TABLE A.3
Determination of the first two nonzero roots of $\alpha = \tan\alpha$

Trial	First root x	First root y	Second root x	Second root y
		(a) Newton's Method†		
0	4.40000	1.30368	7.70000	1.25713
1	4.53598	−1.07376	7.73028	−0.31270
2	4.50186	−0.17769	7.72545	−0.01188
3	4.49375	−0.00679	7.72525	−0.00002
4	4.49341	−0.00001	7.72525	−0.00000
5	4.49341	−0.00000		
		(b) Secant Method‡		
0	4.40000	1.30368	7.80000	−10.70682
0	4.50000	−0.13733	7.70000	1.25713
1	4.49047	0.05854	7.71051	0.78849
2	4.49332	0.00184	7.72819	−0.17931
3	4.49341	−000003	7.72491	0.02025
4	4.49341	0.00000	7.72524	0.00047
5			7.72525	−0.00000

†The calculation continues without assistance after the initial trial value has been selected

‡Two x, y pairs are required for each stage of the calculation. After the first trial, the most recently calculated x, y pair was used with whichever of the two previous pairs was closer to the root.

solve for the parameters a and b. We used the secant method to solve these equations.

Consider the two equations

$$f_a(u, v) = 0 \quad \text{and} \quad f_b(u, v) = 0 \tag{A.47}$$

which we wish to solve for u and v. We define the first *partial divided differences*,

$$\begin{aligned} f_{au} &= f_a[u_0, v_0; u_1, v_0] \equiv \frac{f_a(u_1, v_0) - f_a(u_0, v_0)}{u_1 - u_0} \\ f_{av} &= f_a[u_0, v_0; u_0, v_1] \equiv \frac{f_a(u_0, v_1) - f_a(u_0, v_0)}{v_1 - v_0} \\ f_{bu} &= f_b[u_0, v_0; u_1, v_0] \equiv \frac{f_b(u_1, v_0) - f_b(u_0, v_0)}{u_1 - u_0} \\ f_{bv} &= f_b[u_0, v_0; u_0, v_1] \equiv \frac{f_b(u_0, v_1) - f_b(u_0, v_0)}{v_1 - v_0} \end{aligned} \tag{A.48}$$

and, following Equation (A.43), write for a first-order expansion

$$\begin{aligned} f_a(u, v) &= f_a(u_0, v_0) + (u - u_0)f_{au} + (v - v_0)f_{av} \\ \text{and} \quad f_b(u, v) &= f_b(u_0, v_0) + (u - u_0)f_{bu} + (v - v_0)f_{bv} \end{aligned} \tag{A.49}$$

If we assume that f_a and f_b are linear in u and v, we can find a first approximation to the roots by setting $f_a(u, v)$ and $f_b(u, v)$ to zero in Equation (A.49) and solving the two coupled linear equations for u and v:

$$\begin{aligned} uf_{au} + vf_{av} - u_0f_{au} - v_0f_{av} + f_a(u_0, v_0) &= 0 \\ uf_{bu} + vf_{bv} - u_0f_{bu} - v_0f_{bv} + f_b(u_0, v_0) &= 0 \end{aligned} \tag{A.50}$$

Solution by the determinant method gives

$$\begin{aligned} u_2 = u &= (Af_{bv} - Bf_{av})/D \\ \text{and} \quad v_2 = v &= (Bf_{au} - Af_{bu})/D \end{aligned} \tag{A.51}$$

with

$$\begin{aligned} D &= f_{au}f_{bv} - f_{av}f_{bu} \\ A &= -u_0f_{au} - v_0f_{av} + f_a(u_0, v_0) \\ B &= -u_0f_{bu} - v_0f_{bv} + f_b(u_0, v_0) \end{aligned} \tag{A.52}$$

We then repeat the procedure with coordinate pairs (u_1, v_1) and (u_2, v_2), to obtain the next approximation, until the roots have been found to the desired degree of accuracy.

A.6 DATA SMOOTHING

The concept of smoothing is not one that meets with universal approval. The discussion that follows should be considered with one caveat: For rigorously valid least-squares fitting, smoothing is neither desirable nor permissible; however, there are cases where smoothing can be beneficial, and, therefore, the techniques are introduced.

Consider, for example, the discussion of Section 9.2 of the determination of the area under a peak from a least-squares fit to a histogram of the data. Least-squares fitting techniques applied to data that are distributed according to Poisson distributions, rather than Gaussian distributions, underestimate the area of a peak by an amount equal to the value of χ^2. We have seen that we can improve the result by decreasing the value of χ^2 at its minimum. Similarly, if the shape of the fitting function does not exactly simulate that of the parent distribution, a better fit to the data by decreasing χ^2 can yield an improved estimate of the area under a peak.

Another example that might benefit from application of a smoothing algorithm is the parameterization of data for use in a Monte Carlo or other program. In preparing experimental proposals, it is often necessary to estimate yields and distributions based on currently available data. Such data are often sparse and generally must be expressed in parametric form for ease and speed of use in the Monte Carlo simulation program. Smoothing can be useful to average out fluctuations and allow the data to be expressed with a few parameters by a least-squares fit or an interpolation procedure.

In other words, if rigorously valid results are not required, but rather an averaged estimate of the distribution, smoothing may help obtain more reliable estimates. The improvement in the estimate of one parameter must, of course, be accompanied by a decrease in information of some other parameter or parameters.

For example, an improved estimate of the area under a peak would be accompanied by an increased uncertainty in the estimates of the width and position of the peak.

Whatever smoothing or other manipulation is done must conserve the information pertaining to the desired parameters. The averaging techniques that we shall discuss, for example, conserve the area under a peak but not the width of the peak Similarly, this method would be useful for improving the estimate of the constant term of a polynomial but not the coefficients of the other terms.

Data smoothing is similar to the data "smearing" introduced in Chapter 5 to simulate measuring uncertainties in "measurements" generated by a Monte Carlo program. In the Monte Carlo program we used Gaussian smearing; that is, we allowed each event a Gaussian probability distribution about its mean.

In this section, we are dealing with binned data, and thus, for Gaussian smoothing, could consider a Gaussian integration that spreads each event over adjacent bins. Because our object here is to smooth the data, we are at liberty to choose the width of the smearing function to produce the desired degree of uniformity in the data, limited by the requirement that we do not damage the very variable we are trying to study.

The binomial distribution is a useful smoothing function. Suppose we want to smooth low statistics experimental data that follow a Gaussian peak in a way that preserves the area under the peak. Let us assume that the background slope is gentle enough that smoothing will not affect its determination drastically.

We can approximate the Gaussian peak with a binomial distribution with $p = \frac{1}{2}$ (see Section 2.1):

$$y(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-1/2}\left(\frac{x-\mu}{\sigma}\right)^2 \simeq \left(\frac{1}{2}\right)^n \frac{n!}{X!(n-X)!} \tag{A.53}$$

We can relate the widths σ and the means of the two distributions

$$\sigma_B^2 = np(1-p) = n/4 = \sigma^2 \quad \bar{X} = np = n/2 \quad \bar{x} = \mu \tag{A.54}$$

to find the relationships among the parameters

$$n = 4\sigma^2 \quad X = x - \mu + n/2 = x - \mu + 2\sigma^2 \tag{A.55}$$

We can then express the binomial distribution of Equation (A.53) as

$$y(x) = \left(\frac{1}{2}\right)^n \frac{n!}{(n/2 + x - \mu)!(n/2 - x + \mu)!} \tag{A.56}$$

Let us smooth the data by averaging over adjacent channels with a binomial distribution spanning three channels:

$$y'(x) = 1/4y(x-1) + 1/2y(x) + 1/4y(x+1) \tag{A.57}$$

If we fold this averaging into the distribution of Equation (A.53), the result is also binomial:

$$y'(x) = \left(\frac{1}{2}\right)^{n+2} \frac{(n+2)!}{(n/2 + 1 + x - \mu)!(n/2 + 1 - x + \mu)!} \tag{A.58}$$

The new distribution has the same mean $\bar{x} = \mu$ but a larger width $\sigma'^2 = n'/4 = (n+2)/4$ with the variance increased by $\frac{1}{2}$:

$$\sigma'^2 = \sigma^2 + \tfrac{1}{2} \tag{A.59}$$

Similarly, we could smooth over five channels by using a formula similar to Equation (A.57) but with five terms with coefficients given by the binomial expansion

$$y''(x) = 1/16y(x-2) + 1/4y(x-1) + 3/8y(x) + 1/4y(x+1) + 1/16y(x+2) \tag{A.60}$$

A five-channel smoothing is identical to two successive smoothings over three channels and yields a variance that is increased accordingly, $\sigma''^2 = \sigma^2 + 1$. Any such smoothings over $2n+1$ adjacent channels is equivalent to n smoothings over three channels.

If we apply the smoothing of Equation (A.57) to a Gaussian distribution, the resulting distribution will also be nearly Gaussian because the shapes of the binomial and Gaussian distributions are nearly alike. In fact, if we are applying the smoothing because the original shape is not Gaussian enough, the averaging may make the shape more nearly Gaussian. If we apply binomial smoothing to a distribution that is not Gaussian, we should be aware that we are distorting the shape of the peak and making it more Gaussian.

If the width of the original Gaussian is not too small ($\sigma > 1$), the increase of Equation (A.59) should not be drastic because the addition is in quadrature. For a width $\sigma = 2$, for example, the new width $\sigma' = 2$ is only 5% larger. If the original width is very small ($\sigma < 1$), the approximation of Equation (A.53) is not valid because the Gaussian and binomial distributions are only similar in the limits of large n. A Gaussian fit to the data without smoothing would not be valid either, however, because the parameters of the fit are only meaningful if $\sigma \geq 1$. Because the averaging itself is a binomial distribution, the result is still expected to be a better approximation to a Gaussian distribution than the original data. For a smoothing over three channels, a Gaussian fit requires $\sigma \geq \sqrt{1/2}$ for the original data.

APPENDIX B

MATRICES

B.1 DETERMINANTS

In applying the method of least squares to both linear and nonlinear functions, we required the solution of a set of n simultaneous equations in n unknowns a_i similar to the following:

$$\begin{aligned} y_1 &= a_1X_{11} + a_2X_{12} + a_3X_{13} \\ y_2 &= a_1X_{21} + a_2X_{22} + a_3X_{23} \\ y_3 &= a_1X_{31} + a_2X_{32} + a_3X_{33} \end{aligned} \tag{B.1}$$

where the constants y_i and X_{ij} are known quantities calculated from the data.

The symmetry of the right-hand side suggests that we write elements of the equations in a two-dimensional array

$$\alpha = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} \tag{B.2}$$

and separate the other terms and coefficients into one-dimensional arrays.

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\beta} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{B.3}$$

Such arrays are called matrices, and we can write Equations (B.1) in matrix form as

$$\boldsymbol{\beta} = \boldsymbol{\alpha} \cdot \mathbf{a} \tag{B.4}$$

Alternatively, because in our problems the matrix α is always symmetric, that is, the element α_{ij} is equal to the element α_{ji}, we can write the matrices a and β as row matrices

$$a = [a_1 \quad a_2 \quad a_3] \quad \text{and} \quad b = [y_1 \quad y_2 \quad y_3] \tag{B.5}$$

and express Equation (B.1) as

$$\boldsymbol{\beta} = \mathbf{a} \cdot \boldsymbol{\alpha} \tag{B.6}$$

We shall be concerned primarily with linear one-dimensional matrices and with symmetric square two-dimensional matrices that have the same number of rows and columns and are mirror-symmetric about the diagonal. Consider a square matrix A:

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2k} & \cdots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{j1} & A_{j2} & \cdots & A_{jk} & \cdots & A_{jn} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nk} & \cdots & A_{nn} \end{bmatrix} \tag{B.7}$$

The *degree* of the matrix A is the number n of rows and columns; the jkth *element* (or *component*) of the matrix is A_{jk}; the *diagonal terms* are A_{jj}. If the matrix is diagonally *symmetric*, $A_{jk} = A_{kj}$ and there are n^2 elements but only $n(n + 1)/2$ different elements.

Matrix Algebra

If A and B are two square symmetric matrices of degree n, then their sum S is a square symmetric matrix of degree n with elements that are the sums of the corresponding elements of the two matrices

$$\mathbf{A} + \mathbf{B} = \mathbf{S} \qquad S_{jk} = A_{jk} + B_{jk} \tag{B.8}$$

The product P of the matrices A and B is a square matrix of degree n, with elements determined in the following way:

$$\mathbf{AB} = \mathbf{P} \qquad P_{jk} = \sum_{m=1}^{n} (A_{jm} B_{mk}) \tag{B.9}$$

The elements of the jth row of A are multiplied by the elements of the kth column of B and the products are summed to obtain the jkth element of P. In general, the matrix P will not be symmetric.

If a is a linear one-dimensional matrix, the product of A and a is only well defined if the product is taken in a particular order. If a is a column matrix, it must be multiplied on the left by the square matrix to yield another column matrix c:

$$\begin{bmatrix} A_{11} & \cdots & & \cdots & A_{1n} \\ \vdots & \cdots & & \vdots & \vdots \\ A_{j1} & \cdots & A_{jk} & \cdots & A_{jn} \\ \vdots & \cdots & & \vdots & \vdots \\ A_{n1} & \cdots & & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_k \\ \vdots \\ c_n \end{bmatrix} \quad c_j = \sum_{k=1}^{n} (A_{jk} a_k) \tag{B.10}$$

If a is a row matrix, it must multiply the square matrix on the left to yield another row matrix r.

$$\begin{bmatrix} a_1 & \cdots & a_j & \cdots & a_n \end{bmatrix} \begin{bmatrix} A_{11} & \cdots & & \cdots & A_{1n} \\ \vdots & \cdots & & \cdots & \vdots \\ A_{j1} & \cdots & A_{jk} & \cdots & A_{jn} \\ \vdots & \cdots & & \cdots & \vdots \\ A_{n1} & \cdots & & \cdots & A_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} r_1 & \cdots & r_k & \cdots & r_n \end{bmatrix} \qquad r_j = \sum_{j=1}^{n} (a_j A_{jk}) \tag{B.11}$$

The product of two linear matrices depends on the order of multiplication. The product of a row matrix a times a column matrix b is a scalar. If the order is reversed, the result is a square matrix that is *diagonal;* that is, for which only the diagonal terms are nonzero:

$$\begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \sum_{j=1}^{n} (a_j b_j)$$

$$\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} = \begin{bmatrix} a_1 b_1 & \cdots & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & \cdots & a_j b_j & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & \cdots & a_n b_n \end{bmatrix} \tag{B.12}$$

Determinants

The *determinant* of a square matrix is defined in terms of its algebra. The *order* of the determinant of a square matrix is equal to the degree n of the matrix. In this section, we shall mainly use determinants of order 3 as examples, although, unless otherwise specified, the comments apply to matrices of all orders. Manipulation of the rows may be substituted for columns throughout.

1. The determinant of the *unity matrix* is 1 where the unity matrix is defined as the diagonal matrix with all diagonal elements equal to 1:

$$|1| = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = 1 \tag{B.13}$$

2. If a column matrix of degree n is added to one column of a square matrix of degree n, the determinant of the result is the sum of the determinant of the original square matrix plus that of another square matrix obtained by substituting the column matrix for the modified column:

$$\begin{vmatrix} A_{11} + a_1 & A_{12} & A_{13} \\ A_{21} + a_2 & A_{22} & A_{23} \\ A_{31} + a_3 & A_{32} & A_{33} \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} + \begin{vmatrix} a_1 & A_{12} & A_{13} \\ a_2 & A_{22} & A_{23} \\ a_3 & A_{32} & A_{33} \end{vmatrix} \tag{B.14}$$

3. If one column of a square matrix is multiplied by a scalar, the determinant of the result is the product of the scalar and the determinant of the original matrix:

$$\begin{vmatrix} cA_{11} & A_{12} & A_{13} \\ cA_{21} & A_{22} & A_{23} \\ cA_{31} & A_{32} & A_{33} \end{vmatrix} = c \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} \tag{B.15}$$

4. If two columns of a square matrix are interchanged, the determinant retains the same magnitude but changes sign:

$$\begin{vmatrix} A_{12} & A_{11} & A_{13} \\ A_{22} & A_{21} & A_{23} \\ A_{32} & A_{31} & A_{33} \end{vmatrix} = - \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} \tag{B.16}$$

5. The *minor* A^{jk} of an element A_{jk} of a square matrix of degree n is defined as the determinant of the square matrix of degree $n - 1$ formed by removing the jth row and the kth column:

$$\mathbf{A} = \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} \qquad A^{21} = \begin{vmatrix} A_{12} & A_{13} \\ A_{32} & A_{33} \end{vmatrix} \tag{B.17}$$

6. The *cofactor* $\text{cof}(A_{jk})$ of an element A_{jk} of a square matrix of degree n is defined as the product of the minor and a phase factor:

$$\text{cof}(A_{jk}) \equiv (-1)^{j+k} A^{jk} \tag{B.18}$$

7. With the preceding definitions 5 and 6, the determinant of a square matrix of degree n can be expressed in terms of cofactors of minors:

$$|\mathbf{A}| = \sum_{k=1}^{n} [A_{jk} \text{cof}(A_{jk})] = \sum_{k=1}^{n} [(-1)^{j+k} A_{jk} A^{jk}] \tag{B.19}$$

Equation (B.19) is an iterative definition, because the cofactor is itself a determinant. The determinant of a matrix of degree 1, however, is equal to the single element of that matrix. The determinant of a square matrix of degree 2 is encountered often enough to make its explicit formula useful:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc \tag{B.20}$$

and we can evaluate the determinant of a third-order matrix with the help of Equation (B.19):

$$\begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} = A_{11}\begin{vmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{vmatrix} - A_{12}\begin{vmatrix} A_{21} & A_{23} \\ A_{31} & A_{33} \end{vmatrix} + A_{13}\begin{vmatrix} A_{21} & A_{22} \\ A_{31} & A_{32} \end{vmatrix} \tag{B.21}$$

Computation

Matrix computation is generally simpler if we can manipulate matrices into diagonal form in which only the diagonal elements A_{jj} are nonzero. The determinant of a diagonal matrix is equal to the product of all the diagonal elements and the *trace* is their sum:

$$|\mathbf{A}_{\text{diag}}| = \prod_{j=1}^{n} A_{jj} \tag{B.22}$$

If we combine rules 2, 3, and 4 of the algebra for determinants, we can show that the determinant of a matrix is unchanged if the elements of any column, multiplied by an arbitrary scalar, are added to the elements of any other column. The determinant of the sum is equal to the sum of the two determinants, but one of these determinants has two identical columns except for a scalar factor that may be extracted, and is therefore equal to 0:

$$\begin{aligned} \begin{vmatrix} A_{11}+cA_{12} & A_{12} & A_{13} \\ A_{21}+cA_{22} & A_{22} & A_{23} \\ A_{31}+cA_{32} & A_{32} & A_{33} \end{vmatrix} &= \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} + c\begin{vmatrix} A_{12} & A_{12} & A_{13} \\ A_{22} & A_{22} & A_{23} \\ A_{32} & A_{32} & A_{33} \end{vmatrix} \\ &= |\mathbf{A}| \end{aligned} \tag{B.23}$$

Thus, it is possible to eliminate all elements except one from a row by successively subtracting one column, appropriately scaled, from each of the others. For example, if we perform the subtraction

$$A'_{jk} = A_{jk} - A_{1j}\frac{A_{j1}}{A_{11}} \tag{B.24}$$

on each row except the first, we eliminate all elements of the first column except A_{11} to obtain

$$\mathbf{A}' = \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A'_{22} & A'_{23} \\ 0 & A'_{32} & A'_{33} \end{vmatrix} \tag{B.25}$$

Similarly, if we subsequently start with element A'_{22} and subtract an appropriately scaled second row from the rest of the rows,

$$A''_{jk} = A'_{jk} - A'_{2k}A'_{jk}/A'_{22} \tag{B.26}$$

all the elements of the second column vanish except A'_{22}:

$$\mathbf{A}'' = \begin{vmatrix} A_{11} & 0 & A''_{13} \\ 0 & A'_{22} & A''_{23} \\ 0 & 0 & A''_{33} \end{vmatrix} \tag{B.27}$$

Note that A'_{22} is not the original value A_{22}, but is modified as a result of the first subtraction.

By successively subtracting rows (or columns) scaled to their diagonal elements, we can produce a matrix that is diagonal. In practice, it is sufficient to eliminate only half of the nondiagonal elements so that all elements on one side of a diagonal are 0:

$$\begin{aligned} \mathbf{A} &= \begin{vmatrix} A_{11} & 0 & 0 \\ A_{21} & A_{22} & 0 \\ A_{31} & A_{32} & A_{33} \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & A_{33} \end{vmatrix} = \begin{vmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & 0 \\ 0 & 0 & A_{33} \end{vmatrix} \\ &= A_{11}A_{22}A_{33} \end{aligned} \tag{B.28}$$

B.2 SOLUTION OF SIMULTANEOUS EQUATIONS BY DETERMINANTS

Consider the following set of three equations in three coefficients a_1, a_2, and a_3. We shall consider the y_k and X_{jk} to be known quantities; that is, constants:

$$\begin{aligned} y_1 &= a_1X_{11} + a_2X_{12} + a_3X_{13} \\ y_2 &= a_1X_{21} + a_2X_{22} + a_3X_{23} \\ y_3 &= a_1X_{31} + a_2X_{32} + a_3X_{33} \end{aligned} \tag{B.29}$$

Let us consider the set of equations as if they were one matrix equation as in Equation (B.10):

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{B.30}$$

with a and y represented by linear matrices and X represented by a square matrix. If we multiply the first equation of Equations (B.29) by the cofactor of X_{11} in the matrix of Equation (B.30), multiply the second equation by the cofactor of X_{21}, and multiply the third by the cofactor of X_{31}, then the sum of the three equations is an equation involving determinants according to Equation (B.18):

$$\begin{aligned} \begin{vmatrix} y_1 & X_{12} & X_{13} \\ y_2 & X_{22} & X_{23} \\ y_3 & X_{32} & X_{33} \end{vmatrix} &= a_1\begin{vmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{vmatrix} + a_2\begin{vmatrix} X_{12} & X_{12} & X_{13} \\ X_{22} & X_{22} & X_{23} \\ X_{32} & X_{32} & X_{33} \end{vmatrix} \\ &\quad + a_3\begin{vmatrix} X_{13} & X_{12} & X_{13} \\ X_{23} & X_{22} & X_{23} \\ X_{33} & X_{32} & X_{33} \end{vmatrix} \end{aligned} \tag{B.31}$$

The determinants in the two rightmost terms of Equation (B.31) both vanish because they have two columns that are identical. Thus, the solution for the coefficient a_1 is the ratio of the two determinants:

$$a_1 = \frac{\begin{vmatrix} y_1 & X_{12} & X_{13} \\ y_2 & X_{22} & X_{23} \\ y_3 & X_{32} & X_{33} \end{vmatrix}}{\begin{vmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{vmatrix}} \tag{B.32}$$

The denominator is the determinant of the square matrix X of Equation (B.30) and the numerator is the determinant of a matrix that is formed by substituting the column matrix y for the first column of the X matrix.

Similarly, *Cramér's rule* gives the solution for the jth coefficient a_j of a set of n simultaneous equations as the ratio of two determinants:

$$y_k = \sum_{j=1}^{n} (a_j X_{kj}) \qquad k = 1, n$$

$$a_j = \frac{|\mathbf{X}'(j)|}{|\mathbf{X}|} \tag{B.33}$$

The denominator is the determinant of the X matrix. The numerator $|\mathbf{X}'(j)|$ is the determinant of the matrix formed by substituting the y matrix for the jth column.

