International Tables for Crystallography (2006). Vol. C, Chapter 8.1, pp. 678–688.

8.1. Least squares

BY E. PRINCE AND P. T. BOGGS

The process of arriving at a model for a crystal structure may usefully be considered to consist of two distinct stages. The first, which may be called determination, involves the use of chemical and physical intuition, direct methods, Fourier and Patterson methods, and other techniques to arrive at an approximate model for the structure that incorporates unit-cell dimensions, space group, chemical composition, and information with respect to the immediate environment of each atom. The second stage, which we shall call refinement, involves finding the values of adjustable parameters in the model that give the best fit between the predicted diffraction intensities and those observed in an experiment, in order to extract precise information about interatomic distances and bond angles, thermal motion, site occupancies, electron distribution, and so forth. Although there are several different criteria for the best fit to data, such as maximum likelihood and maximum entropy, one of the most commonly used is the method of least squares. This chapter discusses both numerical and statistical aspects of refinement by the method of least squares. Because both aspects make extensive use of linear algebra, we begin with a summary of definitions and fundamental operations in linear algebra (Stewart, 1973; Prince, 1994) and of basic definitions and concepts in mathematical statistics (Draper & Smith, 1981; Box, Hunter & Hunter, 1978). We then discuss the principles of linear and nonlinear least squares and conclude with an extensive discussion of numerical methods used in practical implementation of the technique.

8.1.1. Definitions

8.1.1.1. *Linear algebra*

A *matrix* is an ordered, rectangular array of numbers, real or complex. Matrices will be denoted by upper-case, bold italic letters, $\boldsymbol{A}$. Their individual elements will be denoted by upper-case, italic letters with subscripts. A_{ij} denotes the element in the ith row and the jth column of $\boldsymbol{A}$. A matrix with only one row is a *row vector*; a matrix with only one column is a *column vector*. Vectors will be denoted by lower-case, bold roman letters, and their elements will be denoted by lower-case, italic letters with single subscripts. Scalar constants will usually be denoted by lower-case, Greek letters.

A matrix with the same number of rows as columns is *square*. If $A_{ij} = 0$ for all $i > j$, $\boldsymbol{A}$ is *upper triangular*. If $A_{ij} = 0$ for all $i < j$, $\boldsymbol{A}$ is *lower triangular*. If $A_{ij} = 0$ for all $i \neq j$, $\boldsymbol{A}$ is *diagonal*. If $A_{ij} = 0$ for all i and j, $\boldsymbol{A}$ is *null*. A matrix, $\boldsymbol{B}$, such that $B_{ij} = A_{ji}$ for all i and j is the *transpose* of $\boldsymbol{A}$, and is denoted by $\mathbf{A}^T$. Matrices with the same dimensions may be added and subtracted: $(\boldsymbol{A}+\boldsymbol{B})_{ij} = A_{ij} + B_{ij}$. A matrix may be multiplied by a scalar: $(\alpha\boldsymbol{A})_{ij} = \alpha A_{ij}$. Multiplication of matrices is defined by $(\boldsymbol{AB})_{ij} = \sum_{k=1}^{m} A_{ik}B_{kj}$, where m is the number of columns of $\boldsymbol{A}$ and the number of rows of $\boldsymbol{B}$ (which must be equal). Addition and multiplication of matrices obey the associative law: $(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{C} = \boldsymbol{A}+(\boldsymbol{B}+\boldsymbol{C})$; $(\boldsymbol{AB})\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{BC})$. Multiplication of matrices obeys the distributive law: $\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C}) = \boldsymbol{AB}+\boldsymbol{AC}$. Addition of matrices obeys the commutative law: $\boldsymbol{A}+\boldsymbol{B} = \boldsymbol{B}+\boldsymbol{A}$, but multiplication, except in certain (important) special cases, does not: $\boldsymbol{AB} \neq \boldsymbol{BA}$. The transpose of a product is the product of the transposes of the factors in reverse order: $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$.

The *trace* of a square matrix is the sum of its diagonal elements. The *determinant* of an $n \times n$ square matrix, $\boldsymbol{A}$, denoted by $|\boldsymbol{A}|$, is the sum of $n!$ terms, each of which is a product of the diagonal elements of a matrix derived from $\boldsymbol{A}$ by permuting columns or rows (see Stewart, 1973). The *rank* of a matrix (not necessarily square) is the dimension of the largest square submatrix that can be formed from it, by selecting rows and columns, whose determinant is not equal to zero. A matrix has *full column rank* if its rank is equal to its number of columns. A square matrix whose diagonal elements are equal to one and whose off-diagonal elements are equal to zero is an *identity matrix*, denoted by $\boldsymbol{I}$. If $|\boldsymbol{A}| \neq 0$, $\boldsymbol{A}$ is *nonsingular*, and there exists a matrix $\boldsymbol{A}^{-1}$, the *inverse* of $\boldsymbol{A}$, such that $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$. If $|\boldsymbol{A}| = 0$, $\boldsymbol{A}$ is *singular*, and has no inverse. The *adjoint*, or *conjugate transpose*, of $\boldsymbol{A}$ is a matrix, $A^\dagger$, such that $A^\dagger_{ij} = A^*_{ji}$, where the asterisk indicates complex conjugate. If $\boldsymbol{A}^\dagger = \boldsymbol{A}^{-1}$, $\boldsymbol{A}$ is *unitary*. If the elements of a unitary matrix are real, it is *orthogonal*. From this definition, if $\boldsymbol{A}$ is orthogonal, it follows that

$$\sum_{i=1}^{n} A_{ij}^2 = 1$$

for all j, and

$$\sum_{i=1}^{n} A_{ij}A_{ik} = 0$$

if $j \neq k$. By analogy, two column vectors, $\mathbf{x}$ and $\mathbf{y}$, are said to be orthogonal if $\mathbf{x}^T\mathbf{y} = 0$.

For any square matrix, $\boldsymbol{A}$, there exists a set of vectors, $\mathbf{x}_i$, such that $\boldsymbol{A}\mathbf{x}_i = \lambda_i\mathbf{x}_i$, where λ_i is a scalar. The values λ_i are the *eigenvalues* of $\boldsymbol{A}$, and the vectors $\mathbf{x}_i$ are the corresponding *eigenvectors*. If $\boldsymbol{A} = \boldsymbol{A}^\dagger$, $\boldsymbol{A}$ is *Hermitian*, and, if the elements are real, $\boldsymbol{A} = \boldsymbol{A}^T$, so that $\boldsymbol{A}$ is *symmetric*. It can be shown (see, for example, Stewart, 1973) that, if $\boldsymbol{A}$ is Hermitian, all eigenvalues are real, and there exists a unitary matrix, $\boldsymbol{T}$, such that $\boldsymbol{D} = \boldsymbol{T}^\dagger\boldsymbol{A}\boldsymbol{T}$ is diagonal, with the elements of $\boldsymbol{D}$ equal to the eigenvalues of $\boldsymbol{A}$, and the columns of $\boldsymbol{T}$ are the eigenvectors. An $n \times n$ symmetric matrix therefore has n mutually orthogonal eigenvectors. If the product $\mathbf{x}^T\boldsymbol{A}\mathbf{x}$ is greater than (or equal to) zero for any non-null vector, $\mathbf{x}$, $\boldsymbol{A}$ is *positive (semi)definite*. Because $\mathbf{x}$ may be, in particular, an eigenvector, all eigenvalues of a positive (semi)definite matrix are greater than (or equal to) zero. Any matrix of the form $\boldsymbol{B}^T\boldsymbol{B}$ is positive semi definite, and, if $\boldsymbol{B}$ has full column rank, $\boldsymbol{A} = \boldsymbol{B}^T\boldsymbol{B}$ is positive definite. If $\boldsymbol{A}$ is positive definite, there exists an upper triangular matrix, $\boldsymbol{R}$, or, equivalently, a lower triangular matrix, $\boldsymbol{L}$, with positive diagonal elements, such that $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{A}$. $\boldsymbol{R}$, or $\boldsymbol{L}$, is called the *Cholesky factor* of $\boldsymbol{A}$. The *magnitude*, *length* or *Euclidean norm* of a vector, $\mathbf{x}$, denoted by $\|\mathbf{x}\|$, is defined by $\|\mathbf{x}\| = (\mathbf{x}^T\mathbf{x})^{1/2}$. The *induced matrix norm* of a matrix, $\boldsymbol{B}$, denoted $\|\boldsymbol{B}\|$, is defined as the maximum value of $\|\boldsymbol{B}\mathbf{x}\|/\|\mathbf{x}\| = (\mathbf{x}^T\boldsymbol{B}^T\boldsymbol{B}\mathbf{x}/\mathbf{x}^T\mathbf{x})^{1/2}$ for $\|\mathbf{x}\| > 0$. Because $\mathbf{x}^T\boldsymbol{B}^T\boldsymbol{B}\mathbf{x}$ will have its maximum value for a fixed value of $\mathbf{x}^T\mathbf{x}$ when $\mathbf{x}$ is parallel to the eigenvector that corresponds to the largest eigenvalue of $\boldsymbol{B}^T\boldsymbol{B}$, this definition implies that $\|\boldsymbol{B}\|$ is equal to the square root of the largest eigenvalue of $\boldsymbol{B}^T\boldsymbol{B}$. The *condition number* of $\boldsymbol{B}$ is the square root of the ratio of the largest and smallest eigenvalues of $\boldsymbol{B}^T\boldsymbol{B}$. (Other definitions of norms exist, with corresponding definitions of condition number. We shall not be concerned with any of these.)

We shall make extensive use of the so-called QR decomposition, which is defined as follows: For any $n \times p$ ($n \geq p$) real matrix, $\boldsymbol{Z}$, there exists an $n \times n$ orthogonal matrix, $\boldsymbol{Q}$, such that

$$\boldsymbol{Q}^T \boldsymbol{Z} = \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{O} \end{pmatrix}, \tag{8.1.1.1}$$

where $\boldsymbol{R}$ is a $p \times p$ upper triangular matrix, and $\boldsymbol{O}$ denotes an $(n-p) \times p$ null matrix. Thus, we have

$$\boldsymbol{Z} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{O} \end{pmatrix}, \tag{8.1.1.2}$$

which is known as the QR decomposition of $\boldsymbol{Z}$. If we partition $\boldsymbol{Q}$ as $(\boldsymbol{Q}_Z, \boldsymbol{Q}_\perp)$, where $\boldsymbol{Q}_Z$ has dimensions $n \times p$, and $\boldsymbol{Q}_\perp$ has dimensions $n \times (n-p)$, (8.1.1.2) becomes

$$\boldsymbol{Z} = \boldsymbol{Q}_Z \boldsymbol{R}, \tag{8.1.1.3}$$

which is known as the QR factorization. We shall make use of the following facts. First, $\boldsymbol{R}$ is nonsingular if and only if the columns of $\boldsymbol{Z}$ are linearly independent; second, the columns of $\boldsymbol{Q}_Z$ form an orthonormal basis for the range space of $\boldsymbol{Z}$, that is, they span the same space as $\boldsymbol{Z}$; and, third, the columns of $\boldsymbol{Q}_\perp$ form an orthonormal basis for the null space of $\boldsymbol{Z}^T$, that is, $\boldsymbol{Z}^T \boldsymbol{Q}_\perp = \boldsymbol{O}$.

