3.4 Cubic Spline Interpolation

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 \([a, b]\).]

9. Let \( z_0 = x_0, z_1 = x_0, z_2 = x_1, \) and \( z_3 = x_1 \). Form the following divided-difference table.

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

Show that the cubic Hermite polynomial \( H_3(x) \) can also be written as \( f[z_0] + f[z_0, z_1](x - x_0) + f[z_0, z_1, z_2](x - x_0)^2 + f[z_0, z_1, z_2, z_3](x - x_0)^3(x - x_1) \).

### 3.4 Cubic Spline Interpolation

The previous sections concerned the approximation of arbitrary functions on closed intervals by the use of polynomials. However, the oscillatory nature of high-degree polynomials and the property that a fluctuation over a small portion of the interval can induce large fluctuations over the entire range restricts their use. We will see a good example of this in Figure 3.12 at the end of this section.

An alternative approach is to divide the interval into a collection of subintervals and construct a (generally) different approximating polynomial on each subinterval. Approximation by functions of this type is called 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 on page 142.

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 \([x_0, x_n]\).

*The proofs of the theorems in this section rely on results in Chapter 6.*
determine the appropriate Hermite cubic polynomial on a given interval is simply a matter of computing \( H_3(x) \) for that interval. Since the Lagrange interpolating polynomials needed to determine \( H_3 \) are of first degree, 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, which is frequently unavailable.

The remainder of this section considers approximation using piecewise polynomials that require no 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 one 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. Since 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, flexibility exists that permits the quadratic to be chosen so that the interpolant has a continuous derivative on \([x_0, x_n]\).

The difficulty arises when there is a 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 22.)

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.)


**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. \( S(x_j) = f(x_j) \) for each \( j = 0, 1, \ldots, n \);

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

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

e. \( S''(x_0) = S''(x_n) = 0 \) (free or natural boundary);

f. One of the following sets of boundary conditions is satisfied:
   
   (i) \( S'(x_0) = S'(x_n) = 0 \) (free or natural boundary);
   
   (ii) \( S'(x_0) = f'(x_0) \) and \( S'(x_n) = f'(x_n) \) (clamped boundary).

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 since 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.

To construct the cubic spline interpolant for a given function \( f \), the conditions in the definition are applied to the cubic polynomials.
\[ S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \]
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 \( j = 0, 1, \ldots, n - 2. \)

Since the terms \( x_{j+1} - x_j \) are used repeatedly in this development, 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 \( a_0 = f(x_0), \) then the equation
\[ a_{j+1} = a_j + b_jh_j + c_j h_j^2 + d_j h_j^3 \] (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
\[ b_{j+1} = b_j + 2c_jh_j + 3d_jh_j^2, \] (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 \( j = 0, 1, \ldots, n - 1, \)
\[ c_{j+1} = c_j + 3d_jh_j. \] (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
\[ a_{j+1} = a_j + b_jh_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) \] (3.18)
and
\[ b_{j+1} = b_j + h_j(c_j + c_{j+1}). \] (3.19)

The final relationship involving the coefficients is obtained by solving the appropriate equation in the form of equation (3.18), first for \( b_j, \)
\[ b_j = \frac{1}{h_j^2}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}), \] (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

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

for each \( j = 1, 2, \ldots, n-1 \). This system involves only the \( \{c_j\}_{j=0}^{n-1} \) as unknowns since the values of \( \{h_j\}_{j=0}^{n-1} \) and \( \{a_j\}_{j=0}^{n-1} \) are given, respectively, by the spacing of the nodes \( \{x_j\}_{j=0}^{n-1} \) and the values of \( f \) at the nodes.

Note that once the values of \( \{c_j\}_{j=0}^{n-1} \) 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), and to 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 \( \{c_j\}_{j=0}^{n-1} \) 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.

**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 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 \( Ax = b \), where \( A \) is the \((n+1) \times (n+1)\) matrix

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

and \( b \) and \( x \) are the vectors
The matrix $A$ is strictly diagonally dominant, so it satisfies the hypotheses of Theorem 6.19 in Section 6.6. Therefore, the linear system has 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**

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 $x \leq x_j \leq x_{j+1}$.)

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

**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 5.7.)

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

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

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

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

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

A result similar to Theorem 3.11 holds in the case of clamped boundary conditions.

**Theorem 3.12**

If \(f\) is defined at \(a = x_0 < x_1 < \cdots < x_n = b\) and differentiable at \(a\) and \(b\), then \(f\) has a unique clamped spline interpolant \(S\) on the nodes \(x_0, x_1, \ldots, x_n\), that is, a spline interpolant that satisfies the boundary conditions \(S'(a) = f'(a)\) and \(S'(b) = f'(b)\).

