ALGORITHM 573
NL2SOL—An Adaptive Nonlinear Least-Squares Algorithm [E4]

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

Key Words and Phrases unconstrained optimization, nonlinear least squares, nonlinear regression, quasi-Newton methods, secant methods
CR Categories 5 14, 5 5
Language FORTRAN

1 PURPOSE

Given a continuously differentiable function (residual vector) $R(x) = (R_1(x), R_2(x), \ldots, R_n(x))^T$ of p parameters $x = (x_1, x_2, \ldots, x_p)^T$, NL2SOL attempts to find a parameter vector x^* that minimizes the sum-of-squares function $F(x) = \sum_{i=1}^{n} R_i(x)^2$.

2 METHOD

Reference [1] explains the algorithm realized by NL2SOL in detail. The algorithm amounts to a variation on Newton's method in which part of the Hessian matrix is computed exactly and part is approximated by a secant (quasi-Newton) updating method. Once the iterates come sufficiently close to a local solution, they usually converge quite rapidly. To promote convergence from poor starting guesses, NL2SOL uses a model/trust-region technique along with an adaptive

Received 13 September 1977, revised 18 August 1979 and 25 September 1980; accepted 8 April 1981

Research leading to the NL2SOL package 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
Authors' addresses J E Dennis, Jr , 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.

choice of the model Hessian. Consequently, the algorithm sometimes reduces to a Gauss-Newton or Levenberg-Marquardt method. On large residual problems (in which $F(x^*)$ is large), however, NL2SOL often works much better than these methods.

3 DESCRIPTION

3.1 Calling Sequence

CALL NL2SOL (N, P, X, CALCR, CALCJ, IV, V, UIPARM, URPARM, UFPARM)

Note: NL2SOL is written in American National Standard FORTRAN (1966) and the comments below assume that the calling program is also written in FORTRAN. These comments refer to the single-precision version of NL2SOL. In the double-precision version, all quantities termed REAL below are actually DOUBLE PRECISION.

N (input INTEGER) is the number of components in the residual vector R.

P (input INTEGER) is the number of parameters on which R depends.

X (I/O REAL array of length P) on input is an initial guess at the desired solution x^*. When NL2SOL returns, X contains the best parameter estimate found so far.

CALCR (input subroutine) computes the residual vector R = R(X) when invoked by

CALL CALCR(N, P, X, NF, R, UIPARM, URPARM, UFPARM)

When CALCR is called, NF is the invocation count for CALCR; it is included for possible use with CALCJ. If X is out of bounds (e.g., if R(X) would overflow), then CALCR should set NF to 0, which will cause a shorter step to be attempted. CALCR should not change N, P, or X and should be declared EXTERNAL in the calling program. R should be declared REAL R(N).

CALCJ (input subroutine) computes the Jacobian matrix J = J(X) of first partials, $J_{ij} = \partial J_i(X)/\partial x_j$, when invoked by

CALL CALCJ(N, P, X, NF, J, UIPARM, URPARM, UFPARM)

When CALCJ is called, NF is the invocation count for CALCR at the time when R(X) was evaluated. The X passed to CALCJ is usually the one passed to CALCR on either its most recent invocation or the one prior to it. Thus if CALCR saves intermediate results for use by CALCJ, then it is possible to tell from NF whether they are valid for the current X (or which copy is valid if two are kept). If J cannot be computed at X, then CALCJ should set NF to 0. CALCJ should not change N, P, or X and should be declared EXTERNAL in the calling program. J should be declared REAL J(N, P).

IV (I/O INTEGER array of length P + 60) on input contains certain values (such as limits on the number of iterations and calls on CALCR) that control the behavior of NL2SOL and on output con-

tains various counts and other items of interest: see Sections 3.3 and 3.4. If IV(1) = 0 on input, then default values are supplied for the input components of both IV and V. The caller may supply nondefault values for selected components of IV and V by CALLing DFAULT(IV, V) and then assigning the appropriate nondefault values before calling NL2SOL.

V (I/O REAL array of length 93 + N(P + 3) + P(3P + 33)/2) on input contains certain values (such as convergence tolerances) that control the behavior of NL2SOL and on output contains various items of interest (such as F(X) and R(X)): see Sections 3.5 and 3.15.

UIPARM (INTEGER array of length determined by the caller) is passed without change to CALCR and CALCJ and may be used by them in any way that the caller may find convenient.

URPARM (REAL array of length determined by the caller), like UIPARM, is passed without change to CALCR and CALCJ.

UFPARM (subroutine), like UIPARM, is passed without change (and without having been invoked) to CALCR and CALCJ. If there is no need for such a subroutine, then on many systems it suffices to pass an arbitrary variable or constant for UFPARM. But if an actual subroutine is passed, then it must be declared EXTERNAL in the calling program.

