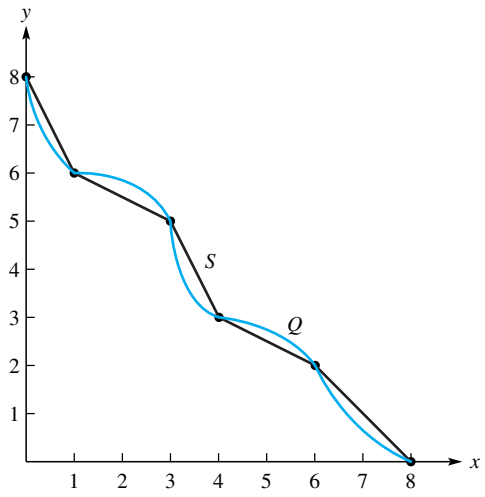

FIGURE 9.4 First-degree and second-degree spline functions

Subbotin Quadratic Spline

A useful approximation process, first proposed by Subbotin [1967], consists of interpolation with *quadratic* splines, where the nodes for interpolation are chosen to be the first and last knots and the midpoints between the knots. Remember that **knots** are defined as the points where the spline function is permitted to change in form from one polynomial to another. The **nodes** are the points where values of the spline are specified. In the Subbotin quadratic spline function, there are $n + 2$ interpolation conditions and $2(n - 1)$ conditions from the continuity of Q and Q'. Hence, we have the exact number of conditions needed, $3n$, to define the quadratic spline function completely.

We outline the theory here, leaving details for the reader to fill in. Suppose that knots $a = t_0 < t_1 < \cdots < t_n = b$ have been specified; let the nodes be the points

$$\begin{cases} \tau_0 = t_0 & \tau_{n+1} = t_n \\ \tau_i = \frac{1}{2}(t_i + t_{i-1}) & (1 \leqq i \leqq n) \end{cases}$$

We seek a quadratic spline function Q that has the given knots and takes prescribed values at the nodes:

$$Q(\tau_i) = y_i \qquad (0 \leqq i \leqq n + 1)$$

as in Figure 9.5. The knots create n subintervals, and in each of them, Q can be a different quadratic polynomial. Let us say that on $[t_i, t_{i+1}]$, Q is equal to the quadratic polynomial Q_i. Since Q is a quadratic spline, it and its first derivative should be continuous. Thus, $z_i \equiv Q'(t_i)$ is well defined, although as yet we do not know its values. It is easy to see that

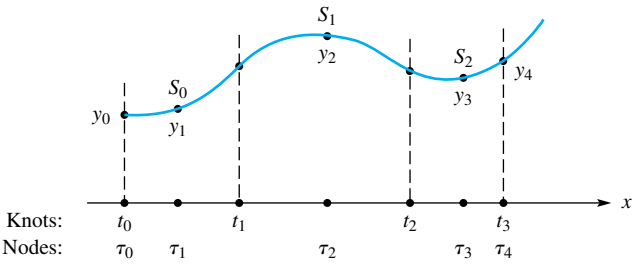

FIGURE 9.5
Subbotin quadratic splines ($t_0 = \tau_0$, $t_3 = \tau_4$)

on $[t_i, t_{i+1}]$, our quadratic polynomial can be represented in the form

$$Q_i(x) = y_{i+1} + \frac{1}{2}(z_{i+1} + z_i)(x - \tau_{i+1}) + \frac{1}{2h_i}(z_{i+1} - z_i)(x - \tau_{i+1})^2 \tag{7}$$

in which $h_i = t_{i+1} - t_i$. To verify the correctness of Equation (7), we must check that $Q_i(\tau_{i+1}) = y_{i+1}$, $Q_i'(t_i) = z_i$, and $Q_i'(t_{i+1}) = z_{i+1}$. When the polynomial pieces $Q_0, Q_1, \ldots, Q_{n-1}$ are joined together to form Q, the result may be discontinuous. Hence, we impose continuity conditions at the interior knots:

$$\lim_{x \to t_i^-} Q_{i-1}(x) = \lim_{x \to t_i^+} Q_i(x) \qquad (1 \leqq i \leqq n-1)$$

The reader should carry out this analysis, which leads to

$$h_{i-1}z_{i-1} + 3(h_{i-1} + h_i)z_i + h_i z_{i+1} = 8(y_{i+1} - y_i) \qquad (1 \leqq i \leqq n-1) \tag{8}$$

The first and last interpolation conditions must also be imposed:

$$Q(\tau_0) = y_0 \qquad Q(\tau_{n+1}) = y_{n+1}$$

These two equations lead to

$$3h_0z_0 + h_0z_1 = 8(y_1 - y_0)$$
$$h_{n-1}z_{n-1} + 3h_{n-1}z_n = 8(y_{n+1} - y_n)$$

The system of equations governing the vector $\mathbf{z} = [z_0, z_1, \ldots, z_n]^T$ then can be written in the matrix form

$$\begin{bmatrix} 3h_0 & h_0 & & & & & \\ h_0 & 3(h_0+h_1) & h_1 & & & & \\ & h_1 & 3(h_1+h_2) & h_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & h_{n-2} & 3(h_{n-2}+h_{n-1}) & h_{n-1} & \\ & & & & h_{n-1} & 3h_{n-1} \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix}$$
$$= 8 \begin{bmatrix} y_1 - y_0 \\ y_2 - y_1 \\ y_3 - y_2 \\ \vdots \\ y_n - y_{n-1} \\ y_{n+1} - y_n \end{bmatrix}$$

This system of $n+1$ equations in $n+1$ unknowns can be conveniently solved by procedure *Tri* in Chapter 7. After the z vector has been obtained, values of $Q(x)$ can be computed from Equation (7). The writing of suitable code to carry out this interpolation method is left as a programming project.