There are two common procedures for computing the QR factorization. The first makes use of *Householder transformations*, which are defined by

$$\boldsymbol{H} = \boldsymbol{I} - 2\mathbf{x}\mathbf{x}^T, \tag{8.1.1.4}$$

where $\mathbf{x}^T\mathbf{x} = 1$. $\boldsymbol{H}$ is symmetric, and $\boldsymbol{H}^2 = \boldsymbol{I}$, so that $\boldsymbol{H}$ is orthogonal. In three dimensions, $\boldsymbol{H}$ corresponds to a reflection in a mirror plane perpendicular to $\mathbf{x}$, because of which Stewart (1973) has suggested the alternative term *elementary reflector*. $\boldsymbol{A}$ vector $\mathbf{v}$ is transformed by $\boldsymbol{H}\mathbf{v}$ into the vector $\|\mathbf{v}\|\mathbf{e}$, where $\mathbf{e}$ represents a vector with $e_1 = 1$, and $e_i = 0$ for $i \neq 1$, if

$$\mathbf{x} = [\mathbf{v} - \|\mathbf{v}\|\mathbf{e}] \big/ \big\|[\mathbf{v} - \|\mathbf{v}\|\mathbf{e}]\big\|. \tag{8.1.1.5}$$

The factorization procedure for an $n \times p$ matrix, $\boldsymbol{A}$ (Stewart, 1973; Anderson *et al.*, 1992), takes as $\mathbf{v}$ in the first step the first column of $\boldsymbol{A}$, and forms $\boldsymbol{A}_1 = \boldsymbol{H}_1\boldsymbol{A}$, which has zeros in all elements of the first column below the diagonal. In the second step, $\mathbf{v}$ has a zero as the first element and is filled out by those elements of the second column of $\boldsymbol{A}_1$ on or below the diagonal. $\boldsymbol{A}_2 = \boldsymbol{H}_2\boldsymbol{A}_1$ then has zeros in all elements below the diagonal in the first two columns. This process is repeated $(p-2)$ more times, after which $\boldsymbol{Q} = \boldsymbol{H}_p \ldots \boldsymbol{H}_2\boldsymbol{H}_1$, and $\boldsymbol{R} = \boldsymbol{A}_p$ is upper triangular.

QR factorization by Householder transformations requires for efficiency that the entire $n \times p$ matrix be stored in memory, and requires of order np^2 operations. A procedure that requires storage of only the upper triangle makes use of *Givens rotations*, which are 2×2 matrices of the form

$$\boldsymbol{G} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}. \tag{8.1.1.6}$$

Multiplication of a $2 \times m$ matrix, $\boldsymbol{B}$, by $\boldsymbol{G}$ will put a zero in the B_{21} element if $\alpha = \arctan B_{21}/B_{11}$. The factorization of $\boldsymbol{A}$ involves reading, or computing, the rows of $\boldsymbol{A}$ one at a time. In the first step, matrix $\boldsymbol{B}_1$ consists of the first row of $\boldsymbol{R}$ and the current row of $\boldsymbol{A}$, from which the first element is eliminated. In the second step, $\boldsymbol{B}_{21}$ is the second row of $\boldsymbol{R}$ and the $(p-1)$ nonzero elements of the second row of the transformed $\boldsymbol{B}_1$. After the first p rows have been treated, each additional row of $\boldsymbol{A}$ requires $2p(p+1)$ multiplications to fill it with zeros. However, because the operation is easily vectorized, the time required may be a small proportion of the total computing time on a vector oriented computer.

8.1.1.2. *Statistics*

A *probability density function*, which will be abbreviated p.d.f., is a function, $\Phi(x)$, such that the probability of finding the random variable x in the interval $a \leq x \leq b$ is given by

$$p(a \leq x \leq b) = \int_a^b \Phi(x)\,\mathrm{d}x.$$

A p.d.f. has the properties

$$\Phi(x) \geq 0, \qquad -\infty < x < +\infty,$$

and

$$\int_{-\infty}^{+\infty} \Phi(x)\,\mathrm{d}x = 1.$$

A *cumulative distribution function*, which will be abbreviated c.d.f., is defined by

$$\Psi(x) = \int_{-\infty}^{x} \Phi(t)\,\mathrm{d}t.$$

The properties of $\Phi(x)$ imply that $0 \leq \Psi(x) \leq 1$, and $\Phi(x) = \mathrm{d}\Psi(x)/\mathrm{d}x$. The *expected value* of a function, $f(x)$, of random variable x is defined by

$$\big\langle f(x) \big\rangle = \int_{-\infty}^{+\infty} f(x)\Phi(x)\,\mathrm{d}x.$$

If $f(x) = x^n$, $\big\langle f(x) \big\rangle = \langle x^n \rangle$ is the *nth moment* of $\Phi(x)$. The first moment, often denoted by μ, is the *mean* of $\Phi(x)$. The second moment about the mean, $\big\langle (x - \langle x \rangle)^2 \big\rangle$, usually denoted by σ^2, is the *variance* of $\Phi(x)$. The positive square root of the variance is the *standard deviation*.

For a vector, $\mathbf{x}$, of random variables, $x_1, x_2, \ldots, x_n$, the *joint probability density function*, or *joint p.d.f.*, is a function, $\Phi_J(\mathbf{x})$, such that

$$\begin{aligned} &p(a_1 \leq x_1 \leq b_1;\ a_2 \leq x_2 \leq b_2; \ldots;\ a_n \leq x_n \leq b_n) \\ &\quad = \int_{a_1}^{b_1} \int_{a_2}^{b_2} \ldots \int_{a_n}^{b_n} \Phi_J(\mathbf{x})\,\mathrm{d}x_1\,\mathrm{d}x_2 \ldots \mathrm{d}x_n. \end{aligned} \tag{8.1.1.7}$$

The *marginal p.d.f.* of an element (or a subset of elements), x_i, is a function, $\Phi_M(x_i)$, such that

$$\begin{aligned} p(a_i \leq x_i \leq b_i) &= \int_{a_i}^{b_i} \Phi_M(x_i)\,\mathrm{d}x_i \\ &= \int_{-\infty}^{+\infty} \ldots \int_{a_i}^{b_i} \ldots \int_{-\infty}^{\infty} \Phi_J(\mathbf{x})\,\mathrm{d}x_1 \ldots \mathrm{d}x_i \ldots \mathrm{d}x_n. \end{aligned} \tag{8.1.1.8}$$

This is a p.d.f. for x_i alone, irrespective of the values that may be found for any other element of $\mathbf{x}$. For two random variables, x and y (either or both of which may be vectors), the *conditional p.d.f. of x given* $y = y_0$ is defined by

$$\Phi_C(x|y_0) = c\Phi_J(x, y)_{y=y_0},$$

where $c = 1/\Phi_M(y_0)$ is a *renormalizing factor*. This is a p.d.f. for x when it is known that $y = y_0$. If $\Phi_C(x|y) = \Phi_M(x)$ for all y, or, equivalently, if $\Phi_J(x, y) = \Phi_M(x)\Phi_M(y)$, the random variables x and y are said to be *statistically independent*.

Moments may be defined for multivariate p.d.f.s in a manner analogous to the one-dimensional case. The mean is a vector defined by

$$\mu_i = \langle x_i \rangle = \int x_i \Phi(\mathbf{x})\, d\mathbf{x},$$

where the volume of integration is the entire domain of $\mathbf{x}$. The *variance–covariance matrix* is defined by

$$\begin{aligned} V_{ij} &= \big\langle (x_i - \langle x_i \rangle)(x_j - \langle x_j \rangle) \big\rangle \\ &= \int (x_i - \langle x_i \rangle)(x_j - \langle x_j \rangle) \Phi_J(\mathbf{x})\, d\mathbf{x}. \end{aligned} \tag{8.1.1.9}$$

The diagonal elements of $\boldsymbol{V}$ are the variances of the marginal p.d.f.s of the elements of $\mathbf{x}$, that is, $V_{ii} = \sigma_i^2$. It can be shown that, if x_i and x_j are statistically independent, $V_{ij} = 0$ when $i \neq j$. If two vectors of random variables, $\mathbf{x}$ and $\mathbf{y}$, are related by a linear transformation, $\mathbf{x} = \boldsymbol{B}\mathbf{y}$, the means of their joint p.d.f.s are related by $\mu_x = B\mu_y$, and their variance–covariance matrices are related by $\boldsymbol{V}_\mathbf{x} = \boldsymbol{B}\boldsymbol{V}_\mathbf{y}\boldsymbol{B}^T$.

8.1.2. Principles of least squares

The method of least squares may be formulated as follows: Given a set of n observations, $y_i\ (i = 1, 2, \ldots, n)$, that are measurements of quantities that can be described by differentiable model functions, $M_i(\mathbf{x})$, where $\mathbf{x}$ is a vector of parameters, $x_j (j = 1, 2, \ldots, p)$, find the values of the parameters for which the sum

$$S = \sum_{i=1}^{n} w_i [y_i - M_i(\mathbf{x})]^2 \tag{8.1.2.1}$$

is minimum. Here, w_i represents a weight assigned to the ith observation. The values of the parameters that give the minimum value of S are called *estimates* of the parameters, and a function of the data that locates the minimum is an *estimator*. A necessary condition for S to be a minimum is for the gradient to vanish, which gives a set of simultaneous equations, the *normal equations*, of the form

$$\frac{\partial S}{\partial x_j} = -2 \sum_{i=1}^{n} w_i [y_i - M_i(\mathbf{x})] \frac{\partial M_i(\mathbf{x})}{\partial x_j} = 0. \tag{8.1.2.2}$$

The model functions, $M_i(\mathbf{x})$, are, in general, nonlinear, and there are no direct ways to solve these systems of equations. Iterative methods for solving them are discussed in Section 8.1.4. Much of the analysis of results, however, is based on the assumption that linear approximations to the model functions are good approximations in the vicinity of the minimum, and we shall therefore begin with a discussion of linear least squares.

To express linear relationships, it is convenient to use matrix notation. Let $\boldsymbol{M}(\mathbf{x})$ and $\mathbf{y}$ be column vectors whose ith elements are $M_i(\mathbf{x})$ and y_i. Similarly, let $\mathbf{b}$ be a vector and $\boldsymbol{A}$ be a matrix such that a linear approximation to the ith model function can be written

$$M_i(\mathbf{x}) = b_i + \sum_{j=1}^{p} A_{ij} x_j. \tag{8.1.2.3}$$

Equations (8.1.2.3) can be written, in matrix form,

$$\boldsymbol{M}(\mathbf{x}) = \mathbf{b} + \boldsymbol{A}\mathbf{x}, \tag{8.1.2.4}$$

and, for this linear model, (8.1.2.1) becomes

$$S = [(\mathbf{y} - \mathbf{b}) - \boldsymbol{A}\mathbf{x}]^T \boldsymbol{W} [(\mathbf{y} - \mathbf{b}) - \boldsymbol{A}\mathbf{x}], \tag{8.1.2.5}$$

where $\boldsymbol{W}$ is a diagonal matrix whose diagonal elements are $W_{ii} = w_i$. In this notation, the normal equations (8.1.2.2) can be written

$$\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A}\mathbf{x} = \boldsymbol{A}^T \boldsymbol{W} (\mathbf{y} - \mathbf{b}), \tag{8.1.2.6}$$

and their solution is

$$\hat{\mathbf{x}} = (\boldsymbol{A}^T \boldsymbol{W} \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{W} (\mathbf{y} - \mathbf{b}). \tag{8.1.2.7}$$

If $W_{ii} > 0$ for all i, and $\boldsymbol{A}$ has full column rank, then $\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$ will be positive definite, and S will have a unique minimum at $\mathbf{x} = \hat{\mathbf{x}}$. The matrix $\boldsymbol{H} = (\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{W}$ is a $p \times n$ matrix that relates the n-dimensional observation space to the p-dimensional parameter space and is known as the *least-squares estimator*; because each element of $\hat{\mathbf{x}}$ is a linear function of the observations, it is a *linear estimator*. [Note that, in actual practice, the matrix $\boldsymbol{H}$ is not actually evaluated, except, possibly, in very small problems. Rather, the linear system $\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\mathbf{x} = \boldsymbol{A}^T\boldsymbol{W}(\mathbf{y} - \mathbf{b})$ is solved using the methods of Section 8.1.3.]