3.2 Example

Let $n = 3$, $p = 2$, and

$$R(x) = \begin{bmatrix} x_1^2 + x_2^2 + x_1x_2 \\ \sin x_1 \\ \cos x_2 \end{bmatrix}.$$

(This problem is due to Madsen [3].) The following FORTRAN code minimizes $F(x) = \frac{1}{2}R(x)^TR(x)$, starting from the initial guess $(3, 1)^T$, using a single-precision version of NL2SOL.

```
INTEGER IV(62), UI(1)
REAL V(147), X(2), UR(1)
EXTERNAL MADR, MADJ
X(1) = 3.0
X(2) = 1.0
IV(1) = 0
CALL NL2SOL (3, 2, X, MADR, MADJ, IV, V, UI, UR, MADR)
STOP
END
SUBROUTINE MADR (N, P, X, NF, R, UI, UR, UF)
INTEGER N, P, NF, UI(1)
REAL X(P), R(N), UR(1)
EXTERNAL UF
R(1) = X(1)**2 + X(2)**2 + X(1)*X(2)
R(2) = SIN(X(1))
R(3) = COS(X(2))
RETURN
END
SUBROUTINE MADJ (N, P, X, NF, J, UI, UR, UF)
```

```
INTEGER N, P, NF, UI(1)
REAL X(P), J(N, P), UR(1)
EXTERNAL UF
J(1, 1) = 2.0*X(1) + X(2)
J(1, 2) = 2.0*X(2) + X(1)
J(2, 1) = COS(X(1))
J(2, 2) = 0.0
J(3, 1) = 0.0
J(3, 2) = -SIN(X(2))
RETURN
END
```

The main program above passes MADR as CALCR and MADJ as CALCJ. No use is made of UIPARM, URPARM, or UFPARM in this simple example.

When the above is executed, NL2SOL prints the initial X vector, a summary of the iterations performed, the final X vector, and some statistics, including the final F(X) and a covariance matrix. If REAL is changed to DOUBLE PRECISION and the above is run on an IBM 370 computer, then NL2SOL reports relative function convergence (IV(1) = 4—see Section 3.3) after 12 calls on MADR and MADJ and returns X(1) ≈ −0.155437, X(2) = 0.694564, and F(X) = 0.386600.

If, say, we wanted to suppress the iteration summary, we could do so by replacing the statement IV(1) = 0 in the main program by

```
CALL DFAULT(IV, V)
IV(19) = 0
```

(See the description of IV(OUTLEV) in Section 3.4.)

3.3 Return Codes

When NL2SOL returns, IV(1) contains one of the following return codes:

3	= X-convergence. The scaled relative difference between the current parameter vector X and a locally optimal parameter x^* is very likely at most V(XCTOL): see Section 3.5.
4	= relative function-convergence. The relative difference between the current function value and its locally optimal value is very likely at most V(RFCTOL): see Section 3.5.
5	= both X and relative function-convergence, that is, the conditions for IV(1) = 3 and IV(1) = 4 both hold.
6	= absolute function-convergence. The current function value (half the sum of squares) is at most V(AFCTOL): see Section 3.5.
7	= singular convergence. The Hessian near the current X appears to be singular or nearly so, and a step of scaled length at most V(LMAX0) is unlikely to yield a relative function decrease of more than V(RFCTOL). This means that the model is overspecified (i.e., contains too many parameters), at least near X. It is possible that a different starting guess would lead NL2SOL to find an X giving a smaller F(X) and strong convergence (IV(1) = 3, 4, 5, or 6).

8 = false convergence. The iterates appear to be converging to a noncritical point. This may mean that the false convergence tolerance (V(XFTOL)—see Section 3.5) is too large, that the convergence tolerances (V(AFCTOL), V(RFCTOL), V(XCTOL)) are too small for the accuracy to which CALCR and CALCJ compute R and J, that there is an error in computing the Jacobian matrix J, or that R is discontinuous near X.

If the NPRELDF value printed in the summary statistics (or in the iteration summary for the final iteration) is negative and not too much larger than V(RFCTOL) in absolute value, then V(RFCTOL) is too small and singular convergence would be detected if V(RFCTOL) were increased above |NPRELDF|: see Sections 3.5 and 3.11.

9 = function evaluation limit reached without other convergences: see IV(MXFCAL) in Section 3.4.

10 = iteration limit reached without other convergence: see IV(MXITER) in Section 3.4.

11 = STOPX returned .TRUE. (external interrupt): see Section 3.14.

13 = F(X) cannot be computed at the initial X.

14 = bad parameters passed to ASSESS. (This should not occur.)

15 = the Jacobian could not be computed at X (see CALCJ above).

16 = N or P (or parameter NN to NL2ITR) out of range: $P < 0$ or $N < P$ or $NN < N$.

17 = a restart was attempted with N, P, or parameter NN to NL2ITR changed: see Section 3.7.

18 = IV(INITS) out of range: see Section 3.4.

19-45 = V(IV(1)) is out of range.

50 = IV(1) was out of range when NL2SOL (or NL2SNO or NL2ITR) was called.

