ANL-80-74 ANL-80-74

USER GUIDE FOR MINPACK-1

by

Jorge J. Moré, Burton S. Garbow,
and Kenneth E. Hillstrom

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Prepared for the U. S. DEPARTMENT OF ENERGY
under Contract W-31-109-Eng-38

Printed in the United States of America
Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A12
Microfiche copy: A01

Distribution Category:
Mathematics and Computers
(UC-32)

ANL-80-74

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

USER GUIDE FOR MINPACK-1

by

Jorge J. Moré, Burton S. Garbow, Kenneth E. Hillstrom

Applied Mathematics Division

August 1980

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

TABLE OF CONTENTS

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

ABSTRACT

MINPACK-1 is a package of Fortran subprograms for the numerical solution of systems of nonlinear equations and nonlinear least squares problems. This report provides an overview of the algorithms and software in the package and includes the documentation and program listings.

Preface

The MINPACK Project is a research effort whose goal is the development of a systematized collection of quality optimization software. The first step towards this goal has been realized in MINPACK-1, a package of Fortran programs for the numerical solution of systems of nonlinear equations and nonlinear least squares problems.

The design of the algorithms and software in MINPACK-1 has several objectives; the main ones are reliability, ease of use, and transportability.

At the algorithmic level, reliability derives from the underlying algorithms having a sound theoretical basis. Entirely satisfactory global convergence results are available for the MINPACK-1 algorithms and, in addition, their properties allow scale invariant implementations.

At the software level, reliability derives from extensive testing. The heart of the testing aids is a large collection of test problems (Moré, Garbow, and Hillstrom [1978]). These test problems have been used to measure the performance of the software on the following computing systems: IBM 360/370, CDC 6000-7000, Univac 1100, Cray-1, Burroughs 6700, DEC PDP-10, Honeywell 6000, Prime 400, Itel AS/6, and ICL 2980. At Argonne, software performance has been further measured with the help of WATFIV and BRNANL (Fosdick [1974]). WATFIV detects run-time errors such as undefined variables and out-of-range subscripts, while BRNANL provides execution counts for each block of a program and, in particular, has established that the MINPACK-1 test problems execute every non-trivial program block.

Reliability further implies efficient and robust implementations. For example, MINPACK-1 programs access matrices sequentially along columns (rather than rows), since this improves efficiency, especially on paged systems. Also, there are extensive checks on the input parameters, and computations are formulated to avoid destructive underflows and overflows. Underflows can then be safely ignored; overflows due to the problem should of course be investigated.

Ease of use derives from the design of the user interface. Each algorithmic path in MINPACK-1 includes a core subroutine and a driver with a simplified calling sequence made possible by assuming default settings for certain parameters and by returning a limited amount of information; many applications do not require full flexibility and in these cases the drivers can be invoked. On the other hand, the core subroutines enable, for example, scaling of the variables and printing of intermediate results at specified iterations.

Ease of use is also facilitated by the documentation. Machine-readable documentation is provided for those programs normally called by the user. The documentation includes discussions of all calling sequence parameters and an actual example illustrating the use of the corresponding algorithm. In addition, each program includes detailed prologue comments on its purpose and the roles of its parameters; in-line comments introduce major blocks in the body of the program.

To further clarify the underlying structure of the algorithms, the programs have been formatted by the TAMPR system of Boyle and Dritz [1974]. TAMPR produces implementations in which the loops and logical structure of the programs are clearly delineated. In addition, TAMPR has been used to produce the single precision version of the programs from the master (double precision) version.

Transportability requires that a satisfactory transfer to a different computing system be possible with only a small number of changes to the software. In MINPACK-1, a change to a new computing system only requires changes to one program in each precision; all other programs are written in a portable subset of ANSI standard Fortran acceptable to the PFORT verifier (Ryder [1974]). This one machine-dependent program provides values of the machine precision, the smallest magnitude, and the largest magnitude. Most of the values for these parameters were obtained from the corresponding PORT library program (Fox, Hall, and Schryer [1978]); in particular, values are provided for all of the computing systems on which the programs were tested.

MINPACK-1 is fully supported. Comments, questions, and reports of poor or incorrect performance of the MINPACK-1 programs should be directed to

Burton S. Garbow
Applied Mathematics Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439
Phone: (312) 972-7184

Of particular interest would be reports of performance of the MINPACK-1 package on machines not covered in the testing.

The MINPACK-1 package consists of the programs, their documentation, and the testing aids. The package comprises approximately 28,000 card images and is transmitted on magnetic tape. The tape is available from the following two sources.

National Energy Software Center
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439
Phone: (312) 972-7250

IMSL
Sixth Floor-NBC Building
7500 Bellaire Blvd.
Houston, TX 77036
Phone: (713) 772-1927

The package includes both single and double precision versions of the programs, and for those programs normally called by the user machine-readable documentation is provided in both single and double precision forms. An implementation guide (Garbow, Hillstrom, and Moré [1980]) is also included with the tape.

Acknowledgments

The MINPACK-1 testing was conducted by the following individuals; their assistance and suggestions were invaluable to the project.

Poul Arendal, The World Bank (Burroughs)
Richard Bartels, University of Waterloo (Honeywell)
Mary Ann Berg, University of Illinois at Urbana-Champaign (CDC)
W. Robert Boland, Air Force Weapons Laboratory (CDC)
Roger Crane, RCA Laboratories (IBM)
Dona Crawford, Sandia Laboratories, Livermore (CDC)
John Dennis, Rice University (Itel)
Jeremy DuCroz, Numerical Algorithms Group Ltd. (ICL)
Jay Fleisher, The University of Wisconsin (Univac)
Fred Fritsch, Lawrence Livermore Laboratory (CDC,Cray)
Patrick Gaffney, Union Carbide Corporation (CDC,Cray,DEC,IBM)
David Gay, Massachusetts Institute of Technology (IBM)
Kathie Hiebert, Sandia Laboratories, Albuquerque (CDC)
L. W. Lucas, Naval Weapons Center (Univac)
Dan O'Reilly, Data Resources, Inc. (Burroughs)
Gerald Ruderman, Management Decision Systems, Inc. (Prime)
Nora Sabelli, University of Illinois at Chicago Circle (IBM)
Susan Sherry, Federal Reserve Board (IBM)
Danny Sorensen, University of Kentucky (IBM)
Jesse Wang, Argonne National Laboratory (IBM)

Many others have contributed to the package in various ways; in particular, we acknowledge Beverly Arnoldy, Jim Boyle, Ken Brown, Wayne Cowell, Jim Cody, Tom Coleman, Bill Davidon, Jack Dongarra, Dudley Goetschel, Jerry Kreuser, James Lyness, Mike Minkoff, Larry Nazareth, Mike Powell, Rich Raffenetti, Bob Schnabel, Greg Shubert, Brian Smith, David Thuente, and Richard Wilk.

Special thanks go to Jim Pool, the originator of the project, and to Paul Messina, the head of the Applied Mathematical Sciences section of the Applied Mathematics Division at Argonne. Finally, thanks to Judy Beumer for her usual outstanding job of typing this report.

CHAPTER 1
Introduction to MINPACK-1

The purpose of this chapter is to provide an overview of the algorithms and software in MINPACK-1. Most users need only be acquainted with the first six sections of this chapter; the remaining two sections describe lower-level software called from the main programs.

1.1 Systems of Nonlinear Equations

If n functions $f_1,f_2,\ldots,f_n$ of the n variables $x_1,x_2,\ldots,x_n$ are specified, then MINPACK-1 subroutines can be used to find values for $x_1,x_2,\ldots,x_n$ that solve the system of nonlinear equations

$$f_i(x_1,x_2,\ldots,x_n) = 0 , \qquad 1 \le i \le n .$$

To solve this system we have implemented a modification of Powell's hybrid algorithm. There are two variants of this algorithm. The first variant only requires that the user calculate the functions f_i, while the second variant requires that the user calculate both the functions f_i and the n by n Jacobian matrix

$$\left(\frac{\partial f_i(x)}{\partial x_j}\right) , \qquad 1 \le i \le n , \quad 1 \le j \le n .$$

1.2 Nonlinear Least Squares Problems

If m functions $f_1,f_2,\ldots,f_m$ of the n variables $x_1,x_2,\ldots,x_n$ are specified with $m \ge n$, then MINPACK-1 subroutines can be used to find values for $x_1,x_2,\ldots,x_n$ that solve the nonlinear least squares problem

$$\min\left\{\sum_{i=1}^{m} f_i(x)^2 : x \in R^n\right\} .$$

To solve this problem we have implemented a modification of the Levenberg-Marquardt algorithm. There are three variants of this algorithm. The first variant only requires that the user calculate the functions f_i, while the second variant requires that the user calculate both the functions f_i and the m by n Jacobian matrix

$$\left(\frac{\partial f_i(x)}{\partial x_j}\right), \qquad 1 \le i \le m \ , \ 1 \le j \le n \ .$$

The third variant also requires that the user calculate the functions and the Jacobian matrix, but the latter only one row at a time. This organization only requires the storage of an n by n matrix (rather than m by n), and is thus attractive for nonlinear least squares problems with a large number of functions and a moderate number of variables.

1.3 Derivative Checking

The main advantage of providing the Jacobian matrix is increased reliability; for example, the algorithm is then much less sensitive to functions subject to errors. However, providing the Jacobian matrix is an error-prone task. To help identify errors, MINPACK-1 also contains a subroutine CHKDER that checks the Jacobian matrix for consistency with the function values.

1.4 Algorithmic Paths: Core Subroutines and Easy-to-Use Drivers

There are five general algorithmic paths in MINPACK-1. Each path includes a core subroutine and an easy-to-use driver with a simplified calling sequence made possible by assuming default settings for certain parameters and by returning a limited amount of information; many applications do not require full flexibility and in these cases easy-to-use drivers can be invoked. On the other hand, the core subroutines enable, for example, scaling of the variables and printing of intermediate results at specified iterations.

1.5 MINPACK-1 Subroutines: Systems of Nonlinear Equations

The MINPACK-1 subroutines for the numerical solution of systems of nonlinear equations are HYBRD1, HYBRD, HYBRJ1, and HYBRJ. These subroutines provide alternative ways to solve the system of nonlinear equations

$$f_i(x_1,x_2,\ldots,x_n) = 0 \ , \qquad 1 \le i \le n$$

by a modification of Powell's hybrid algorithm. The principal requirements of the subroutines are as follows (see also Figure 1).

HYBRD1, HYBRD

The user must provide a subroutine to calculate the functions $f_1, f_2, \ldots, f_n$. The Jacobian matrix is then calculated by a forward-difference approximation or by an update formula of Broyden. HYBRD1 is the easy-to-use driver for the core subroutine HYBRD.

