0	0.0000			
		0.9549		
$\pi/6$	0.5000		−0.2443	
		0.6990		−0.1139
$2\pi/6$	0.8660		−0.4232	
		0.2559		
$3\pi/6$	1.0000			

The degree 3 interpolating polynomial is therefore

$$\begin{aligned} P_3(x) &= 0 + 0.9549x - 0.2443x(x - \pi/6) - 0.1139x(x - \pi/6)(x - \pi/3) \\ &= 0 + x(0.9549 + (x - \pi/6)(-0.2443 + (x - \pi/3)(-0.1139))). \end{aligned} \tag{3.5}$$

This polynomial is graphed together with the sine function in Figure 3.3. At this level of resolution, $P_3(x)$ and $\sin x$ are virtually indistinguishable on the interval $[0, \pi/2]$. We have compressed the infinite amount of information held by the sine curve into a few stored coefficients and the ability to perform the 3 adds and 3 multiplies in (3.5). ◀

How close are we to designing the `sin` key on a calculator? Certainly we need to handle inputs from the entire real line. But due to the symmetries of the sine function, we have done the hard part. The interval $[0, \pi/2]$ is a so-called **fundamental domain** for sine, meaning that an input from any other interval can be referred back to it. Given an input x from $[\pi/2, \pi]$, say, we can compute $\sin x$ as $\sin(\pi - x)$, since sin is symmetric about $x = \pi/2$. Given an input x from $[\pi, 2\pi]$, $\sin x = -\sin(2\pi - x)$ due to antisymmetry about $x = \pi$. Finally, because sin repeats its behavior on the interval $[0, 2\pi]$ across the entire real line, we can calculate for any input by first reducing modulo 2π. This leads to a straightforward design for the `sin` key:

```
%Program 3.3 Building a sin calculator key, attempt #1
%Approximates sin curve with degree 3 polynomial
%     (Caution: do not use to build bridges,
%     at least until we have discussed accuracy.)
%Input:  x
%Output: approximation for sin(x)
function y=sin1(x)
%First calculate the interpolating polynomial and
%  store coefficients
b=pi*(0:3)/6;yb=sin(b);     % b holds base points
c=newtdd(b,yb,4);
%For each input x, move x to the fundamental domain and evaluate
%      the interpolating polynomial
s=1;                        % Correct the sign of sin
x1=mod(x,2*pi);
if x1>pi
  x1 = 2*pi-x1;
  s = -1;
end
if x1 > pi/2
  x1 = pi-x1;
end
y = s*nest(3,c,x1,b);
```

Most of the work in Program 3.3 is to place x into the fundamental domain. Then we evaluate the degree 3 polynomial by nested multiplication. Here is some typical output from Program 3.3:

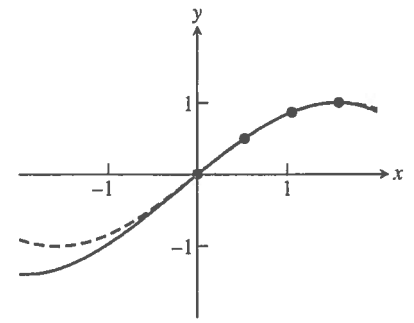

Figure 3.3 Degree 3 interpolation of sin x. The interpolation polynomial (solid curve) is plotted along with $y = \sin x$. Equally spaced interpolation nodes are at $0, \pi/6, 2\pi/6$, and $3\pi/6$. The approximation is very close between 0 and $\pi/2$.

x	$\sin x$	`sin1(x)`	error
1	0.8415	0.8411	0.0004
2	0.9093	0.9102	0.0009
3	0.1411	0.1428	0.0017
4	−0.7568	−0.7557	0.0011
14	0.9906	0.9928	0.0022
1000	0.8269	0.8263	0.0006

This is not bad for the first try. The error is usually under 1 percent. In order to get enough correct digits to fill the calculator readout, we'll need to know a little more about interpolation error, the topic of the next section.