A matrix is singular if its determinant is 0. If the X matrix is singular, there is no solution for Equation (B.33). For example, if two of the n simultaneous equations are identical, except for a scale factor, there are really only $n - 1$ independent simultaneous equations, and therefore no solution for the n unknowns. In this case, the X matrix has two identical rows and therefore a 0 determinant.

Solution by Matrix Equations

Let us consider Equation (B.33) as if it were a matrix equation as in Equation (B.30). If the X matrix is square, we can consider the y and a linear matrices as either column matrices as in Equation (B.10) or row matrices as in Equation (B.11):

$$[y_k] = [a_j][X_{kj}] \tag{B.34}$$

If we could multiply this matrix by another matrix $\mathbf{X}'$ such that the right-hand side becomes just the linear matrix a, then we will have our solution for the coefficients a_j directly. The multiplication of matrices is associative; that is,

$$\mathbf{A(BC) = (AB)C} \tag{B.35}$$

Therefore, we require a matrix $\mathbf{X}'$ such that if it is multiplied by the matrix X, the result is the unity matrix:

$$[X_{kj}][X'_{kj}] = 1 \tag{B.36}$$

The matrix $\mathbf{X}'$ that satisfies Equation (B.36) is called the inverse matrix $\mathbf{X}^{-1}$ of X. Equation (B.34) multiplied from the right by $\mathbf{X}^{-1}$ gives the coefficients a_j explicitly, because any matrix is unchanged when multiplied by the unity matrix:

$$[y_k][X_{jk}]^{-1} = [a_j]1 = [a_j] \tag{B.37}$$

We can express Equation (B.37) in more conventional form to give the solution for each of the coefficients a_j:

$$a_j = \sum_{k=1}^{n} (y_k X_{kj}^{-1}) \tag{B.38}$$

Thus, the solution for the n unknowns with n simultaneous equations is reduced to evaluating the elements of the inverse matrix $\mathbf{X}^{-1}$.

B.3 MATRIX INVERSION

The adjoint $\mathbf{A}^\dagger$ of a matrix A is defined as the matrix obtained by substituting for each element A_{jk} the cofactor of the transposed element A_{kj}:

$$A_{jk}^\dagger = \text{cof}(A_{kj}) \tag{B.39}$$

For a square symmetric matrix, the transposition makes no difference.

The inverse matrix $\mathbf{A}^{-1}$ defined in Equation (B.36) may be evaluated by dividing the adjoint matrix $\mathbf{A}^\dagger$ by the determinant of A:

$$A_{jk}^{-1} = \frac{A_{jk}^\dagger}{|\mathbf{A}|} \tag{B.40}$$

To show that this equality holds, we multiply both sides of Equation (B.40) by $|\mathbf{A}|\mathbf{A}$.

$$|\mathbf{A}|\mathbf{A}\mathbf{A}^{-1} = |\mathbf{A}|\mathbf{1} = \mathbf{A}\mathbf{A}^\dagger \tag{B.41}$$

Diagonal terms of the matrices in Equation (B.41) are equivalent to the formula of Equation (B.19) for evaluating the determinant:

$$|\mathbf{A}| = \sum_{k=1}^{n} (A_{jk} A_{jk}^\dagger) = \sum_{k=1}^{n} [A_{jk}\text{cof}(A_{jk})] \tag{B.42}$$

Off-diagonal elements can be shown to vanish like those of the determinants of Equation (B.31). If the matrix A is singular (that is, if $|\mathbf{A}| = 0$), the inverse matrix $\mathbf{A}^{-1}$ does not exist and there is no solution to the matrix equation of Equation (B.34).

Gauss-Jordan Elimination

The formula of Equation (B.40) is generally too cumbersome for use in computing the inverse of a matrix. Instead, the Gauss-Jordan method of elimination is used to invert a matrix by building up the inverse matrix from a unity matrix while reducing the original matrix to unity.

Consider the inverse matrix $\mathbf{A}^{-1}$ as the ratio of the unity matrix divided by the original matrix, $\mathbf{A}^{-1} = \mathbf{1/A}$. If we manipulate the numerator and denominator of this ratio in the same manner (multiplying rows or columns by the same constant factor

and adding the same rows scaled to the same constants), the ratio remains unchanged. If we perform the proper manipulation, we can change the denominator into the unity matrix; the numerator must then become equal to the inverse matrix $\mathbf{A}^{-1}$.

Let us write the 3×3 matrix A and the 3×3 unity matrix side by side and manipulate both to reduce the matrix A to the unity matrix. We start by using the formula of Equation (B.24) to eliminate the two off-diagonal elements of the first column:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} - A_{12}\dfrac{A_{21}}{A_{11}} & A_{23} - A_{13}\dfrac{A_{21}}{A_{11}} \\ 0 & A_{32} - A_{12}\dfrac{A_{31}}{A_{11}} & A_{33} - A_{13}\dfrac{A_{31}}{A_{11}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -\dfrac{A_{21}}{A_{11}} & 1 & 0 \\ -\dfrac{A_{21}}{A_{11}} & 0 & 1 \end{bmatrix} \tag{B.43}$$

Now, we divide the first row by A_{11} to get a diagonal element of

$$\begin{bmatrix} 1 & \dfrac{A_{12}}{A_{11}} & \dfrac{A_{13}}{A_{11}} \\ 0 & A_{22} - A_{12}\dfrac{A_{21}}{A_{11}} & A_{23} - A_{13}\dfrac{A_{21}}{A_{11}} \\ 0 & A_{32} - A_{12}\dfrac{A_{31}}{A_{11}} & A_{33} - A_{13}\dfrac{A_{31}}{A_{11}} \end{bmatrix} \begin{bmatrix} \dfrac{1}{A_{11}} & 0 & 0 \\ -\dfrac{A_{21}}{A_{11}} & 1 & 0 \\ -\dfrac{A_{21}}{A_{11}} & 0 & 1 \end{bmatrix} \tag{B.44}$$

The left matrix now has the proper first column. Let us relabel the matrices B (on the left) and B′ (on the right) and perform the corresponding manipulations to obtain zeros in place of B_{12} and B_{32}, and then divide the second row by B_{22}:

$$\begin{bmatrix} 1 & 0 & B_{13} - B_{23}\dfrac{B_{12}}{B_{22}} \\ 0 & 1 & \dfrac{B_{23}}{B_{22}} \\ 0 & 0 & B_{33} - B_{23}\dfrac{B_{32}}{B_{22}} \end{bmatrix} \begin{bmatrix} B'_{11} - B'_{21}\dfrac{B_{12}}{B_{22}} & -\dfrac{B_{12}}{B_{22}} & 0 \\ \dfrac{B'_{21}}{B_{22}} & \dfrac{1}{B_{22}} & 0 \\ B'_{31} - B'_{21}\dfrac{B_{32}}{B_{22}} & -\dfrac{B_{32}}{B_{22}} & 1 \end{bmatrix} \tag{B.45}$$

After similar manipulation of the third column, the matrix on the left becomes the unity matrix and that on the right, therefore, must be the inverse matrix.

For computational purposes, even this method is somewhat inefficient in that two matrices must be manipulated throughout. Note, however, that at each stage of the reduction, there are only n (or three) useful columns of information in the two matrices. As each column is eliminated from the left matrix, the corresponding column is accumulated on the right.

Therefore, we can combine the manipulation into the range of a single matrix. We start with the matrix A and use the formula of Equation (B.24) as for Equation (B.43), but instead of applying this formula to the first column, we divide the first column by $-A_{11}$ to get the first column on the right of Equation (B.43); the diagonal element must be divided twice to become $1/A_{11}$. Divide the rest of the first row by A_{11} to get the composite of the two matrices of Equation (B.44):

$$\begin{bmatrix} \dfrac{1}{A_{11}} & \dfrac{A_{12}}{A_{11}} & \dfrac{A_{13}}{A_{11}} \\ -\dfrac{A_{21}}{A_{11}} & A_{22} - A_{12}\dfrac{A_{21}}{A_{13}} & A_{23} - A_{13}\dfrac{A_{21}}{A_{11}} \\ -\dfrac{A_{21}}{A_{11}} & A_{32} - A_{12}\dfrac{A_{31}}{A_{11}} & A_{33} - A_{13}\dfrac{A_{31}}{A_{11}} \end{bmatrix} \tag{B.46}$$

A corresponding manipulation of the second column yields a matrix with the first two columns identical to those of the right side of Equation (B.45) whereas the last column is identical to that of the left side of Equation (B.45). Thus the inverse matrix is accumulated in the space vacated by the original matrix.

Computer Routine PROGRAM B.1 MATRIX (WEBSITE) includes two routines, MATINV and LINEARBYSQUARE. MATINV inverts a square matrix and calculates its determinant, substituting the inverted matrix into the same array as the original matrix.[1] Input variables are ARRAY, the matrix to be inverted, and NORDER, the order of its determinant.

The initial program loop iterates through the n columns of the matrix, reorganizing the matrix to get the largest element in the diagonal in order to reduce rounding errors and improve computational precision. The inversion procedure discussed above is then carried out and the determinant DET of the matrix is calculated from the diagonalized matrix. After inversion, the inverted matrix is stored back in ARRAY and the variable DET, the value of the determinant of the original matrix, is returned.

LINEARBYSQUARE multiplies a linear matrix (on the right) by a square matrix (on the left). For example, see Equation (B.30).

[1]The subroutine MATINV follows the procedure of the subroutine MINV of the IBM System/360 Scientific Subroutine Package.

APPENDIX C

GRAPHS AND TABLES

The tables and graphs in this appendix are provided for easy reference. Computer routines for calculating several of the distributions and probability functions are listed in Appendix E. Routines are also available on the website for calculating probabilities.

C.1 GAUSSIAN PROBABILITY DISTRIBUTION

The probability density function $p_G(x; \mu, \sigma)$ for the Gaussian or normal error distribution is given by

$$p_G(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

If measurements of a quantity x are distributed in this manner around a mean μ with standard deviation σ, the probability $dP_G(x; \mu, \sigma)$ for observing a value of x, within an infinitesimally small interval dx, in a random sample measurement is given by

$$dP_G(x; \mu, \sigma) = p_G(x; \mu, \sigma)\, dx$$

Values of the probability density function $p_G(x; \mu, \sigma)$ are tabulated in Table C.1 as a function of the dimensionless deviation

$$z = |x - \mu|/\sigma$$

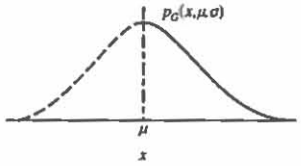

TABLE C.1
Gaussian probability density distribution. The Gaussian or normal error distribution $p_G(x; \mu, \sigma)$ versus $z = |x - \mu|/\sigma$

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.39894	0.39892	0.39886	0.39876	0.39862	0.39844	0.39822	0.39797	0.39767	0.39733
0.1	0.39695	0.39654	0.39608	0.39559	0.39505	0.39448	0.39387	0.39322	0.39253	0.39181
0.2	0.39104	0.39024	0.38940	0.38853	0.38762	0.38667	0.38568	0.38466	0.38361	0.38251
0.3	0.38139	0.38023	0.37903	0.37780	0.37654	0.37524	0.37391	0.37255	0.37115	0.36973
0.4	0.36827	0.36678	0.36526	0.36371	0.36213	0.36053	0.35889	0.35723	0.35553	0.35381
0.5	0.35207	0.35029	0.34849	0.34667	0.34482	0.34294	0.34105	0.33912	0.33718	0.33521
0.6	0.33322	0.33121	0.32918	0.32713	0.32506	0.32297	0.32086	0.31874	0.31659	0.31443
0.7	0.31225	0.31006	0.30785	0.30563	0.30339	0.30114	0.29887	0.29659	0.29431	0.29200
0.8	0.28969	0.28737	0.28504	0.28269	0.28034	0.27799	0.27562	0.27324	0.27086	0.26848
0.9	0.26609	0.26369	0.26129	0.25888	0.25647	0.25406	0.25164	0.24923	0.24681	0.24439
1.0	0.24197	0.23995	0.23713	0.23471	0.23230	0.22988	0.22747	0.22506	0.22266	0.22025
1.1	0.21785	0.21546	0.21307	0.21069	0.20831	0.20594	0.20357	0.20122	0.19887	0.19652
1.2	0.19419	0.19186	0.18955	0.18724	0.18494	0.18265	0.18038	0.17811	0.17585	0.17361
1.3	0.17137	0.16915	0.16694	0.16475	0.16256	0.16039	0.15823	0.15609	0.15395	0.15184
1.4	0.14973	0.14764	0.14557	0.14351	0.14147	0.13944	0.13742	0.13543	0.13344	0.13148
1.5	0.12952	0.12759	0.12567	0.12377	0.12189	0.12002	0.11816	0.11633	0.11451	0.11271
1.6	0.11093	0.10916	0.10741	0.10568	0.10397	0.10227	0.10059	0.09893	0.09729	0.09567
1.7	0.09406	0.09247	0.09090	0.08934	0.08780	0.08629	0.08478	0.08330	0.08184	0.08039
1.8	0.07896	0.07755	0.07615	0.07477	0.07342	0.07207	0.07075	0.06944	0.06815	0.06688
1.9	0.06562	0.06439	0.06316	0.06196	0.06077	0.05960	0.05845	0.05731	0.05619	0.05509
2.0	0.05400	0.05293	0.05187	0.05083	0.04981	0.04880	0.04781	0.04683	0.04587	0.04492
2.1	0.04399	0.04307	0.04217	0.04129	0.04041	0.03956	0.03871	0.03788	0.03707	0.03627
2.2	0.03548	0.03471	0.03395	0.03320	0.03247	0.03175	0.03104	0.03034	0.02966	0.02899
2.3	0.02833	0.02769	0.02705	0.02643	0.02582	0.02522	0.02464	0.02406	0.02350	0.02294
2.4	0.02240	0.02187	0.02135	0.02083	0.02033	0.01984	0.01936	0.01889	0.01843	0.01798
2.5	0.01753	0.01710	0.01667	0.01626	0.01585	0.01545	0.01506	0.01468	0.01431	0.01394
2.6	0.01359	0.01324	0.01290	0.01256	0.01224	0.01192	0.01160	0.01130	0.01100	0.01071
2.7	0.01042	0.01015	0.00987	0.00961	0.00935	0.00910	0.00885	0.00861	0.00837	0.00814
2.8	0.00792	0.00770	0.00749	0.00728	0.00707	0.00688	0.00668	0.00649	0.00631	0.00613
2.9	0.00595	0.00578	0.00562	0.00546	0.00530	0.00514	0.00500	0.00485	0.00471	0.00457

	0.00	0.10	0.20	0.30	0.40
3.0	0.0044318	0.0032668	0.0023841	0.0017226	0.0012322
3.5	0.00087269	0.00061191	0.00042479	0.00029195	0.00019866
4.0	0.00013383	0.000089264	0.000058945	0.000038536	0.000024943
4.5	0.000015984	0.000010141	0.0000063701	0.0000039615	0.0000024391
5.0	0.0000014868	0.00000089730	0.00000053614	0.00000031716	0.00000018575
5.5	0.00000010771	0.00000006183	0.00000003514	0.00000001978	0.00000001102

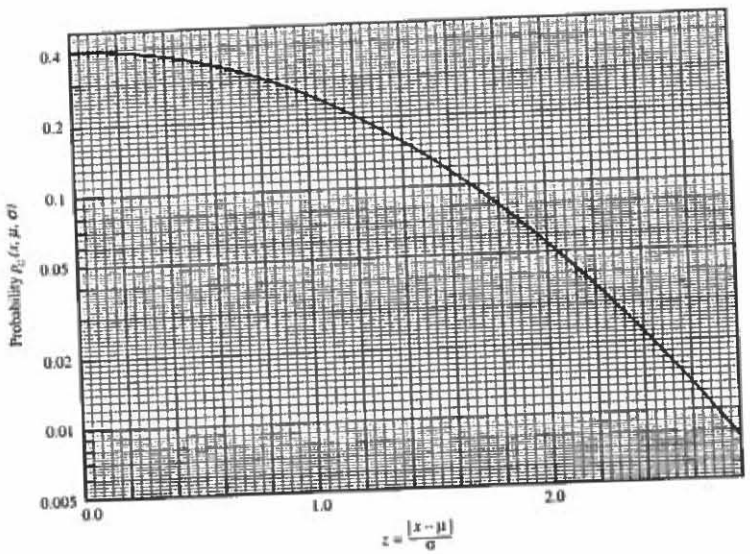

FIGURE C.1
Gaussian probability density distribution, $p_G(x; \mu, \sigma)$ versus $z = |x - \mu|/\sigma$

for z ranging from 0.0 to 3.0 in increments of 0.01 and up to 5.9 in increments of 0.1. This function is graphed on a semi-logarithmic scale as a function of z in Figure C.1.

C.2 INTEGRAL OF GAUSSIAN DISTRIBUTION

The integral $P_G(x; \mu, \sigma)$ of the probability density function $p_G(x; \mu, \sigma)$ for the Gaussian or normal error distribution is given by

$$P_G(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_{\mu-z\sigma}^{\mu+z\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] dx$$

with

$$z = \frac{|x-\mu|}{\sigma}$$

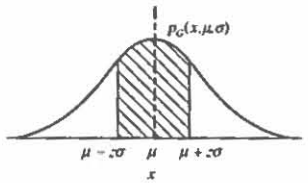

TABLE C.2
Integral of Gaussian distribution. The integral of the Gaussian probability density distribution. $P_G(x; \mu, \sigma)$ versus $z = |x - \mu|/\sigma$

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.0	0.00798	0.01596	0.02393	0.03191	0.03988	0.04784	0.05581	0.06376	0.07171
0.1	0.07966	0.08759	0.09552	0.10343	0.11134	0.11924	0.12712	0.13499	0.14285	0.15069
0.2	0.15852	0.16633	0.17413	0.18191	0.18967	0.19741	0.20514	0.21284	0.22052	0.22818
0.3	0.23582	0.24344	0.25103	0.25860	0.26614	0.27366	0.28115	0.28862	0.29605	0.30346
0.4	0.31084	0.31819	0.32551	0.33280	0.34006	0.34729	0.35448	0.36164	0.36877	0.37587
0.5	0.38292	0.38995	0.39694	0.40389	0.41080	0.41768	0.42452	0.43132	0.43809	0.44481
0.6	0.45149	0.45814	0.46474	0.47131	0.47783	0.48431	0.49075	0.49714	0.50350	0.50981
0.7	0.51607	0.52230	0.52847	0.53461	0.54070	0.54674	0.55274	0.55870	0.56461	0.57047
0.8	0.57629	0.58206	0.58778	0.59346	0.59909	0.60467	0.61021	0.61570	0.62114	0.62653
0.9	0.63188	0.63718	0.64243	0.64763	0.65278	0.65789	0.66294	0.66795	0.67291	0.67783
1.0	0.68269	0.68750	0.69227	0.69699	0.70166	0.70628	0.71085	0.71538	0.71985	0.72428
1.1	0.72866	0.73300	0.73728	0.74152	0.74571	0.74985	0.75395	0.75799	0.76199	0.76595
1.2	0.76985	0.77371	0.77753	0.78130	0.78502	0.78869	0.79232	0.79591	0.79945	0.80294
1.3	0.80639	0.80980	0.81316	0.81647	0.81975	0.82298	0.82616	0.82930	0.83240	0.83546
1.4	0.83848	0.84145	0.84438	0.84727	0.85012	0.85293	0.85570	0.85843	0.86112	0.86377
1.5	0.86638	0.86895	0.87148	0.87397	0.87643	0.87885	0.88123	0.88358	0.88588	0.88816
1.6	0.89039	0.89259	0.89476	0.89689	0.89898	0.90105	0.90308	0.90507	0.90703	0.90896
1.7	0.91086	0.91272	0.91456	0.91636	0.91813	0.91987	0.92158	0.92326	0.92491	0.92654
1.8	0.92813	0.92969	0.93123	0.93274	0.93422	0.93568	0.93711	0.93851	0.93988	0.94123
1.9	0.94256	0.94386	0.94513	0.94638	0.94761	0.94882	0.95000	0.95115	0.95229	0.95340
2.0	0.95449	0.95556	0.95661	0.95764	0.95864	0.95963	0.96059	0.96154	0.96247	0.96338
2.1	0.96426	0.96513	0.96599	0.96682	0.96764	0.96844	0.96922	0.96999	0.97074	0.97147
2.2	0.97219	0.97289	0.97358	0.97425	0.97490	0.97555	0.97617	0.97679	0.97739	0.97797
2.3	0.97855	0.97911	0.97965	0.98019	0.98071	0.98122	0.98172	0.98221	0.98268	0.98315
2.4	0.98360	0.98404	0.98448	0.98490	0.98531	0.98571	0.98610	0.98648	0.98686	0.98722
2.5	0.98758	0.98792	0.98826	0.98859	0.98891	0.98922	0.98953	0.98983	0.99012	0.99040
2.6	0.99067	0.99094	0.99120	0.99146	0.99171	0.99195	0.99218	0.99241	0.99264	0.99285
2.7	0.99306	0.99327	0.99347	0.99366	0.99385	0.99404	0.99422	0.99439	0.99456	0.99473
2.8	0.99489	0.99504	0.99520	0.99534	0.99549	0.99563	0.99576	0.99589	0.99602	0.99615
2.9	0.99627	0.99638	0.99650	0.99661	0.99672	0.99682	0.99692	0.99702	0.99712	0.99721

	0.00	0.10	0.20	0.30	0.40
3.0	0.9973002	0.9980648	0.9986257	0.99903315	0.99932614
3.5	0.99953474	0.99968178	0.99978440	0.99985530	0.999903805
4.0	0.999936656	0.999958684	0.999973308	0.999982920	0.999989174
4.5	0.9999932043	0.9999957748	0.9999973982	0.9999984132	0.9999990449
5.0	0.99999942657	0.99999966024	0.99999980061	0.99999988410	0.99999993327
5.5	0.99999996193	0.99999997847	0.99999998793	0.99999999328	0.99999999627

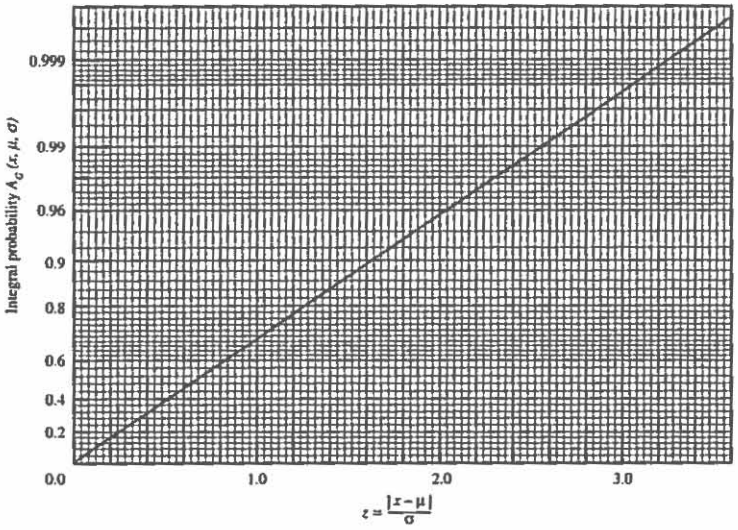

FIGURE C.2
Integral of the Gaussian probability density distribution, $p_G(x; \mu, \sigma)$ versus $z = |x - \mu|/\sigma$

If measurements of the quantity x are distributed according to the Gaussian distribution around a mean μ with standard deviation σ, $P_G(x; \mu, \sigma)$ is equal to the probability for observing a value of x in a random sample measurement that is between $\mu - z\sigma$ and $\mu + z\sigma$; that is, it is the probability that $|x - \mu| < z\sigma$.