87 . . . (86 + P) = JTOL(IV(1)-86), that is, V(IV(1)), is not positive: see V(DFAC) in Section 3.5.

Just before NL2SOL returns, a brief description of the return code is printed (unless all printing is turned off by IV(PRUNIT) = 0).

3.4 IV Values

3.4.1 *IV Input Values (Supplied by DFAULT)*

IV(1) . . . IV(1) should have a value between 0 and 12 when NL2SOL is called. 0 and 12 both mean that this is a fresh start; 0 means DFAULT(IV, V) should be invoked to supply default values to the input components of IV and V, while 12 (the value that DFAULT assigns to IV(1)) means that the caller has already called DFAULT(IV, V) and has possibly changed some IV or V entries to nondefault values. IV(1) input values between 3 and 11 mean that NL2SOL should restart; see Section 3.7. Default = 12.

IV(COVPRT) . . . IV(14) = 1 means print a covariance matrix at the solution. This matrix is computed as IV(COVREQ) dictates just before a return

with IV(1) = 3, 4, 5, or 6. IV(COVPRT) = 0 means skip this printing. Default = 1.

IV(COVREQ): IV(15) $\neq 0$ means compute a covariance matrix before a return with IV(1) = 3, 4, 5, or 6. In this case, an approximate covariance matrix is obtained in one of several ways. Let k = |IV(COVREQ)| and let σ = 2F(X)/max (1, N−P), where 2F(X) is the residual sum of squares. If k = 1 or 2, then a finite-difference Hessian approximation H is obtained. If H is positive-definite (or, for k = 3, if the Jacobian has full rank), then one of the following is computed:

$$k = 1 \Rightarrow \sigma H^{-1}(J^{\mathrm{T}}J)H^{-1}$$

$$k = 2 \Rightarrow \sigma H^{-1}$$

$$k = 3 \Rightarrow \sigma(J^{\mathrm{T}}J)^{-1}.$$

If IV(COVREQ) > 0, then both function and gradient values (calls on CALCR and CALCJ) are used in computing H (with step sizes determined by V(DELTA0); see Section 3.5), while if IV(COVREQ) < 0, then only function values (calls on CALCR) are used (with step sizes determined by V(DLTFDC)). If IV(COVREQ) = 0, then no attempt is made to compute a covariance matrix (unless IV(COVPRT) = 1, in which case NL2SOL assumes IV(COVREQ) = 1 and NL2SNO assumes IV(COVREQ) = −1). See IV(COVMAT) below. Default = 1.

IV(DTYPE) ... IV(16) tells how the scale vector D (see [1]) should be chosen. IV(DTYPE) > 0 means choose D as described below with V(DFAC). IV(DTYPE) ≤ 0 means the caller has chosen D and has stored it in V starting at V(94 + 2N + P(3P + 31)/2). Default = 1.

IV(INITS) ... IV(25) tells how the S matrix (see [1]) should be initialized. 0 means initialize S to all zeros and start with the Gauss-Newton model. 1 and 2 mean that the caller has stored the lower triangle of the initial S rowwise in V starting at V(87 + 2P). IV(INITS) = 1 means start with the Gauss-Newton model, while IV(INITS) = 2 means start with the augmented model; see [1]. Default = 0.

IV(MXFCAL) ... IV(17) gives the maximum number of function evaluations (calls on CALCR, excluding those used to compute the covariance matrix and, in the case of NL2SNO, the Jacobian matrices) allowed. If this number does not suffice, then NL2SOL returns with IV(1) = 9. Default = 200.

IV(MXITER) ... IV(18) gives the maximum number of iterations allowed. It also indirectly limits the number of gradient evaluations (calls on CALCJ) to IV(MXITER) + 1. If IV(MXITER) iterations do not suffice, then NL2SOL returns with IV(1) = 10. Default = 150.

IV(OUTLEV) ... IV(19) controls the number and length of iteration summary lines printed (by ITSMRY). IV(OUTLEV) = 0 means do not print any summary lines. Otherwise print a summary line after each |IV(OUTLEV)| iterations. Long summary lines are printed if IV(OUTLEV) > 0, short lines if IV(OUTLEV) < 0. See Section 3.11 for more details. Default = 1.

IV(PARPRT) ... IV(20) = 1 means print any nondefault V values on a fresh start or any changed V values on a restart. IV(PARPRT) = 0 means skip this printing. Default = 1.

IV(PRUNIT) ... IV(21) is the output unit number on which all printing is done. IV(PRUNIT) = 0 means suppress all printing. (Setting IV(PRUNIT) to 0 is the only way to suppress the one-line termination message printed before NL2SOL returns.) Default = standard output unit (unit 6 on most systems); the default for IV(PRUNIT) is actually IMDCON(1); see Section 3.12.

IV(SOLPRT) ... IV(22) = 1 means print the final X (the one returned), along with the final gradient and scale vector D. IV(SOLPRT) = 0 means skip this printing. Default = 1.