3.1 Exercises

1. Use Lagrange interpolation to find a polynomial that passes through the points.
   (a) $(0, 1), (2, 3), (3, 0)$
   (b) $(-1, 0), (2, 1), (3, 1), (5, 2)$
   (c) $(0, -2), (2, 1), (4, 4)$
2. Use Newton's divided differences to find the interpolating polynomials of the points in Exercise 1, and verify agreement with the Lagrange interpolating polynomial.
3. How many degree d polynomials pass through the four points $(-1, 3), (1, 1), (2, 3), (3, 7)$? Write one down if possible. (a) $d = 2$ (b) $d = 3$ (c) $d = 6$.
4. (a) Find a polynomial $P(x)$ of degree 3 or less whose graph passes through the points $(0, 0), (1, 1), (2, 2), (3, 7)$. (b) Find two other polynomials (of any degree) that pass through these four points. (c) Decide whether there exists a polynomial $P(x)$ of degree 3 or less whose graph passes through the points $(0, 0), (1, 1), (2, 2), (3, 7)$, and $(4, 2)$.
5. (a) Find a polynomial $P(x)$ of degree 3 or less whose graph passes through the four data points $(-2, 8), (0, 4), (1, 2), (3, -2)$. (b) Describe any other polynomials of degree 4 or less which pass through the four points in part (a).
6. Write down a polynomial of degree exactly 5 that interpolates the four points $(1, 1), (2, 3), (3, 3), (4, 4)$.

7. Find $P(0)$, where $P(x)$ is the degree 10 polynomial that is zero at $x = 1, \ldots, 10$ and satisfies $P(12) = 44$.

8. Let $P(x)$ be the degree 9 polynomial that takes the value 112 at $x = 1$, takes the value 2 at $x = 10$, and equals zero for $x = 2, \ldots, 9$. Calculate $P(0)$.

9. Give an example of the following, or explain why no such example exists. (a) A degree 6 polynomial $L(x)$ that is zero at $x = 1, 2, 3, 4, 5, 6$ and equal to 10 at $x = 7$. (b) A degree 6 polynomial $L(x)$ that is zero at $x = 1, 2, 3, 4, 5, 6$, equal to 10 at $x = 7$, and equal to 70 at $x = 8$.

10. Let $P(x)$ be the degree 5 polynomial that takes the value 10 at $x = 1, 2, 3, 4, 5$ and the value 15 at $x = 6$. Find $P(7)$.

11. Let P_1, P_2, P_3, and P_4 be four different points lying on a parabola $y = ax^2 + bx + c$. How many cubic (degree 3) polynomials pass through those four points? Explain your answer.

12. Can a degree 3 polynomial intersect a degree 4 polynomial in exactly five points? Explain.

13. Let $P(x)$ be the degree 10 polynomial through the 11 points $(-5, 5), (-4, 5), (-3, 5), (-2, 5), (-1, 5), (0, 5), (1, 5), (2, 5), (3, 5), (4, 5), (5, 42)$. Calculate $P(6)$.

14. Write down 4 noncollinear points $(1, y_1), (2, y_2), (3, y_3), (4, y_4)$ that do not lie on any polynomial $y = P_3(x)$ of degree exactly three.

15. Write down the degree 25 polynomial that passes through the points $(1, -1), (2, -2), \ldots, (25, -25)$ and has constant term equal to 25.

16. List all degree 42 polynomials that pass through the eleven points $(-5, 5), (-4, 4), \ldots, (4, -4), (5, -5)$ and have constant term equal to 42.

17. The estimated mean atmospheric concentration of carbon dioxide in earth's atmosphere is given in the table that follows, in parts per million by volume. Find the degree 3 interpolating polynomial of the data and use it to estimate the CO_2 concentration in (a) 1950 and (b) 2050. (The actual concentration in 1950 was 310 ppm.)

year	CO_2 (ppm)
1800	280
1850	283
1900	291
2000	370

18. The expected lifetime of an industrial fan when operated at the listed temperature is shown in the table that follows. Estimate the lifetime at 70°C by using (a) the parabola from the last three data points (b) the degree 3 curve using all four points.

temp (°C)	hrs (×1000)
25	95
40	75
50	63
60	54

3.1 Computer Problems

1. Apply the following world population figures to estimate the 1980 population, using (a) the straight line through the 1970 and 1990 estimates; (b) the parabola through the 1960, 1970, and 1990 estimates; and (c) the cubic curve through all four data points. Compare with the 1980 estimate of 4452584592.

year	population
1960	3039585530
1970	3707475887
1990	5281653820
2000	6079603571

2. Write a version of Program 3.2 that is a MATLAB function, whose inputs `x` and `y` are equal length vectors of data points, and whose output is a plot of the interpolating polynomial. In this way, the points can be entered more accurately than by mouse input. Check your program by replicating Figure 3.2.