HYBRJ1, HYBRJ

The user must provide a subroutine to calculate the functions $f_1, f_2, \ldots, f_n$ and the Jacobian matrix

$$\left(\frac{\partial f_i(x)}{\partial x_j}\right), \qquad 1 \leq i \leq n, \quad 1 \leq j \leq n .$$

(Subroutine CHKDER can be used to check the Jacobian matrix for consistency with the function values.) HYBRJ1 is the easy-to-use driver for the core subroutine HYBRJ.

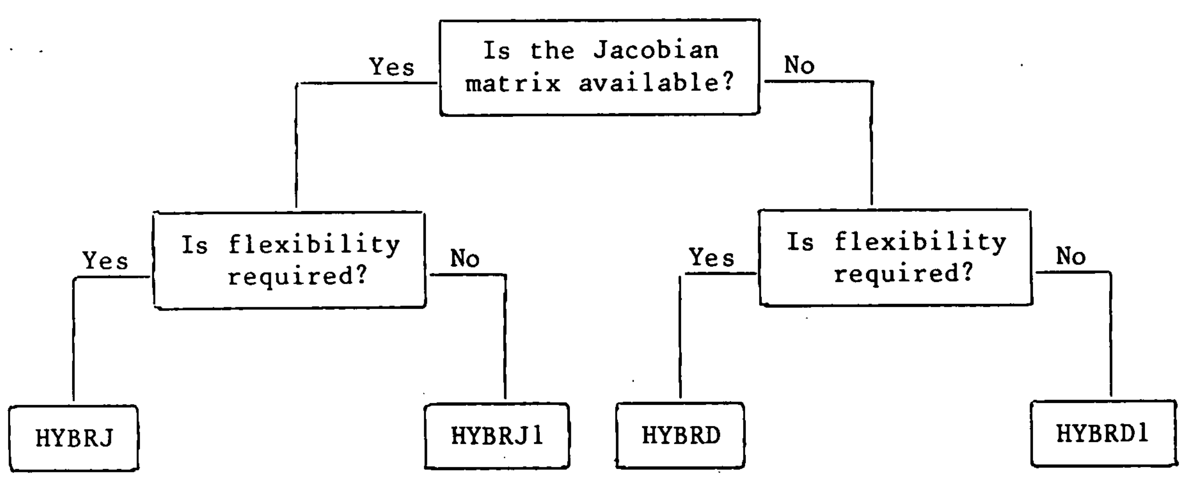

Figure 1
Decision Tree for Systems of Nonlinear Equations

1.6 MINPACK-1 Subroutines: Nonlinear Least Squares Problems

The MINPACK-1 subroutines for the numerical solution of nonlinear least squares problems are LMDIF1, LMDIF, LMDER1, LMDER, LMSTR1, and LMSTR. These subroutines provide alternative ways to solve the nonlinear least squares problem

$$\min\left\{\sum_{i=1}^{m} f_i(x)^2 : x \in R^n\right\}$$

by a modification of the Levenberg-Marquardt algorithm. The principal requirements of the subroutines are as follows (see also Figure 2).

LMDIF1, LMDIF

The user must provide a subroutine to calculate the functions $f_1, f_2, \ldots, f_m$. The Jacobian matrix is then calculated by a forward-difference approximation. LMDIF1 is the easy-to-use driver for the core subroutine LMDIF.

LMDER1, LMDER

The user must provide a subroutine to calculate the functions $f_1, f_2, \ldots, f_m$ and the Jacobian matrix

$$\left(\frac{\partial f_i(x)}{\partial x_j}\right), \qquad 1 \le i \le m, \quad 1 \le j \le n .$$

(Subroutine CHKDER can be used to check the Jacobian matrix for consistency with the function values.) LMDER1 is the easy-to-use driver for the core subroutine LMDER.

LMSTR1, LMSTR

The user must provide a subroutine to calculate the functions $f_1, f_2, \ldots, f_m$ and the rows of the Jacobian matrix

$$\left(\frac{\partial f_i(x)}{\partial x_j}\right), \qquad 1 \le i \le m, \quad i \le j \le n ,$$

one row per call. (Subroutine CHKDER can be used to check the row of the Jacobian matrix for consistency with the corresponding function value.) LMSTR1 is the easy-to-use driver for the core subroutine LMSTR.

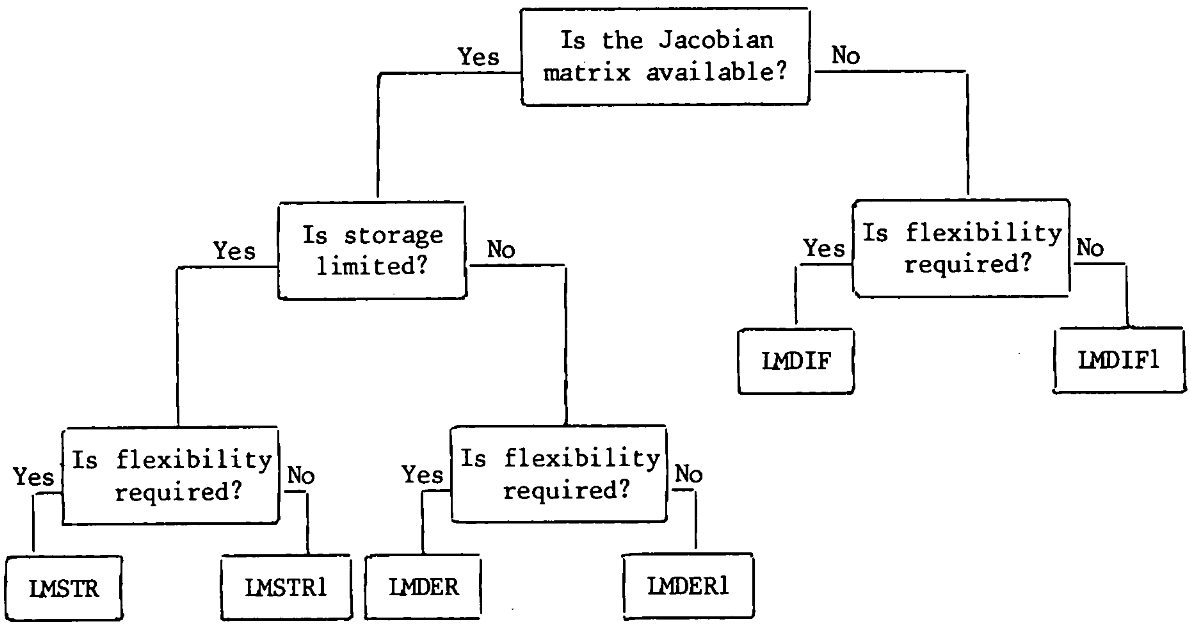

Figure 2
Decision Tree for Nonlinear Least Squares Problems

1.7 Machine-Dependent Constants

There are three machine-dependent constants that have to be set before the single or double precision version of MINPACK-1 can be used; for most machines the correct values of these constants are encoded into DATA statements in functions SPMPAR (single precision) and DPMPAR (double precision). These constants are:

$\beta^{1-\ell}$, the machine precision ,

$\beta^{e_{min}-1}$, the smallest magnitude ,

$(1 - \beta^{-\ell})\beta^{e_{max}}$, the largest magnitude ,

where ℓ is the number of base β digits on the machine, e_{min} is the smallest machine exponent, and e_{max} is the largest machine exponent.

The most critical of the constants is the machine precision ε_M, since the MINPACK-1 subroutines treat two numbers a and b as equal if they satisfy

$$|b-a| \leq \varepsilon_M |a| ,$$

and the above test forms the basis for deciding that no further improvement is possible with the algorithm.

1.8 MINPACK-1 Internal Subprograms

Most users of MINPACK-1 need only be acquainted with the core subroutines and easy-to-use drivers described in the previous sections. Some users, however, may wish to experiment by modifying an algorithmic path to improve the performance of the algorithm on a particular application. A modification to an algorithmic path can often be achieved by modifying or replacing one of the internal subprograms. Additionally, the internal subprograms may be useful independent of the MINPACK-1 algorithmic paths in which they are employed.

For these reasons brief descriptions of the MINPACK-1 internal subprograms are included below; more complete descriptions can be found in the prologue comments in the program listings of Chapter 5.

DOGLEG

Given the QR factorization of an m by n matrix A, an n by n nonsingular diagonal matrix D, an m-vector b, and a positive number Δ, this subroutine determines the convex combination of the Gauss-Newton and scaled gradient directions that solves the problem

$$\min\{\|Ax-b\| \ : \ \|Dx\| \leq \Delta\} \ .$$

ENORM

This function computes the Euclidean norm of a vector x.

FDJAC1

This subroutine computes a forward-difference approximation to the Jacobian matrix associated with n functions in n variables. It includes a banded Jacobian option.

FDJAC2

This subroutine computes a forward-difference approximation to the Jacobian matrix associated with m functions in n variables.

LMPAR

Given the QR factorization of an m by n matrix A, an n by n nonsingular diagonal matrix D, an m-vector b, and a positive number Δ, this subroutine is used to solve the problem

$$\min\{\|Ax-b\| : \|Dx\| \leq \Delta\} .$$

QFORM

Given the QR factorization of a rectangular matrix, this subroutine accumulates the orthogonal matrix Q from its factored form.

QRFAC

This subroutine uses Householder transformations with optional column pivoting to compute a QR factorization of an arbitrary rectangular matrix.

QRSOLV

Given the QR factorization of an m by n matrix A, an n by n diagonal matrix D, and an m-vector b, this subroutine solves the linear least squares problem

$$\binom{A}{D}x \cong \binom{b}{0} .$$

RWUPDT

This subroutine is used in updating the upper triangular part of the QR decomposition of a matrix A after a row is added to A.

R1MPYQ

This subroutine multiplies a matrix by an orthogonal matrix given as a product of Givens rotations.

R1UPDT

This subroutine is used in updating the lower triangular part of the LQ decomposition of a matrix A after a rank-1 matrix is added to A.

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

CHAPTER 2
Algorithmic Details

The purpose of this chapter is to provide information about the algorithms and to point out some of the ways in which this information can be used to improve their performance. The first two sections are essential for the rest of the chapter since they provide the necessary background, but the other sections are independent of each other.

2.1 Mathematical Background

To describe the algorithms for the solution of systems of nonlinear equations and nonlinear least squares problems, it is necessary to introduce some notation.

Let R^n represent the n-dimensional Euclidean space of real n-vectors

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix},$$

and $\|x\|$ the Euclidean norm of x,

$$\|x\| = \left(\sum_{j=1}^{n} x_j^2 \right)^{\frac{1}{2}} .$$

A function F with domain in R^n and range in R^m is denoted by $F: R^n \to R^m$. Such a function can be expressed as

$$F(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{pmatrix},$$

where the component function $f_i: R^n \to R$ is sometimes called the i-th residual of F. The terminology derives from the fact that a common problem is to fit a model g(t,x) to data y, in which case the f_i are of the form

$$f_i(x) = y_i - g(t_i,x) ,$$

where y_i is measured at t_i and x is the set of fit parameters.

