An Adaptive Nonlinear Least-Squares Algorithm

JOHN E. DENNIS, JR.
Rice University
and
DAVID M. GAY and ROY E. WELSCH
Massachusetts Institute of Technology

NL2SOL is a modular program for solving nonlinear least-squares problems that incorporates a number of novel features. It maintains a secant approximation S to the second-order part of the least-squares Hessian and adaptively decides when to use this approximation. S is "sized" before updating, something that is similar to Oren–Luenberger scaling. The step choice algorithm is based on minimizing a local quadratic model of the sum of squares function constrained to an elliptical trust region centered at the current approximate minimizer This is accomplished using ideas discussed by Moré, together with a special module for assessing the quality of the step thus computed. These and other ideas behind NL2SOL are discussed, and its evolution and current implementation are also described briefly.

Key Words and Phrases: unconstrained optimization, nonlinear least squares, nonlinear regression, quasi-Newton methods, secant methods
CR Categories: 5 14, 5.5
The Algorithm: NL2SOL: An Adaptive Nonlinear Least-Squares Algorithm. *ACM Trans. Math. Softw.* 7, 3(Sept 1981), 348–368

1. INTRODUCTION

This project began in order to meet a need for a nonlinear least-squares algorithm which, in the large residual case, would be more reliable than the Gauss–Newton or Levenberg–Marquardt method [15] and more efficient than the secant or variable metric algorithms [17], such as the Davidon–Fletcher–Powell method, which are intended for general function minimization.

We have developed a satisfactory computer program called NL2SOL based on ideas in [18], and our primary purpose here is to report the details and to give

This work was supported in part by National Science Foundation Grants DCR75-10143, MCS76-00324, and SOC76-14311 to the National Bureau of Economic Research, Inc, and MCS79-06671 to the Massachusetts Institute of Technology, and was sponsored in part by NSF Grant MCS78-09525 and United States Army Contract DAAG29-75-C-0024 to the Mathematics Research Center at the University of Wisconsin-Madison.
Authors' addresses: J E Dennis, Department of Mathematical Sciences, Rice University, P.O. Box 1892, Houston, TX 77001; D.M. Gay, M.I.T./CCREMS, Room E38-278, Cambridge, MA 02139; R.E. Welsch, M.I.T./CCREMS, Room E53-383, Cambridge, MA 02139.

some test results. On the other hand, we learned so much during the development that seems likely to be applicable in the development of other algorithms that we have chosen to expand our exposition to include some of this experience.

In Section 2 we set out the problem and the notation we intend to use. Section 3 deals with our way of supplementing the classical Gauss–Newton approximation to the least-squares Hessian by various analogs of the Davidon–Fletcher–Powell method. Section 4 briefly describes our interpretation of the Oren–Luenburger [33] sizing strategy for this augmentation. In Section 5 we describe our adaptive quadratic modeling of the objective function. Section 6 contains a discussion of the stopping criteria and covariance matrices. Section 7 contains test results, and Section 8 discusses the size of NL2SOL and the time it takes for housekeeping. The NL2SOL Usage Summary is included in the accompanying algorithm.

2. THE NONLINEAR LEAST-SQUARES PROBLEM

There are good reasons for numerical analysts to study least-squares problems. In the first place, they are a computation of primary importance in statistical data analysis and hence in the social sciences, as well as in the more traditional areas within the physical sciences. Thus a computer algorithm able to deal efficiently with both sorts of data is widely applicable.

Although applicability should always constitute sufficient justification to tackle a problem, in this case there is also an opportunity for more far-reaching progress in numerical optimization. In order to be more specific, it will be useful to have a formal statement of the nonlinear least-squares problem.

We adopt notation consistent with fitting a model to n pieces of data using p parameters: Given $R: \mathbb{R}^p \rightarrow \mathbb{R}^n$, we wish to solve the unconstrained minimization problem

$$\min f(x) = \frac{1}{2} R(x)^{\mathrm{T}} R(x) = \frac{1}{2} \sum_{i=1}^{n} r_i(x)^2. \tag{2.1}$$

Notice that if $J(x) = R'(x) = (\partial_j r_i(x))$, then the gradient of f is

$$\nabla f(x) = J(x)^{\mathrm{T}} R(x) \tag{2.2}$$

and the Hessian of f is

$$\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x) + \sum_{i=1}^{n} r_i(x) \nabla^2 r_i(x). \tag{2.3}$$

Since we are seeking a minimum of f, we wish to have $f(x^*) = 0$, an obviously global minimum; in the more realistic case where f is not anywhere near zero, we will be forced to terminate on small parameter changes or to use some other convergence criteria (see Section 6). It is clear from (2.2) that $\nabla f(x^*) = 0$ and $R(x^*) \neq 0$ corresponds to $R(x^*) \perp C(J(x^*))$, the column space of $J(x^*)$. Thus it is essential as the iteration proceeds that $C(J(x_k))$ be approximated very well in the usual case where $p < n$ and $R(x^*) \neq 0$.

In addition to making a precise convergence test possible, having an accurate Jacobian matrix means that a good approximation to a portion of the Hessian is available as a by-product of the gradient computation. In fact, it is often possible to ignore the second-order term $\Sigma r_i(x)\nabla^2 r_i(x)$ of the Hessian altogether on the grounds that if the nonzero residuals are not of the sort that reinforce their nonlinearity [41-43, 15], then $J(x)^T J(x)$ is a sufficiently good Hessian approximation. In the resulting Gauss-Newton method, the "Newton step" from x_k is defined by the linear system of equations

$$J(x_k)^T J(x_k) s_k = -J(x_k)^T R(x_k). \tag{2.4}$$

Since (2.4) is the system of normal equations for the linear least-squares problem

$$\min_s (J(x_k)s + R(x_k))^T (J(x_k)s + R(x_k)), \tag{2.5}$$

it is better to obtain s_k from a QR decomposition of $J(x_k)$ (see [27]).

We can view (2.5) as defining a quadratic model in $x = x_k + s$ of the least-squares criterion function (2.1):

$$\begin{aligned} q_k^G(x) = {} & \tfrac{1}{2} R(x_k)^T R(x_k) + (x - x_k)^T J(x_k)^T R(x_k) \\ & + \tfrac{1}{2}(x - x_k)^T J(x_k)^T J(x_k)(x - x_k). \end{aligned} \tag{2.6}$$

From (2.1)-(2.3) we see that the difference between this Gauss-Newton model and the usual Newton model obtained from a quadratic Taylor expansion around x_k is just the term $\frac{1}{2}(x - x_k)^T[\Sigma r_i(x_k)\nabla^2 r_i(x_k)](x - x_k)$.

The conceptual difference between these two models is interesting in that it exposes some reasons for the deficiencies of the Gauss-Newton algorithm. The Newton model is based on the assumption that f can be adequately modeled by a quadratic, while the Gauss-Newton model (2.6) is shown by (2.5) to result from the stronger assumption that R can be adequately modeled by an affine function.

3. AN AUGMENTATION OF THE GAUSS-NEWTON HESSIAN

Our purpose in this section is to suggest a way to augment the Gauss-Newton model (2.6) by adding an approximation to the difference between it and the quadratic Taylor expansion to obtain

$$\begin{aligned} q_k^S(x) = {} & \tfrac{1}{2} R(x_k)^T R(x_k) + (x - x_k)^T J(x_k)^T R(x_k) \\ & + \tfrac{1}{2}(x - x_k)^T [J(x_k)^T J(x_k) + S_k](x - x_k). \end{aligned} \tag{3.1}$$

We suggest an approximation rule for S_k that is simple, general, and geometric. The approach is to decide on a set of desirable characteristics for the approximant and then to select S_{k+1} to be the nearest such feasible point to S_k. The rationale is that every point in the feasible set incorporates equally well the new information gained at x_{k+1} and that taking the nearest point (in a sense to be explained later) corresponds to destroying as little of the information stored in S_k as possible.

Currently we begin with $S_0 = 0$, since this is both cheap and reasonable in the sense that $q_0^S = q_0^G$. Suppose S_k is available. First let us decide on the properties S_{k+1} should have. Remember that it is to approximate $\Sigma r_i(x_{k+1})\nabla^2 r_i(x_{k+1})$ and so it should obviously be symmetric. It is easy to find examples where the term to be approximated is indefinite, so we reject any restriction on the eigenvalues of S_{k+1}. Finally, we want to incorporate the new information about the problem,

J_{k+1} and R_{k+1}, into S_{k+1}. The standard way to do this is to ask the second-order approximant to transform the current x-change into the observed first-order change, that is,

$$\begin{aligned} S_{k+1}\Delta x_k &= \Sigma r_i(x_{k+1})\nabla^2 r_i(x_{k+1})\Delta x_k \\ &\doteq \Sigma r_i(x_{k+1})(\nabla r_i(x_{k+1}) - \nabla r_i(x_k)) \\ &= J_{k+1}^{\mathrm{T}} R_{k+1} - J_k^{\mathrm{T}} R_{k+1} =: y_k. \end{aligned} \tag{3.2}$$

It is perhaps worth noting in passing that we tested several choices for y_k, including the Broyden-Dennis [14] choice $J_{k+1}^{\mathrm{T}}R_{k+1} - J_k^{\mathrm{T}}R_k - J_{k+1}^{\mathrm{T}}J_{k+1}\Delta x_k$ and the Betts [7] choice $J_{k+1}^{\mathrm{T}}R_{k+1} - J_k^{\mathrm{T}}R_k - J_k^{\mathrm{T}}J_k\Delta x_k$. Happily, (3.2), which makes more use of the structure of the problem, was the slight but clear winner. In summary, we choose $S_0 = 0$, $S_{k+1} \in Q := \{S{:}S = S^{\mathrm{T}} \text{ and } S\Delta x_k = y_k\}$.