3. Write a MATLAB function `polyinterp.m` that takes as input a set of (x, y) interpolating points and another x_0, and outputs y_0, the value of the interpolating polynomial at x_0. The first line of the file should be `function y0=polyinterp(x,y,x0)`, where `x` and `y` are input vectors of data points. Your function may call `newtdd` from Program 3.1 and `nest` from Chapter 0, and may be structured similarly to Program 3.2, but without the graphics. Demonstrate that your function works.

4. Remodel the `sin1` calculator key in Program 3.3 to build `cos1`, a cosine key that follows the same principles. First decide on the fundamental domain for cosine.

5. (a) Use the addition formulas for sin and cos to prove that $\tan(\pi/2 - x) = 1/\tan x$. (b) Show that $[0, \pi/4]$ can be used as a fundamental domain for $\tan x$. (c) Design a tangent key, following the principles of Program 3.3, using degree 3 polynomial interpolation on this fundamental domain. (d) Empirically calculate the maximum error of the tangent key in $[0, \pi/4]$.

3.2 INTERPOLATION ERROR

The accuracy of our `sin` calculator key depends on the approximation in Figure 3.3. How close is it? We presented a table indicating that, for a few examples, the first two digits are fairly reliable, but after that the digits are not always correct. In this section, we investigate ways to measure this error and determine how to make it smaller.

3.2.1 Interpolation error formula

Assume that we start with a function $y = f(x)$ and take data points from it to build an interpolating polynomial $P(x)$, as we did with $f(x) = \sin x$ in Example 3.7. The **interpolation error** at x is $f(x) - P(x)$, the difference between the original function that provided the data points and the interpolating polynomial, evaluated at x. The interpolation error is the vertical distance between the curves in Figure 3.3. The next theorem gives a formula for the interpolation error that is usually impossible to evaluate exactly, but often can at least lead to an error bound.

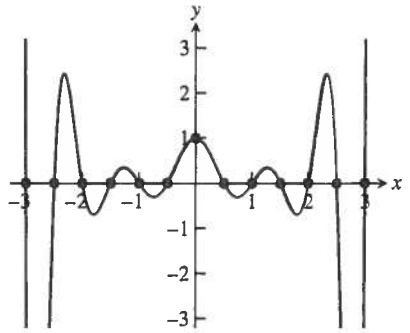

Figure 3.5 Interpolation of Triangular Bump Function. The interpolating polynomial wiggles much more than the input data points.

that is characteristic of the Runge phenomenon: polynomial wiggle near the ends of the interpolation interval. ◀

As we have seen, examples with the Runge phenomenon characteristically have large error near the outside of the interval of data points. The cure for this problem is intuitive: Move some of the interpolation points toward the outside of the interval, where the function producing the data can be better fit. We will see how to accomplish this in the next section on Chebyshev interpolation.

3.2 Exercises

1. (a) Find the degree 2 interpolating polynomial $P_2(x)$ through the points $(0, 0)$, $(\pi/2, 1)$, and $(\pi, 0)$. (b) Calculate $P_2(\pi/4)$, an approximation for $\sin(\pi/4)$. (c) Use Theorem 3.3 to give an error bound for the approximation in part (b). (d) Using a calculator or MATLAB, compare the actual error to your error bound.
2. (a) Given the data points $(1, 0)$, $(2, \ln 2)$, $(4, \ln 4)$, find the degree 2 interpolating polynomial. (b) Use the result of (a) to approximate $\ln 3$. (c) Use Theorem 3.3 to give an error bound for the approximation in part (b). (d) Compare the actual error to your error bound.
3. Assume that the polynomial $P_9(x)$ interpolates the function $f(x) = e^{-2x}$ at the 10 evenly spaced points $x = 0, 1/9, 2/9, 3/9, \ldots, 8/9, 1$. (a) Find an upper bound for the error $|f(1/2) - P_9(1/2)|$. (b) How many decimal places can you guarantee to be correct if $P_9(1/2)$ is used to approximate e?
4. Consider the interpolating polynomial for $f(x) = 1/(x + 5)$ with interpolation nodes $x = 0, 2, 4, 6, 8, 10$. Find an upper bound for the interpolation error at (a) $x = 1$ and (b) $x = 5$.
5. Assume that a function $f(x)$ has been approximated by the degree 5 interpolating polynomial $P(x)$, using the data points $(x_i, f(x_i))$, where $x_1 = .1$, $x_2 = .2$, $x_3 = .3$, $x_4 = .4$, $x_5 = .5$, $x_6 = .6$. Do you expect the interpolation error $|f(x) - P(x)|$ to be smaller for $x = .35$ or for $x = .55$? Quantify your answer.
6. Assume that the polynomial $P_5(x)$ interpolates a function $f(x)$ at the six data points $(x_i, f(x_i))$ with x-coordinates $x_1 = 0, x_2 = .2, x_3 = .4, x_4 = .6, x_5 = .8$, and $x_6 = 1$. Assume that the interpolation error at $x = .3$ is $|f(.3) - P_5(.3)| = .01$. Estimate the new interpolation error