In this notation a system of nonlinear equations is specified by a function $F: R^n \to R^n$, and a solution vector x^* in R^n is such that

$$F(x^*) = 0 .$$

Similarly, a nonlinear least squares problem is specified by a function $F: R^n \to R^m$ with $m \geq n$, and a solution vector x^* in R^n is such that

$$\|F(x^*)\| \leq \|F(x)\| \quad \text{for} \quad x \in N(x^*) ,$$

where $N(x^*)$ is a neighborhood of x^*. If $N(x^*)$ is the entire domain of definition of the function, then x^* is a global solution; otherwise, x^* is a local solution.

Some of the MINPACK-1 algorithms require the specification of the Jacobian matrix of the mapping $F: R^n \to R^m$; that is, the m by n matrix $F'(x)$ whose (i,j) entry is

$$\frac{\partial f_i(x)}{\partial x_j} .$$

A related concept is the gradient of a function $f: R^n \to R$, which is the mapping $\nabla f: R^n \to R^n$ defined by

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix} .$$

Note that the i-th row of the Jacobian matrix $F'(x)$ is the gradient $\nabla f_i(x)$ of the i-th residual.

It is well-known that if x^* is a solution of the nonlinear least squares problem, then x^* solves the system of nonlinear equations

$$\sum_{i=1}^{m} f_i(x)\nabla f_i(x) = 0 .$$

In terms of the Jacobian matrix this implies that

$$F'(x^*)^T F(x^*) = 0 ,$$

and shows that at the solution the vector of residuals is orthogonal to the columns of the Jacobian matrix. This orthogonality condition is also satisfied at maximizers and saddle points, but algorithms usually take precautions to avoid these critical points.

2.2 Overview of the Algorithms

Consider a mapping $F\colon R^n \to R^m$, where $m = n$ for systems of nonlinear equations and $m \geq n$ for nonlinear least squares problems. The MINPACK-1 algorithms in these two problem areas seek a solution x^* of the problem

$$\text{(1)} \qquad \min\{\|F(x)\|: x \in R^n\} .$$

In particular, if $m = n$ it is expected that $F(x^*) = 0$.

Our initial description of the algorithms will be at the macroscopic level where the techniques used in each problem area are similar.

With each algorithm the user provides an initial approximation $x = x_0$ to the solution of the problem. The algorithm then determines a correction p to x that produces a sufficient decrease in the residuals of F at the new point $x+p$; it then sets

$$x_+ = x + p$$

and begins a new iteration with x_+ replacing x.

A sufficient decrease in the residuals implies, in particular, that

$$\|F(x+p)\| < \|F(x)\| ,$$

and thus the algorithms guarantee that

$$\|F(x_+)\| < \|F(x)\| .$$

The correction p depends upon a diagonal scaling matrix D, a step bound Δ, and an approximation J to the Jacobian matrix of F at x. Users of the core subroutines can specify initial values D_0 and Δ_0; in the easy-to-use drivers D_0 and Δ_0 are set internally. If the user is providing the Jacobian matrix, then $J_0 = F'(x_0)$; otherwise the algorithm sets J_0 to a forward difference approximation to $F'(x_0)$.

To compute p, the algorithm solves (approximately) the problem

$$\text{(2)} \qquad \min\{\|f+Jp\|: \|Dp\| \leq \Delta\} ,$$

where f is the m-vector of residuals of F at x. If the solution of this problem does not provide a suitable correction, then Δ is decreased and, if appropriate, J is updated. A new problem is now solved, and this process is repeated (usually only once or twice) until a p is obtained at which there is sufficient decrease in the residuals, and then x is replaced by x+p. Before the start of the next iteration, D, Δ, and J are also replaced.

The motivation for using (2) to obtain the correction p is that for appropriate choices of J and Δ, the solution of (2) is an approximate solution of

$$\min\{\|F(x+p)\|: \|Dp\| \leq \Delta\} .$$

It follows that if there is a solution x* such that

$$\text{(3)} \qquad \|D(x-x^*)\| \leq \Delta ,$$

then x+p is close to x*. If this is not the case, then at least x+p is a better approximation to x* than x. Under reasonable conditions, it can be shown that (3) eventually holds.

The algorithms for systems of nonlinear equations and for nonlinear least squares problems differ, for example, in the manner in which the correction p is obtained as an approximate solution of (2). The nonlinear equations algorithm obtains a p that minimizes $\|f+Jp\|$ in a two-dimensional subspace of the ellipsoid $\{p: \|Dp\| \le \Delta\}$. The nonlinear least squares algorithm obtains a p that is the exact solution of (2) with a small (10%) perturbation of Δ. Other differences in the algorithms include convergence criteria (Section 2.3) and the manner in which J is computed (Section 2.4).

It is appropriate to close this overview of the algorithms by discussing two of their limitations. First, the algorithms are limited by the precision of the computations. Although the algorithms are globally convergent under reasonable conditions, the convergence proofs are only valid in exact arithmetic and the algorithms may fail in finite precision due to roundoff. This implies that the algorithms tend to perform better in higher precision. It also implies that the calculation of the function and the Jacobian matrix should be as accurate as possible and that improved performance results when the user can provide the Jacobian analytically.

Second, the algorithms are only designed to find local solutions. To illustrate this point, consider

$$F(x) = x^3 - 3x + 18 .$$

In this case, problem (1) has the global solution $x^* = -3$ with $F(x^*) = 0$ and the local solution $x^* = 1$ with $F(x^*) = 16$; depending on the starting point, the algorithms may converge either to the global solution or to the local solution.

2.3 Convergence Criteria

The convergence test in the MINPACK-1 algorithms for systems of nonlinear equations is based on an estimate of the distance between the current approximation x and an actual solution x^* of the problem. If D is the current scaling matrix, then this convergence test (X-convergence) attempts to guarantee that

$$(1) \qquad \|D(x-x^*)\| \le \text{XTOL}\cdot\|Dx^*\| ,$$

where XTOL is a user-supplied tolerance.

There are three convergence tests in the MINPACK-1 algorithms for nonlinear least squares problems. One test is again for X-convergence, but the main convergence test is based on an estimate of the distance between the Euclidean norm $\|F(x)\|$ of the residuals at the current approximation x and the optimal value $\|F(x^*)\|$ at an actual solution x^* of the problem. This convergence test (F-convergence) attempts to guarantee that

$$(2) \qquad \|F(x)\| \leq (1 + \mathrm{FTOL}) \cdot \|F(x^*)\| ,$$

where FTOL is a second user-supplied tolerance.

The third convergence test for the nonlinear least squares problem (G-convergence) guarantees that

$$(3) \qquad \max\left\{ \frac{|a_i^T f|}{\|a_i\| \|f\|} : 1 \leq i \leq n \right\} \leq \mathrm{GTOL} ,$$

where $a_1, a_2, \ldots, a_n$ are the columns of the current approximation to the Jacobian matrix, f is the vector of residuals, and GTOL is a third user-supplied tolerance.

Note that individual specification of the above three tolerances for the nonlinear least squares problem requires direct user call of the appropriate core subroutine. The easy-to-use driver only accepts the single value TOL. It then internally sets FTOL = XTOL = TOL and GTOL = 0.

The X-convergence condition (1) is a relative error test; it thus fails when $x^* = 0$ unless $x = 0$ also. Also note that if (1) is satisfied with $\mathrm{XTOL} = 10^{-k}$, then the larger components of Dx have k significant digits, but smaller components may not be as accurate. For example, if D is the identity matrix, XTOL = 0.001, and

$$x^* = (2.0,\ 0.003) ,$$

then

$$x = (2.001,\ 0.002)$$

satisfies (1), yet the second component of x has no significant digits. This may or may not be important. However, note that if instead

$$D = \mathrm{diag}(1,1000) ,$$

then (1) is not satisfied even for XTOL = 0.1. These scaling considerations can make it important to choose D carefully. See Section 2.5 for more information on scaling.

Since x* is unknown, the actual criterion for X-convergence cannot be based on (1); instead it depends on the step bound Δ. That is, the actual convergence test is

$$\Delta \le \mathrm{XTOL} \cdot \|Dx\| .$$

The F-convergence condition (2) is a relative error test; it thus fails when $F(x^*) = 0$ unless $F(x) = 0$ also. It is for this reason that F-convergence is not tested for systems of nonlinear equations where $F(x^*) = 0$ is the expected result. Also note that if (2) is satisfied with FTOL = 10^{-k}, then $\|F(x)\|$ has k significant digits, but x may not be as accurate. For example, if FTOL = 10^{-6} and

$$F(x) = \begin{pmatrix} x - 1 \\ 1 \end{pmatrix} ,$$

then $x^* = 1$, $\|F(x^*)\| = 1$, and if $x = 1.001$ then (2) is satisfied with FTOL = 10^{-6}, but (1) is only satisfied with XTOL = 10^{-3}.

In many least squares problems, if FTOL = $(\mathrm{XTOL})^2$ then X-convergence implies F-convergence. This result, however, does not hold if $\|F(x^*)\|$ is very small. For example, if

$$F(x) = \begin{pmatrix} x - 1 \\ 0.0001 \end{pmatrix} ,$$

then $x^* = 1$ and $\|F(x^*)\| = 0.0001$, but if $x = 1.001$ then (1) is satisfied with XTOL = 10^{-3} and yet

$$\|F(x)\| \ge 10\|F(x^*)\| .$$

Since $\|F(x^*)\|$ is unknown, the actual criterion for F-convergence cannot be literally (2); instead it is based on estimates of the terms in (2). If f and f_+ are the vectors of residuals at the current solution approximation x and at x+p, respectively, then the (relative) actual reduction is

$$\text{ACTRED} = \left(\|f\| - \|f_+\|\right)/\|f\| ,$$

while the (relative) predicted reduction is

$$\text{PRERED} = \left(\|f\| - \|f+Jp\|\right)/\|f\| .$$

The F-convergence test then requires that

$$\text{PRERED} \le \text{FTOL}$$
$$|\text{ACTRED}| \le \text{FTOL}$$
$$\text{ACTRED} \le 2\cdot\text{PRERED}$$

all hold.

The X-convergence and F-convergence tests are quite reliable, but it is important to note that their validity depends critically on the correctness of the Jacobian. If the user is providing the Jacobian, he may make an error. (CHKDER can be used to check the Jacobian.) If the algorithm is estimating the Jacobian matrix, then the approximation may be incorrect if, for example, the function is subject to large errors and EPSFCN is chosen poorly. (For more details see Section 2.4.) In either case the algorithm usually terminates suspiciously near the starting point; recommended action if this occurs is to rerun the problem from a different starting point. If the algorithm also terminates near the new starting point, then it is very likely that the Jacobian is being determined incorrectly.

The X-convergence and F-convergence tests may also fail if the tolerances are too large. In general, XTOL and FTOL should be smaller than 10^{-5}; recommended values for these tolerances are on the order of the square root of the machine precision. As described in Section 1.7, the single precision value of the machine precision can be obtained from the MINPACK-1 function SPMPAR and the double precision value from DPMPAR. Note, however, that on some machines the square root of machine precision is larger than 10^{-5}.