The least-squares estimator has some special properties in statistical analysis. Suppose that the elements of $\mathbf{y}$ are experimental observations drawn at random from populations whose means are given by the model, $\boldsymbol{M}(\mathbf{x})$, for some unknown $\mathbf{x}$, which we wish to estimate. This may be written

$$\langle \mathbf{y} - \mathbf{b} \rangle = \boldsymbol{A}\mathbf{x}. \tag{8.1.2.8}$$

The expected value of the least-squares estimate is

$$\begin{aligned} \langle \hat{\mathbf{x}} \rangle &= \big\langle (\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{W}(\mathbf{y} - \mathbf{b}) \big\rangle \\ &= (\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{W}\langle \mathbf{y} - \mathbf{b} \rangle \\ &= (\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}\mathbf{x} \\ &= \mathbf{x}. \end{aligned} \tag{8.1.2.9}$$

If the expected value of an estimate is equal to the variable to be estimated, the estimator is said to be *unbiased*. Equation (8.1.2.9) shows that the least-squares estimator is an unbiased estimator for $\mathbf{x}$, independent of $\boldsymbol{W}$, provided only that $\mathbf{y}$ is an unbiased estimate of $\boldsymbol{M}(\mathbf{x})$, the matrix $\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$ is nonsingular, and the elements of $\boldsymbol{W}$ are constants independent of $\mathbf{y}$ and $\boldsymbol{M}(\mathbf{x})$. Let $\boldsymbol{V}_\mathbf{x}$ and $\boldsymbol{V}_\mathbf{y}$ be the variance–covariance matrices for the joint p.d.f.s of the elements of $\mathbf{x}$ and $\mathbf{y}$, respectively. Then, $\boldsymbol{V}_\mathbf{x} = \boldsymbol{H}\boldsymbol{V}_\mathbf{y}\boldsymbol{H}^T$. Let $\boldsymbol{G}$ be the matrix $(\boldsymbol{A}^T\boldsymbol{V}_\mathbf{y}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{V}^{-1}$, so that $\hat{\mathbf{x}} = \boldsymbol{G}(\mathbf{y} - \mathbf{b})$ is the particular least-squares estimate for which $\boldsymbol{W} = \boldsymbol{V}_\mathbf{y}^{-1}$. Then, $\boldsymbol{V}_\mathbf{x} = \boldsymbol{G}\boldsymbol{V}_\mathbf{y}\boldsymbol{G}^T$. If $\boldsymbol{V}_\mathbf{y}$ is positive definite, its lower triangular Cholesky factor, $\boldsymbol{L}$, exists, so that $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{V}_\mathbf{y}$. [If $\boldsymbol{V}$ is diagonal, $\boldsymbol{L}$ is also diagonal, with $L_{ii} = (\boldsymbol{V}_\mathbf{y})_{ii}^{1/2}$.] It is readily verified that the matrix product $[(\boldsymbol{H} - \boldsymbol{G})\boldsymbol{L}][(\boldsymbol{H} - \boldsymbol{G})\boldsymbol{L}]^T = \boldsymbol{H}\boldsymbol{V}_\mathbf{y}\boldsymbol{H}^T - \boldsymbol{G}\boldsymbol{V}_\mathbf{y}\boldsymbol{G}^T$, but the diagonal elements of this product are the sums of squares of the elements of rows of $(\boldsymbol{H} - \boldsymbol{G})\boldsymbol{L}$, and are therefore greater than or equal to zero. Therefore, the diagonal elements of $\boldsymbol{V}_\mathbf{x}$, which are the variances of the marginal p.d.f.s of the elements of $\hat{\mathbf{x}}$, are minimum when $\boldsymbol{W} = \boldsymbol{V}_\mathbf{y}^{-1}$.

Thus, the least-squares estimator is unbiased for any positive-definite weight matrix, $\boldsymbol{W}$, but the variances of the elements of the vector of estimated parameters are minimized if $\boldsymbol{W} = \boldsymbol{V}_\mathbf{y}^{-1}$. [Note also that $\boldsymbol{V}_\mathbf{x} = (\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A})^{-1}$ if, *and only if*, $\boldsymbol{W} = \boldsymbol{V}_\mathbf{y}^{-1}$.] For this reason, the least-squares estimator with weights proportional to the reciprocals of the variances of the observations is referred to as the *best linear unbiased estimator* for the parameters of a model describing those observations. (These specific results are included in a more general result known as the *Gauss–Markov theorem*.)

The analysis up to this point has assumed that the model is linear, that is that the expected values of the observations can be expressed by $\langle \mathbf{y} \rangle = \mathbf{b} + \boldsymbol{A}\mathbf{x}$, where $\boldsymbol{A}$ is some matrix. In crystallography, of course, the model is highly nonlinear, and this assumption is not valid. The principles of linear least squares

can be extended to nonlinear model functions by first finding, by numerical methods, a point in parameter space, $\mathbf{x}_0$, at which the gradient vanishes and then expanding the model functions about that point in Taylor's series, retaining only the linear terms. Equation (8.1.2.4) then becomes

$$\boldsymbol{M}(\mathbf{x}) \approx \boldsymbol{M}(\mathbf{x}_0) + \boldsymbol{A}(\mathbf{x} - \mathbf{x}_0), \tag{8.1.2.10}$$

where $A_{ij} = \partial M_i(\mathbf{x})/\partial x_j$ evaluated at $\mathbf{x} = \mathbf{x}_0$. Because we have already found the least-squares solution, the estimate

$$\begin{aligned}\widehat{\mathbf{x}} &= \mathbf{x}_0 + (\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{W}[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_0)] \\ &= \mathbf{x}_0 + \boldsymbol{H}[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_0)]\end{aligned} \tag{8.1.2.11}$$

reduces to $\widehat{\mathbf{x}} = \mathbf{x}_0$. It is important, however, not to confuse $\mathbf{x}_0$, which is a convenient origin, with $\widehat{\mathbf{x}}$, which is a random variable describable by a joint p.d.f. with mean $\mathbf{x}_0$ and a variance–covariance matrix $\boldsymbol{V}_\mathbf{x} = \boldsymbol{H}\boldsymbol{V}_\mathbf{y}\boldsymbol{H}^T$, reducing to $(\boldsymbol{A}^T\boldsymbol{V}_\mathbf{y}^{-1}\boldsymbol{A})^{-1}$ when $\boldsymbol{W} = \boldsymbol{V}_\mathbf{y}^{-1}$.

This variance–covariance matrix is the one appropriate to the linear approximation given in (8.1.2.10), and it is valid (and the estimate is unbiased) only to the extent that the approximation is a good one. *A* useful criterion for an adequate approximation (Fedorov, 1972) is, for each j and k,

$$\left|\sum_{i=1}^{n} w_i\sigma_i \frac{\partial^2 M_i(\mathbf{x}_0)}{\partial x_j \partial x_k}\right| \ll \left(\left\{\sum_{i=1}^{n} w_i\left[\frac{\partial M_i(\mathbf{x}_0)}{\partial x_j}\right]^2\right\} \times \left\{\sum_{i=1}^{n} w_i\left[\frac{\partial M_i(\mathbf{x}_0)}{\partial x_k}\right]^2\right\}\right)^{1/2}, \tag{8.1.2.12}$$

where σ_i is the estimated standard deviation or *standard uncertainty* (Schwarzenbach, Abrahams, Flack, Prince & Wilson, 1995) of y_i. This criterion states that the curvature of $S(\mathbf{y}, \mathbf{x})$ in a region whose size is of order σ in observation space is small; it ensures that the effect of second-derivative terms in the normal-equations matrix on the eigenvalues and eigenvectors of the matrix is negligible. [For a further discussion and some numerical tests of alternatives, see Donaldson & Schnabel (1986).]

The process of refinement can be viewed as the construction of a conditional p.d.f. of a set of model parameters, $\mathbf{x}$, given a set of observations, $\mathbf{y}$. An important expression for this p.d.f. is derived from two equivalent expressions for the joint p.d.f. of $\mathbf{x}$ and $\mathbf{y}$:

$$\Phi_J(\mathbf{x}, \mathbf{y}) = \Phi_C(\mathbf{x}|\mathbf{y})\Phi_M(\mathbf{y}) = \Phi_C(\mathbf{y}|\mathbf{x})\Phi_M(\mathbf{x}). \tag{8.1.2.13}$$

Provided $\Phi_M(\mathbf{y}) > 0$, the conditional p.d.f. we seek can be written

$$\Phi_C(\mathbf{x}|\mathbf{y}) = \Phi_C(\mathbf{y}|\mathbf{x})\Phi_M(\mathbf{x})/\Phi_M(\mathbf{y}). \tag{8.1.2.14}$$

Here, the factor $[1/\Phi_M(\mathbf{y})]$ is the factor that is required to normalize the p.d.f. $\Phi_C(\mathbf{y}|\mathbf{x})$ is the conditional probability of observing a set of values of $\mathbf{y}$ as a function of $\mathbf{x}$. When the observations have already been made, however, this can also be considered a density function for $\mathbf{x}$ that measures the *likelihood* that those particular values of $\mathbf{y}$ would have been observed for various values of $\mathbf{x}$. It is therefore frequently written $\ell(\mathbf{x}|\mathbf{y})$, and (8.1.2.14) becomes

$$\Phi_C(\mathbf{x}|\mathbf{y}) = c\ell(\mathbf{x}|\mathbf{y})\Phi_M(\mathbf{x}), \tag{8.1.2.15}$$

where $c = [1/\Phi_M(\mathbf{y})]$ is the normalizing constant. $\Phi_M(\mathbf{x})$, the marginal p.d.f. of $\mathbf{x}$ in the absence of any additional information, incorporates all previously available information concerning $\mathbf{x}$, and is known as the *prior p.d.f.*, or, frequently, simply as the *prior* of $\mathbf{x}$. Similarly, $\Phi_C(\mathbf{x}|\mathbf{y})$ is the *posterior p.d.f.*, or the *posterior*, of $\mathbf{x}$. The relation in (8.1.2.14) and (8.1.2.15) was first stated in the eighteenth century by Thomas Bayes, and it is therefore known as Bayes's theorem (Box & Tiao, 1973). Although its validity has never been in serious question, its application has divided statisticians into two vehemently disputing camps, one of which, the frequentists, considers that Bayesian methods give nonobjective results, while the other, the Bayesians, considers that only by careful construction of a 'noninformative' prior can true objectivity be achieved (Berger & Wolpert, 1984).

Diffraction data, in general, contain no phase information, so the likelihood function for the structure factor, F, given a value of observed intensity, will have a value significantly different from zero in an annular region of the complex plane with a mean radius equal to $|F|$. Because this is insufficient information with which to determine a crystal structure, a prior p.d.f. is constructed in one (or some combination) of two ways. Either the prior knowledge that electron density is non negative is used to construct a joint p.d.f. of amplitudes and phases, given amplitudes for all reflections and phases for a few of them (direct methods), or chemical knowledge and intuition are used to construct a trial structure from which structure factors can be calculated, and the phase of $F_{\rm calc}$ is assigned to $F_{\rm obs}$. Both of these procedures can be considered to be applications of Bayes's theorem. In fact, $F_{\rm calc}$ for a refined structure can be considered a Bayesian estimate of F.

8.1.3. Implementation of linear least squares

In this section, we consider in detail numerical methods for solving linear least-squares problems, that is, the situation where (8.1.2.4) and (8.1.2.5) apply exactly.

