4 Interpolation and Numerical Differentiation

The viscosity of water has been experimentally determined at different temperatures, as indicated in the following table:

Temperature	0°	5°	10°	15°
Viscosity	1.792	1.519	1.308	1.140

From this table, how can we estimate a reasonable value for the viscosity at temperature 8°?

The method of polynomial interpolation, described in Section 4.1, can be used to create a polynomial of degree 3 that assumes the values in the table. This polynomial should provide acceptable intermediate values for temperatures not tabulated. The value of that polynomial at the point 8° turns out to be 1.386.

4.1 Polynomial Interpolation

Preliminary Remarks

We pose three problems concerning the representation of functions to give an indication of the subject matter in this chapter, in Chapter 9 (on splines), and in Chapter 12 (on least squares).

(1) First, suppose that we have a table of numerical values of a function:

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

Is it possible to find a simple and convenient formula that reproduces the given points exactly?

(2) The second problem is similar, but it is assumed that the given table of numerical values is contaminated by errors, as might occur if the values came from a physical experiment. Now we ask for a formula that represents the data (approximately) and, if possible, filters out the errors.

(3) As a third problem, a function f is given, perhaps in the form of a computer procedure, but it is an expensive function to evaluate. In this case, we ask for another function g that is simpler to evaluate and produces a reasonable approximation to f. Sometimes in this problem, we want g to approximate f with full machine precision.

In all of these problems, a simple function p can be obtained that represents or approximates the given table or function f. The representation p can always be taken to be a polynomial, although many other types of simple functions can also be used. Once a simple function p has been obtained, it can be used in place of f in many situations. For example, the integral of f could be estimated by the integral of p, and the latter should generally be easier to evaluate.

In many situations, a polynomial solution to the problems outlined above will be unsatisfactory from a practical point of view, and other classes of functions must be considered. In this book, one other class of versatile functions is discussed: the spline functions (see Chapter 9). The present chapter concerns polynomials exclusively, and Chapter 12 discusses general linear families of functions, of which splines and polynomials are important examples.

The obvious way in which a polynomial can *fail* as a practical solution to one of the preceding problems is that its degree may be unreasonably high. For instance, if the table considered contains 1,000 entries, a polynomial of degree 999 may be required to represent it. Polynomials also may have the surprising defect of being highly oscillatory. If the table is precisely represented by a polynomial p, then $p(x_i) = y_i$ for $0 \leq i \leq n$. For points other than the given x_i, however, $p(x)$ may be a very poor representation of the function from which the table arose. The example in Section 4.2 involving the Runge function illustrates this phenomenon.

Polynomial Interpolation

We begin again with a table of values:

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

and assume that the x_i's form a set of $n + 1$ distinct points. The table represents $n + 1$ points in the Cartesian plane, and we want to find a polynomial curve that passes through all points. Thus, we seek to determine a polynomial that is defined for *all* x, and takes on the corresponding values of y_i for each of the $n + 1$ distinct x_i's in this table. A polynomial p for which $p(x_i) = y_i$ when $0 \leq i \leq n$ is said to **interpolate** the table. The points x_i are called **nodes**.

Consider the first and simplest case, $n = 0$. Here, a constant function solves the problem. In other words, the polynomial p of degree 0 defined by the equation $p(x) = y_0$ reproduces the one-node table.

The next simplest case occurs when $n = 1$. Since a straight line can be passed through two points, a linear function is capable of solving the problem. Explicitly, the polynomial p defined by

$$\begin{aligned} p(x) &= \left(\frac{x - x_1}{x_0 - x_1}\right) y_0 + \left(\frac{x - x_0}{x_1 - x_0}\right) y_1 \\ &= y_0 + \left(\frac{y_1 - y_0}{x_1 - x_0}\right)(x - x_0) \end{aligned}$$

is of first degree (at most) and reproduces the table. That means (in this case) that $p(x_0) = y_0$ and $p(x_1) = y_1$, as is easily verified. This p is used for **linear interpolation**.

EXAMPLE 1 Find the polynomial of least degree that interpolates this table:

x	1.4	1.25
y	3.7	3.9

Solution By the equation above, the polynomial that is sought is

$$\begin{aligned} p(x) &= \left(\frac{x-1.25}{1.4-1.25}\right)3.7 + \left(\frac{x-1.4}{1.25-1.4}\right)3.9 \\ &= 3.7 + \left(\frac{3.9-3.7}{1.25-1.4}\right)(x-1.4) \\ &= 3.7 - \frac{4}{3}(x-1.4) \end{aligned}$$

■

As we can see, an interpolating polynomial can be written in a variety of forms; among these are those known as the Newton form and the Lagrange form. The Newton form is probably the most convenient and efficient; however, conceptually, the Lagrange form has several advantages. We begin with the Lagrange form, since it may be easier to understand.

Interpolating Polynomial: Lagrange Form

Suppose that we wish to interpolate arbitrary functions at a set of fixed nodes $x_0, x_1, \ldots, x_n$. We first define a system of $n+1$ special polynomials of degree n known as **cardinal polynomials** in interpolation theory. These are denoted by $\ell_0, \ell_1, \ldots, \ell_n$ and have the property

$$\ell_i(x_j) = \delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

Once these are available, we can interpolate *any* function f by the **Lagrange form of the interpolation polynomial**:

$$p_n(x) = \sum_{i=0}^{n} \ell_i(x) f(x_i) \tag{1}$$

polynomial ← $p_n(x)$; f → original (unknown) function

This function p_n, being a linear combination of the polynomials ℓ_i, is itself a polynomial of degree at most n. Furthermore, when we evaluate p_n at x_j, we get $f(x_j)$:

$$p_n(x_j) = \sum_{i=0}^{n} \ell_i(x_j) f(x_i) = \ell_j(x_j) f(x_j) = f(x_j)$$

Thus, p_n is the interpolating polynomial for the function f at nodes $x_0, x_1, \ldots, x_n$. It remains now only to write the formula for the **cardinal polynomial** ℓ_i, which is

$$\ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j}\right) \qquad (0 \leq i \leq n) \tag{2}$$

This formula indicates that $\ell_i(x)$ is the product of n linear factors:

$$\ell_i(x) = \left(\frac{x - x_0}{x_i - x_0}\right)\left(\frac{x - x_1}{x_i - x_1}\right) \cdots \left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right)\left(\frac{x - x_{i+1}}{x_i - x_{i+1}}\right) \cdots \left(\frac{x - x_n}{x_i - x_n}\right)$$

(The denominators are just numbers; the variable x occurs only in the numerators.) Thus, ℓ_i is a polynomial of degree n. Notice that when $\ell_i(x)$ is evaluated at $x = x_i$, each factor in the preceding equation becomes 1. Hence, $\ell_i(x_i) = 1$. But when $\ell_i(x)$ is evaluated at any *other* node, say, x_j, one of the factors in the above equation will be 0, and $\ell_i(x_j) = 0$, for $i \neq j$.