The G-convergence test (3) measures the angle between the residual vector and the columns of the Jacobian matrix and thus can be expected to fail if either $F(x^*) = 0$ or any column of $F'(x^*)$ is zero. Also note that there is no clear relationship between G-convergence and either X-convergence or F-convergence. Furthermore, the G-convergence test detects other critical points, namely maximizers and saddle points; therefore, termination with G-convergence should be examined carefully.

An important property of the tests described above is that they are scale invariant. (See Section 2.5 for more details on scaling.) Scale invariance is a feature not shared by many other convergence tests. For example, the convergence test

(4) $$\|f\| \leq \text{AFTOL} ,$$

where AFTOL is a user-supplied tolerance, is not scale invariant, and this makes it difficult to choose an appropriate AFTOL. As an illustration of the difficulty with this test, consider the function

$$F(x) = (3x - 10)\exp(10x) .$$

On a computer with 15 decimal digits

$$|F(x^*)| \geq 1 ,$$

where x^* is the closest machine-representable number to 10/3, and thus a suitable AFTOL is not apparent.

If the user, however, wants to use (4) as a termination test, then he can do this by setting NPRINT positive in the call to the respective core subroutine. (See Section 2.9 for more information on NPRINT.) This provides him periodic opportunity, through subroutine FCN with IFLAG = 0, to affect the iteration sequence, and in this instance he might insert the following program segment into FCN.

```
      IF (IFLAG .NE. 0) GO TO 10
      FNORM = ENORM(LFVEC,FVEC)
      IF (FNORM .LE. AFTOL) IFLAG = -1
      RETURN
   10 CONTINUE
```

In this program segment it is assumed that LFVEC = N for systems of nonlinear equations and LFVEC = M for nonlinear least squares problems. It is also assumed that the MINPACK-1 function ENORM is declared to the precision of the computation.

2.4 Approximations to the Jacobian Matrix

If the user does not provide the Jacobian matrix, then the MINPACK-1 algorithms compute an approximation J. In the algorithms for nonlinear least squares problems, J is always determined by a forward difference approximation, while in the algorithms for systems of nonlinear equations, J is sometimes determined by a forward-difference approximation but more often by an update formula of Broyden. It is important to note that the update formula is also used in the algorithms for systems of nonlinear equations where the user is providing the Jacobian matrix, since the updating tends to improve the efficiency of the algorithms.

The forward-difference approximation to the j-th column of the Jacobian matrix can be written

$$\text{(1)} \qquad \frac{F(x+h_j e_j) - F(x)}{h_j} ,$$

where e_j is the j-th column of the identity matrix and h_j is the difference parameter. The choice of h_j depends on the precision of the function evaluations, which is specified in the MINPACK-1 algorithms by the parameter EPSFCN. To be specific,

$$h_j = (\text{EPSFCN})^{\frac{1}{2}}|x_j|$$

unless $x_j = 0$, in which case

$$h_j = (\mathrm{EPSFCN})^{1/2} .$$

In the easy-to-use drivers EPSFCN is set internally to the machine precision (see Section 1.7), since these subroutines assume that the functions can be evaluated accurately. In the core subroutines EPSFCN is a user-supplied parameter; if there are errors in the evaluations of the functions, then EPSFCN may need to be much larger than the machine precision. For example, if the specification of the function requires the numerical evaluation of an integral, then EPSFCN should probably be on the order of the tolerance in the integration routine.

One advantage of approximation (1) is that it is scale invariant. (See Section 2.5 for more details on scaling.) A disadvantage of (1) is that it assumes EPSFCN the same for each variable, for each component function of F, and for each vector x. These assumptions may make it difficult to determine a suitable value for EPSFCN. The user who is uncertain of an appropriate value of EPSFCN can run the algorithm with two or three values of EPSFCN and retain the value that gives the best results. In general, overestimates are better than underestimates.

The update formula of Broyden depends on the current approximation x, the correction p, and J. Since

$$F(x+p) - F(x) = \left[\int_0^1 F'(x+\theta p)\,d\theta \right] p ,$$

it is natural to ask that the approximation J_+ at x+p satisfy the equation

$$J_+ p = F(x+p) - F(x) ,$$

and among the possible choices be the one closest to J. To define an appropriate measure of distance, let D be the current diagonal scaling matrix and define the matrix norm

$$\|A\|_D = \left(\sum_{j=1}^{n} \left(\frac{\|a_j\|}{d_j} \right)^2 \right)^{1/2} ,$$

where $a_1, a_2, \ldots, a_n$ are the columns of A. It is now easy to verify that the solution of the problem

$$\min\{\|\tilde{J}-J\|_D : \tilde{J}p = F(x+p)-F(x)\} ,$$

is given by

$$J_+ = J + \frac{\big(F(x+p)-F(x)-Jp\big)(D^TDp)^T}{\|Dp\|^2} .$$

There are many properties of this formula that justify its use in algorithms for systems of nonlinear equations, but a discussion of these properties is beyond the scope of this work.

2.5 Scaling

Scale invariance is a desirable feature of an optimization algorithm. Algorithms for systems of nonlinear equations and nonlinear least squares problems are scale invariant if, given problems related by the change of scale

$$\begin{aligned} \tilde{F}(x) &= \alpha F(D_V x) \\ \tilde{x}_o &= D_V^{-1} x_o , \end{aligned}$$

where α is a positive scalar and D_V is a diagonal matrix with positive entries, the approximations x and $\tilde{x}$ generated by the algorithms satisfy

$$\tilde{x} = D_V^{-1} x .$$

Scale invariance is a natural requirement that can have a significant effect on the implementation and performance of an algorithm. To the user scale invariance means, in particular, that he can work with either problem and obtain equivalent results.

The core subroutines in MINPACK-1 are scale invariant provided that the initial choice of the scaling matrix satisfies

(1) $$\tilde{D}_o = \alpha D_V D_o ,$$

where D_o and $\tilde{D}_o$ are the initial scaling matrices of the respective problems defined by F and x_o and by $\tilde{F}$ and $\tilde{x}_o$. If the user of the core subroutines has requested internal scaling (MODE = 1), then the internal scaling matrix is set to

$$\mathrm{diag}(\|a_1\|,\|a_2\|,\ldots,\|a_n\|) \ ,$$

where a_i is the i-th column of the initial Jacobian approximation, and (1) holds. If the user has stipulated external scaling (MODE = 2), then the initial scaling matrix is specified by the contents of the array DIAG, and scale invariance is only achieved if the user's choice satisfies (1).

There are certain cases in which scale invariance may be lost, as when the Jacobian matrix at the starting point has a column of zeroes and internal scaling is requested. In this case D_o would have a zero element and be singular, but this possibility is not catered to in the current implementation. Instead, the zero element is arbitrarily set to 1, preserving nonsingularity but giving up scale invariance. In practice, however, these cases seldom arise and scale invariance is usually maintained.

Our experience is that internal scaling is generally preferable for nonlinear least squares problems and external scaling for systems of nonlinear equations. This experience is reflected in the settings built into the easy-to-use drivers; MODE = 1 is specified in the drivers for nonlinear least squares problems and MODE = 2 for systems of nonlinear equations. In the latter case, D_o is set to the identity matrix, a choice that generally works out well in practice; if this choice is not appropriate, recourse to the core subroutine would be indicated.

It is important to note that scale invariance does not relieve the user of choosing an appropriate formulation of the problem or a reasonable starting point. In particular, note that an appropriate formulation may involve a scaling of the equations or a nonlinear transformation of the variables and that the performance of the MINPACK-1 algorithms can be affected by these transformations. For example, the algorithm for systems of nonlinear equations usually generates different approximations for problems defined by functions $\tilde{F}$ and F, where

$$\tilde{F}(x) = D_E F(x) \ ,$$
$$\tilde{x}_o = x_o \ ,$$

and D_E is a diagonal matrix with positive entries. The main reason for this is that the algorithm usually decides that x_+ is a better approximation than x if

$$\|F(x_+)\| < \|F(x)\| \ ,$$

and it is entirely possible that

$$\|\tilde{F}(x_+)\| > \|\tilde{F}(x)\| \ .$$

The user should thus scale his equations (i.e., choose D_E) so that the expected errors in the residuals are of about the same order of magnitude.

2.6 Subroutine FCN: Calculation of the Function and Jacobian Matrix

The MINPACK-1 algorithms require that the user provide a subroutine with name of his choosing, say FCN, to calculate the residuals of the function $F: R^n \to R^m$, where $m = n$ for systems of nonlinear equations and $m \geq n$ for nonlinear least squares problems. Some of the algorithms also require that FCN calculate the Jacobian matrix of the mapping F.

It is important that the calculation of the function and Jacobian matrix be as accurate as possible. It is also important that the coding of FCN be as efficient as possible, since the timing of the algorithm is strongly influenced by the time spent in FCN. In particular, when the residuals f_i have common subexpressions it is usually worthwhile to organize the computation so that these subexpressions need be evaluated only once. For example, if the residuals are of the form

$$f_i(x) = g(x) + h_i(x) \ , \quad 1 \leq i \leq m$$

with g(x) common to all of them, then the coding of FCN is best expressed in the following form.

$$\begin{array}{l} \tau = g(x) \\ \text{For } i = 1,2,\ldots,m \\ \quad f_i(x) = \tau + h_i(x) \ . \end{array}$$

As another example, assume that the residuals are of the form

$$f_i(x) = \sum_{j=1}^{n} \left(\alpha_{ij}\cos(x_j) + \beta_{ij}\sin(x_j)\right) ,$$

where the α_{ij} and β_{ij} are given constants. The following program segment evaluates the f_i efficiently.

For $i = 1,2,\ldots,m$
    $f_i(x) = 0$
For $j = 1,2,\ldots,n$
    $\gamma = \cos(x_j)$
    $\sigma = \sin(x_j)$
    For $i = 1,2,\ldots,m$
        $f_i(x) = f_i(x) + \gamma\alpha_{ij} + \sigma\beta_{ij}$.

If the user is providing the Jacobian matrix of the mapping F, then it is important that its calculation also be as efficient as possible. In particular, when the elements of the Jacobian matrix have common subexpressions, it is usually worthwhile to organize the computation so that these subexpressions need be evaluated only once. For example, if

$$f_i(x) = g(x) + h_i(x) , \qquad 1 \le i \le m ,$$

then the rows of the Jacobian matrix are

$$\nabla f_i(x) = \nabla g(x) + \nabla h_i(x) , \qquad 1 \le i \le m ,$$

and the subexpression $\nabla g(x)$ is thus common to all the rows of the Jacobian matrix.

