b. Repeat (a) with the Hermite interpolating polynomial of degree at most 5, using $x_0 = 1$, $x_1 = 1.05$, and $x_2 = 1.07$.

5. Use the error formula and MATLAB to find a bound for the errors in the approximations of $f(x)$ in (a) and (c) of Exercise 2.

6. The following table lists data for the function described by $f(x) = e^{0.1x^2}$. Approximate $f(1.25)$ by using $H_5(1.25)$ and $H_3(1.25)$, where H_5 uses the nodes $x_0 = 1$, $x_1 = 2$, and $x_2 = 3$ and H_3 uses the nodes $\bar{x}_0 = 1$ and $\bar{x}_1 = 1.5$. Find error bounds for these approximations.

x	$f(x) = e^{0.1x^2}$	$f'(x) = 0.2xe^{0.1x^2}$
$x_0 = \bar{x}_0 = 1$	1.105170918	0.2210341836
$\bar{x}_1 = 1.5$	1.252322716	0.3756968148
$x_1 = 2$	1.491824698	0.5967298792
$x_2 = 3$	2.459603111	1.475761867

7. A car traveling along a straight road is clocked at a number of points. The data from the observations are given in the following table, where the time is in seconds, the distance is in feet, and the speed is in feet per second.

Time	0	3	5	8	13
Distance	0	225	383	623	993
Speed	75	77	80	74	72

a. Use a Hermite polynomial to predict the position of the car and its speed when $t = 10$ s.

b. Use the derivative of the Hermite polynomial to determine whether the car ever exceeds a 55-mi/h speed limit on the road. If so, what is the first time the car exceeds this speed?

c. What is the predicted maximum speed for the car?

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

$$\begin{array}{llllll}
z_0 = x_0 & f[z_0] = f(x_0) & & & \\
 & & f[z_0, z_1] = f'(x_0) & & \\
z_1 = x_0 & f[z_1] = f(x_0) & & f[z_0, z_1, z_2] & \\
 & & f[z_1, z_2] & & f[z_0, z_1, z_2, z_3] \\
z_2 = x_1 & f[z_2] = f(x_1) & & f[z_1, z_2, z_3] & \\
 & & f[z_2, z_3] = f'(x_1) & & \\
z_3 = x_1 & f[z_3] = f(x_1) & & &
\end{array}$$

Show if

$$P(x) = f[z_0] + f[z_0, z_1](x - x_0) + f[z_0, z_1, z_2](x - x_0)^2 + f[z_0, z_1, z_2, z_3](x - x_0)^2(x - x_1),$$

then

$$P(x_0) = f(x_0), \ P(x_1) = f(x_1), \ P'(x_0) = f'(x_0), \text{ and } P'(x_1) = f'(x_1),$$

which implies that $P(x) \equiv H_3(x)$.

3.5 Spline Interpolation

The previous sections use polynomials to approximate arbitrary functions. However, relatively high-degree polynomials are needed for accurate approximation and these have some serious disadvantages. They can have an oscillatory nature, and a fluctuation over a small portion of the interval can induce large fluctuations over the entire range. We will see an example of this later in this section.