8.1.3.1. *Use of the QR factorization*

The linear least-squares problem can be viewed geometrically as the problem of finding the point in a p-dimensional subspace, defined as the set of points that can be reached by a linear combination of the columns of $\boldsymbol{A}$, closest to a given point, $\mathbf{y}$, in an n-dimensional observation space. Since this is equivalent to finding the orthogonal projection of point $\mathbf{y}$ into that subspace, it is not surprising that an orthogonal decomposition of $\boldsymbol{A}$ helps to solve the problem. For convenience in this discussion, let us remove the weight matrix from the problem by defining the standardized design matrix by

$$\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{A}, \tag{8.1.3.1}$$

where $\boldsymbol{U}$ is the upper triangular Cholesky factor of $\boldsymbol{W}$.

Consider the least-squares problem with the QR factorization of $\boldsymbol{Z}$, as given in Subsection 8.1.1.1. For $\mathbf{y}' = \boldsymbol{U}(\mathbf{y} - \mathbf{b})$, (8.1.2.5) becomes

$$\begin{aligned}S &= (\mathbf{y}' - \boldsymbol{Z}\mathbf{x})^T(\mathbf{y}' - \boldsymbol{Z}\mathbf{x}) \\ &= [\boldsymbol{Q}^T(\mathbf{y}' - \boldsymbol{Z}\mathbf{x})]^T[\boldsymbol{Q}^T(\mathbf{y}' - \boldsymbol{Z}\mathbf{x})],\end{aligned} \tag{8.1.3.2}$$

which reduces to

$$S = (\boldsymbol{Q}_Z^T\mathbf{y}' - \boldsymbol{R}\mathbf{x})^T(\boldsymbol{Q}_Z^T\mathbf{y}' - \boldsymbol{R}\mathbf{x}) + \mathbf{y}'^T\boldsymbol{Q}_\perp\boldsymbol{Q}_\perp^T\mathbf{y}'. \tag{8.1.3.3}$$

The second term in (8.1.3.3) is independent of $\mathbf{x}$, and is therefore the sum of squared residuals. The first term vanishes if

$$\boldsymbol{R}\mathbf{x} = \boldsymbol{Q}_Z^T\mathbf{y}', \tag{8.1.3.4}$$

which, because $\boldsymbol{R}$ is upper triangular, is easily solved for $\mathbf{x}$. The QR decomposition of $\boldsymbol{Z}$ therefore leads naturally to the following algorithm for solving the linear least-squares problem:

(1) compute the QR factorization of $\boldsymbol{Z}$;
(2) compute $\boldsymbol{Q}_Z^T\mathbf{y}'$;
(3) solve $\boldsymbol{R}\mathbf{x} = \boldsymbol{Q}_Z^T\mathbf{y}'$ for $\mathbf{x}$.
(4) compute the residual sum of squares by $\mathbf{y}'^T\mathbf{y}' - \mathbf{y}'\boldsymbol{Q}_Z\boldsymbol{Q}_Z^T\mathbf{y}'$.
(5) compute the variance–covariance matrix from $\boldsymbol{V}_\mathbf{x} = \boldsymbol{R}^{-1}(\boldsymbol{R}^{-1})^T$.

8.1.3.2. *The normal equations*

Let us now consider the relationship of the QR procedure for solving the linear least-squares problem to the classical method based on the normal equations. The normal equations can be derived by differentiating (8.1.3.2) and equating the result to a null vector. This yields

$$\boldsymbol{Z}^T\boldsymbol{Z}\mathbf{x} = \boldsymbol{Z}^T\mathbf{y}'. \tag{8.1.3.5}$$

The algorithm is therefore to compute the cross-product matrix, $\boldsymbol{B} = \boldsymbol{Z}^T\boldsymbol{Z}$, and the right-hand side, $\mathbf{d} = \boldsymbol{Z}^T\mathbf{y}'$, and to solve the resulting system of equations, $\boldsymbol{B}\mathbf{x} = \mathbf{d}$. This is usually accomplished by computing the Cholesky decomposition of $\boldsymbol{B}$, that is $\boldsymbol{B} = \boldsymbol{C}^T\boldsymbol{C}$, where $\boldsymbol{C}$ is upper triangular, and then solving the two triangular systems $\boldsymbol{C}^T\mathbf{v} = \mathbf{d}$ and $\boldsymbol{C}\mathbf{x} = \mathbf{v}$. Because $\boldsymbol{Z} = \boldsymbol{Q}_Z\boldsymbol{R}$, equation (8.1.3.5) becomes

$$\boldsymbol{R}^T\boldsymbol{Q}_Z^T\boldsymbol{Q}_Z\boldsymbol{R}\mathbf{x} = \boldsymbol{R}^T\boldsymbol{Q}_Z^T\mathbf{y}', \tag{8.1.3.6}$$

or

$$\boldsymbol{R}^T\boldsymbol{R}\mathbf{x} = \boldsymbol{R}^T\boldsymbol{Q}_Z^T\mathbf{y}'. \tag{8.1.3.7}$$

It is clear that $\boldsymbol{R}$ is the Cholesky factor of $\boldsymbol{Z}^T\boldsymbol{Z}$, although it is formed in a different way. This procedure requires of order $(np^2)/2$ operations to form the product $\boldsymbol{Z}^T\boldsymbol{Z}$ and $p^3/3$ operations for the Cholesky decomposition. In some situations, the extra time to compute the QR factorization is justified because of greater stability, as will be discussed below. Most other quantities of statistical interest can be computed directly from the QR factorization.

8.1.3.3. *Conditioning*

The condition number of $\boldsymbol{Z}$, which is defined (Subsection 8.1.1.1) as the square root of the ratio of the largest to the smallest eigenvalue of $\boldsymbol{Z}^T\boldsymbol{Z}$, is an indicator of the effect a small change in an element of $\boldsymbol{Z}$ will have on the elements of $(\boldsymbol{Z}^T\boldsymbol{Z})^{-1}$and of $\hat{\mathbf{x}}$. A large value of the condition number means that small errors in computing an element of $\boldsymbol{Z}$, owing possibly to truncation or roundoff in the computer, can introduce large errors into the elements of the inverse matrix. Also, when the condition number is large, the standard uncertainties of some estimated parameters will be large. A large condition number, as defined in this way, can result from either scaling or correlation or some combination of these. To illustrate this, consider the matrices

$$\boldsymbol{Z}^T\boldsymbol{Z} = \begin{pmatrix} 2+\varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix}$$

and

$$\boldsymbol{Z}^T\boldsymbol{Z} = \begin{pmatrix} 1 & 1-\varepsilon \\ 1-\varepsilon & 1 \end{pmatrix},$$

where ε represents machine precision, which can be defined as the smallest number in machine representation that, when added to 1, gives a result different from 1. By the conventional definition, both of these matrices have a condition number for $\boldsymbol{Z}$ of $[(2+\varepsilon)/\varepsilon]^{1/2}$. Because numbers of order ε can be perfectly well represented, however, the first one can be inverted without loss of precision, whereas an inverse for the second would be totally meaningless. It is good practice, therefore, to factor the design matrix, $\boldsymbol{Z}$, into the form

$$\boldsymbol{Z} = \boldsymbol{T}\boldsymbol{S}, \tag{8.1.3.8}$$

where $\boldsymbol{S}$ is a $p \times p$ diagonal matrix whose elements define some kind of 'natural' unit appropriate to the parameter represented in each column of $\boldsymbol{Z}$. The ideal natural unit would be the standard uncertainty of that parameter, but this is not available until after the calculation has been completed. If correlation is not too severe, suitable values for the elements of $\boldsymbol{S}$, of the same order of magnitude as those derived from the standard uncertainty, are the column Euclidean norms, that is

$$S = \|\mathbf{z}_j\| = (\mathbf{z}_j^T\mathbf{z}_j)^{1/2}, \tag{8.1.3.9}$$

where $\mathbf{z}_j$ denotes the jth column of $\boldsymbol{Z}$. This scaling causes all diagonal elements of $\boldsymbol{Z}^T\boldsymbol{Z}$ to be equal to one, and errors in the elements of $\boldsymbol{Z}$ will have roughly equal effects.

Ill conditioning that results from correlation, as in the second example above, is more difficult to deal with. It is an indication that some linear combination of parameters, some eigenvector of the normal equations matrix, is poorly determined by the available data. Use of the QR factorization of $\boldsymbol{Z}$ to compute the Cholesky factor of $\boldsymbol{Z}^T\boldsymbol{Z}$ may be advantageous, in spite of the additional computation time, because better numerical stability is obtained in marginal situations. As a practical matter, however, it is important to recognize that an ill conditioned matrix is a symptom of a flaw in the model or in the experimental design (or both). Use can be made of the fact that, although determining the entire set of eigenvalues and eigenvectors of a large matrix is computationally an inherently difficult problem, a relatively simple algorithm, known as a *condition estimator* (Anderson *et al.*, 1992), can produce a good approximation to the eigenvector that corresponds to the smallest eigenvalue of a nearly singular matrix. This information can be used in either or both of two ways. First, without any fundamental modification to the model or the experiment, a simple, linear transformation of the parameters so that the problem eigenvector is one of the independent parameters, followed by rescaling, can resolve the numerical difficulties in computing the estimates. A common example is the situation where a phase transition results in the doubling of a unit cell, with pairs of atoms almost but not quite related by a lattice translation. A transformation that makes the estimated parameters the sums and differences of corresponding parameters in related pairs of atoms can make a dramatic improvement in the condition number. Alternatively, the problem eigenvector can be set to some value determined from theory or from some other experiment (see Section 8.3.1), or additional data can be collected that are selected to make that combination of parameters determinate.

8.1.4. Methods for nonlinear least squares

Recall (equation 8.1.2.1) that the general, nonlinear problem can be stated in the form: find the minimum of

$$S(\mathbf{x}) = \sum_{i=1}^{n} w_i[y_i - M_i(\mathbf{x})]^2, \tag{8.1.4.1}$$

where $\mathbf{x}$ is a vector of p parameters, and $\boldsymbol{M}(\mathbf{x})$ represents a set of model functions that predict the values of observations, $\mathbf{y}$. In this section, we discuss two useful ways of solving this problem and consider the relative merits of each. The first is based on iteratively linearizing the functions $M_i(\mathbf{x})$ and approximating

(8.1.4.1) by one of the form in (8.1.2.5). The second uses an unconstrained minimization algorithm, based on Newton's method, to minimize $S(\mathbf{x})$.

8.1.4.1. *The Gauss–Newton algorithm*

Let $\mathbf{x}_c$ be the current approximation to $\hat{\mathbf{x}}$, the solution to (8.1.4.1). We construct a linear approximation to $M_i(\mathbf{x})$ in the vicinity of $\mathbf{x} = \mathbf{x}_c$ by expanding it in a Taylor series through the linear terms, obtaining

$$M_i(\mathbf{x}) = M_i(\mathbf{x}_c) + \sum_{j=1}^{p} J_{ij}(\mathbf{x} - \mathbf{x}_c)_j, \tag{8.1.4.2}$$

where $\boldsymbol{J}$ is the Jacobian matrix, defined by $J_{ij} = \partial M_i(\mathbf{x})/\partial x_j$. A straightforward procedure, known as the *Gauss–Newton algorithm*, may be formally stated as follows:

(1) compute $\mathbf{d}$ as the solution of the linear system

$$\boldsymbol{J}^T\boldsymbol{W}\boldsymbol{J}\mathbf{d} = \boldsymbol{J}^T\boldsymbol{W}[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_c)];$$

(2) set $\mathbf{x}_c = \mathbf{x}_c + \mathbf{d}$;

(3) if not converged (see Subsection 8.1.4.4), go to (1), else stop.

