# 3

# Polynomial Approximation, Interpolation, and Orthogonal Polynomials

• • •

In the last chapter we saw that the eigen-equation for a matrix was a polynomial whose roots were the eigenvalues of the matrix. However, polynomials play a much larger role in numerical analysis than providing just eigenvalues. Indeed, the foundation of most numerical analysis methods rests on the understanding of polynomials. As we shall see, numerical methods are usually tailored to produce exact answers for polynomials. Thus, if the solution to a problem is a polynomial, it is often possible to find a method of analysis, which has zero formal truncation error. So the extent to which a problem's solution resembles a polynomial will generally determine the accuracy of the solution. Therefore we shall spend some time understanding polynomials themselves so that we may better understand the methods that rely on them.

## 3.1 Polynomials and Their Roots

When the term polynomial is mentioned, one generally thinks of a function made up of a sum of terms of the form $a_i x^i$. However, it is possible to have a much broader definition where instead of the simple function $x^i$ we may use any general function $\varphi_i(x)$ so that a general definition of a polynomial would have the form

$$P(x) = \sum_{i=0}^{n} a_i \phi_i(x) \quad . \tag{3.1.1}$$

Here the quantity n is known as the degree of the polynomial and is usually one less than the number of terms in the polynomial. While most of what we develop in this chapter will be correct for general polynomials such as those in equation (3.1.1), we will use the more common representation of the polynomial so that

$$\phi_i(x) = x^i \quad . \tag{3.1.2}$$

Thus the common form for a polynomial would be

$$P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n \quad . \tag{3.1.3}$$

Familiar as this form may be, it is not the most convenient form for evaluating the polynomial. Consider the last term in equation (3.1.3). It will take n+1 multiplications to evaluate that term alone and n multiplications for the next lowest order term. If one sums the series, it is clear that it will take (n+1)n/2 multiplications and n additions to evaluate P(x). However, if we write equation (3.1.3) as

$$P(x) = a_0 + (a_1 + \cdots (a_{n-1} + a_n x)x)x \cdots)x \ , \tag{3.1.4}$$

then, while there are still n additions required for the evaluation of P(x), the number of multiplications has been reduced to n. Since the time required for a computer to carry out a multiplication is usually an order of magnitude greater than that required for addition, equation (3.1.4) is a considerably more efficient way to evaluate P(x) than the standard form given by equation (3.1.3). Equation (3.1.4) is sometimes called the "*factored form*" of the polynomial and can be immediately written down for any polynomial. However, there is another way of representing the polynomial in terms of factors, namely

$$P(x) = a_n(x - x_1)(x - x_2)(x - x_3)\cdots(x - x_n) \ . \tag{3.1.5}$$

Here the last n coefficients of the polynomial have been replaced by n quantities known as the roots of the polynomial. It is important to note that, in general, there are (n+1) parameters specifying a polynomial of degree n. These parameters can be either the (n+1) coefficients or the n roots and a multiplicative scale factor $a_n$. In order to fully specify a polynomial this many parameters must be specified. We shall see that this requirement sets constraints for interpolation.

The n quantities known as the roots are not related to the coefficients in a simple way. Indeed, it is not obvious that the polynomial should be able to be written in the form of equation (3.1.5). The fact that a polynomial of degree n has exactly n such roots is known as the *fundamental theorem of algebra* and its proof is not simple. As we shall see, simply finding the roots is not simple and constitutes one of the more difficult problems in numerical analysis. Since the roots may be either real or complex, the most general approach will have to utilize complex arithmetic. Some polynomials may have multiple roots (i.e. more than one root with the same numerical value). This causes trouble for some root finding methods. In general, it is useful to remove a root (or a pair if they are complex) once it is found thereby reducing the polynomial to a lower degree. Once it has been reduced to a quadratic or even a cubic, the analytic formulae for these roots maybe used. There is an analytic form for the general solution of a quartic (i.e. polynomial of 4th degree), but it is so cumbersome that it is rarely used. Since it has been shown that there is no general form for the roots of polynomials of degree 5 or higher, one will usually have to resort to numerical methods in order to find the roots of such polynomials. The absence of a general scheme for finding the roots in terms of the coefficients means that we shall have to learn as much about the polynomial as possible before looking for the roots.

## *a. Some Constraints on the Roots of Polynomials*

This subject has been studied by some of the greatest mathematical minds of the last several centuries and there are numerous theorems that can be helpful in describing the roots. For example, if we re-multiply equation (3.1.5) the coefficient of $x^{n-1}$ is just $a_n$ times the negative summation of the roots so that

$$a_{n-1} = a_n \sum_{i=1}^{n} x_i \quad . \tag{3.1.6}$$

In a similar manner we find that

$$a_{n-2} = a_n \sum_{i\neq j} \sum_{j} x_i x_j \ . \tag{3.1.7}$$

We will see that it is possible to use these relations to obtain estimates of the magnitude of the roots. In addition, the magnitude of the roots is bounded by

$$\left( |a_{\max}| + 1 \right)^{-1} \le |x_j| \le \left( |a_{\max}| + 1 \right). \tag{3.1.8}$$

Finally there is Descarte's *rule of signs* which we all learned at one time but usually forgot. If we reverse the order of equation (3.1.3) so that the terms appear in descending powers of x as

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_0 \quad , \tag{3.1.9}$$

then any change of sign between two successive terms is called a *variation* in sign. Coefficients that are zero are ignored. With that definition of a sign variation we can state Descarte's rule of signs as

*The number of positive roots of P(x)=0 cannot exceed the number of variations of sign in P(x) and, in any case, differs from the number of variations by an* <u>*even integer*</u>*.*

A useful and easily proved corollary to this is

*The number of negative roots of P(x)=0 cannot exceed the number of variations in sign in P(-x) and, in any case, differs from the number of variations by an* <u>*even integer*</u>*.*

The phrasing concerning the "even integer" results from the possibility of the existence of complex roots, which occur, in pairs (providing the coefficients are real) where one is the complex conjugate of the other. With these tools, it is often possible to say a good deal about the properties of the roots of the polynomial in question. Since most of the methods for finding roots are sequential and require the removal of the roots leading to a new polynomial of lower degree, we should say something about how this is accomplished.

### b. *Synthetic Division*

If we wish to remove a factor from a polynomial we may proceed as if we were doing long division with the added proviso that we keep track of the appropriate powers of x. Thus if (x-r) is to be factored out of P(x) we could proceed in exactly the same fashion as long division. Consider the specific case where r = 2 and

$$P(x) = x^4 + 3x^3 - 17x^2 + 6x - 18 \quad . \tag{3.1.10}$$

The long division would then look like

$$\left.\begin{array}{r}
x^3 + 5x^2 - 7x - 11 \\
(x-2)\overline{)x^4 + 3x^3 - 17x^2 + 3x - 18} \\
\underline{x^4 - 2x^3} \qquad\qquad\qquad\qquad \\
5x^3 - 17x^2 \qquad\qquad\quad \\
\underline{5x^3 - 10x^2} \qquad\qquad\quad \\
-7x^2 + 3x \qquad\quad \\
\underline{-7x^2 + 14x} \qquad\quad \\
-11x - 18 \\
\underline{-11x + 22} \\
-40
\end{array}\right\} \quad . \tag{3.1.11}$$

Thus we can write P(x) as

$$P(x) = (x-2)(x^3+5x^2-7x-11) - 40/(x-2) \quad , \tag{3.1.12}$$

or in general as

$$P(x) = (x-r)Q(x) + R \; . \tag{3.1.13}$$

So if we evaluate the polynomial for x = r we get

$$P(r) = R \quad . \tag{3.1.14}$$

Now if R(r) is zero, then r is a root by definition. Indeed, one method for improving roots is to carry out repeated division, varying r until the remainder R is acceptably close to zero. A cursory inspection of the long division expression (3.1.11) shows that much more is being written down than is necessary. In order for the division to proceed in an orderly fashion, there is no freedom in what is to be done with the lead coefficients of the largest powers of x. Indeed, the coefficients of the resultant polynomial Q(x) are repeated below. Also, when searching for a root, the lead coefficient of x in the divisor is always one and therefore need not be written down. Thus if we write down only the coefficients and r-value for the division process, we can compress the notation so that

$$\left.\begin{array}{c} r = 2\overline{)+1+3-17\ +\ 3-18} = P(x) \\ \underline{+2+10-14-22} \\ Q(x) = +1+5-\ 7\ -11| -40 = R \end{array}\right\} . \tag{3.1.15}$$

This shorthand form of keeping track of the division is known as *synthetic division*. Even this notation can be formulated in terms of a recursive procedure. If we let the coefficients of the quotient polynomial Q(x) be $b_i$ so that

$$Q(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n-1}x^{n-1}\ , \tag{3.1.16}$$

then the process of finding the $b_i$'s in terms of the coefficients $a_i$ of the original polynomial P(x) can be written as

$$\left.\begin{array}{l} b_{n-1} = a_n \\ b_{i-1} = rb_i + a_i \quad i = 0 \cdots n-1 \\ R = b_{-1} \end{array}\right\} . \tag{3.1.17}$$

Here the remainder R is given by $b_{-1}$ and should it be zero, then r is a root. Therefore, once a root has been found, it can be removed by synthetic division leading to an new polynomial Q(x). One can then begin again to find the roots of Q(x) until the original polynomial has been reduced to a cubic or less. Because of the complexity of the general cubic, one usually uses the quadratic formula. However, even here Press et al[1] suggest caution and recommend the use of both forms of the formula, namely

$$\left.\begin{array}{l} x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a} \\ x = \dfrac{2c}{-b \pm \sqrt{b^2 - 4ac}} \end{array}\right\} . \tag{3.1.18}$$

Should a or c be small the discriminate will be nearly the same as b and the resultant solution will suffer from round-off error. They suggest the following simple solution to this problem. Define

$$q = -[b + \mathrm{sgn}(b)\sqrt{b^2 - 4ac}\,]/2\,. \tag{3.1.19}$$

Then the two roots will be given by

$$\left.\begin{array}{l} x = q/a \\ x = c/q \end{array}\right\} \quad . \tag{3.1.20}$$

Let us see how one might analyze our specific polynomial in equation (3.1.10). Descartes' rule of signs for P(x) tells us that we will have no more than three real positive roots while for P(-x) it states that we will have no more than one real negative root. The degree of the polynomial itself indicates that there will be four roots in all. When the coefficients of a polynomial are integer, it is tempting to look for integer roots. A little exploring with synthetic division shows that we can find two roots so that

$$P(x) = (x-3)(x+6)(x^2+1) \quad , \tag{3.1.21}$$

and clearly the last two roots are complex. For polynomials with real coefficients, one can even use synthetic division to remove complex roots. Since the roots will appear in conjugate pairs, simply form the quadratic polynomial

$$(x-r)(x-r^*) = x^2 - (r+r^*)x + r\,r^* \quad , \tag{3.1.22}$$

which will have real coefficients as the imaginary part of r cancels out of $(r+r^*)$ and $rr^*$ is real by definition. One then uses synthetic division to divide out the quadratic form of equation (3.1.22). A general recurrence relation similar to equation (3.1.17) can be developed for the purposes of machine computation.

Normally the coefficients of interesting polynomials are not integers and the roots are not simple numbers. Therefore the synthetic division will have a certain round off error so that R(r) will not be zero. This points out one of the greatest difficulties to be encountered in finding the roots of a polynomial. The round off error in R(r) accumulates from root to root and will generally depend on the order in which the roots are found. Thus the final quadratic polynomial that yields the last two roots may be significantly different than the correct polynomial that would result in the absence of round off error. One may get a feeling for the extent of this problem by redoing the calculation but finding the roots in a different order. If the values are independent of the order in which they are found, then they are probably accurate. If not, then they are not.

### c. *The Graffe Root-Squaring Process*

We discuss this process not so much for its practical utility as to show the efficacy of the constraints given in equations (3.1.6,7). Consider evaluating a polynomial for values of $x = x_i$ where $x_i$ are the roots so that

$$P(x_i) = \sum_j a_i x_i^j = \sum_k \; a_{2k} x_i^{2k} + a_{2k+1} x_i^{2k+1} \quad . \tag{3.1.23}$$

We may separate the terms of the polynomial into even and odd powers of x and since $P(x_i)=0$, we may arrange the odd powers so that they are all on one side of the equation as

$$\left[\sum_k a_{2k} x_i^{2k}\right]^2 = \left[\sum_k a_{2k+1} x_i^{2k+1}\right]^2 . \tag{3.1.24}$$

Squaring both sides produces exponents with even powers and a polynomial with new coefficients $a_i^{(p)}$ and having the form

$$S(x) = a_n^{(p)} x^{2pn} + a_n^{(p)} x^{2pn-2} + \cdots + a_0^{(p)} \quad . \tag{3.1.25}$$

These new coefficients can be generated by the recurrence relation from

$$\left.\begin{array}{l} a_i^{(p+1)} = 2a_n^{(p)} a_{2\ell}^{(p)} - 2\sum\limits_{k=\ell-1}^{n-1} a_k^{(p)} a_{2\ell-k}^{(p)} + (-1)^\ell (a_1^{(p)})^2 \\ a_i^{(p)} = 0, \quad i > n \end{array}\right\} . \tag{3.1.26}$$

If we continue to repeat this process it is clear that the largest root will dominate the sum in equation (3.1.6) so that

$$x_{\max}^{2p} = \operatorname*{Lim}_{p\to\infty} \sum_{i=1}^{n} x_i^{2p} = \operatorname*{Lim}_{p\to\infty} \left[ a_{n-1}^{(p)} \big/ a_n^{(p)} \right] . \tag{3.1.27}$$

Since the product of the largest two roots will dominate the sums of equation (3.1.7), we may generalize the result of eq (3.1.27) so that each root will be given by

$$x_i^{2p} \cong \operatorname*{Lim}_{p\to\infty} \left[ a_{i-1}^{(p)} \big/ a_n^{(p)} \right] \quad . \tag{3.1.28}$$

While this method will in principle yield all the roots of the polynomial, the coefficients grow so fast that roundoff error quickly begins to dominate the polynomial. However, in some instance it may yield approximate roots that will suffice for initial guesses required by more sophisticated methods. Impressive as this method is theoretically, it is rarely used. While the algorithm is reasonably simple, the large number of digits required by even a few steps makes the programming of the method exceedingly difficult.