Figure 4.1 shows the first few Lagrange cardinal polynomials: $\ell_0(x)$, $\ell_1(x)$, $\ell_2(x)$, $\ell_3(x)$, $\ell_4(x)$, and $\ell_5(x)$.

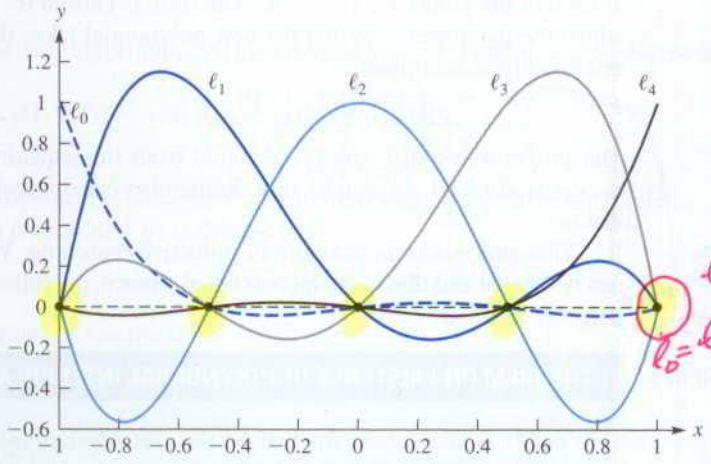

FIGURE 4.1 First few Lagrange cardinal polynomials

EXAMPLE 2 Write out the cardinal polynomials appropriate to the problem of interpolating the following table, and give the Lagrange form of the interpolating polynomial:

x	$\frac{1}{3}$	$\frac{1}{4}$	1
$f(x)$	2	−1	7

Solution Using Equation (2), we have

$$\ell_0(x) = \frac{(x - \frac{1}{4})(x - 1)}{(\frac{1}{3} - \frac{1}{4})(\frac{1}{3} - 1)} = -18\left(x - \frac{1}{4}\right)(x - 1)$$

$$\ell_1(x) = \frac{(x - \frac{1}{3})(x - 1)}{(\frac{1}{4} - \frac{1}{3})(\frac{1}{4} - 1)} = 16\left(x - \frac{1}{3}\right)(x - 1)$$

$$\ell_2(x) = \frac{(x - \frac{1}{3})(x - \frac{1}{4})}{(1 - \frac{1}{3})(1 - \frac{1}{4})} = 2\left(x - \frac{1}{3}\right)\left(x - \frac{1}{4}\right)$$

Therefore, the interpolating polynomial in Lagrange's form is

$$p_2(x) = -36\left(x - \frac{1}{4}\right)(x - 1) - 16\left(x - \frac{1}{3}\right)(x - 1) + 14\left(x - \frac{1}{3}\right)\left(x - \frac{1}{4}\right) \quad \blacksquare$$

Existence of Interpolating Polynomial

The Lagrange interpolation formula proves the existence of an interpolating polynomial for any table of values. There is another constructive way of proving this fact, and it leads to a different formula.

Suppose that we have succeeded in finding a polynomial p that reproduces *part* of the table. Assume, say, that $p(x_i) = y_i$ for $0 \leq i \leq k$. We shall attempt to add to p another term that will enable the new polynomial to reproduce one more entry in the table. We consider

$$p(x) + c(x - x_0)(x - x_1) \cdots (x - x_k)$$

where c is a constant to be determined. This is surely a polynomial. It also reproduces the first k points in the table because p itself does so, and the added portion takes the value 0 at each of the points $x_0, x_1, \ldots, x_k$. (Its form is chosen for precisely this reason.) Now we adjust the parameter c so that the new polynomial takes the value y_{k+1} at x_{k+1}. Imposing this condition, we obtain

$$p(x_{k+1}) + c(x_{k+1} - x_0)(x_{k+1} - x_1) \cdots (x_{k+1} - x_k) = y_{k+1}$$

The proper value of c can be obtained from this equation because none of the factors $x_{k+1} - x_i$, for $0 \leq i \leq k$, can be zero. Remember our original assumption that the x_i's are all distinct.

This analysis is an example of inductive reasoning. We have shown that the process can be started and that it can be continued. Hence, the following formal statement has been partially justified:

■ THEOREM 1

THEOREM ON EXISTENCE OF POLYNOMIAL INTERPOLATION

If points $x_0, x_1, \ldots, x_n$ are distinct, then for arbitrary real values $y_0, y_1, \ldots, y_n$, there is a unique polynomial p of degree at most n such that $p(x_i) = y_i$ for $0 \leq i \leq n$.

Two parts of this formal statement must still be established. First, the degree of the polynomial increases by at most 1 in each step of the inductive argument. At the beginning, the degree was at most 0, so at the end, the degree is at most n.

Second, we establish the uniqueness of the polynomial p. Suppose that another polynomial q claims to accomplish what p does; that is, q is also of degree at most n and satisfies $q(x_i) = y_i$ for $0 \leq i \leq n$. Then the polynomial $p - q$ is of degree at most n and takes the value 0 at $x_0, x_1, \ldots, x_n$. Recall, however, that a *nonzero* polynomial of degree n can have at most n roots. We conclude that $p = q$, which establishes the uniqueness of p.

Interpolating Polynomial: Newton Form

In Example 2, we found the Lagrange form of the interpolating polynomial:

$$p_2(x) = -36\left(x - \frac{1}{4}\right)(x - 1) - 16\left(x - \frac{1}{3}\right)(x - 1) + 14\left(x - \frac{1}{3}\right)\left(x - \frac{1}{4}\right)$$

It can be simplified to

$$p_2(x) = -\frac{79}{6} + \frac{349}{6}x - 38x^2$$

We will now learn that this polynomial can be written in another form called the nested Newton form:

$$p_2(x) = 2 + \left(x - \frac{1}{3}\right)\left[36 + \left(x - \frac{1}{4}\right)(-38)\right]$$

It involves the fewest arithmetic operations and is recommended for evaluating $p_2(x)$. It can not be overemphasized that the Newton and Lagrange forms are just two different derivations for precisely the same polynomial. The Newton form has the advantage of easy extensibility to accommodate additional data points.

The preceding discussion provides a method for constructing an interpolating polynomial. The method is known as the **Newton algorithm**, and the resulting polynomial is the **Newton form of the interpolating polynomial**.