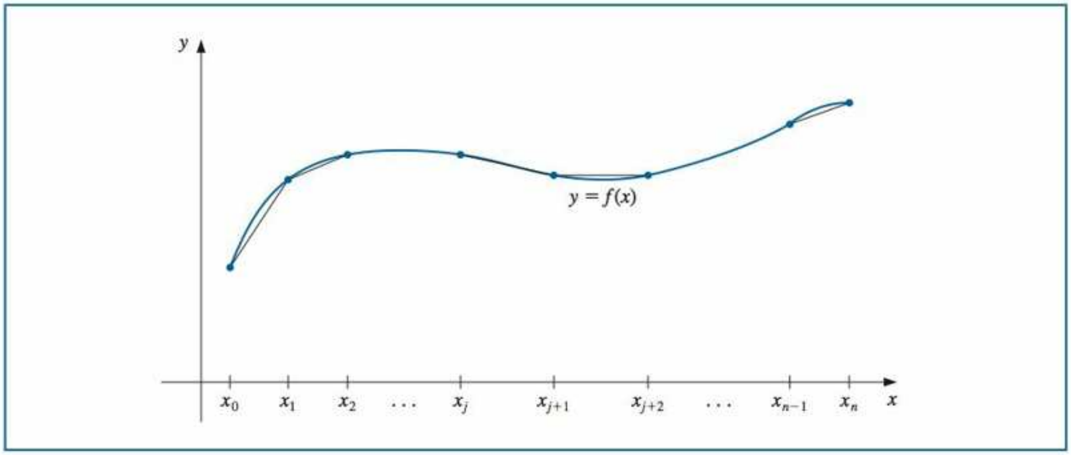
An alternative approach is to divide the interval into a collection of subintervals and construct a different approximating polynomial on each subinterval. This is called **piecewise polynomial approximation.**

Piecewise-Polynomial Approximation

The simplest piecewise polynomial approximation joins the data points $(x_0, f(x_0))$, $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$ by a series of straight lines, such as those shown in Figure 3.7.

A disadvantage of linear approximation is that the approximation is generally not differentiable at the endpoints of the subintervals, so the interpolating function is not "smooth" at these points. It is often clear from physical conditions that smoothness is required, and the approximating function must be continuously differentiable.

Figure 3.7



One remedy for this problem is to use a piecewise polynomial of Hermite type. For example, if the values of f and 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 an approximating function that has a continuous derivative on the interval $[x_0, x_n]$. To determine the appropriate cubic Hermite polynomial on a given interval, we simply compute the function $H_3(x)$ for that interval.

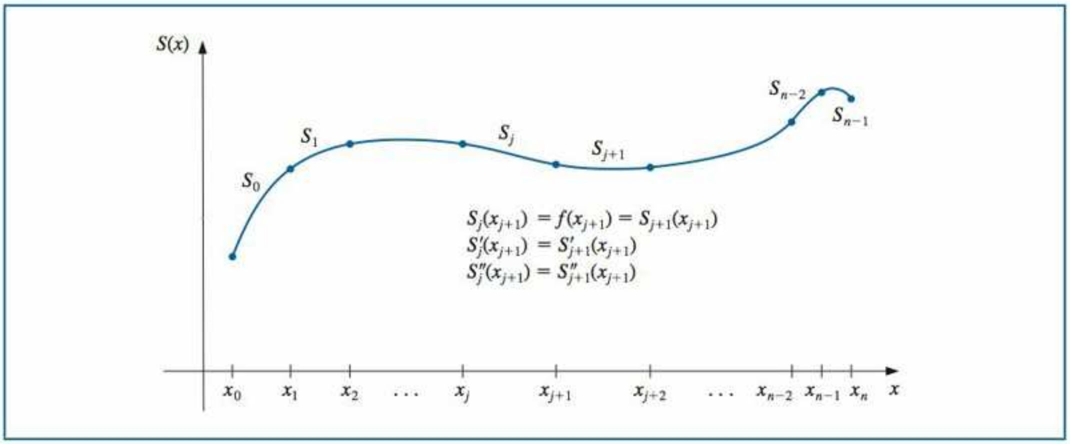
The Hermite polynomials are commonly used in application problems to study the motion of particles in space. The difficulty with using Hermite piecewise polynomials for general interpolation problems concerns the need to know the derivative of the function being approximated. The remainder of this section considers approximations using piecewise polynomials that require no derivative information, except perhaps at the endpoints of the interval on which the function is being approximated.

Isaac Jacob Schoenberg (1903–1990) developed his work on splines during World War II at the Army's Ballistic Research Laboratory in Aberdeen, Maryland, while on leave from the University of Pennsylvania. 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.