Summary

(1) We are given $n+1$ pairs of points (t_i, y_i) with distinct **knots** $a = t_0 < t_1 < \cdots < t_{n-1} < t_n = b$ over the interval $[a, b]$. A **first-degree spline function** S is a piecewise linear polynomial defined on the interval $[a, b]$ so that it is continuous. It has the form

$$S(x) = \begin{cases} S_0(x) & x \in [t_0, t_1] \\ S_1(x) & x \in [t_1, t_2] \\ \vdots & \vdots \\ S_{n-1}(x) & x \in [t_{n-1}, t_n] \end{cases}$$

where

$$S_i(x) = y_i + \left(\frac{y_{i+1} - y_i}{t_{i+1} - t_i}\right)(x - t_i)$$

on the interval $[t_i, t_{i+1}]$. Clearly, $S(x)$ is continuous, since $S_{i-1}(t_i) = S_i(t_i) = y_i$ for $1 \leqq i \leqq n$.

(2) A **second-degree spline function** Q is a piecewise quadratic polynomial with Q and Q' continuous on the interval $[a, b]$. It has the form

$$Q(x) = \begin{cases} Q_0(x) & x \in [t_0, t_1] \\ Q_1(x) & x \in [t_1, t_2] \\ \vdots & \vdots \\ Q_{n-1}(x) & x \in [t_{n-1}, t_n] \end{cases}$$

where

$$Q_i(x) = \left(\frac{z_{i+1} - z_i}{2(t_{i+1} - t_i)}\right)(x - t_i)^2 + z_i(x - t_i) + y_i$$

on the interval $[t_i, t_{i+1}]$. The coefficients $z_0, z_1, \ldots, z_n$ are obtained by selecting z_0 and then using the recurrence relation

$$z_{i+1} = -z_i + 2\left(\frac{y_{i+1} - y_i}{t_{i+1} - t_i}\right) \qquad (0 \leqq i \leqq n-1)$$

(3) A **Subbotin quadratic spline function** Q is a piecewise quadratic polynomial with Q and Q' continuous on the interval $[a, b]$ and with interpolation condition at the endpoints of the interval $[a, b]$ and at the midpoints of the subintervals, namely, $Q(\tau_i) = y_i$ for $0 \leqq i \leqq n+1$, where

$$\tau_0 = t_0, \qquad \tau_i = \frac{1}{2}(t_i + t_{i-1}) \quad (1 \leqq i \leqq n), \qquad \tau_{n+1} = t_n$$

It has the form

$$Q_i(x) = y_{i+1} + \frac{1}{2}(z_{i+1} + z_i)(x - \tau_{i+1}) + \frac{1}{2h_i}(z_{i+1} - z_i)(x - \tau_{i+1})^2$$

where $h_i = t_{i+1} - t_i$. The coefficients z_i are found by solving the tridiagonal system

$$\begin{cases} 3h_0 z_0 + h_0 z_1 = 8(y_1 - y_0) \\ h_{i-1} z_{i-1} + 3(h_{i-1} + h_i) z_i + h_i z_{i+1} = 8(y_{i+1} - y_i) \qquad (1 \leqq i \leqq n-1) \\ h_{n-1} z_{n-1} + 3h_{n-1} z_n = 8(y_{n+1} - y_n) \end{cases}$$

as discussed in Section 7.3.

9.2 Natural Cubic Splines

Introduction

The first- and second-degree splines discussed in the preceding section, though useful in certain applications, suffer an obvious imperfection: Their low-order derivatives are discontinuous. In the case of the first-degree spline (or polygonal line), this lack of smoothness is immediately evident because the slope of the spline may change abruptly from one value to another at each knot. For the quadratic spline, the discontinuity is in the second derivative and is therefore not so evident. But the *curvature* of the quadratic spline changes abruptly at each knot, and the curve may not be pleasing to the eye.

The general definition of spline functions of arbitrary degree is as follows.

■ **DEFINITION 1** **SPLINE OF DEGREE *k***

A function S is called a **spline of degree k** if:

1. The domain of S is an interval $[a, b]$.
2. $S, S', S'', \ldots, S^{(k-1)}$ are all continuous functions on $[a, b]$.
3. There are points t_i (the knots of S) such that $a = t_0 < t_1 < \cdots < t_n = b$ and such that S is a polynomial of degree at most k on each subinterval $[t_i, t_{i+1}]$.

Observe that no mention has been made of interpolation in the definition of a spline function. Indeed, splines are such versatile functions that they have many applications other than interpolation.

Higher-degree splines are used whenever more smoothness is needed in the approximating function. From the definition of a spline function of degree k, we see that such a function will be continuous and have continuous derivatives $S', S'', \ldots, S^{(k-1)}$. If we want the approximating spline to have a continuous mth derivative, a spline of degree at least $m + 1$ is selected. To see why, consider a situation in which knots $t_0 < t_1 < \cdots < t_n$ have been prescribed. Suppose that a piecewise polynomial of degree m is to be defined, with its pieces joined at the knots in such a way that the resulting spline S has m continuous derivatives. At a typical interior knot t, we have the following circumstances: To the left of t, $S(x) = p(x)$; to the right of t, $S(x) = q(x)$, where p and q are mth-degree polynomials. The continuity of the mth derivative $S^{(m)}$ implies the continuity of the lower-order derivatives $S^{(m-1)}, S^{(m-2)}, \ldots, S', S$. Therefore, at the knot t,

$$\lim_{x \to t^-} S^{(k)}(x) = \lim_{x \to t^+} S^{(k)}(x) \qquad (0 \leqq k \leqq m)$$

from which we conclude that

$$\lim_{x \to t^-} p^{(k)}(x) = \lim_{x \to t^+} q^{(k)}(x) \qquad (0 \leqq k \leqq m) \tag{1}$$

Since p and q are polynomials, their derivatives of all orders are continuous, and so Equation (1) is the same as

$$p^{(k)}(t) = q^{(k)}(t) \qquad (0 \leqq k \leqq m)$$

This condition forces p and q to be the *same* polynomial because by Taylor's Theorem,

$$p(x) = \sum_{k=0}^{m} \frac{1}{k!} p^{(k)}(t)(x-t)^k = \sum_{k=0}^{m} \frac{1}{k!} q^{(k)}(t)(x-t)^k = q(x)$$

This argument can be applied at each of the interior knots $t_1, t_2, \ldots, t_{n-1}$, and we see that S is simply one polynomial throughout the entire interval from t_0 to t_n. Thus, we need a piecewise polynomial of degree $m+1$ with at most m continuous derivatives to have a spline function that is not just a single polynomial throughout the entire interval. (We already know that ordinary polynomials usually do not serve well in curve fitting. See Section 4.2.)