The convergence rate of the Gauss–Newton algorithm depends on the size of the residual, that is, on $S(\hat{\mathbf{x}})$. If $S(\hat{\mathbf{x}}) = 0$, then the convergence rate is quadratic; if it is small, then the rate is linear; but if $S(\hat{\mathbf{x}})$ is large, then the procedure is not locally convergent at all. Fortunately, this procedure can be altered so that it is always locally convergent, and even globally convergent, that is, convergent to a relative minimum from any starting point. There are two possibilities. First, the procedure can be modified to include a line search. This is accomplished by computing the direction $\mathbf{d}$ as above and then choosing α such that $S(\mathbf{x}_c + \alpha\mathbf{d})$ is sufficiently smaller than $S(\mathbf{x}_c)$. In order to guarantee convergence, one uses the test

$$S(\mathbf{x}_c + \alpha\mathbf{d}) < S(\mathbf{x}_c) - \alpha\gamma\mathbf{d}^T\boldsymbol{J}(\mathbf{x}_c)^T[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_c)], \tag{8.1.4.3}$$

where, as actually implemented in modern codes, γ has values of the order of 10^{-4}. [In theory, a slightly stronger condition is necessary, but this is usually avoided by other means. See Dennis & Schnabel (1983) for details.] While this improves the situation dramatically, it still suffers from the deficiency that it is very slow on some problems, and it is undefined if the rank of the Jacobian, $\boldsymbol{J}(\hat{\mathbf{x}})$, is less than p. $\boldsymbol{J}(\mathbf{x})$ usually has rank p near the solution, but it may be rank deficient far away. Also, it may be 'close' to rank deficient, and give numerical difficulties (see Subsection 8.1.3.3).

8.1.4.2. *Trust-region methods – the Levenberg–Marquardt algorithm*

These remaining problems can be addressed by the utilization of a trust-region modification to the basic Gauss–Newton algorithm. For this procedure, we compute the step, $\mathbf{d}$, as the solution to the linear least-squares problem subject to the constraint that $\|\mathbf{d}\| \le \tau_c$, where τ_c is the *trust-region radius*. Here, the linearized model is modified by admitting that the linearization is only valid within a limited region around the current approximation. It is clear that, if the trust region is sufficiently large, this constrained step will in fact be unconstrained, and the step will be the same as the Gauss–Newton step. If the constraint is active, however, the step has the form

$$[\boldsymbol{J}(\mathbf{x}_c)^T\boldsymbol{W}\boldsymbol{J}(\mathbf{x}_c) + \mu\boldsymbol{I}]\mathbf{d}(\mu) = \boldsymbol{J}(\mathbf{x}_c)\boldsymbol{W}[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_c)], \tag{8.1.4.4}$$

where μ is the Lagrange multiplier corresponding to the constraint (see Section 8.3.1), that is, μ is the value such that $\|\mathbf{d}(\mu)\| = \tau_c$. Formula (8.1.4.4) is known as the *Levenberg–Marquardt* equation. It can be seen from this formula that the step direction is intermediate between the Gauss–Newton direction and the direction of steepest descent, for which reason it is frequently known as ''Marquardt's compromise'' (Draper & Smith, 1981).

Efficient numerical calculation of $\mathbf{d}$ for a given value of μ is accomplished by noting that (8.1.4.4) is equivalent to the linear least-squares problem, find the minimum of

$$S = \left\| \begin{pmatrix} \boldsymbol{J}(\mathbf{x}_c) \\ \sqrt{\mu}\boldsymbol{I} \end{pmatrix}\mathbf{d} - \begin{pmatrix} [\mathbf{y} - \boldsymbol{M}(\mathbf{x}_c)] \\ \boldsymbol{O} \end{pmatrix} \right\|^2. \tag{8.1.4.5}$$

This problem can be solved by saving the QR factorization for $\mu = 0$ and updating it for various values of μ greater than 0. The actual computation of μ is accomplished by a modified Newton method applied to the constraint equation (see Dennis & Schnabel, 1983, for details). Having calculated the constrained value of $\mathbf{d}$, we set $\mathbf{x}_+ = \mathbf{x}_c + \mathbf{d}$. The algorithm is completed by specifying a procedure for updating the trust-region parameter, τ_c, after each step. This is done by comparing the actual value of $S(\mathbf{x}_+)$ with the predicted value based on the linearization. If there is good agreement between these values, τ is increased. If there is not good agreement, τ is left unchanged, and if $S(\mathbf{x}_+) > S(\mathbf{x}_c)$, the step is rejected, and τ is reduced. Global convergence can be shown under reasonable conditions, and very good computational performance has been observed in practice.

8.1.4.3. *Quasi-Newton, or secant, methods*

The second class of methods for solving the general, nonlinear least-squares problem is based on the unconstrained minimization of the function $S(\mathbf{x})$, as defined in (8.1.4.1). Newton's method for solving this problem is derived by constructing a quadratic approximation to S at the current trial point, $\mathbf{x}_c$, giving

$$S_c(\mathbf{d}) = S(\mathbf{x}_c) + \mathbf{g}(\mathbf{x}_c)^T\mathbf{d} + (1/2)\mathbf{d}^T\boldsymbol{H}(\mathbf{x}_c)\mathbf{d}, \tag{8.1.4.6}$$

from which the Newton step is obtained by minimizing S_c with respect to $\mathbf{d}$. Here, $\mathbf{g}$ represents the gradient of S and $\boldsymbol{H}$ is the Hessian matrix, the $p \times p$ symmetric matrix of second partial derivatives, $H_{jk} = \partial^2 S/\partial x_j \partial x_k$. Thus, $\mathbf{d}$ is calculated by solving the linear system

$$\boldsymbol{H}(\mathbf{x}_c)\mathbf{d} = -\mathbf{g}(\mathbf{x}_c). \tag{8.1.4.7}$$

[Note: In the literature on optimization, the notation $\nabla^2 S(\mathbf{x})$ is often used to denote the Hessian matrix of the function $S(\mathbf{x})$. This should not be confused with the Laplacian operator.] While Newton's method is locally quadratically convergent, it suffers from well known drawbacks. First, it is not globally convergent without employing some form of line search or trust region to safeguard it. Second, it requires the computation of the Hessian of S in each iteration.

The Hessian, however, has the form

$$\boldsymbol{H}(\mathbf{x}_c) = \boldsymbol{J}(\mathbf{x}_c)^T\boldsymbol{W}\boldsymbol{J}(\mathbf{x}_c) + \boldsymbol{B}(\mathbf{x}_c), \tag{8.1.4.8}$$

where $\boldsymbol{B}(\mathbf{x})$ is given by

$$B(\mathbf{x}_c)_{jk} = \sum_{i=1}^{n} w_i[y_i - M_i(\mathbf{x}_c)]\partial M_i^2(\mathbf{x}_c)/\partial x_j \partial x_k. \tag{8.1.4.9}$$

The first term of the Hessian, which is dependent only on the Jacobian, is readily available, but $\boldsymbol{B}$, even if it is available analytically, is, typically, expensive to compute. Furthermore,

in some situations, such as in the vicinity of a more symmetric model, $\boldsymbol{H}(\mathbf{x}_c)$ may not be positive definite.

In order to overcome these difficulties, various methods have been proposed to approximate $\boldsymbol{H}$ without calculating $\boldsymbol{B}$. Because these methods are based on approximations to Newton's method, they are often referred to as *quasi-Newton methods*. Two popular versions use approximations to $\boldsymbol{B}$ that are known as the Davidon–Fletcher–Powell (DFP) update and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) update, with BFGS being apparently superior in practice. The basic idea of both procedures is to use values of the gradient to update an approximation to the Hessian in such a way as to preserve as much previous information as possible, while incorporating the new information obtained in the most recent step. From (8.1.4.6), the gradient of $S_c(\mathbf{d})$ is

$$\mathbf{g}(\mathbf{x}_c + \mathbf{d}) = \mathbf{g}(\mathbf{x}_c) + \boldsymbol{H}(\mathbf{x}_c)\mathbf{d}. \tag{8.1.4.10}$$

If the true function is not quadratic, or if $\boldsymbol{H}$ is only approximately known, the gradient at the point $\mathbf{x}_c + \mathbf{d}$, where $\mathbf{d}$ is the solution of (8.1.4.7), will not vanish. We make use of the value of $\mathbf{g}(\mathbf{x}_c + \mathbf{d})$ to compute a correction to $\boldsymbol{H}$ to give a Hessian that would have predicted what was actually found. That is, find an *update matrix*, $\boldsymbol{B}'$, such that

$$[\boldsymbol{H}(\mathbf{x}_c) + \boldsymbol{B}'(\mathbf{x}_c)]\mathbf{d} = \mathbf{g}(\mathbf{x}_c + \mathbf{d}) - \mathbf{g}(\mathbf{x}_c). \tag{8.1.4.11}$$

This is known as the *secant*, or *quasi-Newton*, relation. An algorithm based on the BFGS update goes as follows: Given $\mathbf{x}_c$ and $\boldsymbol{H}_c$ (usually a scaled, identity matrix is used as the initial approximation, in which case the first step is in the steepest descent direction),

(1) solve $\boldsymbol{H}_c\mathbf{d} = -\mathbf{g}(\mathbf{x}_c)$ for the direction $\mathbf{d}$;
(2) compute α such that $S(\mathbf{x}_c + \alpha\mathbf{d})$ satisfies condition (8.1.4.3);
(3) set $\mathbf{x}_c = \mathbf{x}_c + \alpha\mathbf{d}$;
(4) update $\boldsymbol{H}_c$ by $\boldsymbol{H}_c + \mathbf{g}(\mathbf{x}_c)\mathbf{g}(\mathbf{x}_c)^T/\mathbf{d}^T\mathbf{g}(\mathbf{x}_c) + \mathbf{q}\mathbf{q}^T/\alpha\mathbf{q}^T\mathbf{d}$ (see Nash & Sofer, 1995), where $\mathbf{q} = \mathbf{g}(\mathbf{x}_c + \alpha\mathbf{d}) - \mathbf{g}(\mathbf{x}_c)$;
(5) if not converged, go to (1), else stop.

If, in step (2), the additional condition is applied that $\mathbf{d}^T\mathbf{g}(\mathbf{x}_c + \alpha\mathbf{d}) = 0$, it becomes an *exact line search*. This, however, is an unnecessarily severe condition, because it can be shown that, if $\mathbf{d}^T[\mathbf{g}(\mathbf{x}_c + \alpha\mathbf{d}) - \mathbf{g}(\mathbf{x}_c)] > 0$, the approximation to the Hessian remains positive definite. The two terms in this expression are values of $\mathrm{d}S/\mathrm{d}\alpha$, and the condition states that S does not decrease more rapidly as α increases, which will always be true for some range of values of $\alpha > 0$. This algorithm is globally convergent; it *will* find a point at which the gradient vanishes, although that point may be a false minimum. It should be noted, however, that the procedure does not produce a final Hessian matrix whose inverse necessarily has any resemblance to a variance–covariance matrix. To get that it is necessary to compute $\boldsymbol{J}(\widehat{\mathbf{x}})$ and from it compute $(\boldsymbol{J}^T\boldsymbol{W}\boldsymbol{J})^{-1}$.

If a scaled, identity matrix is used as the initial approximation to $\boldsymbol{H}$, no use is made of the fact that, at least in the vicinity of the minimum, a major part of the Hessian matrix is given by $\boldsymbol{J}(\mathbf{x}_c)^T\boldsymbol{W}\boldsymbol{J}(\mathbf{x}_c)$. Thus, if there can be reasonable confidence in the general features of the model, it is useful to use $\boldsymbol{J}(\mathbf{x}_c)^T\boldsymbol{W}\boldsymbol{J}(\mathbf{x}_c)$ as the initial approximation to $\boldsymbol{H}$, in which case the first step is in the Gauss–Newton direction. The line-search provision, however, guarantees convergence, and, when n and p are both large, as in many crystallographic applications of least squares, a quasi-Newton update gives an adequate approximation to the new Hessian with a great deal less computation.

