b. Derive the error term in Theorem 3.9. [*Hint:* Use the same method as in the Lagrange error derivation, Theorem 3.3, defining

$$g(t) = f(t) - H_{2n+1}(t) - \frac{(t-x_0)^2 \cdots (t-x_n)^2}{(x-x_0)^2 \cdots (x-x_n)^2}[f(x) - H_{2n+1}(x)]$$

and using the fact that $g'(t)$ has $(2n+2)$ *distinct* zeros in $[a, b]$.]

12. Let $z_0 = x_0$, $z_1 = x_0$, $z_2 = x_1$, and $z_3 = x_1$. Form the following divided-difference table.

$z_0 = x_0$	$f[z_0] = f(x_0)$			
		$f[z_0, z_1] = f'(x_0)$		
$z_1 = x_0$	$f[z_1] = f(x_0)$		$f[z_0, z_1, z_2]$	
		$f[z_1, z_2]$		$f[z_0, z_1, z_2, z_3]$
$z_2 = x_1$	$f[z_2] = f(x_1)$		$f[z_1, z_2, z_3]$	
		$f[z_2, z_3] = f'(x_1)$		
$z_3 = x_1$	$f[z_3] = f(x_1)$			

Show that the cubic Hermite polynomial $H_3(x)$ can also be written as $f[z_0] + f[z_0, z_1](x - x_0) + f[z_0, z_1, z_2](x - x_0)^2 + f[z_0, z_1, z_2, z_3](x - x_0)^2(x - x_1)$.

3.5 Cubic Spline Interpolation[1]

The previous sections concerned the approximation of arbitrary functions on closed intervals using a single polynomial. However, high-degree polynomials can oscillate erratically, that is, a minor fluctuation over a small portion of the interval can induce large fluctuations over the entire range. We will see a good example of this in Figure 3.14 at the end of this section.

An alternative approach is to divide the approximation interval into a collection of subintervals and construct a (generally) different approximating polynomial on each subinterval. This is called **piecewise-polynomial approximation**.

Piecewise-Polynomial Approximation

The simplest piecewise-polynomial approximation is **piecewise-linear** interpolation, which consists of joining a set of data points

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$$

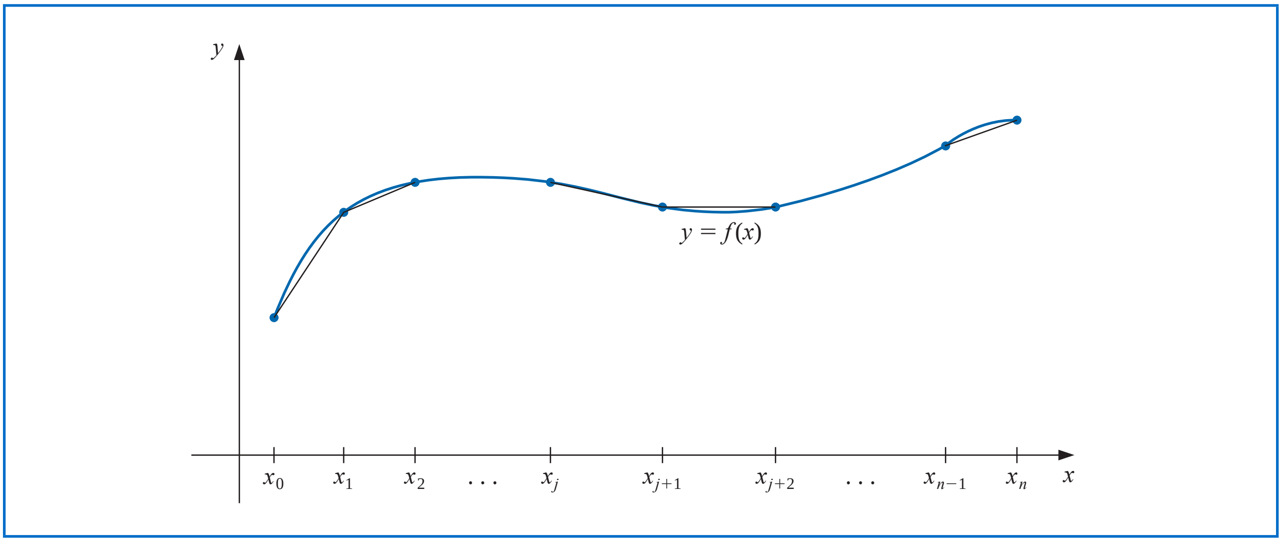
by a series of straight lines, as shown in Figure 3.7.