Values of the integral $P_G(x; \mu, \sigma)$ are tabulated in Table C.2 as a function of z, for z ranging from 0.0 to 3.0 in increments of 0.01 and up to 5.9 in increments of 0.1. This function is graphed on a probability scale as a function of z in Figure C.2.

A related function is the *error function* erf Z:

$$\text{erf } Z = \frac{1}{\sqrt{\pi}} \int_{-z}^{z} e^{-z^2} dz = P_G(z\sqrt{2}; 0, 1)$$

The function that is tabulated and graphed is the shaded area between the limits $\mu \pm z\sigma$ as indicated.

C.3 LINEAR-CORRELATION COEFFICIENT

The probability distribution $p_r(r, \nu)$ for the linear-correlation coefficient r for ν degrees of freedom is given by

$$p_r(r; \nu) = \frac{1}{\sqrt{\pi}} \frac{\Gamma[(\nu + 1)/2]}{\Gamma(\nu/2)} (1 - r^2)^{(\nu-2)/2}$$

The probability of observing a value of the correlation coefficient larger than r for a random sample of N observations with ν degrees of freedom is the integral of this probability $P_c(r; N)$:

$$P_c(r; N) = \frac{1}{\sqrt{\pi}} \frac{\Gamma[(\nu + 1)/2]}{\Gamma(\nu/2)} \int_{|r|}^{1} (1 - x^2)^{(\nu-2)/2} dx \qquad \nu = N - 2$$

If two variables of a parent population are uncorrelated, the probability that a random sample of N observations will yield a correlation coefficient for those two variables greater in magnitude than $|r|$ is given by $P_c(r; N)$.

Values of the coefficient $|r|$corresponding to various values of the probability $P_c(r; N)$ are tabulated in Table C.3 for N ranging from 3 to 100, and values of $P_c(r; N)$ ranging from 0.001 to 0.5. The functional dependence of r corresponding to representative values of $P_c(r, N)$ is graphed on a semi-logarithmic scale as a smooth variation with the number of observations N in Figure C.3.

The function that is tabulated and graphed is the shaded area under the tails of the probability curve for values larger than $|r|$ as indicated.

C.4 χ^2 DISTRIBUTION

The probability density distribution $p_\chi(\chi^2; \nu)$ for χ^2 is given by

$$p_\chi(\chi^2; \nu) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)} (\chi^2)^{(\nu-2)/2} e^{-\chi^2/2}$$

The probability of observing a value of χ^2 that is larger than a particular value for a random sample of N observations with ν degrees of freedom is the integral of this probability $P_\chi(\chi^2; \nu)$:

$$P_\chi(\chi^2; \nu) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)} \int_{\chi^2}^{\infty} (x^2)^{(\nu-2)/2} e^{-x^2/2} d(x^2)$$

Values of the reduced chi-square $\chi^2_\nu = \chi^2/\nu$ corresponding to various values of the integral probability $P_\chi(\chi^2; \nu)$ of exceeding χ^2 in a measurement with ν degrees of freedom are tabulated in Table C.4 for ν ranging from 1 to 200. The functional dependence of $P_\chi(\chi^2; \nu)$ corresponding to representative values of ν is graphed in Figure C.4 as a smooth variation with the reduced chi-square χ^2_ν.

The function that is tabulated and graphed is the shaded area under the tail of the probability curve for values larger than χ^2 as indicated.

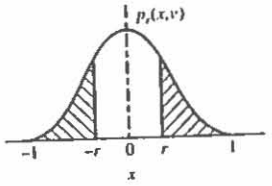

TABLE C.3
Linear-correlation coefficient. The linear-correlation coefficient r versus the number of observations N and the corresponding probability $P_c(r; N)$ of exceeding r in a random sample of observations taken from an uncorrelated parent population ($\rho = 0$)

	P								
N	0.50	0.20	0.10	0.050	0.020	0.010	0.005	0.002	0.001
3	0.707	0.951	0.988	0.997	1.000	1.000	1.000	1.000	1.000
4	0.500	0.800	0.900	0.950	0.980	0.990	0.995	0.998	0.999
5	0.404	0.687	0.805	0.878	0.934	0.959	0.974	0.986	0.991
6	0.347	0.608	0.729	0.811	0.882	0.917	0.942	0.963	0.974
7	0.309	0.551	0.669	0.754	0.833	0.875	0.906	0.935	0.951
8	0.281	0.507	0.621	0.707	0.789	0.834	0.870	0.905	0.925
9	0.260	0.472	0.582	0.666	0.750	0.798	0.836	0.875	0.898
10	0.242	0.443	0.549	0.632	0.715	0.765	0.805	0.847	0.872
11	0.228	0.419	0.521	0.602	0.685	0.735	0.776	0.820	0.847
12	0.216	0.398	0.497	0.576	0.658	0.708	0.750	0.795	0.823
13	0.206	0.380	0.476	0.553	0.634	0.684	0.726	0.772	0.801
14	0.197	0.365	0.458	0.532	0.612	0.661	0.703	0.750	0.780
15	0.189	0.351	0.441	0.514	0.592	0.641	0.683	0.730	0.760
16	0.182	0.338	0.426	0.497	0.574	0.623	0.664	0.711	0.742
17	0.176	0.327	0.412	0.482	0.558	0.606	0.647	0.694	0.725
18	0.170	0.317	0.400	0.468	0.543	0.590	0.631	0.678	0.708
19	0.165	0.308	0.389	0.456	0.529	0.575	0.616	0.662	0.693
20	0.160	0.299	0.378	0.444	0.516	0.561	0.602	0.648	0.679
22	0.152	0.284	0.360	0.423	0.492	0.537	0.576	0.622	0.652
24	0.145	0.271	0.344	0.404	0.472	0.515	0.554	0.599	0.629
26	0.138	0.260	0.330	0.388	0.453	0.496	0.534	0.578	0.607
28	0.133	0.250	0.317	0.374	0.437	0.479	0.515	0.559	0.588
30	0.128	0.241	0.306	0.361	0.423	0.463	0.499	0.541	0.570
32	0.124	0.233	0.296	0.349	0.409	0.449	0.484	0.526	0.554
34	0.120	0.225	0.287	0.339	0.397	0.436	0.470	0.511	0.539
36	0.116	0.219	0.279	0.329	0.386	0.424	0.458	0.498	0.525
38	0.113	0.213	0.271	0.320	0.376	0.413	0.446	0.486	0.513
40	0.110	0.207	0.264	0.312	0.367	0.403	0.435	0.474	0.501
42	0.107	0.202	0.257	0.304	0.358	0.393	0.425	0.463	0.490
44	0.104	0.197	0.251	0.297	0.350	0.384	0.416	0.453	0.479
46	0.102	0.192	0.246	0.291	0.342	0.376	0.407	0.444	0.469
48	0.100	0.188	0.240	0.285	0.335	0.368	0.399	0.435	0.460
50	0.098	0.184	0.235	0.279	0.328	0.361	0.391	0.427	0.451
60	0.089	0.168	0.214	0.254	0.300	0.330	0.358	0.391	0.414
70	0.082	0.155	0.198	0.235	0.278	0.306	0.332	0.363	0.385
80	0.077	0.145	0.185	0.220	0.260	0.286	0.311	0.340	0.361
90	0.072	0.136	0.174	0.207	0.245	0.270	0.293	0.322	0.341
100	0.068	0.129	0.165	0.197	0.232	0.256	0.279	0.305	0.324

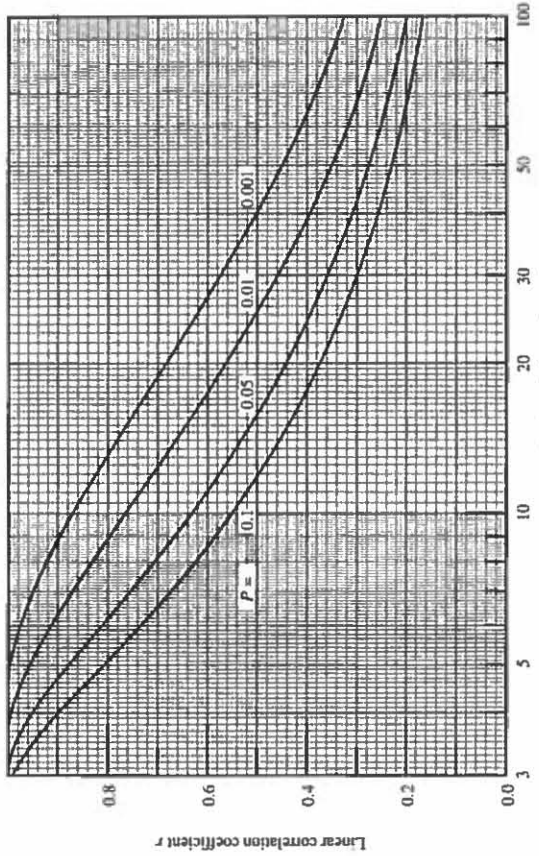

FIGURE C.3
The linear-correlation coefficient r versus the number of observations N and the corresponding probability $P_c(r; N)$ that the variables are not correlated.

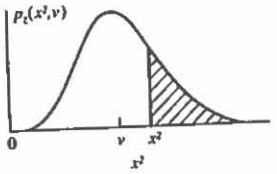

TABLE C.4

χ^2 distribution. Values of the reduced chi-square $\chi^2_\nu = \chi^2/\nu$ corresponding to the probability $P_\chi(\chi^2; \nu)$ of exceeding χ^2 versus the number of degrees of freedom ν

	P							
ν	0.99	0.98	0.95	0.90	0.80	0.70	0.60	0.50
1	0.00016	0.00063	0.00393	0.0158	0.0642	0.148	0.275	0.455
2	0.0100	0.0202	0.0515	0.105	0.223	0.357	0.511	0.693
3	0.0383	0.0617	0.117	0.195	0.335	0.475	0.623	0.789
4	0.0742	0.107	0.178	0.266	0.412	0.549	0.688	0.839
5	0.111	0.150	0.229	0.322	0.469	0.600	0.731	0.870
6	0.145	0.189	0.273	0.367	0.512	0.638	0.762	0.891
7	0.177	0.223	0.310	0.405	0.546	0.667	0.785	0.907
8	0.206	0.254	0.342	0.436	0.574	0.691	0.803	0.918
9	0.232	0.281	0.369	0.463	0.598	0.710	0.817	0.927
10	0.256	0.306	0.394	0.487	0.618	0.727	0.830	0.934
11	0.278	0.328	0.416	0.507	0.635	0.741	0.840	0.940
12	0.298	0.348	0.436	0.525	0.651	0.753	0.848	0.945
13	0.316	0.367	0.453	0.542	0.664	0.764	0.856	0.949
14	0.333	0.383	0.469	0.556	0.676	0.773	0.863	0.953
15	0.349	0.399	0.484	0.570	0.687	0.781	0.869	0.956
16	0.363	0.413	0.498	0.582	0.697	0.789	0.874	0.959
17	0.377	0.427	0.510	0.593	0.706	0.796	0.879	0.961
18	0.390	0.439	0.522	0.604	0.714	0.802	0.883	0.963
19	0.402	0.451	0.532	0.613	0.722	0.808	0.887	0.965
20	0.413	0.462	0.543	0.622	0.729	0.813	0.890	0.967
22	0.434	0.482	0.561	0.638	0.742	0.823	0.897	0.970
24	0.452	0.500	0.577	0.652	0.753	0.831	0.902	0.972
26	0.469	0.516	0.592	0.665	0.762	0.838	0.907	0.974
28	0.484	0.530	0.605	0.676	0.771	0.845	0.911	0.976
30	0.498	0.544	0.616	0.687	0.779	0.850	0.915	0.978
32	0.511	0.556	0.627	0.696	0.786	0.855	0.918	0.979
34	0.523	0.567	0.637	0.704	0.792	0.860	0.921	0.980
36	0.534	0.577	0.646	0.712	0.798	0.864	0.924	0.982
38	0.545	0.587	0.655	0.720	0.804	0.868	0.926	0.983
40	0.554	0.596	0.663	0.726	0.809	0.872	0.928	0.983
42	0.563	0.604	0.670	0.733	0.813	0.875	0.930	0.984
44	0.572	0.612	0.677	0.738	0.818	0.878	0.932	0.985
46	0.580	0.620	0.683	0.744	0.822	0.881	0.934	0.986
48	0.587	0.627	0.690	0.749	0.825	0.884	0.936	0.986
50	0.594	0.633	0.695	0.754	0.829	0.886	0.937	0.987
60	0.625	0.662	0.720	0.774	0.844	0.897	0.944	0.989
70	0.649	0.684	0.739	0.790	0.856	0.905	0.949	0.990
80	0.669	0.703	0.755	0.803	0.865	0.911	0.952	0.992
90	0.686	0.718	0.768	0.814	0.873	0.917	0.955	0.993
100	0.701	0.731	0.779	0.824	0.879	0.921	0.958	0.993
120	0.724	0.753	0.798	0.839	0.890	0.928	0.962	0.994
140	0.743	0.770	0.812	0.850	0.898	0.934	0.965	0.995
160	0.758	0.784	0.823	0.860	0.905	0.938	0.968	0.996
180	0.771	0.796	0.833	0.868	0.910	0.942	0.970	0.996
200	0.782	0.806	0.841	0.874	0.915	0.945	0.972	0.997

TABLE C.4
(continued)

	P							
ν	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.001
1	0.708	1.074	1.642	2.706	3.841	5.412	6.635	10.827
2	0.916	1.204	1.609	2.303	2.996	3.912	4.605	6.908
3	0.982	1.222	1.547	2.084	2.605	3.279	3.780	5.423
4	1.011	1.220	1.497	1.945	2.372	2.917	3.319	4.617
5	1.026	1.213	1.458	1.847	2.214	2.678	3.017	4.102
6	1.035	1.205	1.426	1.774	2.099	2.506	2.802	3.743
7	1.040	1.198	1.400	1.717	2.010	2.375	2.639	3.475
8	1.044	1.191	1.379	1.670	1.938	2.271	2.511	3.266
9	1.046	1.184	1.360	1.632	1.880	2.187	2.407	3.097
10	1.047	1.178	1.344	1.599	1.831	2.116	2.321	2.959
11	1.048	1.173	1.330	1.570	1.789	2.056	2.248	2.842
12	1.049	1.168	1.318	1.546	1.752	2.004	2.185	2.742
13	1.049	1.163	1.307	1.524	1.720	1.959	2.130	2.656
14	1.049	1.159	1.296	1.505	1.692	1.919	2.082	2.580
15	1.049	1.155	1.287	1.487	1.666	1.884	2.039	2.513
16	1.049	1.151	1.279	1.471	1.644	1.852	2.000	2.453
17	1.048	1.148	1.271	1.457	1.623	1.823	1.965	2.399
18	1.048	1.145	1.264	1.444	1.604	1.797	1.934	2.351
19	1.048	1.142	1.258	1.432	1.586	1.773	1.905	2.307
20	1.048	1.139	1.252	1.421	1.571	1.751	1.878	2.266
22	1.047	1.134	1.241	1.401	1.542	1.712	1.831	2.194
24	1.046	1.129	1.231	1.383	1.517	1.678	1.791	2.132
26	1.045	1.125	1.223	1.368	1.496	1.648	1.755	2.079
28	1.045	1.121	1.215	1.354	1.476	1.622	1.724	2.032
30	1.044	1.118	1.208	1.342	1.459	1.599	1.696	1.990
32	1.043	1.115	1.202	1.331	1.444	1.578	1.671	1.953
34	1.042	1.112	1.196	1.321	1.429	1.559	1.649	1.919
36	1.042	1.109	1.191	1.311	1.417	1.541	1.628	1.888
38	1.041	1.106	1.186	1.303	1.405	1.525	1.610	1.861
40	1.041	1.104	1.182	1.295	1.394	1.511	1.592	1.835
42	1.040	1.102	1.178	1.288	1.384	1.497	1.576	1.812
44	1.039	1.100	1.174	1.281	1.375	1.485	1.562	1.790
46	1.039	1.098	1.170	1.275	1.366	1.473	1.548	1.770
48	1.038	1.096	1.167	1.269	1.358	1.462	1.535	1.751
50	1.038	1.094	1.163	1.263	1.350	1.452	1.523	1.733
60	1.036	1.087	1.150	1.240	1.318	1.410	1.473	1.660
70	1.034	1.081	1.139	1.222	1.293	1.377	1.435	1.605
80	1.032	1.076	1.130	1.207	1.273	1.351	1.404	1.560
90	1.031	1.072	1.123	1.195	1.257	1.329	1.379	1.525
100	1.029	1.069	1.117	1.185	1.243	1.311	1.358	1.494
120	1.027	1.063	1.107	1.169	1.221	1.283	1.325	1.446
140	1.026	1.059	1.099	1.156	1.204	1.261	1.299	1.410
160	1.024	1.055	1.093	1.146	1.191	1.243	1.278	1.381
180	1.023	1.052	1.087	1.137	1.179	1.228	1.261	1.358
200	1.022	1.050	1.083	1.130	1.170	1.216	1.247	1.338

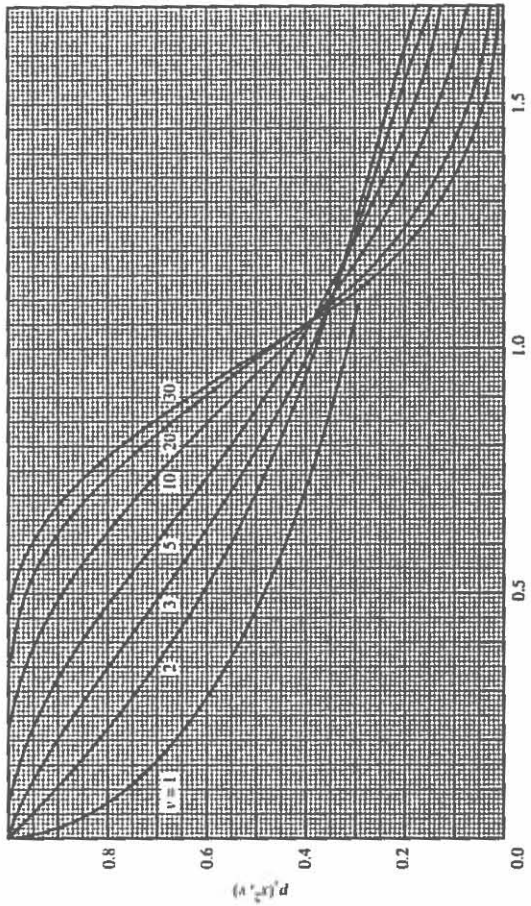

FIGURE C.4
The probability $P_\chi(\chi^2; \nu)$ of exceeding χ^2 versus the reduced chi-square $\chi^2_\nu = \chi^2/\nu$ and the number of degrees of freedom ν.

C.5 *F* DISTRIBUTION

The probability distribution for F is given by

$$p_f(f, \nu_1, \nu_2) = \frac{\Gamma[(\nu_1 + \nu_2)/2]}{\Gamma(\nu_1/2)\Gamma(\nu_2/2)} \left(\frac{\nu_1}{\nu_2}\right)^{\nu_1/2} \frac{f^{(\nu_1-2)/2}}{(1 + f\nu_1/\nu_2)^{1/2(\nu_1+\nu_2)}}$$

The probability of observing a value of F that is larger than a particular value for a random sample with ν_1 and ν_2 degrees of freedom is the integral of this probability:

$$P_F(F; \nu_1, \nu_2) = \int_F^\infty p_f(f; \nu_1, \nu_2)\, df$$

Values of F corresponding to various values of the integral probability $P_F(F; \nu_1, \nu_2)$ of exceeding F in a measurement are tabulated in Table C.5 for $\nu_1 = 1$ and graphed in Figure C.5 as a smooth variation with the probability. Values of F corresponding to various values of ν_1 and ν_2 ranging from 1 to ∞ are listed in Table C.6 and graphed in Figure C.6 for $P_F(F; \nu_1, \nu_2) = 0.05$ and in Table C.7 and Figure C.7 for $P_F(F; \nu_1, \nu_2) = 0.01$. These values were adapted by permission from Dixon and Massey (1969).

The function that is tabulated and graphed is the shaded area under the tail of the probability curve for values larger than F as indicated.

C.6 STUDENT'S *t* DISTRIBUTION

The probability distribution for Student's t is given by[1]

$$f(t, \nu) = \frac{1}{\sqrt{(\nu\pi)}} \frac{\Gamma[(\nu + 1)/2]}{\Gamma(\nu/2)} \left(1 + \frac{t^2}{\nu}\right)^{-(\nu+1)/2}$$

Student's t distribution describes, as a function of the number of degrees of freedom ν, the distribution of the parameter $t = |x - \bar{x}|/s_\mu$, where t is the number of standard deviations s_μ of the sample distribution by which x differs from $\bar{x}$. This distribution takes account of the fact that the sample standard deviation s_μ is an *estimate* of the parent standard error σ_μ and, as such, will vary for different samples drawn from the same parent distribution, just as the sample means vary. If $\bar{x}$ represents the mean of N numbers and x is not derived from the data, then $\nu = N - 1$. If both x and $\bar{x}$ are means, s_μ must be the joint standard deviation of both x and $\bar{x}$, and ν must be the total number of degrees of freedom. In the limit of large numbers of degrees of freedom, Student's t and Gaussian probability distributions agree; for small ν, that is, low-statistics experiments, the Gaussian distribution overestimates the probability and Student's t is preferred.

Table C.8 lists probabilities obtained by integrating Student's t distribution from $x = \bar{x} - ts_\mu$ to $x = \bar{x} + ts_\mu$ where $t = |\bar{x} - x|/s_\mu$. The integrals are listed as functions of t and of the number of degrees of freedom ν. The values corresponding to Gaussian probability (which are independent of ν) are listed in the last column.

[1] "Review of Particle Physics" *The European Physical Journal C*, vol. 15 (2000), p. 193.

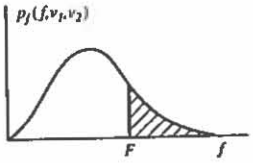

TABLE C.5
F distribution, $\nu = 1$. Values of F corresponding to the probability $P_F(F;1,\nu_2)$ of exceeding F (with $\nu_1 = 1$ degrees of freedom) versus the larger number of degrees of freedom ν_2.

Degrees of freedom ν_2	Probability (P) of exceeding F							
	0.50	0.25	0.10	0.05	0.025	0.01	0.005	0.001
1	1.000	5.83	39.90	161.00	648.00	4050.00	16200.00	406000.0
2	0.667	2.57	8.53	18.50	38.50	98.50	198.00	998.0
3	0.585	2.02	5.54	10.10	17.40	34.10	55.60	167.0
4	0.549	1.81	4.54	7.71	12.20	21.20	31.30	74.1
5	0.528	1.69	4.06	6.61	10.00	16.30	22.80	47.2
6	0.515	1.62	3.78	5.99	8.81	13.70	18.60	35.5
7	0.506	1.57	3.59	5.59	8.07	12.20	16.20	29.2
8	0.499	1.54	3.46	5.32	7.57	11.30	14.70	25.4
9	0.494	1.51	3.36	5.12	7.21	10.60	13.60	22.9
10	0.490	1.49	3.28	4.96	6.94	10.00	12.80	21.0
11	0.486	1.47	3.23	4.84	6.72	9.65	12.20	19.7
12	0.484	1.46	3.18	4.75	6.55	9.33	11.80	18.6
15	0.478	1.43	3.07	4.54	6.20	8.68	10.80	16.6
20	0.472	1.40	2.97	4.35	5.87	8.10	9.94	14.8
24	0.469	1.39	2.93	4.26	5.72	7.82	9.55	14.0
30	0.466	1.38	2.88	4.17	5.57	7.56	9.18	13.3
40	0.463	1.36	2.84	4.08	5.42	7.31	8.83	12.6
60	0.461	1.35	2.79	4.00	5.29	7.08	8.49	12.0
120	0.458	1.34	2.75	3.92	5.15	6.85	8.18	11.4
∞	0.455	1.32	2.71	3.84	5.02	6.63	7.88	10.8

Note: For larger values of the probability P, the value of F is approximately $F = [1.25(1 - P)]^2$.