EXAMPLE 3 Using the Newton algorithm, find the interpolating polynomial of least degree for this table:

x	0	1	−1	2	−2
y	−5	−3	−15	39	−9

Solution In the construction, five successive polynomials will appear; these are labeled p_0, p_1, p_2, p_3, and p_4. The polynomial p_0 is defined to be

$$p_0(x) = -5$$

The polynomial p_1 has the form

$$p_1(x) = p_0(x) + c(x - x_0) = -5 + c(x - 0)$$

The interpolation condition placed on p_1 is that $p_1(1) = -3$. Therefore, we have $-5 + c(1 - 0) = -3$. Hence, $c = 2$, and p_1 is

$$p_1(x) = -5 + 2x$$

The polynomial p_2 has the form

$$p_2(x) = p_1(x) + c(x - x_0)(x - x_1) = -5 + 2x + cx(x - 1)$$

The interpolation condition placed on p_2 is that $p_2(-1) = -15$. Hence, we have $-5 + 2(-1) + c(-1)(-1 - 1) = -15$. This yields $c = -4$, so

$$p_2(x) = -5 + 2x - 4x(x - 1)$$

The remaining steps are similar, and the final result is the Newton form of the interpolating polynomial:

$$p_4(x) = -5 + 2x - 4x(x - 1) + 8x(x - 1)(x + 1) + 3x(x - 1)(x + 1)(x - 2)$$ ■

Later, we will develop a better algorithm for constructing the Newton interpolating polynomial. Nevertheless, the method just explained is a systematic one and involves very little computation. An important feature to notice is that each new polynomial in the algorithm is obtained from its predecessor by adding a new term. Thus, at the end, the final polynomial exhibits all the previous polynomials as constituents.

Nested Form

Before continuing, let us rewrite the Newton form of the interpolating polynomial for efficient evaluation.

EXAMPLE 4 Write the polynomial p_4 of Example 3 in *nested* form and use it to evaluate $p_4(3)$.

Solution We write p_4 as

$$p_4(x) = -5 + x(2 + (x-1)(-4 + (x+1)(8 + (x-2)3)))$$

Therefore,

$$\begin{aligned} p_4(3) &= -5 + 3(2 + 2(-4 + 4(8 + 3))) \\ &= 241 \end{aligned}$$

Another solution, also in nested form, is

$$p_4(x) = -5 + x(4 + x(-7 + x(2 + 3x)))$$

from which we obtain

$$p_4(3) = -5 + 3(4 + 3(-7 + 3(2 + 3 \cdot 3))) = 241$$

This form is obtained by expanding and systematic factoring of the original polynomial. It is also known as a **nested form** and its evaluation is by **nested multiplication**. ■

To describe nested multiplication in a formal way (so that it can be translated into a code), consider a general polynomial in the Newton form. It might be

$$\begin{aligned} p(x) = a_0 &+ a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots \\ &+ a_n(x - x_0)(x - x_1) \cdots (x - x_{n-1}) \end{aligned}$$

The nested form of $p(x)$ is

$$\begin{aligned} p(x) &= a_0 + (x - x_0)(a_1 + (x - x_1)(a_2 + \cdots + (x - x_{n-1})a_n)) \cdots)) \\ &= (\cdots((a_n(x - x_{n-1}) + a_{n-1})(x - x_{n-2}) + a_{n-2}) \cdots)(x - x_0) + a_0 \end{aligned}$$

The **Newton interpolation polynomial** can be written succinctly as

$$p_n(x) = \sum_{i=0}^{n} a_i \prod_{j=0}^{i-1} (x - x_j) \tag{3}$$

Here $\prod_{j=0}^{-1}(x - x_j)$ is interpreted to be 1. Also, we can write it as

$$p_n(x) = \sum_{i=0}^{n} a_i \pi_i(x)$$

where

$$\pi_i(x) = \prod_{j=0}^{i-1} (x - x_j) \tag{4}$$

Figure 4.2 shows the first few Newton polynomials: $\pi_0(x)$, $\pi_1(x)$, $\pi_2(x)$, $\pi_3(x)$, $\pi_4(x)$, and $\pi_5(x)$.

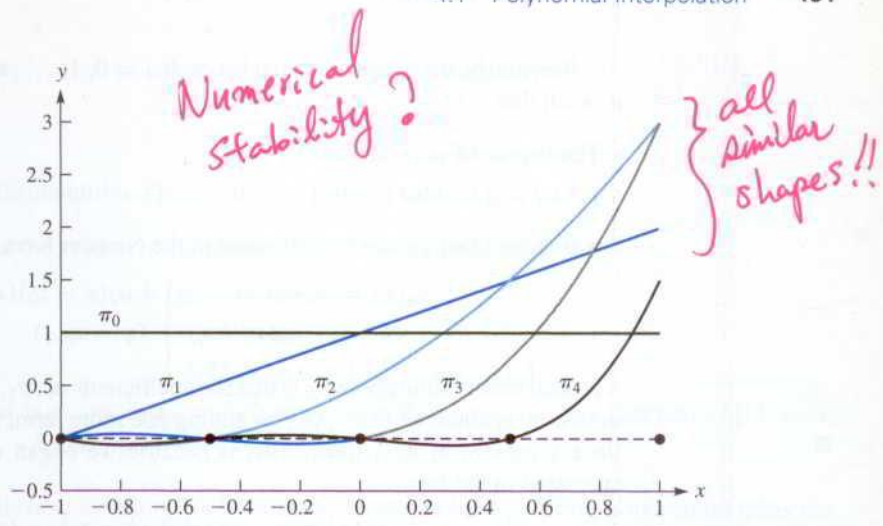

FIGURE 4.2 First few Newton polynomials

In evaluating $p(t)$ for a given numerical value of t, we naturally start with the innermost parentheses, forming successively the following quantities:

$$\begin{aligned} v_0 &= a_n \\ v_1 &= v_0(t - x_{n-1}) + a_{n-1} \\ v_2 &= v_1(t - x_{n-2}) + a_{n-2} \\ &\vdots \\ v_n &= v_{n-1}(t - x_0) + a_0 \end{aligned}$$

The quantity v_n is now $p(t)$. In the following pseudocode, a subscripted variable is not needed for v_i. Instead, we can write

```
integer i, n;   real t, v;   real array (a_i)_{0:n}, (x_i)_{0:n}
v ← a_n
for i = n − 1 to 0 step −1 do
    v ← v(t − x_i) + a_i
end for
```

Here, the array $(a_i)_{0:n}$ contains the $n+1$ coefficients of the Newton form of the interpolating polynomial (3) of degree at most n, and the array $(x_i)_{0:n}$ contains the $n+1$ nodes x_i.