### *d. Iterative Methods*

Most of the standard algorithms used to find the roots of polynomials scan the polynomial in an orderly fashion searching for the root. Any such scheme requires an initial guess, a method for predicting a better guess, and a system for deciding when a root has been found. It is possible to cast any such method in the form of a fixed-point iterative function such as was discussed in section 2.3d. Methods having this form are legion so we will discuss only the simplest and most widely used. Putting aside the problem of establishing the initial guess, we will turn to the central problem of predicting an improved value for the root. Consider the simple case of a polynomial with real roots and having a value $P(x_k)$ for some value of the independent variable $x_k$ (see Figure 3.1).

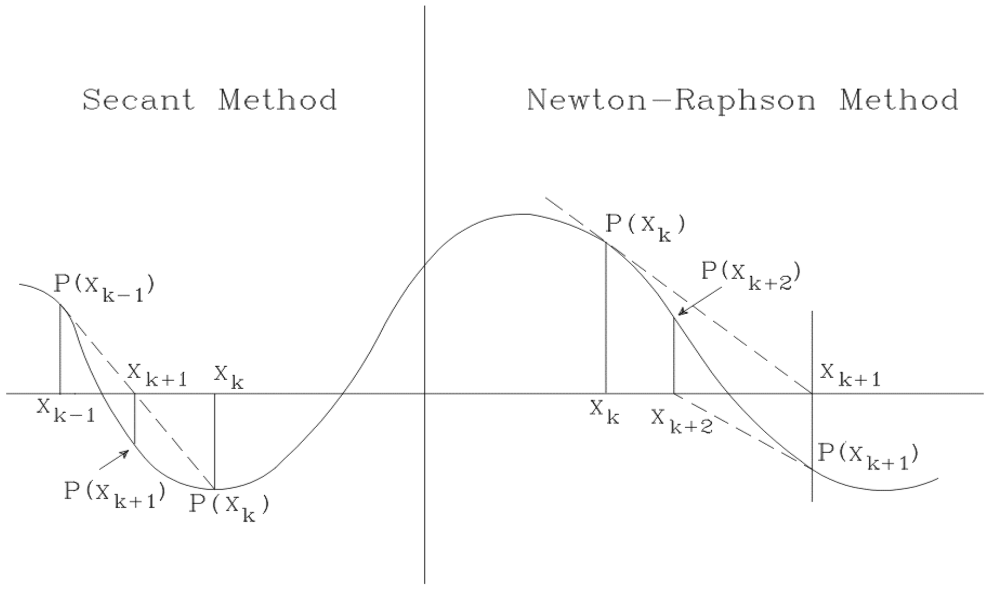

*Figure 3.1 depicts a typical polynomial with real roots. Construct the tangent to the curve at the point* $x_k$ *and extend this tangent to the x-axis. The crossing point* $x_{k+1}$ *represents an improved value for the root in the Newton-Raphson algorithm. The point* $x_{k-1}$ *can be used to construct a secant providing a second method for finding an improved value of* x.

Many iterative techniques use a straight line extension of the function P(x) to the x-axis as a means of determining an improved value for x. In the case where the straight-line approximation to the function is obtained from the local tangent to the curve at the point $x_k$, we call the method the *Newton-Raphson* method. We can cast this in the form of a fixed-point iterative function since we are looking for the place where P(x) = 0. In order to find the iterative function that will accomplish this let us assume that an improved value of the root $x^{(k)}$ will be given by

$$x^{(k+1)} = x^{(k)} + [x^{(k+1)}-x^{(k)}] \equiv x^{(k)} + \Delta x^{(k)} \ . \tag{3.1.29}$$

Now since we are approximating the function locally by a straight line, we may write

$$\left.\begin{array}{l} P[x^{(k)}] = \alpha x^{(k)} + \beta \\ P[x^{(k+1)}] = \alpha x^{(k+1)} + \beta \end{array}\right\} . \tag{3.1.30}$$

Subtracting these two equations we get

$$P[x^{(k)}] = \alpha[x^{(k)} - x^{(k+1)}] = -\alpha\Delta x^{(k)} \ . \tag{3.1.31}$$

However the slope of the tangent line α is given by the derivative so that

$$\alpha = dP[x^{(k)}]/dx \ . \tag{3.1.32}$$

Thus the Newton-Raphson iteration scheme can be written as

$$x^{(k+1)} = x^{(k)} - P[x^{(k)}]/P'[x^{(k)}] \ . \tag{3.1.33}$$

By comparing equation (3.1.33) to equation (2.3.18) it is clear that the fixed-point iterative function for Newton-Raphson iteration is

$$\Phi(x) = x - P(x)/P'(x) \ . \tag{3.1.34}$$

We can also apply the convergence criterion given by equation (2.3.20) and find that the necessary and sufficient condition for the convergence of the Newton-Raphson iteration scheme is

$$\left|\frac{P(x)P''(x)}{[P'(x)]^2}\right| < 1 \,, \quad \forall x \ \varepsilon \ x^{(k)} \le x \le x_0 \quad . \tag{3.1.35}$$

Since this involves only one more derivative than is required for the implementation of the scheme, it provides a quite reasonable convergence criterion and it should be used in conjunction with the iteration scheme.

The Newton-Raphson iteration scheme is far more general than is implied by its use in polynomial root finding. Indeed, many non-linear equations can be dealt with by means of equations (3.1.34, 35). From equation (3.1.33), it is clear that the scheme will yield 'exact' answers for first degree polynomials or straight lines. Thus we can expect that the error at any step will depend on $[\Delta x^{(k)}]^2$. Such schemes are said to be second order schemes and converge quite rapidly. In general, if the error at any step can be written as

$$E(x) = \mathbf{K}\times(\Delta x)^n \ , \tag{3.1.36}$$

where K is approximately constant throughout the range of approximation, the approximation scheme is said to be of (order) $O(\Delta x)^n$. It is also clear that problems can occur for this method in the event that the root of interest is a multiple root. Any multiple root of P(x) will also be a root of P'(x). Geometrically this implies that the root will occur at a point where the polynomial becomes tangent to the x-axis. Since the denominator of equation (3.1.35) will approach zero at least quadratically while the numerator may approach zero linearly in the vicinity of the root(s), it is unlikely that the convergence criterion will be met. In practice, the shallow slope of the tangent will cause a large correction to $x^{(k)}$ moving the iteration scheme far from the root.

A modest variation of this approach yields a rather more stable iteration scheme. If instead of using the local value of the derivative to obtain the slope of our approximating line, we use a prior point from the iteration sequence, we can construct a secant through the prior point and the present point instead of the local tangent. The straight line approximation through these two points will have the form

$$\left.\begin{array}{l} P[x^{(k)}] = \alpha x^{(k)} + \beta \\ P[x^{(k-1)}] = \alpha x^{(k-1)} + \beta \end{array}\right\} , \tag{3.1.37}$$

which, in the same manner as was done with equation (3.1.30) yields a value for the slope of the line of

$$\alpha = \frac{P[x^{(k)}] - P[x^{(k-1)}]}{x^{(k)} - x^{(k-1)}} \quad . \tag{3.1.38}$$

So the iterative form of the *secant iteration scheme* is

$$x^{(k+1)} = x^{(k)} - \frac{P[x^{(k)}]\left[x^{(k)} - x^{(k-1)}\right]}{P[x^{(k)}] - P[x^{(k-1)}]} \quad . \tag{3.1.39}$$

Useful as these methods are for finding real roots, as presented, they will be ineffective in locating complex roots. There are numerous methods that are more sophisticated and amount to searching the complex plane for roots. For example *Bairstow's method* synthetically divides the polynomial of interest by an initial quadratic factor which yields a remainder of the form

$$R = \alpha x + \beta \ , \tag{3.1.40}$$

where α and β depend on the coefficients of the trial quadratic form. For that form to contain two roots of the polynomial both α and β must be zero. These two constraints allow for a two-dimensional search in the complex plane to be made usually using a scheme such as Newton-Raphson or versions of the secant method. Press et al strongly suggest the use of the *Jenkins-Taub method* or the *Lehmer-Schur method*. These rather sophisticated schemes are well beyond the scope of this book, but may be studied in Acton[2].

Before leaving this subject, we should say something about the determination of the initial guess. The limits set by equation (3.1.8) are useful in choosing an initial value of the root. They also allow for us to devise an orderly progression of finding the roots - say from large to small. While most general root finding programs will do this automatically, it is worth spending a little time to see if the procedure actually follows an orderly scheme. Following this line, it is worth repeating the cautions raised earlier concerning the difficulties of finding the roots of polynomials. The blind application of general programs is almost certain to lead to disaster. At the very least, one should check to see how well any given root satisfies the original polynomial. That is, to what extent is $P(x_i) = 0$. While even this doesn't guarantee the accuracy of the root, it is often sufficient to justify its use in some other problem.

## 3.2 Curve Fitting and Interpolation

The very processes of interpolation and curve fitting are basically attempts to get "something for nothing". In general, one has a function defined at a discrete set of points and desires information about the function at some other point. Well that information simply doesn't exist. One must make some assumptions about the behavior of the function. This is where some of the "art of computing" enters the picture. One needs some knowledge of what the discrete entries of the table represent. In picking an interpolation scheme to generate the missing information, one makes some assumptions concerning the functional nature of the tabular entries. That assumption is that they behave as polynomials. All interpolation theory is based on polynomial approximation. To be sure the polynomials need not be of the simple form of equation (3.1.3), but nevertheless they will be polynomials of some form such as equation (3.1.1).

Having identified that missing information will be generated on the basis that the tabular function is represented by a polynomial, the problem is reduced to determining the coefficients of that polynomial. Actually some thought should be given to the form of the functions $\varphi_i(x)$ which determines the basic form of the polynomial. Unfortunately, more often than not, the functions are taken to be $x^i$ and any difficulties in representing the function are offset by increasing the order of the polynomial. As we shall see, this is a dangerous procedure at best and can lead to absurd results. It is far better to see if the basic data is - say exponential or periodic in form and use basis functions of the form $e^{ix}$, $\sin(i\,\pi\,x)$, or some other appropriate functional form. One will be able to use interpolative functions of lower order which are subject to fewer large and unexpected fluctuations between the tabular points thereby producing a more reasonable result.

Having picked the basis functions of the polynomial, one then proceeds to determine the coefficients. We have already observed that an nth degree polynomial has (n+1) coefficients which may be regarded as (n+1) degrees of freedom, or n+1 free parameters to adjust so as to provide the best fit to the tabular entry points. However, one still has the choice of how much of the table to fit at any given time. For interpolation or curve-fitting, one assumes that the tabular data are known with absolute precision. Thus we expect the approximating polynomial to reproduce the data points exactly, but the number of data points for which we will make this demand at any particular part of the table remains at the discretion of the investigator. We shall develop our interpolation formulae initially without regard to the degree of the polynomial that will be used. In addition, although there is a great deal of literature developed around interpolating equally spaced data, we will allow the spacing to be arbitrary. While we will forgo the elegance of the finite difference operator in our derivations, we will be more than compensated by the generality of the results. These more general formulae can always be used for equally spaced data. However, we shall limit our generality to the extent that, for examples, we shall confine ourselves to basis functions of the form $x^i$. The generalization to more exotic basis functions is usually straightforward. Finally, some authors make a distinction between interpolation and curve fitting with the latter being extended to a single functional relation, which fits an entire tabular range. However, the approaches are basically the same so we shall treat the two subjects as one. Let us then begin by developing *Lagrange Interpolation formulae*.