IV(STATPR) ... IV(23) = 1 means print summary statistics upon returning. These consist of the function value (half the residual sum of squares) at X, the scaled relative size of the last step taken (see V(RELDX) below), the number of function and gradient evaluations (calls on CALCR and CALCJ, excluding any calls made only for computing covariance matrices), the relative function reductions predicted for the last step taken and for a Newton step (or perhaps a step of length bounded by V(LMAX0)—see the descriptions of PRELDF and NPRELDF in Section 3.11 below), and, if an attempt was made to compute a covariance matrix, the number of calls on CALCR and CALCJ used in trying to compute the covariance matrix. IV(STATPR) = 0 means skip this printing. Default = 1.

IV(X0PRT) ... IV(24) = 1 means print the initial X and scale vector D (on a fresh start only). IV(X0PRT) = 0 means skip this printing. Default = 1.

3.4.2 *IV Output Values of Primary Interest*

IV(1) ... IV(1) is the return code; see Section 3.3.

IV(COVMAT) ... IV(26) tells whether a covariance matrix was computed. If IV(COVMAT) > 0, then the lower triangle of the covariance matrix is stored row-wise in V, starting at V(IV(COVMAT)). If IV(COVMAT) = 0, then no attempt was made to compute a covariance matrix. If IV(COVMAT) = −1, then the finite-difference Hessian H was indefinite (or, for | IV(COVREQ) | = 3, the current Jacobian matrix is rank deficient): like singular convergence (see Section 3.3), this may mean that the model is overspecified (contains too many parameters). And if IV(COVMAT) = −2, then a successful finite-difference step could not be found for some component of X (i.e., CALCR set NF to 0 for each of two trial steps).

Note that IV(COVMAT) is reset to 0 after each successful step, so if such a step is taken after a restart, then the covariance matrix will be recomputed.

IV(D) ... IV(27) is the starting subscript in V of the current scale vector D.

IV(G) ... IV(28) is the starting subscript in V of the current least-squares gradient vector J^TR.

IV(NFCALL) ... IV(6) is the number of calls so far made on CALCR (i.e., function evaluations, including those used in computing covariance matrices).

IV(NFCOV) . . . IV(40) is the number of calls made on CALCR when computing covariance matrices.

IV(NGCALL) . . . IV(30) is the number of calls on CALCJ (gradient evaluations) so far made, including those used in computing covariance matrices.

IV(NGCOV) . . . IV(41) is the number of calls made on CALCJ when computing covariance matrices.

IV(NITER) . . , IV(31) is the number of iterations performed.

IV(R) . . . IV(50) is the starting subscript in V of the residual vector R corresponding to the final X.

3.5 V Values of Primary Interest

Many of the V input components described here and in Section 3.15 must lie in certain intervals. If such a component lies outside the interval indicated for it below (or in Section 3.15) at the beginning of its description, then module PARCHK will print an error message (unless IV(PRUNIT) = 0) and will force NL2SOL to return immediately with IV(1) > 18.

Frequent reference is made below to two quantities: MACHEP and the scale vector D. MACHEP is the unit roundoff for the floating-point arithmetic being used—see Section 3.12. The scale vector D is the diagonal of the diagonal scale matrix D_k discussed in [1, Sections 5 and 7]; this scale matrix is denoted by diag(D) below.

3.5.1 *V Input Values of Primary Interest (Supplied by DFAULT)*

V(AFCTOL) . . . V(31) > 0 is the absolute function convergence tolerance. If NL2SOL finds a point where the function value (half the sum of squares) is less than V(AFCTOL), and if NL2SOL does not return with IV(1) = 3, 4, or 5, then it returns with IV(1) = 6.

$$\text{Default} = \max\{10^{-20}, \text{MACHEP}^2\}.$$

V(DELTA0) . . . V(44) $\in$ [MACHEP, 1] is a factor used in choosing the finite-difference step sizes used in computing covariance matrices when IV(COVREQ) = 1 or 2. For differences involving X(i), step size

$$\text{V(DELTA0)} \cdot \max\{|\text{X}(i)|, 1/\text{D}(i)\} \cdot \text{sign}(\text{X}(i))$$

is used, where D is the current scale vector; see [1]. If this results in CALCR setting NF to 0, then −0.5 times this step is also tried. Default = MACHEP$^{1/2}$.

V(DFAC) . . . V(41) $\in$ [0, 1] and the D0 and JTOL arrays (see V(D0INIT) and V(JTINIT)) are used in updating the scale vector D when IV(DTYPE) > 0. (D is initialized according to V(DINIT).) Let

$$\text{D1}(i) = \max\{[\text{JCNORM}(i)^2 + \max\{S_{ii}, 0\}]^{1/2}, \text{V(DFAC)D}(i)\},$$

where JCNORM(i) is the 2-norm of the ith column of the current Jacobian matrix and S is the S matrix of [1]. If IV(DTYPE) = 1, then D(i) is set to D1(i) unless D1(i) < JTOL(i), in which case D(i) is set to max{(D0(i), JTOL(i)}. If IV(DTYPE) > 1, then D is updated during the first iteration as for IV(DTYPE)

= 1 (after any initialization due to V(DINIT)) and is left unchanged thereafter. Default = 0.6.