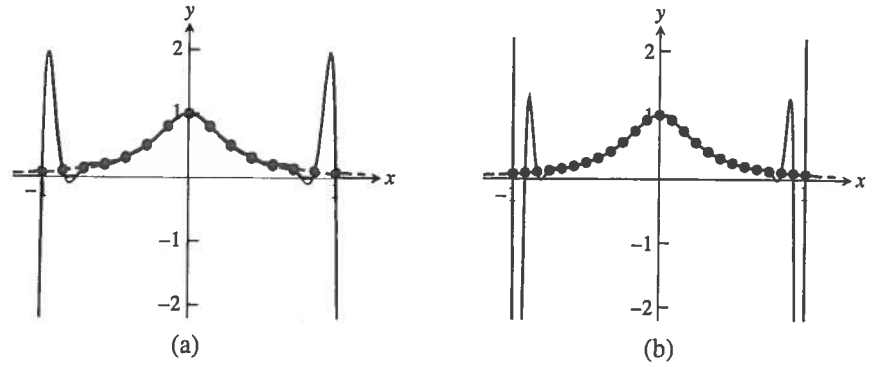

Figure 3.6 Runge Example. Polynomial interpolation of the Runge function of Example 3.9 at evenly spaced base points causes extreme variation near the ends of the interval, similar to Figure 3.5 (a) 15 base points (b) 25 base points.

$|f(.3) - P_7(.3)|$ that would result if two additional interpolation points $(x_6, y_6) = (.1, f(.1))$ and $(x_7, y_7) = (.5, f(.5))$ are added. What assumptions have you made to produce this estimate?

3.2 Computer Problems

1. (a) Use the method of divided differences to find the degree 4 interpolating polynomial $P_4(x)$ for the data $(0.6, 1.433329)$, $(0.7, 1.632316)$, $(0.8, 1.896481)$, $(0.9, 2.247908)$, and $(1.0, 2.718282)$. (b) Calculate $P_4(0.82)$ and $P_4(0.98)$. (c) The preceding data come from the function $f(x) = e^{x^2}$. Use the interpolation error formula to find upper bounds for the error at $x = 0.82$ and $x = 0.98$, and compare the bounds with the actual error. (d) Plot the actual interpolation error $P(x) - e^{x^2}$ on the intervals $[.5, 1]$ and $[0, 2]$.
2. Plot the interpolation error of the `sin1` key from Program 3.3 on the interval $[-2\pi, 2\pi]$.
3. The total world oil production in millions of barrels per day is shown in the table that follows. Determine and plot the degree 9 polynomial through the data. Use it to estimate 2010 oil production. Does the Runge phenomenon occur in this example? In your opinion, is the interpolating polynomial a good model of the data? Explain.

year	bbl/day ($\times 10^6$)
1994	67.052
1995	68.008
1996	69.803
1997	72.024
1998	73.400
1999	72.063
2000	74.669
2001	74.487
2002	74.065
2003	76.777

4. Use the degree 3 polynomial through the first four data points in Computer Problem 3 to estimate the 1998 world oil production. Is the Runge phenomenon present?