Cubic Splines

The most common piecewise-polynomial approximation uses cubic polynomials between pairs 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 has two continuous derivatives on the interval. The derivatives of the cubic spline do not, in general, however, agree with the derivatives of the function, even at the nodes. (See Figure 3.8.)

Figure 3.8



$S(x)$
S_0
S_1
S_j
S_{j+1}
S_{n-2}
S_{n-1}
$S_j(x_{j+1}) = f(x_{j+1}) = S_{j+1}(x_{j+1})$
$S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$
$S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$
x_0 x_1 x_2 ... x_j x_{j+1} x_{j+2} ... x_{n-2} x_{n-1} x_n x

Cubic Spline Interpolation

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

(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).

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.

Although cubic splines are defined with other boundary conditions, the conditions given in (f) are sufficient for our purposes. When the natural boundary conditions are used, the spline assumes the shape that a long flexible rod would take if forced to go through the points $\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$. This spline continues linearly when $x \le x_0$ and when $x \ge x_n$.

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

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

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

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

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.

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, we need 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 term $x_{j+1} - x_j$ is 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 $a_n = f(x_n)$, then the equation

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \tag{3.1}$$

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

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

$$S_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2 \tag{3.2}$$

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

$$b_{j+1} = b_j + 2c_jh_j + 3d_jh_j^2, \tag{3.3}$$

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

Another relation between the coefficients of S_j is obtained by defining $c_n = S''(x_n)/2$ and applying condition (e). In this case,

$$c_{j+1} = c_j + 3d_jh_j, \tag{3.4}$$

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

Solving for d_j in Eq. (3.4) and substituting this value into Eqs. (3.1) and (3.3) gives the new equations

$$a_{j+1} = a_j + b_jh_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) \tag{3.5}$$

and

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}) \tag{3.6}$$

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

The final relationship involving the coefficients is obtained by solving the appropriate equation in the form of Eq. (3.5) for b_j,

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}), \tag{3.7}$$

and then, with a reduction of the index, for b_{j-1}, which 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. (3.6), when the index is reduced by 1, gives the linear system of equations

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_jc_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}) \tag{3.8}$$

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

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. (3.7) and $\{d_j\}_{j=0}^{n-1}$ from Eq. (3.4) and to construct the cubic polynomials $\{S_j(x)\}_{j=0}^{n-1}$. In the case of the clamped spline, we also need equations involving the $\{c_j\}$ that ensure that $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$. In Eq. (3.2) we have $S_j'(x)$ in terms of b_j, c_j, and d_j. Since we now know b_j and d_j in terms of c_j, we can use this equation to show that the appropriate equations are

$$2h_0c_0 + h_0c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(x_0) \tag{3.9}$$

Program NCUBSP34 creates a Natural Cubic Spline

and

$$h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f'(x_n) - \frac{3}{h_{n-1}}(a_n - a_{n-1}). \tag{3.10}$$

The solution to the cubic spline problem with the natural boundary conditions $S''(x_0) = S''(x_n) = 0$ can be obtained by applying the program NCUBSP34. The program CCUBSP35 determines the cubic spline with the clamped boundary conditions $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$.

Example 2 Determine the clamped cubic spline for $f(x) = x \sin 4x$ using the nodes $x_0 = 0, x_1 = 0.25$, $x_2 = 0.4$, and $x_3 = 0.6$.

Program CCUBSP35 creates a Clamped Cubic Spline

Solution We first define the function $f(x)$ and its derivative $fp(x) \equiv f'(x)$ in MATLAB with

```
f = inline('x*sin(4*x)','x')
fp = inline('sin(4*x)+4*x*cos(4*x)','x')
```

We define the nodes using the MATLAB capability of defining subscripted variables within square brackets, where a blank is used to separate entries. The subscripts in MATLAB begin with 1 so $x(1) = 0, x(2) = 0.25, x(3) = 0.4$, and $x(4) = 0.6$. This is entered in MATLAB as

```
x = [0 0.25 0.4 0.6]
```