Another procedure that makes use of the fact that the linear approximation gives a major part of the Hessian constructs an approximation of the form

$$\boldsymbol{H}(\mathbf{x}) = \boldsymbol{J}(\mathbf{x})^T\boldsymbol{W}\boldsymbol{J}(\mathbf{x}) + \boldsymbol{B}(\mathbf{x}), \tag{8.1.4.12}$$

where $\boldsymbol{B}$ is generated by quasi-Newton methods. The quasi-Newton condition that must be satisfied in this case is

$$[\boldsymbol{J}(\mathbf{x}_+)^T\boldsymbol{W}\boldsymbol{J}(\mathbf{x}_+) + \boldsymbol{B}(\mathbf{x}_+)]\mathbf{d}_c = \mathbf{q}_c, \tag{8.1.4.13}$$

where $\mathbf{d}_c = \mathbf{x}_+ - \mathbf{x}_c$, and

$$\mathbf{q}_c = \boldsymbol{J}(\mathbf{x}_+)^T\boldsymbol{W}[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_+)] - \boldsymbol{J}(\mathbf{x}_c)^T\boldsymbol{W}[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_c)].$$

A formula based on the BFGS update (Nash & Sofer, 1995) is

$$\boldsymbol{B}(\mathbf{x}_+) = \boldsymbol{B}(\mathbf{x}_c) - \boldsymbol{G}_c\mathbf{d}_c\mathbf{d}_c^T\boldsymbol{G}_c/\mathbf{d}_c^T\boldsymbol{G}\mathbf{d}_c + \mathbf{q}_c\mathbf{q}_c^T/\mathbf{q}_c^T\mathbf{d}_c, \tag{8.1.4.14}$$

where $\boldsymbol{G}_c = \boldsymbol{J}(\mathbf{x}_+)^T\boldsymbol{W}\boldsymbol{J}(\mathbf{x}_+) + \boldsymbol{B}(\mathbf{x}_c)$. Since $\boldsymbol{J}(\mathbf{x}_+)$ and $\boldsymbol{M}(\mathbf{x}_+)$ must be computed anyway, this technique maximizes the use of information with little additional computing. The resulting approximation to the Hessian, however, may not be positive definite, and precautions must be taken to ensure convergence. In the vicinity of a correct solution, where the residuals are small, the addition of $\boldsymbol{B}$ is not likely to help much, and it can be dropped. Far from the solution, however, it can be very helpful. An implementation of this procedure has been described by Bunch, Gay & Welsch (1993); it appears to be at least as efficient as the trust region, Levenberg–Marquardt procedure, and is probably better when residuals are large.

In actual practice, it is not the Hessian matrix itself that is updated, but rather its Cholesky factor (Prince, 1994). This requires approximately the same number of operations and allows the solution of the linear system for computing $\mathbf{d}$ in a time that increases in proportion to p^2. This strategy also allows the use of the approximate Hessian in convergence checks with no significant computational overhead and no extra storage, as would be required for storing both the Hessian and its inverse.

8.1.4.4. *Stopping rules*

An iterative algorithm must contain some criterion for termination of the iteration process. This is a delicate part of all nonlinear optimization codes, and depends strongly on the scaling of the parameters. Although exceptions exist to almost all reasonable scaling rules, a basic principle is that a unit change in any variable should have approximately the same effect on the sum of squares. Thus, as discussed in Subsection 8.1.3.3 (equation 8.1.3.9), the ideal unit for each parameter is the standard uncertainty of its estimate, which can usually be adequately approximated by the reciprocal of the column norm of $\boldsymbol{J}$. In modern codes, the user has the option of supplying a diagonal scaling matrix whose elements are the reciprocals of some estimate of a typical 'significant' shift in the corresponding parameter.

In principle, the following conditions should hold when the convergence of a well scaled, least-squares procedure has reached its useful limit:

(1) $S(\mathbf{x}_c) = S(\widehat{\mathbf{x}})$;
(2) $\boldsymbol{J}(\mathbf{x}_c)^T\boldsymbol{W}[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_c)] = \boldsymbol{O}$;
(3) $\mathbf{x}_c = \widehat{\mathbf{x}}$.

The actual stopping rules must be chosen relative to the algorithm used (other conditions also exist) and the particular application. It is clear, however, that, since $\widehat{\mathbf{x}}$ is not known, the last two conditions cannot be used as stated. Also, these tests are dependent on the scaling of the problem, and the variables are not related to the sizes of the quantities involved. We present tests, therefore, that are relative error tests that take into account the scaling of the variables.

The general, relative error test is stated as follows. Two scalar quantities, a and b, are said to satisfy a relative error test with respect to a tolerence T if

$$\frac{|a-b|}{|a|} \leq T. \tag{8.1.4.15}$$

Roughly speaking, if T is of the form 10^{-q} then a and b agree to q digits. Obviously, there is a problem with this test if $a = 0$ and there will be numerical difficulties if a is close to 0. Thus, in practice, (8.1.4.15) is replaced by

$$|a-b| \leq (|a|+1)T, \tag{8.1.4.16}$$

which reduces to an absolute error test as $a \to 0$. A careful examination may be required to set this tolerance correctly, but, typically, if one of the fast, stable algorithms is used, only a few more iterations are necessary to get six or eight digits if one or two are already known. Note also that the actual value depends on ε, the relative machine precision. It is fruitless to seek more digits of accuracy than are expressed in the machine representation.

A test based on condition (1) is often implemented by using the linear approximation to $\boldsymbol{M}$ or the quadratic approximation to S. Thus, using the quadratic approximation to S, we can compute the predicted reduction by

$$\Delta_{\text{pred}} = S(\mathbf{x}_c) - \mathbf{d}^T\boldsymbol{J}^T\boldsymbol{W}[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_c)]. \tag{8.1.4.17}$$

Similarly, the actual reduction is

$$\Delta_{\text{act}} = S(\mathbf{x}_c) - S(\mathbf{x}_+). \tag{8.1.4.18}$$

The test then becomes $\Delta_{\text{pred}} \leq [1 + S(\mathbf{x}_c)]T$, $\Delta_{\text{act}} \leq [1 + S(\mathbf{x}_c)]T$, and $\Delta_{\text{act}} \leq 2\Delta_{\text{pred}}$. That is, we want both the predicted and actual reductions to be small and the actual reduction to agree reasonably well with the predicted reduction. A typical value for T should be 10^{-4}, although the value again depends on ε, and the user is cautioned not to make this tolerance too loose.

For a test on condition (2), we compute the cosine of the angle between the vector of residuals and the linear subspace spanned by the columns of $\mathbf{J}$,

$$\cos\zeta = \{\mathbf{d}^T\boldsymbol{J}^T\boldsymbol{W}[\mathbf{y} - \boldsymbol{M}(\mathbf{x}_+)]\}/[(\mathbf{d}^T\boldsymbol{J}^T\boldsymbol{W}\boldsymbol{J}\mathbf{d})S(\mathbf{x}_+)]^{1/2}. \tag{8.1.4.19}$$

The test is $\cos\zeta \leq T$, where, again, T should be 10^{-4} or smaller.

Test 3 above is usually only present to prevent the process from continuing when almost nothing is happening. Clearly, we do not know $\hat{\mathbf{x}}$, thus the test is typically that corresponding elements of $\mathbf{x}_+$ and $\mathbf{x}_c$ satisfy (8.1.4.16), where T is chosen to be 10^{-q}. A recommended value of q is half the number of digits carried in the computation, *e.g.* $q = 8$ for standard 64-bit (double-precision or 16 digit) calculations. Sometimes, the relative error test is of the form $|(\mathbf{x}_+)_j - (\mathbf{x}_c)_j|/\sigma_j \leq T$, where σ_j is the standard uncertainty computed from the inverse Hessian in the last iteration. Although this test has some statistical validity, it is quite expensive and usually not worth the work involved to compute.

8.1.4.5. *Recommendations*

One situation in which the Gauss–Newton algorithm behaves particularly poorly is in the vicinity of a saddle point in parameter space, where the true Hessian matrix is not positive definite. This occurs in structure refinement where a symmetric model is refined to convergence and then is replaced by a less symmetric model. The hypersurface of S will have negative curvature in a finite sized region of the parameter space for the less-symmetric model, and it is *essential* to use a safeguarded algorithm, one that incorporates a line search or a trust region, in order to get out of that region.

On the basis of this discussion, we can draw the following conclusions:

(1) In cases where the fit is poor, owing to an incomplete model or in the initial stages of refinement, methods based on the quadratic approximation to S (quasi-Newton methods) often perform better. This is particularly important when the model is close to a more symmetric configuration. These methods are more expensive per iteration and generally require more storage, but their greater stability in such problems usually justifies the cost.

(2) With small residual problems, where the model is complete and close to the solution, a safeguarded Gauss–Newton method is preferred. The trust-region implementation (Levenberg–Marquardt algorithm) has been very successful in practice.

(3) The best advice is to pick a good implementation of either method and stay with it.

8.1.5. Numerical methods for large-scale problems

Because the least-squares problems arising in crystallography are often very large, the methods we have discussed above are not always the most efficient. Some large problems have special structure that can be exploited to produce quite efficient algorithms. A particular special structure is sparsity, that is, the problems have Jacobian matrices that have a large fraction of their entries zero. Of course, not all large problems are sparse, so we shall also discuss approaches applicable to more general problems.

8.1.5.1. *Methods for sparse matrices*

We shall first discuss large, sparse, linear least-squares problems (Heath, 1984), since these techniques form the basis for nonlinear extensions. As we proceed, we shall indicate how the procedures should be modified in order to handle nonlinear problems. Recall that the problem is to find the minimum of the quadratic form $[\mathbf{y} - \boldsymbol{A}\mathbf{x}]^T\boldsymbol{W}[\mathbf{y} - \boldsymbol{A}\mathbf{x}]$, where $\mathbf{y}$ is a vector of observations, $\boldsymbol{A}\mathbf{x}$ represents a vector of the values of a set of linear model functions that predict the values of $\mathbf{y}$, and $\boldsymbol{W}$ is a positive-definite weight matrix. Again, for convenience, we make the transformation $\mathbf{y}' = \boldsymbol{U}\mathbf{y}$, where $\boldsymbol{U}$ is the upper triangular Cholesky factor of $\boldsymbol{W}$, and $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{A}$, so that the quadratic form becomes $(\mathbf{y}' - \boldsymbol{Z}\mathbf{x})^T(\mathbf{y}' - \boldsymbol{Z}\mathbf{x})$, and the minimum is the solution of the system of normal equations, $\boldsymbol{Z}^T\boldsymbol{Z}\mathbf{x} = \boldsymbol{Z}^T\mathbf{y}'$. Even if $\boldsymbol{Z}$ is sparse, it is easy to see that $\boldsymbol{H} = \boldsymbol{Z}^T\boldsymbol{Z}$ need not be sparse, because if even one row of $\boldsymbol{Z}$ has all of its elements nonzero, all elements of $\boldsymbol{H}$ will be nonzero. Therefore, the direct use of the normal equations may preclude the efficient exploitation of sparsity. But suppose $\boldsymbol{H}$ *is* sparse. The next step in solving the normal equations is to compute the Cholesky decomposition of $\boldsymbol{H}$, and it may turn out that the Cholesky factor is not sparse. For example, if $\boldsymbol{H}$ has the form

$$\boldsymbol{H} = \begin{pmatrix} x & x & x & x \\ x & x & 0 & 0 \\ x & 0 & x & 0 \\ x & 0 & 0 & x \end{pmatrix},$$

where x represents a nonzero element, then the Cholesky factor, $\boldsymbol{R}$, will not be sparse, but if

$$\boldsymbol{H} = \begin{pmatrix} x & 0 & 0 & x \\ 0 & x & 0 & x \\ 0 & 0 & x & x \\ x & x & x & x \end{pmatrix},$$

then $\boldsymbol{R}$ has the form

$$\boldsymbol{R} = \begin{pmatrix} x & 0 & 0 & x \\ 0 & x & 0 & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{pmatrix}.$$

These examples show that, although the sparsity of $\boldsymbol{R}$ is independent of the row ordering of $\boldsymbol{Z}$, the column order can have a profound effect. Procedures exist that analyse $\boldsymbol{Z}$ and select a permutation of the columns that reduces the 'fill' in $\boldsymbol{R}$. An algorithm for using the normal equations is then as follows:

(1) determine a permutation, $\boldsymbol{P}$ (an orthogonal matrix with one and only one 1 in each row and column, and all other elements zero), that tends to ensure a sparse Cholesky factor;
(2) store the elements of $\boldsymbol{R}$ in a sparse format;
(3) compute $\boldsymbol{Z}^T\boldsymbol{Z}$ and $\boldsymbol{Z}^T\mathbf{y}'$;
(4) factor $\boldsymbol{P}^T(\boldsymbol{Z}^T\boldsymbol{Z})\boldsymbol{P}$ to get $\boldsymbol{R}$;
(5) solve $\boldsymbol{R}^T\mathbf{z} = \boldsymbol{P}^T\boldsymbol{Z}^T\mathbf{y}'$;
(6) solve $\boldsymbol{R}\tilde{\mathbf{x}} = \mathbf{z}$;
(7) set $\hat{\mathbf{x}} = \boldsymbol{P}^T\tilde{\mathbf{x}}$.