The choice of degree most frequently made for a spline function is 3. The resulting splines are termed **cubic splines**. In this case, we join cubic polynomials together in such a way that the resulting spline function has two continuous derivatives everywhere. At each knot, three continuity conditions will be imposed. Since S, S', and S'' are continuous, the graph of the function will appear smooth to the eye. Discontinuities, of course, will occur in the third derivative but cannot be easily detected visually, which is one reason for choosing degree 3. Experience has shown, moreover, that using splines of degree greater than 3 seldom yields any advantage. For technical reasons, odd-degree splines behave better than even-degree splines (when interpolating at the knots). Finally, a very elegant theorem, to be proved later, shows that in a certain precise sense, the cubic interpolating spline function is the best interpolating function available. Thus, our emphasis on the cubic splines is well justified.

Natural Cubic Spline

We turn next to interpolating a given table of function values by a cubic spline whose knots coincide with the values of the independent variable in the table. As earlier, we start with the table:

x	t_0	t_1	$\cdots$	t_n
y	y_0	y_1	$\cdots$	y_n

The t_i's are the knots and are assumed to be arranged in ascending order.

The function S that we wish to construct consists of n cubic polynomial pieces:

$$S(x) = \begin{cases} S_0(x) & (t_0 \leqq x \leqq t_1) \\ S_1(x) & (t_1 \leqq x \leqq t_2) \\ \vdots & \vdots \\ S_{n-1}(x) & (t_{n-1} \leqq x \leqq t_n) \end{cases}$$

In this formula, S_i denotes the cubic polynomial that will be used on the subinterval $[t_i, t_{i+1}]$. The interpolation conditions are

$$S(t_i) = y_i \qquad (0 \leqq i \leqq n)$$

The continuity conditions are imposed only at the *interior* knots $t_1, t_2, \ldots, t_{n-1}$. (Why?) These conditions are written as

$$\lim_{x \to t_i^-} S^{(k)}(t_i) = \lim_{x \to t_i^+} S^{(k)}(t_i) \qquad (k = 0, 1, 2)$$

It turns out that two more conditions must be imposed to use all the degrees of freedom available. The choice that we make for these two extra conditions is

$$S''(t_0) = S''(t_n) = 0 \tag{2}$$

The resulting spline function is then termed a **natural cubic spline**. Additional ways to close the system of equations for the spline coefficients are **periodic cubic splines** and **clamped cubic splines**. A clamped spline is a spline curve whose slope is fixed at both end points: $S'(t_0) = d_0$ and $S'(t_n) = d_n$. A periodic cubic spline has $S(t_0) = S(t_n)$, $S'(t_0) = S'(t_n)$, and $S''(t_0) = S''(t_n)$. For all continuous differential functions, clamped and natural cubic splines yield the least oscillations about the function f that it interpolates.

We now verify that the number of conditions imposed equals the number of coefficients available. There are $n + 1$ knots and hence n subintervals. On each of these subintervals, we shall have a different cubic polynomial. Since a cubic polynomial has four coefficients, a total of $4n$ coefficients are available. As for the conditions imposed, we have specified that within each interval the interpolating polynomial must go through two points, which gives $2n$ conditions. The continuity adds no additional conditions. The first and second derivatives must be continuous at the $n - 1$ interior points, for $2(n - 1)$ more conditions. The second derivatives must vanish at the two endpoints for a total of $2n+2(n-1)+2 = 4n$ conditions.

EXAMPLE 1 Derive the equations of the natural cubic interpolating spline for the following table:

x	-1	0	1
y	1	2	-1

Solution Our approach is to determine the parameters a, b, c, d, e, f, g, and h so that $S(x)$ is a natural cubic spline, where

$$S(x) = \begin{cases} S_0(s) = ax^3 + bx^2 + cx + d & x \in [-1, 0] \\ S_1(s) = ex^3 + fx^2 + gx + h & x \in [0, 1] \end{cases}$$

where the two cubic polynomials are $S_0(x)$ and $S_1(x)$. From these interpolation conditions, we have interpolation conditions $S(-1) = S_0(-1) = -a + b - c + d = 1$, $S(0) = S_0(0) = d = 2$, $S(0) = S_1(0) = h = 2$, and $S(1) = S_1(1) = e + f + g + h = -1$. Taking the first derivatives, we obtain

$$S'(x) = \begin{cases} S_0'(x) = 3ax^2 + 2bx + c \\ S_1'(x) = 3ex^2 + 2fx + g \end{cases}$$

From the continuity condition of S', we have $S_0'(0) = S_1'(0)$, and we set $c = g$. Next taking the second derivatives, we obtain

$$S''(x) = \begin{cases} S_0''(x) = 6ax + 2b \\ S_1''(s) = 6ex + 2f \end{cases}$$

From the continuity condition of S'', we have $S_0''(0) = S_1''(0)$, and we let $b = f$. For S to be a natural cubic spline, we must have $S_0''(-1) = 0$ and $S_1''(1) = 0$, and we obtain $3a = b$ and $3e = -f$. From all of these equations, we obtain $a = -1$, $b = -3$, $c = -1$, $d = 2$, $e = 1$, $f = -3$, $g = -1$, and $h = 2$. ■

Algorithm for Natural Cubic Spline

From the previous example, it is evident that we need to develop a systematic procedure for determining the formula for a natural cubic spline, given a table of interpolation values. This is our objective in the material on the next several pages.

Since S'' is continuous, the numbers

$$z_i \equiv S''(t_i) \qquad (0 \leqq i \leqq n)$$

are unambiguously defined. We do not yet know the values $z_1, z_2, \ldots, z_{n-1}$, but, of course, $z_0 = z_n = 0$ by Equation (2).