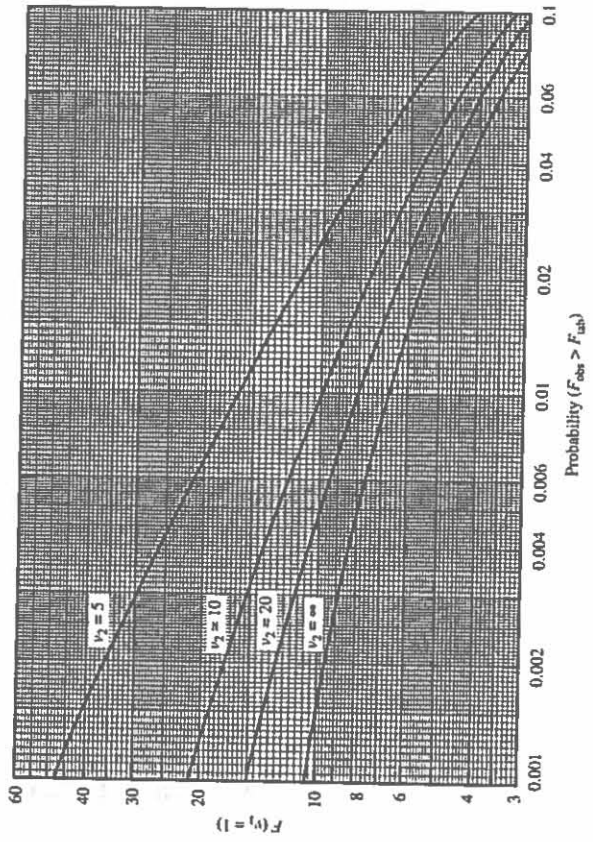

FIGURE C.5
The probability $P_F(F; 1, \nu_2)$ of exceeding F versus F and ν_2 for $\nu_1 = 1$.

TABLE C.6
F distribution, 5%. Values of F corresponding to the probability $P_F(F; \nu_1, \nu_2) = 0.05$ of exceeding F for ν_1 versus ν_2 degrees of freedom

Degrees of freedom ν_2	Degrees of freedom ν_1							
	2	4	6	8	10	15	20	100
1	200.00	225.00	234.00	239.00	242.00	246.00	248.00	253.00
2	19.00	19.20	19.30	19.40	19.40	19.40	19.40	19.50
3	9.55	9.12	8.94	8.85	8.79	8.70	8.66	8.55
4	6.94	6.39	6.16	6.04	5.96	5.86	5.80	5.66
5	5.79	5.19	4.95	4.82	4.73	4.62	4.56	4.41
6	5.14	4.53	4.28	4.15	4.60	3.94	3.87	3.71
7	4.74	4.12	3.87	3.73	3.64	3.51	3.44	3.27
8	4.46	3.84	3.58	3.44	3.35	3.22	3.15	2.97
9	4.26	3.63	3.37	3.23	3.14	3.01	2.94	2.76
10	4.10	3.48	3.22	3.07	2.98	2.85	2.77	2.59
11	3.98	3.36	3.09	2.95	2.85	2.72	2.65	2.46
12	3.89	3.26	3.00	2.85	2.75	2.62	2.54	2.35
15	3.68	3.06	2.79	2.64	2.54	2.40	2.33	2.12
20	3.49	2.87	2.60	2.45	2.35	2.20	2.12	1.91
24	3.40	2.78	2.51	2.36	2.25	2.11	2.03	1.80
30	3.32	2.69	2.42	2.27	2.16	2.01	1.93	1.70
40	3.23	2.61	2.34	2.18	1.08	1.92	1.84	1.59
60	3.15	2.53	2.25	2.10	1.99	1.84	1.75	1.48
120	3.07	2.45	2.18	2.02	1.91	1.75	1.66	1.37
∞	3.00	2.37	2.10	1.94	1.83	1.67	1.57	1.24

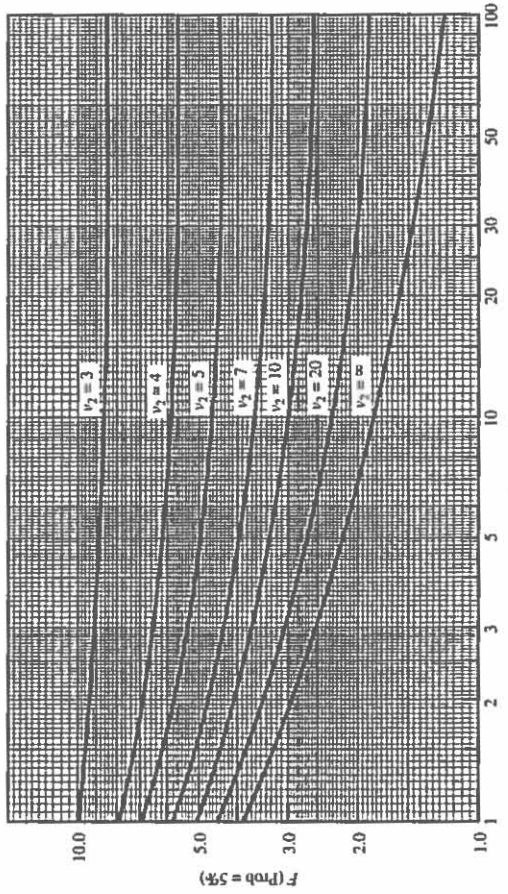

FIGURE C.6
Test values of $F(\nu_1, \nu_2)$ versus the numbers of degrees of freedom ν_1 and ν_2 for a probability $P_F(F; \nu_1, \nu_2) = 0.05$ of exceeding F.

TABLE C.7
F distribution, 1%. Values of F corresponding to the probability $P_F(F; \nu_1, \nu_2) = 0.01$ of exceeding F for ν_1 versus ν_2 degrees of freedom

Degrees of freedom ν_2	Degrees of freedom ν_1							
	2	4	6	8	10	15	20	100
1	5000.00	5620.00	5860.00	5980.00	6060.00	6160.00	6210.00	6330.00
2	99.00	99.20	99.30	99.40	99.40	99.40	99.40	99.50
3	30.80	28.70	27.90	27.50	27.20	26.90	26.70	26.20
4	18.00	16.00	15.20	14.80	14.50	14.20	14.00	13.60
5	13.30	11.40	10.70	10.30	10.10	9.72	9.55	9.13
6	10.90	9.15	8.47	8.10	7.87	7.56	7.40	6.99
7	9.55	7.85	7.19	6.84	6.62	6.31	6.16	5.75
8	8.65	7.01	6.37	6.03	5.81	5.52	5.36	4.96
9	8.02	6.42	5.80	5.47	5.26	4.96	4.81	4.42
10	7.56	5.99	5.39	5.06	4.85	4.56	4.41	4.01
11	7.21	5.67	5.07	4.74	4.54	4.25	4.10	3.71
12	6.93	5.41	4.82	4.50	4.30	4.01	3.86	3.47
15	6.36	4.89	4.32	4.00	3.80	3.52	3.37	2.98
20	5.85	4.43	3.87	3.56	3.37	3.09	2.94	2.54
24	5.61	4.22	3.67	3.36	3.17	2.89	2.74	2.33
30	5.39	4.02	3.47	3.17	2.98	2.70	2.55	2.13
40	5.18	3.83	3.29	2.99	2.80	2.52	2.37	1.94
60	4.98	3.65	3.12	2.82	2.63	2.35	2.20	1.75
120	4.79	3.48	2.96	2.66	2.47	2.19	2.03	1.56
∞	4.61	3.32	2.80	2.51	2.32	2.04	1.88	1.36

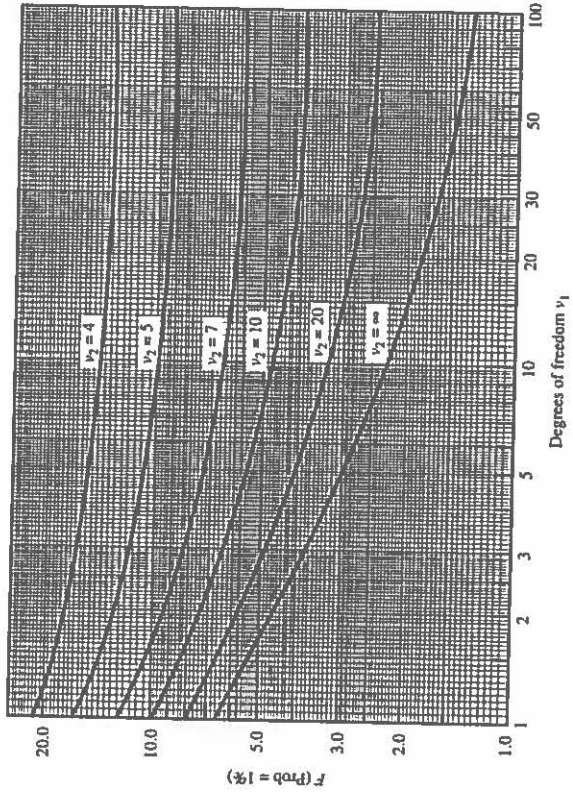

FIGURE C.7
Test values of $F(\nu_1, \nu_2)$ versus the numbers of degrees of freedom ν_1 and ν_2 for a probability $P_F(F; \nu_1, \nu_2) = 0.01$ of exceeding F.

TABLE C.8
$P_T(x; \mu, \sigma)$ versus $t = |x - \mu|/\sigma$; Integral of Student's t distribution between $x = \bar{x} - ts_\mu$ and $\bar{x} + ts_\mu$, expressed in percent.

	$\nu = N - 1$															
t	2	3	4	5	6	8	10	12	16	20	25	30	35	40	50	Gaussian probability
0.6	39.1	40.9	41.9	42.5	43.0	43.5	43.8	44.0	44.3	44.5	44.6	44.7	44.8	44.8	44.9	45.1
0.7	44.4	46.6	47.8	48.5	49.0	49.6	50.0	50.3	50.6	50.8	51.0	51.1	51.1	51.2	51.3	51.6
0.8	49.3	51.8	53.2	54.0	54.6	55.3	55.8	56.1	56.5	56.7	56.9	57.0	57.1	57.2	57.3	57.6
0.9	53.7	56.6	58.1	59.1	59.7	60.6	61.1	61.4	61.9	62.1	62.3	62.5	62.6	62.7	62.8	63.2
1.0	57.8	60.9	62.6	63.7	64.4	65.3	65.9	66.3	66.8	67.1	67.3	67.5	67.6	67.7	67.8	68.3
1.1	61.4	64.8	66.7	67.9	68.7	69.7	70.3	70.7	71.2	71.6	71.8	72.0	72.1	72.2	72.3	72.9
1.2	64.7	68.4	70.4	71.6	72.5	73.6	74.2	74.7	75.2	75.6	75.9	76.0	76.2	76.3	76.4	77.0
1.3	67.7	71.6	73.7	75.0	75.9	77.0	77.7	78.2	78.8	79.2	79.5	79.7	79.8	79.9	80.0	80.6
1.4	70.4	74.4	76.6	78.0	78.9	80.1	80.8	81.3	81.9	82.3	82.6	82.8	83.0	83.1	83.2	83.8
1.5	72.8	77.0	79.2	80.6	81.6	82.8	83.6	84.1	84.7	85.1	85.4	85.6	85.7	85.9	86.0	86.6
1.6	75.0	79.2	81.5	83.0	83.9	85.2	85.9	86.4	87.1	87.5	87.8	88.0	88.1	88.3	88.4	89.0
1.7	76.9	81.3	83.6	85.0	86.0	87.3	88.0	88.5	89.2	89.5	89.9	90.1	90.2	90.3	90.5	91.1
1.8	78.7	83.1	85.4	86.8	87.8	89.1	89.8	90.3	90.9	91.3	91.6	91.8	92.0	92.1	92.2	92.8
1.9	80.2	84.7	87.0	88.4	89.4	90.6	91.3	91.8	92.4	92.8	93.1	93.3	93.4	93.5	93.7	94.3
2.0	81.7	86.1	88.4	89.8	90.8	92.0	92.7	93.1	93.7	94.1	94.4	94.5	94.7	94.8	94.9	95.4
2.1	83.0	87.4	89.7	91.0	92.0	93.1	93.8	94.3	94.8	95.1	95.4	95.6	95.7	95.8	95.9	96.4
2.2	84.1	88.5	90.8	92.1	93.0	94.1	94.8	95.2	95.7	96.0	96.3	96.4	96.6	96.6	96.8	97.2
2.3	85.2	89.5	91.7	93.0	93.9	95.0	95.6	96.0	96.5	96.8	97.0	97.1	97.3	97.3	97.4	97.9
2.4	86.2	90.4	92.6	93.9	94.7	95.7	96.3	96.7	97.1	97.4	97.6	97.7	97.8	97.9	98.0	98.4
2.5	87.1	91.3	93.3	94.6	95.4	96.3	96.9	97.2	97.6	97.9	98.1	98.2	98.3	98.3	98.4	98.8
2.6	87.9	92.0	94.0	95.2	95.9	96.8	97.4	97.7	98.1	98.3	98.5	98.6	98.6	98.7	98.8	99.1
2.7	88.6	92.6	94.6	95.7	96.5	97.3	97.8	98.1	98.4	98.6	98.8	98.9	98.9	99.0	99.1	99.3
2.8	89.3	93.2	95.1	96.2	96.9	97.7	98.1	98.4	98.7	98.9	99.0	99.1	99.2	99.2	99.3	99.5
2.9	89.9	93.8	95.6	96.6	97.3	98.0	98.4	98.7	99.0	99.1	99.2	99.3	99.4	99.4	99.5	99.6
3.0	90.5	94.3	96.0	97.0	97.6	98.3	98.7	98.9	99.2	99.3	99.4	99.5	99.5	99.5	99.6	99.7
3.2	91.5	95.1	96.7	97.6	98.2	98.7	99.1	99.2	99.4	99.6	99.6	99.7	99.7	99.7	99.8	99.9
3.4	92.4	95.8	97.3	98.1	98.6	99.1	99.3	99.5	99.6	99.7	99.8	99.8	99.8	99.9	99.9	99.9
3.6	93.1	96.3	97.7	98.5	98.9	99.3	99.5	99.6	99.8	99.8	99.9	99.9	99.9	99.9	99.9	100.0
3.8	93.7	96.8	98.1	98.8	99.1	99.5	99.7	99.8	99.8	99.9	99.9	99.9	99.9	100.0	100.0	100.0
4.0	94.3	97.2	98.4	99.0	99.3	99.6	99.8	99.8	99.9	99.9	100.0	100.0	100.0	100.0	100.0	100.0

Note: The Gaussian probability for each value of t is listed in the last column.

APPENDIX D

HISTOGRAMS AND GRAPHS

Graphs of experimental data and of theoretical predictions have always been important tools for scientists, in both the actual performance of research and in presentations of results. In recent years we have seen a proliferation of graphics displays as fast inexpensive computers and printers have facilitated the display-making process. Scientists have benefited from the new techniques and equipment, with many excellent commercial programs available for creating high-quality scientific graphics suitable for publication.

In science, the object is to present results in a straightforward manner so that relevant points are illustrated clearly and without bias. Graphs with suppressed zeros, which are common in advertisements, are not often seen in scientific papers. Bar graphs tend to be simple histograms rather than the multibar, brightly colored displays of magazines and newspapers. In fact, although the use of color is growing, especially in direct publication on the Internet, few scientific preprints and papers are printed in color, although discrete use can clarify graphical presentations significantly. Error bars, which are rare indeed in advertisements, are essential in a scientific presentation. Exaggerated perspective and distorted scales have very limited use in scientific work whereas semilogarithmic plots that are often used in science are not often seen in business publications.

It is often convenient to have graphics routines that are part of a simulation or an analysis program, rather than to use a separate graphing program. For example, in a Monte Carlo simulation, it is essential to be able to produce histograms and data graphs quickly at each stage of the study. Generations of scientists have made simple histograms on monitors or printers to make preliminary studies of their data. We provide some simple routines of this type in the source files associated with this book.

More elegant and detailed graphs can be created by using the graphics features of particular programming languages, and those provided by data analysis programs and spreadsheets. Such programs can produce high-quality graphs and charts suitable for presentations and publications. Many of the graphs in this book, such as those in Chapter 2, were created by programs written in Fortran and Pascal. Others, such as those in Chapter 11, were created in Origin, a very powerful data analysis program with strong graphing features.

D.1 MAKING A GRAPH

Whether a scientific graph is produced by hand or by computer, there are several basic principles that should be followed. The graph should be large enough to be read and understood easily, with appropriately proportioned abscissa and ordinate. Axes should be labeled with large, clean letters, and the axes scales should be clearly indicated. If more than a single function is displayed, or if both data and curves are displayed, a box, or legend, may be superimposed on the graph to indicate the meaning of different symbols. In scientific journals, a description of the graph is generally included as text below the abscissa label. In internal papers and preprints, these descriptions are often collected in a separate section of the paper. For visual presentation, some descriptive material may be included in a box on the graph, but it is important that text be large enough to be clearly legible. One should avoid scattering too much material over any graph, which gives a busy appearance. A properly made graph should not require many words of explanation.

It is generally advisable to plot the independent variable as the abscissa and the dependent variable as the ordinate. However, if the independent data have a high degree of uncertainty while the corresponding measurements of the dependent data can be made with high precision, then it might be wise to interchange the two axes to simplify least-squares fitting.

Reasonable, convenient values and intervals should be chosen for the scale marks on the two axes. For example, if abscissa values range from 0 to 400, it might be reasonable to divide the x-axis into eight parts and thus to mark the abscissa with major, labeled ticks at 0, 100, 200, 300, and 400, with minor ticks half-way between. Dividing the axis into six parts and putting ticks at 66.7, 133.3, and so forth, would make it very difficult for a reader to interpret.

In general, error bars should be included for ordinate variables except for simple histograms where the text clearly specifies that the uncertainties are statistical and therefore given by the square root of the value of the coordinate. Unless otherwise noted, error bars generally indicate the standard deviation. Error bars usually are not necessary for abscissa variables. However, if appropriate, they may be drawn to indicate the resolution of the measurement or setting, or they may simply indicate the range of the variable over which data have been collected or grouped, as in the case of the width of a histogram bin. The text must explain the meaning of such error bars. If no error bar is shown for the abscissa, then it is useful to draw a circle or other symbol at each data point to indicate the position of the central values of each coordinate pair.

D.2 GRAPHICAL ESTIMATION OF PARAMETERS

A graph of y versus x often provides a convenient way of estimating parameters of the relation $y = y(x)$. The simplest example is the straight line

$$y = A + Bx \tag{D.1}$$

where the slope and the intercept can be estimated by making a graph and drawing a straight line that relates y to x. Clearly the better way to handle this problem is by a least-squares fitting technique, but the graphical method can be useful in both research and instructional laboratories for obtaining quick preliminary estimates of experimental results.

If we wish to find from the graph the uncertainty in our estimate of the slope, then we should attempt to draw two lines through the data, corresponding to estimates of the largest and smallest *reasonable* slopes, s_1 and s_2. We should take account in the uncertainties in the data points, if they are available, and, because we are trying to estimate the uncertainty as a standard deviation, we should attempt to draw these two lines to bracket about two-thirds of the data points—not all the points. Making this estimate is often difficult and subjective, especially if there are few points and they exhibit a lot of scatter. The mean slope s is just the average of our two slopes,

$$s = (s_1 + s_2)/2 \tag{D.2}$$

and an approximate estimate of the uncertainty is the magnitude of half the difference

$$\sigma = |s_1 - s_2|/2 \tag{D.3}$$

To gain practice in determining parameters from a graph, it is a worthwhile exercise to estimate the parameters from the graph and to compare those estimates with the results of a least-squares fit to the data. We should note that the two lines selected to give a reasonable estimate of the uncertainty in the slope may not be the same two lines we might draw to obtain a reasonable estimate of the uncertainty in the intercept. Figure D.1 displays the data of Figure 1.1b, with lines bracketing the points to show (a) reasonable ranges for estimating the intercept, and (b) reasonable ranges for estimating the slope. These lines were actually calculated from the results of a least-squares fit of the equation $Y = A + Bx$ to the data, which yielded the parameters A and B and their uncertainties σ_A and σ_B. We calculated the two lines in Figure D.1a from the equations $Y = (A \pm \sigma_A) + Bx$ and those in Figure D.1b from the equations $Y = A + (B \pm \sigma_B)x$. We note that these lines are just particular examples of an infinite number of such lines corresponding to all combinations of the slope and intercept within one standard deviation ranges, and in any given graph, a decision must be made on which lines to draw. In particular, allowing the lines to intersect at the intercept as in Figure D.1b may not give the best solution, although it can be a good starting point.

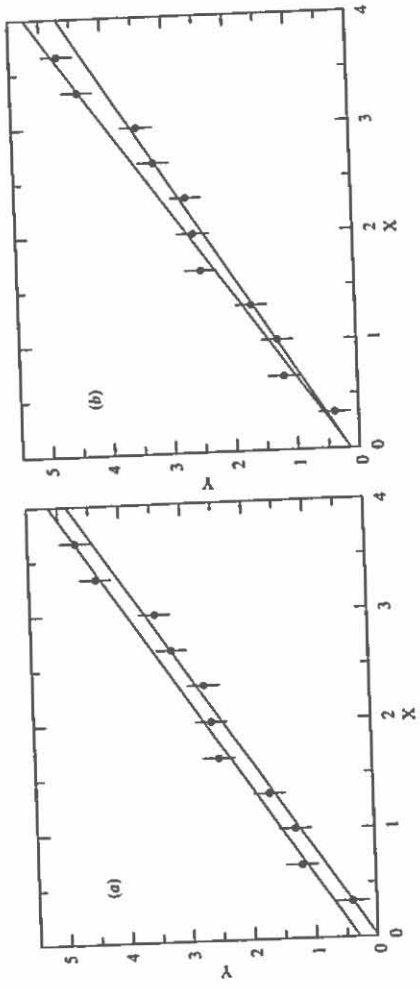

FIGURE D.1
Results of a least-squares fit of a straight line to the data of Figure 1.1b plotted to show separately the range of ±1 standard deviation (*a*) in the intercept, and (*b*) in the slope. Units on the axes are arbitrary.

Semilogarithmic Graphs

When dealing with an exponential decay function, it is convenient to display the activity as a function of time on a semilogarithmic graph. That is, if the relation is

$$y(t) = y_0 e^{-at} \tag{D.4}$$

we plot a graph of $\log(y)$ versus x. Fortunately semilogarithmic graph paper is readily available so that it is not necessary actually to calculate any logarithms to make this plot. We merely have to select paper with the appropriate number of powers of 10 for our plot, label the axes, and plot y versus x on the graph. Such a graph is illustrated in Figure 8.1 for Example 8.1.