The step sizes are defined by

```
h = [x(2)-x(1)    x(3)-x(2)    x(4)-x(3)]
```

and the values of the function at the nodes by

```
a = [f(x(1))    f(x(2))    f(x(3))    f(x(4))]
```

MATLAB responds to this last command with

$$a = 0 \quad 0.210367746201974 \quad 0.399829441216602 \quad 0.405277908330691$$

A 4×4 array A is defined whose rows are defined by the system of equations used to determine the quadratic coefficients, that is, the c's. These are given in Eqs. (3.8), (3.9), and (3.10), but Eq. (3.9) involves an index of 0, which MATLAB does not permit. So the indices must all be increased by 1 to compensate. So A is defined as follows:

Row 1: the left-hand side of Eq. (3.9), with all the indices increased by 1; that is:

$$2h_1c_1 + h_1c_2 + 0 \cdot c_3 + 0 \cdot c_4.$$

Row 2: the left-hand side of Eq. (3.8) when $j = 2$:

$$h_1c_1 + 2(h_1 + h_2)c_2 + h_2 \cdot c_3 + 0 \cdot c_4.$$

Row 3: the left-hand side of Eq. (3.8) when $j = 3$:

$$0 \cdot c_1 + h_2c_2 + 2(h_2 + h_3)c_3 + h_3c_4.$$

Row 4: the left-hand side of Eq. (3.10) when $n = 4$:

$$0 \cdot c_1 + 0 \cdot c_2 + h_3c_3 + 2h_3c_4.$$

This gives

```
A = [ 2*h(1)  h(1)  0  0; h(1)  2*(h(1)+h(2))  h(2) 0;  0  h(2)
    2*(h(2)+h(3))  h(3);  0  0  h(3)  2*h(3)]
```

The right-hand sides of the same equations are stored in the 4×1 array B.

Row 1: the right-hand side of Eq. (3.9), with all the indices increased by 1; that is:

$$\frac{3}{h_1}(a_2 - a_1) - 3f'(x_1).$$

Row 2: the right-hand side of Eq. (3.8) when $j = 2$:

$$\frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1).$$

Row 3: the right-hand side of Eq. (3.8) when $j = 3$:

$$\frac{3}{h_3}(a_4 - a_3) - \frac{3}{h_2}(a_3 - a_2).$$

Row 4: the right-hand side of Eq. (3.10) when $n = 4$:

$$3f'(x_4) - \frac{3}{h_3}(a_4 - a_3).$$

So B is defined by

```
B = [3*(a(2)-a(1))/h(1) - 3*fp(x(1));
    3*(a(3)-a(2))/h(2)-3*(a(2)-a(1))/h(1);
    3*(a(4)-a(3))/h(3)-3*(a(3)-a(2))/h(2);
    3*fp(x(4))-3*(a(4)-a(3))/h(3)]
```

MATLAB gives these as

$$A = \begin{bmatrix} 0.500000000000000 & 0.250000000000000 & 0 & 0 \\ 0.250000000000000 & 0.800000000000000 & 0.150000000000000 & 0 \\ 0 & 0.150000000000000 & 0.700000000000000 & 0.200000000000000 \\ 0 & 0 & 0.200000000000000 & 0.400000000000000 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 2.524412954423689 \\ 1.264820945868869 \\ -3.707506893581232 \\ -3.364572216954841 \end{bmatrix}$$

We now can have MATLAB solve the system using the `linsolve` command.

```
c = linsolve(A,B)
```

MATLAB responds with solution consisting of $c(1)$, $c(2)$, $c(3)$, and $c(4)$ as

$$c = \begin{bmatrix} 4.649673230468573 \\ 0.798305356757612 \\ -3.574944314362422 \\ -6.623958385205892 \end{bmatrix}$$