## *a. Lagrange Interpolation*

Let us assume that we have a set of data points $Y(x_i)$ and that we wish to approximate the behavior of the function between the data points by a polynomial of the form

$$\Phi(x) = \sum_{j=0}^{n} a_j x^j \ . \tag{3.2.1}$$

Now we require exact conformity between the interpolative function $\Phi(x_i)$ and the data points $Y(x_i)$ so that

$$Y(x_i) = \Phi(x_i) = \sum_{j=0}^{n} a_j x_i^j \ , \quad i = 0 \cdots n \ . \tag{3.2.2}$$

Equation (3.2.2) represents n+1 inhomogeneous equations in the n+1 coefficients $a_j$ which we could solve using the techniques in chapter 2. However, we would then have a single interpolation formula that would have to be changed every time we changed the values of the dependent variable $Y(x_i)$. Instead, let us combine equations (3.2.1) and (3.2.2) to form n+2 homogeneous equations of the form

$$\left.\begin{array}{l}\sum\limits_{j=0}^{n} a_j x_i^j - Y(x_i) \\ \sum\limits_{j=0}^{n} a_j x^j - \Phi(x)\end{array}\right\} = 0 \quad . \tag{3.2.3}$$

These equations will have a solution if and only if

$$\text{Det}\begin{vmatrix} 1 & x_0 & x_0^2 & x_0^3 & \cdots & x_0^n & -Y_0 \\ 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^n & -Y_1 \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots & x_2^n & -Y_2 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x & x^2 & x^3 & \cdots & x^n & -\Phi(x) \end{vmatrix} = 0 \quad . \tag{3.2.4}$$

Now let x = $x_i$ and subtract the last row of the determinant from the ith row so that expansion by minors along that row will yield

$$[\Phi(x_i) - Y_i]\,\left|x_k^j\right|_i = 0 \quad . \tag{3.2.5}$$

Since $\left|x_k^j\right|_i \neq 0$, the value of $\Phi(x_i)$ must be $Y(x_i)$ satisfying the requirements given by equation (3.2.2). Now expand equation (3.2.4) by minors about the last column so that

$$\Phi(x)\left|x_k^j\right| = \begin{vmatrix} 1 & x_0 & x_0^2 & x_0^3 & \cdots & x_0^n & -Y_0 \\ 1 & x_1 & x_1^2 & x_1^3 & \cdots & x_1^n & -Y_1 \\ 1 & x_2 & x_2^2 & x_2^3 & \cdots & x_2^n & -Y_2 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x & x^2 & x^3 & \cdots & x^n & 0 \end{vmatrix} = \sum_{i=0}^{n} Y(x_i)A_i(x) \quad . \tag{3.2.3}$$

Here the $A_i(x)$ are the minors that arise from the expansion down the last column and they are independent of the $Y_i$'s. They are simply linear combinations of the $x^j$' s and the coefficients of the linear combination depend only on the $x_i$'s. Thus it is possible to calculate them once for any set of independent variables $x_i$ and use the results for any set of $Y_i$'s. The determinant $\left|x_k^j\right|$ depends only on the spacing of the tabular values of the independent variable and is called the *Vandermode determinant* and is given by

$$V_d = \left|x_k^j = \prod_{i>j=0}^{n}(x_i - x_j)\right| \quad . \tag{3.2.7}$$

Therefore dividing $A_i(x)$ in equation (3.2.6) by the Vandermode determinant we can write the interpolation formula given by equation (3.2.6) as

$$\Phi(x) = \sum_{i=0}^{n} Y(x_i)L_i(x) \quad , \tag{3.2.8}$$

where $L_i(x)$ is known as the *Lagrange Interpolative Polynomial* and is given by

$$L_i(x) = \prod_{\substack{j \neq i \\ j=0}}^{n} (x - x_j) \Big/ (x_i - x_j) \quad . \tag{3.2.9}$$

This is a polynomial of degree n with roots $x_j$ for $j \neq i$ since one term is skipped (i.e. when i = j) in a product of n+1 terms. It has some interesting properties. For example

$$L_i(x_k) = \prod_{\substack{j \neq i \\ j=0}}^{n} (x_k - x_j) \Big/ (x_i - x_j) \quad = \quad \delta_{ki} \quad , \tag{3.2.10}$$

where $\delta_{ik}$ is Kronecker's delta. It is clear that for values of the independent variable equally separated by an amount h the Lagrange polynomials become

$$L_i(x) = \frac{(-1)^n}{(n-i)!i!h^n} \prod_{\substack{j \neq i \\ j=0}}^{n} (x - x_j) \quad . \tag{3.2.11}$$

The use of the Lagrangian interpolation polynomials as described by equations (3.2.8) and (3.2.9) suggest that entire range of tabular entries be used for the interpolation. This is not generally the case. One picks a subset of tabular points and uses them for the interpolation. The use of all available tabular data will generally result in a polynomial of a very high degree possessing rapid variations between the data points that are unlikely to represent the behavior of the tabular data.

Here we confront specifically one of the "artistic" aspects of numerical analysis. We know only the values of the tabular data. The scheme we choose for representing the tabular data at other values of the independent variable must only satisfy some aesthetic sense that we have concerning that behavior. That sense cannot be quantified for the objective information on which to evaluate it simply does not exist. To illustrate this and quantify the use of the Lagrangian polynomials, consider the functional values for $x_i$ and $Y_i$ given in Table 3.1. We wish to obtain a value for the dependent variable Y when the independent variable x = 4. As shown in figure 3.2, the variation of the tabular values $Y_i$ is rapid, particularly in the vicinity of x = 4. We must pick some set of points to determine the interpolative polynomials.

## Table 3.1

### Sample Data and Results for Lagrangian Interpolation Formulae

| I | X | ${}^1_2L_i(4)$ | ${}^2_1L_i(4)$ | ${}^2_2L_i(4)$ | ${}^3_1L_i(4)$ | $Y_I$ | ${}^1_1\Phi_i(4)$ | ${}^2_1\Phi_i(4)$ | ${}^2_2\Phi_i(4)$ | ${}^3_1\Phi_i(4)$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | | | | | 1 | | | | |
| 1 | 2 | | -1/3 | | -2/9 | 3 | | | | |
| 3 | 3 | +1/2 | +1 | +2/5 | +4/5 | 8 | | | | |
| | 4 | | | | | | 6 | 25/3 | 86/15 | 112/15 |
| 4 | 5 | +1/2 | +1/3 | +2/3 | +4/9 | 4 | | | | |
| 5 | 8 | | | -1/15 | -1/45 | 2 | | | | |
| 6 | 10 | | | | | 1 | | | | |

The number of points will determine the order and we must decide which points will be used. The points are usually picked for their proximity to the desired value of the independent variable. Let us pick them consecutively beginning with tabular entry $x_k$. Then the nth degree Lagrangian polynomials will be

$$ {}_k^n L_i(x) = \prod_{\substack{j\neq i\\ j=k}}^{n+k} (x-x_j) \Big/ (x_i - x_j) \quad . \tag{3.2.12}$$

Should we choose to approximate the tabular entries by a straight line passing through points bracketing the desired value of $x = 4$, we would get

$$\left.\begin{aligned} {}_2^1 L_1(x) &= \frac{(x-x_3)}{(x_2-x_3)} = \tfrac{1}{2} \quad \text{for } x = 4 \\ {}_2^1 L_2(x) &= \frac{(x-x_2)}{(x_3-x_2)} = \tfrac{1}{2} \quad \text{for } x = 4 \end{aligned}\right\} \quad . \tag{3.2.13}$$

Thus the interpolative value ${}_2^1\Phi(4)$ given in table 3.1 is simply the average of the adjacent values of $Y_i$. As can be seen in figure 3.2, this instance of linear interpolation yields a reasonably pleasing result. However, should we wish to be somewhat more sophisticated and approximate the behavior of the tabular function with a parabola, we are faced with the problem of which three points to pick. If we bracket the desired point with two points on the left and one on the right we get Lagrangian polynomials of the form

$$\left.\begin{aligned} {}_1^2 L_1(x) &= \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} = -\frac{1}{3}, \quad x = 4 \\ {}_1^2 L_2(x) &= \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} = -1, \quad x = 4 \\ {}_1^2 L_3(x) &= \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} = +\frac{1}{3}, \quad x = 4 \end{aligned}\right\} \quad . \tag{3.2.14}$$

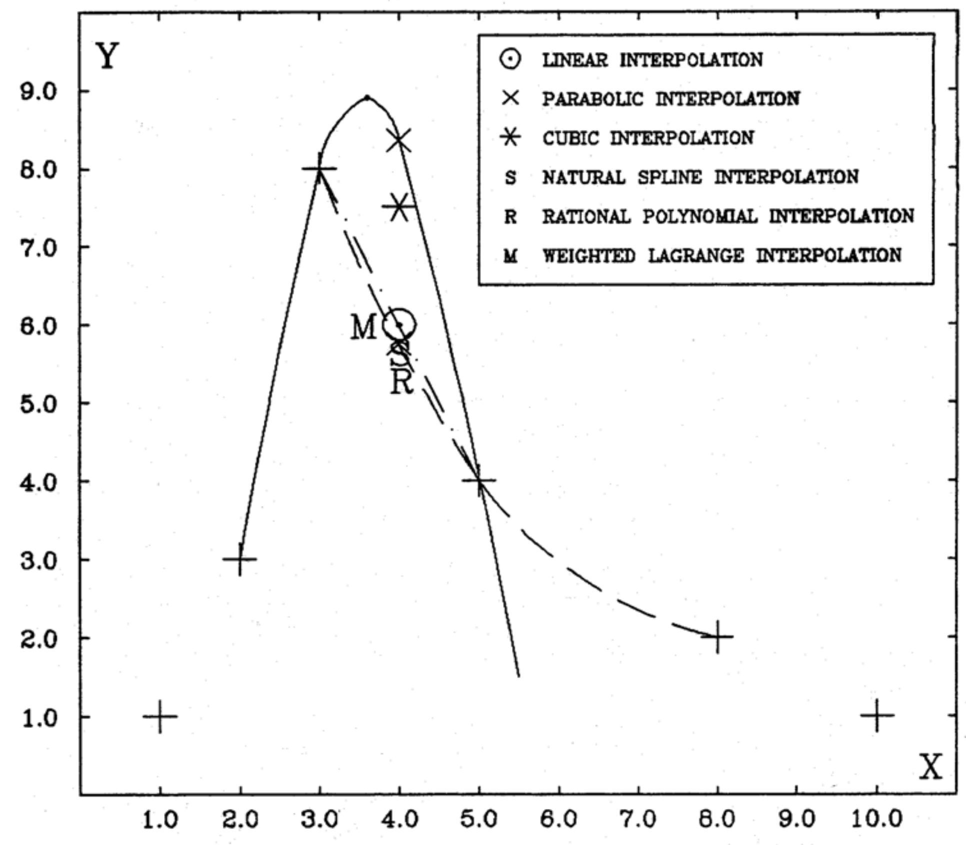

*Figure 3.2 shows the behavior of the data from Table 3.1. The results of various forms of interpolation are shown. The approximating polynomials for the linear and parabolic Lagrangian interpolation are specifically displayed. The specific results for cubic Lagrangian interpolation, weighted Lagrangian interpolation and interpolation by rational first degree polynomials are also indicated.*

Substituting these polynomials into equation (3.2.8) and using the values for $Y_i$ from Table 3.1, we get an interpolative polynomial of the form

$$P_1(x) = 3\,{}^2_1L_1(x) + 8\,{}^2_1L_2(x) + 4\,{}^2_1L_3(x) = -(7x^2-50x+63)/3 \quad . \tag{3.2.15}$$

Had we chosen the bracketing points to include two on the left and only one on the right the polynomial would have the form

$$P_2(x) = 8\,{}^2_2L_1(x) + 4\,{}^2_2L_2(x) + 2\,{}^2_2L_3(x) = 2(2x^2-31x+135)/15 \quad . \tag{3.2.16}$$

However, it is not necessary to functionally evaluate these polynomials to obtain the interpolated value. Only the numerical value of the Lagrangian polynomials for the specific value of the independent variable given on the right hand side of equations (3.2.14) need be substituted directly into equation (3.2.8) along with the appropriate values of $Y_i$. This leads to the values for ${}^2_1\Phi(4)$ and ${}^2_2\Phi(4)$ given in Table 3.1. The values are quite different, but bracket the result of the linear interpolation.

While equations (3.13) - (3.16) provide an acceptable method of carrying out the interpolation, there are more efficiently and readily programmed methods. One of the most direct of these is a recursive procedure where values of an interpolative polynomial of degree k are fit to successive sets of the data points. In this method the polynomial's behavior with x is not found, just its value for a specific choice of x. This value is given by

$$\left.\begin{aligned} P_{i,i+1,\cdots,i+k}(x) &= \frac{(x-x_{i+k})P_{i,i+1,\cdots,i+k-1}+(x_i-x)P_{i+1,i+2,\cdots,i+k}(x)}{(x_i-x_{i+k})} \\ P_{i,i}(x) &= Y_i\,, \quad \text{for } k=0 \end{aligned}\right\} . \qquad (3.2.17)$$

For our test data given in table 3.1 the recursive formula given by equation (3.2.17) yields

$$\left.\begin{aligned} P_{1,2}(4) &= \frac{(4-x_2)Y_1+(x_1-4)Y_2}{(x_1-x_2)} = \frac{(4-3)\times 3+(2-4)\times 8}{(2-3)} = +13 \\ P_{2,3}(4) &= \frac{(4-x_3)Y_2+(x_2-4)Y_3}{(x_2-x_3)} = \frac{(4-5)\times 8+(3-4)\times 4}{(3-5)} = +6 \\ P_{3,4}(4) &= \frac{(4-x_4)Y_3+(x_3-4)Y_4}{(x_3-x_4)} = \frac{(4-5)\times 4+(5-4)\times 2}{(5-8)} = +\frac{14}{3} \end{aligned}\right\} . \qquad (3.2.18)$$

for k = 1. Here we see that $P_{2,3}(4)$ corresponds to the linear interpolative value obtained using points $x_2$ and $x_3$ given in table 3.1 as ${}^1_2\Phi(4)$. In general, the values of $P_{i,i+1}(x)$ correspond to the value of the straight line passing through points $x_i$ and $x_{i+1}$ evaluated at x. The next generation of recursive polynomial-values will correspond to parabolas passing through the points $x_i$, $x_{i+1}$, and $x_{i+2}$ evaluated at x.