Semilogarithmic graph paper comes in various *cycles*, corresponding to the number of *decades* or powers of 10 that can be plotted on a single sheet. Thus, for example, on three-cycle paper we can plot y values that range from 1 to 1000 (or from 0.01 to 10.0, etc.). Note that we can never plot y values that are zero or negative on semilogarithmic paper. This is a problem when dealing with subtracted distributions, such as the counting experiment of Example 8.1, where, if we wish to plot the number of counts remaining after we have subtracted the average background from cosmic rays, we discover that, at large times, some bins have negative net counts. Those points, of course, cannot be displayed on a semilogarithmic graph. A full, least-squares fit to the total, unsubtracted data sample is clearly the right way to solve this problem, but if we are to attempt a graphical solution, we should be aware of this limitation.

We can determine from our data the parameter a in Equation (D.4) by finding the slope of the straight line on the semilogarithmic graph just as we found the slope on ordinary graph paper for a simple linear plot. Note that when calculating the slope we must compute the logarithms of the y values. Thus, if the two ends of the straight line have coordinates (x_1, y_1) and (x_2, y_2), the slope is given by

$$s = \frac{\ln(y_2) - \ln(y_1)}{x_2 - x_1} = \frac{\ln(y_2/y_1)}{x_2 - x_1} \tag{D.5}$$

The uncertainty in the slope can again be determined by drawing two straight lines that bracket the mean slope, although the logarithmic form of the plot decreases the accuracy in this determination.

Full-Logarithmic Graphs

If we wish to display a power relation of the form $y = Ax^n$, we may make a plot of y versus x on full-logarithmic paper or *log-log* paper. The result will be a straight line with slope n and we can obtain the slope, and therefore the exponent n, from the graph. This technique could be used, for example, to check the $1/r^2$ law for radiation intensity as a function of distance, by plotting a graph of intensity versus distance on log-log paper.

In Section 7.4 we discuss variable transformation as a method of converting a nonlinear fitting problem to a linear problem, and the distortions that may be

introduced into the uncertainties in the process. Plotting on semilogarithmic or full-logarithmic paper is equivalent to such a variable change and we should attempt to compensate for these distortions, if necessary.

D.3 HISTOGRAMS AND FREQUENCY PLOTS

If we wish to display the frequency distribution of a measured variable x, then a histogram is generally the simplest and clearest form of presentation. For example, we may have observed particles emitted in the decay of an unstable state and wish to present the number detected in successive time intervals as in Example 2.4. Alternatively, we may have measured secondary particles in a scattering experiment and wish to display the distribution of their energies. In such cases, we can display the frequency distribution of the individual measurements, or *events*, as a histogram of $f(x)$ versus x, where $f(x_i)$ is the number of events that have values of x between x_i and $x_i + \Delta x$, and Δx is the histogram interval or bin width.

An alternate procedure for displaying binned data, which is especially useful for distributions with large numbers of bins, or for data with nonstatistical uncertainties, is to make a regular graph of frequency versus the measured variable, a *frequency plot*, with the data points indicated by crosses and uncertainties by error bars. This procedure is especially convenient when there are many bins or when error bars must be displayed, as illustrated in Figure 8.1.

A convenient procedure for finding the frequency distribution of (or *binning*) a continuous variable x is to label a bin with a tick mark at the lower limit x_i of the bin and to count within a bin those events for which $x_i \leq x < x_i + \Delta x$. This is suitable for most, but not all, data sets. Choice of the bin width depends on a number of factors. In the ideal situation with a large quantity of high-precision data, the bin width could be chosen to be very small. However, in real experiments, the number of events may not be very large and each x coordinate will have some uncertainty. As a general rule, the bin width should not be less than the uncertainty in the measured variable x and one should be very wary of any data structure that is narrower than the uncertainty in x. If the number of events is relatively small, then even wider binning may be necessary. With such data, the competition between statistical significance and resolution of narrow effects in the histogram may become important. A histogram with less than ten events in its highest bin is not generally very informative, considering that the uncertainty in that bin will be over 30%.

A problem arises when the bin width of a histogram is close to or equal to the least count of the data. This can happen when the data are integral numbers or with data that have been collected by a digital device. The previous suggestions that the histogram bins be labeled with the lower limit at the left of the bin may not be reasonable for such data, and it may be better to place tick marks at the center of the bins.

EXAMPLE D.1 A student in an introductory physics laboratory attempts to measure the value of the acceleration of gravity by timing a ball that she drops 50 times from a height of 3 m. She uses an electronic timer with a least count of 0.01 s. The timer starts when the ball is released and stops when it hits the floor. Uncertainties in the measurements come mainly from variations in the starting and stopping times.

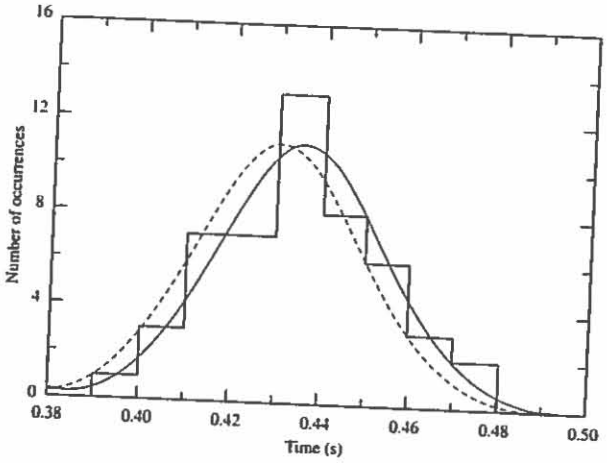

FIGURE D.2
Histogram of measured times plotted with the bin width equal to the least count of a digital clock. The numbers on the abscissa correspond to the lower time limit of the bin. The dashed Gaussian curve was calculated from the mean and standard deviation of the measurements. The solid curve was calculated with the mean increased by half the bin width to correct for the truncation of the data.

The student's measurements have been plotted in the histogram of Figure D.2 where the bin width is equal to the least count (0.01ts). We assume that the digital clock truncates the measured times so each time measurement corresponds to the left-hand edge of a bin and the actual value of the time is somewhere within the bin limits. Thus, in this case it is appropriate to indicate the lower value of the bin limit at the left-hand edge of the bin.

The dashed Gaussian curve was calculated from the mean ($\bar{t} = 0.431$ s) and standard deviation ($s = 0.0184$) of the measurements. The curve clearly is shifted to the left relative to the data. The discrepancy is caused by the fact that we neglected to correct for the truncation of the data by the digital clock. To correct the mean we must add to it half the width of a bin to obtain $\bar{t} = 0.431 + 0.005 = 0.436$ s. The Gaussian curve, calculated from the corrected mean, is shown as a solid line.

Normalized Curves on Histograms

When superimposing a theoretical curve on a data histogram, we often want to scale the area of the curve to that of the histogram, or to *normalize* the curve. The required scale factor can be determined in the following way. We assume that the curve has been calculated from a probability distribution function that is normalized

to unit area, such as the Gaussian probability function of Equation (2.23). The area of one event on the histogram is equal to the bin width Δx multiplied by a unit interval on the ordinate. Thus, the total area of the histogram is equal to the bin width multiplied by the total number of events ($A = N\Delta x$). To scale the curve to the area of the histogram, we multiply the values $p(x_i)$, calculated from the equation of the probability distribution, by the product of the number of events on the plot and the bin width, so that the plotted curve becomes

$$y(x_i) = p(x_i) \times N\Delta x \tag{D.6}$$

D.4 GRAPHICS ROUTINES

We include source files on the website for routines which can be used to make simple graph and histograms. Most of the sample computer routines in this book make calls to these routines.

Program D.1 QUIKSCRP (website) accepts data that define graphs and histograms and writes a script file that can be read and interpreted by the executable program QDISPLAY.EXE (website) to produce displays on the monitor. Details of the calling procedures can be seen in the routine PLOTIT in the program unit \CHAPT-6\FITUTIL (website) called from the program \CHAPT-6\FITLINE (APPENDIX E). For this program QuikScrp writes an output file FITLINE.SCR.

Program D.2 QUIKHIST (website) collects data and presents a character-based histogram on the monitor. Printed output is also available. PROGRAM 5.2: \CHAPT-5\POISDCAY illustrates use of this program.

Program D.3 QDISPLAY.EXE (website) is an executable program that reads a script file written by QUIKSCRP and interprets the file to create a graphics display on the monitor. The command line instruction for running QDISPLAY with the script file produced by the program FITLINE is QDISPLAY FITLINE.

APPENDIX E

COMPUTER ROUTINES IN FORTRAN

This appendix lists several routines that illustrate the material of the text. The routines are listed in Fortran 77, an old, but quite readable version of that ever-popular programming language. All routines have been tested; however, most of them requires subsidiary routines and drivers that are not listed. Complete programs and routines are available on the Web in C++ as well as in Fortran. Readers are urged to log onto the website at www.mhhe.com/bevington to download these programs.

We have tried to keep the routines simple, trading efficiency for clarity where necessary. To make explicit which modules are required to form a complete program, and to avoid the need for command strings to link the object programs into an executable program, we have chosen to use the INCLUDE statement to present the compiler with a single source file from which to compile a single object module incorporating all required routines. We also use the INCLUDE statement to copy blocks of COMMON and other variable-defining statements into routines.

Because readers may not be familiar with Fortran, we list a few basic principles that should help in understanding the instructions and following their logic. This list includes only a selection of language elements that appear in the sample programs.

STATEMENTS

The format of Fortran statements was defined in terms of the 80-column Hollerith card:

column 1: blank or with a "C" to indicate that the information on the line is a "Comment";

columns 2–5: statement label (a number);

column 6: reserved for a single digit number to indicate a continuation of the statement from the previous line;

columns 7–72: program statements;

columns 73–80: not used.

Although it is not necessary to follow rigorously this scheme with a modern interactive compiler on a personal computer (for example, "tabs" can be used), the general order must be followed.

PROGRAM FLOW

Program flow can be controlled by IF statements, by IF THEN statements (with ELSEIF and ENDIF), by DO AND DOWHILE statements that may refer to a termination label (all statements labels are numerical) or to the DO terminator; ENDDO, and by GOTO statements. Excessive use of the GOTO statement can lead to very confusing programs. In order to facilitate following the program flow, we have indented groups of instructions that are accessed through a control statement, such as IF THEN, or DO.

Examples

```
DO 100I = 1 TO 20
  X = I
  Y(I) = SQRT(X)
100 CONTINUE
```

```
DO I = 1 TO 20
  X = I
  Y = SQRT(X)
ENDDO
```

```
X = 1
DO WHILE X .LE. 20
  Y = SQRT(X)
  X = X + 1
ENDDO
```

VARIABLE DEFINITIONS

Fortran does not require the rigorous variable typing of newer languages. As default typing, variables with names beginning with I, J, K, L, M, or N are defined as INTEGER; variable names beginning with other letters are identified as REAL. However, we have attempted to identify most of the variables in the routines and in some instances have violated the default typing for program clarity.

Examples

```
INTEGER S1, S3, N/10/
REAL X, T, TPRIME, SIGMAT/1.0/
LOGICAL NEXTVAR/.FALSE./
```

Note that the variables N, SIGMAT and, NEXTVAR in the preceding examples have been initialized to the values 10, 1.0, and FALSE, respectively. The DATA statement also can be used to initialize variables. DATA SQRTPI/1.7724539/

Other types include:

```
CHARACTER
COMPLEX
DOUBLE PRECISION
```

Variables defined in named COMMON statements are available to any routine that includes the statement. Local variables can be defined in DIMENSION STATEMENTS. Array sizes may be defined in PARAMETER statements or directly in a COMMON or DIMENSION statement.

Examples

```
PARAMETER(MAXPARAM 10)
COMMON/FITVARS/ NPTS, M, NFREE, MARRAY(MAXPARAM),
ZARRAY(200)
DIMENSION NPLAN(30).
```

Fortran has several types of subprograms that can be called from another routine: SUBROUTINE and FUNCTION are the most common. Data types defined in a subprogram must be consistent with the definitions in the calling routine. A function name must specify its own data type.

Examples

```
CALL SETRANDOMDEVIATESEED(S1, S2, S3)
TPRIME = GAUSSSMEAR(T,SIGMAT)
REAL FUNCTION GAUSSSMEAR(X,DX)
SUBROUTINE SETRANDOMDEVIATESEED(SA,SB,SC)
```

The INCLUDE statement copies the specified file into the body of the program.

Example

```
INCLUDE '\CHAPT-5\MONTEINC.FOR'
```

As well as comment statements that begin with a "C" in column 1, comments may appear in statement lines, preceded by the exclamation point (!).

E.1 Routines from Chapter 5

```
C PROGRAM 5.1: \CHAPT-5\HOTROD.FOR
C SIMULATED VARIATION OF TEMPERATURE ALONG A METAL ROD
C 10 CM ROD-TEMPERATURE IS ZERO AT ONE END, 100 DEGREES C AT OTHER.
C USES MONTELIB
      PROGRAM HOTROD
      INTEGER S1, S2, S3, N/10/          !--- GENERATE 10 POINTS AT 1 CM INTERVALS
      REAL  X, T, TPRIME, SIGMAT/1.0/ !--- WITH AN UNCERTAINTY OF +-1 DEGREE
      REAL GAUSSSMEAR
      S1 = 1171
      S2 = 343
      S3 = 1322
      CALL SETRANDOMDEVIATESEED(S1, S2, S3)
      PRINT *,'      HOT ROD TEST DATA,   SIGMA=', SIGMAT
      X = -0.5
      DO 100 I = 1, N
         X     = X + 1.0          !--- POSITION ALONG ROD
         T     = 10.0*X           !--- CALCULATE MEAN TEMPERATURE AT POINT
         TPRIME = GAUSSSMEAR(T,SIGMAT)          !--- SMEAR IT
         PRINT *,I, X, T, TPRIME
  100 CONTINUE
      CALL EXIT
      END
      INCLUDE C:\CHAPT-5\MONTELIB.FOR

C PROGRAM 5.2: \CHAPT-5\POISDCAY.FOR
C SIMULATED DECAY OF AN UNSTABLE STATE.
C USES QUIKHIST, MONTELIB
      PROGRAM POISDCAY       !--- GENERATE A 200-EVENT POISSON  HISTOGRAM
      REAL LO/0/, INT/1/, HI/22/
      INTEGER NEVENTS/400/, POISSONDEVIATE
      REAL MU/8.4/
      INTEGER S1, S2, S3, I, K
      REAL   X
      S1 = 1171
      S2 = 343
      S3 = 1322
      CALL SETRANDOMDEVIATESEED(S1, S2, S3)
      CALL HISTINIT(' ')          !---OUTPUT FILE NAME OR ' ' FOR MONITOR OUTPUT
      CALL HISTSETUP(1,LO,INT,HI,'POISSON - COUNTS/10 SEC')
      K=POISSONDEVIATE(MU,.TRUE.)          !--- INITIALIZE - MAKE THE TABLE
      DO 100 I = 1, NEVENTS
         K = POISSONDEVIATE(MU,.FALSE.)
         X = K
         CALL HISTOGRAM(1,X)
  100 CONTINUE
      CALL HISTDISPLAYALL(.FALSE.) !DUMMY ARG-COMPAT. WITH QUIKSCRP
      CALL EXIT
      END
      INCLUDE \CHAPT-5\MONTELIB.FOR
      INCLUDE \APPEND-D\QUIKHIST.FOR !REPLACE WITH QUIKSCRP FOR GRAPHICS
```



```
C PROGRAM 5.3: \CHAPT-5\MONTELIB.FOR
C MONTE CARLO LIBRARY ROUTINES
      SUBROUTINE SETRANDOMDEVIATESEED(SA,SB,SC)
      INCLUDE '\CHAPT-5\MONTEINC.FOR'
      INTEGER SA, SB, SC
      SEED1 = SA
      SEED2 = SB
      SEED3 = SC
      RETURN
      END

      SUBROUTINE GETRANDOMDEVIATESEED(SA,SB,SC)
      INCLUDE '\CHAPT-5\MONTEINC.FOR'
      INTEGER SA, SB, SC
      SA = SEED1
      SB = SEED2
      SC = SEED3
      RETURN
      END

      REAL FUNCTION RANDOMDEVIATE()          !--- WICHMANN AND HILL
      INCLUDE '\CHAPT-5\MONTEINC.FOR'
      REAL  TEMP
      SEED1 = 171*MOD(SEED1,177) - 2*(SEED1 / 177)
      IF (SEED1 .LT. 0 ) SEED1 = SEED1 + 30269
      SEED2 = 172*MOD(SEED2,176) - 35*(SEED2 / 176)
      IF (SEED2 .LT. 0 ) SEED2 = SEED2 + 30307
      SEED3 = 170*MOD(SEED3,178) - 63*(SEED3 / 178)
      IF (SEED3 .LT. 0 ) SEED3 = SEED3 + 30323
      TEMP = SEED1/30269. + SEED2/30307. + SEED3/30323.
      RANDOMDEVIATE = TEMP-AINT(TEMP)
      RETURN
      END

C -FIND A RANDOM VARIABLE DRAWN FROM THE GAUSSIAN DISTRIBUTION-
      REAL FUNCTION RANDOMGAUSSDEVIATE()          !--- BOX-MUELLER
      INCLUDE '\CHAPT-5\MONTEINC.FOR'
      LOGICAL  NEXTVAR/.FALSE./
      REAL R, F, Z1, Z2, X1RANGAUSS, RANDOMDEVIATE
      IF (NEXTVAR) THEN
         NEXTVAR = .FALSE.
         RANDOMGAUSSDEVIATE = X2RANGAUSS
      ELSE
  100    Z1 = -1 + 2*RANDOMDEVIATE()
         Z2 = -1 + 2*RANDOMDEVIATE()
         R  = Z1*Z1 + Z2*Z2
         IF (R .GE. 1) GOTO 100
         F = SQRT(-2*ALOG(R)/R)
         X1RANGAUSS = Z1*F
         X2RANGAUSS = Z2*F
         RANDOMGAUSSDEVIATE = X1RANGAUSS
```

```
        NEXTVAR = .TRUE.
      ENDIF
      RETURN
      END

      REAL FUNCTION GAUSSSMEAR(X,DX)
      REAL X, DX
      REAL RANDOMGAUSSDEVIATE
      GAUSSSMEAR = X + RANDOMGAUSSDEVIATE() * DX
      RETURN
      END

C -RECURSION METHOD FOR POISSON PROBABILITY (P(N,M). TO FIND P(N,M) MUST
C   ALL WITH SUCCESSIVE ARGUMENTS J=0,1,..N. MAX MU=85, NO LIMIT ON X
      REAL FUNCTION POISSONRECUR(J, M)
      INCLUDE '\CHAPT-5\MONTEINC.FOR'
      INTEGER J
      REAL M
      IF (J.EQ.0 ) THEN
         POISS = EXP(-M)
      ELSE
         POISS = (POISS*M)/J        !--- POISS =  (M^J)EXP(-MU/J)
      ENDIF
      POISSONRECUR = POISS
      RETURN
      END

C -FIND A RANDOM VARIABLE DRAWN FROM THE POISSON DISTRIBUTION
      INTEGER FUNCTION POISSONDEVIATE(MU, INIT)
      INCLUDE '\CHAPT-5\MONTEINC.FOR'
      INTEGER   I, X, N
      REAL      MU, P , R, POISSONRECUR
      LOGICAL   INIT
      IF (INIT ) THEN                        ! --- MAKE TABLE OF SUMS ---
         N = AINT(MU + 8* SQRT(MU))          ! ---IE., 8*SIGMA
         IF (N .GT. MAXBINS ) THEN
            PRINT *, 'OVERFLOW ERROR IN ROUTINE POISSON DEVIATE'
            CALL EXIT
         ENDIF
         PTABLE(0) = POISSONRECUR(0,MU)
         DO 100 I = 1, N-1
            P = POISSONRECUR(I,MU)
            PTABLE(I) = PTABLE(I-1)+P
  100    CONTINUE
         PTABLE(N) = 1                       ! --- ASSURE UNIT PROBABILITY ---
      ELSE                                   ! --- GENERATE AN EVENT ---
         X = -1
         R = RANDOMDEVIATE()
  200    X = 1 + X
         IF (PTABLE(X) .LE. R) GOTO 200 !--- REPEAT UNTIL PTABLE(X) >= X
         POISSONDEVIATE = X
```



```
      ENDIF
      RETURN
      END

PROGRAM 5.4:  \CHAPT-5\KDECAY.FOR            (WEBSITE)
C ILLUSTRATION OF EXAMPLE 5.7

C PROGRAM 5.5:  \CHAPT-5\MONTEINC.FOR
C COMMON FOR MONTE CARLO LIBRARY
      COMMON/MC/ SEED1, SEED2, SEED3, X2RANGAUSS, POISS, PTABLE
      PARAMETER (MAXBINS = 100)
      INTEGER SEED1, SEED2, SEED3
      REAL X2RANGAUSS, PTABLE(0:MAXBINS)
      REAL*8 POISS
C-------------END MONTEINC ------------------------
```