As another example, assume that

$$f_i(x) = \sum_{j=1}^{n} \left(\alpha_{ij}\cos(x_j) + \beta_{ij}\sin(x_j)\right) ,$$

where the α_{ij} and β_{ij} are given constants. In this case,

$$\frac{\partial f_i(x)}{\partial x_j} = - \alpha_{ij}\sin(x_j) + \beta_{ij}\cos(x_j) ,$$

and the following program segment evaluates the Jacobian matrix efficiently.

For $j = 1,2,\ldots,n$

$\gamma = \cos(x_j)$

$\sigma = \sin(x_j)$

For $i = 1,2,\ldots,m$

$$\frac{\partial f_i(x)}{\partial x_j} = -\sigma\alpha_{ij} + \gamma\beta_{ij} .$$

The previous example illustrates further the possibility of common subexpressions between the function and the Jacobian matrix. For the nonlinear least squares algorithms advantage can be taken of this, because a call to FCN to evaluate the Jacobian matrix at x is always preceded by a call to evaluate the function at x. This is not the case for the nonlinear equations algorithms.

To specifically illustrate this possibility of sharing information between function and Jacobian matrix, assume that

$$f_i(x) = g(x)^2 + h_i(x) , \qquad 1 \le i \le m .$$

Then the rows of the Jacobian matrix are

$$\nabla f_i(x) = 2g(x)\nabla g(x) + \nabla h_i(x) , \qquad 1 \le i \le m ,$$

and the coding of FCN is best done as follows.

If FUNCTION EVALUATION then

$\tau = g(x)$

Save τ in COMMON

For $i = 1,2,\ldots,m$

$f_i(x) = \tau^2 + h_i(x)$

If JACOBIAN EVALUATION then

$v = \nabla g(x)$

For $i = 1,2,\ldots,m$

$\nabla f_i(x) = 2\tau v + \nabla h_i(x)$.

2.7 Constraints

Systems of nonlinear equations and nonlinear least squares problems often impose constraints on the solution. For example, on physical grounds it is sometimes necessary that the solution vector have positive components.

At present there are no algorithms in MINPACK that formally admit constraints, but in some cases they can be effectively achieved with ad hoc strategies. In this section we describe two strategies for restricting the solution approximations to a region **D** of R^n.

The user has control over the initial approximation x_0. It may happen, however, that x is in **D** but the algorithm computes a correction p such that x+p is not in **D**. If this correction is permitted, the algorithm may never recover; that is, the approximations may now converge to an unacceptable solution outside of **D**.

The simplest strategy to restrict the corrections is to impose a penalty on the function if the algorithm attempts to step outside of **D**. For example, let μ be any number such that

$$|f_i(x_0)| \leq \mu \ , \qquad 1 \leq i \leq m \ ,$$

and in FCN define

$$f_i(x) = \mu \ , \qquad 1 \leq i \leq m$$

whenever x does not belong to **D**. If FCN is coded in this way, a correction p for which x+p lies outside of **D** will not decrease the residuals and is therefore not acceptable. It follows that this penalty on FCN forces all the approximations x to lie in **D**.

Note that this strategy restricts all the corrections, and as a consequence may lead to very slow convergence if the solution is near the boundary of **D**. It usually suffices to only restrict the initial correction, and users of the core subroutines can do this in several ways.

Recall from Section 2.2 that the initial correction p_0 satisfies a bound of the form

$$\|D_o p_o\| \leq \Delta_o ,$$

where D_o is a diagonal scaling matrix and Δ_o is a step bound. The contents of D_o are governed by the parameter MODE. If MODE = 1 then D_o is internally set, while if MODE = 2 then D_o is specified by the user through the array DIAG. The step bound Δ_o is determined from the parameter FACTOR. By definition

$$\Delta_o = \text{FACTOR} \cdot \|D_o x_o\| ,$$

unless x_o is the zero vector, in which case

$$\Delta_o = \text{FACTOR} .$$

It is clear from this definition that smaller values of FACTOR lead to smaller steps. For a sufficiently small value of FACTOR (usually 0.01 suffices), an improved point x_o+p_o will be found that belongs to **D**.

Be aware that the step restriction is on $D_o p_o$ and not on p_o directly. A small element of D_o, which can be set by internal scaling when MODE = 1, may lead to a large component in the correction p_o. In many cases it is not necessary to control p_o directly, but if this is desired then MODE = 2 must be used.

When MODE = 2, the contents of D_o are specified by the user, and this allows direct control of p_o. If, for example, it is desired to restrict the components of p_o to small relative corrections of the corresponding components of x_o (assumed nonzero), then this can be done by setting

$$D_o = \text{diag}\left(\frac{1}{|\xi_1|}, \frac{1}{|\xi_2|}, \ldots, \frac{1}{|\xi_n|}\right) ,$$

where ξ_i is the i-th component of x_o, and by choosing FACTOR appropriately. To justify this choice, note that p_o satisfies

$$\|D_o p_o\| \leq \Delta_o = \text{FACTOR} \cdot \|D_o x_o\| ,$$

and that the choice of D_o guarantees that

$$\|D_o x_o\| = n^{1/2} .$$

Thus, if ρ_i is the i-th component of p_o, then

$$|\rho_i| \leq n^{1/2}\cdot \mathrm{FACTOR}\cdot |\xi_i| ,$$

which justifies the choice of D_o.

2.8 Error Bounds

A problem of general interest is the determination of error bounds on the components of a solution vector. It is beyond the scope of this work to discuss this topic in depth, so the discussion below is limited to the computation of bounds on the sensitivity of the parameters, and of the covariance matrix. The discussion is in terms of the nonlinear least squares problem, but some of the results also apply to systems of nonlinear equations.

Let $F\colon R^n \to R^m$ define a nonlinear least squares problem $(m \geq n)$, and let x^* be a solution. Given $\varepsilon > 0$, the problem is to determine sensitivity (upper) bounds $\sigma_1, \sigma_2, \ldots, \sigma_n$ such that, for each i, the condition

$$|x_i - x_i^*| \leq \sigma_i , \qquad \text{with } x_j = x_j^* \text{ for } j \neq i ,$$

implies that

$$\|F(x)\| \leq (1 + \varepsilon)\|F(x^*)\| .$$

Of particular interest are values of σ_i which are large relative to $|x_i|$, since then the residual norm $\|F(x)\|$ is insensitive to changes in the i-th parameter and may therefore indicate a possible deficiency in the formulation of the problem.

A first order estimate of the sensitivity bounds σ_i shows that

$$\sigma_i = \varepsilon^{1/2}\left(\frac{\|F(x^*)\|}{\|F'(x^*)\cdot e_i\|}\right) , \tag{1}$$

where $F'(x^*)$ is the Jacobian matrix of F at x^* and e_i is the i-th column of the identity matrix. Note that if $\|F'(x^*)\cdot e_i\|$ is small relative to $\|F(x^*)\|$, then the residual norm is insensitive to changes in the i-th parameter.

If x is an approximation to the solution x* and J is an approximation to F'(x*), then the bounds (1) can usually be replaced by

(2) $$\sigma_i = \varepsilon^{\frac{1}{2}} \left(\frac{\| F(x) \|}{\| Je_i \|} \right) .$$

The MINPACK-1 nonlinear least squares programs (except LMDIF1) return enough information to compute the sensitivity bounds (2). On a normal exit, these programs return F(x) and part of the QR decomposition of J; namely, an upper triangular matrix R and a permutation matrix P such that

(3) $$JP = QR$$

for some matrix Q with orthogonal columns. The vector F(x) is returned in the array FVEC and the matrix R is returned in the upper triangular part of the array FJAC. The permutation matrix P is defined by the contents of the integer array IPVT; if

$$IPVT = (p(1), p(2), \ldots, p(n)) ,$$

then the j-th column of P is the p(j)-th column of the identity matrix.

The norms of the columns of the Jacobian matrix can be computed by noting that (3) implies that

$$Je_{p(j)} = QRe_j ,$$

and hence,

$$\| Je_{p(j)} \| = \| Re_j \| .$$

The following loop uses this relationship to store $\| Je_\ell \|$ in the ℓ-th position of an array FJNORM; with this information it is then easy to compute the sensitivity bounds (2).

```
      DO 10 J = 1, N
         L = IPVT(J)
         FJNORM(L) = ENORM(J,FJAC(1,J))
10       CONTINUE
```

This loop assumes that ENORM and FJNORM have been declared to the precision of the computation.

In addition to sensitivity bounds for the individual parameters, it is sometimes desirable to determine a bound for the sensitivity of the residual norm to changes in some linear combination of the parameters. Given $\varepsilon > 0$ and a vector v with $\|v\| = 1$, the problem is to determine a bound σ such that

$$\|F(x^*+\sigma v)\| \leq (1 + \varepsilon)\|F(x^*)\| .$$

A first order estimate of σ is now

$$\sigma = \varepsilon^{\frac{1}{2}}\left(\frac{\|F(x^*)\|}{\|F'(x^*)\cdot v\|}\right) ;$$

if $\|F'(x^*)\cdot v\|$ is small relative to $\|F(x^*)\|$, then σ is large and the residual norm is insensitive to changes in the linear combination of the parameters specified by v.

For example, if the level set

$$\{x: \ \|F(x)\| \leq (1 + \varepsilon)\|F(x^*)\|\}$$

is as in Figure 3, then the residual norm, although sensitive to changes in x_1 and x_2, is relatively insensitive to changes along $v = (1,1)$.

If the residual norm is relatively insensitive to changes in some linear combination of the parameters, then the Jacobian matrix at the solution is nearly rank-deficient, and in these cases it may be worthwhile to attempt to determine a set of linearly independent parameters. In some statistical applications, the covariance matrix

$$(J^TJ)^{-1}$$

is used for this purpose.