V(DINIT) ... V(38) ≥ −10: if V(DINIT) ≥ 0, then it is the value to which all components of the scale vector D are initialized during a fresh start. Default = 0.

V(DLTFDC) ... V(40) ∈ [MACHEP, 1] helps choose the step sizes used in computing covariance matrices when IV(COVREQ) = −1 or −2. For differences involving X(i), the step size first tried is

$$\text{V(DLTFDC)} \cdot \max\{|\text{X}(i)|, 1/\text{D}(i)\},$$

where D is the current scale vector (see [1]). If this step is too big the first time it is tried, that is, if CALCR sets NF to 0, then −0.5 times this step is also tried. Default = MACHEP$^{1/3}$.

V(DLTFDJ) ... V(36) ∈ [MACHEP, 1] helps choose the step sizes used when NL2SNO computes a finite-difference approximation to the Jacobian matrix. For differences involving X(i), the step size first tried is

$$\text{V(DLTFDJ)} \cdot \max\{|\text{X}(i)|, 1/\text{D}(i)\},$$

where D is the current scale vector (see [1]). If the first step is too big, that is, if CALCR sets NF to 0, then smaller steps are tried until the step size is shrunk below 1000 · MACHEP. Default = MACHEP$^{1/2}$.

V(D0INIT) ... V(37) ≥ 0: if V(D0INIT) > 0, it is the value to which all components of the D0 vector (see V(DFAC)) are initialized. If V(D0INIT) = 0, then it is assumed that the caller has stored D0 in V starting at V(P + 87). Default = 1.0.

V(JTINIT) ... V(39) ≥ 0: if V(JTINIT) > 0, it is the value to which all components of the JTOL array (see V(DFAC)) are initialized. If V(JTINIT) = 0, then it is assumed that the caller has stored JTOL in V starting at V(87). Default = 10^{-6}.

V(LMAX0) ... V(35) > 0 gives the maximum 2-norm allowed for diag(D) times the very first step that NL2SOL attempts. It is also used in testing for singular convergence: if the function reduction predicted for a step of length bounded by V(LMAX0) is at most V(RFCTOL) $|F_0|$, where F_0 is the function value at the start of the current iteration, and if NL2SOL does not return with IV(1) = 3, 4, 5, or 6, then it returns with IV(1) = 7. Default = 100.

V(RFCTOL) ... V(32) ∈ [MACHEP, 0.1] is the relative function-convergence tolerance. If the current model predicts a maximum possible function reduction (see V(NREDUC)) of at most V(RFCTOL) $|F_0|$, where F_0 is the function value at the start of the current iteration, and if the last step attempted achieved no more than twice the predicted function decrease, then NL2SOL returns with IV(1) = 4 (or 5). Default = max{10^{-10}, MACHEP$^{2/3}$}.

V(TUNER1) ... V(26) ∈ [0, 0.5] helps decide when to check for false convergence and to consider switching models. This is done if the actual function decrease

from the current step is no more than V(TUNER1) times its predicted value. Default = 0.1.

V(XCTOL) ... V(33) ∈ [0, 1] is the X-convergence tolerance. If a Newton step (see V(NREDUC)) is tried that has V(RELDX) ≤ V(XCTOL) and if this step yields at most twice the predicted function decrease, then NL2SOL returns with IV(1) = 3 (or 5). Default = MACHEP$^{1/2}$.

V(XFTOL) ... V(34) ∈ [0, 1] is the false-convergence tolerance. If a step is tried that gives no more than V(TUNER1) times the predicted function reduction and that has V(RELDX) ≤ V(XFTOL), and if NL2SOL does not return with IV(1) = 3, 4, 5, 6, or 7, then it returns with IV(1) = 8. (See the description of V(RELDX) below.) Default = 100·MACHEP.

V(*) ... DFAULT supplies to V a number of tuning constants, with which it should normally be unnecessary to tinker. See Section 3.15.

3.5.2 *V Output Values of Primary Interest*

V(DGNORM) ... V(1) = $\| \operatorname{diag}(D)^{-1} g \|_2$, where g is the most recently computed gradient and D is the corresponding scale vector.

V(DSTNRM) ... V(2) = $\| \operatorname{diag}(D) \Delta x \|_2$, where Δx is the most recently computed step and D is the current scale vector.

V(F) ... V(10) is the current function value (half the residual sum of squares).

V(F0) ... V(13) is the function value at the start of the current iteration.

V(NREDUC) ... V(6), if positive, is the maximum function reduction possible according to the current model, that is, the function reduction predicted for a Newton step: $\Delta x = -H^{-1} g$, where $g = J^T R$ is the current gradient and H is the current Hessian approximation:

$H = J^T J$ for the Gauss–Newton model

$H = J^T J + S$ for the augmented model.