If the z_i's were known, we could construct S as now described. On the interval $[t_i, t_{i+1}]$, S'' is a linear polynomial that takes the values z_i and z_{i+1} at the endpoints. Thus,

$$S_i''(x) = \frac{z_{i+1}}{h_i}(x - t_i) + \frac{z_i}{h_i}(t_{i+1} - x) \tag{3}$$

with $h_i = t_{i+1} - t_i$ for $0 \leqq i \leqq n-1$. To verify that Equation (3) is correct, notice that $S_i''(t_i) = z_i$, $S_i''(t_{i+1}) = z_{i+1}$, and S_i'' is linear in x. If this is integrated twice, we obtain S_i itself:

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + cx + d$$

where c and d are constants of integration. By adjusting the integration constants, we obtain a form for S_i that is easier to work with, namely,

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + C_i(x - t_i) + D_i(t_{i+1} - x) \tag{4}$$

where C_i and D_i are constants. If we differentiate Equation (4) twice, we obtain Equation (3).

The interpolation conditions $S_i(t_i) = y_i$ and $S_i(t_{i+1}) = y_{i+1}$ can be imposed now to determine the appropriate values of C_i and D_i. The reader should do so (Problem 9.2.27) and verify that the result is

$$\begin{aligned} S_i(x) = {} & \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 \\ & + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right)(t_{i+1} - x) \end{aligned} \tag{5}$$

When the values $z_0, z_1, \ldots, z_n$ have been determined, the spline function $S(x)$ is obtained from equations of this form for $S_0(x), S_1(x), \ldots, S_{n-1}(x)$.

We now show how to determine the z_i's. One condition remains to be imposed—namely, the continuity of S'. At the interior knots t_i for $1 \leqq i \leqq n-1$, we must have $S_{i-1}'(t_i) = S_i'(t_i)$, as can be seen in Figure 9.6.

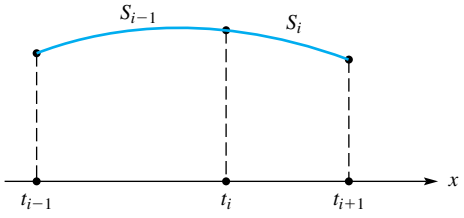

FIGURE 9.6
Cubic spline: adjacent pieces S_{i-1} and S_i

We have, from Equation (5),

$$S_i'(x) = \frac{z_{i+1}}{2h_i}(x - t_i)^2 - \frac{z_i}{2h_i}(t_{i+1} - x)^2 + \frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1} - \frac{y_i}{h_i} + \frac{h_i}{6}z_i$$

This gives

$$S_i'(t_i) = -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + b_i \tag{6}$$

where

$$b_i = \frac{1}{h_i}(y_{i+1} - y_i) \tag{7}$$

Analogously, we have

$$S_{i-1}'(t_i) = \frac{h_{i-1}}{6}z_{i-1} + \frac{h_{i-1}}{3}z_i + b_{i-1}$$

When these are set equal to each other, the resulting equation can be rearranged as

$$h_{i-1}z_{i-1} + 2(h_{i-1} + h_i)z_i + h_i z_{i+1} = 6(b_i - b_{i-1})$$

for $1 \leqq i \leqq n - 1$. By letting

$$\begin{aligned} u_i &= 2(h_{i-1} + h_i) \\ v_i &= 6(b_i - b_{i-1}) \end{aligned} \tag{8}$$

we obtain a tridiagonal system of equations:

$$\begin{cases} z_0 = 0 \\ h_{i-1}z_{i-1} + u_i z_i + h_i z_{i+1} = v_i \qquad (1 \leqq i \leqq n-1) \\ z_n = 0 \end{cases} \tag{9}$$

to be solved for the z_i's. The simplicity of the first and last equations is a result of the natural cubic spline conditions $S''(t_0) = S''(t_n) = 0$.

EXAMPLE 2 Repeat Example 1 by constructing the natural cubic spline through the points $(-1, 1)$, $(0, 2)$, and $(1, -1)$. Also, plot the results in order to visualize the spline curve.

Solution From the given values, we have $t_0 = -1, t_1 = 0, t_2 = 1, y_0 = 1, y_1 = 2$, and $y_2 = -1$. Consequently, we obtain $h_0 = t_1 - t_0 = 1$, $h_1 = t_2 - t_1 = 1$, $b_0 = (y_1 - y_0)/h_0 = 1$, $b_1 = (y_2 - y_1)/h_1 = -3$, $u_1 = 2(h_0 - h_1) = 4$, and $v_1 = 6(b_1 - b_0) = -24$. Then the tridiagonal system of equations (9) is

$$\begin{cases} z_0 = 0 \\ z_0 + 4z_1 + z_2 = -24 \\ z_2 = 0 \end{cases}$$

Evidently, we obtain the solution $z_0 = 0, z_1 = -6$, and $z_2 = 0$. From Equation (5), we have

$$S(x) = \begin{cases} S_0(x) = -(x+1)^3 + 3(x+1) - x & x \in [-1, 0] \\ S_1(x) = -(1-x)^3 - x + 3(1-x) & x \in [0, 1] \end{cases}$$

or

$$S(x) = \begin{cases} S_0(x) = -x^3 - 3x^2 - x + 2 & x \in [-1, 0] \\ S_1(x) = x^3 - 3x^2 - x + 2 & x \in [0, 1] \end{cases}$$

This agrees with the results from Example 1. The resulting natural spline curve through the given points is shown in Figure 9.7.