Our choice of S_{k+1} from Q is made in analogy with the DFP method for unconstrained minimization [17]. Before giving the formula and its properties, we review some useful notation.

If A is any real matrix, then the Frobenius norm of A is $\| A \|_{\mathrm{F}} := (\Sigma A_{ij}^2)^{1/2}$. If B is any symmetric positive definite matrix, then it has a symmetric, positive definite square root, $B^{1/2}$. Define $\| A \|_{\mathrm{F},B} := \| B^{-1/2}AB^{-1/2} \|_{\mathrm{F}}$. This weighted Frobenius norm is a natural analog of the Frobenius norm for a matrix when the standard inner product norm on the domain is replaced by $\| x \|_B = (x^{\mathrm{T}}Bx)^{1/2}$, because $\| A \|_{\mathrm{F},B}^2 = \Sigma \| Av_j \|_B^2$ for any set of vectors $v_1, v_2, \ldots, v_p$ orthonormal in the inner product on $\mathbb{R}^p$ defined by $(x, y) := x^{\mathrm{T}}By$. The following theorem gives the update formulas as well as their defining properties. It is just a restatement of [17, Theorem 7.3].

THEOREM 3.1. *Let $v^T\Delta x_k > 0$. Then for any positive definite symmetric matrix H for which $H\Delta x_k = v$,*

$$\min \| S - S_k \|_{F,H} \quad \text{for} \quad S \in Q$$

is solved by

$$S_{k+1} = S_k + \frac{(y_k - S_k\Delta x_k)v^T + v(y_k - S_k\Delta x_k)^T}{\Delta x_k^T v} - \frac{\Delta x_k^T(y_k - S_k\Delta x_k)vv^T}{(\Delta x_k^T v)^2}.$$

In NL2SOL we compute S_{k+1} corresponding to $v = \Delta g_k = J_{k+1}^{\mathrm{T}}R_{k+1} - J_k^{\mathrm{T}}R_k$. This corresponds to weighting the change by any positive definite symmetric matrix that sends Δx_k to Δg_k. Thus we hope the metric being used is not too different from that induced by the natural scaling of the problem.

4. SIZING THE HESSIAN AUGMENTATION

It is well known by now that the update methods do not generate approximations that become arbitrarily accurate as the iteration proceeds. On the other hand, we know that for zero residual problems, S_k should ideally converge to zero and that if it does not at least become small in those cases, then the augmented model (3.1) cannot hope to compete with (2.6), the Gauss-Newton model.

The crux of the problem can be seen by observing that even if R_{k+1} happened to be zero and even if y_k defined by (3.2) were used to make the update to S_k, then $S_{k+1}\Delta x_k = y_k = 0$, but S_{k+1} would be the same as S_k on the orthogonal complement of $\{\Delta x_k, v\}$.

We use a straightforward modification of the Oren-Luenburger *self-scaling* technique [33]. The idea is to update $\tau_k S_k$ rather than S_k to get S_{k+1}. The scalar τ_k is chosen to try to shift the spectrum of S_k in hopes that the spectrum of $\tau_k S_k$ will overlap that of the second-order term we are approximating. We could take the scalar to be

$$\frac{\Delta x_k^T y_k}{\Delta x_k^T S_k \Delta x_k} \doteq \left[\frac{\Delta x_k^T[\Sigma r_i(x_{k+1})\nabla^2 r_i(x_{k+1})]\Delta x_k}{\Delta x_k^T \Delta x_k}\right]\left[\frac{\Delta x_k^T S_k \Delta x_k}{\Delta x_k^T \Delta x_k}\right]^{-1}.$$

We prefer to call this *sizing*, and since we are primarily concerned with S_k being too large, we actually take

$$\tau_k = \min\{|\Delta x_k^T y_k|/|\Delta x_k^T S_k \Delta x_k|, 1\}. \tag{4.1}$$

Whatever this strategy is called, notice that when $R_{k+1} = 0$, our $y_k = 0$, and so $\tau_k = 0$ and $S_{k+1} = 0$. The use of sizing factor (4.1) made a significant difference in the performance of the algorithm. (See Table IV.)

5. ADAPTIVE QUADRATIC MODELING

In Section 3 we noted that $S_0 = 0$, which means that the augmented model (3.1) is initially equal to the Gauss-Newton model (2.6). Tests have shown that often $q_k^G(x_{k+1})$ predicts $f(x_{k+1})$ better than $q_k^S(x_{k+1})$ for small k; so it seems useful to have some way to decide which model to use to determine the step.

Betts [7] also starts with $S_0 = 0$ and takes Gauss-Newton steps for at least p iterations and until Δx_k is small enough to make it likely that x_{k+1} is near x^*. It seems therefore as though his aim is to make a last few refining iterations based on the augmented Hessian. The heuristic we use in NL2SOL usually uses the augmented Hessian much sooner. This heuristic is intimately connected with our choice of Δx_k.

NL2SOL uses a model/trust-region strategy to pick Δx_k. The step is of the form

$$\Delta x_k = -(H_k + \lambda_k D_k^2)^{-1}\nabla f(x_k), \tag{5.1}$$

where H_k is the current Hessian approximation, D_k is a diagonal scaling matrix discussed more in Section 7, and $\lambda_k \geq 0$ is chosen by the safeguarded Reinsch [39] iteration as in [31], with the case of near singularity in $H_k + \lambda_k D_k^2$ handled as in [24]. The important thing is the idea of having at x_k a local quadratic model q_k of f and an estimate of a region in which q_k is trusted to represent f. The next point, x_{k+1}, is chosen to approximately minimize q_k in this region or to minimize q_k in an approximation to this region. In either case, the information gained about f at x_{k+1} is then used to update the model and also to update the size or shape of the trust region.

We begin with the assumption that q_0^G holds globally. Since the trust region revision is always based on the length of the step just taken, this causes the radius to be set automatically by the initial Gauss-Newton step. This scheme often works well, but it can have problems. If the Gauss-Newton step is too long, the

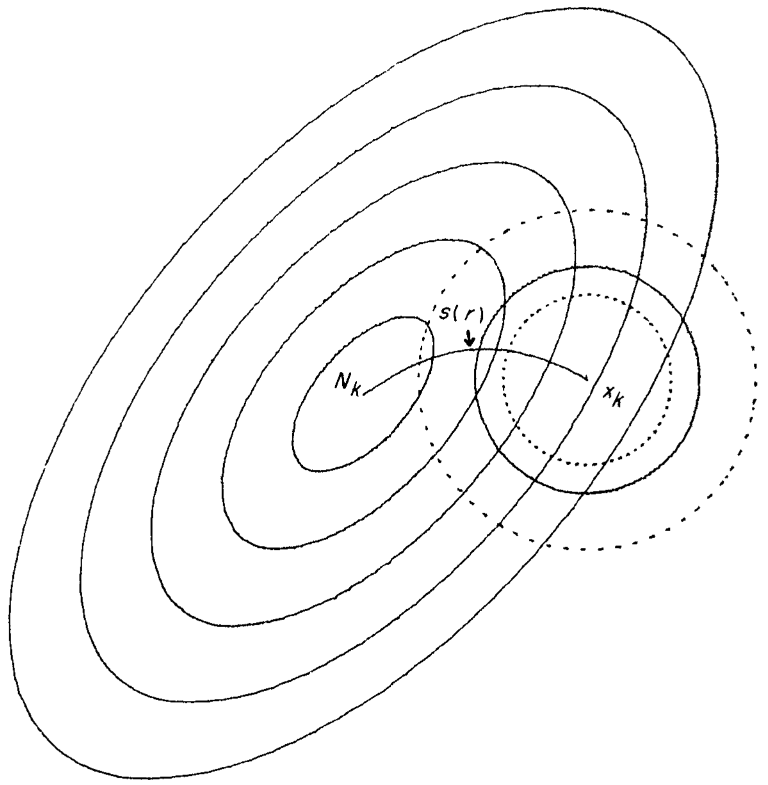


Figure 1.

trust region may have to be shrunk repeatedly with attendant evaluations of the residual function R to obtain an acceptable x_1. Much more serious is the possibility of overflow. The initial step bound b_0, that is, the maximum length allowed for the very first step attempted, is a parameter in NL2SOL; so the initial assumption of global linearity can be overruled by making b_0 small.

Figure 1 will perhaps be helpful at this point. The ellipses represent the contours of q_k and the circle is the trust region—our picture assumes the diagonal scaling matrix D_k to be the identity and the Hessian approximation to be positive definite. The point N_k is the "Newton point" or global minimizer of the convex quadratic model q_k, and the curve $s(r)$ represents the locus of minimizers of $q_k(x_k + s)$ constrained by $\| s \|_2 \le r$, $0 < r < \infty$. Complete details, based largely on [31], can be found in [24], including the case where H_k is not positive definite, but we choose $\Delta x_k = s(r)$ so that $\| D_k \Delta x_k \|_2$ lies between 0.9 and 1.1 of the current trust radius.

Since we were using this adaptive approach, it is not surprising that we also thought of using the new information at x_{k+1} to choose between q^S_{k+1} and q^G_{k+1} for use in determining x_{k+2}. We begin by default with $S = 0$ and hence with the Gauss-Newton model. Before giving our decision rules for step choice and model switching, we give some informal remarks that will probably be sufficient explanation for everyone except the specialist reader.

It would certainly be simpler to completely separate model selection from trust radius selection, and we do so except in one instance that we feel calls for their interaction. If the currently preferred model and trust region propose an unacceptable step, then we may decrease the trust radius; but the difficulty might equally well lie in our model preference. The easy route would be always to blame an excessive trust radius for a bad step, but our experience (see Table IV in Section 7) indicates that we obtain a more reliable algorithm if we try changing models in the reasonable manner that we now describe.