Now use Eq. (3.7) to obtain the values of $b(1)$, $b(2)$, and $b(3)$ with the command

```
b = [(a(2)-a(1))/h(1) - h(1)*(2*c(1)+c(2))/3;
     (a(3)-a(2))/h(2) - h(2)*(2*c(2)+c(3))/3;
     (a(4)-a(3))/h(3) - h(3)*(2*c(3)+c(4))/3]
```

which MATLAB gives as

$$b = \begin{bmatrix} 0.000000000000000 \\ 1.361994646806546 \\ 0.945498803165825 \end{bmatrix}$$

Finally, the values of $d(1)$, $d(2)$, and $d(3)$ are obtained using Eq. (3.4) and the command

```
d = [(c(2)-c(1))/(3*h(1));  (c(3)-c(2))/(3*h(2));
     (c(4)-c(3))/(3*h(3))]
```

producing

$$d = \begin{bmatrix} -5.135157164947948 \\ -9.718332602488962 \\ -5.081690118072451 \end{bmatrix}$$

This implies that the cubic spline, to three decimal places, is as shown in Table 3.13.

Table 3.13

j	x_j	a_j	b_j	c_j	d_j
0	0.000	0.000	0.000	4.650	−5.135
1	0.250	0.210	1.362	0.798	−9.718
2	0.400	0.400	0.945	−3.575	−5.082
3	0.600	0.405		−6.624	

■

In the following Illustration the shape of the curve is much more complex. Placing a minimal number of data points along the curve to get a good representation would require some experimentation.

Illustration Figure 3.10 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 3.14 lists the coordinates of 21 data points relative to the superimposed coordinate system shown in Figure 3.11. Notice that more points are used when the curve is changing rapidly than when it is changing more slowly.

Figure 3.10

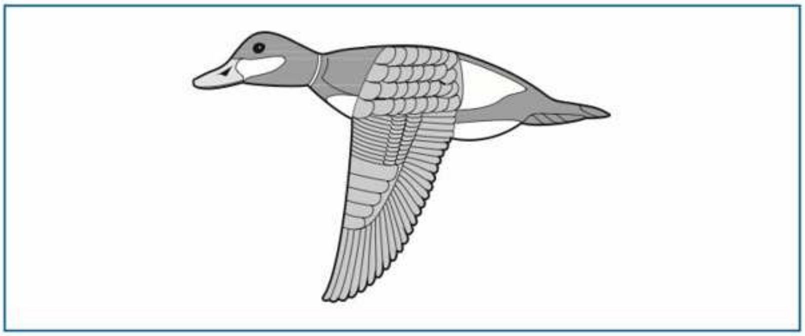


Figure 3.11

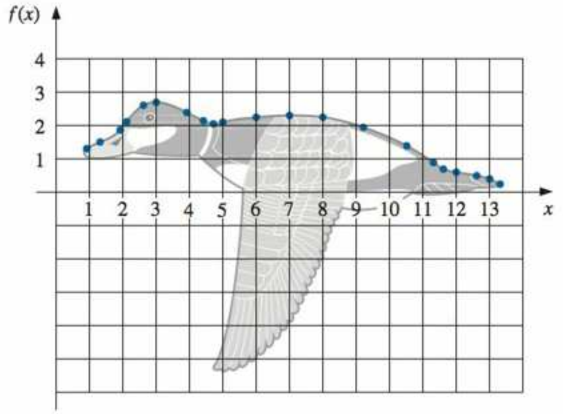


Table 3.14

x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3
$f(x)$	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25

Using the program NCUBSP34 to generate the natural cubic spline for this data produces the coefficients shown in Table 3.15. This spline curve is nearly identical to the profile, as shown in Figure 3.12.

For comparison purposes, Figure 3.13 gives an illustration of the curve that is generated using a Lagrange interpolating polynomial to fit the data given in Table 3.14. The interpolating polynomial in this case is of degree 20 and oscillates wildly. It produces a very strange illustration of the back of a duck, in flight or otherwise.