Calculating Coefficients a_i Using Divided Differences

We turn now to the problem of determining the coefficients $a_0, a_1, \ldots, a_n$ efficiently. Again we start with a table of values of a function f:

x	x_0	x_1	x_2	$\cdots$	x_n
$f(x)$	$f(x_0)$	$f(x_1)$	$f(x_2)$	$\cdots$	$f(x_n)$

The points $x_0, x_1, \ldots, x_n$ are assumed to be distinct, but no assumption is made about their positions on the real line.

Vandermonde Matrix

Another view of interpolation is that for a given set of $n + 1$ data points (x_0, y_0), (x_1, y_1), $\ldots$, (x_n, y_n), we want to express an interpolating function $f(x)$ as a linear combination of a set of *basis functions* $\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n$ so that

$$f(x) \approx c_0\varphi_0(x) + c_1\varphi_1(x) + c_2\varphi_2(x) + \cdots + c_n\varphi_n(x)$$

Here the coefficients $c_0, c_1, c_2, \ldots, c_n$ are to be determined. We want the function f to interpolate the data (x_i, y_i). This means that we have linear equations of the form

$$f(x_i) = c_0\varphi_0(x_i) + c_1\varphi_1(x_i) + c_2\varphi_2(x_i) + \cdots + c_n\varphi_n(x_i) = y_i$$

for each $i = 0, 1, 2, \ldots, n$. This is a system of linear equations

$$\mathbf{Ac} = \mathbf{y}$$

Here, the entries in the coefficient matrix $\mathbf{A}$ are given by $a_{ij} = \varphi_j(x_i)$, which is the value of the jth basis function evaluated at the ith data point. The right-hand side vector $\mathbf{y}$ contains the known data values y_i, and the components of the vector $\mathbf{c}$ are the unknown coefficients c_i. Systems of linear equations are discussed in Chapters 7 and 8.

Polynomials are the simplest and most common basis functions. The natural basis for $\mathbb{P}_n$ consists of the *monomials*

$$\varphi_0(x) = 1, \varphi_1(x) = x, \varphi_2(x) = x^2, \ldots, \varphi_n(x) = x^n$$

Figure 4.3 shows the first few monomials: 1, x, x^2, x^3, x^4, and x^5.

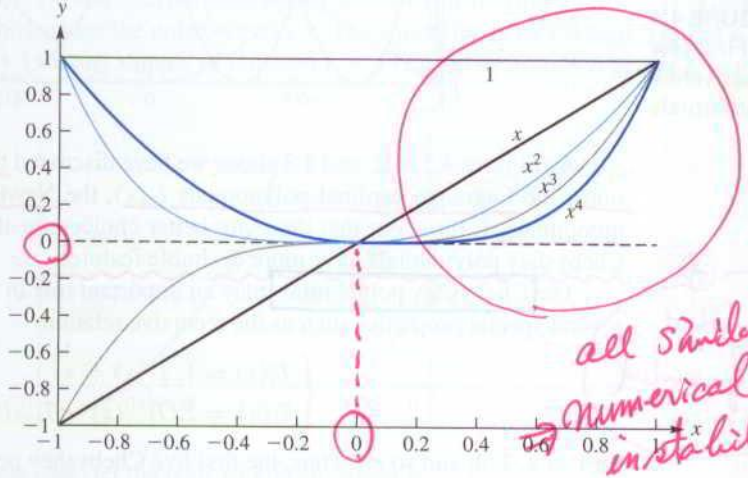

FIGURE 4.3 First few monomials

Consequently, a given polynomial p_n has the form

$$p_n(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n$$

The corresponding linear system $\mathbf{Ac} = \mathbf{y}$ has the form

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

The coefficient matrix is called a *Vandermonde matrix*. It can be shown that this matrix is nonsingular provided that the points $x_0, x_1, x_2, \ldots, x_n$ are distinct. So we can, in theory, solve the system for the polynomial interpolant. Although the Vandermonde matrix is nonsingular, it is ill-conditioned as n increases. For large n, the monomials are less distinguishable from one another, as shown in Figure 4.4. Moreover, the columns of the Vandermonde become nearly linearly dependent in this case. High-degree polynomials often oscillate wildly and are highly sensitive to small changes in the data.

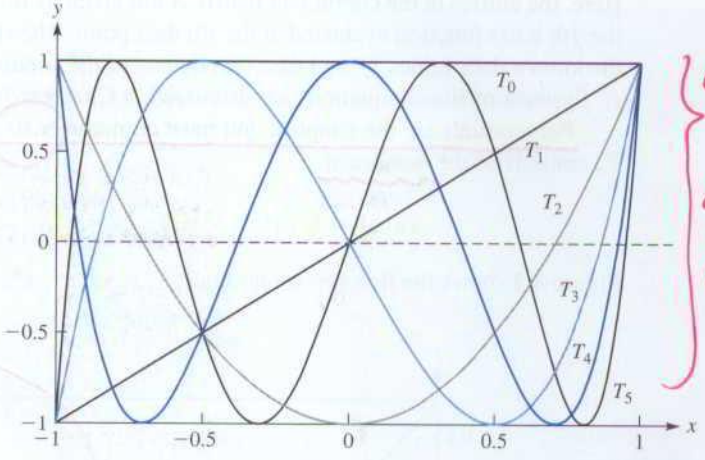

FIGURE 4.4 First few Chebyshev polynomials

As Figures 4.1, 4.2, and 4.3 show, we have discussed three choices for the basis functions: the Lagrange cardinal polynomials $\ell_i(x)$, the Newton polynomials $\pi_i(x)$, and the monomials. It turns out that there are better choices for the basis functions; namely, the Chebyshev polynomials have more desirable features.

The Chebyshev polynomials play an important role in mathematics because they have several special properties such as the recursive relation

$$\begin{cases} T_0(x) = 1, T_1(x) = x \\ T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x) \end{cases}$$

for $i = 2, 3, 4$, and so on. Thus, the first five Chebyshev polynomials are

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_2(x) = 2x^2 - 1, \quad T_3(x) = 4x^3 - 3x$$
$$T_4(x) = 8x^4 - 8x^2 + 1, \quad T_5(x) = 16x^5 - 20x^3 + 5x$$

These curves for these polynomials, as is shown in Figure 4.4, are quite different from one another. The Chebyshev polynomials are usually employed on the interval $[-1, 1]$.

With changes of variable, they can be used on any interval, but the results will be more complicated.

One of the important properties of the Chebyshev polynomials is the equal oscillation property. Notice in Figure 4.4 that successive extreme points of the Chebyshev polynomials are equal in magnitude and alternate in sign. This property tends to distribute the error uniformly when the Chebyshev polynomials are used as the basis functions. In polynomial interpolation for continuous functions, it is particularly advantageous to select as the interpolation points the roots or the extreme points of a Chebyshev polynomial. This causes the maximum error over the interval of interpolation to be minimized. An example of this is given in Section 4.2. In Section 12.2, we discuss Chebyshev polynomials in more detail.