E.2 Routines from Chapter 6

```
C PROGRAM 6.1: \CHAPT-6 FITLINE.FOR
C LEAST-SQUARES FIT TO A STRAIGHT LINE BY METHOD OF DETERMINANTS
C USES FITUTIL
      PROGRAM FITLINE
C ------------------------MAIN  ROUTINE----------------------
      INCLUDE '\CHAPT-6 FITVARS.FOR'
      CHARACTER*40  TITLE
      CHARACTER*1 VORG, READCHAR
      INTEGER I
      REAL DET, CHI2, CALCCHISQ
      M = 2                                          !--- FIND 2 PARAMETERS
      PRINT *, '(V)OLTS OR (G)EIGER? '
      VORG = READCHAR()
      IF (VORG .EQ. 'V') THEN
         CALL FETCHDATA('\CHAPT-6\VOLTS.DAT' ,TITLE)     !--- EXAMPLE 6.1
      ELSEIF ((VORG .EQ. 'G') .OR. (VORG .EQ. 'G')) THEN
         CALL FETCHDATA('\CHAPT-6\GEIGER.DAT',TITLE)     !--- EXAMPLE 6.2
         DO 100 I = 1 , NPTS
         X(I)  = 1/X(I)**2                            !--- FITTING 1/R^2
  100    CONTINUE
      ENDIF
      CALL LINEFIT(DET)
      CALL CALCULATEY        !--- FILL ARRAY YCALC FOR CALCCHISQ AND PLOTIT
      CHI2 = CALCCHISQ()
      CALL OUTPUT(.FALSE. , 'CON', CHI2, TITLE)  !--- FALSE FOR NO ERROR MATRIX
      IF (VORG .EQ. 'V') THEN
         CALL PLOTIT('FITLINE.SCR',.FALSE.,.FALSE., !---SCRPT FILE,LOG?,SPLINE?
     1   'C', ABS(X(2)-X(1))/20,                !--- DATA CIRCLE, RAD OF CIRCLE
     2   0.0, 0.0, 100.0, 3.0,                  !---  X1,Y1,X2,Y2
```

```
     3     5, 6,                                  !---  # X-DIV, # Y-DIV
     4     'X (CM)', 'POTENTIAL DIFF(ERENCE (VOLTS)')  !---  AXIS LABELS
         ELSEIF (VORG .EQ. 'G') THEN
            CALL PLOTIT('FITLINE.SCR',.FALSE. ,.FALSE.,
     1     'C', ABS(X(2)-X(1))/50,  0.0, 0.0, 30.0, 1000.0,  6, 5,
     2     'SQUARED INVERSE DISTANCE (1/M^2)', 'NUMBER OF COUNTS PER SEC')
         ENDIF
         READ *
         CALL CLOSEGRAPHICS
         END

         SUBROUTINE CALCULATEY        !---  FILLS ARRAY YCALC
         INCLUDE '\CHAPT-6\FITVARS.FOR'
         INTEGER I
         DO 100 I= 1 , NPTS
            YCALC(I) = A(1) + A(2)*X(I)
  100    CONTINUE
         RETURN
         END

         REAL FUNCTION CALCCHISQ() !---  ASSUMES ARRAY YCALC HAS BEEN FILLED
         INCLUDE '\CHAPT-6\FITVARS.FOR'
         INTEGER  I
         REAL     CHI2
         CHI2=0.
         DO 100 I = 1 , NPTS
            CHI2 = CHI2 + ( (Y(I)-YCALC(I))/SIGY(I))**2
  100    CONTINUE
         CALCCHISQ = CHI2
         RETURN
         END

         SUBROUTINE LINEFIT(DET)
         INCLUDE '\CHAPT-6\FITVARS.FOR'
         REAL DET
         INTEGER I
         REAL SUMWT, SUMX, SUMY, SUMX2, SUMY2, SUMXY, WEIGHT
         SUMWT = 0
         SUM   = 0
         SUMY  = 0
         SUMX2 = 0
         SUMY2 = 0
         SUMXY = 0
C --------- ACCUMULATE WEIGHTED SUMS ----------
         DO 100 I= 1 , NPTS
            WEIGHT = 1/SIGY(I)**2
            SUMWT  = SUMWT  + WEIGHT
            SUMX   = SUMX   + WEIGHT * X(I)
```



```
            SUMY   = SUMY   + WEIGHT * Y(I)
            SUMX2  = SUMX2  + WEIGHT * X(I)**2
            SUMY2  = SUMY2  + WEIGHT * Y(I)**2
            SUMXY  = SUMXY  + WEIGHT * X(I)*Y(I)
  100    CONTINUE
C ---CALCULATE THE PARAMETERS - CUT OUT IF DETERMINANT IS NOT > 0 ---
         DET =  SUMWT * SUMX2 - SUMX * SUMX
         IF (DET .GT. 0 )  THEN
            A(1)    = (SUMX2*SUMY  - SUMX*SUMXY)/DET
            A(2)    = (SUMXY*SUMWT - SUMX*SUMY) /DET
            SIGA(1) = SQRT(SUMX2/DET)
            SIGA(2) = SQRT(SUMWT/DET)
         ELSE
            CALL ERRORABORT('DETERMINANT < OR = 0 IN LINEFIT')
         ENDIF
         RETURN
         END
         INCLUDE '\CHAPT-6\FITUTIL.FOR' ! FITUTIL INCLUDES QUIKSCRP.FOR


C PROGRAM 6.2: \CHAPT-6\FITVARS.FOR          (WEBSITE)
C INCLUDE FILE OF CONSTANTS, VARIABLES AND  ARRAYS FOR LEAST-SQUARES FITS
C ALL GLOBAL TYPES, CONSTANTS AND VARIABLES ARE DECLARED HERE.
C THE ARRAY LIMITS MAXDATA AND MAXPARAM CAN BE SET AS REQUIRED
                                              FOR PARTICULAR PROBLEMS.

C PROGRAM 6.3: \CHAPT-6\FITUTIL.FOR          (WEBSITE)
C GENERAL UTILITY ROUTINES
```

E.3 Routines from Chapter 7

```
C PROGRAM 7.1: \CHAPT-7\MULTREGR.FOR
C LEAST-SQUARES FIT TO A POWER SERIES AND TO LEGENDRE POLYNOMIALS.
C USES  FITFUNC7, MAKEAB7, MATRIX, FITUTIL

         PROGRAM MULTREGR
C        M = NUM OF PARAMETERS, NPTS=NUMBER OF DATA PAIRS,
C        DATA AND UNCERTAINTIES ARE IN ARRAYS  X, Y, DY.
         INCLUDE '\CHAPT-6\FITVARS.FOR'
         COMMON /FITVARS7/PAE
         CHARACTER * 1 PAE
         REAL     DET, CHI2, CALCCHISQ
         INTEGER  I
         LOGICAL  SPL
         CHARACTER*1 READCHAR
         CHARACTER*40 TITLE
         PRINT *,   '(P)OWER SERIES, (A)LL LEGENDRE TERMS TO L = 4,'
```

```
      PRINT *,   'OR (E)VEN LEGENDRE TERMS(L = 0,2,4).'
      PRINT *,   'TYPE P,A OR E '
      PAE = READCHAR()
1000  FORMAT(A1)
      IF (PAE .EQ. 'P') THEN
         CALL FETCHDATA ('\CHAPT-7\THERMCOU.DAT', TITLE)
         PRINT *,   'TYPE NUMBER OF PARAMETERS '
         READ *, M
      ELSEIF (PAE .EQ. 'A') THEN
         CALL FETCHDATA('\CHAPT-7\LEGENDRE.DAT',TITLE)
         M = 5
      ELSEIF (PAE .EQ. 'E' ) THEN
         CALL FETCHDATA('\CHAPT-7\LEGENDRE.DAT',TITLE)
         M = 3
      ENDIF  !--- PAE
      CALL MAKEBETA                    !--- SET UP THE LINEAR BETA MATRIX
      CALL MAKEALPHA                   !--- SET UP THE SQUARE ALPHA MATRIX
      CALL MATINV(M, ALPHA, DET)       !--- INVERT ALPH TO GET EPSILON MATRIX
      CALL LINEARBYSQUARE(M,BETA,ALPHA,A) !--- BETA X EPS = PARAMETER MATRIX
      CALL CALCULATEY
      CHI2 = CALCCHISQ()
      DO 100 I = 1, M
         SIGA(I) = SQRT(ALPHA(I,I))
  100 CONTINUE
      CALL OUTPUT(.TRUE., 'CON', CHI2, TITLE) !--- TRUE TO PRINT ERROR MATRIX
      IF (M .GT. 2 ) THEN
         SPL = .TRUE.                  !--- PLOT A CURVE
      ELSE
         SPL = .FALSE.                 !--- PLOT A LINE
      ENDIF
      IF (PAE .EQ. 'P') THEN
         CALL PLOTIT('MULTREGR.SCR', .FALSE., SPL,   !--- FILE,LOG?,SPLINE
     1   'C', (X(2)-X(1))/12,      !--- DATA CIRCLES, RADIUS OF DATA CIR
     2   -10., -2., 110., 4.,      !--- X1,Y1, X2,Y2
     3    6, 6,                    !--- X,Y GRID MARKS
     4   'TEMPERATURE (DEGREES CELSIUS)','VOLTAGE (MV)')
      ELSE IF ((PAE .EQ. 'A') .OR. (PAE .EQ. 'E')) THEN
         CALL PLOTIT('MULTREGR.SCR', .FALSE., .TRUE.,
     1   'C', (X(2)-X(1))/10,   0., 0.,180., 1500., 6, 6,
     2   'THETA(DEGREES)', 'NUMBER OF COUNTS')
      ENDIF                                !--- PAE
      CALL CLOSEGRAPHICS
      END
      INCLUDE '\CHAPT-7\FITFUNC7.FOR'
      INCLUDE '\CHAPT-7\MAKEAB7.FOR'
      INCLUDE '\CHAPT-6\FITUTIL.FOR'
      INCLUDE '\APPEND-B\MATRIX.FOR'

C PROGRAM 7.2: \CHAPT-7\FITFUNC7.FOR
C FITTING FUNCTIONS FOR CHAPTER 7 EXAMPLES.
```



```
      REAL FUNCTION POWERFUNC(K, XX)
      INTEGER K
      REAL   XX
      REAL   YY
      INTEGER I
      YY = 1
      IF (K .GT. 1 ) THEN
         DO 100 I= 2, K
         YY = XX * YY
  100 CONTINUE
      ENDIF
      POWERFUNC = YY
      RETURN
      END

      REAL FUNCTION LEGFUNC(K, XX)
C DEFINE SEPARATE TERMS IN A SERIES, Y = A0*L0(X) + A1*L1(X) + ..
C NOTE K = 1 CORRESPONDS TO ZEROTH ORDER.
C VAR   PAE : CHAR   'P'-POWER SERIES,
C 'A'-ALL LEGENDRE TERMS TO ORDER M,
C 'E'-EVEN LEGENDRE TERMS)
C
      COMMON /FITVARS7/PAE
      CHARACTER *1 PAE
      INTEGER K
      REAL   XX
      INTEGER KK, I
      REAL C, PI/3.14159/, LEGPOLY(11) !--- I.E., 0TH THRU 10TH ORDER
      IF (PAE .EQ. 'E') KK = 2*K - 1
      IF (PAE .EQ.'A')  KK = K
      C = COS(PI*XX/180)
      LEGPOLY(1) = 1 !--- FOR BETTER EFFICIENCY, COULD CALC ONCE AND SAVE
      IF (KK .GT. 1 ) THEN
         LEGPOLY(2) = C
         IF (KK .GT. 2 ) THEN
         DO 100 I = 3, KK
           LEGPOLY(I)=((2*I-1)*C*LEGPOLY(I-1)-(I-1)*LEGPOLY(I-2))/I
  100    CONTINUE
      ENDIF            !--- KK > 2
      ENDIF              !--- KK > 1
      LEGFUNC = LEGPOLY(KK)
      RETURN
      END

      REAL FUNCTION FUNCT(K, XX)
      INTEGER K
      REAL XX
      REAL LEGFUNC, POWERFUNC
      COMMON /FITVARS7/PAE
      CHARACTER * 1 PAE
```

```
      IF ((PAE .EQ. 'A') .OR. (PAE.EQ.'E')) FUNCT = LEGFUNC(K,XX)
      IF  (PAE .EQ. 'P')  FUNCT = POWERFUNC(K,XX)
      RETURN
      END

      SUBROUTINE CALCULATEY
      INTEGER  I, K
      REAL     YY, FUNCT
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      DO 100 I=1, NPTS
         YY = 0
         DO 200 K = 1, M
            YY = YY + A(K) * FUNCT(K,X(I))
200      CONTINUE
         YCALC(I) = YY
100   CONTINUE
      RETURN
      END

      REAL FUNCTION CALCCHISQ() !--- ASSUMES ARRAY YCALC HAS BEEN FILLED
      INTEGER I
      REAL    CHI2
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      CHI2=0.
      DO 100 I = 1, NPTS
         CHI2 = CHI2 + ( (Y(I)-YCALC(I)) / SIGY(I))**2
100   CONTINUE
      CALCCHISQ = CHI2
      RETURN
      END

C PROGRAM 7.3: \CHAPT-7\MAKEAB7.FOR
C ROUTINES TO SET UP THE BETA AND ALPHA MATRICES FOR LINEAR REGRESSION
C USES  MATRIX, FITFUNC7
C
      SUBROUTINE MAKEBETA             !--- MAKE THE BETA MATRICES
      INTEGER  I, K
      REAL FUNCT
      INCLUDE 'C:\CHAPT-6\FITVARS.FOR'
      DO 100  K=1, M
         BETA(K)=0
         DO 200 I=1, NPTS
         BETA(K)=BETA(K) + Y(I)*FUNCT(K, X(I))/SIGY(I)**2
200   CONTINUE
100   CONTINUE
      RETURN
      END

      SUBROUTINE MAKEALPHA            !--- MAKE THE ALPHA MATRICES
      INTEGER I,J,K
      REAL FUNCT
```



```
      INCLUDE 'C:\CHAPT-6\FITVARS.FOR'
      DO 100 J=1, M
         DO 200 K=1, M
            ALPHA(J,K)=0
               DO 300 I=1, NPTS
                  ALPHA(J,K) = ALPHA(J,K)+FUNCT(J, X(I))*FUNCT(K, X(I))/SIGY(I)**2
               300 CONTINUE
         200   CONTINUE
      100   CONTINUE
      RETURN
      END
```

E.4 Routines from Chapter 8

```
C PROGRAM 8.0: \CHAPT-8\NONLINFT.FOR
C MAIN CALLING ROUTINE FOR NON-LINEAR FITTING METHODS
C USES GRIDSEAR, GRADSEAR, EXPNDFIT, MARQFIT, FITFUNC8, MAKEAB8,
C      NUMDERIV, MATRIX, FITUTIL
       PROGRAM NONLINFT
       INTEGER  TRIAL, J, METHOD
       REAL STEPDOWN, LAMBDA, CHISQR, CALCCHISQ
       CHARACTER*40 TITLE
       REAL STEPSCALE(4)/0.49999, 0.99999, 0.001, 0.001/
       INCLUDE '\CHAPT-6\FITVARS.FOR'
       PRINT *,'    (1)GRID SEARCH,    (2)GRADIENT SEARCH'
       PRINT *,'    (3)CHISQ EXPANSION, (4)FUNCTION EXPANSION'
       PRINT *, 'TYPE 1, 2, 3, OR 4 --- '
       READ *, METHOD
       CHICUT   = 0.01
       STEPDOWN  = 0.1       !--- STEP DOWN THE GRADIENT IN GRADLS
       LAMBDA   = 0.001      !--- FOR MARQUARDT METHOD ONLY
       STEPSIZE  = STEPSCALE(METHOD)  !--- SCALES DELTAA(J)
       CALL FETCHDATA('\CHAPT-8\RADIODK.HST',TITLE)
       CALL FETCHPARAMETERS     !--- USES NPTS, MUST FOLLOW FETCHDATA
       TRIAL    = 0
       CHISQR  = CALCCHISQ()
       CHIOLD = CHISQR + CHICUT + 1
       DO WHILE (ABS(CHIOLD - CHISQR) .GE. CHICUT)
          CHIOLD = CHISQR
          PRINT 1000, TRIAL, CHISQR
1000 FORMAT(' TRIAL #', I4, ' CHISQ =', F10.1)
          PRINT 1100, (A(J), J = 1,M)
1100 FORMAT(6F12.4)
          PRINT *
          GOTO (110, 120, 130, 140) , METHOD
110       CALL GRIDLS(CHISQR)
          GOTO 150
120       CALL GRADLS(CHISQR, STEPDOWN)
          GOTO 150
130       CALL CHIFIT(CHISQR)
          GOTO 150
```

```
      IF ((PAE .EQ. 'A') .OR. (PAE.EQ.'E')) FUNCT = LEGFUNC(K,XX)
      IF  (PAE .EQ. 'P')  FUNCT = POWERFUNC(K,XX)
      RETURN
      END

      SUBROUTINE CALCULATEY
      INTEGER  I, K
      REAL     YY, FUNCT
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      DO 100 I=1, NPTS
         YY = 0
         DO 200 K = 1, M
            YY = YY + A(K) * FUNCT(K,X(I))
200      CONTINUE
         YCALC(I) = YY
100   CONTINUE
      RETURN
      END

      REAL FUNCTION CALCCHISQ() !--- ASSUMES ARRAY YCALC HAS BEEN FILLED
      INTEGER I
      REAL    CHI2
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      CHI2=0.
      DO 100 I = 1, NPTS
         CHI2 = CHI2 + ( (Y(I)-YCALC(I)) / SIGY(I))**2
100   CONTINUE
      CALCCHISQ = CHI2
      RETURN
      END

C PROGRAM 7.3: \CHAPT-7\MAKEAB7.FOR
C ROUTINES TO SET UP THE BETA AND ALPHA MATRICES FOR LINEAR REGRESSION
C USES  MATRIX, FITFUNC7
C
      SUBROUTINE MAKEBETA            !--- MAKE THE BETA MATRICES
      INTEGER  I, K
      REAL FUNCT
      INCLUDE 'C:\CHAPT-6\FITVARS.FOR'
      DO 100  K=1, M
         BETA(K)=0
         DO 200 I=1, NPTS
         BETA(K)=BETA(K) + Y(I)*FUNCT(K, X(I))/SIGY(I)**2
200   CONTINUE
100   CONTINUE
      RETURN
      END

      SUBROUTINE MAKEALPHA           !--- MAKE THE ALPHA MATRICES
      INTEGER I,J,K
      REAL FUNCT
```



```
      INCLUDE 'C:\CHAPT-6\FITVARS.FOR'
      DO 100 J=1, M
         DO 200 K=1, M
            ALPHA(J,K)=0
               DO 300 I=1, NPTS
                  ALPHA(J,K) = ALPHA(J,K)+FUNCT(J, X(I))*FUNCT(K, X(I))/SIGY(I)**2
               300 CONTINUE
         200   CONTINUE
      100   CONTINUE
      RETURN
      END
```