A disadvantage of linear function approximation is that there is likely no differentiability at the endpoints of the subintervals, which, in a geometrical context, means that the interpolating function is not "smooth." Often it is clear from physical conditions that smoothness is required, so the approximating function must be continuously differentiable.

An alternative procedure is to use a piecewise polynomial of Hermite type. For example, if the values of f and of f' are known at each of the points $x_0 < x_1 < \cdots < x_n$, a cubic Hermite polynomial can be used on each of the subintervals $[x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n]$ to obtain a function that has a continuous derivative on the interval $[x_0, x_n]$.

[1]The proofs of the theorems in this section rely on results in Chapter 6.

Figure 3.7



Isaac Jacob Schoenberg (1903–1990) developed his work on splines during World War II while on leave from the University of Pennsylvania to work at the Army's Ballistic Research Laboratory in Aberdeen, Maryland. His original work involved numerical procedures for solving differential equations. The much broader application of splines to the areas of data fitting and computer-aided geometric design became evident with the widespread availability of computers in the 1960s.

To determine the appropriate Hermite cubic polynomial on a given interval is simply a matter of computing $H_3(x)$ for that interval. The Lagrange interpolating polynomials needed to determine H_3 are of first degree, so this can be accomplished without great difficulty. However, to use Hermite piecewise polynomials for general interpolation, we need to know the derivative of the function being approximated, and this is frequently unavailable.

The remainder of this section considers approximation using piecewise polynomials that require no specific derivative information, except perhaps at the endpoints of the interval on which the function is being approximated.

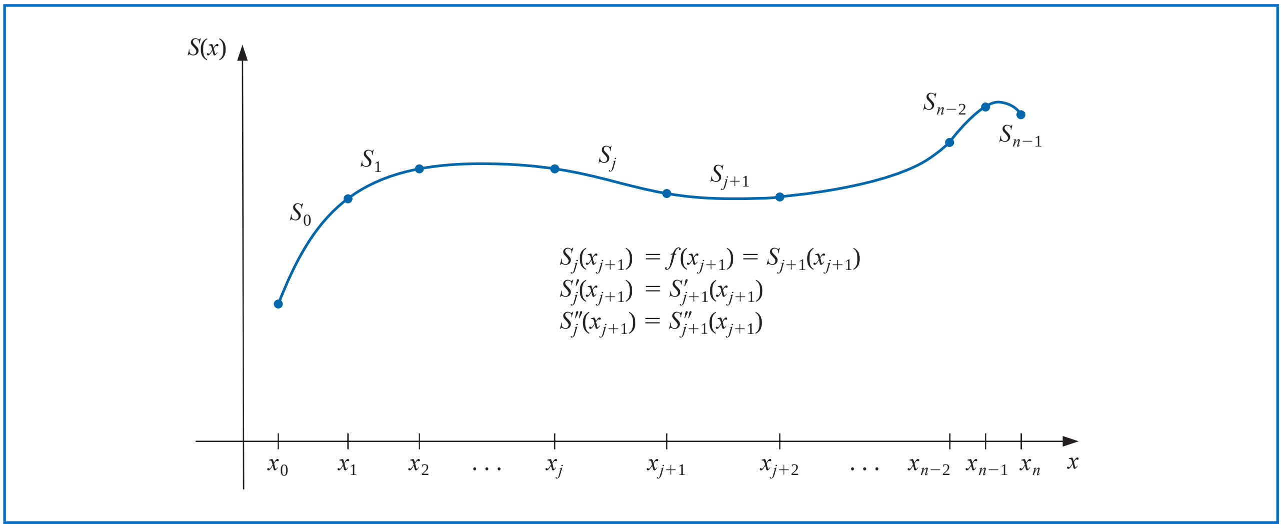
The simplest type of differentiable piecewise-polynomial function on an entire interval $[x_0, x_n]$ is the function obtained by fitting one quadratic polynomial between each successive pair of nodes. This is done by constructing a quadratic on $[x_0, x_1]$ agreeing with the function at x_0 and x_1, another quadratic on $[x_1, x_2]$ agreeing with the function at x_1 and x_2, and so on. A general quadratic polynomial has three arbitrary constants—the constant term, the coefficient of x, and the coefficient of x^2—and only two conditions are required to fit the data at the endpoints of each subinterval. So flexibility exists that permits the quadratics to be chosen so that the interpolant has a continuous derivative on $[x_0, x_n]$. The difficulty arises because we generally need to specify conditions about the derivative of the interpolant at the endpoints x_0 and x_n. There is not a sufficient number of constants to ensure that the conditions will be satisfied. (See Exercise 26.)