Inverse Interpolation

A process called **inverse interpolation** is often used to approximate an inverse function. Suppose that values $y_i = f(x_i)$ have been computed at $x_0, x_1, \ldots, x_n$. Using the table

y	y_0	y_1	$\cdots$	y_n
x	x_0	x_1	$\cdots$	x_n

we form the interpolation polynomial

$$p(y) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1} (y - y_j)$$

The original relationship, $y = f(x)$, has an inverse, under certain conditions. This inverse is being approximated by $x = p(y)$. Procedures *Coef* and *Eval* can be used to carry out the inverse interpolation by reversing the arguments x and y in the calling sequence for *Coef*.

Inverse interpolation can be used to find where a given function f has a root or *zero*. This means inverting the equation $f(x) = 0$. We propose to do this by creating a table of values $(f(x_i), x_i)$ and interpolating with a polynomial, p. Thus, $p(y_i) = x_i$. The points x_i should be chosen near the unknown root, r. The approximate root is then given by $r \approx p(0)$. See Figure 4.5 for an example of function $y = f(x)$ and its inverse function $x = g(y)$ with the root $r = g(0)$.

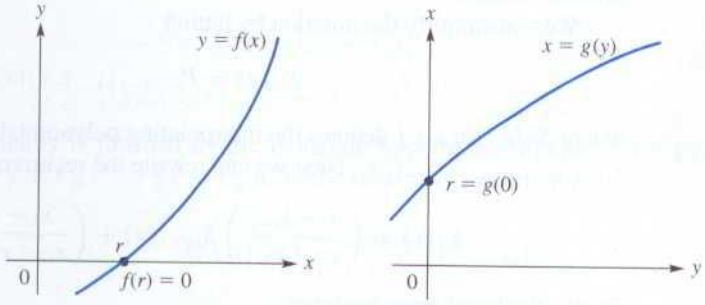

FIGURE 4.5 Function $y = f(x)$ and inverse function $x = g(y)$

EXAMPLE 9 For a concrete case, let the table of known values be

y	−0.57892 00	−0.36263 70	−0.18491 60	−0.03406 42	0.09698 58
x	1.0	2.0	3.0	4.0	5.0

Find the inverse interpolation polynomial.

Clearly, $p_2(x_0) = f(x_0)$, $p_2(x_1) = f(x_1)$, and $p_2(x_2) = f(x_2)$. Next, we form the divided-difference table:

$$\begin{array}{c|c|c|c} x_0 & f(x_0) & & \\ & & f[x_0, x_1] & \\ x_1 & f(x_1) & & f[x_0, x_1, x_2] \\ & & f[x_1, x_2] & \\ x_2 & f(x_2) & & \end{array}$$

Using the divided-difference entries from the top diagonal, we have

$$p_n(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

Again, it can be easily shown that $p_2(x_0) = f(x_0)$, $p_2(x_1) = f(x_1)$, and $p_2(x) = f(x_2)$.

(4) We can use inverse polynomial interpolation to find an approximate value of a root r of the equation $f(x) = 0$ from a table of values (x_i, y_i) for $1 \leqq i \leqq n$. Here we are assuming that the table values are in the vicinity of this zero of the function f. Flipping the table values, we use the reversed table values (y_i, x_i) to determine the interpolating polynomial called $p_n(y)$. Now evaluating it at 0, we find a value that approximates the desired zero, namely, $r \approx p_n(0)$ and $f(p_n(0)) \approx f(r) = 0$.

(5) Other advanced polynomial interpolation methods discussed are **Neville's algorithm** and **bivariate function interpolation**.

Problems 4.1

1. Use the Lagrange interpolation process to obtain a polynomial of least degree that assumes these values:

x	0	2	3	4
y	7	11	28	63

2. (Continuation) Rearrange the points in the table of the preceding problem and find the Newton form of the interpolating polynomial. Show that the polynomials obtained are identical, although their forms may differ.

3. For the four interpolation nodes $-1, 1, 3, 4$, what are the ℓ_i Functions (2) required in the Lagrange interpolation procedure? Draw the graphs of these four functions to show their essential properties.

4. Verify that the polynomials

$$p(x) = 5x^3 - 27x^2 + 45x - 21, \qquad q(x) = x^4 - 5x^3 + 8x^2 - 5x + 3$$

interpolate the data

x	1	2	3	4
y	2	1	6	47

and explain why this does not violate the uniqueness part of the theorem on existence of polynomial interpolation.

10. For Example 8, compare the results from your code with that in the text. Redo using linear interpolation based on the ten equidistant points. How do the errors compare at intermediate points? Plot curves to visualize the difference between linear interpolation and a higher-degree polynomial interpolation.

11. Use mathematical software such as Matlab, Maple, or Mathematica to find an interpolation polynomial for the points (0, 0), (1, 1), (2, 2.001), (3, 3), (4, 4), (5, 5). Evaluate the polynomial at the point $x = 14$ or $x = 20$ to show that slight roundoff errors in the data can lead to suspicious results in extrapolation.

12. Use symbolic mathematical software such as Matlab, Maple, or Mathematica to generate the interpolation polynomial for the data points in Example 3. Plot the polynomial and the data points.

13. (Continuation.) Repeat these instructions using Example 7.

14. Carry out the details in Example 8 by writing a computer program that plots the data points and the curve for the interpolation polynomial.

15. (Continuation.) Repeat the instructions for Problem 14 on Example 9.

16. Using mathematical software, carry out the details and verify the results in the introductory example to this chapter.

17. (**Padé interpolation**) Find a rational function of the form

$$g(x) = \frac{a + bx}{1 + cx}$$

that interpolates the function $f(x) = \arctan(x)$ at the points $x_0 = 1$, $x_1 = 2$, and $x_2 = 3$. On the same axes, plot the graphs of f and g, using dashed and dotted lines, respectively.

4.2 Errors in Polynomial Interpolation

When a function f is approximated on an interval $[a, b]$ by means of an interpolating polynomial p, the discrepancy between f and p will (theoretically) be zero at each node of interpolation. A natural expectation is that the function f will be well approximated at all intermediate points and that as the number of nodes increases, this agreement will become better and better.

In the history of numerical mathematics, a severe shock occurred when it was realized that this expectation was ill-founded. Of course, if the function being approximated is not required to be continuous, then there may be no agreement at all between $p(x)$ and $f(x)$ except at the nodes.

EXAMPLE 1 Consider these five data points: (0, 8), (1, 12), (3, 2), (4, 6), (8, 0). Construct and plot the interpolation polynomial using the two outermost points. Repeat this process by adding one additional point at a time until all the points are included. What conclusions can you draw?