V(NREDUC) = 0 means H is not positive definite.

If V(NREDUC) < 0, then V(NREDUC) is used in the singular convergence test: It is the negative of the function reduction predicted for a step computed with a step bound of V(LMAX0).

V(PREDUC) ... V(7) is the function reduction predicted (by the current quadratic model) for the current step. This (divided by V(F0)) is used in testing for relative function convergence.

V(RADIUS) ... V(8) is the trust region radius (i.e., step bound) used for the last step tried.

V(RELDX) ... V(17) is the scaled relative change in X caused by the current step, Δx, computed as

$$\max\{|D(i)[X(i) - X_0(i)]|\}/\max\{D(i)[|X(i)| + |X_0(i)|]\},$$

where $X = X_0 + \Delta x$.

3.6 Finite-Difference Jacobians: NL2SNO

Those who do not wish to code a subroutine CALCJ for (analytically) computing the Jacobian matrix may avoid doing so by calling NL2SNO instead of NL2SOL. NL2SNO computes an approximate Jacobian matrix by forward differences (using a step size determined by V(DLTFDJ)—see Section 3.5). The calling sequence for NL2SNO amounts to the one for NL2SOL with CALCJ omitted:

```
CALL NL2SNO (N, P, X, CALCR, IV, V, UIPARM, URPARM, UFPARM)
```

The parameters for NL2SNO are the same as the corresponding ones for NL2SOL, with the minor exception of IV(COVREQ): If IV(COVPRT) = 1 and IV(COVREQ) = 0, then NL2SNO sets IV(COVREQ) to −1; otherwise, it sets IV(COVREQ) to −|IV(COVREQ)|. Thus NL2SNO uses function values only in computing covariance matrices and V(DELTA0) is not used.

3.7 Restarting

After any return with $3 \leq \text{IV}(1) \leq 11$, it is possible to change some of the IV and V input components (such as the convergence tolerances and the iteration and function evaluation limits) and call NL2SOL (or NL2SNO) again with IV(1) unchanged. This causes the algorithm to be resumed at the point where it was interrupted. (It is even possible to save IV, V, and X and then restart in a subsequent run.)

3.8 Scaling

Problems sometimes arise that are poorly scaled in the sense that the various components of X are expressed in widely differing units. With the default choice of the scale vector D (see V(DFAC) and the beginning of Section 3.5), the behavior of NL2SOL is largely insensitive to this kind of poor scaling. On well-scaled problems, the performance of NL2SOL can sometimes be improved by choosing D to be a vector of ones, that is, by setting IV(DTYPE) to 0 and V(DINIT) to 1.0. Occasionally it may also be worthwhile to fix $D(i)$, $1 \leq i \leq P$, at the 2-norm of the ith column of the initial Jacobian matrix by setting IV(DTYPE) to 2.

3.9 LMAX0: The Initial Step Bound

On some problems it is necessary to give V(LMAX0) = V(35) a small value to prevent a disasterously large first step, one that might result in exponent overflow or arguments out of range to intrinsic functions. Even if no disaster occurs, if NL2SOL takes a number of function evaluations on the first iteration, then this number can be reduced on subsequent reruns by setting V(LMAX0) to the value in the D*STEP column of the iteration summary for iteration 1.

3.10 Local Solutions

It can easily happen that NL2SOL only finds a local minimizer of the sum-of-squares function F(X) and that a different starting guess would cause a point to be found at which F has a still smaller value. Except for cases where special conditions (such as convexity of the objective function) prevail, this shortcoming is shared by all minimization algorithm implementations.

3.11 Printed Output

Any printing is done by one of two modules: ITSMRY and PARCHK. PARCHK reports any V input components that are out of range and optionally lists any such components that have nondefault or changed values (on a fresh start or restart, respectively). ITSMRY does the remaining printing. Various IV input components control what printing is done; see Section 3.4.

If IV(OUTLEV) > 0, then ITSMRY produces an iteration summary which includes the following values: IT, the current iteration number; NF, the number of function evaluations (calls on CALCR), excluding any extra ones needed for computing covariance matrices and, in the case of NL2SNO, excluding the extra ones needed to compute finite-difference Jacobian matrices; F, the current function value (half the residual sum of squares); RELDF, the relative difference between the previous and the current function value (i.e., the difference in function values divided by the previous function value); PRELDF, the value of RELDF predicted by the quadratic model used to compute the step just taken; RELDX, the relative change in X caused by the step just taken—see V(RELDX) in Section 3.5; MODEL, a code that tells which models were used in choosing the current step (G = the Gauss–Newton model; S = the augmented model; G-S means the Gauss–Newton model was tried first and a switch was then made to the augmented model; S-G, G-S-G, and S-G-S have analogous meanings); STPPAR, the Marquardt parameter λ for the last step, Δx, computed: $\lambda > 0$ means Δx satisfies