**Proof** Since \(f'(a) = S'(a) = S'(x_0) = b_0, \text{ Eq. (3.20)} with \(j = 0\) implies

\[
f'(a) = \frac{1}{h_0}(a_1 - a_0) - \frac{h_0}{3}(2c_0 + c_1).
\]

Consequently,

\[
2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a).
\]

Similarly,

\[
f'(b) = b_n = b_{n-1} + h_{n-1}(c_{n-1} + c_n),
\]

so Eq. (3.20) with \(j = n - 1\) implies that

\[
f'(b) = \frac{a_n - a_{n-1}}{h_{n-1}} - \frac{h_{n-1}}{3}(2c_{n-1} + c_n) + h_{n-1}(c_{n-1} + c_n)
= \frac{a_n - a_{n-1}}{h_{n-1}} + \frac{h_{n-1}}{3}(c_{n-1} + 2c_n),
\]

and

\[
h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1}).
\]

Equations (3.21) together with the equations

\[
2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a)
\]

and

\[
h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1})
\]

determine the linear system \(Ax = b\), where
The matrix $A$ is strictly diagonally dominant, so it satisfies the conditions of Theorem 6.19. Therefore, the linear system has a unique solution for $c_0, c_1, \ldots, c_n$.

The solution to the cubic spline problem with the boundary conditions $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ can be obtained by applying Algorithm 3.5.

**Clamped 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 $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$:

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

**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 $x_j \leq x \leq x_{j+1}$.)

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

**Step 2** Set $a_0 = \frac{3(a_1 - a_0)}{h_0} - 3FPO$;

$a_n = 3FPN - 3(a_n - a_{n-1})/h_{n-1}$.

**Step 3** For $i = 1, 2, \ldots, n - 1$

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

**Step 4** Set $l_0 = 2h_0$; (Steps 4,5,6, and part of Step 7 solve a tridiagonal linear system using a method described in Algorithm 6.7.)

$\mu_0 = 0.5$;

$z_0 = a_0/l_0$. 

3.4 Cubic Spline Interpolation

**Step 5** For $i = 1, 2, \ldots, n - 1$
- set $l_i = 2(x_{i+1} - x_{i-1}) - h_{i-1} \mu_{i-1}$;
- $\mu_i = h_i / l_i$;
- $z_i = (a_i - h_{i-1} z_{i-1}) / l_i$.

**Step 6** Set $l_n = h_{n-1}(2 - \mu_{n-1})$;
- $z_n = (a_n - h_{n-1} z_{n-1}) / l_n$;
- $c_n = z_n$.

**Step 7** 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 8** OUTPUT $(a_j, b_j, c_j, d_j$ for $j = 0, 1, \ldots, n - 1)$; STOP.

**EXAMPLE 1**
Figure 3.9 shows a ruddy duck in flight. To approximate the top profile of the duck, we have chosen points along the curve through which we want the approximating curve to pass. Table 3.15 lists the coordinates of 21 data points relative to the superimposed coordinate system shown in Figure 3.10 on page 150. Notice that more points are used when the curve is changing rapidly than when it is changing more slowly.

![Figure 3.9](image)

<table>
<thead>
<tr>
<th>Table 3.15</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
</tr>
<tr>
<td>$f(x)$</td>
</tr>
</tbody>
</table>

Using Algorithm 3.4 to generate the free cubic spline for this data produces the coefficients shown in Table 3.16. This spline curve is nearly identical to the profile, as shown in Figure 3.11.
### Table 3.16