For this example they are

$$\left.\begin{aligned} P_{1,2,3}(4) &= \frac{(4-x_2)P_{1,2}(4)+(x_1-4)P_{2,3}(4)}{(x_1-x_3)} = \frac{(4-3)\times 3+(2-4)\times 8}{(2-5)} = +\frac{25}{3} \\ P_{2,3,4}(4) &= \frac{(4-x4)P_{2,3}(4)+(x_2-4)P_{3,4}(4)}{(x_2-x_4)} = \frac{(4-8)\times 6+(3-4)\times(^{14}\!/\!_3)}{(3-8)} = +\frac{86}{15} \end{aligned}\right\} , \qquad (3.2.20)$$

which correspond to the values for ${}^2_1\Phi(4)$ and ${}^2_2\Phi(4)$ in table 3.1 respectively. The cubic which passes through points $x_1$, $x_2$, $x_3$, and $x_4$ is the last generation of the polynomials calculated here by this recursive procedure and is

$$P_{1,2,3,4}(4) = \frac{(4-x_3)P_{1,2,3}(4)+(x_1-4)P_{2,3,4}(4)}{(x_1-x_4)} = \frac{(4-8)\times(^{25}\!/\!_3)+(2-4)\times(^{86}\!/\!_{15})}{(2-8)} = +\frac{112}{15} . \qquad (3.2.21)$$

The procedure described by equation (3.2.17) is known as Neville's algorithm and can nicely be summarized by a Table 3.2.

The fact that these results exactly replicate those of table 3.1 is no surprise as the polynomial of a particular degree k that passes through a set of k+1 points is unique. Thus this algorithm describes a particularly efficient method for carrying out Lagrangian interpolation and, like most recursive proceedures, is easily implemented on a computer.

How are we to decide which of the parabolas is "better". In some real sense, both are equally likely. The large value of ${}^{2}_{1}\Phi(4)$ results because of the rapid variation of the tabular function through the three chosen points (see figure 3.1) and most would reject the result as being too high. However, we must remember that this is a purely subjective judgment. Perhaps one would be well advised to always have the same number of points on either side so as to insure the tabular variation on either side is equally weighted. This would lead to interpolation by polynomials of an odd degree. If we chose two points either side of the desired value of the independent variable, we fit a cubic through the local points and obtain ${}^{3}_{1}\Phi(4)$ which is rather close to ${}^{2}_{1}\Phi(4)$. It is clear that the rapid tabular variation of the points preceding x = 4 dominate the interpolative polynomials. So which one is correct? We must emphasis that there is no objectively "correct" answer to this question. Generally one prefers an interpolative function that varies no more rapidly that the tabular values themselves, but when those values are sparse this criterion is difficult to impose. We shall consider additional interpolative forms that tend to meet this subjective notion over a wide range of conditions. Let us now turn to methods of their construction.

**Table 3.2**

**Parameters for the Polynomials Generated by Neville's Algorithm**

| I | X | $Y_I$ | $P_{I,I}$ | $P_{I,I+1}$ | $P_{I,I+1,I+2}$ | $P_{I,I+1,I+2,I+3}$ |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | | | |
| | | | | 0 | | |
| 1 | 2 | 3 | 3 | | 0 | |
| | | | | +13 | | |
| 2 | 3 | 8 | 8 | | +25/3 | 0 |
| | | | | | | |
| | 4 | | | +6 | | 112/15 |
| | | | | | | |
| 3 | 5 | 4 | 4 | | +86/15 | 0 |
| | | | | +14/3 | | |
| 4 | 8 | 2 | 2 | | 0 | |
| | | | | 0 | | |
| 5 | 10 | 1 | 0 | | | |

It is possible to place additional constraints on the interpolative function which will make the appropriate interpolative polynomials somewhat more difficult to obtain, but it will always be possible to obtain them through consideration of the determinantal equation similar to equation (3.2.6). For example, let us consider the case where constraints are placed on the derivative of the function at a given number of values for the independent variable.

## b. *Hermite Interpolation*

While we will use the Hermite interpolation formula to obtain some highly efficient quadrature formulae later, the primary reason for discussing this form of interpolation is to show a powerful approach to generating interpolation formulae from the properties of the Lagrange polynomials. In addition to the functional constraints of Lagrange interpolation given by equation (3.2.2), let us assume that the functional values of the derivative $Y'(x_i)$ are also specified at the points $x_i$. This represents an additional (n+1) constraints. However, since we have assumed that the interpolative function will be a polynomial, the relationship between a polynomial and its derivative means we shall have to be careful in order that these 2n+2 constraints remain linearly independent. While a polynomial of degree n has (n+1) coefficients, its derivative will have only n coefficients. Thus the specification of the derivative at the various values of the independent variable allow for a polynomial with 2n+2 coefficients to be used which is a polynomial of degree 2n+1.

Rather than obtain the determinantal equation for the 2n+2 constraints and the functional form of the interpolative function, let us derive the interpolative function from what we know of the properties of $L_i(x)$. For the interpolative function to be independent of the values of the dependent variable and its derivative, it must have a form similar to equation (3.2.8) so that

$$\Phi(x) = \sum_{j=0}^{n} Y(x_j)h_j(x) + Y'(x_j)H_j(x) \quad . \tag{3.2.21}$$

As before we shall require that the interpolative function yield the exact values of the function at the tabular values of the independent variable. Thus,

$$\Phi(x_i) = Y(x_i) = \sum_{j=0}^{n} Y(x_j)h_j(x_i) + Y'(x_j)H_j(x_i) \quad . \tag{3.2.22}$$

Now the beauty of an interpolation formula is that it is independent of the values of the dependent variable and, in this case, its derivative. Thus equation (3.2.22) must hold for any set of data points $Y_i$ and their derivatives $Y'_i$. So lets consider a very specific set of data points given by

$$\left.\begin{array}{l} Y(x_i) = 1 \\ Y(x_j) = 0, \quad j \neq i \\ Y'(x_j) = 0, \quad \forall j \end{array}\right\} \quad . \tag{3.2.23}$$

This certainly implies that $h_i(x_i)$ must be one. A different set of data points that have the properties that

$$\left.\begin{array}{l} Y(x_i) = 0, \quad j \neq k \\ Y(x_k) = 0, \\ Y'(x_j) = 0, \quad \forall j \end{array}\right\} \quad , \tag{3.2.24}$$

will require that $h_k(x_j)$ be zero. However, the conditions on $h_i(x_j)$ must be independent of the values of the independent variable so that both conditions must hold. Therefore

$$h_j(x_i) = \delta_{ij} \ . \tag{3.2.25}$$

where $\delta_{ij}$ is Kronecker's delta. Finally one can consider a data set where

$$\left.\begin{array}{l} Y(x_j) = 0 \\ Y'(x_j) = 1, \end{array}\right\} \ \forall j \ . \qquad 3.2.26)$$

Substitution of this set of data into equation (3.2.22) clearly requires that

$$H_j(x_i) = 0 \ . \tag{3.2.27}$$

Now let us differentiate equation (3.2.21) with respect to x and evaluate at the tabular values of the independent variable $x_i$. This yields

$$\Phi'(x_i) = Y(x_i) = \sum_{j=0}^{n} Y(x_j)h'_j(x_i) + Y'(x_j)H'_j(x_i) \ . \tag{3.2.28}$$

By choosing our data sets to have the same properties as in equations (3.2.23,24) and (3.2.26), but with the roles of the function and its derivative reversed, we can show that

$$\left.\begin{array}{l} h'_j(x_i) = 0 \\ H'_j(x_i) = \delta_{ij} \end{array}\right\} \ . \tag{3.2.29}$$

We have now place constraints on the interpolative functions $h_j(x)$, $H_j(x)$ and their derivatives at each of the n+1 values of the independent variable. Since we know that both $h_j(x)$ and $H_j(x)$ are polynomials, we need only express them in terms of polynomials of the correct degree which have the same properties at the points $x_i$ to uniquely determine their form.

We have already shown that the interpolative polynomials will have a degree of (2n+1). Thus we need only find a polynomial that has the form specified by equations (3.2.25) and (3.2.29). From equation (3.2.10) we can construct such a polynomial to have the form

$$h_j(x) = v_j(x)L_j^2(x) \ , \tag{3.2.30}$$

where $v_j(x)$ is a linear polynomial in x which will have only two arbitrary constants. We can use the constraint on the amplitude and derivative of $h_j(x_i)$ to determine those two constants. Making use of the constraints in equations (3.2.25) and (3.2.29) we can write that

$$\left.\begin{array}{l} h_i(x_i) = v_i(x_i)L_i^2(x_i) = 1 \\ h'_j(x_i) = v'_j(x_i)L_j^2(x_i) + 2v_j(x_i)L'_j(x_i)L_j(x_i) = 0 \end{array}\right\} \ . \tag{3.2.31}$$

Since $v_i(x)$ is a linear polynomial, we can write

$$v_i(x) = a_i x + b_i \ . \tag{3.2.32}$$

Specifically putting the linear form for $v_i(x)$ into equation (3.2.31) we get

$$\left.\begin{aligned} v_i(x_i) &= a_i x_i + b_i = 1 \\ v'_i(x_i) &= a_i = -2(a_i x_i + b_i)L'_i(x\ \ ) \end{aligned}\right\} , \tag{3.2.33}$$

which can be solved for $a_i$ and $b_i$ to get

$$\left.\begin{aligned} a_i &= -2L'_i(x_i) \\ b_i &= 1 + 2x_i L'_i(x_i) \end{aligned}\right\} . \tag{3.2.34}$$

Therefore the linear polynomial $v_i(x)$ will have the particular form

$$v_i(x) = 1 - 2(x-x_i)L'_i(x_i) \ . \tag{3.2.35}$$

We must follow the same procedure to specify $H_j(x)$. Like $h_j(x)$, it will also be a polynomial of degree 2n+1 so let us try the same form for it as we did for $h_j(x)$. So

$$H_j(x) = u_j(x)L_j^2(x) \ , \tag{3.2.36}$$

where $u_j(x)$ is also a linear polynomial whose coefficients must be determined from

$$\left.\begin{aligned} H_i(x_i) &= u_i(x_i)L_i^2(x_i) = 0 \\ H'_i(x_i) &= u'_i(x_i)L_i^2(x_i) + 2u_i(x_i)L'_i(x_i)L_i(x_i) = 1 \end{aligned}\right\} . \tag{3.2.37}$$

Since $L_i^2(x_i)$ is unity, these constraints clearly limit the values of $u_i(x)$ and its derivative at the tabular points to be

$$\left.\begin{aligned} u_i(x_i) &= 0 \\ u'_i(x_i) &= 1 \end{aligned}\right\} . \tag{3.2.38}$$

Since $u_i(x)$ is linear and must have the form

$$u_i(x) = \alpha_i x + \beta_i \ , \tag{3.2.39}$$

we can use equation (3.2.38) to fine the constants $\alpha_i$ and $\beta_i$ as

$$\left.\begin{aligned} \alpha_i &= 1 \\ \beta_i &= -x_i \\ u_i(x) &= (x - x_i) \end{aligned}\right\} , \tag{3.2.40}$$

thereby completely specifying $u_i(x)$. Therefore, the two functions $h_j(x)$ and $H_j(x)$ will have the specific form

$$\left.\begin{aligned} h_j(x) &= [1 - 2(x - x_j)L'_j(x_j)]L_j^2(x) \\ H_j(x) &= (x - x_j)L_j^2(x) \end{aligned}\right\} . \tag{3.2.41}$$

All that remains is to find $L'_j(x_j)$. By differentiating equation (3.2.9) with respect to x and setting x to $x_j$, we get

$$L'_j(x_j) = \sum_{k\neq j}(x_j - x_k)^{-1} \ , \tag{3.2.42}$$

which means that $v_j(x)$ will simplify to

$$v_j(x) = 1 - 2\sum_{k\neq j} \frac{(x - x_j)}{(x_j - x_k)} \quad . \tag{3.2.43}$$

Therefore the Hermite interpolative function will take the form

$$\Phi(x) = \sum_i [Y_i v_i(x) + Y'_i\, u_i(x)] \left[\prod_{j\neq i} (x - x_j)/(x_i - x_j)\right]^2 \quad . \tag{3.2.44}$$

This function will match the original function $Y_i$ and its derivative at each of the tabular points. This function is a polynomial of degree 2n-1 with 2n coefficients. These 2n coefficients are specified by the 2n constraints on the function and its derivative. Therefore this polynomial is unique and whether it is obtained in the above manner, or by expansion of the determinantal equation is irrelevant to the result. While such a specification is rarely needed, this procedure does indicate how the form of the Lagrange polynomials can be used to specify interpolative functions that meet more complicated constraints. We will now consider the imposition of a different set of constraints that lead to a class of interpolative functions that have found wide application.