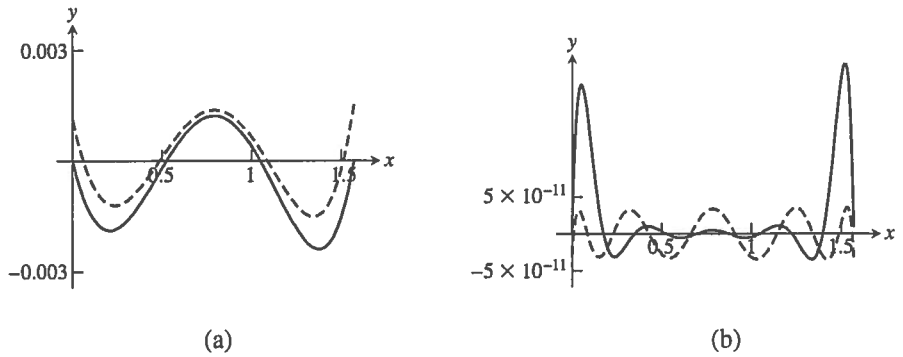

Figure 3.11 Interpolation error for approximating $f(x) = \sin x$. (a) Interpolation error for degree 3 interpolating polynomial with evenly spaced base points (solid curve) and Chebyshev base points (dashed curve). (b) Same as (a), but degree 9.

$$|\sin x - P_{n-1}(x)| = \frac{|(x - x_1)\cdots(x - x_n)|}{n!}|f^{(n)}(c)|$$
$$\leq \frac{\left(\frac{\pi}{2}-0\right)^n}{n!2^{n-1}}1.$$

This equation is not simple to solve for n, but a little trial and error finds that for $n = 9$ the error bound is $\approx 0.1224 \times 10^{-8}$, and for $n = 10$ it is $\approx 0.4807 \times 10^{-10}$. The latter meets our criterion for 10 correct decimal places. Figure 3.11(b) compares the actual error of the Chebyshev interpolation polynomial with the error of the evenly spaced interpolation polynomial.

The 10 Chebyshev base points on $[0, \pi/2]$ are $\pi/4 + (\pi/4)\cos(\text{odd } \pi/20)$. The key can be designed by storing the 10 y-values for sine at the base points and doing a nested multiplication evaluation for each key press. ◄

The following MATLAB code `sin2.m` carries out the preceding task. The code is a bit awkward as written: We have to do 10 sin evaluations, at the 10 Chebyshev nodes, in order to set up the interpolating polynomial to approximate sin at one point. Of course, in a real implementation, these numbers would be computed once and stored.

```
%Program 3.4 Building a sin calculator key, attempt #2
%Approximates sin curve with degree 9 polynomial
%Input:  x
%Output: approximation for sin(x), correct to 10 decimal places
function y=sin2(x)
%First calculate the interpolating polynomial and
%   store coefficients
n=10;
b=pi/4+(pi/4)*cos((1:2:2*n-1)*pi/(2*n));
yb=sin(b);                          % b holds Chebyshev base points
c=newtdd(b,yb,n);
%For each input x, move x to the fundamental domain and evaluate
%   the interpolating polynomial
s=1;                                % Correct the sign of sin
x1=mod(x,2*pi);
if x1>pi
  x1 = 2*pi-x1;
  s = -1;
```



```
end
if x1 > pi/2
  x1 = pi-x1;
end
y = s*nest(n-1,c,x1,b);
```

In this chapter, we have often illustrated polynomial interpolation, either evenly spaced or using Chebyshev nodes, for the purpose of approximating the trigonometric functions. Although polynomial interpolation can be used to approximate sine and cosine to arbitrarily high accuracy, most calculators use a slightly more efficient approach called the CORDIC (Coordinate Rotation Digital Computer) algorithm (Volder [1959]). CORDIC is an elegant iterative method, based on complex arithmetic, that can be applied to several special functions. Polynomial interpolation remains a simple and useful technique for approximating general functions and for representing and compressing data.