The root of the word "spline" is the same as that of splint. It was originally a small strip of wood that could be used to join two boards. Later the word was used to refer to a long flexible strip, generally of metal, that could be used to draw continuous smooth curves by forcing the strip to pass through specified points and tracing along the curve.

Cubic Splines

The most common piecewise-polynomial approximation uses cubic polynomials between each successive pair of nodes and is called **cubic spline interpolation**. A general cubic polynomial involves four constants, so there is sufficient flexibility in the cubic spline procedure to ensure that the interpolant is not only continuously differentiable on the interval, but also has a continuous second derivative. The construction of the cubic spline does not, however, assume that the derivatives of the interpolant agree with those of the function it is approximating, even at the nodes. (See Figure 3.8.)

Figure 3.8



Definition 3.10 Given a function f defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 < \cdots < x_n = b$, a **cubic spline interpolant** S for f is a function that satisfies the following conditions:

(a) $S(x)$ is a cubic polynomial, denoted $S_j(x)$, on the subinterval $[x_j, x_{j+1}]$ for each $j = 0, 1, \ldots, n-1$;

(b) $S_j(x_j) = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$ for each $j = 0, 1, \ldots, n-1$;

(c) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$; (*Implied by* **(b)**.)

(d) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;

(e) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;

(f) One of the following sets of boundary conditions is satisfied:

(i) $S''(x_0) = S''(x_n) = 0$ (**natural** *(or free)* **boundary**);

(ii) $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (**clamped boundary**). ■

A natural spline has no conditions imposed for the direction at its endpoints, so the curve takes the shape of a straight line after it passes through the interpolation points nearest its endpoints. The name derives from the fact that this is the natural shape a flexible strip assumes if forced to pass through specified interpolation points with no additional constraints. (See Figure 3.9.)

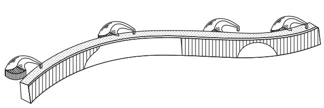


Figure 3.9

Although cubic splines are defined with other boundary conditions, the conditions given in (**f**) are sufficient for our purposes. When the free boundary conditions occur, the spline is called a **natural spline**, and its graph approximates the shape that a long flexible rod would assume if forced to go through the data points $\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$.

In general, clamped boundary conditions lead to more accurate approximations because they include more information about the function. However, for this type of boundary condition to hold, it is necessary to have either the values of the derivative at the endpoints or an accurate approximation to those values.

Example 1 Construct a natural cubic spline that passes through the points $(1, 2)$, $(2, 3)$, and $(3, 5)$.

Solution This spline consists of two cubics. The first for the interval $[1, 2]$, denoted

$$S_0(x) = a_0 + b_0(x-1) + c_0(x-1)^2 + d_0(x-1)^3,$$

and the other for [2, 3], denoted

$$S_1(x) = a_1 + b_1(x-2) + c_1(x-2)^2 + d_1(x-2)^3.$$

There are 8 constants to be determined, which requires 8 conditions. Four conditions come from the fact that the splines must agree with the data at the nodes. Hence

$$2 = f(1) = a_0, \quad 3 = f(2) = a_0 + b_0 + c_0 + d_0, \quad 3 = f(2) = a_1, \quad \text{and}$$
$$5 = f(3) = a_1 + b_1 + c_1 + d_1.$$

Two more come from the fact that $S_0'(2) = S_1'(2)$ and $S_0''(2) = S_1''(2)$. These are

$$S_0'(2) = S_1'(2): \quad b_0 + 2c_0 + 3d_0 = b_1 \qquad \text{and} \qquad S_0''(2) = S_1''(2): \quad 2c_0 + 6d_0 = 2c_1$$