This algorithm is fast, and it will produce acceptable accuracy if the condition number of $\boldsymbol{Z}$ is not too large. If extension to the nonlinear case is considered, it should be kept in mind that the first two steps need only be done once, since the sparsity pattern of the Jacobian does not, in general, change from iteration to iteration.

The QR decomposition of matrices that may be kept in memory is most often performed by the use of Householder transformations (see Subsection 8.1.1.1). For sparse matrices, or for matrices that are too large to be held in memory, this technique has several drawbacks. First, the method works by inserting zeros in the columns of $\boldsymbol{Z}$, working from left to right, but at each step it tends to fill in the columns to the right of the one currently being worked on, so that columns that are initially sparse cease to be so. Second, each Householder transformation needs to be applied to all of the remaining columns, so that the entire matrix must be retained in memory to make efficient use of this procedure.

The alternative procedure for obtaining the QR decomposition by the use of Givens rotations overcomes these problems if the entire upper triangular matrix, $\boldsymbol{R}$, can be stored in memory. Since this only requires about $p^2/2$ locations, it is usually possible. Also, it may happen that $\boldsymbol{R}$ has a sparse representation, so that even fewer locations will be needed. The algorithm based on Givens rotations is as follows:

(1) bring in the first p rows;
(2) find the QR decomposition of this $p \times p$ matrix;
(3) for $i = p + 1$ to n, do
  (*a*) bring in row i;
  (*b*) eliminate row i using $\boldsymbol{R}$ and at most p Givens rotations.

In order to specify how to use this algorithm to solve the linear least-squares problem, we must also state how to account for $\boldsymbol{Q}$. We could accumulate $\boldsymbol{Q}$ or save enough information to generate it later, but this usually requires excessive storage. The better alternatives are either to apply the steps of $\boldsymbol{Q}$ to $\mathbf{y}'$ as we proceed or to simply discard the information and solve $\boldsymbol{R}^T\boldsymbol{R}\mathbf{x} = \boldsymbol{Z}^T\mathbf{y}'$. It should be noted that the order of rows can make a significant difference. Suppose

$$\boldsymbol{Z} = \begin{pmatrix} x & 0 & 0 & 0 & 0 \\ 0 & x & 0 & 0 & 0 \\ 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & x & 0 \\ 0 & 0 & 0 & 0 & x \\ x & 0 & 0 & 0 & 0 \\ x & 0 & 0 & 0 & 0 \\ x & 0 & 0 & 0 & 0 \\ x & x & x & x & x \end{pmatrix}.$$

The work to complete the QR decomposition is of order p^2 operations, because each element below the main diagonal can be eliminated by one Givens rotation with no fill, whereas for

$$\boldsymbol{Z} = \begin{pmatrix} x & x & x & x & x \\ x & 0 & 0 & 0 & 0 \\ x & 0 & 0 & 0 & 0 \\ x & 0 & 0 & 0 & 0 \\ x & 0 & 0 & 0 & 0 \\ 0 & x & 0 & 0 & 0 \\ 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & x & 0 \\ 0 & 0 & 0 & 0 & x \end{pmatrix}$$

each Givens rotation fills an entire row, and the QR decomposition requires of order np^2 operations.

8.1.5.2. *Conjugate-gradient methods*

A numerical procedure that is applicable to large-scale problems that may not be sparse is called the *conjugate-gradient method*. Conjugate-gradient methods were originally designed to solve the quadratic minimization problem, find the minimum of

$$S(\mathbf{x}) = (1/2)\mathbf{x}^T\boldsymbol{H}\mathbf{x} - \mathbf{b}^T\mathbf{x}, \tag{8.1.5.1}$$

where $\boldsymbol{H}$ is a symmetric, positive-definite matrix. The gradient of S is

$$\mathbf{g}(\mathbf{x}) = \boldsymbol{H}\mathbf{x} - \mathbf{b}, \tag{8.1.5.2}$$

and its Hessian matrix is $\boldsymbol{H}$. Given an initial estimate, $\mathbf{x}_0$, the conjugate-gradient algorithm is

(1) define $\mathbf{d}_0 = -\mathbf{g}(\mathbf{x}_0)$;
(2) for $k = 0, 1, 2, ..., p - 1$, $\mathbf{d}_0$
  (*a*) $\alpha_k = -\mathbf{d}_k^T\mathbf{g}(\mathbf{x}_k)/\mathbf{d}_k^T\boldsymbol{H}\mathbf{d}_k$;
  (*b*) $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$;
  (*c*) $\gamma_k = \mathbf{g}(\mathbf{x}_{k+1})^T\mathbf{g}(\mathbf{x}_{k+1})/\mathbf{g}(\mathbf{x}_k)^T\mathbf{g}(\mathbf{x}_k)$;
  (*d*) $\mathbf{d}_{k+1} = -\mathbf{g}(\mathbf{x}_k) + \gamma_k\mathbf{d}_k$.

This algorithm finds the exact solution for the quadratic function in not more than p steps.

This algorithm cannot be used directly for the nonlinear case because it requires $\boldsymbol{H}$ to compute α_k, and the goal is to solve the problem without computing the Hessian. To accomplish this, the exact computation of α is replaced by an actual line search, and the termination after at most p steps is replaced by a convergence test. Thus, we obtain, for a given starting value $\mathbf{x}_0$ and a general, nonquadratic function S:

(1) define $\mathbf{d}_0 = -\mathbf{g}(\mathbf{x}_0)$;
(2) set $k = 0$;
(3) do until convergence
  (*a*) $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$, where α is chosen by a line search;
  (*b*) $\gamma_k = \mathbf{g}(\mathbf{x}_{k+1})^T\mathbf{g}(\mathbf{x}_{k+1})/\mathbf{g}(\mathbf{x}_k)^T\mathbf{g}(\mathbf{x}_k)$;
  (*c*) $\mathbf{d}_{k+1} = -\mathbf{g}(\mathbf{x}_{k+1}) + \gamma_k\mathbf{d}_k$;
  (*d*) $k = k + 1$.

Note that, as promised, $\boldsymbol{H}$ is not needed. In practice, it has been observed that the line search need not be exact, but that

periodic restarts in the steepest-descent direction are often helpful. This procedure often requires more iterations and function evaluations than methods that store approximate Hessians, but the cost per iteration is small. Thus, it is often the overall least-expensive method for large problems.

For the least-squares problem, recall that we are finding the minimum of

$$S(\mathbf{x}) = (1/2)[\mathbf{y}' - \mathbf{Z}\mathbf{x}]^T[\mathbf{y}' - \mathbf{Z}\mathbf{x}], \tag{8.1.5.3}$$

for which

$$\mathbf{g}(\mathbf{x}) = \mathbf{Z}^T(\mathbf{Z}\mathbf{x} - \mathbf{y}'). \tag{8.1.5.4}$$

By using these definitions in the conjugate-gradient algorithm, it is possible to formulate a specific algorithm for linear least squares that requires only the calculation of $\boldsymbol{Z}$ times a vector and $\boldsymbol{Z}^T$ times a vector, and never requires the calculation or factorization of $\boldsymbol{Z}^T\boldsymbol{Z}$.

In practice, such an algorithm will, due to roundoff error, sometimes require more than p iterations to reach a solution. A detailed examination of the performance of the procedure shows, however, that fewer than p iterations will be required if the eigenvalues of $\boldsymbol{Z}^T\boldsymbol{Z}$ are bunched, that is, if there are sets of multiple eigenvalues. Specifically, if the eigenvalues are bunched into k distinct sets, then the conjugate-gradient method will converge in k iterations. Thus, significant improvements can be made if the problem can be transformed to one with bunched eigenvalues. Such a transformation leads to the so-called *preconditioned conjugate-gradient method*. In order to analyse the situation, let $\boldsymbol{C}$ be a $p \times p$ matrix that transforms the variables, such that

$$\mathbf{x}' = \boldsymbol{C}\mathbf{x}. \tag{8.1.5.5}$$

Then,

$$\mathbf{y}' - \mathbf{Z}\mathbf{x} = \mathbf{y}' - \boldsymbol{Z}\boldsymbol{C}^{-1}\mathbf{x}'. \tag{8.1.5.6}$$

Therefore, $\boldsymbol{C}$ should be such that the system $\boldsymbol{C}\mathbf{x} = \mathbf{x}'$ is easy to solve, and $(\boldsymbol{Z}\boldsymbol{C}^{-1})^T\boldsymbol{Z}\boldsymbol{C}^{-1}$ has bunched eigenvalues. The ideal choice would be $\boldsymbol{C} = \boldsymbol{R}$, where $\boldsymbol{R}$ is the upper triangular factor of the QR decomposition, since $\boldsymbol{Z}\boldsymbol{R}^{-1} = \boldsymbol{Q}_{\boldsymbol{Z}}$. $\boldsymbol{Q}_{\boldsymbol{Z}}^T\boldsymbol{Q}_{\boldsymbol{Z}} = \boldsymbol{I}$ has all of its eigenvalues equal to one, and, since $\boldsymbol{R}$ is triangular, the system is easy to solve. If $\boldsymbol{R}$ were known, however, the problem would already be exactly solved, so this is not a useful alternative. Unfortunately, no universal best choice seems to exist, but one approach is to choose a sparse approximation to $\boldsymbol{R}$ by ignoring rows that cause too much fill in or by making $\boldsymbol{R}$ a diagonal matrix whose elements are the Euclidean norms of the columns of $\boldsymbol{Z}$. Bear in mind that, in the nonlinear case, an expensive computation to choose $\boldsymbol{C}$ in the first iteration may work very well in subsequent iterations with no further expense. One should be aware of the trade off between the extra work per iteration of the preconditioned-conjugate gradient method *versus* the reduction in the number of iterations. This is especially important in nonlinear problems.

The solution of large, least-squares problems is currently an active area of research, and we have certainly not given an exhaustive list of methods in this chapter. The choice of method or approach for any particular problem is dependent on many conditions. Some of these are:

(1) The size of the problem. Clearly, as computer memories continue to grow, the boundary between small and large problems also grows. Nevertheless, even if a problem can fit into memory, its sparsity structure may be exploited in order to obtain a more efficient algorithm.

(2) The number of times the problem (or similar ones) will be solved. If it is a one-shot problem (a rare occurrence), then one is usually most strongly influenced by easy-to-use, existing software. Exceptions, of course, exist where even a single solution of the problem requires extreme care.

