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

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

Similarly, the actual reduction is

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

The test then becomes $\Delta_{\rm pred} \leq [1 + S(\mathbf{x}_c)]T$, $\Delta_{\rm act} \leq [1 + S(\mathbf{x}_c)]T$, and $\Delta_{\rm act} \leq 2\Delta_{\rm pred}$. 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 $\widehat{\mathbf{x}}$, 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

references