The final two come from the natural boundary conditions:

$$S_0''(1) = 0: \quad 2c_0 = 0 \qquad \text{and} \qquad S_1''(3) = 0: \quad 2c_1 + 6d_1 = 0.$$

Solving this system of equations gives the spline

$$S(x) = \begin{cases} 2 + \frac{3}{4}(x-1) + \frac{1}{4}(x-1)^3, & \text{for } x \in [1,2] \\ 3 + \frac{3}{2}(x-2) + \frac{3}{4}(x-2)^2 - \frac{1}{4}(x-2)^3, & \text{for } x \in [2,3] \end{cases}$$

■

Construction of a Cubic Spline

As the preceding example demonstrates, a spline defined on an interval that is divided into n subintervals will require determining $4n$ constants. To construct the cubic spline interpolant for a given function f, the conditions in the definition are applied to the cubic polynomials

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3,$$

Clamping a spline indicates that the ends of the flexible strip are fixed so that it is forced to take a specific direction at each of its endpoints. This is important, for example, when two spline functions should match at their endpoints. This is done mathematically by specifying the values of the derivative of the curve at the endpoints of the spline.

for each $j = 0, 1, \ldots, n-1$. Since $S_j(x_j) = a_j = f(x_j)$, condition **(c)** can be applied to obtain

$$a_{j+1} = S_{j+1}(x_{j+1}) = S_j(x_{j+1}) = a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3,$$

for each $j = 0, 1, \ldots, n-2$.

The terms $x_{j+1} - x_j$ are used repeatedly in this development, so it is convenient to introduce the simpler notation

$$h_j = x_{j+1} - x_j,$$

for each $j = 0, 1, \ldots, n-1$. If we also define $a_n = f(x_n)$, then the equation

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \tag{3.15}$$

holds for each $j = 0, 1, \ldots, n-1$.

In a similar manner, define $b_n = S'(x_n)$ and observe that

$$S_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$$

implies $S_j'(x_j) = b_j$, for each $j = 0, 1, \ldots, n - 1$. Applying condition **(d)** gives

$$b_{j+1} = b_j + 2c_jh_j + 3d_jh_j^2, \tag{3.16}$$

for each $j = 0, 1, \ldots, n - 1$.

Another relationship between the coefficients of S_j is obtained by defining $c_n = S''(x_n)/2$ and applying condition **(e)**. Then, for each $j = 0, 1, \ldots, n - 1$,

$$c_{j+1} = c_j + 3d_jh_j. \tag{3.17}$$

Solving for d_j in Eq. (3.17) and substituting this value into Eqs. (3.15) and (3.16) gives, for each $j = 0, 1, \ldots, n - 1$, the new equations

$$a_{j+1} = a_j + b_jh_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) \tag{3.18}$$

and

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}). \tag{3.19}$$

The final relationship involving the coefficients is obtained by solving the appropriate equation in the form of equation (3.18), first for b_j,

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}), \tag{3.20}$$

and then, with a reduction of the index, for b_{j-1}. This gives

$$b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j).$$

Substituting these values into the equation derived from Eq. (3.19), with the index reduced by one, gives the linear system of equations

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_jc_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}), \tag{3.21}$$

for each $j = 1, 2, \ldots, n - 1$. This system involves only the $\{c_j\}_{j=0}^n$ as unknowns. The values of $\{h_j\}_{j=0}^{n-1}$ and $\{a_j\}_{j=0}^n$ are given, respectively, by the spacing of the nodes $\{x_j\}_{j=0}^n$ and the values of f at the nodes. So once the values of $\{c_j\}_{j=0}^n$ are determined, it is a simple matter to find the remainder of the constants $\{b_j\}_{j=0}^{n-1}$ from Eq. (3.20) and $\{d_j\}_{j=0}^{n-1}$ from Eq. (3.17). Then we can construct the cubic polynomials $\{S_j(x)\}_{j=0}^{n-1}$.

The major question that arises in connection with this construction is whether the values of $\{c_j\}_{j=0}^n$ can be found using the system of equations given in (3.21) and, if so, whether these values are unique. The following theorems indicate that this is the case when either of the boundary conditions given in part **(f)** of the definition are imposed. The proofs of these theorems require material from linear algebra, which is discussed in Chapter 6.