E.4 Routines from Chapter 8

```
C PROGRAM 8.0: \CHAPT-8\NONLINFT.FOR
C MAIN CALLING ROUTINE FOR NON-LINEAR FITTING METHODS
C USES GRIDSEAR, GRADSEAR, EXPNDFIT, MARQFIT, FITFUNC8, MAKEAB8,
C      NUMDERIV, MATRIX, FITUTIL
       PROGRAM NONLINFT
       INTEGER  TRIAL, J, METHOD
       REAL STEPDOWN, LAMBDA, CHISQR, CALCCHISQ
       CHARACTER*40 TITLE
       REAL STEPSCALE(4)/0.49999, 0.99999, 0.001, 0.001/
       INCLUDE '\CHAPT-6\FITVARS.FOR'
       PRINT *,'    (1)GRID SEARCH,     (2)GRADIENT SEARCH'
       PRINT *,'    (3)CHISQ EXPANSION, (4)FUNCTION EXPANSION'
       PRINT *, 'TYPE 1, 2, 3, OR 4  --- '
       READ *, METHOD
       CHICUT   = 0.01
       STEPDOWN  = 0.1       !--- STEP DOWN THE GRADIENT IN GRADLS
       LAMBDA    = 0.001     !--- FOR MARQUARDT METHOD ONLY
       STEPSIZE  = STEPSCALE(METHOD)  !--- SCALES DELTAA(J)
       CALL FETCHDATA('\CHAPT-8\RADIODK.HST',TITLE)
       CALL FETCHPARAMETERS      !--- USES NPTS, MUST FOLLOW FETCHDATA
       TRIAL    = 0
       CHISQR  = CALCCHISQ()
       CHIOLD = CHISQR + CHICUT + 1
       DO WHILE (ABS(CHIOLD - CHISQR) .GE. CHICUT)
          CHIOLD = CHISQR
          PRINT 1000, TRIAL, CHISQR
1000 FORMAT(' TRIAL #', I4, ' CHISQ =', F10.1)
          PRINT 1100, (A(J), J = 1,M)
1100 FORMAT(6F12.4)
          PRINT *
          GOTO (110, 120, 130, 140) , METHOD
110       CALL GRIDLS(CHISQR)
          GOTO 150
120       CALL GRADLS(CHISQR, STEPDOWN)
          GOTO 150
130       CALL CHIFIT(CHISQR)
          GOTO 150
```

```
140     CALL MARQUARDT(CHISQR, CHICUT, LAMBDA)
150     TRIAL = TRIAL +1
      ENDDO
151   CALL CALCULATEY
      IF ((METHOD .EQ. 1) .OR. (METHOD .EQ. 2)) THEN
         DO 200 J = 1, M
            SIGA(J) = SIGPARAB(J)                    !--- DCHI2 = 1
200      CONTINUE
         CALL OUTPUT(.FALSE., 'CON' ,CHISQR, TITLE) !--- NO ERROR MATRIX
      ELSEIF ((METHOD .EQ. 3) .OR. (METHOD .EQ. 4)) THEN
         IF (METHOD .EQ. 4 ) THEN
            CALL MARQUARDT(CHISQR,CHICUT,0)          !--- GET ERROR MATRI
         ENDIF
         DO 300 J = 1, M
            SIGA(J) = SIGMATRX(J)                    !--- ERROR MATRIX
300      CONTINUE
         CALL OUTPUT(.TRUE., 'CON', CHISQR, TITLE) !--- WITH ERROR MATRIX
      ENDIF
      CALL PLOTIT('NONLIN.SCR', .TRUE., .TRUE., !--- SCRPT FILE, LOG?, SPLINE?
     1  'C', (X(2)-X(1))/5,     !--- DATA CIRCLES, RADIUS OF CIRCLES
     2  0., 1., 900., 1000.,    !--- RANGES-X1,Y1,X2,Y2
     3  6, 6,                   !--- NUM X-AXIS DIV, NUM Y-AXIS DIV
     4  'TIME (SEC)', 'NUMBER OF COUNTS')  !--- AXIS LABELS
      CALL CLOSEGRAPHICS
      END

C SAMPLE FITTING FUNCTION FOR NON-LINEAR FITS
C EXAMPLE IS SUM OF 2 EXPONENTIALS ON A CONSTANT BACKGROUND
      REAL FUNCTION  EXPF(A,X)
      REAL A,X
      REAL YY, ARG
      ARG = ABS(X/A)
      IF (ARG .GT. 60 ) THEN
         YY = 0
      ELSE
         YY = EXP(-ARG)
      ENDIF
      EXPF = YY
      RETURN
      END

      FUNCTION  YFUNCTION(XX) !--- REAL
      REAL YFUNCTION, XX, EXPF
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      YFUNCTION = A(1) + A(2)*EXPF(A(4),XX) + A(3)*EXPF(A(5),XX)
      RETURN
      END

      INCLUDE '\CHAPT-8\GRIDSEAR.FOR'    !--- 1-GRID SEARCH METHOD
      INCLUDE '\CHAPT-8\GRADSEAR.FOR'    !--- 2-GRADIENT SEARCH METHOD
      INCLUDE '\CHAPT-8\EXPNDFIT.FOR'    !--- 3-FUNCTION EXPANSION METHOD
```



```
      INCLUDE '\CHAPT-8\MARQFIT.FOR'  !--- 4-MARQUARDT METHOD
      INCLUDE '\CHAPT-6\FITUTIL.FOR'
      INCLUDE '\CHAPT-8\FITFUNC8.FOR'    !--- USED BY ALL METHODS
      INCLUDE '\CHAPT-8\MAKEAB8.FOR'     !--- USED BY METHODS 4 AND 5
      INCLUDE '\CHAPT-8\NUMDERIV.FOR'    !--- USED BY METHODS 4 AND 5
      INCLUDE '\APPEND-B\MATRIX.FOR'     !--- USED BY METHODS 4 AND 5

C  PROGRAM 8.1: \CHAPT-8\GRIDSEAR.FOR
C NON-LINEAR FIT BY THE GRID-SEARCH METHOD
C USES FITFUNC8, FITUTIL
      SUBROUTINE GRIDLS(CHISQR)
      REAL CHISQR
      REAL CALCCHISQ
      REAL SAVE,DELTA, DELTA1,DEL1,DEL2,AA,BB,CC,DISC,ALPH,X1,X2
      INTEGER J
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      CHISQ2 = CALCCHISQ()
C  -FIND LOCAL MINIMUM FOR EACH PARAMETER-
      DO 100 J = 1, M
         DELTA   = DELTAA(J)
         A(J)    = A(J) + DELTA
         CHISQ3  = CALCCHISQ()
         IF (CHISQ3 .GT. CHISQ2 ) THEN
            DELTA  = -DELTA            !--- STARTED IN WRONG DIRECTION
            A(J)   = A(J) + DELTA
            SAVE   = CHISQ2            !--- INTERCHANGE 2 AND 3 SO 3 IS LOWER
            CHISQ2 =  CHISQ3
            CHISQ3 =  SAVE
         ENDIF                         !--- IF (CHISQ3 ...
C  -INCREMENT OR DECREMENT A(J) UNTIL CHI SQUARED INCREASES-
  110 CONTINUE
      CHISQ1 = CHISQ2                  !--- MOVE BACK TO PREPARE FOR QUAD FIT
      CHISQ2 = CHISQ3
      A(J)   = A(J) + DELTA
      CHISQ3 = CALCCHISQ()
      IF (CHISQ3 .LE.  CHISQ2) GOTO 110
C  -FIND MINIMUM OF PARABOLA DEFINED BY LAST THREE POINTS-
      DEL1 = CHISQ2 - CHISQ1
      DEL2 = CHISQ3 - 2*CHISQ2 + CHISQ1
      DELTA1 = DELTA * (DEL1/DEL2 + 1.5)
      A(J) = A(J)  - DELTA1
      CHISQ2 = CALCCHISQ()             !--- AT NEW LOCAL MINIMUM
C  -ADJUST DELTA FOR CHANGE OF 2 FROM CHISQ AT MINIMUM-
      AA = DEL2/2                      !--- CHISQ = AA*A(J)**2 + BB*A(J) + CC
      BB = DEL1 - DEL2/2
      CC = CHISQ1-CHISQ2
      DISC = BB**2 -4*AA*(CC-2)        !--- CHISQR DIFF(ERENCE) = 2
      IF (DISC .GT. 0 ) THEN           !--- IF NOT, THEN PROBABLY NOT PARABOLIC YET
         DISC =SQRT(DISC)
         ALPH = (-BB - DISC)/(2*AA)
```

```
      X1 = ALPH*DELTA +  A(1) - 2*DELTA   !--- A(J) AT CHISQ MINIMUM+2
      DISC = BB**2 - 4*AA*CC
      IF (DISC.GT.0 ) THEN
         DISC = SQRT(DISC)
      ELSE
         DISC = 0                         !--- ELIM ROUNDING ERR
      ENDIF
      ALPH = (-BB - DISC)/(2*AA)
      X2 = ALPH*DELTA + A(1) - 2*DELTA    !--- A(J) AT CHISQ MINIMUM
      DELTA = X1 - X2
      DELTAA(J) = DELTA
    ENDIF                                 !--- IF (DISC .GT. 0 ...
  100 CONTINUE                            !--- DO J
    CHISQR = CHISQ2
    RETURN
    END

C PROGRAM 8.2: \CHAPT-8\GRADSEAR.FOR
C NON-LINEAR LEAST-SQUARES FIT BY GRADIENT SEARCH METHOD
C USES FITFUNC8, FITUTIL
      SUBROUTINE CALCGRAD
      INTEGER J
      REAL  SUM, DELTA, FRACT/0.001/, CALCCHISQ
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      SUM = 0
      DO 100 J = 1, M
         CHISQ2 = CALCCHISQ()
         DELTA  = FRACT * DELTAA(J) !--- DIFF(ERENTIAL ELEMENT FOR GRADENT
         A(J)   = A(J) + DELTA
         CHISQ1  = CALCCHISQ()
         A(J)   = A(J) - DELTA
         GRAD(J) = CHISQ2 - CHISQ1          !--- 2*DELTA*GRAD
         SUM    = SUM + GRAD(J)**2
  100 CONTINUE
      DO 200 J = 1, M
         GRAD(J) =  DELTAA(J)*GRAD(J)/SQRT(SUM)    !--- STEP * GRAD
  200 CONTINUE
      RETURN
      END

      SUBROUTINE GRADLS(CHISQR, STEPDOWN)
      REAL CHISQR, STEPDOWN
      REAL STEPSUM, STEP1, CALCCHISQ
      INTEGER J
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      CALL CALCGRAD             !--- CALCULATE THE GRADIENT
C -EVALUATE CHISQR AT NEW POINT AND MAKE SURE CHISQR DECREASES-
      CHISQ3 = CHISQ2 + 1
      DO WHILE (CHISQ3 .GT. CHISQ2)
         DO J = 1, M
            A(J) = A(J) + STEPDOWN * GRAD(J) ! SLIDE DOWN
```



```
         ENDDO
         CHISQ3 = CALCCHISQ()
         IF (CHISQ3 .GE. CHISQ2 ) THEN
            DO J = 1, M      ! MUST HAVE OVERSHOT MINIMUM
               A(J) = A(J) - STEPDOWN * GRAD(J) ! RESTORE
            ENDDO
            STEPDOWN = STEPDOWN/2          ! DECREASE STEPSIZE
         ENDIF
      ENDDO
      STEPSUM = 0
C -INCREMENT PARAMETERS UNTIL CHISQR STARTS TO INCREASE-
      DO WHILE (CHISQ3 .LT. CHISQ2)
         STEPSUM = STEPSUM + STEPDOWN   ! COUNTS TOTAL INCREMENT
         CHISQ1 = CHISQ2
         CHISQ2 = CHISQ3
         DO J = 1, M
            A(J) = A(J) + STEPDOWN * GRAD(J)
         ENDDO
         CHISQ3 = CALCCHISQ()
      ENDDO              !DOWHILE
C -FIND MINIMUM OF PARABOLA DEFINED BY LAST THREE POINTS-
      STEP1=STEPDOWN*((CHISQ3-CHISQ2)/(CHISQ1-2*CHISQ2+CHISQ3)+0.5)
      DO J = 1, M
         A(J) = A(J) - STEP1 * GRAD(J)    ! MOVE TO MINIMUM
      ENDDO
      CHISQR = CALCCHISQ()
      STEPDOWN = STEPSUM              ! START WITH THIS NEXT TIME
      RETURN
      END

C PROGRAM 8.3: \CHAPT-8\EXPNDFIT.FOR
C NON-LINEAR LEAST-SQUARES FIT BY EXPANSION OF THE FITTING FUNCTION
C USES  FITFUNC8, MAKEABB, MATRIX
      SUBROUTINE CHIFIT(CHISQR)
      INTEGER J
      REAL DET, CALCCHISQ
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      CALL MAKEBETA
      CALL MAKEALPHA
      CALL MATINV(M, ALPHA, DET)      !--- INVERT MATRIX
      CALL LINEARBYSQUARE(M,BETA,ALPHA,DA)  !--- EVALULATE PARAM
INCREMENTS
      DO 100 J = 1, M
         A(J) = A(J) + DA(J)              !--- INCREMENT TO NEXT SOLUTION.
  100 CONTINUE
      PRINT *,'A',(A(J),J=1,M)
      CHISQR =  CALCCHISQ()
      RETURN
      END
```

```
C PROGRAM 8.4: \CHAPT-8\MARQFIT.FOR
C NON-LINEAR FIT BY THE GRADIENT-EXPANSION (MARQUARDT) METHOD
C USES  FITFUNC9, MAKEAB8, MATRIX
      SUBROUTINE MARQUARDT(CHISQR, XICUT, LAMBDA)
      INTEGER J
      REAL CHISQR, XICUT, LAMBDA
      REAL DET, CALCCHISQ
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      DO
         CALL MAKEBETA
         CALL MAKEALPHA
         DO 100 J = 1, M
            ALPHA(J,J) = (1 + LAMBDA) * ALPHA(J,J)
  100    CONTINUE
         CALL MATINV(M, ALPHA, DET)          !--- INVERT MATRIX
         IF (LAMBDA .LE. 0 ) RETURN !--- FINAL CALL TO GET THE ERROR MATRIX.
         CALL LINEARBYSQUARE(M,BETA,ALPHA,DA)!--- EVAL PARAM INCREMENTS
         CHISQ1 = CHISQR
         DO 200 J = 1, M
            A(J) = A(J) + DA(J)              !--- INCR TO NEXT SOLUTION
  200    CONTINUE
         CHISQR = CALCCHISQ()
         IF ( CHISQR .LE. CHISQ1 + XICUT ) RETURN
         DO 300 J = 1, M
            A(J) = A(J)-DA(J)     !--- RETURN TO PREV SOLUTION
  300    CONTINUE
         CHISQR = CALCCHISQ()
         LAMBDA = 10*LAMBDA  !--- AND REPEAT THE CALC, WITH LARGER LAMBDA
      END DO
      END

C PROGRAM 8.5:  \CHAPT-8\FITFUNC8.FOR
C USES FITVARS
C -THE FOLLOWING ROUTINES ARE GENERAL FOR FITTING ANY FUNCTION-
      SUBROUTINE CALCULATEY
      REAL YFUNCTION
      INCLUDE '\CHAPT-6/FITVARS.FOR'
      DO 100 I = 1, NPTS
         YCALC(I) = YFUNCTION(X(I))
  100 CONTINUE
      RETURN
      END

      REAL FUNCTION CALCCHISQ()
      REAL  CHI2, YFUNCTION
      INCLUDE '\CHAPT-6/FITVARS.FOR'
      CHI2=0.
      DO 100 I = 1, NPTS
         CHI2 = CHI2 + ( (Y(I)-YFUNCTION(X(I)))/SIGY(I))**2
  100 CONTINUE
      CALCCHISQ = CHI2
```



```
      RETURN
      END

C -STANDARD DEVIATION CALC'D FROM CHISQ CHANGE OF 1 (PARABOLA FIT)
      REAL FUNCTION  SIGPARAB(J)
      INTEGER J
      REAL CALCCHISQ
      INCLUDE '\CHAPT-6/FITVARS.FOR'
      CHISQ2 = CALCCHISQ()
      A(J)   = A(J) + DELTAA(J)
      CHISQ3 = CALCCHISQ()
      A(J)   = A(J) - 2*DELTAA(J)
      CHISQ1 = CALCCHISQ()
      A(J)   = A(J) + DELTAA(J)
      SIGPARAB = DELTAA(J)*SQRT(2/(CHISQ1-2*CHISQ2+CHISQ3))
      RETURN
      END

C -STANDARD DEVIATION CALC'D FROM DIAGONAL TERMS IN ERROR MATRIX
      REAL FUNCTION SIGMATRX(J)
      INTEGER J
      REAL SIG
      INCLUDE '\CHAPT-6/FITVARS.FOR'
      SIG = SQRT(ABS(ALPHA(J,J)))
      IF (ALPHA(J,J) .LT. 0 ) SIG = - SIG  !--- NOTE- AN ERROR
      SIGMATRX = SIG
      RETURN
      END

C PROGRAM 8.6: \CHAPT-8\MAKEAB8.FOR
C MATRIX SET-UP FOR NON-LINEAR FITS
C USES FITFUNC8, NUMDERIV
C
      SUBROUTINE MAKEBETA    !---MAKE BETA MATRICES FOR NON-LINEAR FITTING
      INTEGER  J
      INCLUDE '\CHAPT-6/FITVARS.FOR'
      DO 100 J = 1, M
         BETA(J) = -0.5*DXISQ_DA(J)
100   CONTINUE
      RETURN
      END

      SUBROUTINE MAKEALPHA  !--- ALPHA MATRICES FOR NON-LINEAR FITTING
      INTEGER  J, K
      INCLUDE '\CHAPT-6\FITVARS.FOR'
      DO 100 J = 1, M
         ALPHA(J,J) =  0.5 * D2XISQ_DA2(J)
         IF (ALPHA(J,J) .EQ. 0 ) THEN
            PRINT *, 'DIAGONAL ELEMENT IS ZERO, J =',J
         STOP
         ENDIF
```

```
      IF (J .GT. I ) THEN
         DO 200 K = 1, J-1
            ALPHA(J,K) = 0.5*D2XISQ_DAJK(J,K)
            ALPHA(K,J) =  ALPHA(J,K)
200      CONTINUE              !--- DO K
      ENDIF                    !--- IF J
100   CONTINUE                 !---  DO J
      DO 300 J = 1, M
         IF (ALPHA(J,J) .LT. 0 ) THEN
         ALPHA(J,J) = -ALPHA(J,J)
         IF (J .GT. 1 ) THEN
            DO 400 K = 1, J-1
               ALPHA(J,K) = 0
               ALPHA(K,J) = 0
400         CONTINUE             !--- DO K
         ENDIF                   !--- IF J
      ENDIF                      !--- IF ALPHA
300   CONTINUE                   !--- FOR J
      RETURN
      END
```

E.5 Routines from Chapter 9

```
C PROGRAM 9.1: \CHAPT-9\LORENFIT.FOR
C MAIN CALLING ROUTINE FOR FIT TO LORENTZIAN + POLYNOMIAL
C USES  FITFUNC9, MARQFIT, MATRIX,  NUMDERIV, MAKEAB8,  FITUTIL
      PROGRAM LORENFIT
      CHARACTER*40  TITLE
      INTEGER TRIAL, J
      REAL   XSHIFT, CHISQR, LAMBDA, YFUNCTION
      REAL  STEPSCALE(4)/ 0.49999, 0.99999, 0.001, 0.001/
      INCLUDE 'C:\F\CHAPT-6\FITVARS.FOR'
      CHICUT    = 0.01
      LAMBDA    = 0.001              ! FOR MARQUARDT METHOD ONLY
      STEPSIZE  = STEPSCALE(4)       ! SCALES DELTAA[J]
      CALL FETCHDATA('\F\CHAPT-9\SINGLE.HST',TITLE)
      XSHIFT = (X(2)+ X(1))/2
      DO  J = 1, NPTS
         X(J) = X(J) + XSHIFT        ! MOVE TO BIN CENTER
      ENDDO
      CALL FETCHPARAMETERS           ! USES NPTS, MUST FOLLOW FETCHDATA
      TRIAL     = 0
      CHISQR    = CALCCHISQ()
      CHIOLD = CHISQR + CHICUT +1
      DO WHILE (ABS(CHIOLD - CHISQR) .GT. CHICUT)
         CHIOLD = CHISQR
         PRINT *,'TRIAL #',TRIAL,' CHISQ = ',CHISQR
         PRINT *, (A(J), J = 1, M)
         CALL MARQUARDT(CHISQR, CHICUT, LAMBDA)
         TRIAL  = 1 + TRIAL
```



```
      ENDDO
      CALL CALCULATEY
      CALL MARQUARDT(CHISQR,CHICUT,0)        ! GET ERROR MATRIX
      DO J = 1, M
         SIGA(J) = SIGMATRX(J)               ! ERROR MATRIX
      ENDDO
      CALL OUTPUT(.TRUE., 'CON', CHISQR,TITLE) ! WITH ERROR MATRIX
      DO J = 1, NPTS
         X(J) = X(J) - XSHIFT                ! RESTORE TO LEFT EDGE
      ENDDO
      CALL PLOTIT('LORENFIT.SCR',.FALSE.,.TRUE.,! SCRIPT FILE, LOG?, SPLINE?
     1  'H', 0.0,                  ! HIST, 0(NOT USED)
     2  0.0, 0.0, 3.0, 220.0,      ! X1, Y1, X2, Y2 FOR PLOT
     3  6, 6,                      ! NUM GRID MARKS X,Y
     4  'E (GEV)', 'NUMBER OF COUNTS')    ! LABELS
C -PLOT THE BACKGROUND-
      A(4) = 0.0
      A(7) = 0.0
      DO J = 1, NPTS
         YCALC(J) = YFUNCTION(X(J))
      ENDDO
      CALL SPLINEMAKE(NPTS,0,0,X,YCALC)
      CALL SCURVE(1, 40, 5, 0.025, X)    ! SPLINE CURVE
      CALL CLOSEGRAPHICS
      END

C LORENTZIAN PEAK ON A QUADRATIC BACKGROUND
      REAL FUNCTION YFUNCTION(XX) ! LORENTZIAN ON POLYNOMIAL
      REAL XX
      REAL YY, PI/3.1415927/
      INCLUDE '\F\CHAPT-6\FITVARS.FOR'
      YY = A(1) + A(2)*XX + A(3)*XX**2 + A(4)*A(6)/(2*PI)
     1  /((XX-A(5))**2 + A(6)**2/4)
      YFUNCTION = YY
      RETURN
      END
      INCLUDE '\F\CHAPT-6\FITUTIL.FOR'
      INCLUDE '\F\CHAPT-9\FITFUNC9.FOR'
      INCLUDE '\F\CHAPT-8\MARQFIT.FOR'       ! MARQUARDT METHOD
      INCLUDE '\F\CHAPT-8\MAKEAB8.FOR'       ! USED BY MARQFIT
      INCLUDE '\F\CHAPT-8\NUMDERIV.FOR'      ! USED BY MARQFIT
      INCLUDE '\F\APPEND-B\MATRIX.FOR'       ! USED BY MARQFIT
```

E.6 Routines from Chapter 10

```
C PROGRAM 10.1: \CHAPT-10\MAXLIKE.FOR
C DIRECT MAXIMUM LIKELIHOOD EXAMPLE
C  USES FITUTIL, QUIKSCRP
      PROGRAM MAXLIKE
      REAL    SIGTAU, TAUMAX, MAXM  !--- M IS LOG OF LIKELIHOOD FUNCTION
      INCLUDE '\CHAPT-10\MAXLINCL.FOR'
```

```
      CALL GETDATA('\CHAPT-10\TEST.DAT')      !--- WAS DA50
      CALL SEARCH(TAUMAX, MAXM)
      CALL WRITEOUTPUT(SIGTAU, TAUMAX, MAXM)
      CALL PLOTLIKECURVE(TAUMAX, SIGTAU, MAXM)
      CALL CLOSEGRAPHICS
      END

      SUBROUTINE GETDATA(INFILE)
      INTEGER   IEVNUM
      CHARACTER*(*) INFILE
      CHARACTER TITLE(80)
      INCLUDE '\CHAPT-10\MAXLINCL.FOR'
      C = 3.00
      LOSEARCH = 0.50
      HISEARCH = 1.5              !--- SEARCH RANGE
      TAUSTEP  = 0.01
      XLO      = 0.50             !--- PLOT RANGE
      XHI      = 1.2
      YLO      = 0.0
      YHI      = 1.2
      NTRIALS  = (HISEARCH - LOSEARCH)/TAUSTEP
      OPEN(5, INFILE)             !--- INPUT DATA FILE
      READ(5, *) TITLE
      PRINT *,' ',TITLE
      READ(5, *) NEVENTS, MASS, D1, D2
      IEVNUM = 1
      NEVENTS = 0
      DO WHILE (IEVNUM .GT.0)
         READ(5, *) IEVNUM, XPRODUCTION, PLAB, XDECAY
         IF (IEVNUM .GT.0) THEN
            IF ((XDECAY .GE. D1) .AND. (XDECAY .LT. D2)) THEN
               NEVENTS = 1 + NEVENTS
               LTOTSCALE =  MASS/(C*PLAB)  !--- = 1/(C*BETA*GAMMA)
               TIMES(NEVENTS)=(XDECAY - XPRODUCTION)*LTOTSCALE !---PROPER T
C  CONVERT D1 AND D2 TO TIME LIMITS, LOTLIM AND HITLIM,
C   I.E., INTEGRATION LIMITS IN PROPER TIME FROM THE PRODUCTION VERTEX.
               LOTLIM(NEVENTS) = (D1 - XPRODUCTION)*LTOTSCALE
               HITLIM(NEVENTS) = (D2 - XPRODUCTION)*LTOTSCALE
            ENDIF
         ENDIF
      ENDDO
      PRINT *, 'END OF FILE - ', IEVNUM, ' EVENTS READ'
      PAUSE
      RETURN
      END

      REAL FUNCTION LOGPROB(K, TAU)
      INTEGER K
      REAL TAU
      REAL A, B
      INCLUDE '\CHAPT-10\MAXLINCL.FOR'
```



```
C     D1 AND D2 ARE BEGINNING AND END OF THE FIDUCIAL REGION.
C     MUST CVT TO LOTLIM AND HITLIM WHICH ARE INTEGRATION LIMITS IN PROPER
      TIME,
C     MEASURED FROM PRODUCTION VERTEX.
C     NOW, CALC PROBABILITY-
      B = EXP(-HITLIM(K)/TAU)
      A = EXP(-LOTLIM(K)/TAU)
      PROB = EXP(-TIMES(K)/TAU)/(TAU*(A - B))
      LOGPROB = ALOG(PROB)
      RETURN
      END

      REAL FUNCTION LOGLIKE(T)
      REAL T, LOGPROB
      INTEGER I
      REAL    M, PROB
      INCLUDE '\CHAPT-10\MAXLINCL.FOR'
      M = 0.0
      DO 100 I = 1, NEVENTS
         PROB = LOGPROB(I,T)
         M = PROB + M
  100 CONTINUE
      LOGLIKE = M
      RETURN
      END

      SUBROUTINE SEARCH(TAUATMAX, MAXM)
      REAL TAUATMAX, MAXM
      INTEGER TRIAL
      REAL M1, M2, M3, DEL1, DEL2, DELTA1, TAU, MLIKELI, LOGLIKE
      INCLUDE '\CHAPT-10\MAXLINCL.FOR'
      M2   = -1000
      MAXM  = -1.0E20
      TAU  =  LOSEARCH
      DO 100 TRIAL = 0, NTRIALS
         MLIKELI = LOGLIKE(TAU)
         PRINT *,'TRIAL',TRIAL,' TAU=', TAU,' LOG LIKELIHOOD=',MLIKELI
         M3 = MLIKELI
         IF (M3 .GT. M2 ) THEN   !--- REMEMBER, THESE ARE NEGATIVE
            M1 = M2
            M2 = M3
         ELSE                    !--- LEAVING MAXIMUM
C     FIND MAXIMUM OF PARABOLA DEFINED BY LAST THREE POINTS-
            DEL1    = M2 - M1
            DEL2    = M3 - 2*M2 + M1
            DELTA1  = TAUSTEP * (DEL1/DEL2 + 1.5)
            TAU     = TAU  - DELTA1
            TAUATMAX = TAU
            MAXM    = LOGLIKE(TAU)   !--- AT MAXIMUM OF PARABOLA
            RETURN
         ENDIF
```

```
        TAU = TAU + TAUSTEP
100   CONTINUE
      RETURN
      END

      REAL FUNCTION ERROR(T, DT)   !--- 1/SQRT(-2ND DERIVATIVE OF LOG(L))
      REAL T, DT
      REAL  T1, T2, D2YDT2, ERR, LOGLIKE
      T1 = T - DT
      T2 = T + DT
      D2YDT2 = (LOGLIKE(T2) - 2*LOGLIKE(T) + LOGLIKE(T1))/DT**2
      ERR    = 1/SQRT(-D2YDT2)
      ERROR  = ERR
      RETURN
      END

C PROGRAM 10.2 \CHAPT-10\MAXLINCL.FOR     (WEBSITE)
C INCLUDE FILE FOR MAXLIKE
```