### c. Splines

Splines are interpolative polynomials that involve information concerning the derivative of the function at certain points. Unlike Hermite interpolation that explicitly invokes knowledge of the derivative, splines utilize that information implicitly so that specific knowledge of the derivative in not required. Unlike general interpolation formulae of the Lagrangian type, which maybe used in a small section of a table, splines are constructed to fit an entire run of tabular entries of the independent variable. While one can construct splines of any order, the most common ones are cubic splines as they generate tri-diagonal equations for the coefficients of the polynomials. As we saw in chapter 2, tri-diagonal equations lend themselves to rapid solution involving about N steps. In this case N would be the number of tabular entries of the independent variable. Thus for relatively few arithmetic operations, one can construct a set of cubic polynomials which will represent the function over its entire tabular range. If one were to make a distinction between interpolation and curve fitting, that would be it. That is, one may obtain a local value of a function by interpolation, but if one desires to describe the entire range of a tabular function, one would call that *curve fitting*. Because of the common occurrence of cubic splines, we shall use them as the basis for our discussion. Generalization to higher orders is not difficult, but will generate systems of equations for their coefficients that are larger than tri-diagonal. That removes much of the attractiveness of the splines for interpolation.

To understand how splines can be constructed, consider a function with n tabular points whose independent variable we will denote as $x_i$ and dependent values as $Y_i$. We will approximate the functional values between any two adjacent points $x_i$ and $x_{i+1}$ by a cubic polynomial denoted by $\Psi_i(x)$. Also let the interval between $x_{i+1}$ and $x_i$ be called

$$\Delta x_i \equiv x_{i+1} - x_i \quad . \tag{3.2.45}$$

Since the cubic interpolative polynomials $\Psi_i(x)$ cover each of the n-1 intervals between the n tabular points, there will be 4(n-1) constants to be determined to specify the interpolative functions. As with Lagrange interpolation theory we will require that the interpolative function reproduce the tabular entries so that

$$\left.\begin{array}{l}\Psi_i(x_i) = Y_i \\ \Psi_i(x_{i+1}) = Y_{i+1}\end{array}\right\}\quad i = 1\cdots n-1 \quad . \tag{3.2.46}$$

Requiring that a single polynomial match two successive points means that two adjacent polynomials will have the same value where they meet, or

$$\Psi_i(x_{i+1}) = \Psi_{i+1}(x_{i+1}) \quad i = 1\cdots n-2 \quad . \tag{3.2.47}$$

The requirement to match n tabular points represents n linearly independent constraints on the 4n-4 coefficients of the polynomials. The remaining constraints come from conditions placed on the functional derivatives. Specifically we shall require that

$$\left.\begin{array}{l}\Psi'_{i-1}(x_i) = \Psi'_i(x_i) \\ \Psi''_{i-1}(x_i) = \Psi''_i(x_i)\end{array}\right\}\quad i = 2\cdots n-1 \quad . \tag{3.2.48}$$

Unlike Hermite interpolation, we have not specified the magnitude of the derivatives at the tabular points, but only that they are the same for two adjacent functions $\Psi_{i-1}(x_i)$ and $\Psi_i(x_i)$ at the points $x_i$ all across the tabular range. Only at the end points have we made no restrictions. Requiring the first two derivatives of adjacent polynomials to be equal where they overlap will guarantee that the overall effect of the splines will be to generate a smoothly varying function over the entire tabular range. Since all the interpolative polynomials are cubics, their third derivatives are constants throughout the interval $\Delta x_i$ so that

$$\Psi_i'''(x_i) = \Psi_i'''(x_{i+1}) = \text{const.}, \quad i = 1\cdots n-1 \quad . \tag{3.2.49}$$

Thus the specification of the functional value and equality of the first two derivatives of adjacent functions essentially forces the value of the third derivative on each of the functions $\Psi_i(x)$. This represents n-1 constraints. However, the particular value of that constant for all polynomials is not specified so that this really represents only n-2 constraints. In a similar manner, the specification of the equality of the derivative of two adjacent polynomials for n-2 points represents another n-2 constraints. Since two derivatives are involved we have an additional 2n-4 constraints bringing the total to 4n-6. However, there were 4n-4 constants to be determined in order that all the cubic splines be specified. Thus the system as specified so far is under-determined. Since we have said nothing about the end points it seems clear that that is where the added constraints must be made. Indeed, we shall see that additional constraints must be placed either on the first or second derivative of the function at the end points in order that the problem have a unique solution. However, we shall leave the discussion of the specification of the final two constraints until we have explored the consequences of the 4n-6 constraints we have already developed.

Since the value of the third derivative of any cubic is a constant, the constraints on the equality of the second derivatives of adjacent splines require that the constant be the same for all splines. Thus the second derivative for all splines will have the form

$$\Psi_i''(x) = ax + b \quad . \tag{3.2.50}$$

If we apply this form to two successive tabular points, we can write

$$\left.\begin{array}{l}\Psi''_i(x_i) = ax_i + b = Y''_i \\ \Psi''_{i+1}(x_{i+1}) = ax_{i+1} + b = Y''_{i+1}\end{array}\right\}\quad . \tag{3.2.51}$$

Here we have introduced the notation that $\Psi''_i(x_i)=Y''_i$. The fact of the matter is that $Y''_i$ doesn't exist. We have no knowledge of the real values of the derivatives of the tabular function anywhere. All our constraints are applied to the interpolative polynomials $\Psi_i(x)$ otherwise known as the cubic splines. However, the notation is clear, and as long as we keep the philosophical distinction clear, there should be no confusion about what $Y''_i$ means. In any event they are unknown and must eventually be found. Let us press on and solve equations (3.2.51) for a and b getting

$$\left.\begin{aligned} a &= (Y''_i+Y''_{i+1})/(x_i-x_{i+1}) = (Y''_i+Y''_{i+1})/\Delta x_i \\ b &= Y''_i - x_i(Y''_{i+1}-Y''_i)/\Delta x_i \end{aligned}\right\} \quad . \qquad (3.2.52)$$

Substituting these values into equation (3.2.50) we obtain the form of the second derivative of the cubic spline as

$$\Psi''_i(x) = [Y''_{i+1}(x-x_i) - Y''_i(x-x_{i+1})]/\Delta x_i \quad . \qquad (3.2.53)$$

Now we may integrate this expression twice making use of the requirement that the function and its first derivative are continuous across a tabular entry point, and evaluate the constants of integration to get

$$\Psi_i(x) = \{Y_i - Y''_i[(\Delta x_i)^2-(x_{i+1}-x)^2]/6\}[(x_{i+1}-x)/\Delta x_i] - \{Y_{i+1} - Y''_{i+1}[(\Delta x_i)^2-(x_i-x)^2]/6\}[(x_i-x)/\Delta x_i] \quad . \qquad (3.2.54)$$

This fairly formidable expression for the cubic spline has no quadratic term and depends on those unknown constants $Y''_i$.

To get equation (3.2.54) we did not explicitly use the constraints on $\Psi'_i(x)$ so we can use them now to get a set of equations that the constants $Y''_i$ must satisfy. If we differentiate equation (3.2.54) and make use of the condition on the first derivative that

$$\Psi'_{i-1}(x_i) = \Psi'_i(x_i) \quad , \qquad (3.2.55)$$

we get after some algebra that

$$Y''_{i-1}\Delta x_{i-1}+2Y''_i(\Delta x_{i-1}+\Delta x_i)+Y''_{i+1}\Delta x_i = 6[(Y_{i+1}-Y_i)/\Delta x_i + (Y_i-Y_{i-1})/\Delta x_{i-1}] \quad i=2 \cdots n-1 \quad . \qquad (3.2.56)$$

Everything on the right hand side is known from the tabular entries while the left hand side contains three of the unknown constants $Y''_i$. Thus we see that the equations have the form of a tri-diagonal system of equations amenable to fast solution algorithms. Equation (3.2.56) represents n-2 equations in n unknowns clearly pointing out that the problem is still under determined by two constants. If we arbitrarily take $Y''_1=Y''_n=0$, then the splines that arise from the solution of equation (3.2.56) are called *natural splines*. Keeping the second derivative zero will reduce the variation in the function near the end points and this is usually considered desirable. While this arbitrary choice may introduce some error near the end points, the effect of that error will diminish as one moves toward the middle of the tabular range. If one is given nothing more than the tabular entries $Y_i$ and $x_i$, then there is little more that one can do and the natural splines are as good as any other assumption. However, should anything be known about the first or second derivatives at the end points one can make a more rational choice for the last two constants of the problem? For example, if the values of the first derivatives are known at the end points then differentiating equation (3.2.56) and evaluating it at the end points yields two more equations of condition which depend on the end point first derivatives as

$$\left.\begin{aligned} Y''_1 + \tfrac{1}{2}Y''_2 &= 3[(Y_2-Y_1)/\Delta x_1 - Y'_1]/\Delta x_1 \\ Y''_n - Y''_{n-1}/6 &= 2[(Y_{n-1}-Y_n)/\Delta x_{n-1} - Y'_n]/\Delta x_{n-1} \end{aligned}\right\} \quad . \qquad (3.2.57)$$

These two added conditions complete the system of equations without destroying their tri-diagonal form and pose a significant alternative to natural splines should some knowledge of the endpoint derivatives exist. It is clear that any such information at any point in the tabular range could be used to further constrain the system so that a unique solution exists. In the absence of such information one has little choice but to use the aesthetically pleasing natural splines. One may be somewhat disenchanted that it is necessary to appeal to esthetics to justify a solution to a problem, but again remember that we are trying to get "something for nothing" in any interpolation or curve fitting problem. The "true" nature of the solution between the tabular points simply does not exist. Thus we have another example of where the "art of computing" is utilized in numerical analysis.

In order to see the efficacy of splines, consider the same tabular data given in Table 3.1 and investigate how splines would yield a value for the table at x = 4. Unlike Lagrangian interpolation, the constraints that determine the values for the splines will involve the entire table. Thus we shall have to set up the equations specified by equation (3.2.56). We shall assume that natural splines will be appropriate for the example so that

$$Y''_0 = Y''_5 = 0 \quad . \tag{3.2.58}$$

For i = 1, equation (3.2.56) and the tabular values from table 3.1 yield

$$4Y''_1 + Y''_2 = 6[(8-3)/1 + (3-1)/1] = 42 \ \ , \ i=1 \ , \tag{3.2.59}$$

and the entire system of linear equations for the $Y''_i$'s can be written as

$$\begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 6 & 2 & 0 \\ 0 & 2 & 10 & 3 \\ 0 & 0 & 3 & 10 \end{pmatrix} \begin{pmatrix} Y''_1 \\ Y''_2 \\ Y''_3 \\ Y''_4 \end{pmatrix} = \begin{pmatrix} 42 \\ 18 \\ -16 \\ -7 \end{pmatrix} \quad . \tag{3.2.60}$$

The solution for this tri-diagonal system can be found by any of the methods described in Chapter 2, but it is worth noting the increase in efficiency afforded by the tri-diagonal nature of the system. The solution is given in Table 3.3.

The first item to note is that the assumption of natural splines may not be the best, for the value of $Y_1'' \times 10$ is significantly different from the zero assumed for $Y_0''$. The value of Y" then proceeds to drop smoothly toward the other boundary implying that the assumption of $Y''_5 = 0$ is pretty good. Substituting the solution for $Y''_i$ into equation (3.2.54) we get that

$$\Psi_2(4) = \{8 - 1.9876[4-(5-4)^2]/6\}(4-3)/2 \ - \{4 - (-1.9643)[4-(3-4)^2]/6\}(3-4)/2 = 5.9942 \ . \tag{3.2.61}$$

As can be seen from Table 3.3, the results for the natural cubic splines are nearly identical to the linear interpolation, and are similar to that of the second parabolic Lagrangian interpolation. However, the most appropriate comparison would be with the cubic Lagrangian interpolation ${}^3{}_1\Phi(4)$ as both approximating functions are cubic polynomials. Here the results are quite different illustrating the importance of the constraints on the derivatives of the cubic splines. The Lagrangian cubic interpolation utilizes tabular information for $2 \dot{\le} x \dot{\le} 8$ in order to specify the interpolating polynomial. The splines rely on the more local information involving the function and its derivatives specified in the range 3x5. This minimizes the large tabular variations elsewhere in the table that affect the Lagrangian cubic polynomial and make for a smoother functional variation. The negative aspect of the spline approach is that it requires a solution throughout the table. If the number of tabular entries is large and the required number of interpolated values is small, the additional numerical effort maybe difficult to justify. In the next section we shall find esthetics and efficiency playing an even larger role in choosing the appropriate approximating method.