Natural Splines

Theorem 3.11 If f is defined at $a = x_0 < x_1 < \cdots < x_n = b$, then f has a unique natural spline interpolant S on the nodes $x_0, x_1, \ldots, x_n$; that is, a spline interpolant that satisfies the natural boundary conditions $S''(a) = 0$ and $S''(b) = 0$. ■

Proof The boundary conditions in this case imply that $c_n = S''(x_n)/2 = 0$ and that

$$0 = S''(x_0) = 2c_0 + 6d_0(x_0 - x_0),$$

so $c_0 = 0$. The two equations $c_0 = 0$ and $c_n = 0$ together with the equations in (3.21) produce a linear system described by the vector equation $A\mathbf{x} = \mathbf{b}$, where A is the $(n+1) \times (n+1)$ matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & \ddots & & & \vdots \\ 0 & h_1 & 2(h_1 + h_2) & h_2 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & 0 & 1 \end{bmatrix},$$

and $\mathbf{b}$ and $\mathbf{x}$ are the vectors

$$\mathbf{b} = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}.$$

The matrix A is strictly diagonally dominant, that is, in each row the magnitude of the diagonal entry exceeds the sum of the magnitudes of all the other entries in the row. A linear system with a matrix of this form will be shown by Theorem 6.21 in Section 6.6 to have a unique solution for $c_0, c_1, \ldots, c_n$. ■ ■ ■

The solution to the cubic spline problem with the boundary conditions $S''(x_0) = S''(x_n) = 0$ can be obtained by applying Algorithm 3.4.

ALGORITHM
3.4

Natural Cubic Spline

To construct the cubic spline interpolant S for the function f, defined at the numbers $x_0 < x_1 < \cdots < x_n$, satisfying $S''(x_0) = S''(x_n) = 0$:

INPUT $n; x_0, x_1, \ldots, x_n; a_0 = f(x_0), a_1 = f(x_1), \ldots, a_n = f(x_n)$.

OUTPUT a_j, b_j, c_j, d_j for $j = 0, 1, \ldots, n-1$.

(*Note:* $S(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$ *for* $x_j \le x \le x_{j+1}$.)

Step 1 For $i = 0, 1, \ldots, n-1$ set $h_i = x_{i+1} - x_i$.

Step 2 For $i = 1, 2, \ldots, n-1$ set

$$\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}).$$

Step 3 Set $l_0 = 1$; (*Steps 3, 4, 5, and part of Step 6 solve a tridiagonal linear system using a method described in Algorithm 6.7.*)

$\mu_0 = 0$;
$z_0 = 0$.

Step 4 For $i = 1, 2, \ldots, n-1$
set $l_i = 2(x_{i+1} - x_{i-1}) - h_{i-1}\mu_{i-1}$;
$\mu_i = h_i/l_i$;
$z_i = (\alpha_i - h_{i-1}z_{i-1})/l_i$.

Step 5 Set $l_n = 1$;
$z_n = 0$;
$c_n = 0$.

Step 6 For $j = n-1, n-2, \ldots, 0$
set $c_j = z_j - \mu_j c_{j+1}$;
$b_j = (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3$;
$d_j = (c_{j+1} - c_j)/(3h_j)$.

Step 7 OUTPUT $(a_j, b_j, c_j, d_j$ for $j = 0, 1, \ldots, n-1)$;
STOP. ■

Example 2 At the beginning of Chapter 3 we gave some Taylor polynomials to approximate the exponential $f(x) = e^x$. Use the data points $(0, 1)$, $(1, e)$, $(2, e^2)$, and $(3, e^3)$ to form a natural spline $S(x)$ that approximates $f(x) = e^x$.