Table 3.15

j	x_j	a_j	b_j	c_j	d_j
0	0.9	1.3	5.40	0.00	−0.25
1	1.3	1.5	0.42	−0.30	0.95
2	1.9	1.85	1.09	1.41	−2.96
3	2.1	2.1	1.29	−0.37	−0.45
4	2.6	2.6	0.59	−1.04	0.45
5	3.0	2.7	−0.02	−0.50	0.17
6	3.9	2.4	−0.50	−0.03	0.08
7	4.4	2.15	−0.48	0.08	1.31
8	4.7	2.05	−0.07	1.27	−1.58
9	5.0	2.1	0.26	−0.16	0.04
10	6.0	2.25	0.08	−0.03	0.00
11	7.0	2.3	0.01	−0.04	−0.02
12	8.0	2.25	−0.14	−0.11	0.02
13	9.2	1.95	−0.34	−0.05	−0.01
14	10.5	1.4	−0.53	−0.10	−0.02
15	11.3	0.9	−0.73	−0.15	1.21
16	11.6	0.7	−0.49	0.94	−0.84
17	12.0	0.6	−0.14	−0.06	0.04
18	12.6	0.5	−0.18	0.00	−0.45
19	13.0	0.4	−0.39	−0.54	0.60
20	13.3	0.25			

Figure 3.12

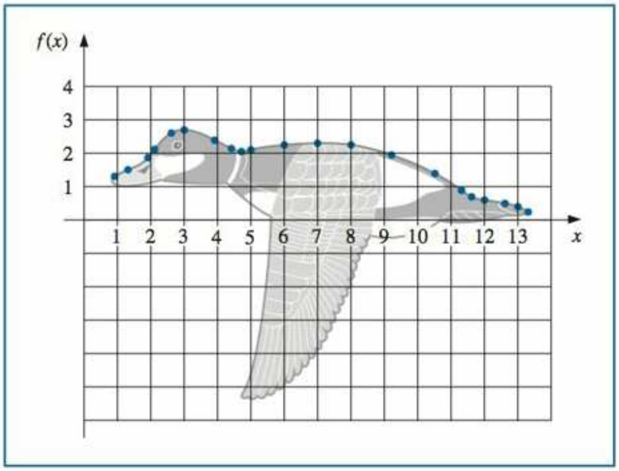
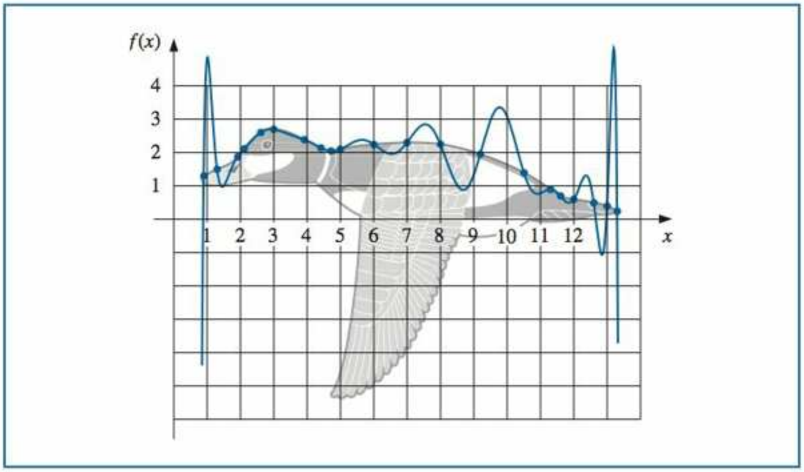


Figure 3.13



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 natural cubic spline to the curve of the top profile. □

Cubic splines generally agree quite well with the function being approximated, provided that the points are not too far apart and the fourth derivative of the function is well behaved. For example, suppose that f has four continuous derivatives on $[a, b]$ and that the fourth derivative on this interval has a magnitude bounded by M. Then the clamped cubic spline

$S(x)$ agreeing with $f(x)$ at the points $a = x_0 < x_1 < \cdots < x_n = b$ has the property that for all x in $[a, b]$,

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

A similar—but more complicated—result holds for the natural cubic splines.