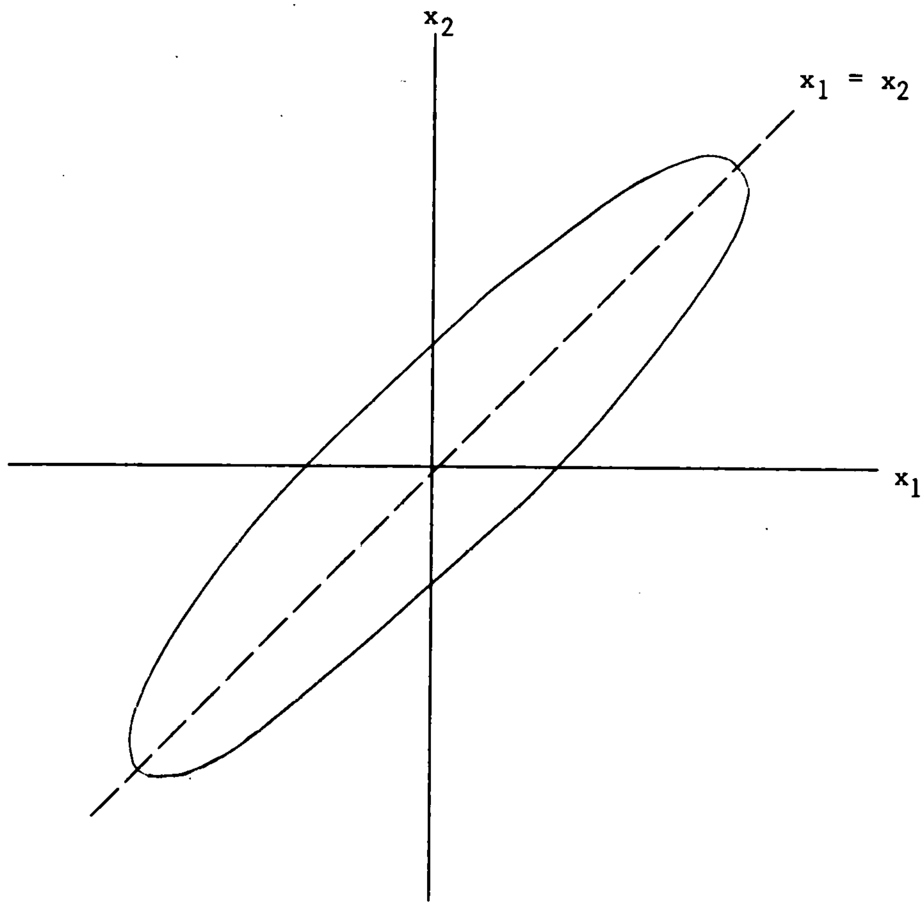

Figure 3

Subroutine COVAR, which appears at the end of this section, will compute the covariance matrix. The computation of the covariance matrix from the QR factorization of J depends on the relationship

$$(J^TJ)^{-1} = P(R^TR)^{-1}P^T \quad , \tag{4}$$

which is an easy consequence of (3). Subroutine COVAR overwrites R with the upper triangular part of $(R^TR)^{-1}$ and then computes the covariance matrix from (4).

Note that for proper execution of COVAR the QR factorization of J must have used column pivoting. This guarantees that for the resulting R

$$|r_{kk}| \ge |r_{ij}| \quad , \qquad k \le i \le j \quad , \tag{5}$$

thereby allowing a reasonable determination of the numerical rank of J. Most of the MINPACK-1 nonlinear least squares subroutines return the correct factorization; the QR factorization in LMSTR1 and LMSTR, however, satisfies

$$JP_1 = Q_1R_1$$

but R_1 does not usually satisfy (5). To obtain the correct factorization, note that the QR factorization with column pivoting of R_1 satisfies

$$R_1P_2 = Q_2R_2$$

where R_2 satisfies (5), and therefore

$$J(P_1P_2) = (Q_1Q_2)R_2$$

is the desired factorization of J. The program segment below uses the MINPACK-1 subroutine QRFAC to compute R_2 from R_1.

```
      DO 30 J = 1, N
         JP1 = J + 1
         IF (N .LT. JP1) GO TO 20
         DO 10 I = JP1, N
            FJAC(I,J) = ZERO
   10       CONTINUE
   20    CONTINUE
   30    CONTINUE
      CALL QRFAC(N,N,FJAC,LDFJAC,.TRUE.,IPVT2,N,WA1,WA2,WA3)
      DO 40 J = 1, N
         FJAC(J,J) = WA1(J)
         L = IPVT2(J)
         IPVT2(J) = IPVT1(L)
   40    CONTINUE
```

Note that QRFAC sets the contents of the array IPVT2 to define the permutation matrix P_2, and the final loop in the program segment overwrites IPVT2 to define the permutation matrix P_1P_2.

```
      SUBROUTINE COVAR(N,R,LDR,IPVT,TOL,WA)
      INTEGER N,LDR
      INTEGER IPVT(N)
      DOUBLE PRECISION TOL
      DOUBLE PRECISION R(LDR,N),WA(N)
C     **********
C
C     SUBROUTINE COVAR
C
C     GIVEN AN M BY N MATRIX A, THE PROBLEM IS TO DETERMINE
C     THE COVARIANCE MATRIX CORRESPONDING TO A, DEFINED AS
C
C                    T
C           INVERSE(A *A) .
C
C     THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM
C     IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE
C     QR FACTORIZATION, WITH COLUMN PIVOTING, OF A. THAT IS, IF
C     A*P = Q*R, WHERE P IS A PERMUTATION MATRIX, Q HAS ORTHOGONAL
C     COLUMNS, AND R IS AN UPPER TRIANGULAR MATRIX WITH DIAGONAL
C     ELEMENTS OF NONINCREASING MAGNITUDE, THEN COVAR EXPECTS
C     THE FULL UPPER TRIANGLE OF R AND THE PERMUTATION MATRIX P.
C     THE COVARIANCE MATRIX IS THEN COMPUTED AS
C
C                      T     T
C           P*INVERSE(R *R)*P  .
C
C     IF A IS NEARLY RANK DEFICIENT, IT MAY BE DESIRABLE TO COMPUTE
C     THE COVARIANCE MATRIX CORRESPONDING TO THE LINEARLY INDEPENDENT
C     COLUMNS OF A. TO DEFINE THE NUMERICAL RANK OF A, COVAR USES
C     THE TOLERANCE TOL. IF L IS THE LARGEST INTEGER SUCH THAT
C
C           ABS(R(L,L)) .GT. TOL*ABS(R(1,1)) ,
C
C     THEN COVAR COMPUTES THE COVARIANCE MATRIX CORRESPONDING TO
C     THE FIRST L COLUMNS OF R. FOR K GREATER THAN L, COLUMN
C     AND ROW IPVT(K) OF THE COVARIANCE MATRIX ARE SET TO ZERO.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE COVAR(N,R,LDR,IPVT,TOL,WA)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R.
C
C       R IS AN N BY N ARRAY. ON INPUT THE FULL UPPER TRIANGLE MUST
C         CONTAIN THE FULL UPPER TRIANGLE OF THE MATRIX R. ON OUTPUT
C         R CONTAINS THE SQUARE SYMMETRIC COVARIANCE MATRIX.
C
C       LDR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY R.
C
C       IPVT IS AN INTEGER INPUT ARRAY OF LENGTH N WHICH DEFINES THE
```

```
C        PERMUTATION MATRIX P SUCH THAT A*P = Q*R. COLUMN J OF P        COVR0550
C        IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.                      COVR0560
C                                                                       COVR0570
C      TOL IS A NONNEGATIVE INPUT VARIABLE USED TO DEFINE THE           COVR0580
C        NUMERICAL RANK OF A IN THE MANNER DESCRIBED ABOVE.             COVR0590
C                                                                       COVR0600
C      WA IS A WORK ARRAY OF LENGTH N.                                  COVR0610
C                                                                       COVR0620
C    SUBPROGRAMS CALLED                                                 COVR0630
C                                                                       COVR0640
C      FORTRAN-SUPPLIED ... DABS                                        COVR0650
C                                                                       COVR0660
C    ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. AUGUST 1980.         COVR0670
C    BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE              COVR0680
C                                                                       COVR0690
C    **********                                                         COVR0700
     INTEGER I,II,J,JJ,K,KM1,L                                          COVR0710
     LOGICAL SING                                                       COVR0720
     DOUBLE PRECISION ONE,TEMP,TOLR,ZERO                                COVR0730
     DATA ONE,ZERO /1.0D0,0.0D0/                                        COVR0740
C                                                                       COVR0750
C    FORM THE INVERSE OF R IN THE FULL UPPER TRIANGLE OF R.             COVR0760
C                                                                       COVR0770
     TOLR = TOL*DABS(R(1,1))                                            COVR0780
     L = 0                                                              COVR0790
     DO 40 K = 1, N                                                     COVR0800
        IF (DABS(R(K,K)) .LE. TOLR) GO TO 50                            COVR0810
        R(K,K) = ONE/R(K,K)                                             COVR0820
        KM1 = K - 1                                                     COVR0830
        IF (KM1 .LT. 1) GO TO 30                                        COVR0840
        DO 20 J = 1, KM1                                                COVR0850
           TEMP = R(K,K)*R(J,K)                                         COVR0860
           R(J,K) = ZERO                                                COVR0870
           DO 10 I = 1, J                                               COVR0880
              R(I,K) = R(I,K) - TEMP*R(I,J)                             COVR0890
   10         CONTINUE                                                  COVR0900
   20      CONTINUE                                                     COVR0910
   30   CONTINUE                                                        COVR0920
        L = K                                                           COVR0930
   40   CONTINUE                                                        COVR0940
   50 CONTINUE                                                          COVR0950
C                                                                       COVR0960
C    FORM THE FULL UPPER TRIANGLE OF THE INVERSE OF (R TRANSPOSE)*R     COVR0970
C    IN THE FULL UPPER TRIANGLE OF R.                                   COVR0980
C                                                                       COVR0990
     IF (L .LT. 1) GO TO 110                                            COVR1000
     DO 100 K = 1, L                                                    COVR1010
        KM1 = K - 1                                                     COVR1020
        IF (KM1 .LT. 1) GO TO 80                                        COVR1030
        DO 70 J = 1, KM1                                                COVR1040
           TEMP = R(J,K)                                                COVR1050
           DO 60 I = 1, J                                               COVR1060
              R(I,J) = R(I,J) + TEMP*R(I,K)                             COVR1070
   60         CONTINUE                                                  COVR1080
```

```
   70       CONTINUE
   80    CONTINUE
         TEMP = R(K,K)
         DO 90 I = 1, K
            R(I,K) = TEMP*R(I,K)
   90       CONTINUE
  100    CONTINUE
  110 CONTINUE
C
C     FORM THE FULL LOWER TRIANGLE OF THE COVARIANCE MATRIX
C     IN THE STRICT LOWER TRIANGLE OF R AND IN WA.
C
      DO 130 J = 1, N
         JJ = IPVT(J)
         SING = J .GT. L
         DO 120 I = 1, J
            IF (SING) R(I,J) = ZERO
            II = IPVT(I)
            IF (II .GT. JJ) R(II,JJ) = R(I,J)
            IF (II .LT. JJ) R(JJ,II) = R(I,J)
  120       CONTINUE
         WA(JJ) = R(J,J)
  130    CONTINUE
C
C     SYMMETRIZE THE COVARIANCE MATRIX IN R.
C
      DO 150 J = 1, N
         DO 140 I = 1, J
            R(I,J) = R(J,I)
  140       CONTINUE
         R(J,J) = WA(J)
  150    CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE COVAR.
C
      END
```

2.9 Printing

No printing is done in any of the MINPACK-1 subroutines. However, printing of certain parameters through FCN can be facilitated with the integer parameter NPRINT that is available to users of the core subroutines. For these subroutines, setting NPRINT positive results in special calls to FCN with IFLAG = 0 at the beginning of the first iteration and every NPRINT iterations thereafter and immediately prior to return. On these calls to FCN, the parameters X and FVEC are available for printing; FJAC is additionally available if using LMDER.