**Table 3.3**

**A Comparison of Different Types of Interpolation Formulae**

| I | X | ${}^1_2Y_i$ | ${}^1_1\Phi(4)$ | ${}^2_1\Phi(4)$ | ${}^2_2\Phi(4)$ | ${}^3_1\Phi(4)$ | $\Delta x_i$ | $Y''_i$ | $\Psi_2(4)$ | $R_{1,2,3,4}$ | ${}^2_{1,2}\Phi_w(4)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | | | | | 1 | 0.0000 | | | |
| 1 | 2 | 3 | | | | | 1 | 10.003 | | | |
| 2 | 3 | 8 | | | | | 2 | 1.988 | | | |
| | 4 | | 6.000 | 8.333 | 5.733 | 7.467 | | | 5.994 | 5.242 | 6.000 |
| 3 | 5 | 4 | | | | | 3 | -1.965 | | | |
| 4 | 8 | 2 | | | | | 2 | -0.111 | | | |
| 5 | 10 | 1 | | | | | -- | -0.000 | | | |

### d. *Extrapolation and Interpolation Criteria*

So far we have obtained several different types of interpolation schemes, but said little about choosing the degree of the polynomial to be used, or the conditions under which one uses Lagrange interpolation or splines to obtain the information missing from the table. The reason for this was alluded to in the previous paragraph - there is no correct answer to the question. One can dodge the philosophical question of the "correctness" of the interpolated number by appealing to the foundations of polynomial approximation - namely that to the extent that the function represented by the tabular points can be represented by a polynomial, the answer is correct. But this is indeed a dodge. For if it were true that the tabular function was indeed a polynomial, one would simply use the interpolation scheme to determine the polynomial that fit the entire table and use it. In science, one generally does know something about the nature of a tabular function. For example, many such tables result from a complicated computation of such length that it is not practical to repeat the calculation to obtain additional tabular values. One can usually guarantee that the results of such calculations are at least continuous differentiable functions. Or if there are discontinuities, their location is known and can be avoided. This may not seem like much knowledge, but it guarantees that one can locally approximate the table by a polynomial. The next issue is what sort of polynomial should be used and over what part of the tabular range.

In section 3.1 we pointed out that a polynomial can have a very general form [see equation (3.1.1)]. While we have chosen our basis functions $\varphi_i(x)$ to be $x^i$ for most of the discussion, this need not have been the case. Interpolation formulae of the type developed here for $x^i$ can be developed for any set of basis functions $\varphi_i(x)$. For example, should the table exhibit exponential growth with the independent variable, it might be advisable to choose

$$\varphi_i(x) = e^{i\alpha x} = [e^{\alpha x}]^i \; z^i \; . \tag{3.2.62}$$

The simple transformation of $z = e^{\alpha x}$ allows all previously generated formulae to be immediately carried over to the exponential polynomials. The choice of $\alpha$ will be made to suit the particular table. In general, it is far better to use basis functions $\varphi_i(x)$ that characterize the table than to use some set of functions such as the convenient $x^i$ and a larger degree for interpolation. One must always make the choice between fitting the tabular form and using the *lowest* degree polynomial possible. The choice of basis functions that have the proper form will allow the use of a lower degree polynomial.

Why is it so desirable to choose the lowest degree polynomial for interpolation? There is the obvious reason that the lower the degree the faster the computation and there are some cases where this may be an overriding concern. However, plausibility of the result is usually the determining factor. When one fits a polynomial to a finite set of points, the value of the polynomial tends to oscillate between the points of constraint. The higher the degree of the polynomial, the larger is the amplitude and frequency of these oscillations. These considerations become even more important when one considers the use of the interpolative polynomial outside the range specified by the tabular entries. We call such use *extrapolation* and it should be done with only the greatest care and circumspection. It is a fairly general characteristic of polynomials to vary quite rapidly outside the tabular range to which they have been constrained. The variation is usually characterized by the largest exponent of the polynomial. Thus if one is using polynomials of the forth degree, he/she is likely to find the interpolative polynomial varying as $x^4$ immediately outside the tabular range. This is likely to be unacceptable. Indeed, there are some who regard any extrapolation beyond the tabular range that varies more than linearly to be unjustified. There are, of course, exceptions to such a hard and fast rule. Occasionally asymptotic properties of the function that yield the tabular entries are known, then extrapolative functions that mimic the asymptotic behavior maybe justifiable.

There is one form of extrapolation that reduces the instabilities associated with polynomials. It is a form of approximation that abandons the classical basis for polynomial approximation and that is approximation by *rational functions* or more specifically *quotient polynomials*. Let us fit such a function through the (k − i +1) points i → k. Then we can define a quotient polynomial as

$$R_{i,i+1,\cdots,i+k}(x) = \frac{P(x)}{Q(x)} = \frac{\sum\limits_{j=0} a_0 x^j}{\sum\limits_{j=0} b_0 x^j} \quad . \tag{3.2.63}$$

This function would appear to have (m+n+2) free parameters, but we can factor $a_0$ from the numerator and $b_0$ from the denominator so that only their ratio is a free parameter. Therefore there are only (m+n+1) free parameters so we must have

$$k+1 = m+n+1 \; , \tag{3.2.64}$$

functional points to determine them. However, the values of n and m must also be specified separately.

Normally the determination of the coefficients of such a function is rather difficult, but Stoer and Bulirsch[3] have obtained a recurrence formula for the value of the function itself, which is

$$\left.\begin{aligned}
R_{i,i+1,\cdots,i+k}(x) &= R_{i+1,\cdots,i+k}(x)\frac{R_{i+1,\cdots,i+k}(x)-R_{i,\cdots,i+k-1}(x)}{\left[\dfrac{(x-x_i)}{(x-x_{i+k})}\right]\left[1-\dfrac{R_{i+1,\cdots,i+k}(x)-R_{i,i+1,\cdots,i+k-1}(x)}{R_{i+1,\cdots,i+k}(x)-R_{i+1,\cdots,i+k-1}(x)}\right]-1} \\
R_{i,i} &= f(x_i) \\
R_{i,k} &= 0\,, \quad k<1
\end{aligned}\right\} \quad . \tag{3.2.65}$$

This recurrence relation produces a function where n = m if the number of points used is odd, but where m = n+1 should the number of points be even. However, its use eliminates the need for actually knowing the values of the coefficients as the relationship gives the value of the approximated function itself. That is

$$f(x) \cong R_{i,i+1,\cdots,i+k} \quad . \tag{3.2.66}$$

Equation (3.2.65) conceals most of the difficulties of using rational functions or quotient polynomials. While the great utility of such approximating functions are their stability for extrapolation, we shall demonstrate their use for interpolation so as to compare the results with the other forms of interpolation we have investigated. Since the bulk of the other methods have four parameters available for the specification of the interpolating polynomial (i.e. they are cubic polynomials), we shall consider a quotient polynomial with four free parameters. This will require that we use four tabular entries which we shall choose to bracket the point x = 4 symmetrically. Such an approximating function would have the form

$$R_{1,2,3,4}(x) = (ax+b)/(\alpha x+\beta) \quad . \tag{3.2.67}$$

However, the recursive form of equation (3.2.65) means that we will never determine the values of a, b, α, and β. The subscript notation used in equations (3.2.63) – (3.2.66) is designed to explicitly convey the recursive nature of the determination of the interpolative function. Each additional subscript denotes a successive "generation" in the development of the final result. One begins with the tabular data and the second of equations (3.2.65). Taking the data from table 3.3 so that $f(x_i) = Y_i$, we get for the second generation that represents the interpolative value at x = 4

$$\left.\begin{aligned}
R_{1,2}(x) &= R_{2,2}(x) + \frac{R_{2,2}(x) - R_{1,1}(x)}{\left[\frac{(x-x_1)}{(x-x_2)}\right]\left[1 - \frac{R_{2,2}(x) - R_{1,1}(x)}{R_{2,2}(x)}\right] - 1} = 8 + \left[\frac{8-3}{\left[\frac{4-2}{4-3}\right]\left[1-\frac{8-3}{8}\right]-1}\right] = -12 \\
R_{2,3}(x) &= R_{3,3}(x) + \frac{R_{3,3}(x) - R_{2,2}(x)}{\left[\frac{(x-x_2)}{(x-x_3)}\right]\left[1 - \frac{R_{3,3}(x) - R_{2,2}(x)}{R_{3,3}(x)}\right] - 1} = 4 + \left[\frac{4-8}{\left[\frac{4-3}{4-5}\right]\left[1-\frac{4-8}{4}\right]-1}\right] = +\frac{16}{3} \\
R_{3,4}(x) &= R_{4,4}(x) + \frac{R_{4,4}(x) - R_{3,3}(x)}{\left[\frac{(x-x_3)}{(x-x_4)}\right]\left[1 - \frac{R_{4,4}(x) - R_{3,3}(x)}{R_{4,4}(x)}\right] - 1} = 2 + \left[\frac{2-4}{\left[\frac{4-2}{4-5}\right]\left[1-\frac{2-4}{4}\right]-1}\right] = +\frac{26}{5}
\end{aligned}\right\} . \quad (3.2.68)$$

The third generation will contain only two terms so that

$$\left.\begin{aligned}
R_{1,2,3}(x) &= R_{2,3}(x) + \frac{R_{2,2}(x) - R_{1,1}(x)}{\left[\frac{(x-x_1)}{(x-x_2)}\right]\left[1 - \frac{R_{2,2}(x) - R_{1,1}(x)}{R_{2,3}(x) - R_{2,2}(x)}\right] - 1} \\
R_{2,3,4}(x) &= R_{3,4}(x) + \frac{R_{3,4}(x) - R_{2,3}(x)}{\left[\frac{(x-x_2)}{(x-x_4)}\right]\left[1 - \frac{R_{3,4}(x) - R_{2,3}(x)}{R_{3,4}(x) - R_{3,3}(x)}\right] - 1}
\end{aligned}\right\} . \quad (3.2.69)$$

Finally the last generation will have the single result.

$$\left.R_{1,2,3,4}(x) = R_{2,3,4}(x) + \frac{R_{2,3,4}(x) - R_{1,2,3}(x)}{\left[\frac{(x-x_1)}{(x-x_4)}\right]\left[1 - \frac{R_{2,3,4}(x) - R_{1,2,3}(x)}{R_{2,3,4}(x) - R_{2,3,3}(x)}\right] - 1}\right\} . \quad (3.2.70)$$

We can summarize this process neatly in the form of a "difference" Table (similar to Table 3.2 and Table 4.2) below.

Note how the recursion process drives the successive 'generations' of R toward the final result. This is a clear demonstration of the stability of this sort of scheme. It is this type of stability that makes the method desirable for extrapolation. In addition, such recursive procedures are very easy to program and quite fast in execution. The final result is given in equation (3.2.70), tabulated for comparison with other methods in Table 3.3, and displayed in Figure 3.2. This result is the smallest of the six results listed indicating that the rapid tabular variation of the middle four points has been minimized. However, it still compares favorably with the second parabolic Lagrangian interpolation. While there is not a great deal of differentiation between these methods for interpolation, there is for extrapolation. The use of quotient polynomials for extrapolation is vastly superior to the use of polynomials, but one should always remember that one is basically after "free-lunch" and that more sophisticated is not necessarily better. Generally, it is risky to extrapolate any function far beyond one typical tabular spacing.

We have seen that the degree of the polynomial that is used for interpolation should be as low as possible to avoid unrealistic rapid variation of the interpolative function. This notion of providing a general "smoothness" to the function was also implicit in the choice of constraints for cubic splines. The constraints at the interior tabular points guarantee continuity up through the second derivative of the interpolative function throughout the full tabular range. The choice of $Y''_1 = Y''_n = 0$ that produces "natural" splines means that the interpolative function will vary no faster than linearly near the endpoints. In general, when one has to make an assumption concerning unspecified constants in an interpolation scheme, one chooses them so as to provide a slowly varying function. The extension of this concept to more complicated interpolation schemes is illustrated in the following highly successful interpolation algorithm.

## Table 3.4

### Parameters for Quotient Polynomial Interpolation

| I | X | YI | $R_{I,\,I}$ | $R_{I,\,I+1,}$ | $R_{I,\,I+1,\,I+2}$ | $R_{I,\,I+1,\,I+2,\,I+3}$ |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | | | |
| | | | | 0 | | |
| 1 | 2 | 3 | 3 | | 0 | |
| | | | | -12 | | |
| 2 | 3 | 8 | 8 | | 6.5714 | 0 |
| | | | | | | |
| | 4 | | | +16/3 | | 5.2147 |
| | | | | | | |
| 3 | 5 | 4 | 4 | | 5.3043 | 0 |
| | | | | +26/5 | | |
| 4 | 8 | 2 | 2 | | 0 | |
| | | | | 0 | | |
| 5 | 10 | 1 | 0 | | | |

One of the most commonly chosen polynomials to be used for interpolation is the parabola. It tends not to vary rapidly and yet is slightly more sophisticated than linear interpolation. It will clearly require three tabular points to specify the three arbitrary constants of the parabola. One is then confronted with the problem of which of the two intervals between the tabular points should the point to be interpolated be placed. A scheme that removes this problem while far more importantly providing a gently varying function proceeds as follows: Use four points symmetrically placed about the point to be interpolated. But instead of fitting a cubic to these four points, fit two parabolas, one utilizing the first three points and one utilizing the last three points. At this point one exercises an artistic judgment. One may choose to use the parabola with that exhibits the least curvature (i.e. the smallest value of the quadratic coefficient).