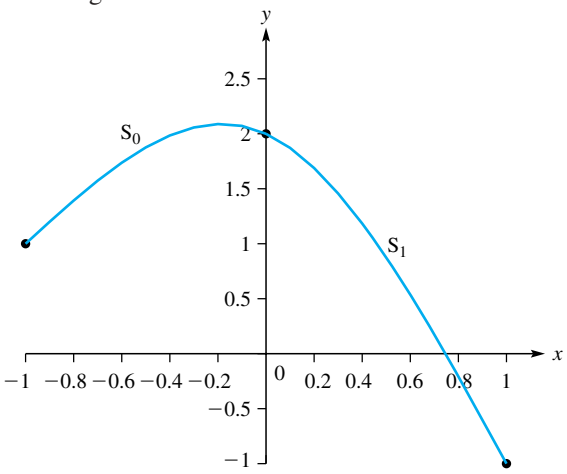

FIGURE 9.7 Natural cubic spline for Examples 1 and 2

Now consider System (9) in matrix form:

$$\begin{bmatrix} 1 & 0 & & & & \\ h_0 & u_1 & h_1 & & & \\ & h_1 & u_2 & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-2} & u_{n-1} & h_{n-1} \\ & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_{n-1} \\ z_n \end{bmatrix} = \begin{bmatrix} 0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ 0 \end{bmatrix}$$

On eliminating the first and last equations, we have

$$\begin{bmatrix} u_1 & h_1 & & & \\ h_1 & u_2 & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} \tag{10}$$

which is a symmetric tridiagonal system of order $n - 1$. We could use procedure *Tri* developed in Section 7.3 to solve this system. However, we can design an algorithm specifically for it (based on the ideas in Section 7.3). In Gaussian elimination *without pivoting*, the forward elimination phase would modify the u_i's and v_i's as follows:

$$\begin{cases} u_i \leftarrow u_i - \dfrac{h_{i-1}^2}{u_{i-1}} \\ v_i \leftarrow v_i - \dfrac{h_{i-1}v_{i-1}}{u_{i-1}} \end{cases} \qquad (i = 2, 3, \ldots, n-1)$$

The back substitution phase yields

$$\begin{cases} z_{n-1} \leftarrow \dfrac{v_{n-1}}{u_{n-1}} \\ z_i \leftarrow \dfrac{v_i - h_i z_{i+1}}{u_i} \end{cases} \qquad (i = n-2, n-3, \ldots, 1)$$

Putting all this together leads to the following algorithm, designed especially for the tridiagonal System (10).

■ **ALGORITHM 1** ***Solving the Natural Cubic Spline Tridiagonal System Directly***

Given the interpolation points (t_i, y_i) for $i = 0, 1, \ldots, n$:

1. Compute for $i = 0, 1, \ldots, n-1$:

$$\begin{cases} h_i = t_{i+1} - t_i \\ b_i = \dfrac{1}{h_i}(y_{i+1} - y_i) \end{cases}$$

2. Set

$$\begin{cases} u_1 = 2(h_0 + h_1) \\ v_1 = 6(b_1 - b_0) \end{cases}$$

and compute inductively for $i = 2, 3, \ldots, n-1$:

$$\begin{cases} u_i = 2(h_i + h_{i-1}) - \dfrac{h_{i-1}^2}{u_{i-1}} \\ v_i = 6(b_i - b_{i-1}) - \dfrac{h_{i-1}v_{i-1}}{u_{i-1}} \end{cases}$$

3. Set

$$\begin{cases} z_n = 0 \\ z_0 = 0 \end{cases}$$

and compute inductively for $i = n-1, n-2, \ldots, 1$:

$$z_i = \frac{v_i - h_i z_{i+1}}{u_i}$$

This algorithm conceivably could fail because of divisions by zero in steps 2 and 3. Therefore, let us prove that $u_i \neq 0$ for all i. It is clear that $u_1 > h_1 > 0$. If $u_{i-1} > h_{i-1}$, then $u_i > h_i$ because

$$u_i = 2(h_i + h_{i-1}) - \frac{h_{i-1}^2}{u_{i-1}} > 2(h_i + h_{i-1}) - h_{i-1} > h_i$$

Then by induction, $u_i > 0$ for $i = 1, 2, \ldots, n-1$.

Equation (5) is not the best computational form for evaluating the cubic polynomial $S_i(x)$. We would prefer to have it in the form

$$S_i(x) = A_i + B_i(x - t_i) + C_i(x - t_i)^2 + D_i(x - t_i)^3 \tag{11}$$

because nested multiplication can then be utilized.

Notice that Equation (11) is the Taylor expansion of S_i about the point t_i. Hence,

$$A_i = S_i(t_i), \qquad B_i = S_i'(t_i), \qquad C_i = \tfrac{1}{2}S_i''(t_i), \qquad D_i = \tfrac{1}{6}S_i'''(t_i)$$

Therefore, $A_i = y_i$ and $C_i = z_i/2$. The coefficient of x^3 in Equation (11) is D_i, whereas the coefficient of x^3 in Equation (5) is $(z_{i+1} - z_i)/6h_i$. Therefore,

$$D_i = \frac{1}{6h_i}(z_{i+1} - z_i)$$

Finally, Equation (6) provides the value of $S_i'(t_i)$, which is

$$B_i = -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + \frac{1}{h_i}(y_{i+1} - y_i)$$

Thus, the nested form of $S_i(x)$ is

$$S_i(x) = y_i + (x - t_i)\left(B_i + (x - t_i)\left(\frac{z_i}{2} + \frac{1}{6h_i}(x - t_i)(z_{i+1} - z_i)\right)\right) \tag{12}$$

Pseudocode for Natural Cubic Splines

We now write routines for determining a natural cubic spline based on a table of values and for evaluating this function at a given value. First, we use Algorithm 1 for directly solving the tridiagonal System (10). This procedure, called *Spline3_Coef*, takes $n + 1$ table values (t_i, y_i) in arrays (t_i) and (y_i) and computes the z_i's, storing them in array (z_i). Intermediate (working) arrays (h_i), (b_i), (u_i), and (v_i) are needed.

```
procedure Spline3_Coef(n, (t_i), (y_i), (z_i))
integer i, n;   real array (t_i)_{0:n}, (y_i)_{0:n}, (z_i)_{0:n}
allocate real array (h_i)_{0:n-1}, (b_i)_{0:n-1}, (u_i)_{1:n-1}, (v_i)_{1:n-1}
for i = 0 to n - 1 do
    h_i ← t_{i+1} - t_i
    b_i ← (y_{i+1} - y_i)/h_i
end for
u_1 ← 2(h_0 + h_1)
v_1 ← 6(b_1 - b_0)
for i = 2 to n - 1 do
    u_i ← 2(h_i + h_{i-1}) - h_{i-1}^2/u_{i-1}
    v_i ← 6(b_i - b_{i-1}) - h_{i-1}v_{i-1}/u_{i-1}
end for
z_n ← 0
for i = n - 1 to 1 step -1 do
    z_i ← (v_i - h_i z_{i+1})/u_i
end for
z_0 ← 0
deallocate array (h_i), (b_i), (u_i), (v_i)
end procedure Spline3_Coef
```