3.3 Exercises

1. List the Chebyshev interpolation nodes $x_1, \ldots, x_n$ in the given interval. (a) $[-1, 1], n = 6$ (b) $[-2, 2], n = 4$ (c) $[4, 12], n = 6$ (d) $[-0.3, 0.7], n = 5$
2. Find the upper bound for $|(x - x_1)\ldots(x - x_n)|$ on the intervals and Chebyshev nodes in Exercise 1.
3. Assume that Chebyshev interpolation is used to find a fifth degree interpolating polynomial $Q_5(x)$ on the interval $[-1, 1]$ for the function $f(x) = e^x$. Use the interpolation error formula to find a worst-case estimate for the error $|e^x - Q_5(x)|$ that is valid for x throughout the interval $[-1, 1]$. How many digits after the decimal point will be correct when $Q_5(x)$ is used to approximate e^x?
4. Answer the same questions as in Exercise 3, but for the interval $[0.6, 1.0]$.
5. Find an upper bound for the error on $[0, 2]$ when the degree 3 Chebyshev interpolating polynomial is used to approximate $f(x) = \sin x$.
6. Assume that you are to use Chebyshev interpolation to find a degree 3 interpolating polynomial $Q_3(x)$ that approximates the function $f(x) = x^{-3}$ on the interval $[3, 4]$. (a) Write down the (x, y) points that will serve as interpolation nodes for Q_3. (b) Find a worst-case estimate for the error $|x^{-3} - Q_3(x)|$ that is valid for all x in the interval $[3, 4]$. How many digits after the decimal point will be correct when $Q_3(x)$ is used to approximate x^{-3}?
7. Suppose you are designing the **ln** key for a calculator whose display shows six digits to the right of the decimal point. Find the least degree d for which Chebyshev interpolation on the interval $[1, e]$ will approximate within this accuracy.
8. Let $T_n(x)$ denote the degree n Chebyshev polynomial. Find a formula for $T_n(0)$.
9. Determine the following values: (a) $T_{999}(-1)$ (b) $T_{1000}(-1)$ (c) $T_{999}(0)$ (d) $T_{1000}(0)$ (e) $T_{999}(-1/2)$ (f) $T_{1000}(-1/2)$.

3.3 Computer Problems

1. Rebuild Program 3.3 to implement the Chebyshev interpolating polynomial with four nodes on the interval $[0, \pi/2]$. (Only one line of code needs to be changed.) Then plot the polynomial and the sine function on the interval $[-2, 2]$.

2. Build a MATLAB program to evaluate the cosine function correct to 10 decimal places using Chebyshev interpolation. Start by interpolating on a fundamental domain $[0, \pi/2]$, and extend your answer to inputs between -10^4 and 10^4. You may want to use some of the MATLAB code written in this chapter.

3. Carry out the steps of Computer Problem 2 for $\ln x$, for inputs x between 10^{-4} and 10^4. Use $[1, e]$ as the fundamental domain. What is the degree of the interpolation polynomial that guarantees 10 correct digits? Your program should begin by finding the integer k such that $e^k \le x < e^{k+1}$. Then xe^{-k} lies in the fundamental domain. Demonstrate the accuracy of your program by comparing it with MATLAB's `log` command.

4. Let $f(x) = e^{|x|}$. Compare evenly spaced interpolation with Chebyshev interpolation by plotting degree n polynomials of both types on the interval $[-1, 1]$, for $n = 10$ and 20. For evenly spaced interpolation, the left and right interpolation base points should be -1 and 1. By sampling at a 0.01 step size, create the empirical interpolation errors for each type, and plot a comparison. Can the Runge phenomenon be observed in this problem?

5. Carry out the steps of Computer Problem 4 for $f(x) = e^{-x^2}$.

3.4 CUBIC SPLINES

Splines represent an alternative approach to data interpolation. In polynomial interpolation, a single formula, given by a polynomial, is used to meet all data points. The idea of splines is to use several formulas, each a low-degree polynomial, to pass through the data points.

The simplest example of a spline is a linear spline, in which one "connects the dots" with straight-line segments. Assume that we are given a set of data points $(x_1, y_1), \ldots, (x_n, y_n)$ with $x_1 < \cdots < x_n$. A linear spline consists of the $n - 1$ line segments that are drawn between neighboring pairs of points. Figure 3.12(a) shows a linear spline where, between each neighboring pair of points $(x_i, y_i), (x_{i+1}, y_{i+1})$, the linear function $y = a_i + b_i x$ is drawn through the two points. The given data points in the figure are $(1,2), (2,1), (4,4)$, and $(5,3)$, and the linear spline is given by