However, one may combine both polynomials to form a single quadratic polynomial where the contribution of each is weighted inversely by its curvature. Specifically, one could write this as

$$ {}_k^2\Phi_w(x) = \{a_{k+1}\,[{}_k^2\Phi(x)] + a_k\,[{}_{k+1}^{\;\;2}\Phi(x)]\}/(a_k+a_{k+1}) \quad , \tag{3.2.71}$$

where $a_k$s are the inverse of the coefficient of the $x^2$ term of the two polynomials and are given by

$$ a_k = \sum_{i=k}^{k+2} Y(x_i) \Big/ \prod_{j\neq i}(x_i - x_j) \quad , \tag{3.2.72}$$

and are just twice the inverse of the curvature of that polynomial. The ${}_k\Phi(x)$ are the Lagrange polynomials of second degree and are

$$ {}_k^2\Phi(x) = \sum_{i=k}^{k+2} Y(x_i)L_i(x) \ . \tag{3.2.73}$$

Since each of the ${}_k\Phi(x)$s will produce the value of $Y(x_i)$ when $x = x_i$, it is clear that equation (3.2.71) will produce the values of $Y(x_2)$ and $Y(x_3)$ at the points $x_2$ and $x_3$ adjacent to the interpolative point. The functional behavior between these two points will reflect the curvature of both polynomials giving higher weight to the flatter, or more slowly varying polynomial. This scheme was developed in the 1960s by researchers at Harvard University who needed a fast and reliable interpolation scheme for the construction of model stellar atmospheres. While the justification of this algorithm is strictly aesthetic, it has been found to function well in a wide variety of situations. We may compare it to the other interpolation formulae by applying it to the same data from tables 3.1 and 3.3 that we have used throughout this section. In developing the parabolic Lagrangian formulae in section 3.1, we obtained the actual interpolative polynomials in equations (3.2.15) and (3.2.16). By differentiating these expressions twice, we obtain the $a_k$s required by equation (3.2.71) so that

$$ \left.\begin{aligned} a_1 &= 2\left|P_1''(4)\right|^{-1} = 3/7 \\ a_2 &= 2\left|P_2''(4)\right|^{-1} = 15/4 \end{aligned}\right\} \ . \tag{3.2.74}$$

Substitution of these values into equation (3.2.71) yields a weighted Lagrangian interpolated value of

$$ {}_{1,2}^{\;\;2}\Phi_w = \{[3P_1(4)/7] + [15P_2(4)/4]\}/[(3/7)+(15/4)] = 6.000 \tag{3.2.75}$$

We have evaluated equation (3.2.75) by using the rational fraction values for $P_1(4)$ and $P_2(4)$ which are identical to the interpolative values given in table 3.1. The values for the relative weights given in equation (3.2.74) show that the first parabola will only contribute about 15% to the final answer do to its rapid variation. The more gently changing second parabola contributes the overwhelming majority of the final result reflecting our aesthetic judgment that slowly varying functions are more plausible for interpolating functions. The fact that the result is identical to the result for linear interpolation is a numerical accident. Indeed, had round-off error not been a factor, it is likely that the result for the cubic splines would have also been exactly 6. However, this coincidence points up a common truth: "more sophisticated is not necessarily better".

Although slightly more complicated than quadratic Lagrangian interpolation, this scheme is rather more stable against rapid variation and is certainly more sophisticated than linear interpolation. In my opinion, its only real competition is the use of cubic splines and then only when the entire range of the table is to be used as in curve fitting. Even here there is no clear distinction as to which produces the more appropriate interpolative values, but an edge might be given to cubic splines on the basis of speed depending on the table size and number of required interpolative values.

It is worth taking a last look at the results in Table 3.3. We used the accuracy implied by the tables to provide a basis for the comparison of different interpolative methods. Indeed, some of the calculations were carried out as rational fractions to eliminate round-off error as the possible source of the difference between methods. The plausible values range from about 5.2 to 6.00. However, based on the tabular data, there is no real reason to prefer one value over another. The appropriate choice should revolve around the extent that one should expect an answer of a particular accuracy. None of the tabular data contain more than two significant figures. There would have to be some compelling reason to include more in the final result. Given the data spacing and the tabular variability, even two significant figures are difficult to justify. With that in mind, one could argue persuasively that linear interpolation is really all that is justified by this problem. This is an important lesson to be learned for it lies at the root of all numerical analysis. There is no need to use numerical methods that are vastly superior to the basic data of the problem.

## 3.3 Orthogonal Polynomials

Before leaving this chapter on polynomials, it is appropriate that we discuss a special, but very important class of polynomials known as the *orthogonal polynomials*. Orthogonal polynomials are defined in terms of their behavior with respect to each other and throughout some predetermined range of the independent variable. Therefore the orthogonality of a specific polynomial is not an important notion. Indeed, by itself that statement does not make any sense. The notion of orthogonality implies the existence of something to which the object in question is orthogonal. In the case of polynomials, that something happens to be other polynomials. In section 1.3 we discussed the notion of orthogonality for vectors and found that for a set of vectors to be orthogonal, no element of the set could be expressed in term of the other members of the set. This will also be true for orthogonal polynomials. In the case of vectors, if the set was complete it was said to span a vector space and any vector in that space could be expressed as a linear combination of the orthogonal basis vectors. Since the notion of orthogonality seems to hinge on two things being perpendicular to each other, it seems reasonable to say that two functions $f_1(x)$ and $f_2(x)$ are orthogonal if they are everywhere perpendicular to each other. If we imagine tangent vectors $\vec{t}_1(x)$ and $\vec{t}_2(x)$ defined at every point of each function, then if

$$\vec{t}_1(x) \bullet \vec{t}_2(x) = 0 \quad \forall x \quad , \tag{3.3.1}$$

one could conclude from equation (3.3.1) that $f_1(x)$ and $f_2(x)$ were mutually perpendicular at each value of x. If one considers the range of x to represent an infinite dimension vector space with each value of x representing a dimension so that the vectors $\vec{t}_1(x)$ represented basis vectors in that space, then orthogonality could be expressed as

$$\int_a^b t_1(x)t_2(x)dx = 0 \quad . \tag{3.3.2}$$

Thus, it is not unreasonable to generalize orthogonality of the functions themselves by

$$\int_a^b f_i(x)f_j(x)dx = 0 \quad , \quad i \neq j. \tag{3.3.3}$$

Again, by analogy to the case of vectors and linear transformations discussed in chapter 1 we can define two functions as being orthonormal if

$$\int_a^b w(x)\, f_i(x)f_j(x)\, dx = \delta_{ij} \quad . \tag{3.3.4}$$

Here we have included an additional function w(x) which is called a weight function. Thus the proper statement is that two functions are said to be orthonormal in the interval $a \le x \le b$, relative to a weight function w(x), if they satisfy equation (3.3.4). In this section we shall consider the subset of functions known as polynomials.

It is clear from equation (3.3.4) that orthonormal polynomials come in sets defined by the weight function and range of x. These parameters provide for an infinite number of such sets, but we will discuss only a few of the more important ones. While we will find it relatively easy to characterize the range of the independent variable by three distinct categories, the conditions for the weight function are far less stringent. Indeed the only constraint on w(x) is

$$w(x) > 0 \quad \forall x \in a \le x \le b \quad . \tag{3.3.5}$$

While one can find orthogonal functions for non-positive weight functions, it turns out that they are not unique and therefore not well defined. Simply limiting the weight function to positive definite functions in the interval a-b, still allows for an infinite number of such weight functions and hence an infinite number of sets of orthogonal polynomials.

Let us begin our search for orthogonal polynomials by using the orthogonality conditions to see how such polynomials can be generated. For simplicity, let us consider a finite interval from a to b. Now an orthogonal polynomial $\varphi_i(x)$ will be orthogonal to every member of the set of polynomials other than itself. In addition, we will assume (it can be proven) that the polynomials will form a complete set so that any polynomial can be generated from a linear combination of the orthogonal polynomials of the same degree or less. Thus, if $q_i(x)$ is an arbitrary polynomial of degree i, we can write

$$\int_a^b w(x)\phi_i(x)q_{i-1}(x)\, dx = 0\, . \tag{3.3.6}$$

Now let

$$w(x)\phi_i(x) = \frac{d^i U_i(x)}{dx^i} \equiv U_i^{(i)}(x) \quad . \tag{3.3.7}$$

The function $U_i(x)$ is called the *generating function* of the polynomials $\varphi_i(x)$ and is itself a polynomial of degree 2i so that the ith derivative $U_i^{(i)}$ is an ith degree polynomial. Now integrate equation (3.3.7) by parts i-times to get

$$\int_a^b U_i^{(i)}(x)q_{i-1}(x)\,dx = 0 = \left[U_i^{(i-1)}(x)q_{i-1}(x) - U_i^{(i-2)}(x)q'_{i-1}(x) + \cdots + (-1)^{i-1}U_i(x)q_{i-1}^{(i-1)}(x)\right]\Big|_b^a \quad . \tag{3.3.8}$$

Since $q_i(x)$ is an arbitrary polynomial each term in equation (3.3.8) must hold separately so that

$$\left.\begin{aligned} U_i(a) = U'_i(a) = \cdots = U_i^{(i-1)}(a) = 0 \\ U_i(b) = U'_i(b) = \cdots = U_i^{(i-1)}(b) = 0 \end{aligned}\right\} \quad . \tag{3.3.9}$$

Since $\phi_i(x)$ is a polynomial of degree i we may differentiate it i+1 times to get

$$\frac{d^{i+1}}{dx^{i=1}}\left[\frac{1}{w(x)}\frac{d^i U_i(x)}{dx^i}\right] = 0 \quad . \tag{3.3.10}$$

This constitutes a differential equation of order 2i+1 subject to the 2i boundary conditions given by equation (3.3.9). The remaining condition required to uniquely specify the solution comes from the normalization constant required to make the integral of $\phi_i^2(x)$ unity. So at this point we can leave $U_i(x)$ uncertain by a scale factor. Let us now turn to the solution of equation (3.3.10) subject to the boundary conditions given by equation (3.3.9) for some specific weight functions w(x).

### a. The Legendre Polynomials

Let us begin by restricting the range to $-1 \lesssim x \lesssim 1$ and taking the simplest possible weight function, namely

$$w(x) = 1 \quad , \tag{3.3.11}$$

so that equation (3.3.9) becomes

$$\frac{d^{2i+1}}{dx^{2i=1}}[U_i(x)] = 0 \quad . \tag{3.3.12}$$

Since $U_i(x)$ is a polynomial of degree 2i, an obvious solution which satisfies the boundary conditions is

$$U_i(x) = C_i(x^2-1)^i \quad . \tag{3.3.13}$$

Therefore the polynomials that satisfy the orthogonality conditions will be given by

$$\phi_i(x) = C_i\frac{d^i(x^2-1)^i}{dx^i} \quad . \tag{3.3.14}$$

If we apply the normalization criterion we get

$$\int_{-1}^{+1}\phi_i^2(x)\,dx = 1 = C_i\int_{-1}^{+1}\left[\frac{d^i(x^2-1)}{dx^i}\right]dx \quad , \tag{3.3.15}$$

so that

$$C_i = [2^i i!]^{-1} \quad . \tag{3.3.16}$$

We call the orthonormal polynomials with that normalization constant and satisfying equation (3.3.14) the *Legendre polynomials* and denote them by

$$P_i(x) = [2^i i!]^{-1} d^i(x^2-1)^i/dx^i \quad . \tag{3.3.17}$$

One can use equation (3.3.17) to verify that these polynomials will satisfy the recurrence relation

$$\left.\begin{array}{l} P_{i+1}(x) = \left[\dfrac{2i+1}{i+1}\right] xP_i(x) - \left[\dfrac{i}{i+1}\right] P_{i-1}(x) \\ P_0(x) = 1 \\ \\ P_1(x) = x \end{array}\right\} \quad , \tag{3.3.18}$$

The set of orthogonal polynomials that covers the finite interval from -1 to +1 and whose members are orthogonal relative to the weight function w(x) = 1 are clearly the simplest of the orthogonal polynomials. One might be tempted to say that we have been unduly restrictive to limit ourselves to such a specific interval, but such is not the case. We may transform equation (3.3.15) to any finite interval by means of a linear transformation of the form

$$y(x) = x[(b-a)/2] + (a+b)/2 \quad , \tag{3.3.19}$$

so that we obtain an integral

$$\left[\frac{2}{b-a}\right] \int_a^b \phi_i(y)\phi_j(y)\, dy = \delta_{ij} \quad , \tag{3.3.20}$$

that resembles equation (3.3.4). Thus the Legendre polynomials form an orthonormal set that spans any finite interval relative to the unit weight function.