When the first trial step of an iteration fails, we test the alternate model to see, roughly speaking, if it would have predicted the observed failure at that point. If so, the alternate model gets a chance to make a trial step with the same trust radius. If we do not decide to try changing models, or if the alternate model fails to suggest a more successful step with the same trust radius, then we assume for the duration of the present iteration that our current model preference is correct. We then decrease the trust radius until x_{k+1} is determined or the algorithm fails.

In order to pin down the above comments about "successful steps" and "reasonable ways to change models," etc., we give a more formal description of our model switching strategy. We use q_k to denote the currently preferred model and q_k^{a} for the alternate model. Our tests depend on comparing predicted and observed function differences at certain points, and so it will be useful to have $\Delta f_k(x) := f(x) - f(x_k)$, $\Delta q_k(x) := q_k(x) - q_k(x_k) = q_k(x) - f(x_k)$, and $\Delta q_k^{\mathrm{a}}(x) := q_k^{\mathrm{a}}(x) - q_k^{\mathrm{a}}(x_k) = q_k^{\mathrm{a}}(x) - f(x_k)$. The Δq_k and Δq_k^{a} are our predictors for Δf_k.

We begin the $(k + 1)$st iteration by computing a prospective x_{k+1}, say x_{k+1}^{p}, based on q_k and the current trust radius. We compute $f(x_{k+1}^{\mathrm{p}})$, but we do not yet compute $\nabla f(x_{k+1}^{\mathrm{p}})$; our only gradient calculation in this iteration is $\nabla f(x_{k+1})$. If

$$\frac{\Delta f_k(x_{k+1}^{\mathrm{p}})}{\Delta q_k(x_{k+1}^{\mathrm{p}})} > 10^{-1}, \tag{5.2}$$

then the step is a good one; so if (see (5.1))

$$\lambda_k > 0 \tag{5.3a}$$

and

$$\Delta f_k(x_{k+1}) \leq 0.75 \cdot \nabla f(x_k)^{\mathrm{T}}(x_{k+1}^{\mathrm{p}} - x_k), \tag{5.3b}$$

that is, if the step constraint is binding and the direction appears worth pursuing, then we save x_{k+1}^{p} and $f(x_{k+1}^{\mathrm{p}})$ and try increasing the trust radius by a factor (between 2 and 4) chosen as in [21] and analogous to the decrease factor described in [31, p. 109]. We compute $x_{k+1}^{\mathrm{p}'}$ on the basis of q_k and the increased trust radius. If $f(x_{k+1}^{\mathrm{p}'}) \geq f(x_{k+1}^{\mathrm{p}})$, then we accept x_{k+1}^{p} as x_{k+1} and start getting ready for the next iteration. If $f(x_{k+1}^{\mathrm{p}'}) < f(x_{k+1}^{\mathrm{p}})$, then we replace x_{k+1}^{p} by $x_{k+1}^{\mathrm{p}'}$ and return to test (5.2). If ever (5.2) is true but (5.3) is false, then x_{k+1}^{p} is accepted as x_{k+1} and we get ready for the next iteration.

Now let us trace the branch that originates when (5.2) is false. In this case, we do not regard x_{k+1}^{p} very highly as a candidate for x_{k+1}, but its fate will be decided by further tests. We first test whether it might be useful to try changing models, but only if this is the first time through (5.2) in the current iteration. If

$$|q_k(x_{k+1}^{\mathrm{p}}) - f(x_{k+1}^{\mathrm{p}})| > 1.5\,|q_k^{\mathrm{a}}(x_{k+1}^{\mathrm{p}}) - f(x_{k+1}^{\mathrm{p}})|, \tag{5.4}$$

then we try the other model in the sense that we compute x^{a}_{k+1} with the same trust radius. If $f(x^{\mathrm{a}}_{k+1}) < f(x^{\mathrm{p}}_{k+1})$, then we change our model preference, so x^{a}_{k+1} becomes x^{p}_{k+1} and we return to test (5.2); otherwise, we retain our current model preference. Note that we test (5.4) only if the very first proposed step of an iteration is bad.

If we reach this point without having decided on x_{k+1}, then we have a poor proposed new iterate x^{p}_{k+1} and we have rejected the notion of switching models. If

$$\frac{\Delta f_k(x^{\mathrm{p}}_{k+1})}{\Delta q_k(x^{\mathrm{p}}_{k+1})} < 10^{-4}, \tag{5.5}$$

then we reject x^{p}_{k+1}, shrink the trust region as suggested by Fletcher [21] and Moré [31], recompute x^{p}_{k+1}, and return to test (5.2). If (5.5) is false, then we accept x^{p}_{k+1} as x_{k+1}, but we shrink the trust region in getting ready for the next iteration.

Once x_{k+1} has been found, we decide what trust region radius to use first when seeking x_{k+2}. The radius chosen has the form $\mu \cdot \| D_{k+1}\, \Delta x_k \|_2$, where $\Delta x_k = x_{k+1} - x_k$. If (5.2) with $x^{\mathrm{p}}_{k+1} := x_{k+1}$ is false, then μ is Fletcher's [21] decrease factor; otherwise, if either (5.3b) holds with $x^{\mathrm{p}}_{k+1} := x_{k+1}$ or

$$\| D^{-1}_{k+1} \{\nabla^2 q_k \Delta x_k - [\nabla f(x_{k+1}) - \nabla f(x_k)]\} \|_2 < \| D^{-1}_{k+1} f(x_{k+1}) \|_2, \tag{5.6a}$$

or

$$\Delta x_k^{\mathrm{T}}\, \nabla f(x_{k+1}) < 0.75\, \Delta x_k^{\mathrm{T}}\, \nabla f(x_k), \tag{5.6b}$$

then $2 \le \mu \le 4$ as above; otherwise $\mu = 1$. This rule for updating the radius is a modification of one described by Powell [36].

After we have found an acceptable x_{k+1}, we decide whether to change model preferences for computing x_{k+2}. We have found that it is best to retain the currently preferred model if (5.4) holds with $x^{\mathrm{p}}_{k+1} := x_{k+1}$, that is, unless the other model does a significantly better job of predicting the new function value. This decision is independent of our choice of the new trust radius.

6. CONVERGENCE CRITERIA AND COVARIANCE

An important, sometimes difficult issue in practical computing is the matter of deciding when to stop an iterative process. We have chosen to include five convergence tests in NL2SOL: tests for "X-convergence," "relative function-convergence," "absolute function-convergence," "singular convergence," and "false convergence."

Absolute function-convergence occurs if an iterate x_k is found with

$$f(x_k) < \epsilon_A \tag{6.1}$$

for a prescribed tolerance ϵ_A. This test is included to cover the rare case where x^* is the zero vector and $f(x^*) = 0$, since the X-convergence and relative function-convergence tests do not work in this case.

The other convergence tests are only performed if the current step Δx_k yields no more than twice the predicted function decrease, that is, if

$$f(x_k) - f(x_k + \Delta x_k) \le 2[f(x_k) - q_k(x_k + \Delta x_k)]. \tag{6.2}$$

These other tests rely heavily on q_k, the current quadratic model, which seems very untrustworthy if (6.2) fails to hold. We do not worry if the latest step Δx_k actually increases the computed function value, since this may happen because of roundoff error. But we do return whichever of x_k and $x_k + \Delta x_k$ gives the smallest computed function value.

Both the X-convergence and false-convergence tests employ the scale matrix $D_k = \operatorname{diag}(d_1^k, \ldots, d_p^k)$ mentioned in Section 5 to compute a scaled relative difference, RELDX(x, y, D), between two vectors $x, y \in \mathbb{R}^p$. This could be defined in any of several ways. For simplicity, we have chosen the definition

$$\text{RELDX}(x, y, D) := \frac{\max_i\{\,|\,d_i(x_i - y_i)\,|\,\}}{\max_j\{d_j(\,|\,x_j\,| + |\,y_j\,|\,)\}}, \tag{6.3}$$

where i and j range between 1 and p.

X-convergence means it appears likely that the current iterate x_k is within a prescribed tolerance ϵ_x of a strong local minimizer x^* (a minimizer at which the Hessian $\nabla^2 f(x^*)$ is positive definite) in the sense that RELDX $(x_k, x^*, D_k) \le \epsilon_x$. We judge this to be the case if the current step is a Newton step (*i.e.*, $\lambda_k = 0$ in (5.1)) for which (6.2) holds and

$$\text{RELDX}\ (x_k, x_k + \Delta x_k, D_k) \le \epsilon_k. \tag{6.4}$$

Relative function-convergence means it appears likely that the current function value $f(x_k)$ is close to its value $f(x^*)$ at a strong local minimizer x^* in the sense that $f(x_k) - f(x^*) \le \epsilon_R f(x_k)$ for a prescribed tolerance ϵ_R. We judge this to be the case if, simultaneously, (6.2) holds, the Hessian $H_k = \nabla^2 q_k$ of the current quadratic model is positive definite, and the function reduction predicted for a Newton step is no more than $\epsilon_R f(x_k)$, that is,

$$\frac{f(x_k) - q_k(x_k - H_k^{-1}\nabla f(x_k))}{f(x_k)} \le \epsilon_R. \tag{6.5}$$

It sometimes happens that (6.4) and (6.5) both hold, and NL2SOL has a special return code for this case.

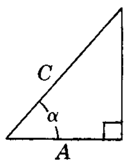
Singular convergence is similar to relative function-convergence, except that the least-squares Hessian $\nabla^2 f(x_k)$ appears to be singular or nearly so. In cases where R arises from a data-fitting problem, this means that the model for the data is overspecified, that is, x has too many components, at least for x near x_k. We declare singular convergence to have occurred if, simultaneously, none of the stopping tests already described is satisfied and the current model predicts that a change of no more than $\epsilon_R f(x_k)$ can be made in the objective function value by any step from x_k bounded by the initial step bound b_0, that is,

$$\max\{f(x_k) - q_k(x) : \|\,D_k(x - x_k)\|_2 \le b_0\} \le \epsilon_R f(x_k). \tag{6.6}$$

If necessary, the left-hand side of (6.6) is evaluated by computing (but not trying) another step of the form (5.1).