$$[H + \lambda \operatorname{diag}(D)^2]\Delta x = -g,$$

where H and g are the old Hessian approximation and gradient, respectively (the ones used in computing the step just taken); SIZE, the sizing factor used in updating the S matrix (see [1]); D*STEP, the 2-norm of diag(D) times the step just taken (see V(DSTNRM) in Section 3.5); and NPRELDF: if NPRELDF > 0, then it is the relative function reduction (i.e., value of RELDF) predicted for a full Newton step; if NPRELDF = 0, then the Hessian approximation failed to be positive definite; and if NPRELDF < 0, then it is the negative of the relative function reduction predicted for a step of length bounded by V(LMAX0). These summary lines are produced every IV(OUTLEV) iterations, and they are 118 characters long (including the carriage control character). If IV(OUTLEV) < 0, then short summary lines are produced every −IV(OUTLEV) iterations; these lines are 79 characters long (55 if IV(COVPRT) = 0), and they include only the first six items listed above (i.e., IT, NF, F, RELDF, PRELDF, and RELDX).

3.12 Changing Computers

The NL2SOL distribution tape contains both single- and double-precision versions of the NL2SOL source code, so it should be unnecessary to change precisions. (On computers having only 32 or 36 bits per REAL word, double precision often gives better performance.)

Only the functions IMDCON and RMDCON contain machine-dependent constants. To change from one computer to another, it should suffice to change the DATA statements in these functions. The DATA statement in IMDCON sets IMDCON(1) to the output unit number that DFAULT supplies to

IV(PRUNIT). The machine-dependent DATA statement in RMDCON provides three values: BIG, ETA, and MACHEP. BIG is the largest floating-point number such that a FORTRAN program can compute SQRT(0.999*BIG)**2 (i.e., DSQRT(0.999D0*BIG)**2 in double precision) without overflowing. Similarly, ETA is the smallest floating-point number such that SQRT(1.001*ETA)**2 (or DSQRT(1.001D0*ETA)**2, respectively) does not underflow. MACHEP is the unit roundoff, that is, the smallest floating-point number such that 1 + MACHEP yields a stored floating-point number greater than 1. (Some computers feature registers that carry more bits than can be stored; MACHEP should only reflect the accuracy of numbers that can be stored.) DATA statements giving suitable values for BIG, ETA, and MACHEP for a variety of computers appear as comments in RMDCON.

Intrinsic functions are explicitly declared in the NL2SOL source code. On certain computers (e.g., Univac), it may be necessary to comment out these declarations. So that this may be done automatically by a simple program, such declarations are preceded by a comment having C/+ in columns 1–3 and blanks in columns 4–72 and are followed by a comment having C/ in columns 1 and 2 and blanks in columns 3–72.

3 13 Using Reverse Communication: NL2ITR

Instead of writing subroutines CALCR and CALCJ to compute the residual vector R(X) and Jacobian matrix J(X), one can call NL2ITR and provide R and J by reverse communication. The calling sequence is

```
CALL NL2ITR (D, IV, J, N, NN, P, R, V, X)
```

Parameters IV, N, P, V, and X are the same as the corresponding ones to NL2SOL, with the following exceptions: V need only contain 93 + 2N + P(3P + 31)/2 elements, since the storage that NL2SOL and NL2SNO allocate for D, J, and R at the end of V is not needed; and components IV(D), IV(J), and IV(R) are not referenced. D is the scale vector (dimensioned D(P)). NN is the (integer) lead dimension for the J array, which is dimensioned J(NN, P); NN must satisfy NN ≥ N.

When NL2ITR is first called (with IV(1) = 0 or 12), J must have been set to J(X), R to R(X). When NL2ITR wants R to be evaluated at a new X, it returns with IV(1) = 1; the caller should then set R to R(X) (unless X is out of range, in which case the caller should set IV(TOOBIG), that is, IV(2), to 1) and call NL2ITR again. Similarly, when NL2ITR wants J to be evaluated at X, it returns with IV(1) = 2, and the caller should then set J to J(X) and call NL2ITR again. (If J cannot be evaluated at X, the caller may set IV(NFGCAL), that is, IV(7), to 0; this will cause NL2ITR to give the error return IV(1) = 15.)

3 14 STOPX

It is possible to arrange for NL2SOL (NL2SNO and NL2ITR) to be interrupted before each evaluation of R(X) when used in an interactive environment. To do this, it is necessary to replace the logical function STOPX supplied with the NL2SOL package (which always returns .FALSE.) by a system-dependent STOPX that returns .TRUE. if and only if the "break" (i.e., "interrupt") key has

been pressed since the last call on STOPX. Once this is done, NL2SOL will return with IV(1) = 11 when the "break" key is pressed before some other return has occurred. It is then possible to change some of the IV and V input components and restart; see Section 3.7.