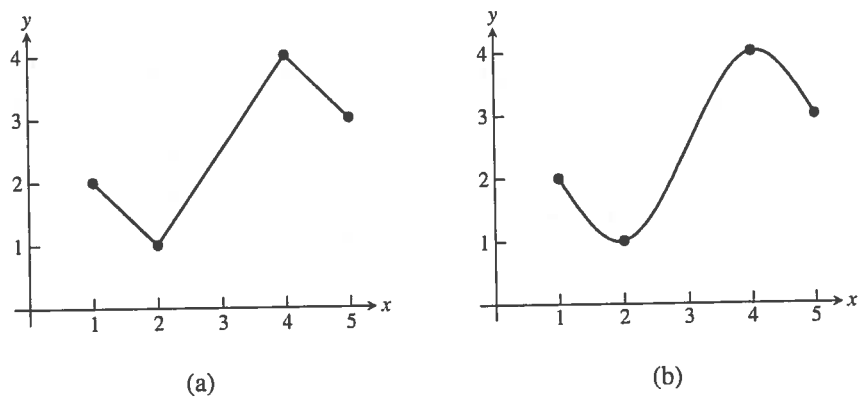

Figure 3.12 Splines through four data points. (a) Linear spline through (1,2), (2,1), (4,4), and (5,3) consists of three linear polynomials given by (3.15). (b) Cubic spline through the same points, given by (3.16).

$$
\begin{aligned}
S_1(x) &= 2 - (x - 1) \text{ on } [1, 2] \\
S_2(x) &= 1 + \frac{3}{2}(x - 2) \text{ on } [2, 4] \\
S_3(x) &= 4 - (x - 4) \text{ on } [4, 5].
\end{aligned} \tag{3.15}
$$

The linear spline successfully interpolates an arbitrary set of n data points. However, linear splines lack smoothness. Cubic splines are meant to address this shortcoming of linear splines. A cubic spline replaces linear functions between the data points by degree 3 (cubic) polynomials.

An example of a cubic spline that interpolates the same points $(1, 2), (2, 1), (4, 4)$, and $(5, 3)$ is shown in Figure 3.12(b). The equations defining the spline are

$$
\begin{aligned}
S_1(x) &= 2 - \frac{13}{8}(x - 1) + 0(x - 1)^2 + \frac{5}{8}(x - 1)^3 \text{ on } [1, 2] \\
S_2(x) &= 1 + \frac{1}{4}(x - 2) + \frac{15}{8}(x - 2)^2 - \frac{5}{8}(x - 2)^3 \text{ on } [2, 4] \\
S_3(x) &= 4 + \frac{1}{4}(x - 4) - \frac{15}{8}(x - 4)^2 + \frac{5}{8}(x - 4)^3 \text{ on } [4, 5].
\end{aligned} \tag{3.16}
$$

Note in particular the smooth transition from one S_i to the next at the base points, or "knots," $x = 2$ and $x = 4$. This is achieved by arranging for the neighboring pieces S_i and S_{i+1} of the spline to have the same zeroth, first, and second derivatives when evaluated at the knots. Just how to do this is the topic of the next section.

Given n points $(x_1, y_1), \ldots, (x_n, y_n)$, there is obviously one and only one linear spline through the data points. This will not be true for cubic splines. We will find that there are infinitely many through any set of data points. Extra conditions will be added when it is necessary to nail down a particular spline of interest.

3.4.1 Properties of splines

To be a little more precise about the properties of a cubic spline, we make the following definition: Assume that we are given the n data points $(x_1, y_1), \ldots, (x_n, y_n)$, where the x_i are distinct and in increasing order. A **cubic spline** $S(x)$ through the data points $(x_1, y_1), \ldots, (x_n, y_n)$ is a set of cubic polynomials

$$
\begin{aligned}
S_1(x) &= y_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 \text{ on } [x_1, x_2] \\
S_2(x) &= y_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3 \text{ on } [x_2, x_3] \\
&\vdots \\
S_{n-1}(x) &= y_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3 \text{on } [x_{n-1}, x_n]
\end{aligned} \tag{3.17}
$$

with the following properties:

Property 1 $S_i(x_i) = y_i$ and $S_i(x_{i+1}) = y_{i+1}$ for $i = 1, \ldots, n - 1$.

Property 2 $S'_{i-1}(x_i) = S'_i(x_i)$ for $i = 2, \ldots, n - 1$.

Property 3 $S''_{i-1}(x_i) = S''_i(x_i)$ for $i = 2, \ldots, n - 1$.

Property 1 guarantees that the spline $S(x)$ interpolates the data points. Property 2 forces the slopes of neighboring parts of the spline to agree where they meet, and Property 3 does the same for the curvature, represented by the second derivative.