False convergence means that the iterates appear to be converging to a noncritical point. We declare it to occur if, simultaneously, none of the previously described tests is satisfied, (6.2) does not hold for the current step Δx_k, and

$$\text{RELDX}(x_k, x_k + \Delta x_k, D_k) < \epsilon_F \tag{6.7}$$



C = current residual: $C^2 = 2f(x_k)$.
B = optimal residual according to the Gauss-Newton model, for which $H_k = J(x_k)^T J(x_k)$: $B^2 = 2q_k(x_k - H_k^{-1}\nabla f(x_k))$.
A = projection of the current residual onto the column space of $J(x_k)$, the current Jacobian. $A^2 = C^2 - B^2$.
c_k = $\cos\alpha = A/C$: c_k^2 = left-hand side of (6.5)

Fig. 2. c_k for the Gauss-Newton model

for a specified tolerance ϵ_F that should generally be less than ϵ_x. This may mean that the convergence tolerances in (6.1) and (6.4)-(6.6) are too small for the accuracy to which f and J are being computed, that there is an error in computing J, or that f or ∇f is discontinuous near x_k.

Earlier versions of NL2SOL included a stopping test called the COSMAX test that measured the cosines of the angles between the columns of the current Jacobian matrix and the corresponding residual vector. We would have preferred to examine c_k, the cosine of the angle between the residual vector and its orthogonal projection onto the column space of the Jacobian matrix, but this cosine would be expensive to compute for the augmented model. By contrast, c_k is readily available for the Gauss-Newton model, since it is then the square root of the left-hand side of (6.5); see Figure 2. For the Gauss-Newton model, (6.5) thus amounts to a test that we would have preferred to the COSMAX test, and for the augmented model it is a natural generalization of this preferred test. Several people have suggested tests based on c_k, including Allen [1] and Bates and Watts [4]. (See also Belsley's weighted gradient stopping test [6].)

Test (6.5) can also be motivated by statistical considerations. Since there is inherent variability in the data, it is generally not useful to continue iterating when a candidate step $\Delta x_k = (\Delta x_1^k, \ldots, \Delta x_p^k)$ is generated for which

$$\max\{|\Delta x_i^k|/\text{s.e.}(x_i^k)\colon 1 \le i \le p\} \tag{6.8}$$

is sufficiently small. Here s.e.(x_i^k) denotes some estimate of the standard error (square root of the variance) of the ith component of the current parameter vector estimate x_k and so is a function of the statistical variability in the data.

An alternative to (6.8) suggested by Pratt [37] is to consider general linear combinations $l^T\Delta x_k$ of the components of Δx_k, that is,

$$\max\{|l^T\Delta x_k|/(l^T V_k l)\colon\ l \neq 0\} = (\Delta x_k^T V_k^{-1}\Delta x_k)^{-1/2}, \tag{6.9}$$

where V_k is a current estimate of the covariance matrix. For s.e.$(x_i^k) = (e_i^T V_k e_i)^{1/2}$, where e_i is the ith standard unit vector, (6.9) clearly dominates (6.8), so it seems reasonable to base a test on (6.9). If we choose $V_k = \hat{\sigma}_k H_k^{-1}$, where $\hat{\sigma}_k$ is the current residual sum of squares divided by $\max\{1, n-p\}$, that is,

$$\hat{\sigma}_k = 2f(x_k)/\max\{1, n-p\}, \tag{6.10}$$

and if Δx_k is a full Newton step, that is, $\Delta x_k = -H_k^{-1}\nabla f(x_k)$, then (6.9) equals $\max\{1, n-p\}$ times the square root of the left-hand side of (6.5).

Many statistical inference procedures require an estimate of the covariance matrix at the solution x^*. NL2SOL provides three possibilities:

$$\hat{\sigma}^2 H^{-1} J^T J H^{-1} \tag{6.11}$$

$$\hat{\sigma}^2 H^{-1} \tag{6.12}$$

$$\hat{\sigma}^2 (J^{\mathrm{T}} J)^{-1} \tag{6.13}$$

where $\hat{\sigma}^2$ is given by (6.10) with $x_k := x^*$. When (6.11) or (6.12) is specified, a symmetric finite difference Hessian approximation H is obtained at the solution x^*. If H is positive definite (or J has full rank at x^* for (6.13)), the specified covariance matrix is computed.

A detailed discussion of all three covariance forms is contained in [3]. The second form (6.12) is based on asymptotic maximum likelihood theory and is perhaps the most common form of estimated covariance matrix. We feel that (6.11), the default, is more useful for smaller sample sizes and in other cases where the conditions necessary for the asymptotic theory [38] may be violated. The third form assumes that the residuals at the solution are small and is therefore often highly suspect.

7. TEST RESULTS

We have run NL2SOL on a number of the test problems reported in the literature. In particular, we have run it on the test problems listed in [26] and on one described in [30]. The original sources for these problems, together with the abbreviated problem names used in Tables II–IV and some notes, are given in Table I.

The behavior of NL2SOL is determined in part by an integer array IV and a floating-point array V, which contain iteration and function evaluation limits, convergence tolerances, and other switches and constants. In the runs summarized in Tables II–IV, most of the IV and V input components (other than the iteration and function evaluation limits) had the default values given them by subroutine DFAULT. In particular, the initial step bound (trust radius), b_0 = V(LMAX0), had the value 100, and the convergence tolerances ϵ_A, ϵ_x, ϵ_R, ϵ_F that appear in (6.1) and (6.4)–(6.7) had the following values: ϵ_A = V(AFCTOL) = 10^{-20}; ϵ_x = V(XCTOL) $\doteq 1.49 \times 10^{-8}$; ϵ_R = V(RFCTOL) = 10^{-10}; and ϵ_F = V(XFTOL) $\doteq 2.22 \times 10^{-14}$. The values just mentioned are the defaults for the double-precision version of NL2SOL on IBM 360 and 370 computers: we obtained Tables II–IV on the IBM 370/168 at the Massachusetts Institute of Technology; the double-precision arithmetic on this machine has a unit roundoff of $16^{-13} = 2.22 \times 10^{-16}$.

(Except as noted below and except for the runs stopped by the iteration or function evaluation limits, all runs reported in Tables III and IV found essentially the same function value listed in Table II.)

Table II summarizes the performance of NL2SOL on the test problem set when all IV and V input components (except the iteration and function evaluation limits) have their default values. Following a suggestion of J. J. Moré [private communication], we obtained new starting guesses for many of the test problems by multiplying the standard starting guess by 10 and 100. The column labeled LS gives the base 10 logarithm of the factor by which the standard starting guess was multiplied. The problem dimensions appear in the columns headed N and P, while the number of function (i.e., $R(x)$) and gradient (i.e., $J(x)$) evaluations performed, respectively, appear under NF and NG. Located under C is a code telling why NL2SOL stopped: X means X-convergence, R means relative function-convergence, B means both X and R, A means absolute function-conver-

Table I Original Sources of Test Problems

Problem	Note	Source
ROSNBROK		[40]
HELIX	1	[22]
SINGULAR		[35]
WOODS		[11]
ZANGWILL	2	[44]
ENGVALL		[19]
BRANIN		[9]
BEALE		[5]
CRAGG	3	[26]
BOX		[8]
DAVIDON1	4	[13]
FRDSTEIN	5	[23]
WATSON6,9,12,20	6	[29]
CHEBQD8		[20]
BROWN	7	[10]
BARD		[2]
JENNRICH		[28]
KOWALIK		[29]
OSBORNE1,2		[34]
MEYER		[30]

Notes

1. The residual vector $R(x)$ for this problem is a discontinuous function of x On those runs where NL2SOL halts with false convergence, the iterates have converged to a point of discontinuity

2. This is a linear least-squares problem that NL2SOL solves in one step when the initial step bound, that is, V(LMAX0), is increased somewhat from its default value of 100 (to at least 174).

3. The original Miele problem described in [12], which Gill and Murray [26] cite as the source for this problem, does not have the residual component $r_5(x) = x_4 - 1$ This new component forces x_4 to move more rapidly toward 1, but otherwise causes no noteworthy change in the performance of NL2SOL.

4 This is a very ill-conditioned linear least-squares problem If V(LMAX0) is set large (to at least $1\ 9 \times 10^7$), then NL2SOL halts with X-convergence after two steps when using double-precision arithmetic on an IBM 370 computer With a double precision of a few bits more accuracy, such as that of the Honeywell 6180 or the Univac 1110, NL2SOL attains absolute function convergence after a single step

5 In all our test runs, NL2SOL found a local solution to this problem. The residual vector vanishes at the global solution

6. WATSON20 lies near the boundary between zero-residual and nonzero-residual problems. After the first dozen or so iterations, NL2SOL can neither make further substantial reductions in the sum of squares nor satisfy any of its default convergence criteria To reduce the computer time spent on this problem, we used a function evaluation limit of 20 and an iteration limit of 15 on all runs of WATSON20 reported here.