Now a procedure called *Spline3_Eval* is written for evaluating Equation (12), the natural cubic spline function $S(x)$, for x a given value. The procedure *Spline3_Eval* first determines the interval $[t_i, t_{i+1}]$ that contains x and then evaluates $S_i(x)$ using the nested form of this cubic polynomial:

```
real function Spline3_Eval(n, (t_i), (y_i), (z_i), x)
integer i;   real h, tmp
real array (t_i)_{0:n}, (y_i)_{0:n}, (z_i)_{0:n}
for i = n - 1 to 0 step -1 do
    if x - t_i ≧ 0 then exit loop
```

```
    end for
    h ← t_{i+1} − t_i
    tmp ← (z_i/2) + (x − t_i)(z_{i+1} − z_i)/(6h)
    tmp ← −(h/6)(z_{i+1} + 2z_i) + (y_{i+1} − y_i)/h + (x − t_i)(tmp)
    Spline3_Eval ← y_i + (x − t_i)(tmp)
    end function Spline3_Eval
```

The function *Spline3_Eval* can be used repeatedly with different values of x after one call to procedure *Spline3_Coef*. For example, this would be the procedure when plotting a natural cubic spline curve. Since procedure *Spline3_Coef* stores the solution of the tridiagonal system corresponding to a particular spline function in the array (z_i), the arguments n, (t_i), (y_i), and (z_i) must not be altered between repeated uses of *Spline3_Eval*.

Using Pseudocode for Interpolating and Curve Fitting

To illustrate the use of the natural cubic spline routines *Spline3_Coef* and *Spline3_Eval*, we rework an example from Section 4.1.

EXAMPLE 3 Write pseudocode for a program that determines the natural cubic spline interpolant for $\sin x$ at ten equidistant knots in the interval $[0, 1.6875]$. Over the same interval, subdivide each subinterval into four equally spaced parts, and find the point where the value of $|\sin x - S(x)|$ is largest.

Solution Here is a suitable pseudocode main program, which calls procedures *Spline3_Coef* and *Spline3_Eval*:

```
procedure Test_Spline3
integer i;   real e, h, x
real array (t_i)_{0:n}, (y_i)_{0:n}, (z_i)_{0:n}
integer n ← 9
real a ← 0, b ← 1.6875
h ← (b − a)/n
for i = 0 to n do
    t_i ← a + ih
    y_i ← sin(t_i)
end for
call Spline3_Coef(n, (t_i), (y_i), (z_i))
temp ← 0
for j = 0 to 4n do
    x ← a + jh/4
    e ← |sin(x) − Spline3_Eval(n, (t_i), (y_i), (z_i), x)|
    if e > temp then temp ← e
    output j, x, e
end for
end Test_Spline3
```

From the computer, the output is $j = 19$, $x = 0.890625$, and $d = 0.930 \times 10^{-5}$. ■

We can use mathematical software such as in Matlab to plot the cubic spline curve for this data, but the Matlab routine `spline` uses the **not-a-knot end condition**, which is different from the natural end condition. It dictates that S''' be a single constant in the first two subintervals and another single constant in the last two subintervals. First, the original data are generated. Next, a finer subdivision of the interval $[a, b]$ on the x-axis is made, and the corresponding y-values are obtained from the procedure `spline`. Finally, the original data points and the spline curve are plotted.

We now illustrate the use of spline functions in fitting a curve to a set of data. Consider the following table:

x	0.0	0.6	1.5	1.7	1.9	2.1	2.3	2.6	2.8	3.0
y	−0.8	−0.34	0.59	0.59	0.23	0.1	0.28	1.03	1.5	1.44

3.6	4.7	5.2	5.7	5.8	6.0	6.4	6.9	7.6	8.0
0.74	−0.82	−1.27	−0.92	−0.92	−1.04	−0.79	−0.06	1.0	0.0

These 20 points were selected from a wiggly freehand curve drawn on graph paper. We intentionally selected more points where the curve bent sharply and sought to reproduce the curve using an automatic plotter. A visually pleasing curve is provided by using the cubic spline routines *Spline3_Coef* and *Spline3_Eval*. Figure 9.8 shows the resulting natural cubic spline curve.

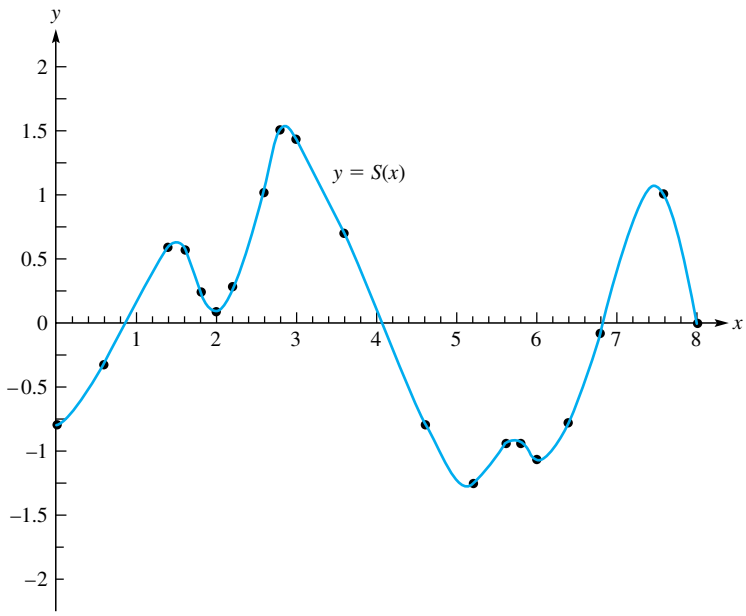

FIGURE 9.8
Natural cubic spline curve

Alternatively, we can use mathematical software such as Matlab, Maple, or Mathematica to plot the cubic spline function for this table.