E.7 Routines from Chapter 11

```
C PROGRAM 11.1: \CHAPT-11\CHI2PROB.FOR
C CALCULATE CHI^2 PROB. DENS. & THE CHI^2 PROB. INTEGRAL
C USES CHIPROBDENS AND CHIPROB
      PROGRAM CHI2PROB
      REAL CHI2,  CHIPROB
      INTEGER NFREE
      PRINT *,'CALCULATE CHI2 PROBABILITY DENSITY FUNCTION & INTEGRAL',
     1  ' PROBABILITY'
      PRINT *, 'TYPE  NUM DEG OF FREEDOM AND  CHI2.  (EXIT ON ^C)'
      READ *,  NFREE, CHI2
      PRINT 1000, CHIPROBDENS(CHI2, NFREE), CHIPROB(NFREE, CHI2)
 1000 FORMAT(' CHI^2 PROB. DENS. = ',F7.3,' , CHI^2 PROBABILITY=',F7.3)
      PRINT *,' ***** NOTE THAT TABLE C.4 REFERS TO CHI^2/NFREE****'
      END

C THE FOLLOWING THREE ROUTINES ARE INCLUDED
C   IN THE PROGRAM UNIT  C:\CHAPT-6\FITUTIL.FOR (WEBSITE)
      REAL FUNCTION CHIPROB(NFREE, CHI2)       !--- MAX NFREE =  56
      EXTERNAL CHIX
      COMMON/UTIL/ GLSIMPS
      REAL CHIX, SIMPSON, GLSIMPS
      INTEGER NFREE
      REAL PI, CHI2, CLIM, INTFROMLIM
      DATA   CLIM /2/,                 !--- EXPANSION LIMIT FOR NFREE = 1
     1  INTFROMLIM /0.157/,            !--- INTEGRAL FROM CLIM TO INFINITY
     2     DX0 /0.2/                   !--- DETERMINES ACCURACY OF INTEGRATION
     3     PI/3.14159/
      INTEGER NINT
      IF (CHI2 .GE. 1) THEN
         NINT = (CHI2+0.0001)/DX0
      ELSE
```



```
         NINT = 5
      ENDIF
      IF (CHI2 .GT. 15*SQRT(NFREE) ) THEN !--- QUICK CUTOUT
         CHIPROB = 0
      ELSE
         GLSIMPS = FLOAT(NFREE)/2          !--- GLSIMPS IS GLOBAL FOR CHIX
        IF (NFREE .EQ. 1 )   THEN
           IF (CHI2 .LT. CLIM ) THEN
          CHIPROB = 1-SQRT(CHI2/2/PI)*
     1    (2 - CHI2*(1/3 - CHI2*(1/20 - CHI2*(1/168 - CHI2/1728))))
           ELSE
             CHIPROB = INTFROMLIM - SIMPSON(CHIX,NINT,CLIM,CHI2)
     1                 /GAMMA(NFREE/2.0)/2.0**(NFREE/2.0)
           ENDIF                            !--- IF (CHI2 ...)
        ELSE IF (NFREE .EQ. 2 ) THEN
           CHIPROB = EXP(-CHI2/2)           !--- INTEGRABLE
        ELSE
           CHIPROB =  1 - SIMPSON(CHIX, NINT, 0, CHI2)
     1             /GAMMA(NFREE/2.0)/2.0**(NFREE/2.0)
        ENDIF                               !--- IF (NFREE ...)
      ENDIF
      RETURN                                !--- IF (NFREE ...)
      END

      REAL FUNCTION CHIPROBDENS(X,NFREE)
      REAL NUM, DEN, H, X
      INTEGER NFREE
      H = NFREE/2.0
      NUM = X**(H-1) * EXP(-X/2)
      DEN =  2**H * GAMMA(H)
      CHIPROBDENS = NUM/DEN
      RETURN
      END

C  USED BY CHIPROB (FOR SIMPSON WHICH ALLOWS ONLY 1 ARGUMENT.)
      REAL FUNCTION CHIX(X)
      COMMON/UTIL/ GLSIMPS
      REAL GLSIMPS
      REAL X
      IF (X.EQ.0) THEN
         CHIX = 0.0
      ELSE
         CHIX = X**(GLSIMPS-1)*EXP(-X/2)  !--- GLSIMPS = H = NFREE/2
      ENDIF
      RETURN
      END

C THIS FOLLOWING ROUTINE IS INCLUDED
C   IN THE PROGRAM UNIT \CHAPT-6\FITUTIL.FOR   (WEBSITE)
C APPROXIMATE GAMMA FUNCTION WITH H = NFREE/2
      REAL FUNCTION GAMMA(H)
```

```
      REAL H, PI/3.1415927/
      GAMMA = SQRT(2.0*PI) * EXP(-H)*(H**(H-0.5)) * (1.0 + 0.0833/H)
      RETURN
      END

C PROGRAM 11.2: \CHAPT-11\LCORPROB.FOR
C CALCULATE LINEAR CORRELATION PROBABILITY INTEGRAL
C USES LCORLATE
      PROGRAM LCORPROB
      INTEGER NOBSERV
      REAL LINCORPROB, RCORR
      PRINT *, 'TEST INTEGRAL OF LINEAR CORRELATION FUNCTION'
      PRINT *, 'TYPE-# OBSERVATIONS, LINEAR CORRELATION COEFFICIENT: '
      READ *, NOBSERV, RCORR
      PRINT *, 'INTEGRAL CORRELATION FUNCTION= ',
     1 LINCORPROB(NOBSERV-2, RCORR)
      END
      INCLUDE '\CHAPT-11\LCORLATE.FOR'

C LINEAR-CORRELATION PROBABILITY FUNCTION AND INTEGRAL
C USES FITUTIL
      REAL FUNCTION LINCORPROB(NFREE, HILIM)
      EXTERNAL LINCORREL    !--- FOR USE IN FUNCTION SIMPSON
      INTEGER NFREE
      REAL   HILIM
      REAL DX /0.01/, LOLIM/0.0/, LINCORREL, SIMPSON
      INTEGER NINT
      COMMON/UTIL/GLSIMPS
      GLSIMPS = NFREE !--- GLOBAL FOR FUNCTION LINCORREL (FOR SIMPSON)
      NINT  = INT((HILIM - LOLIM)/DX)
      LINCORPROB = 1-2*SIMPSON(LINCORREL, NINT, LOLIM,  HILIM)
      RETURN
      END

      REAL FUNCTION LINCORREL(R)
      REAL R
      COMMON/UTIL/GLSIMPS      !--- GLSIMS = NFREE MUST BE GLOBAL FOR
      DATA SQRTPI/1.7724539/   !   FUNCT "SIMPSONS" WHICH ALLOWS ONLY 1 ARG
      LINCORREL = GAMMA((GLSIMPS+1)/2)/GAMMA(GLSIMPS/2)
     1    *EXP( (GLSIMPS-2)/2 * ALOG(1 - R**2))/SQRTPI
      RETURN
      END
```

E.8 Routines from Appendix A

```
PROGRAM A.1 SIMPSON
C THE FOLLOWING ROUTINE IS INCLUDED
C   IN THE PROGRAM UNIT \CHAPT-6\FITUTIL   (WEBSITE)
C -SIMPSON'S RULE FOR "FUNCTX(X:REAL):REAL"
C  IF FUNCTX HAS OTHER PARAMETERS, THEY MUST BE GLOBAL, E.G., GLSIMPS
```



```
      REAL FUNCTION SIMPSON(FUNCTX, NINTS, LOLIM, HILIM) !--- 2 CALCS/INTERVAL
      EXTERNAL FUNCTX !--- THIS STATEMENT REQ'D IN CALLING PGM ALSO
      REAL  FUNCTX, SUM, X, DX, LOLIM, HILIM
      INTEGER  NINTS, I
       X = LOLIM
       DX = (HILIM - LOLIM)/(2*NINTS)
       SUM=FUNCTX(X)
       SUM= SUM - FUNCTX(HILIM)
      DO 100 I = 1, NINTS
      X=X+2*DX
        SUM=SUM + 4*FUNCTX(X-DX) + 2*FUNCTX(X)
  100   CONTINUE
      SUM = SUM
       SIMPSON = SUM*DX/3.0
       RETURN
      END

PROGRAM A.2 SPLINE INTERPOLATION
C PROGRAM A.1: \APPEND-A\SPLINTST.FOR
C TEST CUBIC SPLINE INTERPOLATION
      PROGRAM SPLINTST
      CHARACTER TITLE(80)
      REAL D2A, D2B, XS, X(100), Y(100), SPLINEINT
      INTEGER N, I
      OPEN(5,'\APPEND-A\SPLINE.DAT')     !--- TEST DATA FILE
      READ(5,1000) TITLE
      PRINT 1000, ' ',TITLE
1000 FORMAT(80A1)
      READ(5,*) N, D2A, D2B !--- NO. OF POINTS, 2ND DERIVATIVES AT BOUNDARY
      PRINT *,'DATA TABLE: N=', N
      PRINT *,'    X              Y'
      DO 100 I = 1, N
        READ(5,*) X(I), Y(I)
        PRINT *, X(I), Y(I)
100 CONTINUE
      CALL SPLINEMAKE(N, D2A, D2B, X, Y)
      CLOSE(5)
200 PRINT *,'TYPE A VALUE OF X  (EXIT WITH ^C)'
      READ *, XS
      PRINT *, 'INTERPOLATED Y = ', SPLINEINT(XS)
      GOTO 200
      END

C ROUTINES FOR CUBIC SPLINE INTERPOLATION.
C CONSTANT INTERVALS IN THE INDEPENDENT VARIABLE ARE ASSUMED.
      SUBROUTINE SPLINEMAKE(NN, D2YDX2A, D2YDX2B, XIN, YIN)
      INTEGER NN
      REAL   D2YDX2A, D2YDX2B, XIN(100), YIN(100)
C -COMMON VARIABLES SET IN SPLINEMAKE, USED IN SPLINEINT-
      COMMON/SPLINES/N, H, XX(100), YY(100), D2YDX2(100)
      INTEGER N
```

```
      REAL H, XX, YY, D2YDX2
      INTEGER I
      REAL  A(100), DELT1(100), DELT2(100), B(100)
      N = NN        !--- USED BY SPLINTINT, THROUGH COMMON/SPLINES/
      H = (XIN(N) - XIN(1))/(N-1)
      DO 100 I = 1, N
         XX(I) = XIN(I)
         YY(I) = YIN(I)
100   CONTINUE
      D2YDX2(1) = D2YDX2A !--- END VALUES OF 2ND DERIVATIVES FROM INPUT
      D2YDX2(N) = D2YDX2B
      A(2) = 4
      DO 200 I =  3, N-1
         A(I) = 4-1/A(I-1)              !--- COEFFICIENTS
200   CONTINUE
      DO 300 I = 2, N
         DELT1(I) = YIN(I) - YIN(I-1)   !--- 1ST DIFFERENCES
300   CONTINUE
      DO 400 I = 2, N-1                 !--- 2ND DIFFERENCES X 6
         DELT2(I) = 6*(DELT1(I+1) - DELT1(I))/(H*H)
400   CONTINUE
      B(2) = DELT2(2) - D2YDX2(1)       !--- B COEFFICIENTS
      DO 500 I= 3, N-1
         B(I) = DELT2(I) - B(I-1)/A(I-1)
500   CONTINUE
      B(N-1) = B(N-1) - D2YDX2(N)
      D2YDX2(N-1) = B(N-1)/A(N-1)
      DO 600 I = N-2, 2, -1
         D2YDX2(I) = (B(I) - D2YDX2(I+1))/A(I) !--- 2ND DERIVATIVES
600   CONTINUE
      RETURN
      END

      REAL FUNCTION DYDX(I) !--- FIRST DERIVATIVE (WEBSITE)
      INTEGER I
      COMMON/SPLINES/N, H, XX(100), YY(100), D2YDX2(100)
      INTEGER N
      REAL H, XX, YY, D2YDX2
      DYDX = (YY(I+1)-YY(I))/H - H*(D2YDX2(I)/3+D2YDX2(I+1)/6)
      RETURN
      END

      REAL FUNCTION D3YDX3(I) !--- THIRD DERIVATIVE (WEBSITE)
      INTEGER I
      COMMON/SPLINES/N, H, XX(100), YY(100), D2YDX2(100)
      INTEGER N
      REAL H, XX, YY, D2YDX2
      D3YDX3 = (D2YDX2(I+1) - D2YDX2(I))/H
      RETURN
      END
```



```
      REAL FUNCTION SPLINEINT(X) !--- INTERPOLATE IN TABLE (FROM SPLINEMAKE)
      REAL X
      COMMON/SPLINES/N, H, XX(100), YY(100), D2YDX2(100)
      INTEGER N
      REAL H, XX, YY, D2YDX2, DYDX, D3YDX3, DX
      INTEGER I
      I = INT((X-XX(1))/H)+1
      IF (I .LT. 1 )  I = 1
      IF (I .GT. N-1 ) I = N-1
      DX = X -XX(I)
C -INTERPOLATE
      IF (I .EQ. N ) THEN
         SPLINEINT = YY(I)
      ELSE
         SPLINEINT = YY(I) + (DYDX(I) + (D2YDX2(I)/2 +D3YDX3(I)/6*DX)*DX)*DX
      ENDIF
      RETURN
      END
```

E.9 Routines from Appendix B

```
C PROGRAM B.1: \APPEND-B\MATRIX.FOR
C INVERT A SQUARE MATRIX
C USES FITVARS
      SUBROUTINE MATINV(M, MARRAY, DET)
      INTEGER M
      REAL MARRAY(10,10), DET
      INTEGER IK(10), JK(10)
      INTEGER I, J, K, L
      REAL AMAX, SAVE
      DET=0
C -FIND LARGEST ELEMENT
      DO 100 K = 1, M
         AMAX=0
1500     DO 200  I = K, M
            DO 300 J = K , M
               IF ( ABS(MARRAY(I,J)) .GT. ABS(AMAX) )  THEN
                  AMAX = MARRAY(I,J)
                  IK(K) = I
                  JK(K) = J
               ENDIF
300         CONTINUE          !--- DO J
200      CONTINUE             !--- DO I
         IF (AMAX .EQ. 0 )  RETURN  !--- WITH 0 DETERMINANT AS SIGNAL
         DET = 1
C -INTERCHANGE ROWS AND COLUMNS TO PUT AMAX IN MARRAY(K,K)
         I = IK(K)
         IF (I .LT. K ) THEN
            GOTO 1500
         ELSEIF (I .GT. K )  THEN
            DO 400 J = 1, M
```

```
                SAVE = MARRAY(K,J)
                MARRAY(K,J) = MARRAY(I,J)
                MARRAY(I,J) = -SAVE
400           CONTINUE                  |---  DO J
            ENDIF                       |---  IF I
            J = JK(K)
            IF (J .LT. K ) THEN
               GOTO 100
            ELSEIF (J .GT. K ) THEN
               DO 500 I = 1, M
                  SAVE = MARRAY(I,K)
                  MARRAY(I,K) = MARRAY(I,J)
                  MARRAY(I,J) = -SAVE
500           CONTINUE  |---  DO I
            ENDIF  |---  IF J
C -ACCUMULATE ELEMENTS OF INVERSE MATRIX
            DO 600 I = 1, M
               IF (I .NE. K )
     1           MARRAY(I,K) = -MARRAY(I,K)/AMAX
600         CONTINUE |---  DO I
            DO 700 I = 1, M
               DO 800 J = 1, M
                  IF ((I .NE. K) .AND. (J .NE. K) )
     1              MARRAY(I,J) = MARRAY(I,J) + MARRAY(I,K)*MARRAY(K,J)
800            CONTINUE                  |---  DO J
700         CONTINUE                     |---  DO I
            DO 900 J = 1, M
               IF (J .NE. K )
     1           MARRAY(K,J) = MARRAY(K,J)/AMAX
900         CONTINUE                     |---  DO J
            MARRAY(K,K) = 1/AMAX
            DET = DET * AMAX
100   CONTINUE                           |---  DO K
C -RESTORE ORDERING OF MATRIX
         DO 1000 L = 1, M
            K = M + 1 - L
            J = IK(K)
            IF (J .GT. K ) THEN
               DO 1100 I = 1, M
                  SAVE = MARRAY(I,K)
                  MARRAY(I,K) = -MARRAY(I,J)
                  MARRAY(I,J) = SAVE
1100           CONTINUE                  |---  DO I
            ENDIF                        |---  IF J
            I = JK(K)
            IF (I .GT. K ) THEN
               DO 1200 J = 1, M
               SAVE = MARRAY(K,J)
               MARRAY(K,J) = -MARRAY(I,J)
               MARRAY(I,J) = SAVE
1200        CONTINUE |---  DO J
```



```
            ENDIF                       |---  IF I
1000  CONTINUE                          |---  DO L
            RETURN
            END

            SUBROUTINE LINEARBYSQUARE(M, A, B, C)  |---  MATRIX PRODUCT
            INTEGER M
            REAL A(10), B(10,10), C(10)
            INTEGER  I,J
            DO 100 I = 1, M
               C(I)=0
               DO 200 J = 1, M
               C(I)=C(I) +A(J)*B(I,J)
 200   CONTINUE
 100        CONTINUE
            RETURN
            END
```

E.10 Routines from Appendix C

```
C PROGRAM C.1: \APPEND-C\STUDENTST.FOR
C CALCULATES BOTH THE GAUSSIAN PROBABILITY
C    AND THE STUDENT'S T PROBABILITY FOR EXCEEDING  A GIVEN VALUE
C    OF (MU-X)/SIGMA, WHERE MU IS THE MEAN VALUE OF X AND SIGMA IS
C    THE UNCERTAINTY IN THE MEAN.
C FOR SPEED, AND TO REDUCE POSSIBILITY OF OVERFLOW, WE
C    CALCULATE THE RATIO OF THE GAMMA FUNCTIONS DIRECTLY
C    IN FUNCTION GAMMACONST.
C TO IMPROVE SPEED AND ACCURACY BY USING SIMPSON'S FOR INTEGRATION
C
      PROGRAM STUDENTS_T
      REAL GP,TP, T
      INTEGER NU
      PRINT *, 'TYPE NDOF AND T = |MU - X|/SIGMA '
      READ *, NU, T
      CALL GTPROB(GP, TP, NU, T)
      PRINT 1100, 100*TP, 100*(1-TP)
      PRINT 1200, 100*GP, 100*(1-GP)
1100  FORMAT(' PROB (STUDENT'S T)  = ',F5.2,'%, 1-PROB = ',F5.2, '%')
1200  FORMAT(' PROB (GAUSSIAN)     = ',F5.2, '%, 1-PROB = ',F5.2,'%')
      END

      REAL FUNCTION STUDENTST(NU, T, G)    !STUDENT'S T DISTRIBUTION
      INTEGER NU
      REAL T, G, X
C X = (1/SQRT(NU*PI)  * (GAMMA((NU+1)/2)/GAMMA(NU/2))*(1+T^2/NU)^(-(NU+1)/2)
      X  = G*EXP( (-(NU+1)/2)* ALOG(1+T*T/NU))
      STUDENTST = X
      END

      REAL FUNCTION GAUSS(X)
```

```
      REAL PI/3.14159/,X
      GAUSS = EXP(-X*X/2)/SQRT(2*PI)
      RETURN
      END

C  GAUSSIAN AND STUDENT'S T PROBABILITIES
      SUBROUTINE GTPROB(GPROB, TPROB, N, T) !INTEGRAL FROM -T TO +T
      REAL GPROB, TPROB,T
      INTEGER N
      REAL GAM, T1, SUMT, SUMG, DT
      GAM  = GAMMACONST(N)        !RATIO OF GAMMAS - FOR SPEED
      DT   = 0.0001               !INTEGRATION STEP
      T1   = 0
      SUMT = 0
      SUMG = 0
      DOWHILE ((T1 .LT. T) .AND. (SUMT*DT .LT. 0.5)) !SIMPLE INTEGRATION.
C  REPLACE BY SIMPSON'S RULE FOR BETTER SPEED AND ACCURACY
         SUMT = SUMT + STUDENTST(N,T1,GAM)
         SUMG = SUMG + GAUSS(T1)
         T1   = T1 + DT
      ENDDO
      TPROB  = 2*SUMT*DT
      GPROB  = 2*SUMG*DT
      RETURN
      END

      REAL FUNCTION GAMMACONST(N)
C  G = GAMMA((H+1)/2)/GAMMA(H/2)/SQRT(H*PI)
C  PRE-CALCULATE RATIO FOR SPEED AND TO  AVOID OVERFLOW
      INTEGER N
      REAL PI/3.14159/
      REAL H, Y1,Y2, G
      H = N
      Y1 = -0.5*(H+1) + 0.5*(H)  *ALOG(0.5*(H+1))
      Y2 = -0.5*H     + 0.5*(H-1)*ALOG(0.5*H)
      G  = EXP(Y1-Y2)*(1+0.0833/(0.5*(H+1)))/((1+0.0833/(0.5*H))
      1  *SQRT(H*PI))
      GAMMACONST = G
      RETURN
      END
      END
```

E.11 Routines from Appendix D

```
C PROGRAM D.1: \APPEND-D\QUIKSCRP.FOR
C CREATE A SCRIPT FILE TO DISPLAY SIMPLE GRAPHS AND HISTOGRAMS
C THE FILE IS READ AND INTERPRETED BY \APPEND-D\QDISPLAY.EXE

C PROGRAM D.2: \APPEND-D\QUIKHIST.FOR
C ASSIGNS DATA TO HISTOGRAM BINS AND PLOTS HISTOGRAM EITHER
C  AS SCREEN CHARACTERS OR IN SCREEN GRAPHISC THROUGH QUIKSCRP
```

REFERENCES

Anderson, R. L. and E. E. Houseman, *Tables of Orthogonal Polynomial Values Extended to N = 104*, Research Bulletin 297, Agricultural Experimental Station, Iowa State University (April, 1942).

Arndt, R. A. and M. H. MacGregor, Nucleon-Nucleon Phase Shift Analysis by Chi-Squared Minimization, in *Methods in Computational Physics*, vol. 6, pp. 253–296, Academic Press, New York (1966).

Baird, D. C., *Experimentation: An Introduction to Measurement Theory and Experiment Design*, Prentice-Hall, Englewood Cliffs, N.J. (1988).

Bajpai, A. C., I. M. Calus, and J. A. Fairley, *Numerical Methods for Engineers and Scientists*, Wiley, Chichester (1977).

Beers, Y., *Introduction to the Theory of Error*, Addison-Wesley, Reading, Mass. (1957).

Box, G. E. P. and M. E. Müller, A Note on the Generation of Random Normal Deviates, *Ann. Math. Statist.*, vol. 29, pp. 610–611 (1958).

David, F. N., *Tables of the Correlation Coefficients*, Cambridge University Press, London (1938).

Dixon, W. J. and F. J. Massey, Jr., *Introduction to Statistical Analysis*, McGraw-Hill, New York (1969).

Eadie, W. T., D. Drijard, F. E. James, M. Roos, and B. Sadoulet, *Statistical Methods in Experimental Physics*, North-Holland, Amsterdam (1971).

Hamilton, W. C., *Statistics in Physical Science*, Ronald Press, New York (1964).

Hamming, R. W., *Numerical Methods for Scientists and Engineers*, McGraw-Hill, New York (1962).

Handbook of Chemistry and Physics, Chemical Rubber Co., Cleveland, Ohio (1973).

Hildebrand, F. B., *Introduction to Numerical Analysis*, McGraw-Hill, New York (1956).

Hoel, P. G., *Introduction to Mathematical Statistics*, Wiley, New York (1954).

IBM, *System/360 Scientific Subroutine Package, Programmer's Manual* (360A-CM-03X).

Knuth, D. E., Seminumerical Algorithms, in *The Art of Computer Programming*, vol. 2, pp. 29ff., Addison-Wesley, Reading, Mass. (1981).

Marquardt, D. W., An Algorithm for Least-Squares Estimation of Nonlinear Parameters, *J. Soc. Ind. Appl. Math.*, vol. II, no. 2, pp. 431–441 (1963).

Melkanoff, M. A., T. Sawada, and J. Raynal, Nuclear Optical Model Calculations, in *Methods in Computational Physics*, vol. 6, pp. 2–80, Academic Press, New York (1966).

Merrington, M. and C. M. Thompson, Tables of Percentage Points of the Inverted Beta (*F*) Distribution, *Biometrica*, vol. 33, pt. 1, pp. 74–87 (1943).

Orear, J., Notes on Statistics for Physicists, UCRL-8417, University of California Radiation Laboratory, Berkeley, Calif. (1958).

Ostle, B., *Statistics in Research*, Iowa State College Press, Ames, Iowa (1963).

Pearson, K, *Tables for Statisticians and Biometricians*, Cambridge University Press, London (1924).

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press, New York (1986).