7 Gill and Murray [26] call this problem "Davidon 2"

Table II Default NL2SOL

PROBLEM	LS	N	P	NF	NG	C	F	PRELDF	RELDX
ROSNBROK	0	2	2	26	19	A	0.973E-32	0.100E+01	0.818E-03
ROSNBROK	1	2	2	57	39	A	0.973E-32	0.100E+01	0.594E-04
ROSNBROK	2	2	2	141	121	A	0.973E-32	0.100E+01	0.440E-03
HELIX	0	3	3	13	11	A	0.276E-20	0.100E+01	0.145E-05
HELIX	1	3	3	19	16	A	0.120E-20	0.100E+01	0.244E-05
HELIX	2	3	3	103	45	F	0.120E+05	0.984E+00	0.181E-13
SINGULAR	0	4	4	20	20	A	0.107E-20	0.100E+01	0.333E+00
SINGULAR	1	4	4	26	25	A	0.751E-21	0.100E+01	0.333E+00
SINGULAR	2	4	4	34	27	A	0.224E-20	0.100E+01	0.333E+00
WOODS	0	7	4	70	47	A	0.232E-23	0.100E+01	0.197E-06
WOODS	1	7	4	59	46	A	0.487E-26	0.100E+01	0.426E-07
WOODS	2	7	4	77	53	X	0.0	0.100E+01	0.359E-10
ZANGWILL	0	3	3	3	3	A	0.426E-27	0.100E+01	0.100E+01
ENGVALL	0	5	3	17	13	X	0.279E-32	0.100E+01	0.357E-10
ENGVALL	1	5	3	21	19	X	0.631E-29	0.100E+01	0.268E-08
ENGVALL	2	5	3	31	26	A	0.164E-22	0.100E+01	0.107E-06
BRANIN	0	2	2	2	2	A	0.162E-28	0.100E+01	0.100E+01
BRANIN	1	2	2	18	15	A	0.662E-29	0.100E+01	0.100E+01
BRANIN	2	2	2	20	10	A	0.138E-20	0.100E+01	0.100E+01
BEALE	0	3	2	10	9	A	0.893E-26	0.100E+01	0.116E-06
BEALE	1	3	2	6	6	A	0.148E-21	0.100E+01	0.115E-05
CRAGG	0	5	4	24	23	A	0.217E-20	0.100E+01	0.253E-07
CRAGG	1	5	4	80	47	R	0.617E+05	0.919E-11	0.979E-07
BOX	0	10	3	7	7	X	0.174E-31	0.100E+01	0.196E-09
BOX	1	10	3	16	10	S	0.378E-01	0.537E-10	0.177E-13
DAVIDON1	0	15	15	20	15	X	0.400E-18	0.100E+01	0.105E-08
FRDSTEIN	0	2	2	9	8	R	0.245E+02	0.359E-11	0.769E-06
FRDSTEIN	1	2	2	18	13	R	0.245E+02	0.532E-13	0.934E-07
FRDSTEIN	2	2	2	28	19	R	0.245E+02	0.782E-13	0.418E-07
WATSON6	0	31	6	12	10	B	0.114E-02	0.422E-19	0.142E-10
WATSON9	0	31	9	10	9	R	0.700E-06	0.173E-10	0.360E-07
WATSON12	0	31	12	14	12	R	0.236E-09	0.122E-13	0.254E-07
WATSON20	0	31	20	18	16	I	0.651E-17	0.532E+00	0.270E+00
CHEBQD8	0	8	8	23	18	B	0.176E-02	0.277E-11	0.103E-07
CHEBQD8	1	8	8	77	57	R	0.176E-02	0.392E-10	0.841E-07
BROWN	0	20	4	18	17	R	0.429E+05	0.224E-10	0.228E-06
BROWN	1	20	4	22	16	R	0.429E+05	0.848E-12	0.696E-07
BROWN	2	20	4	31	21	R	0.429E+05	0.111E-10	0.187E-06
BARD	0	15	3	7	7	R	0.411E-02	0.270E-12	0.119E-06
BARD	1	15	3	32	23	S	0.871E+01	0.396E-10	0.243E+00
BARD	2	15	3	70	28	R	0.411E-02	0.411E-10	0.146E-05
JENNRICH	0	10	2	15	13	R	0.622E+02	0.169E-12	0.134E-06
KOWALIK	0	11	4	11	10	R	0.154E-03	0.421E-10	0.423E-06
KOWALIK	1	11	4	130	75	S	0.514E-03	0.689E-10	0.242E+00
KOWALIK	2	11	4	75	58	R	0.154E-03	0.470E-11	0.103E-06
OSBORNE1	0	33	5	27	22	R	0.273E-04	0.332E-11	0.524E-06
OSBORNE2	0	65	11	17	16	B	0.201E-01	0.492E-12	0.933E-08
OSBORNE2	1	65	11	26	12	S	0.895E+00	0.353E-10	0.879E-07
MADSEN	0	3	2	12	12	R	0.387E+00	0.105E-13	0.496E-07
MADSEN	1	3	2	16	15	R	0.387E+00	0.154E-11	0.584E-06
MADSEN	2	3	2	28	20	R	0.387E+00	0.120E-12	0.152E-06
MEYER	0	16	3	335	206	X	0.440E+02	0.705E-05	0.136E-07

Table III Variations on NL2SOL

		D = I			DEFAULT			PURE GN			PURE S			NO SIZING			
PROBLEM	LS	NF	NG	C	NF	NG	C	NF	NG	C	NF	NG	C	NF	NG	C	NOTE
ROSNBROK	0	22	18	A	26	19	A	18	15	A	23	21	A	31	22	A	
ROSNBROK	1	28	24	A	57	39	A	38	29	A	155	69	X	30	25	A	
ROSNBROK	2	77	54	A	141	121	A	115	101	A	400	146	E	89	82	A	
HELIX	0	9	9	X	13	11	A	17	14	A	15	14	X	14	12	X	
HELIX	1	11	9	A	19	16	A	15	13	A	23	18	X	18	14	A	
HELIX	2	16	14	X	103	45	F	28	23	X	25	19	X	80	37	F	
SINGULAR	0	20	20	A	20	20	A	20	20	A	32	32	A	20	20	A	
SINGULAR	1	23	23	A	26	25	A	25	24	A	40	39	A	26	25	A	
SINGULAR	2	23	27	A	34	27	A	34	27	A	49	44	A	34	27	A	
WOODS	0	61	45	X	70	47	A	80	64	A	45	39	A	70	48	A	
WOODS	1	63	46	A	59	46	A	87	70	A	47	39	A	117	70	A	
WOODS	2	72	52	X	77	53	X	85	65	A	63	45	X	93	65	A	
ZANGWILL	0	3	3	A	3	3	A	3	3	A	3	3	A	3	3	A	
ENGVALL	0	17	15	A	17	13	X	14	12	X	19	17	X	18	13	X	
ENGVALL	1	20	18	A	21	19	X	20	19	X	27	21	R	20	18	A	1
ENGVALL	2	27	25	R	31	26	A	100	72	A	44	37	X	36	30	A	2
BRANIN	0	2	2	A	2	2	A	2	2	A	2	2	A	2	2	A	
BRANIN	1	17	15	A	18	15	A	14	12	A	28	25	A	17	15	A	
BRANIN	2	16	14	A	20	10	A	21	12	A	49	38	A	20	10	A	
BEALE	0	10	8	A	10	9	A	10	9	X	19	15	A	10	9	X	
BEALE	1	8	8	A	6	6	A	6	6	A	13	12	A	6	6	A	
CRAGG	0	23	22	A	24	23	A	23	22	A	34	32	X	24	23	A	
CRAGG	1	54	47	A	80	47	R	150	91	R	75	48	R	120	78	R	3
BOX	0	7	7	X	7	7	X	7	7	X	8	8	X	7	7	X	
BOX	1	27	19	R	16	10	S	12	11	R	45	20	S	16	11	F	
DAVIDON1	0	3	3	X	20	15	X	19	14	X	20	16	I	20	15	X	
FRDSTEIN	0	9	9	R	9	8	R	37	16	F	8	8	R	9	7	R	4
FRDSTEIN	1	18	15	R	18	13	R	44	22	F	22	19	R	18	13	R	4
FRDSTEIN	2	28	23	F	28	19	R	53	26	F	38	30	B	30	20	R	4
WATSON6	0	8	8	R	12	10	B	12	11	R	16	12	B	12	10	B	
WATSON9	0	10	9	R	10	9	R	11	9	B	21	14	B	10	9	R	
WATSON12	0	14	11	R	14	12	R	16	14	R	23	17	B	15	12	B	
WATSON20	0	20	14	E	18	16	I	18	16	I	18	16	I	18	16	I	5
CHEBQD8	0	22	16	R	23	18	B	58	36	F	20	16	R	80	35	F	4, 6
CHEBQD8	1	78	63	R	77	57	R	400	109	E	143	105	R	102	78	S	
BROWN	0	14	13	R	18	17	R	305	301	I	19	17	R	32	30	R	
BROWN	1	15	15	R	22	16	R	400	281	E	27	21	R	107	64	R	
BROWN	2	24	23	R	31	21	R	400	296	E	35	26	R	40	27	R	
BARD	0	7	7	R	7	7	R	7	7	R	11	10	B	7	7	R	
BARD	1	36	22	S	32	23	S	32	23	S	81	42	S	32	23	S	7
BARD	2	37	23	R	70	28	R	63	27	R	129	58	R	75	32	S	8
JENNRICH	0	16	12	B	15	13	R	35	19	F	11	11	R	16	14	R	4
KOWALIK	0	14	12	R	11	10	R	18	17	R	15	12	R	11	10	R	
KOWALIK	1	189	88	S	130	75	S	127	77	S	127	73	R	93	65	S	9
KOWALIK	2	112	69	R	75	58	R	96	81	R	400	200	E	138	124	R	
OSBORNE1	0	34	26	R	27	22	R	18	16	R	34	31	R	18	16	R	
OSBORNE2	0	15	13	R	17	16	B	15	14	R	16	15	R	16	15	R	
OSBORNE2	1	16	12	S	26	12	S	13	11	S	28	16	S	27	13	S	
MADSEN	0	12	12	R	12	12	R	33	33	R	12	12	R	13	13	R	
MADSEN	1	14	14	B	16	15	R	39	36	R	19	18	R	21	19	R	
MADSEN	2	21	20	B	28	20	R	47	40	R	28	23	R	37	29	R	
MEYER	0	380	229	B	335	206	X	346	213	B	156	129	B	322	199	B	