(3) The expense of evaluating the function. With a complicated, nonlinear function like the structure-factor formula, the computational effort to determine the values of the function and its derivatives usually greatly exceeds that required to solve the linearized problem. Therefore, a full Gauss–Newton, trust-region, or quasi-Newton method may be warranted.

(4) Other structure in the problem. Rarely does a problem have a random sparsity pattern. Non-zero values usually occur in blocks or in some regular pattern for which special decomposition methods can be devised.

(5) The machine on which the problem is to be solved. We have said nothing about the existing vector and parallel processors. Suffice it to say that the most efficient procedure for a serial machine may not be the right algorithm for one of these novel machines. Appropriate numerical methods for such architectures are also being actively investigated.

8.1.6. Orthogonal distance regression

It is often useful to consider the data for a least-squares problem to be in the form (t_i, y_i), $i = 1, \ldots, n$, where the t_i are considered to be the *independent* variables and the y_i the dependent variables. The implicit assumption in ordinary least squares is that the independent variables are known exactly. It sometimes occurs, however, that these independent variables also have errors associated with them that are significant with respect to the errors in the observations y_i. In such cases, referred to as 'errors in variables' or 'measurement error models', the ordinary least-squares methodology is not appropriate and its use may give misleading results (see Fuller, 1987).

Let us define $\hat{M}(t_i, \mathbf{x})$ to be the model functions that predict the y_i. Observe that ordinary least squares minimizes the sum of the squares of the vertical distances from the observed points y_i to the curve $\hat{M}(t, \mathbf{x})$. If t_i has an error δ_i, and these errors are normally distributed, then the maximum-likelihood estimate of the parameters is found by minimizing the sum of the squares of the *weighted orthogonal distances* from the point y_i to the curve $\hat{M}(t, \mathbf{x})$. More precisely, the optimization problem to be solved is given by

$$\min_{\mathbf{x},\delta} \sum_{i=1}^{n} \left\{ \left[y_i - \hat{M}(t_i + \delta_i, \mathbf{x})\right]^T \mathbf{W}_{\mathbf{y}} \left[y_i - \hat{M}(t_i + \delta_i, \mathbf{x})\right] + \delta_i^T \mathbf{W}_{\mathbf{t}} \delta_i \right\}, \tag{8.1.6.1}$$

where $\mathbf{W}_{\mathbf{y}}$ and $\mathbf{W}_{\mathbf{t}}$ are appropriately chosen weights. Problem (8.1.6.1) is called the *orthogonal distance regression* (ODR) problem. Problem (8.1.6.1) can be solved as a least-squares problem in the combined variables $\mathbf{x}, \delta$ by the methods given above. This, however, is quite inefficient, since such a procedure would not exploit the special structure of the ODR problem. Few algorithms that exploit this structure exist; one has been given by Boggs, Byrd & Schnabel (1987), and the software, called *ODRPACK*, is by Boggs, Byrd, Donaldson & Schnabel (1989). The algorithm is based on the trust-region (Levenberg–Marquardt) method described above, but it exploits the special structure of (8.1.6.1) so that the cost of each iteration is no more expensive than the cost of a similar iteration of the corresponding

ordinary least-squares problems. For a discussion of some of the statistical properties of the resulting estimates, including a procedure for the computation of the variance–covariance matrix, see Boggs & Rogers (1990).

8.1.7. Software for least-squares calculations

Giving even general recommendations on software is a difficult task for several reasons. Clearly, the selection of methods discussed in earlier sections contains implicitly some recommendations for approaches. Among the reasons for avoiding specifics are the following:

(1) Assessing differences in performance among various codes requires a detailed knowledge of the criteria the developer of a particular code used in creating it. A program written to emphasize speed on a certain class of problems on a certain machine is impossible to compare directly with a program written to be very reliable on a wide class of problems and portable over a wide range of machines. Other measures, including ease of maintenance and modification and ease of use, and other design criteria, such as interactive *versus* batch, stand alone *versus* user-callable, automatic computation of related statistics *versus* no statistics, and so forth, make the selection of software analogous to the selection of a car.

(2) Choosing software requires detailed knowledge of the needs of the user and the resources available to the user. Considerations such as problem size, machine size, machine architecture and financial resources all enter into the decision of which software to obtain.

(3) A software recommendation made on the basis of today's knowledge ignores the fact that algorithms continue to be invented, and old algorithm continue to be rethought in the light of new developments and new machine architectures. For example, when vector processors first appeared, algorithms for sparse-matrix calculations were very poor at exploiting this capability, and it was thought that these new machines were simply not appropriate for such calculations. Now, however, recent methods for sparse matrices have achieved a high degree of vectorization. For another example, early programs for crystallographic, full-matrix, least-squares refinement spent a large fraction of the time building the normal-equations matrix. The matrix was then inverted using a procedure called Gaussian elimination, which does not exploit the fact that the matrix is positive definite. Some programs were later converted to use Cholesky decomposition, which is at least twice as fast, but many were not because the inversion process took a small fraction of the total time. Linear algebra, however, is readily adaptable to vector and parallel machines, and procedures such as QR factorization are extremely fast, while the calculation of structure factors, with its repeated evaluations of trigonometric functions, becomes the time-controlling step.

The general recommendation is to analyse carefully the needs and resources in terms of these considerations, and to seek expert assistance whenever possible. *As much as possible, avoid the temptation to write your own codes*. Despite the fact that the quality of existing software is far from uniformly high, the benefits of utilizing high-quality software generally far outweigh the costs of finding, obtaining, and installing it.

Sources of information on software have improved significantly in the past several years. Nevertheless, the task of identifying software in terms of problems that can be solved; organizing, maintaining and updating such a list; and informing the user community still remains formidable.

A current, problem-oriented system that includes both a problem classification scheme and a network tool for obtaining documentation and source code (for software in the public domain) is the *Guide to Available Mathematical Software* (GAMS). This system is maintained by the National Institute of Standards and Technology (NIST) and is continually being updated as new material is received. It gives references to software in several software repositories; the URL is http://math.nist.gov/gams.

where $A_g = U_g^{\rm eff} t$, and J_0 is the Bessel function of zero order. For a small gap, the intensity is proportional to $|U^{\rm eff}|^2$. By many-beam calculations, Gjønnes & Bøe (1994) showed the integrated intensities to be less sensitive to dynamical interactions along the row than that indicated from the Bethe potentials, and that relative intensities are fairly independent of thickness. Coordinate refinement based on intensities from a few high-order Kossel-line segments appear to produce accuracies roughly one order of magnitude poorer than good single-crystal X-ray determination. This may suggest that if some form of three-dimensional intensity data could be collected in electron diffraction the same level of accuracies as with X-rays may be attainable – which, however, remains to be seen.

References

8.1

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S. & Sorenson, D. (1992). *LAPACK user's guide*, 2nd ed. Philadelphia: SIAM Publications.

Berger, J. O. & Wolpert, R. L. (1984). *The likelihood principle*. Hayward, CA: Institute of Mathematical Statistics.

Boggs, P. T., Byrd, R. H., Donaldson, J. R. & Schnabel, R. B. (1989). *ODRPACK – software for weighted orthogonal distance regression. ACM Trans. Math. Softw.* **15**, 348–364.

Boggs, P. T., Byrd, R. H. & Schnabel, R. B. (1987). *A stable and efficient algorithm for nonlinear orthogonal distance regression. SIAM J. Sci. Stat. Comput.* **8**, 1052–1078.

Boggs, P. T. & Rogers, J. E. (1990). *Orthogonal distance regression. Contemporary mathematics: statistical analysis of measurement error models and applications*. Providence, RI: AMS.

Box, G. E. P., Hunter, W. G. & Hunter, J. S. (1978). *Statistics for experimenters: an introduction to design, data analysis and model building*. New York: John Wiley.

Box, G. E. P. & Tiao, G. C. (1973). *Bayesian inference in statistical analysis*. Reading, MA: Addison-Wesley.

Bunch, D. S., Gay, D. M. & Welsch, R. E. (1993). *Algorithm 717: subroutines for maximum likelihood and quasi-likelihood estimation of parameters in nonlinear regression models. ACM Trans. Math. Softw.* **19**, 109–130.

Dennis, J. E. & Schnabel, R. B. (1983). *Numerical methods for unconstrained optimization and nonlinear equations*. Englewood Cliffs, NJ: Prentice Hall.

Donaldson, J. R. & Schnabel, R. B. (1986). *Computational experience with confidence regions and confidence intervals for nonlinear least squares. Computer science and statistics. Proceedings of the Seventeenth Symposium on the Interface*, edited by D. M. Allen, pp. 83–91. New York: North-Holland.

Draper, N. & Smith, H. (1981). *Applied regression analysis*. New York: John Wiley.

Fedorov, V. V. (1972). *Theory of optimal experiments*, translated by W. J. Studden & E. M. Klimko. New York: Academic Press.

Fuller, W. A. (1987). *Measurement error models*. New York: John Wiley & Sons.

Heath, M. T. (1984). *Numerical methods for large, sparse, linear least squares problems. SIAM J. Sci. Stat. Comput.* **5**, 497–513.

Nash, S. & Sofer, A. (1995). *Linear and nonlinear programming*. New York: McGraw-Hill.

Prince, E. (1994). *Mathematical techniques in crystallography and materials science*, 2nd ed. Berlin: Springer.

Schwarzenbach, D., Abrahams, S. C., Flack, H. D., Prince, E. & Wilson, A. J. C. (1995). *Statistical descriptors in crystallography. II. Report of a Working Group on Expression of Uncertainty in Measurement. Acta Cryst.* A**51**, 565–569.

Stewart, G. W. (1973). *Introduction to matrix computations*. New York: Academic Press.

8.2

Belsley, D. A., Kuh, E. & Welsch, R. E. (1980). *Regression diagnostics*. New York: John Wiley.

Box, G. E. P. & Tiao, G. C. (1973). *Bayesian inference in statistical analysis*. Reading, MA: Addison-Wesley.

Collins, D. M. (1982). *Electron density images from imperfect data by iterative entropy maximization. Nature (London)*, **298**, 49–51.

Collins, D. M. (1984). *Scaling by entropy maximization. Acta Cryst.* A**40**, 705–708.

Hoaglin, D. C., Mosteller, M. & Tukey, J. W. (1983). *Understanding robust and exploratory data analysis*. New York: John Wiley.

Huber, P. J. (1973). *Robust regression: asymptotics, conjectures and Monte Carlo. Ann. Stat.* **1**, 799–821.

Huber, P. J. (1981). *Robust statistics*. New York: John Wiley.

Jaynes, E. T. (1979). *Where do we stand on maximum entropy? The maximum entropy formalism,* edited by R. D. Liven & M. Tribus, pp. 44–49. Cambridge, MA: Massachusetts Institute of Technology.

Livesey, A. K. & Skilling, J. (1985). *Maximum entropy theory. Acta Cryst.* A**41**, 113–122.

Nicholson, W. L., Prince, E., Buchanan, J. & Tucker, P. (1982). *A robust/resistant technique for crystal structure refinement. Crystallographic statistics: progress and problems,* edited by S. Rameseshan, M. F. Richardson & A. J. C. Wilson, pp. 220–263. Bangalore: Indian Academy of Sciences.

Rietveld, H. M. (1969). *A profile refinement method for nuclear and magnetic structures. J. Appl. Cryst.* **2**, 65–71.

Shore, J. E. & Johnson, R. W. (1980). *Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. IEEE Trans. Inf. Theory*, IT-**26**, 26–37; correction: IT-**29**, 942–943.

Tukey, J. W. (1974). *Introduction to today's data analysis. Critical evaluation of chemical and physical structural information,* edited by D. R. Lide & M. A. Paul, pp. 3–14. Washington: National Academy of Sciences.

Wilson, A. J. C. (1976). *Statistical bias in least-squares refinement. Acta Cryst.* A**32**, 994–996.