Often it suffices to print some simple measure of the iteration progress, and the Euclidean norm of the residuals is usually a good choice. This norm can be printed by inserting the following program segment into FCN.

```
      IF (IFLAG .NE. 0) GO TO 10
      FNORM = ENORM(LFVEC,FVEC)
      WRITE (---,1000) FNORM
 1000 FORMAT (---)
      RETURN
   10 CONTINUE
```

In this program segment it is assumed that LFVEC = N for systems of nonlinear equations and LFVEC = M for nonlinear least squares problems. It is also assumed that the MINPACK-1 function ENORM is declared to the precision of the computation.

CHAPTER 3
Notes and References

This chapter provides notes relating the MINPACK-1 algorithms and software to other work. The list of references appears at the end.

Powell's Hybrid Method

The MINPACK-1 version of Powell's [1970] hybrid method differs in many respects from the original version. For example, the "special iterations" used in the original algorithm proved to be inefficient and have been replaced. The updating method used is due to Broyden [1965]; the MINPACK-1 algorithm is a scaled version of the original. A comparison of an earlier version of the MINPACK-1 algorithm with other algorithms for systems of nonlinear equations has been made by Hiebert [1980].

The Levenberg-Marquardt Algorithm

There are many versions of the algorithm proposed by Levenberg [1944] and modified by Marquardt [1963]. An advantage of the MINPACK-1 version is that it avoids the difficulties associated with choosing the Levenberg-Marquardt parameter, and this allows a very strong global convergence result. The MINPACK-1 algorithm is based on the work of Hebden [1973] and follows the ideas of Moré [1977]. A comparison of an earlier version of the MINPACK-1 algorithm with other algorithms for nonlinear least squares problems has been made by Hiebert [1979].

Derivative Checking

Subroutine CHKDER is new, but similar routines exist in the Numerical Algorithms Group (NAG) library. An advantage of CHKDER is its generality; it can be used to check Jacobians, gradients, and Hessians (second derivatives). To enable this generality, CHKDER presumes no specific parameter sequence for the function evaluation program, returning control instead to the user. This in turn makes necessary a second call to CHKDER for each check.

MINPACK-1 Internal Subprograms

Subroutines DOGLEG and LMPAR are used to generate search directions in the algorithms for systems of nonlinear equations and nonlinear least squares problems, respectively. The algorithm used in DOGLEG is a fairly straightforward implementation of the ideas of Powell [1970], while LMPAR is a refined version of the algorithm described by Moré [1977]. The LMPAR algorithm is the more complicated; in particular, it requires the solution of a sequence of linear least squares problems of special form. It is for this purpose that subroutine QRSOLV is used.

The algorithm used in ENORM is a simplified version of Blue's [1978] algorithm. An advantage of the MINPACK-1 version is that it does not require machine constants; a disadvantage is that nondestructive underflows are allowed.

The banded Jacobian option in FDJAC1 is based on the work of Curtis, Powell, and Reid [1974].

QRFAC and RWUPDT are based on the corresponding algorithms in LINPACK (Dongarra, Bunch, Moler, and Stewart [1979]).

The algorithm used in R1UPDT is based on the work of Gill, Golub, Murray, and Saunders [1974].

References

Blue, J. L. [1978]. A portable Fortran program to find the Euclidean norm of a vector, ACM Transactions on Mathematical Software 4, 15-23.

Boyle, J. M. and Dritz, K. W. [1974]. An automated programming system to facilitate the development of quality mathematical software, Proceedings IFIP Congress, North-Holland.

Broyden, C. G. [1965]. A class of methods for solving nonlinear simultaneous equations, Math. Comp. 19, 577-593.

Curtis, A. R., Powell, M. J. D., and Reid, J. K. [1974]. On the estimation of sparse Jacobian matrices, J. Inst. Maths Applics 13, 117-119.

Dongarra, J. J., Bunch, J. R., Moler, C. B., and Stewart, G. W. [1979]. LINPACK users' guide, SIAM Publications.

Fosdick, L. D. [1974]. BRNANL, A Fortran program to identify basic blocks in Fortran programs, University of Colorado, Computer Science report CU-CS-040-74.

Fox, P. A., Hall, A. D., and Schryer, N. L. [1978]. The PORT mathematical subroutine library, ACM Transactions on Mathematical Software 4, 104-126.

Garbow, B. S., Hillstrom, K. E., and Moré, J. J. [1980]. Implementation guide for MINPACK-1, Argonne National Laboratory report ANL-80-68.

Gill, P. E., Golub, G. H., Murray, W., and Saunders, M. A. [1974]. Methods for modifying matrix factorizations, Math. Comp. 28, 505-535.

Hebden, M. D. [1973]. An algorithm for minimization using exact second derivatives, Atomic Energy Research Establishment report TP 515, Harwell, England.

Hiebert, K. L. [1979]. A comparison of nonlinear least squares software, Sandia Laboratories report SAND 79-0483, Albuquerque, New Mexico.

Hiebert, K. L. [1980]. A comparison of software which solves systems of nonlinear equations, Sandia Laboratories report SAND 80-0181, Albuquerque, New Mexico.

Levenberg, K. [1944]. A method for the solution of certain nonlinear problems in least squares, Quart. Appl. Math. 2, 164-168.

Marquardt, D. W. [1963]. An algorithm for least-squares estimation of nonlinear parameters, SIAM J. Appl. Math. 11, 431-441.

Moré, J. J. [1977]. The Levenberg-Marquardt algorithm: Implementation and Theory, Numerical Analysis, G. A. Watson, ed., Lecture Notes in Mathematics 630, Springer-Verlag.

Moré, J. J., Garbow, B. S., and Hillstrom, K. E. [1978]. Testing unconstrained optimization software, Argonne National Laboratory, Applied Mathematics Division Technical Memorandum 324 (to appear in ACM Transactions on Mathematical Software).

Powell, M. J. D. [1970]. A hybrid method for nonlinear equations, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach.

Ryder, B. G. [1974]. The PFORT verifier, Software Practice and Experience 4, 359-377.

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

CHAPTER 4
Documentation

This chapter contains the double precision version of the MINPACK-1 documentation; both single and double precision versions of the documentation are available in machine-readable form with the MINPACK-1 package. The documentation appears in the following order:

<u>Systems of nonlinear equations</u>

HYBRD1, HYBRD, HYBRJ1, HYBRJ

<u>Nonlinear least squares problems</u>

LMDIF1, LMDIF, LMDER1, LMDER, LMSTR1, LMSTR

<u>Derivative checking</u>

CHKDER

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Documentation for MINPACK subroutine HYBRD1

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of HYBRD1 is to find a zero of a system of N nonlinear functions in N variables by a modification of the Powell hybrid method. This is done by using the more general nonlinear equation solver HYBRD. The user must provide a subroutine which calculates the functions. The Jacobian is then calculated by a forward-difference approximation.

2. Subroutine and type statements.

```
SUBROUTINE HYBRD1(FCN,N,X,FVEC,TOL,INFO,WA,LWA)
INTEGER N,INFO,LWA
DOUBLE PRECISION TOL
DOUBLE PRECISION X(N),FVEC(N),WA(LWA)
EXTERNAL FCN
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to HYBRD1 and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from HYBRD1.

FCN is the name of the user-supplied subroutine which calculates the functions. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
SUBROUTINE FCN(N,X,FVEC,IFLAG)
INTEGER N,IFLAG
DOUBLE PRECISION X(N),FVEC(N)
----------
CALCULATE THE FUNCTIONS AT X AND
RETURN THIS VECTOR IN FVEC.
----------
RETURN
END
```

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of HYBRD1. In this case set IFLAG to a negative integer.

N is a positive integer input variable set to the number of functions and variables.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length N which contains the functions evaluated at the output X.

TOL is a nonnegative input variable. Termination occurs when the algorithm estimates that the relative error between X and the solution is at most TOL. Section 4 contains more details about TOL.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0 Improper input parameters.

INFO = 1 Algorithm estimates that the relative error between X and the solution is at most TOL.

INFO = 2 Number of calls to FCN has reached or exceeded 200*(N+1).

INFO = 3 TOL is too small. No further improvement in the approximate solution X is possible.

INFO = 4 Iteration is not making good progress.

Sections 4 and 5 contain more details about INFO.

WA is a work array of length LWA.

LWA is a positive integer input variable not less than (N*(3*N+13))/2.

4. Successful completion.

The accuracy of HYBRD1 is controlled by the convergence parameter TOL. This parameter is used in a test which makes a comparison between the approximation X and a solution XSOL. HYBRD1 terminates when the test is satisfied. If TOL is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then HYBRD1 only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible. Unless high precision solutions are required, the recommended value for TOL is the square root of the machine precision.

The test assumes that the functions are reasonably well behaved.

If this condition is not satisfied, then HYBRD1 may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning HYBRD1 with a tighter tolerance.

Convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z, then this test attempts to guarantee that

ENORM(X-XSOL) .LE. TOL*ENORM(XSOL).

If this condition is satisfied with TOL = 10**(-K), then the larger components of X have K significant decimal digits and INFO is set to 1. There is a danger that the smaller components of X may have large relative errors, but the fast rate of convergence of HYBRD1 usually avoids this possibility.

5. Unsuccessful completion.

Unsuccessful termination of HYBRD1 can be due to improper input parameters, arithmetic interrupts, an excessive number of function evaluations, errors in the functions, or lack of good progress.

Improper input parameters. INFO is set to 0 if N .LE. 0, or TOL .LT. 0.D0, or LWA .LT. (N*(3*N+13))/2.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by HYBRD1. In this case, it may be possible to remedy the situation by not evaluating the functions here, but instead setting the components of FVEC to numbers that exceed those in the initial FVEC, thereby indirectly reducing the step length. The step length can be more directly controlled by using instead HYBRD, which includes in its calling sequence the step-length- governing parameter FACTOR.

Excessive number of function evaluations. If the number of calls to FCN reaches 200*(N+1), then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and INFO is set to 2. This situation should be unusual because, as indicated below, lack of good progress is usually diagnosed earlier by HYBRD1, causing termination with INFO = 4.

Errors in the functions. The choice of step length in the forward-difference approximation to the Jacobian assumes that the relative errors in the functions are of the order of the machine precision. If this is not the case, HYBRD1 may fail (usually with INFO = 4). The user should then use HYBRD instead, or one of the programs which require the analytic Jacobian (HYBRJ1 and HYBRJ).

Lack of good progress. HYBRD1 searches for a zero of the system by minimizing the sum of the squares of the functions. In so doing, it can become trapped in a region where the minimum does not correspond to a zero of the system and, in this situation, the iteration eventually fails to make good progress. In particular, this will happen if the system does not have a zero. If the system has a zero, rerunning HYBRD1 from a different starting point may be helpful.

6. Characteristics of the algorithm.

HYBRD1 is a modification of the Powell hybrid method. Two of its main characteristics involve the choice of the correction as a convex combination of the Newton and scaled gradient directions, and the updating of the Jacobian by the rank-1 method of Broyden. The choice of the correction guarantees (under reasonable conditions) global convergence for starting points far from the solution and a fast rate of convergence. The Jacobian is approximated by forward differences at the starting point, but forward differences are not used again until the rank-1 method fails to produce satisfactory progress.