Notes

1. The PURE S run found a local minimizer x^* having $f(x^*) = 56.1$.

2 The D = I run also found $f(x^*) = 56.1$

3. All runs found different local minimizers· for D = I, $f(x^*) = 1.68 \times 10^{-21}$; for DEFAULT, $f(x^*) = 6\ 17 \times 10^{4}$; for PURE GN, $f(x^*) = 1\ 50 \times 10^{5}$, for PURE S, $f(x^*) = 2.30 \times 10^{7}$; and for NO SIZING, $f(x^*) = 1\ 35 \times 10^{5}$

4 *In the* PURE GN *runs of these problems,* NL2SOL *reports false convergence because the* Jacobian is (nearly) singular at the solutions found and the Gauss-Newton Hessian differs sufficiently from the true one that the singular convergence test is not satisfied with the convergence tolerances at their default values If the b_0 in (6 6) were changed from 100 to 1, then NL2SOL would report singular convergence on JENNRICH, and if the ϵ_R in (6.6) were also increased slightly from 10^{-10}, say to $2\ 3 \times 10^{-10}$, then NL2SOL would also report singular convergence on FRDSTEIN. Note that the true Hessian is quite positive definite at the solutions found

5 The final function values were as follows for D = I, $1\ 36 \times 10^{-16}$; for DEFAULT, 6.51×10^{-18}, for PURE GN, $6\ 50 \times 10^{-18}$, for PURE S, 3.49×10^{-16}, and for NO SIZING, $4\ 98 \times 10^{-18}$

6 *In the NO SIZING run,* NL2SOL often tried the augmented model, but always switched back to the Gauss-Newton model (This run computed slightly different iterates than the corresponding PURE GN run because the latter used $S_u = 0$ in (7.1).)

7 The PURE S run found $f(x^*) = 8.57$.

8 The NO SIZING run found $f(x^*) = 5\ 74 \times 10^{-2}$

9. The PURE S run found $f(x^*) = 1.54 \times 10^{-3}$

Table IV. Simplified Iterations

		DEFAULT			NOIMODSW			NOINTDBL			NOGRDTST			
PROBLEM	LS	NF	NG	C	NF	NG	C	NF	NG	C	NF	NG	C	NOTE
ROSNBROK	0	26	19	A	27	19	A	26	21	X	21	18	A	
ROSNBROK	1	57	39	A	36	29	A	74	57	A	45	35	A	
ROSNBROK	2	141	121	A	135	115	A	164	145	A	210	155	A	
HELIX	0	13	11	A	13	11	A	13	11	A	17	14	X	
HELIX	1	19	16	A	17	14	A	19	16	A	18	15	X	
HELIX	2	103	45	F	110	33	F	20	16	X	99	43	F	
SINGULAR	0	20	20	A	20	20	A	20	20	A	20	20	A	
SINGULAR	1	26	25	A	26	25	A	25	25	A	28	25	A	
SINGULAR	2	34	27	A	34	27	A	31	31	A	34	27	A	
WOODS	0	70	47	A	65	47	X	59	51	A	75	49	X	
WOODS	1	59	46	A	71	48	X	41	37	A	54	42	A	
WOODS	2	77	53	X	77	53	X	59	55	X	87	54	A	
ZANGWILL	0	3	3	A	3	3	A	3	3	A	3	3	A	
ENGVALL	0	17	13	X	16	13	X	17	14	X	17	13	X	
ENGVALL	1	21	19	X	23	19	X	16	16	X	22	19	A	
ENGVALL	2	31	26	A	31	26	A	36	34	X	36	28	A	
BRANIN	0	2	2	A	2	2	A	2	2	A	2	2	A	
BRANIN	1	18	15	A	18	15	A	25	25	A	16	13	A	
BRANIN	2	20	10	A	20	10	A	39	38	A	23	12	A	
BEALE	0	10	9	A	11	9	A	10	9	A	10	9	A	
BEALE	1	6	6	A	6	6	A	6	6	A	6	6	A	
CRAGG	0	24	23	A	23	21	A	24	23	A	25	24	A	
CRAGG	1	80	47	R	80	47	R	109	93	B	80	49	B	1
BOX	0	7	7	X	7	7	X	7	7	X	7	7	X	
BOX	1	16	10	S	17	10	S	16	10	S	15	11	R	
DAVIDON1	0	20	15	X	20	15	X	16	16	I	20	13	E	
FRDSTEIN	0	9	8	R	8	7	R	9	8	R	9	8	R	
FRDSTEIN	1	18	13	R	18	13	R	19	17	R	18	14	B	
FRDSTEIN	2	28	19	R	35	22	B	29	27	R	29	20	B	
WATSON6	0	12	10	B	12	10	B	11	10	B	12	10	B	
WATSON9	0	10	9	R	10	9	R	10	9	B	11	9	B	
WATSON12	0	14	12	R	14	12	R	14	13	R	18	13	B	
WATSON20	0	18	16	I	18	16	I	16	16	I	20	15	E	2
CHEBQD8	0	23	18	B	24	19	B	19	14	R	23	18	B	
CHEBQD8	1	77	57	R	118	76	R	112	98	R	104	82	R	
BROWN	0	18	17	R	18	17	R	20	19	R	20	18	R	
BROWN	1	22	16	R	25	19	R	24	23	R	26	20	B	
BROWN	2	31	21	R	32	22	B	30	29	B	32	23	R	
BARD	0	7	7	R	7	7	R	7	7	R	7	7	R	
BARD	1	32	23	S	32	23	S	29	29	S	32	23	S	
BARD	2	70	28	R	77	28	R	66	43	R	66	30	R	
JENNRICH	0	15	13	R	15	13	R	15	13	R	15	13	R	
KOWALIK	0	11	10	R	13	10	B	11	10	R	13	10	R	
KOWALIK	1	130	75	S	244	100	F	109	84	S	124	76	S	3
KOWALIK	2	75	58	R	74	58	R	78	62	R	117	81	R	
OSBORNE1	0	27	22	R	31	22	R	28	23	R	34	23	R	
OSBORNE2	0	17	16	B	17	16	B	17	16	B	18	16	B	
OSBORNE2	1	26	12	S	27	12	S	13	12	S	26	12	S	
MADSEN	0	12	12	R	12	12	R	12	12	R	12	12	R	
MADSEN	1	16	15	R	16	15	R	18	18	B	19	17	R	
MADSEN	2	28	20	R	29	21	R	25	25	R	27	22	R	
MEYER	0	335	206	X	343	214	B	209	181	B	351	213	X	

Notes.

1. The NOINTDBL run found $f(x^*) = 9.30 \times 10^4$, and the NOGRDTST run found $f(x^*) = 8.65 \times 10^4$

2 The final function values were as follows for DEFAULT and NOIMODSW, $6\ 51 \times 10^{-18}$, for NOINTDBL, $3\ 48 \times 10^{-17}$; for NOGRDTST, 9.71×10^{-18}

3 If the defaults for b_0 or ϵ_R in (6 6) were slightly relaxed (e.g., if b_0 were reduced from 100 to 50, or if ϵ_R were increased from 10^{-10} to 1.5×10^{-10}), then the NOIMODSW run would also report singular convergence.

gence, S means singular convergence, F means false convergence, I means iteration limit reached without convergence, and E means function evaluation limit reached without convergence. See the previous section for more details on the convergence criteria. The column labeled F gives the final function value (half the sum of squares of $R(x)$); the one labeled PRELDF gives the relative function reduction predicted, that is, $[f(x_k) - q_k(x_k + \Delta x_k)]/f(x_k)$ for the last step Δx_k attempted; and RELDX gives the value of (6.3) for the last step attempted.

The choice of scale matrices D_k mentioned in Section 5 can significantly affect the performance of NL2SOL. By default, $D_k = \mathrm{diag}(d_1^k, \ldots, d_p^k)$ is updated by the rule

$$d_i^{k+1} := \max\{[\| J_{\cdot,i} \|_2^2 + \max\{0, S_{ii}\}]^{1/2}, 0.6 d_i^k\}, \tag{7.1}$$

beginning with $d_i^{-1} = 0$, where $J_{\cdot,i}$ denotes the ith column of the Jacobian matrix $J(x_{k+1})$. However, if (7.1) results in $d_i^{k+1} < 10^{-6}$, then d_i^{k+1} is set to 1.0. (The factor 0.6 is actually V(DFAC). We experimented with several values of V(DFAC), including 0.0, 0.5, 0.75, and 1.0, and we felt that 0.6 gave the best overall performance of the values tried.) The advantage of this choice of D_k is that it is largely scale invariant.

A choice of D_k that is not at all scale invariant, but that gives better performance on many of our test problems, is $D_k = I$, the identity matrix. Table III shows how these two choices of D compare: Results from Table II are in the columns headed DEFAULT, while results corresponding to $D_k = I$ appear under $D = I$.

Table III also summarizes test runs with three variants of NL2SOL, all of which used the default choice of D_k and the same IV and V inputs as were used for Table II. The columns headed PURE GN show what happens if the augmented model is never used and $S_{ii} = 0$ is used in (7.1), while those headed PURE S show what happens if it is always used. Finally, the columns headed NO SIZING give the results obtained when adaptive modeling is allowed but no sizing is performed. We feel that Table III makes a good case for the use of adaptive modeling with sizing in NL2SOL.