Solution We have $n = 3$, $h_0 = h_1 = h_2 = 1$, $a_0 = 1$, $a_1 = e$, $a_2 = e^2$, and $a_3 = e^3$. So the matrix A and the vectors $\mathbf{b}$ and $\mathbf{x}$ given in Theorem 3.11 have the forms

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 3(e^2 - 2e + 1) \\ 3(e^3 - 2e^2 + e) \\ 0 \end{bmatrix}, \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}.$$

The vector-matrix equation $A\mathbf{x} = \mathbf{b}$ is equivalent to the system of equations

$$\begin{aligned} c_0 &= 0, \\ c_0 + 4c_1 + c_2 &= 3(e^2 - 2e + 1), \\ c_1 + 4c_2 + c_3 &= 3(e^3 - 2e^2 + e), \\ c_3 &= 0. \end{aligned}$$

This system has the solution $c_0 = c_3 = 0$, and to 5 decimal places,

$$c_1 = \frac{1}{5}(-e^3 + 6e^2 - 9e + 4) \approx 0.75685, \quad \text{and} \quad c_2 = \frac{1}{5}(4e^3 - 9e^2 + 6e - 1) \approx 5.83007.$$

Solving for the remaining constants gives

$$b_0 = \frac{1}{h_0}(a_1 - a_0) - \frac{h_0}{3}(c_1 + 2c_0)$$

$$= (e-1) - \frac{1}{15}(-e^3 + 6e^2 - 9e + 4) \approx 1.46600,$$

$$b_1 = \frac{1}{h_1}(a_2 - a_1) - \frac{h_1}{3}(c_2 + 2c_1)$$

$$= (e^2 - e) - \frac{1}{15}(2e^3 + 3e^2 - 12e + 7) \approx 2.22285,$$

$$b_2 = \frac{1}{h_2}(a_3 - a_2) - \frac{h_2}{3}(c_3 + 2c_2)$$

$$= (e^3 - e^2) - \frac{1}{15}(8e^3 - 18e^2 + 12e - 2) \approx 8.80977,$$

$$d_0 = \frac{1}{3h_0}(c_1 - c_0) = \frac{1}{15}(-e^3 + 6e^2 - 9e + 4) \approx 0.25228,$$

$$d_1 = \frac{1}{3h_1}(c_2 - c_1) = \frac{1}{3}(e^3 - 3e^2 + 3e - 1) \approx 1.69107,$$

and

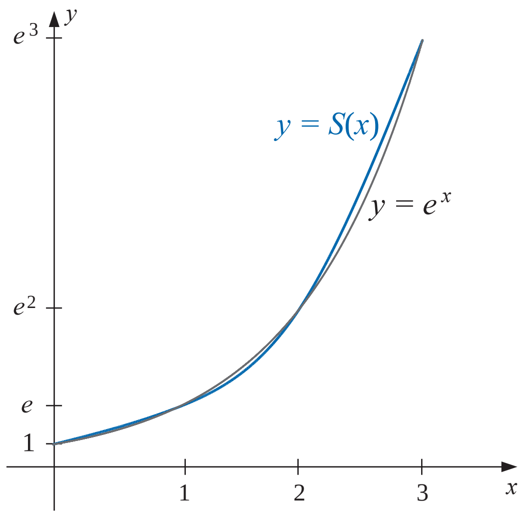
$$d_2 = \frac{1}{3h_2}(c_3 - c_1) = \frac{1}{15}(-4e^3 + 9e^2 - 6e + 1) \approx -1.94336.$$

The natural cubic spine is described piecewise by

$$S(x) = \begin{cases} 1 + 1.46600x + 0.25228x^3, & \text{for } x \in [0,1], \\ 2.71828 + 2.22285(x-1) + 0.75685(x-1)^2 + 1.69107(x-1)^3, & \text{for } x \in [1,2], \\ 7.38906 + 8.80977(x-2) + 5.83007(x-2)^2 - 1.94336(x-2)^3, & \text{for } x \in [2,3]. \end{cases}$$

The spline and its agreement with $f(x) = e^x$ are shown in Figure 3.10. ■

Figure 3.10



The *NumericalAnalysis* package can be used to create a cubic spline in a manner similar to other constructions in this chapter. However, the *CurveFitting* Package in Maple can also be used, and since this has not been discussed previously we will use it to create the natural spline in Example 2. First we load the package with the command

$with(CurveFitting)$

and define the function being approximated with

$f := x \to e^x$

To create a spline we need to specify the nodes, variable, the degree, and the natural endpoints. This is done with

$sn := t \to Spline([[0., 1.0], [1.0, f(1.0)], [2.0, f(2.0)], [3.0, f(3.0)]], t, degree = 3,$
$endpoints = \text{'natural'})$