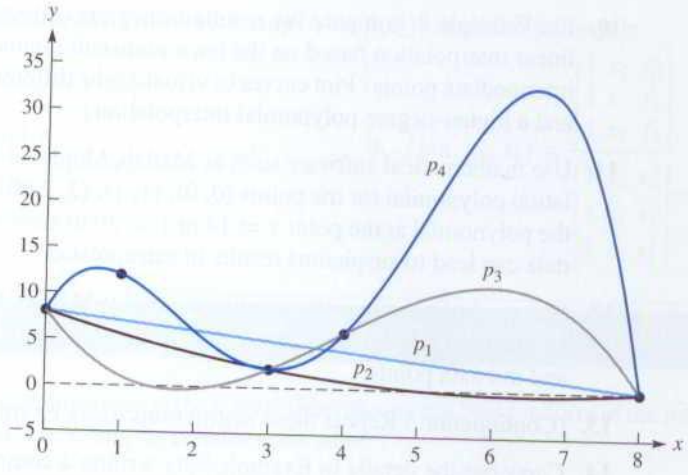

FIGURE 4.6 Interpolant polynomials over data points

Solution The first interpolation polynomial is the line between the outermost points (0, 8) and (8, 0). Then we added the points (3, 2), (4, 5), and (1, 12) in that order and plotted a curve for each additional point. All of these polynomials are shown in Figure 4.6. We were hoping for a smooth curve going through these points without wide fluctuations, but this did not happen. (Why?) It may seem counterintuitive, but as we added more points, the situation became worse instead of better! The reason for this comes from the nature of high-degree polynomials. A polynomial of degree n has n zeros. If all of these zero points are real, then the curve crosses the x-axis n times. The resulting curve must make many turns for this to happen, resulting in wild oscillations. In Chapter 9, we discuss fitting the data points with spline curves. ■

Dirichlet Function

As a pathological example, consider the so-called **Dirichlet function** f, defined to be 1 at each irrational point and 0 at each rational point. If we choose nodes that are rational numbers, then $p(x) \equiv 0$ and $f(x) - p(x) = 0$ for all rational values of x, but $f(x) - p(x) = 1$ for all irrational values of x.

However, if the function f is well-behaved, can we not assume that the differences $|f(x) - p(x)|$ will be small when the number of interpolating nodes is large? The answer is still *no*, even for functions that possess continuous derivatives of all orders on the interval!

Runge Function

A specific example of this remarkable phenomenon is provided by the **Runge function**:

$$f(x) = (1 + x^2)^{-1} \tag{1}$$

on the interval $[-5, 5]$. Let p_n be the polynomial that interpolates this function at $n + 1$ equally spaced points on the interval $[-5, 5]$, including the endpoints. Then

$$\lim_{n \to \infty} \max_{-5 \leq x \leq 5} |f(x) - p_n(x)| = \infty$$

Thus, the effect of requiring the agreement of f and p_n at more and more points is to *increase* the error at nonnodal points, and the error actually increases beyond all bounds!

The moral of this example, then, is that polynomial interpolation of high degree with many nodes is a risky operation; the resulting polynomials may be very unsatisfactory as representations of functions unless the set of nodes is chosen with great care.

The reader can easily observe the phenomenon just described by using the pseudocodes already developed in this chapter. See Computer Problem 4.2.1 for a suggested numerical experiment. In a more advanced study of this topic, it would be shown that the divergence of the polynomials can often be ascribed to the fact that the nodes are equally spaced. Again, contrary to intuition, equally distributed nodes are usually a very poor choice in interpolation. A much better choice for $n + 1$ nodes in $[-1, 1]$ is the set of **Chebyshev nodes**:

$$x_i = \cos\left[\left(\frac{2i+1}{2n+2}\right)\pi\right] \qquad (0 \leq i \leq n)$$

The corresponding set of nodes on an arbitrary interval $[a, b]$ would be derived from a linear mapping to obtain

$$x_i = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left[\left(\frac{2i+1}{2n+2}\right)\pi\right] \qquad (0 \leq i \leq n)$$

Notice that these nodes are numbered from right to left. Since the theory does not depend on any particular ordering of the nodes, this is not troublesome.

A simple graph illustrates this phenomenon best. Again, consider Equation (1) on the interval $[-5, 5]$. First, we select nine equally spaced nodes and use routines *Coef* and *Eval* with an automatic plotter to graph p_8. As shown in Figure 4.7, the resulting curve assumes negative values, which, of course, $f(x)$ does not have! Adding more equally spaced nodes—and thereby obtaining a higher-degree polynomial—only makes matters worse with wilder oscillations. In Figure 4.8, nine Chebyshev nodes are used, and the resulting polynomial curve is smoother. However, cubic splines (discussed in Chapter 9) produce an even better curve fit.

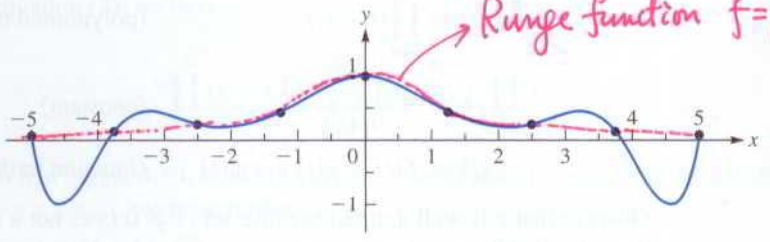

FIGURE 4.7
Polynomial interpolant with nine equally spaced nodes

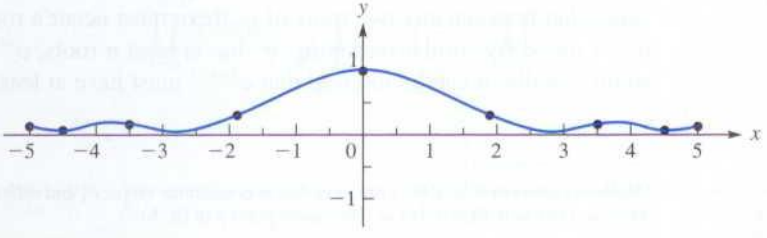

FIGURE 4.8
Polynomial interpolant with nine Chebyshev nodes

11. (**Student research project**) Explore the topic of interpolation of multivariate scattered data, such as arise in geophysics and other areas.

12. Use mathematical software such as found in Matlab, Maple, or Mathematica to reproduce Figures 4.7 and 4.8.

13. Use symbolic mathematical software such as Maple or Mathematica to generate the interpolation polynomial for the data points in Example 2. Plot the polynomial and the data points.