### b. The Laguerre Polynomials

While we noted that the Legendre polynomials could be defined over any finite interval since the linear transformation required to reach such as interval didn't affect the polynomials, we had earlier mentioned that there are three distinct intervals that would have to be treated differently. Here we move to the second of these - the semi-infinite interval where $0 \le x \le \infty$. Clearly the limits of this interval cannot be reached from any finite interval by a linear transformation. A non-linear transformation that would accomplish that result would destroy the polynomic nature of any polynomials obtained in the finite interval. In addition, we shall have to consider a weight function that asymptotically approaches zero as x□□ as any polynomials in x will diverge making it impossible to satisfy the normalization condition. Perhaps the simplest weight function that will force a diverging polynomial to zero as $x \rightarrow \infty$ is $e^{-\alpha x}$**.** Therefore our orthogonal polynomials will take the form

$$\phi_I(x) = e^{\alpha x} \frac{d^i U_i(x)}{dx^i} \quad , \tag{3.3.21}$$

where the generating function will satisfy the differential equation

$$\frac{d^{i+1}}{dx^{i=1}} \left[ e^{\alpha x} \frac{d^i U_i(x)}{dx^i} \right] = 0 \quad , \tag{3.3.22}$$

and be subject to the boundary conditions

$$\left.\begin{array}{l} U_i(0) = U_i'(0) = \cdots = U_i^{(i-1)}(0) = 0 \\ U_i(\infty) = U_i'(\infty) = \cdots = U_i^{(i-1)}(\infty) = 0 \end{array}\right\} . \tag{3.3.23}$$

When subjected to those boundary conditions, the general solution to equation (3.3.22) will be

$$U_i(x) = C_i x^i e^{-\alpha x} \ , \tag{3.3.24}$$

so that the polynomials can be obtained from

$$\phi_i(x) = \frac{e^{\alpha x}}{i!}\frac{d^i(x^i e^{-\alpha x})}{dx^i} \ , \tag{3.3.25}$$

If we set $\alpha = 1$, then the resultant polynomials are called the *Laguerre polynomials* and when normalized have the form

$$\mathbf{L}_i = \frac{e^x}{i!}\frac{d^i(x^i e^{-x})}{dx^i} \ , \tag{3,3,26}$$

and will satisfy the recurrence relation

$$\left.\begin{array}{l} \mathbf{L}_{i+1}(x) = \left[\dfrac{2i+1-x}{i+1}\right]\mathbf{L}_i(x) - \left[\dfrac{i}{i+1}\right]\mathbf{L}_{i-1}(x) \\ \mathbf{L}_0(x) = 1 \\ \mathbf{L}_1(x) = 1-x \end{array}\right\} . \tag{3.3.27}$$

These polynomials form an orthonormal set in the semi-infinite interval relative to the weight function $e^{-x}$.

### c. *The Hermite Polynomials*

Clearly the remaining interval that cannot be reached from either a finite interval or semi-infinite interval by means of a linear transformation is the full infinite interval $-\infty \le x \le +\infty$. Again we will need a weight function that will drive the polynomial to zero at both end points so that it must be symmetric in x. Thus the weight function for the semi-infinite interval will not do. Instead, we pick the simplest symmetric exponential $e^{\alpha^2 x^2}$, which leads to polynomials of the form

$$\phi_i(x) = e^{\alpha^2 x^2}\frac{d^i U_i(x)}{dx^i} \ , \tag{3.3.28}$$

that satisfy the differential equation

$$\frac{d^{i+1}}{dx^i}\left[e^{\alpha^2 x^2}\frac{d^i U_i(x)}{dx^i}\right] = 0 \ , \tag{3.3.29}$$

subject to the boundary conditions

$$U_i(\pm\infty) = U_i'(\pm\infty) = \cdots = U_i^{(i-1)}(\pm\infty) = 0 \ . \tag{3.3.30}$$

This has a general solution satisfying the boundary conditions that look like

$$U_i(x) = C_i e^{-\alpha^2 x^2} \quad , \tag{3.3.31}$$

which when normalized and with α = 1, leads to the *Hermite polynomials* that satisfy

$$H_i(x) = (-1)^i e^{x^2} \frac{d^i e^{-x^2}}{dx^i} \quad . \tag{3.3.32}$$

## Table 3.5

**The First Five Members of the Common Orthogonal Polynomials**

| I | $P_I(X)$ | $L_I(X)$ | $H_I(X)$ |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | x | 1-x | 2x |
| 2 | $(3x^2-1)/2$ | $(2-4x+x^2)/2$ | $2(2x^2-1)$ |
| 3 | $x(5x^2-3)/2$ | $(6-18x+9x^2-x^3)/6$ | $4x(2x^2-3)$ |
| 4 | $(35x^4-30x^2+3)/8$ | $(24-96x+72x^2-6x^3+x^4)/24$ | $4(4x^4-16x^2+3)$ |

Like the other polynomials, the Hermite polynomials can be obtained from a recurrence relation. For the Hermite polynomials that relation is

$$\left.\begin{aligned} &H_{i+1}(x) = 2xH_i(x) - 2iH_{i-1}(x) \\ &H_0(x) = 1 \\ &H_1(x) = 2x \end{aligned}\right\} \quad . \tag{3.3.31}$$

.We have now developed sets of orthonormal polynomials that span the three fundamental ranges of the real variable. Many other polynomials can be developed which are orthogonal relative to other weight functions, but these polynomials are the most important and they appear frequently in all aspects of science.

### d. *Additional Orthogonal Polynomials*

There are as many additional orthogonal polynomials as there are positive definite weight functions. Below we list some of those that are considered to be classical orthogonal polynomials as they turn up frequently in mathematical physics. A little inspection of Table 3.6 shows that the Chebyschev polynomials are special cases of the more general Gegenbauer or Jacobi polynomials. However, they turn up sufficiently frequently that it is worth saying more about them. They can be derived from the generating function in the same manner that the other orthogonal polynomials were, so we will only quote the results. The Chebyschev polynomials of the first kind can be obtained from the reasonably simple trigonometric formula

$$T_i(x) = \cos[i \cos^{-1}(x)] \quad . \tag{3.3.34}$$

## Table 3.6

### Classical Orthogonal Polynomials of the Finite Interval

| Name | Weight Function w(x) |
|---|---|
| Legendre | 1 |
| Gegenbauer or Ultraspherical | $(1-x^2)^{\lambda-\frac{1}{2}}$ |
| Jacobi or Hypergeometric | $(1-x)^{\alpha}(1-x)^{\beta}$ |
| Chebyschev of the first kind | $(1-x^2)^{-\frac{1}{2}}$ |
| Chebyschev of the second kind | $(1-x^2)^{+\frac{1}{2}}$ |

However, in practice they are usually obtained from a recurrence formula similar to those for the other polynomials. Specifically

$$\left.\begin{aligned} T_{i+1}(x) &= 2xT_i(x) - T_{i-1}(x) \\ T_0(x) &= 1 \\ T_1(x) &= x \end{aligned}\right\} . \tag{3.3.35}$$

The Chebyschev polynomials of the second kind are represented by the somewhat more complicated trigonometric formula

$$V_i(x) = \sin[(i+1)\cos^{-1}(x)]/\sin[\cos^{-1}(x)] \ , \tag{3.3.36}$$

and obey the same recurrence formula as Chebyschev polynomials of the first kind so

$$\left.\begin{aligned} V_{i+1}(x) &= 2xV_i(x) - V_{i-1}(x) \\ V_0(x) &= 1 \\ V_1(x) &= 2x \end{aligned}\right\} . \tag{3.3.37}$$

Only the starting values are slightly different. Since they may be obtained from a more general class of polynomials, we should not be surprised if there are relations between them. There are, and they take the form

$$\left.\begin{aligned} T_i(x) &= V_i(x) - xV_{i-1}(x) \\ (1-x^2)V_{i-1}(x) &= xT_i(x) - T_{i+1}(x) \end{aligned}\right\} . \tag{3.3.38}$$

Since the orthogonal polynomials form a complete set enabling one to express an arbitrary polynomial in terms of a linear combination of the elements of the set, they make excellent basis functions for interpolation formulae. We shall see in later chapters that they provide a basis for curve fitting that provides great numerical stability and ease of solution. In the next chapter, they will enable us to generate formulae to evaluate integrals that yield great precision for a minimum of effort. The utility of these functions is of central importance to numerical analysis. However, all of the polynomials that we have discussed so far form orthogonal sets over a continuous range of x. Before we leave the subject of orthogonality, let us consider a set of functions, which form a complete orthogonal set with respect to a *discrete* set of points in the finite interval.

### *e. The Orthogonality of the Trigonometric Functions*

At the beginning of the chapter where we defined polynomials, we represented the most general polynomial in terms of basis functions $\varphi_i(x)$. Consider for a moment the case where

$$\varphi_i(x) = \sin(i\pi x) \quad . \tag{3.3.39}$$

Now integration by parts twice, recovering the initial integral but with a sign change, or perusal of any good table of integrals[4] will convince one that

$$\int_{-1}^{+1} \sin(k\pi x)\sin(j\pi x)\,dx = \int_{-1}^{+1} \cos(k\pi x)\cos(j\pi x)\,dx \quad = \quad \delta_{kj} \; . \tag{3.3.40}$$

Thus sines and cosines form orthogonal sets of functions of the real variable in the finite interval. This will come as no surprise to the student with some familiarity with Fourier transforms and we will make much of it in chapters to come. But what is less well known is that

$$\frac{1}{N}\sum_{x=0}^{2N-1} \sin(k\pi x/N)\sin(j\pi x/N) = \frac{1}{N}\sum_{x=0}^{2N-1} \cos(k\pi x/N)\cos(j\pi x/N) = \delta_{kj} \;, \quad 0 < (k+j) < 2N \;, \tag{3.3.41}$$

which implies that these functions also form an orthogonal set on the finite interval for a discrete set of points. The proof of this result can be obtained in much the same way as the integral, but it requires some knowledge of the finite difference calculus (see Hamming[5] page 44, 45). We shall see that it is this discrete orthogonality that allows for the development of Fourier series and the numerical methods for the calculation of Power Spectra and "Fast Fourier Transforms". Thus the concept of orthogonal functions and polynomials will play a role in much of what follows in this book.

# Chapter 3 Exercises

1. Find the roots of the following polynomial

   $2x^5 - 225x^4 + 2613x^3 - 11516x^2 + 21744x - 14400 = P(x)$ ,

   a. by the Graffe Root-squaring method,
   b. any interative method,
   c. then compare the accuracy of the two methods.

2. Find the roots of the following polynomials:

   a. $P(x) = x^4 - 7x^3 + 13x^2 - 7x + 12$

   b. $P(x) = 2x^4 - 15x^3 + 34x^2 - 25x + 14$

   c. $P(x) = 4x^4 - 9x^3 - 12x^2 - 35x - 18$

   d. $P(x) = +0.0021(x^3+x) + 1.000000011x^2 + 0.000000011$.
   Comment of the accuracy of your solution.

3. Find Lagrangian interpolation formulae for the cases where the basis functions are

   a. $\varphi_i(x) = e^{ix}$

   b. $\varphi_i(x) = \sin(i\pi x/h)$ ,

   where h is a constant interval spacing between the points $x_i$.

4. Use the results from problem 3 to obtain values for f(x) at x=0.5, 0.9 and 10.3 in the following table:

| $x_i$ | $f(x_i)$ |
|---|---|
| 0.0 | 1.0 |
| 0.4 | 2.0 |
| 0.8 | 3.0 |
| 1.2 | 5.0 |
| 2.0 | 3.0 |
| 5.0 | 1.0 |
| 8.0 | 8.0 . |

   Compare with ordinary Lagrangian interpolation for the same degree polynomials and cubic splines. Comment on the result.

5. Given the following table, approximate f(x) by

$$f(x) = \sum_{i=1}^{n} a_i \sin(ix).$$

Determine the "best" value of n for fitting the table. Discuss your reasoning for making the choice you made.

| $x_i$ | $f(x_i)$ |
|---|---|
| 1.0 | +.4546 |
| 2.0 | -.3784 |
| 3.0 | -.1397 |
| 4.0 | +.4947 |
| 5.0 | -.2720 |
| 6.0 | -.2683 |
| 7.0 | +.4953 |
| 8.0 | -.1439 |

6. Find the normalization constants for

   a. Hermite polynomials
   b. Laguerre polynomials
   c. Legendre polynomials that are defined in the interval $-1 \rightarrow +1$.

7. Use the rules for the manipulation of determinants given in chapter 1 (page 8) to show how the Vandermode determinant takes the form given by equation (3.3.7)

8. In a manner similiar to problem 7, show how the Lagrangian polynomials take the form given by equation (3.2.9).

9. Explicitly show how equation (3.2.29) is obtained from equations (3.2.23), (3.2.24), and (3.2.26).

10. Integrate equation (3.2.53) to obtain the tri-diagonal equations (3.2.54). Show explicitly how the constraints of the derivatives of $Y_i$ enter into the problem.

11. By obtaining equation (3.3.18) from equation (3.3.17) show that one can obtain the recurrence relations for orthogonal polynomials from the defining differential equation.

12. Find the generating function for Gegenbauer polynomials and obtain the recurrence relation for them.

13. Show that equation (3.3.41) is indeed correct.

## Chapter 3 References and Supplemental Reading

1. Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., "Numerical Recipies the Art of Scientific Computing" (1986), Cambridge University Press Cambridge, New York, New Rochelle, Melbourne, Sydney.

2. Acton, Forman S., "Numerical Methods That Work", (1970) Harper and Row, New York.

3. Stoer, J. and Bulirsch, R., "Introduction to Numerical Analysis" (1980), Springer-Verlag, New York, §2.2.

4. Gradshteyn, I.S. and Ryzhik,I.M., "Table of Integrals, Series, and Products : corrected and enlarged edition" (1980), (ed. A. Jeffrey), Academic Press, New York, London, Toronto, Sydney, San Francisco, pp 139-140.

5. Hamming, R.W., "Numerical Methods for Scientists and Engineers" (1962) McGraw-Hill Book Co., Inc., New York, San Francisco, Toronto, London.

For an excellent general discussion of polynomials one should read

6. Moursund, D.G., and Duris, C.S., "Elementary Theory and Applications of Numerical Analysis" (1988) Dover Publications, Inc. New York, pp 108-140.

A very complete discussion of classical orthogonal polynomials can be found in

7. Bateman, H., The Bateman Manuscript Project, "Higher Transcendental Functions" (1954) Ed. A. Erdélyi, Vol. 3, McGraw-Hill Book Co., Inc. New York, Toronto, London, pp 153-228.