(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

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

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

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

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

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

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

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

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

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

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

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

[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

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

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

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

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,

references