14. Use graphical software to plot four or five points that happen to generate an interpolating polynomial that exhibits a great deal of oscillations. This piece of software should let you use your computer mouse to *click* on three or four points that visually appear to be part of a smooth curve. Next it uses Newton's interpolating polynomial to sketch the curve through these points. Then add another point that is somewhat remote from the curve and refit all the points. Repeat, adding other points. After a few points have been added in this way, you should have evidence that polynomials can oscillate wildly.

4.3 Estimating Derivatives and Richardson Extrapolation

A numerical experiment outlined in Chapter 1 (at the end of Section 1.1, p. 10) showed that determining the derivative of a function f at a point x is not a trivial numerical problem. Specifically, if $f(x)$ can be computed with only n digits of precision, it is difficult to calculate $f'(x)$ numerically with n digits of precision. This difficulty can be traced to the subtraction between quantities that are nearly equal. In this section, several alternatives are offered for the numerical computation of $f'(x)$ and $f''(x)$.

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

First-Derivative Formulas via Taylor Series

First, consider again the obvious method based on the definition of $f'(x)$. It consists of selecting one or more small values of h and writing

$$f'(x) \approx \frac{1}{h}[f(x+h) - f(x)] \tag{1}$$

What error is involved in this formula? To find out, use Taylor's Theorem from Section 1.2:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(\xi)$$

Rearranging this equation gives

$$f'(x) = \frac{1}{h}[f(x+h) - f(x)] - \frac{1}{2}hf''(\xi) \tag{2}$$

Hence, we see that approximation (1) has error term $-\frac{1}{2}hf''(\xi) = \mathcal{O}(h)$, where ξ is in the interval having endpoints x and $x+h$.

Equation (2) shows that in general, as $h \to 0$, the difference between $f'(x)$ and the estimate $h^{-1}[f(x+h) - f(x)]$ approaches zero at the same rate that h does—that is, $\mathcal{O}(h)$. Of course, if $f''(x) = 0$, then the error term will be $\frac{1}{6}h^2 f'''(\gamma)$, which converges to zero somewhat faster at $\mathcal{O}(h^2)$. But usually, $f''(x)$ is not zero.

Equation (2) gives the **truncation error** for this numerical procedure, namely, $-\frac{1}{2}hf''(\xi)$. This error is present even if the calculations are performed with *infinite* precision; it is due to our imitating the mathematical limit process by means of an approximation formula. Additional (and worse) errors must be expected when calculations are performed on a computer with finite word length.

EXAMPLE 1 In Section 1.1, the program named *First* used the one-sided rule (1) to approximate the first derivative of the function $f(x) = \sin x$ at $x = 0.5$. Explain what happens when a large number of iterations are performed, say $n = 50$.

Solution There is a total loss of all significant digits! When we examine the computer output closely, we find that, in fact, a good approximation $f'(0.5) \approx 0.87758$ was found, but it deteriorated as the process continued. This was caused by the subtraction of two nearly equal quantities $f(x+h)$ and $f(x)$, resulting in a loss of significant digits as well as a magnification of this effect from dividing by a small value of h. We need to stop the iterations sooner! When to stop an iterative process is a common question in numerical algorithms. In this case, one can monitor the iterations to determine when they settle down, namely, when two successive ones are within a prescribed tolerance. Alternatively, we can use the truncation error term. If we want six significant digits of accuracy in the results, we set

$$\left| -\frac{1}{2}hf''(\xi) \right| \leq \frac{1}{2}4^{-n} < \frac{1}{2}10^{-6}$$

since $|f''(x)| < 1$ and $h = 1/4^n$. We find $n > 6/\log 4 \approx 9.97$. So we should stop after about ten steps in the process. (The least error of 3.1×10^{-9} was found at iteration 14.) ■

As we saw in Newton's method (Chapter 3) and will see in the Romberg method (Chapter 5), it is advantageous to have the convergence of numerical processes occur with higher powers of some quantity approaching zero. In the present situation, we want an approximation to $f'(x)$ in which the error behaves like $\mathcal{O}(h^2)$. One such method is easily obtained with the aid of the following two Taylor series:

$$\begin{cases} f(x+h) = f(x) + hf'(x) + \frac{1}{2!}h^2 f''(x) + \frac{1}{3!}h^3 f'''(x) + \frac{1}{4!}h^4 f^{(4)}(x) + \cdots \\ f(x-h) = f(x) - hf'(x) + \frac{1}{2!}h^2 f''(x) - \frac{1}{3!}h^3 f'''(x) + \frac{1}{4!}h^4 f^{(4)}(x) - \cdots \end{cases} \tag{3}$$

By subtraction, we obtain

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{2}{3!}h^3 f'''(x) + \frac{2}{5!}h^5 f^{(5)}(x) + \cdots$$

This leads to a very important formula for approximating $f'(x)$:

$$f'(x) = \frac{1}{2h}[f(x+h) - f(x-h)] - \frac{h^2}{3!}f'''(x) - \frac{h^4}{5!}f^{(5)}(x) - \cdots \tag{4}$$

Expressed otherwise,

$$f'(x) \approx \frac{1}{2h}[f(x+h) - f(x-h)] \tag{5}$$

with an error whose leading term is $-\frac{1}{6}h^2 f'''(x)$, which makes it $\mathcal{O}(h^2)$.

By using Taylor's Theorem with its error term, we could have obtained the following two expressions:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(\xi_1)$$

$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(\xi_2)$$

Then the subtraction would lead to

$$f'(x) = \frac{1}{2h}[f(x+h) - f(x-h)] - \frac{1}{6}h^2\left[\frac{f'''(\xi_1) + f'''(\xi_2)}{2}\right]$$

The error term here can be simplified by the following reasoning: The expression $\frac{1}{2}[f'''(\xi_1) + f'''(\xi_2)]$ is the average of two values of f''' on the interval $[x-h, x+h]$. It therefore lies between the least and greatest values of f''' on this interval. If f''' is continuous on this interval, then this average value is assumed at some point ξ. Hence, the formula with its error term can be written as

$$f'(x) = \frac{1}{2h}[f(x+h) - f(x-h)] - \frac{1}{6}h^2 f'''(\xi)$$

This is based on the sole assumption that f''' is continuous on $[x-h, x+h]$. This formula for numerical differentiation turns out to be very useful in the numerical solution of certain differential equations, as we shall see in Chapter 14 (on boundary value problems) and Chapter 15 (on partial differential equations).

EXAMPLE 2 Modify program *First* in Section 1.1 so that it uses the central difference formula (5) to approximate the first derivative of the function $f(x) = \sin x$ at $x = 0.5$.