EXERCISE SET 3.5

1. Determine the natural 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 natural 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

c.

x	$f(x)$
−0.5	−0.0247500
−0.25	0.3349375
0	1.1010000

d.

x	$f(x)$
0.1	−0.62049958
0.2	−0.28398668
0.3	0.00660095
0.4	0.24842440

4. The data in Exercise 3 were generated using the following functions. Use the cubic splines constructed in Exercise 3 for the given value of x to approximate $f(x)$ and $f'(x)$, and calculate the actual error.
   a. $f(x) = x \ln x$; approximate $f(8.4)$ and $f'(8.4)$.
   b. $f(x) = \sin(e^x - 2)$; approximate $f(0.9)$ and $f'(0.9)$.
   c. $f(x) = x^3 + 4.001x^2 + 4.002x + 1.101$; approximate $f(-\frac{1}{3})$ and $f'(-\frac{1}{3})$.
   d. $f(x) = x \cos x - 2x^2 + 3x - 1$; approximate $f(0.25)$ and $f'(0.25)$.
5. Construct the clamped cubic spline using the data of Exercise 3 and the fact that
   a. $f'(8.3) = 3.116256$ and $f'(8.6) = 3.151762$
   b. $f'(0.8) = 2.1691753$ and $f'(1.0) = 2.0466965$
   c. $f'(-0.5) = 0.7510000$ and $f'(0) = 4.0020000$
   d. $f'(0.1) = 3.58502082$ and $f'(0.4) = 2.16529366$
6. Repeat Exercise 4 using the clamped cubic splines constructed in Exercise 5.
7. **a.** Construct a natural cubic spline to approximate $f(x) = \cos \pi x$ by using the values given by $f(x)$ at $x = 0, 0.25, 0.5, 0.75$, and 1.0.
   b. Integrate the spline over $[0, 1]$, and compare the result to $\int_0^1 \cos \pi x \, dx = 0$.
   c. Use the derivatives of the spline to approximate $f'(0.5)$ and $f''(0.5)$, and compare these approximations to the actual values.
8. **a.** Construct a natural cubic spline to approximate $f(x) = e^{-x}$ by using the values given by $f(x)$ at $x = 0, 0.25, 0.75$, and 1.0.
   b. Integrate the spline over $[0, 1]$, and compare the result to $\int_0^1 e^{-x} \, dx = 1 - 1/e$.
   c. Use the derivatives of the spline to approximate $f'(0.5)$ and $f''(0.5)$, and compare the approximations to the actual values.
9. Repeat Exercise 7, constructing instead the clamped cubic spline with $f'(0) = f'(1) = 0$.
10. Repeat Exercise 8, constructing instead the clamped cubic spline with $f'(0) = -1$, $f'(1) = -e^{-1}$.

11. A natural cubic spline S on $[0, 2]$ is defined by

$$S(x) = \begin{cases} S_0(x) = 1 + 2x - x^3, & \text{if } 0 \le x < 1, \\ S_1(x) = a + b(x-1) + c(x-1)^2 + d(x-1)^3, & \text{if } 1 \le x \le 2. \end{cases}$$

Find a, b, c, and d.

12. A clamped cubic spline s for a function f is defined on $[1, 3]$ by

$$s(x) = \begin{cases} s_0(x) = 3(x-1) + 2(x-1)^2 - (x-1)^3, & \text{if } 1 \le x < 2, \\ s_1(x) = a + b(x-2) + c(x-2)^2 + d(x-2)^3, & \text{if } 2 \le x \le 3. \end{cases}$$

Given $f'(1) = f'(3)$, find a, b, c, and d.