Maple returns

$$t \to CurveFitting\text{:-}Spline([[0., 1.0], [1.0, f(1.0)], [2.0, f(2.0)], [3.0, f(3.0)]], t, degree = 3, endpoints = \text{'natural'})$$

The form of the natural spline is seen with the command

$sn(t)$

which produces

$$\begin{cases} 1. + 1.465998t^2 + 0.2522848t^3 & t < 1.0 \\ 0.495432 + 2.22285t + 0.756853(t-1.0)^2 + 1.691071(t-1.0)^3 & t < 2.0 \\ -10.230483 + 8.809770t + 5.830067(t-2.0)^2 - 1.943356(t-2.0)^3 & otherwise \end{cases}$$

Once we have determined a spline approximation for a function we can use it to approximate other properties of the function. The next illustration involves the integral of the spline we found in the previous example.

Illustration To approximate the integral of $f(x) = e^x$ on $[0, 3]$, which has the value

$$\int_0^3 e^x\,dx = e^3 - 1 \approx 20.08553692 - 1 = 19.08553692,$$

we can piecewise integrate the spline that approximates f on this integral. This gives

$$\begin{aligned}\int_0^3 S(x) = &\int_0^1 1 + 1.46600x + 0.25228x^3\,dx \\ &+ \int_1^2 2.71828 + 2.22285(x-1) + 0.75685(x-1)^2 + 1.69107(x-1)^3\,dx \\ &+ \int_2^3 7.38906 + 8.80977(x-2) + 5.83007(x-2)^2 - 1.94336(x-2)^3\,dx.\end{aligned}$$

Integrating and collecting values from like powers gives

$$\int_0^3 S(x) = \left[x + 1.46600\frac{x^2}{2} + 0.25228\frac{x^4}{4}\right]_0^1$$

$$+ \left[2.71828(x-1) + 2.22285\frac{(x-1)^2}{2} + 0.75685\frac{(x-1)^3}{3} + 1.69107\frac{(x-1)^4}{4}\right]_1^2$$

$$+ \left[7.38906(x-2) + 8.80977\frac{(x-2)^2}{2} + 5.83007\frac{(x-2)^3}{3} - 1.94336\frac{(x-2)^4}{4}\right]_2^3$$

$$= (1 + 2.71828 + 7.38906) + \frac{1}{2}(1.46600 + 2.22285 + 8.80977)$$

$$+ \frac{1}{3}(0.75685 + 5.83007) + \frac{1}{4}(0.25228 + 1.69107 - 1.94336)$$

$$= 19.55229.$$

Because the nodes are equally spaced in this example the integral approximation is simply

$$\int_0^3 S(x)\,dx = (a_0+a_1+a_2)+\frac{1}{2}(b_0+b_1+b_2)+\frac{1}{3}(c_0+c_1+c_2)+\frac{1}{4}(d_0+d_1+d_2). \quad (3.22)$$

□

If we create the natural spline using Maple as described after Example 2, we can then use Maple's integration command to find the value in the Illustration. Simply enter

$int(sn(t), t = 0\,..\,3)$

19.55228648

Clamped Splines

Example 3 In Example 1 we found a natural spline S that passes through the points $(1,2)$, $(2,3)$, and $(3,5)$. Construct a clamped spline s through these points that has $s'(1) = 2$ and $s'(3) = 1$.

Solution Let

$$s_0(x) = a_0 + b_0(x-1) + c_0(x-1)^2 + d_0(x-1)^3,$$

be the cubic on $[1,2]$ and the cubic on $[2,3]$ be

$$s_1(x) = a_1 + b_1(x-2) + c_1(x-2)^2 + d_1(x-2)^3.$$

Then most of the conditions to determine the 8 constants are the same as those in Example 1. That is,

$$2 = f(1) = a_0, \quad 3 = f(2) = a_0 + b_0 + c_0 + d_0, \quad 3 = f(2) = a_1, \quad \text{and}$$
$$5 = f(3) = a_1 + b_1 + c_1 + d_1.$$

$$s_0'(2) = s_1'(2): \quad b_0 + 2c_0 + 3d_0 = b_1 \quad \text{and} \quad s_0''(2) = s_1''(2): \quad 2c_0 + 6d_0 = 2c_1$$