Solution Using the truncation error term for the central difference formula (5), we set

$$\left|-\frac{1}{6}h^2 f'''(\xi)\right| \leq \frac{1}{6}4^{-2n} < \frac{1}{2}10^{-6}$$

or $n > (6 - \log 3)/\log 16 \approx 4.59$. We obtain a good approximation after about five iterations with this higher-order formula. (The least error of 3.6×10^{-12} was at step 9.) ■

Richardson Extrapolation

Returning now to Equation (4), we write it in a simpler form:

$$f'(x) = \frac{1}{2h}[f(x+h) - f(x-h)] + a_2h^2 + a_4h^4 + a_6h^6 + \cdots \tag{6}$$

in which the constants $a_2, a_4, \ldots$ depend on f and x. When such information is available about a numerical process, it is possible to use a powerful technique known as *Richardson extrapolation* to wring more accuracy out of the method. This procedure is now explained, using Equation (6) as our model.

Holding f and x fixed, we define a function of h by the formula

$$\varphi(h) = \frac{1}{2h}[f(x+h) - f(x-h)] \tag{7}$$

From Equation (6), we see that $\varphi(h)$ is an approximation to $f'(x)$ with error of order $\mathcal{O}(h^2)$. Our objective is to compute $\lim_{h\to 0}\varphi(h)$ because this is the quantity $f'(x)$ that we wanted

in the first place. If we select a function f and plot $\varphi(h)$ for $h = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$, then we get a graph (Computer Problem 4.3.5). Near zero, where we cannot actually calculate the value of φ from Equation (7), φ is approximately a quadratic function of h, since the higher-order terms from Equation (6) are negligible. Richardson extrapolation seeks to estimate the limiting value at 0 from some computed values of $\varphi(h)$ near 0. Obviously, we can take any convenient sequence h_n that converges to zero, calculate $\varphi(h_n)$ from Equation (7), and use these as approximations to $f'(x)$.

But something much more clever can be done. Suppose we compute $\varphi(h)$ for some h and then compute $\varphi(h/2)$. By Equation (6), we have

$$\varphi(h) = f'(x) - a_2h^2 - a_4h^4 - a_6h^6 - \cdots$$

$$\varphi\left(\frac{h}{2}\right) = f'(x) - a_2\left(\frac{h}{2}\right)^2 - a_4\left(\frac{h}{2}\right)^4 - a_6\left(\frac{h}{2}\right)^6 - \cdots$$

We can eliminate the dominant term in the error series by simple algebra. To do so, multiply the second equation by 4 and subtract it from the first equation. The result is

$$\varphi(h) - 4\varphi\left(\frac{h}{2}\right) = -3f'(x) - \frac{3}{4}a_4h^4 - \frac{15}{16}a_6h^6 - \cdots$$

We divide by -3 and rearrange this to get

$$\varphi\left(\frac{h}{2}\right) + \frac{1}{3}\left[\varphi\left(\frac{h}{2}\right) - \varphi(h)\right] = f'(x) + \frac{1}{4}a_4h^4 + \frac{5}{16}a_6h^6 + \cdots$$

This is a marvelous discovery. Simply by adding $\frac{1}{3}[\varphi(h/2) - \varphi(h)]$ to $\varphi(h/2)$, we have apparently improved the precision to $\mathcal{O}(h^4)$ because the error series that accompanies this new combination begins with $\frac{1}{4}a_4h^4$. Since h will be small, this is a dramatic improvement.

We can repeat this process by letting

$$\Phi(h) = \frac{4}{3}\varphi\left(\frac{h}{2}\right) - \frac{1}{3}\varphi(h)$$

Then we have from the previous derivation that

$$\Phi(h) = f'(x) + b_4h^4 + b_6h^6 + \cdots$$

$$\Phi\left(\frac{h}{2}\right) = f'(x) + b_4\left(\frac{h}{2}\right)^4 + b_6\left(\frac{h}{2}\right)^6 + \cdots$$

We can combine these equations to eliminate the first term in the error series

$$\Phi(h) - 16\Phi\left(\frac{h}{2}\right) = -15f'(x) + \frac{3}{4}b_6h^6 + \cdots$$

Hence, we have

$$\Phi\left(\frac{h}{2}\right) + \frac{1}{15}\left[\Phi\left(\frac{h}{2}\right) - \Phi(h)\right] = f'(x) - \frac{1}{20}b_6h^5 + \cdots$$

This is yet another apparent improvement in the precision to $\mathcal{O}(h^6)$. And now, to top it off, note that the same procedure can be repeated over and over again to *kill* higher and higher terms in the error. This is **Richardson extrapolation**.

FIGURE 4.13 Central difference: four nodes

This can be arranged in a form in which it probably should be computed with a principal term plus a correction or refining term:

$$f'(x) \approx \frac{1}{2h}[f(x+h) - f(x-h)] - \frac{1}{12h}\{f(x+2h) - 2[f(x+h) - f(x-h)] - f(x-2h)\} \tag{19}$$

The error term is $-\frac{1}{30}h^4 f^{(v)}(\xi) = \mathcal{O}(h^4)$.

Second-Derivative Formulas via Taylor Series

In the numerical solution of differential equations, it is often necessary to approximate second derivatives. We shall derive the most important formula for accomplishing this. Simply *add* the two Taylor series (3) for $f(x+h)$ and $f(x-h)$. The result is

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + 2\left[\frac{1}{4!}h^4 f^{(4)}(x) + \cdots\right]$$

When this is rearranged, we get

$$f''(x) = \frac{1}{h^2}[f(x+h) - 2f(x) + f(x-h)] + E$$

where the error series is

$$E = -2\left[\frac{1}{4!}h^2 f^{(4)}(x) + \frac{1}{6!}h^4 f^{(6)}(x) + \cdots\right]$$

By carrying out the same process using Taylor's formula with a remainder, one can show that E is also given by

$$E = -\frac{1}{12}h^2 f^{(4)}(\xi)$$

for some ξ in the interval $(x-h, x+h)$. Hence, we have the approximation

$$f''(x) \approx \frac{1}{h^2}[f(x+h) - 2f(x) + f(x-h)] \tag{20}$$

with error $\mathcal{O}(h^2)$.

EXAMPLE 4 Repeat Example 2, using the central difference formula (20) to approximate the second derivative of the function $f(x) = \sin x$ at the given point $x = 0.5$.

Solution Using the truncation error term, we set

$$\left|-\frac{1}{12}h^2 f^{(4)}(\xi)\right| \leqq \frac{1}{12}4^{-2n} < \frac{1}{2}10^{-6}$$

and we obtain $n > (6 - \log 6)/\log 16 \approx 4.34$. Hence, the modified program *First* finds a good approximation of $f''(0.5) \approx -0.47942$ after about four iterations. (The least error of 3.1×10^{-9} was obtained at iteration 6.) ■