Table IV shows how NL2SOL performs when some of the procedures described in Section 5 are simplified. All runs were made using the same IV and V input values as for Table II, and the columns labeled DEFAULT summarize the results in Table II. The results under NOIMODSW show what happens when there is no internal model switching, that is, if we do not consider switching models within the current iteration. The columns labeled NOINTDBL show what happens if there is no internal doubling of the radius, that is, if we do not compute $x_{k+1}^{\mathrm{P}'}$ when (5.2) and (5.3) hold. Finally, the results under NOGRDTST show what happens if no gradient tests are used in determining the new trust region radius after x_{k+1} has been found, that is, if (5.6) is not considered and the radius is only increased if (5.2) and (5.3b) hold with $x_{k+1}^{\mathrm{P}} := x_{k+1}$. Table IV clearly demonstrates the value of internal model switching, internal doubling, and the gradient tests.

It is interesting to see how the performance of NL2SOL compares with that of a general-purpose quasi-Newton algorithm. We therefore summarize in Table V the results of running SUMSOL [25] on the same problems used for the earlier tables. SUMSOL uses the BFGS secant update to approximate the Hessian of the objective function and uses the double dogleg scheme of Dennis and Mei [16]

Table V Comparison with SUMSOL

		NL2SOL D = I			SUMSOL (J**T)*J			SUMSOL LMAX0=1			SUMSOL H_0 = I			
PROBLEM	LS	NF	NG	C	NF	NG	C	NF	NG	C	NF	NG	C	NOTE
ROSNBROK	0	22	18	A	50	37	X	39	32	X	40	36	X	
ROSNBROK	1	28	24	A	70	56	A	104	74	X	96	70	X	
ROSNBROK	2	77	54	A	340	251	I	229	174	X	201	146	X	1
HELIX	0	9	9	X	39	30	X	41	33	X	38	28	X	
HELIX	1	11	9	A	47	35	X	57	40	X	48	34	X	
HELIX	2	16	14	X	57	40	X	57	37	X	35	25	X	
SINGULAR	0	20	20	A	45	45	A	77	75	A	80	75	A	
SINGULAR	1	23	23	A	53	53	A	88	86	A	91	82	A	
SINGULAR	2	28	27	A	91	89	A	108	99	A	99	90	A	
WOODS	0	61	45	X	102	75	X	128	89	X	103	79	X	
WOODS	1	63	46	A	130	99	X	92	72	X	80	61	X	
WOODS	2	72	52	X	96	83	X	79	70	X	72	54	X	
ZANGWILL	0	3	3	A	3	3	A	6	3	A	10	7	A	
ENGVALL	0	17	15	A	35	30	X	36	32	X	33	30	X	
ENGVALL	1	20	18	A	53	42	X	56	45	X	43	39	X	
ENGVALL	2	27	25	R	83	75	X	79	71	X	66	55	X	2
BRANIN	0	2	2	A	2	2	A	19	16	A	18	15	A	
BRANIN	1	17	15	A	28	28	A	38	34	A	38	33	A	
BRANIN	2	16	14	A	51	49	A	64	56	A	48	35	A	
BEALE	0	10	8	A	21	17	X	18	13	X	17	14	A	
BEALE	1	8	8	A	19	17	X	16	15	X	16	15	X	
CRAGG	0	23	22	A	118	108	A	119	112	A	115	102	A	
CRAGG	1	54	47	A	88	76	R	128	90	R	185	116	R	3
BOX	0	7	7	X	16	15	X	29	22	X	48	35	A	
BOX	1	27	19	R	39	20	X	52	41	B	37	27	B	
DAVIDON1	0	3	3	X	4	4	X	6	5	X	20	2	F	
FRDSTEIN	0	9	9	R	9	9	R	9	8	R	13	11	R	
FRDSTEIN	1	18	15	R	29	24	R	30	25	R	30	24	R	
FRDSTEIN	2	28	23	B	44	38	R	51	39	R	55	39	R	
WATSON6	0	8	8	R	25	21	R	25	21	R	41	34	R	
WATSON9	0	10	9	R	22	22	B	22	22	B	81	72	R	
WATSON12	0	14	11	R	32	27	R	33	28	B	125	110	R	4
WATSON20	0	20	14	E	16	16	I	16	16	I	18	16	I	5
CHEBQD8	0	22	16	R	40	32	R	38	28	R	39	27	R	
CHEBQD8	1	78	63	R	234	208	B	232	208	B	227	186	R	
BROWN	0	14	13	R	25	21	R	24	20	R	46	35	R	
BROWN	1	15	15	R	45	43	R	52	46	R	41	30	R	
BROWN	2	24	23	R	78	73	R	80	71	R	47	38	R	
BARD	0	7	7	R	20	16	R	17	16	R	22	18	R	
BARD	1	36	22	S	79	59	S	66	46	R	34	23	R	6
BARD	2	37	23	B	80	55	S	89	49	R	73	43	R	7
JENNRICH	0	16	12	B	16	14	R	16	14	R	34	22	R	
KOWALIK	0	14	12	R	27	19	R	27	19	R	42	33	R	
KOWALIK	1	189	88	S	220	159	S	55	48	S	91	73	R	8
KOWALIK	2	112	69	R	78	56	S	112	72	R	221	124	R	9
OSBORNE1	0	34	26	R	56	42	R	56	42	R	83	59	R	
OSBORNE2	0	15	13	R	37	32	R	43	34	R	75	59	R	
OSBORNE2	1	16	12	S	28	20	R	52	31	B	53	31	B	
MADSEN	0	12	12	R	15	15	R	13	13	R	16	16	R	
MADSEN	1	14	14	B	30	28	R	30	28	B	31	28	R	
MADSEN	2	21	20	B	36	35	R	39	32	R	41	36	R	
MEYER	0	380	229	B	400	268	E	400	277	E	400	259	E	10

Notes

1 The (J**T)*J run stopped with $f(x) = 1.32$.

2. NL2SOL found a local solution with $f(x^*) = 56\ 1$, the SUMSOL runs all found the global solution.

3. NL2SOL found the global solution, and each SUMSOL run found a different local solution for (J**T)*J, $f(x^*) = 232$, for LMAX0=1, $f(x^*) = 33\ 0$, and for $H_0 = I$, $f(x^*) = 1.27 \times 10^6$.

4 The $H_0 = I$ run of SUMSOL found $f(x^*) = 1\ 33 \times 10^{-7}$.

5 The final function values were as follows: for NL2SOL, 1.36×10^{-14}, for (J**T)*J, 0 290; for LMAX0=1, 0 293, and for $H_0 = I$, 4.13×10^{-3}

6 The (J**T)*J run found $f(x^*) = 8.51$ and the LMAX0=1 run found $f(x^*) = 1.18$.

7 The (J**T)*J run found $f(x^*) = 5.74 \times 10^{-2}$ and the LMAX0=1 run found $f(x^*) = 0\ 943$.

8 The LMAX0=1 run found $f(x^*) = 2.90 \times 10^{-3}$ and the $H_0 = I$ run found $f(x^*) = 1.54 \times 10^{-4}$ (as did all runs for LS = 0).

9. The (J**T)*J run found $f(x^*) = 3.40 \times 10^{-3}$ and the $H_0 = I$ run found $f(x^*) = 4\ 71 \times 10^{-4}$.

10 The final function values for the SUMSOL runs were as follows; for (J**T)*J, 359.; for LMAX0=1, 189, for $H_0 = I$, 237.

to select the steps it tries. It uses the same convergence tests as NL2SOL (performed, in fact, by the same ASSESS module), so the return codes in the columns labeled C in Table V have the same meaning as for the earlier tables. Like NL2SOL, SUMSOL employs a scale matrix D, which can be updated from the diagonal elements of the Hessian approximation, but to eliminate the effects of different updates to D, we report only results for D = I here. The columns labeled NL2SOL, D = I repeat the D = I columns of Table III. Those labeled (J**T)*J show what happens when the initial Hessian approximation supplied to SUMSOL is $H_0 = J_0^T J_0$, where $J_0 = J(x_0)$ is the initial Jacobian matrix. (SUMSOL actually works only with the Cholesky factor L of its Hessian approximation $H = LL^T$, and the initial L supplied in the (J**T)*J run was obtained from a QR factorization of J_0.) The columns labeled LMAX0=1 show what happens when the initial step bound is decreased from the default value that NL2SOL uses, that is, 100., to the default value for SUMSOL, that is, 1.0, and everything else is the same as for the (J**T)*J run. The columns labeled H_0 = I show what happens when SUMSOL sets its initial Hessian approximation to the identity matrix with everything else as for the LMAX0=1 run. Except as listed in the notes in Table V, all runs found the final function value reported in Table II. None of the SUMSOL runs dominates or is dominated by any of the other SUMSOL runs. On problems where both find the same locally optimal function value, NL2SOL generally requires fewer—sometimes substantially fewer—function and gradient evaluations than SUMSOL, so in cases where function evaluations are expensive, Table V suggests that it is quite worthwhile to exploit the structure present in the least-squares Hessian.

8. CODE SIZE AND TIMING

NL2SOL is substantially larger than a simple Levenberg-Marquardt code, and its size deserves some explanation. The following remarks about code size refer to the object code produced by the version of IBM's FORTHX compiler (optimization level 2) available under CMS at M.I.T. when this work was done. We may regard somewhere between 35 and 40 percent of the code as constituting a Levenberg-Marquardt code. Another 30 percent of the code takes care of switching models and using the augmented model. The remainder of the code is devoted to such "extras" as computing covariance matrices, printing an iteration summary and certain initial and final information, providing default values for various inputs, checking the validity of certain input parameters and reporting ones that have nondefault values, and computing a finite-difference Jacobian approximation (subroutine NL2SNO).