Space Curves

In two dimensions, two cubic spline functions can be used together to form a **parametric representation** of a complicated curve that turns and twists. Select points on the curve and

label them $t = 0, 1, \ldots, n$. For each value of t, read off the x- and y-coordinates of the point, thus producing a table:

t	0	1	$\cdots$	n
x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

Then fit $x = S(t)$ and $y = \overline{S}(t)$, where S and $\overline{S}$ are natural cubic spline interpolants. The two functions S and $\overline{S}$ give a parametric representation of the curve. (See Computer Problem 9.2.6.)

EXAMPLE 4 Select 13 points on the well-known **serpentine curve** given by

$$y = \frac{x}{1/4 + x^2}$$

So that the knots will not be equally spaced, write the curve in parametric form:

$$\begin{cases} x = \frac{1}{2}\tan\theta \\ y = \sin 2\theta \end{cases}$$

and take $\theta = i(\pi/12)$, where $i = -6, -5, \ldots, 5, 6$. Plot the natural cubic spline curve and the interpolation polynomial in order to compare them.

Solution This is example of curve fitting using both the polynomial interpolation routines *Coef* and *Eval* from Chapter 4 and the cubic spline routines *Spline3_Coef* and *Spline3_Eval*. Figure 9.9 shows the resulting cubic spline curve and the high-degree polynomial curve (dashed line) from an automatic plotter. The polynomial becomes extremely erratic after the fourth knot from the origin and oscillates wildly, whereas the spline is a near perfect fit.

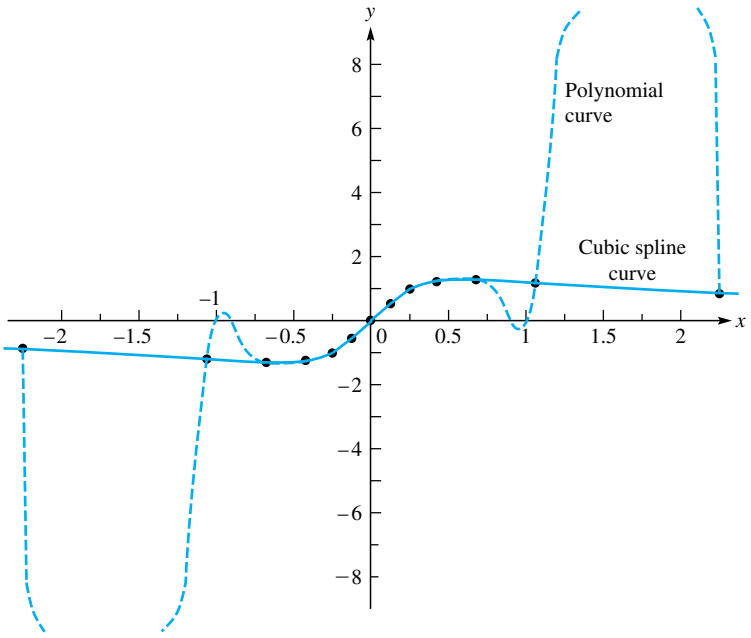

FIGURE 9.9 Serpentine curve

■

EXAMPLE 5 Use cubic spline functions to produce the curve for the following data:

t	0	1	2	3	4	5	6	7
y	1.0	1.5	1.6	1.5	0.9	2.2	2.8	3.1

It is known that the curve is continuous but its slope is not.

Solution A single cubic spline is not suitable. Instead, we can use two cubic spline interpolants, the first having knots 0, 1, 2, 3, 4 and the second having knots 4, 5, 6, 7. By carrying out two separate spline interpolation procedures, we obtain two cubic spline curves that meet at the point (4, 0.9). At this point, the two curves have different slopes. The resulting curve is shown in Figure 9.10.

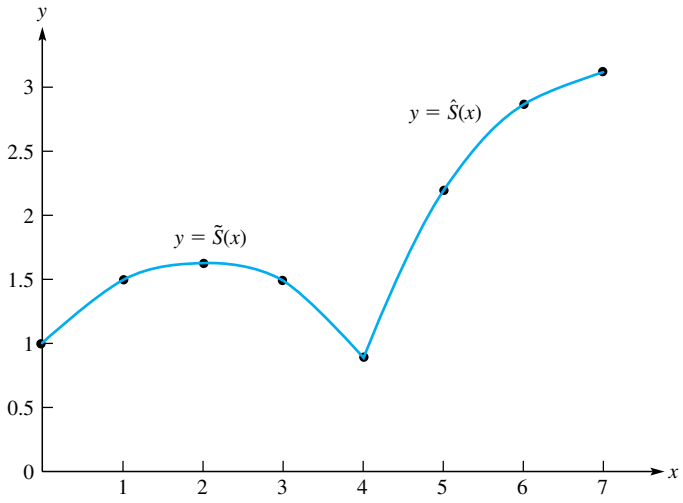

FIGURE 9.10 Two cubic splines

Smoothness Property

Why do spline functions serve the needs of data fitting better than ordinary polynomials? To answer this, one should understand that interpolation by polynomials of high degree is often unsatisfactory because polynomials may exhibit wild *oscillations*. Polynomials are smooth in the technical sense of possessing continuous derivatives of all orders, whereas in this sense, spline functions are *not* smooth.