<table>
<thead>
<tr>
<th>j</th>
<th>( x_j )</th>
<th>( a_j )</th>
<th>( b_j )</th>
<th>( c_j )</th>
<th>( d_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.9</td>
<td>1.3</td>
<td>5.40</td>
<td>0.00</td>
<td>-0.25</td>
</tr>
<tr>
<td>1</td>
<td>1.3</td>
<td>1.5</td>
<td>0.42</td>
<td>-0.30</td>
<td>0.95</td>
</tr>
<tr>
<td>2</td>
<td>1.9</td>
<td>1.85</td>
<td>1.09</td>
<td>1.41</td>
<td>-2.96</td>
</tr>
<tr>
<td>3</td>
<td>2.1</td>
<td>2.1</td>
<td>1.29</td>
<td>-0.37</td>
<td>-0.45</td>
</tr>
<tr>
<td>4</td>
<td>2.6</td>
<td>2.6</td>
<td>0.59</td>
<td>-1.04</td>
<td>0.45</td>
</tr>
<tr>
<td>5</td>
<td>3.0</td>
<td>2.7</td>
<td>-0.02</td>
<td>-0.50</td>
<td>0.17</td>
</tr>
<tr>
<td>6</td>
<td>3.9</td>
<td>2.4</td>
<td>-0.50</td>
<td>-0.03</td>
<td>0.08</td>
</tr>
<tr>
<td>7</td>
<td>4.4</td>
<td>2.15</td>
<td>-0.48</td>
<td>0.08</td>
<td>1.31</td>
</tr>
<tr>
<td>8</td>
<td>4.7</td>
<td>2.05</td>
<td>-0.07</td>
<td>1.27</td>
<td>-1.58</td>
</tr>
<tr>
<td>9</td>
<td>5.0</td>
<td>2.1</td>
<td>0.26</td>
<td>-0.16</td>
<td>0.04</td>
</tr>
<tr>
<td>10</td>
<td>6.0</td>
<td>2.25</td>
<td>0.08</td>
<td>-0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>11</td>
<td>7.0</td>
<td>2.3</td>
<td>0.01</td>
<td>-0.04</td>
<td>-0.02</td>
</tr>
<tr>
<td>12</td>
<td>8.0</td>
<td>2.25</td>
<td>-0.14</td>
<td>-0.11</td>
<td>0.02</td>
</tr>
<tr>
<td>13</td>
<td>9.2</td>
<td>1.95</td>
<td>-0.34</td>
<td>-0.05</td>
<td>-0.01</td>
</tr>
<tr>
<td>14</td>
<td>10.5</td>
<td>1.4</td>
<td>-0.53</td>
<td>-0.10</td>
<td>-0.02</td>
</tr>
<tr>
<td>15</td>
<td>11.3</td>
<td>0.9</td>
<td>-0.73</td>
<td>-0.15</td>
<td>1.21</td>
</tr>
<tr>
<td>16</td>
<td>11.6</td>
<td>0.7</td>
<td>-0.49</td>
<td>0.94</td>
<td>-0.84</td>
</tr>
<tr>
<td>17</td>
<td>12.0</td>
<td>0.6</td>
<td>-0.14</td>
<td>-0.05</td>
<td>0.04</td>
</tr>
<tr>
<td>18</td>
<td>12.6</td>
<td>0.5</td>
<td>-0.18</td>
<td>0.00</td>
<td>-0.45</td>
</tr>
<tr>
<td>19</td>
<td>13.0</td>
<td>0.4</td>
<td>-0.39</td>
<td>-0.54</td>
<td>0.60</td>
</tr>
<tr>
<td>20</td>
<td>13.3</td>
<td>0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For comparison purposes, Figure 3.12 gives an illustration of the curve that is generated using a Lagrange interpolating polynomial to fit the data given in Table 3.15. This produces a very strange illustration of the back of a duck, in flight or otherwise. The interpolating polynomial in this case is of degree 20 and oscillates wildly.
To use a clamped spline to approximate this curve we would need derivative approximations for the endpoints. Even if these approximations were available, we could expect little improvement because of the close agreement of the free cubic spline to the curve of the top profile.

Constructing a cubic spline to approximate the lower profile of the ruddy duck would be more difficult since the curve for this portion cannot be expressed as a function of $x$, and at certain points the curve does not appear to be smooth. These problems can be resolved by using separate splines to represent various portions of the curve, but a more effective approach to curves of this type is considered in the next section.

The clamped boundary conditions are generally preferred when approximating functions by cubic splines, so the derivative of the function must be estimated at the endpoints of the interval. When the nodes are equally spaced near both endpoints, approximations can be obtained by using Eq. (4.7) or any of the other appropriate formulas given in Sections 4.1 and 4.2. When the nodes are unequally spaced, the problem is considerably more difficult.

To conclude this section, we list an error-bound formula for the cubic spline with clamped boundary conditions. The proof of this result can be found in [Schul, pp. 57–58].

**Theorem 3.13** Let $f \in C^4[a, b]$ with $\max_{a \leq x \leq b} |f^{(4)}(x)| = M$. If $S$ is the unique clamped cubic spline interpolant to $f$ with respect to the nodes $a = x_0 < x_1 < \cdots < x_n = b$, then

$$\max_{a \leq x \leq b} |f(x) - S(x)| \leq \frac{5M}{384} \max_{0 \leq j \leq n-1} (x_{j+1} - x_j)^4.$$  

A fourth-order error-bound result also holds in the case of free boundary conditions, but it is more difficult to express. (See [BD, pp. 827–835].)

The free boundary conditions will generally give less accurate results than the clamped conditions near the ends of the interval $[x_0, x_n]$ unless the function $f$ happens to nearly satisfy $f''(x_0) = f''(x_n) = 0$. An alternative to the free boundary condition that does not require knowledge of the derivative of $f$ is the not-a-knot condition, (see [Deb, pp. 55–56]). This condition requires that $S'''(x)$ be continuous at $x_1$ and at $x_{n-1}$.

**Exercise Set 3.4**

1. Determine the free cubic spline $S$ that interpolates the data $f(0) = 0$, $f(1) = 1$, and $f(2) = 2$.
2. Determine the clamped cubic spline $s$ that interpolates the data $f(0) = 0$, $f(1) = 1$, $f(2) = 2$ and satisfies $s'(0) = s'(2) = 1$.
3. Construct the free cubic spline for the following data:
   - **a.**
     | $x$  | $f(x)$  |
     |-----|--------|
     | 8.3 | 17.56492 |
     | 8.6 | 18.50515 |
   - **b.**
     | $x$  | $f(x)$  |
     |-----|--------|
     | 0.8 | 0.22363362 |
     | 1.0 | 0.65809197 |