One feature that increases the code size by somewhere between 5 and 10 percent is the option of providing the residual vector and Jacobian matrix by reverse communication: one initially calls NL2ITR, passing in the starting guess x_0 along with $R(x_0)$ and $J(x_0)$. Whenever NL2ITR requires R or J to be evaluated at a new point x, it returns with a special return code specifying what is needed; one computes the required values and calls NL2ITR again. Subroutine NL2SOL interacts with NL2ITR, using subroutines provided by its caller to compute $R(x)$ and $J(x)$. Subroutine NL2SNO also interacts with NL2ITR, using a subroutine provided by its caller to compute $R(x)$ and approximating $J(x)$ by forward differences. Reverse communication is vital in applications where the calculation of $R(x)$ is so elaborate that it requires a sequence of overlays.

The somewhat elaborate scheme described above for switching models and choosing the new trust region also contributes to the code size. In particular, we had to code a number of things two ways, one assuming that we have the Jacobian matrix, the other assuming that we have its QR factorization, since we save scratch storage by overwriting the Jacobian matrix with its QR factorization (or, more precisely, with the R matrix and the information needed to multiply vectors by Q and Q^T).

We have conducted some timing experiments with NL2SOL and with a recent version of Moré's [32] excellent code LMDER with the aim of discovering how much adaptive modeling and reverse communication cost in terms of execution time. To eliminate time differences due to the step-computing codes, we modified LMDER so that it called the same step-computing code (LMSTEP) that NL2SOL uses. When trying to assess the cost of reverse communication, we also modified NL2SOL to make it act like LMDER, in that it used only the Gauss-Newton model, did not update S, and updated the trust radius and scale vector in the same way as LMDER. We ran both codes for 5 function evaluations on problems SINGULAR, CHEBQD8, WATSON6, WATSON12, OSBORNE2, DAVIDON1, and BROWN (see Table I). For most of these problems, having the option of using reverse communication (but not actually using it, i.e., calling NL2SOL) cost less than a 15 percent increase in execution time; only for SINGULAR (23 percent) and WATSON6 (18 percent) did we observe increases larger than 15 percent.

We repeated the timing tests just described with five FORTRAN utility routines (DOTPRD, VAXPY, VCOPY, VSCOPY, and V2NORM, which compute the inner product of two vectors, add a multiple of one vector to another, copy one vector to another, copy a scalar to all components of a vector, and compute the 2-norm of a vector, respectively) with their assembly language equivalents, and the maximum increase in time for reverse communication dropped to less than 15 percent. More significantly, this simple change reduced the execution times by as much as 33 percent (for WATSON20, one of the larger problems in terms of n and p). Thus it appears substantially more worthwhile (on our computer, anyway) to replace a few simple FORTRAN subroutines by their assembly language equivalents than to remove the option of using reverse communication. (It is interesting to note that the object code for the five FORTRAN utility routines amounted to 2072 bytes, while that for our assembly routines was only 472.)

Adaptive modeling, in particular updating the S matrix, also costs some time. We ran the unmodified NL2SOL on the problems mentioned above for five function evaluations, and it took between 10 and 15 percent longer on most of the problems (35 percent on SINGULAR) than did the modified code that always used the Gauss-Newton model and did not update S.

All our tests used problems whose residual vectors and Jacobian matrices are relatively cheap to compute. On some problems of more practical interest, the ability to find a solution quickly (i.e., in a small number of function evaluations) and reliably is very important. Our experience with NL2SOL suggests that it is well suited to solving such problems.

REFERENCES

1 ALLEN, D M Private communication, 1976

2 BARD, Y Comparison of gradient methods for the solution of nonlinear parameter estimation problems *SIAM J Numer Anal 7* (1970), 157–186

3 BARD, Y *Nonlinear Parameter Estimation* Academic Press, New York, 1974

4 BATES, D M , AND WATTS, D G An orthogonality convergence criterion for nonlinear least squares Queen's Mathematical Preprint 1979-14, Queen's Univ , Kingston, Ont., Canada, 1979

5 BEALE, E M L On an iterative method for finding a local minimum of a function of more than one variable Tech Rep 25, Statistical Techniques Research Group, Princeton Univ., Princeton, N J , 1958

6 BELSLEY, D A On the efficient computation of the nonlinear full-information maximum likelihood estimator Tech Rep 5, Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, Mass , 1980

7 BETTS, J T Solving the nonlinear least squares problem Application of a general method *J Optim Theory Appl 18* (1976), 469–484

8 BOX, M J A comparison of several current optimization methods and the use of transformations in constrained problems *Comput J 9* (1966), 67–77

9 BRANIN, F H Widely convergent method for finding multiple solutions of simultaneous nonlinear equations *IBM J Res Develop 16* (1971), 504–522

10 BROWN, K M , AND DENNIS, J.E A new algorithm for nonlinear least-squares curve fitting In *Mathematical Software*, J R Rice, Ed., Academic Press, New York, 1971, 391–396.

11 COLVILLE, A R A comparative study of nonlinear programming codes Tech Rep 320-2949, IBM New York Scientific Center, 1968

12 CRAGG, E E , AND LEVY, A V Study on a supermemory gradient method for the minimization of functions *J Optim Theory Appl 4* (1969), 191–205

13 DAVIDON, W C New least-square algorithms *J Optim Theory Appl 18* (1976), 187–197.

14 DENNIS, J E Some computational techniques for the nonlinear least squares problem In *Numerical Solutions of Systems of Nonlinear Equations*, G D Byrne and C A Hall, Eds , Academic Press, New York, 1973, pp 157–183

15 DENNIS, J E Nonlinear least squares and equations In *The State of the Art in Numerical Analysis*, D Jacobs, Ed , Academic Press, London, 1977, pp 269–312

16 DENNIS, J E , AND MEI, H.H -W Two new unconstrained optimization algorithms which use function and gradient values *J Optim Theory Appl 28* (1979), 453–482

17 DENNIS, J E , AND MORÉ, J J Quasi-Newton methods, motivation and theory *SIAM Rev 19* (1977), 46–89

18. DENNIS, J.E , AND WELSCH, R E. Techniques for nonlinear least squares and robust regression. *Commun Statist B7* (1978), 345–359

19 ENGVALL, J L Numerical algorithm for solving over-determined systems of nonlinear equations NASA Document N70-35600, 1966

20 FLETCHER, R Function minimization without evaluating derivatives—A review *Comput J 8* (1965), 33–41

21 FLETCHER, R A modified Marquardt subroutine for nonlinear least squares Rep R6799, AERE, Harwell, England, 1971

22 FLETCHER, R., AND POWELL, M.J D A rapidly convergent descent method for minimization. *Comput J 6* (1963), 163–168

23 FREUDENSTEIN, F , AND ROTH, B Numerical solution of systems of nonlinear equations *J. ACM 10*, 4 (Oct 1963), 550–556

24 GAY, D M Computing optimal locally constrained steps *SIAM J Sci Statist. Comput 2*, 2 (June 1981), 186–197

25 GAY, D M Subroutines for general unconstrained minimization using the model/trust-region approach Tech Rep 18, Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, 1980

26 GILL, P E , AND MURRAY, W Algorithm for the solution of the nonlinear least-squares problem *SIAM J Numer. Anal 15* (1978), 977-992
27. GOLUB, G H Matrix decompositions and statistical calculations In *Statistical Computation*, R.C Milton and J.A. Nelder, Eds , Academic Press, New York, 1969, pp 365-397
28 JENNRICH, R I , AND SAMPSON, P F Application of step-wise regression to nonlinear estimation. *Technometrics 10* (1968), 63-72
29 KOWALIK, J S , AND OSBORNE, M R *Methods for Unconstrained Optimization Problems*, American Elsevier, New York, 1968
30 MEYER, R R Theoretical and computational aspects of nonlinear regression In *Nonlinear Programming*, J B Rosen, O L Mangasarian, and K Ritter, Eds , Academic Press, New York, 1970
31 MORÉ, J J The Levenberg-Marquardt algorithm Implementation and theory In *Lecture Notes in Mathematics No 630 Numerical Analysis*, G Watson, Ed , Springer-Verlag, New York, 1978, pp 105-116
32 MORÉ, J J. Implementation and testing of optimization software DAMTP Rep 79/NA4, Cambridge Univ , Cambridge, England, 1979
33 OREN, S S Self-scaling variable metric algorithms without line search for unconstrained minimization *Math Comput 27* (1973), 873-885.
34 OSBORNE, M R Some aspects of nonlinear least squares calculations In *Numerical Methods for Nonlinear Optimization*, F A Lootsma, Ed., Academic Press, New York, 1972
35 POWELL, M J D An iterative method for finding stationary values of a function of several variables *Comput J 5* (1962), 147-151
36 POWELL, M J.D A FORTRAN subroutine for unconstrained minimization, requiring first derivatives of the objective function. Rep AERE-R.6469, AERE Harwell, England, 1970.
37 PRATT, J W When to stop a quasi-Newton search for a maximum likelihood estimate Working Paper 77-16, Harvard School of Business, Cambridge, Mass., 1977
38 RAO, C R. *Linear Statistical Inference and Its Applications*, 2nd ed , Wiley, New York, 1973.
39 REINSCH, C H. Smoothing by spline functions. II *Numer Math 16* (1971), 451-454
40 ROSENBROCK, H H An automatic method for finding the greatest or least value of a function *Comput J 3* (1960), 175-184
41 WEDIN, P -A The non-linear least squares problem from a numerical point of view, I and II Comput Sci Tech Reps , Lund Univ , Lund, Sweden, 1972 and 1974.
42 WEDIN, P -A On surface dependent properties of methods for separable non-linear least squares problems ITM Arbetsrapport nr 23, Inst for Tellampad Matematik, Stockholm, Sweden, 1974.
43 WEDIN, P -A. On the Gauss-Newton method for the non-linear least squares problem ITM Arbetsrapport nr 24, Inst for Tellampad Matematik, Stockholm, Sweden, 1974.
44 ZANGWILL, W J Nonlinear programming via penalty functions *Manage Sci 13* (1967), 344-358

Received September 1977, revised August 1979 and September 1980, accepted April 1981