Wild oscillations in a function can be attributed to its derivatives being very large. Consider the function whose graph is shown in Figure 9.11. The slope of the chord that

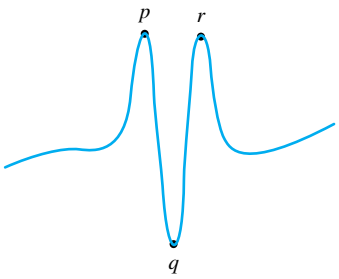

FIGURE 9.11 Wildly oscillating function

joins the points p and q is very large in magnitude. By the Mean-Value Theorem, the slope of that chord is the value of the derivative at some point between p and q. Thus, the derivative must attain large values. Indeed, somewhere on the curve between p and q, there is a point where $f'(x)$ is large and negative. Similarly, between q and r, there is a point where $f'(x)$ is large and positive. Hence, there is a point on the curve between p and r where $f''(x)$ is large. This reasoning can be continued to higher derivatives if there are more oscillations. This is the behavior that spline functions do *not* exhibit. In fact, the following result shows that from a certain point of view, natural cubic splines are the *best* functions to use for curve fitting.

■ **THEOREM 1**

CUBIC SPLINE SMOOTHNESS THEOREM

If S is the natural cubic spline function that interpolates a twice-continuously differentiable function f at knots $a = t_0 < t_1 < \cdots < t_n = b$, then

$$\int_a^b [S''(x)]^2\,dx \leqq \int_a^b [f''(x)]^2\,dx$$

Proof To verify the assertion about $[S''(x)]^2$, we let

$$g(x) = f(x) - S(x)$$

so that $g(t_i) = 0$ for $0 \leqq i \leqq n$, and

$$f'' = S'' + g''$$

Now

$$\int_a^b (f'')^2\,dx = \int_a^b (S'')^2\,dx + \int_a^b (g'')^2\,dx + 2\int_a^b S''g''\,dx$$

If the last integral were 0, we would be finished because then

$$\int_a^b (f'')^2\,dx = \int_a^b (S'')^2\,dx + \int_a^b (g'')^2\,dx \geqq \int_a^b (S'')^2\,dx$$

We apply the technique of integration by parts to the integral in question to show that it is 0.* We have

$$\int_a^b S''g''\,dx = S''g'\Big|_a^b - \int_a^b S'''g'\,dx = -\int_a^b S'''g'\,dx$$

*The formula for **integration by parts** is

$$\int u\,dv = uv - \int v\,du$$

Here, use has been made of the fact that S is a *natural* cubic spline; that is, $S''(a) = 0$ and $S''(b) = 0$. Continuing, we have

$$\int_a^b S''' g' \, dx = \sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} S''' g' \, dx$$

Since S is a cubic polynomial in each interval $[t_i, t_{i+1}]$, its third derivative there is a constant, say c_i. So

$$\int_a^b S''' g' \, dx = \sum_{i=0}^{n-1} c_i \int_{t_i}^{t_{i+1}} g' \, dx = \sum_{i=0}^{n-1} c_i \left[g(t_{i+1}) - g(t_i) \right] = 0$$

because g vanishes at every knot. ■

The interpretation of the integral inequality in the theorem is that the average value of $[S''(x)]^2$ on the interval $[a, b]$ is never larger than the average value of this expression with any twice-continuous function f that agrees with S at the knots. The quantity $[f''(x)]^2$ is closely related to the curvature of the function f.

Summary

(1) We are given $n + 1$ pairs of points (t_i, y_i) with distinct **knots** $a = t_0 < t_1 < \cdots < t_{n-1} < t_n = b$ over the interval $[a, b]$. A **spline function of degree *k*** is a piecewise polynomial function so that $S, S', S'', \ldots, S^{(k-1)}$ are all continuous functions on $[a, b]$ and S is a polynomial of degree at most k on each subinterval $[t_i, t_{i+1}]$.

(2) A **natural cubic spline function** S is a piecewise cubic polynomial defined on the interval $[a, b]$ so that S, S', S'' are continuous and $S''(t_0) = S''(t_n) = 0$. It can be written in the form

$$S(x) = \begin{cases} S_0(x) & x \in [t_0, t_1] \\ S_1(x) & x \in [t_1, t_2] \\ \vdots & \vdots \\ S_{n-1}(x) & x \in [t_{n-1}, t_n] \end{cases}$$

where on the interval $[t_i, t_{i+1}]$,

$$\begin{aligned} S_i(x) = {} & \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 \\ & + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6} z_{i+1} \right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6} z_i \right)(t_{i+1} - x) \end{aligned}$$

and where $h_i = t_{i+1} - t_i$. Clearly, $S(x)$ is continuous, since $S_{i-1}(t_i) = S_i(t_i) = y_i$ for $1 \leqq i \leqq n$. It can be shown that $S'_{i-1}(t_i) = S'_i(t_i)$ and $S''_{i-1}(t_i) = S''_i(t_i) = z_i$ for $1 \leqq i \leqq n$. For efficient evaluation, use the nested form of $S_i(x)$, which is

$$S_i(x) = y_i + (x - t_i)\left(B_i + (x - t_i)\left(\frac{z_i}{2} + \frac{1}{6h_i}(x - t_i)(z_{i+1} - z_i) \right) \right)$$

where $B_i = -(h_i/6)z_{i+1} - (h_i/3)z_i + (y_{i+1} - y_i)/h_i$. The coefficients $z_0, z_1, \ldots, z_n$ are found by letting $b_i = (y_{i+1} - y_i)/h_i$, $u_i = 2(h_{i-1} + h_i)$, $v_i = 6(b_i - b_{i-1})$, and then solving

the tridiagonal system of equations

$$\begin{cases} z_0 = 0 \\ h_{i-1}z_{i-1} + u_i z_i + h_i z_{i+1} = v_i & (1 \leqq i \leqq n-1) \\ z_n = 0 \end{cases}$$

This can be done efficiently by using forward substitution:

$$\begin{cases} u_i \leftarrow u_i - \dfrac{h_{i-1}^2}{u_{i-1}} \\ v_i \leftarrow v_i - \dfrac{h_{i-1}v_{i-1}}{u_{i-1}} & (i = 2, 3, \ldots, n-1) \end{cases}$$

and back substitution:

$$\begin{cases} z_{n-1} \leftarrow \dfrac{v_{n-1}}{u_{n-1}} \\ z_i \leftarrow \dfrac{v_i - h_i z_{i+1}}{u_i} & (i = n-2, n-3, \ldots, 1) \end{cases}$$