13. A natural cubic spline S is defined by

$$S(x) = \begin{cases} S_0(x) = 1 + B(x-1) - D(x-1)^3, & \text{if } 1 \le x < 2, \\ S_1(x) = 1 + b(x-2) - \frac{3}{4}(x-2)^2 + d(x-2)^3, & \text{if } 2 \le x \le 3. \end{cases}$$

If S interpolates the data $(1, 1)$, $(2, 1)$, and $(3, 0)$, find B, D, b, and d.

14. A clamped cubic spline s for a function f is defined by

$$s(x) = \begin{cases} s_0(x) = 1 + Bx + 2x^2 - 2x^3, & \text{if } 0 \le x < 1, \\ s_1(x) = 1 + b(x-1) - 4(x-1)^2 + 7(x-1)^3, & \text{if } 1 \le x \le 2. \end{cases}$$

Find $f'(0)$ and $f'(2)$.

15. Suppose that $f(x)$ is a polynomial of degree 3. Show that $f(x)$ is its own clamped cubic spline but that it cannot be its own natural cubic spline.

16. Suppose the data $\{x_i, f(x_i)\}_{i=1}^{n}$ lie on a straight line. What can be said about the natural and clamped cubic splines for the function f? [*Hint:* Take a cue from the results of Exercises 1 and 2.]

17. The data in the following table give the population of the United States for the years 1960 to 2010 and were considered in Exercise 16 of Section 3.2 and Exercise 6 of Section 3.3.

Year	1960	1970	1980	1990	2000	2010
Population (thousands)	179,323	203,302	226,542	249,633	281,442	307,746

a. Find a natural cubic spline agreeing with these data, and use the spline to predict the population in the years 1950, 1975, and 2020.

b. Compare your approximations with those previously obtained. If you had to make a choice, which interpolation procedure would you choose?

18. A car traveling along a straight road is clocked at a number of points. The data from the observations are given in the following table, where the time is in seconds, the distance is in feet, and the speed is in feet per second.

Time	0	3	5	8	13
Distance	0	225	383	623	993
Speed	75	77	80	74	72

a. Use a clamped cubic spline to predict the position of the car and its speed when $t = 10$ s.

b. Use the derivative of the spline to determine whether the car ever exceeds a 55-mi/h speed limit on the road; if so, what is the first time the car exceeds this speed?

c. What is the predicted maximum speed for the car?

19. The 2011 Kentucky Derby was won by a horse named Animal Kingdom (at 20:1 odds) in a time of 2:02.04 (2 minutes and 2.04 seconds) for the $1\frac{1}{4}$-mile race. Times at the quarter-mile, half-mile, and mile poles were 0:24.26, 0:59.68, and 1:47.95.

a. Use these values together with the starting time to construct a natural cubic spline for Animal Kingdom's race.

b. Use the spline to predict the time at the three-quarter-mile pole, and compare this to the actual time of 1:24.40.

c. Use the spline to approximate Animal Kingdom's speed at the finish line.

20. It is suspected that the high amounts of tannin in mature oak leaves inhibit the growth of the winter moth (*Operophtera bromata L., Geometridae*) larvae that extensively damage these trees in certain years. The following table lists the average weight of two samples of larvae at times in the first 28 days after birth. The first sample was reared on young oak leaves, whereas the second sample was reared on mature leaves from the same tree.

a. Use a natural cubic spline to approximate the average weight curve for each sample.

b. Find an approximate maximum average weight for each sample by determining the maximum of the spline.

Day	0	6	10	13	17	20	28
Sample 1 average weight (mg)	6.67	17.33	42.67	37.33	30.10	29.31	28.74
Sample 2 average weight (mg)	6.67	16.11	18.89	15.00	10.56	9.44	8.89

3.6 Parametric Curves

None of the techniques we have developed can be used to generate curves of the form shown in Figure 3.14, because this curve cannot be expressed as a function of one coordinate variable in terms of the other. In this section we will see how to represent general curves by using a parameter to express both the x- and y-coordinate variables. This technique can be extended to represent general curves and surfaces in space.

Figure 3.14