3.15 Other V Input Values

V(COSMIN) ... V(43) $\in$ [MACHEP, 1] is the minimum absolute cosine allowed between the step just taken, Δx, and the corresponding change in gradients, Δg, for a full update of the S matrix to be made. If $|\Delta x^T \Delta g| / (\|\Delta x\|_2 \cdot \|\Delta g\|_2) <$ V(COSMIN), then $\Delta x^T \Delta g$ is replaced in the update formula by sign $(\Delta x^T \Delta g)$V(COSMIN) $\|\Delta x\|_2 \|\Delta g\|_2$. Default = max{$10^{-6}$, 100 MACHEP}.

V(DECFAC) ... V(22) $\in$ [0.01, 0.8] is the factor by which the trust region radius is shrunk if CALCR sets NF to 0 (or NL2ITR is called with IV(1) = 1 and IV(TOOBIG) = 1). Default = 0.5.

V(EPSLON) ... V(19) $\in$ [0.001, 0.9] is the maximum relative difference allowed between $g^T \Delta x + 1/2 \Delta x^T H \Delta x$ and its optimal value subject to the constraint $\|\text{diag}(D)\Delta x\|_2 \leq$ V(RADIUS), where Δx is the step being computed, g is the current gradient, and H is the current Hessian approximation. This is used in detecting and handling the special case discussed in [2]. Default = 0.1.

V(FUZZ) ... V(45) $\in$ [1.01, 100] is used in the test that decides whether to switch models. If q is the current model for F (near the point X) and $\tilde{q}$ is the other model, and if

$$\text{V(FUZZ)}\,|\tilde{q}(\text{X} + \Delta x) - \text{F(X} + \Delta x)| < |q(\text{X} + \Delta x) - \text{F(X} + \Delta x)|,$$

then the models are switched. Default = 1.5.

V(INCFAC) ... V(23) $\in$ [1.2, 100] is the minimum factor by which the trust region radius is increased (when it is increased at all). Default = 2.

V(PHMNFC) ... V(20) $\in$ [−0.99, −0.001] is the minimum value allowed for $[\|\text{diag}(D)\Delta x\|_2 - \text{V(RADIUS)}]/\text{V(RADIUS)}$. Default = −0.1.

V(PHMXFC) ... V(21) $\in$ [1.2, 100] is the maximum value allowed for $[\|\text{diag}(D)\Delta x\|_2 - \text{V(RADIUS)}]/\text{V(RADIUS)}$. Default = 0.1.

V(RDFCMN) ... V(24) $\in$ [0.01, 0.8] is the minimum factor by which the trust region radius, V(RADIUS), may be shrunk. Default = 0.1.

V(RDFCMX) ... V(25) $\in$ [1.2, 100] is the maximum factor by which the trust region radius, V(RADIUS), may be increased at one time. Default = 4.0.

V(RLIMIT) ... V(42) $\geq 10^{10}$ is the largest value allowed for $\|\text{R(X)}\|_2$ before F(X) is considered to overflow. Default = $(0.999 \cdot \text{BIG})^{1/2}$, where BIG is described in Section 3.12.

V(TUNER2) ... V(27) $\in$ [0, 0.5]. For a step to be accepted, the actual function reduction must be more than V(TUNER2) times its predicted value. Default = 0.0001.

V(TUNER3) ... V(28) ∈ [0.001, 1]. If the actual function decrease is at least V(TUNER3) times the inner product of the step and the gradient (at the start of the step), then the trust region radius is increased. Default = 0.75.

V(TUNER4) ... V(29) ∈ [−1, 1]. If the disposition of the new radius has not yet been decided and either

$$\| \operatorname{diag}(D)^{-1}[H\Delta x - (g - g_0)] \| < \mathrm{V(TUNER4)} \| \operatorname{diag}(D)^{-1} g \|$$

or $g^T \Delta x < \mathrm{V(TUNER5)}\, g_0^T \Delta x$, where Δx is the step just taken, H is the Hessian approximation used in computing Δx, g_0 is the old gradient, g is the new gradient, and D is the newly updated scale vector, then the radius is increased by a factor of V(INCFAC). Otherwise, it is left unchanged. Default = 0.5.

V(TUNER5) ... V(30) ≥ MACHEP is described above with V(TUNER4). Default = 0.75.

3.16 Storage Requirements

NL2SOL, NL2SNO, and the subroutines from the NL2SOL package that they call amount to around 2360 FORTRAN statements (including nonexecutable statements, such as type statements, but excluding comments); the many comments bring the source code up to nearly 5200 lines. When compiled by the *H*-extended compiler on the IBM 370/168 at the Massachusetts Institute of Technology, this source code results in about 56,300 bytes of object code. The amount of variable storage used is listed above in Section 3.1.

REFERENCES

1. DENNIS, J.E., GAY, D.M., AND WELSCH, R.E. An adaptive nonlinear least-squares algorithm *ACM Trans Math Softw 7*, 3 (Sept. 1981), 348–368.
2. GAY, D.M. Computing optimal locally constrained steps. *SIAM J Sci. Statist. Comput 2*, 2 (June 1981), 186–197.
3. MADSEN, K. An algorithm for minimax solution of overdetermined systems of nonlinear equations. Rep. TP 559, AERE Harwell, England, 1973.