Timing. The time required by HYBRD1 to solve a given problem depends on N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by HYBRD1 is about 11.5*(N**2) to process each call to FCN. Unless FCN can be evaluated quickly, the timing of HYBRD1 will be strongly influenced by the time spent in FCN.

Storage. HYBRD1 requires (3*N**2 + 17*N)/2 double precision storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DOGLEG,DPMPAR,ENORM,FDJAC1,HYBRD,
QFORM,QRFAC,R1MPYQ,R1UPDT

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MIN0,MOD

8. References.

M. J. D. Powell, A Hybrid Method for Nonlinear Equations. Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, editor. Gordon and Breach, 1970.

9. Example.

The problem is to determine the values of x(1), x(2), ..., x(9), which solve the system of tridiagonal equations

```
(3-2*x(1))*x(1)               -2*x(2)                     = -1
        -x(i-1) + (3-2*x(i))*x(i)             -2*x(i+1)  = -1, i=2-8
                            -x(8) + (3-2*x(9))*x(9)       = -1
```

```
C     **********
C
C     DRIVER FOR HYBRD1 EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER J,N,INFO,LWA,NWRITE
      DOUBLE PRECISION TOL,FNORM
      DOUBLE PRECISION X(9),FVEC(9),WA(180)
      DOUBLE PRECISION ENORM,DPMPAR
      EXTERNAL FCN
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      N = 9
C
C     THE FOLLOWING STARTING VALUES PROVIDE A ROUGH SOLUTION.
C
      DO 10 J = 1, 9
         X(J) = -1.DO
   10    CONTINUE
C
      LWA = 180
C
C     SET TOL TO THE SQUARE ROOT OF THE MACHINE PRECISION.
C     UNLESS HIGH PRECISION SOLUTIONS ARE REQUIRED,
C     THIS IS THE RECOMMENDED SETTING.
C
      TOL = DSQRT(DPMPAR(1))
C
      CALL HYBRD1(FCN,N,X,FVEC,TOL,INFO,WA,LWA)
      FNORM = ENORM(N,FVEC)
      WRITE (NWRITE,1000) FNORM,INFO,(X(J),J=1,N)
      STOP
 1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
     *        5X,15H EXIT PARAMETER,16X,I10 //
     *        5X,27H FINAL APPROXIMATE SOLUTION // (5X,3D15.7))
C
C     LAST CARD OF DRIVER FOR HYBRD1 EXAMPLE.
C
      END
      SUBROUTINE FCN(N,X,FVEC,IFLAG)
      INTEGER N,IFLAG
      DOUBLE PRECISION X(N),FVEC(N)
C
```

```
C     SUBROUTINE FCN FOR HYBRD1 EXAMPLE.
C
      INTEGER K
      DOUBLE PRECISION ONE,TEMP,TEMP1,TEMP2,THREE,TWO,ZERO
      DATA ZERO,ONE,TWO,THREE /0.D0,1.D0,2.D0,3.D0/
C
      DO 10 K = 1, N
         TEMP = (THREE - TWO*X(K))*X(K)
         TEMP1 = ZERO
         IF (K .NE. 1) TEMP1 = X(K-1)
         TEMP2 = ZERO
         IF (K .NE. N) TEMP2 = X(K+1)
         FVEC(K) = TEMP - TEMP1 - TWO*TEMP2 + ONE
   10    CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

```
Results obtained with different compilers or machines
may be slightly different.

FINAL L2 NORM OF THE RESIDUALS  0.1192636D-07

EXIT PARAMETER                              1

FINAL APPROXIMATE SOLUTION

-0.5706545D+00 -0.6816283D+00 -0.7017325D+00
-0.7042129D+00 -0.7013690D+00 -0.6918656D+00
-0.6657920D+00 -0.5960342D+00 -0.4164121D+00
```

Documentation for MINPACK subroutine HYBRD

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of HYBRD is to find a zero of a system of N nonlinear functions in N variables by a modification of the Powell hybrid method. The user must provide a subroutine which calculates the functions. The Jacobian is then calculated by a forward-difference approximation.

2. Subroutine and type statements.

```
      SUBROUTINE HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,DIAG,
     *                 MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,
     *                 R,LR,QTF,WA1,WA2,WA3,WA4)
      INTEGER N,MAXFEV,ML,MU,MODE,NPRINT,INFO,NFEV,LDFJAC,LR
      DOUBLE PRECISION XTOL,EPSFCN,FACTOR
      DOUBLE PRECISION X(N),FVEC(N),DIAG(N),FJAC(LDFJAC,N),R(LR),QTF(N),
     *                 WA1(N),WA2(N),WA3(N),WA4(N)
      EXTERNAL FCN
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to HYBRD and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from HYBRD.

FCN is the name of the user-supplied subroutine which calculates the functions. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
  SUBROUTINE FCN(N,X,FVEC,IFLAG)
  INTEGER N,IFLAG
  DOUBLE PRECISION X(N),FVEC(N)
  ----------
  CALCULATE THE FUNCTIONS AT X AND
  RETURN THIS VECTOR IN FVEC.
  ----------
  RETURN
  END
```

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of HYBRD. In this case set IFLAG to a negative integer.

N is a positive integer input variable set to the number of functions and variables.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length N which contains the functions evaluated at the output X.

XTOL is a nonnegative input variable. Termination occurs when the relative error between two consecutive iterates is at most XTOL. Therefore, XTOL measures the relative error desired in the approximate solution. Section 4 contains more details about XTOL.

MAXFEV is a positive integer input variable. Termination occurs when the number of calls to FCN is at least MAXFEV by the end of an iteration.

ML is a nonnegative integer input variable which specifies the number of subdiagonals within the band of the Jacobian matrix. If the Jacobian is not banded, set ML to at least N - 1.

MU is a nonnegative integer input variable which specifies the number of superdiagonals within the band of the Jacobian matrix. If the Jacobian is not banded, set MU to at least N - 1.

EPSFCN is an input variable used in determining a suitable step for the forward-difference approximation. This approximation assumes that the relative errors in the functions are of the order of EPSFCN. If EPSFCN is less than the machine precision, it is assumed that the relative errors in the functions are of the order of the machine precision.

DIAG is an array of length N. If MODE = 1 (see below), DIAG is internally set. If MODE = 2, DIAG must contain positive entries that serve as multiplicative scale factors for the variables.

MODE is an integer input variable. If MODE = 1, the variables will be scaled internally. If MODE = 2, the scaling is specified by the input DIAG. Other values of MODE are equivalent to MODE = 1.

FACTOR is a positive input variable used in determining the initial step bound. This bound is set to the product of FACTOR and the Euclidean norm of DIAG*X if nonzero, or else to FACTOR itself. In most cases FACTOR should lie in the interval (.1,100.). 100. is a generally recommended value.

NPRINT is an integer input variable that enables controlled printing of iterates if it is positive. In this case, FCN is called with IFLAG = 0 at the beginning of the first iteration and every NPRINT iterations thereafter and immediately prior to return, with X and FVEC available for printing. If NPRINT is not positive, no special calls of FCN with IFLAG = 0 are made.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0 Improper input parameters.

INFO = 1 Relative error between two consecutive iterates is at most XTOL.

INFO = 2 Number of calls to FCN has reached or exceeded MAXFEV.

INFO = 3 XTOL is too small. No further improvement in the approximate solution X is possible.

INFO = 4 Iteration is not making good progress, as measured by the improvement from the last five Jacobian evaluations.

INFO = 5 Iteration is not making good progress, as measured by the improvement from the last ten iterations.

Sections 4 and 5 contain more details about INFO.

NFEV is an integer output variable set to the number of calls to FCN.

FJAC is an output N by N array which contains the orthogonal matrix Q produced by the QR factorization of the final approximate Jacobian.

LDFJAC is a positive integer input variable not less than N which specifies the leading dimension of the array FJAC.

R is an output array of length LR which contains the upper triangular matrix produced by the QR factorization of the final approximate Jacobian, stored rowwise.

LR is a positive integer input variable not less than (N*(N+1))/2.

QTF is an output array of length N which contains the vector (Q transpose)*FVEC.

WA1, WA2, WA3, and WA4 are work arrays of length N.

4. Successful completion.

The accuracy of HYBRD is controlled by the convergence parameter XTOL. This parameter is used in a test which makes a comparison between the approximation X and a solution XSOL. HYBRD terminates when the test is satisfied. If the convergence parameter is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then HYBRD only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible.

The test assumes that the functions are reasonably well behaved. If this condition is not satisfied, then HYBRD may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning HYBRD with a tighter tolerance.

Convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z and D is the diagonal matrix whose entries are defined by the array DIAG, then this test attempts to guarantee that

```
ENORM(D*(X-XSOL)) .LE. XTOL*ENORM(D*XSOL).
```

If this condition is satisfied with XTOL = 10**(-K), then the larger components of D*X have K significant decimal digits and INFO is set to 1. There is a danger that the smaller components of D*X may have large relative errors, but the fast rate of convergence of HYBRD usually avoids this possibility. Unless high precision solutions are required, the recommended value for XTOL is the square root of the machine precision.

5. Unsuccessful completion.

Unsuccessful termination of HYBRD can be due to improper input parameters, arithmetic interrupts, an excessive number of function evaluations, or lack of good progress.

Improper input parameters. INFO is set to 0 if N .LE. 0, or XTOL .LT. 0.D0, or MAXFEV .LE. 0, or ML .LT. 0, or MU .LT. 0, or FACTOR .LE. 0.D0, or LDFJAC .LT. N, or LR .LT. (N*(N+1))/2.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by HYBRD. In this case, it may be possible to remedy the situation by rerunning HYBRD with a smaller value of FACTOR.

Excessive number of function evaluations. A reasonable value for MAXFEV is 200*(N+1). If the number of calls to FCN reaches MAXFEV, then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and

INFO is set to 2. This situation should be unusual because, as indicated below, lack of good progress is usually diagnosed earlier by HYBRD, causing termination with INFO = 4 or INFO = 5.

Lack of good progress. HYBRD searches for a zero of the system by minimizing the sum of the squares of the functions. In so doing, it can become trapped in a region where the minimum does not correspond to a zero of the system and, in this situation, the iteration eventually fails to make good progress. In particular, this will happen if the system does not have a zero. If the system has a zero, rerunning HYBRD from a different starting point may be helpful.

6. Characteristics of the algorithm.

HYBRD is a modification of the Powell hybrid method. Two of its main characteristics involve the choice of the correction as a convex combination of the Newton and scaled gradient directions, and the updating of the Jacobian by the rank-1 method of Broyden. The choice of the correction guarantees (under reasonable conditions) global convergence for starting points far from the solution and a fast rate of convergence. The Jacobian is approximated by forward differences at the starting point, but forward differences are not used again until the rank-1 method fails to produce satisfactory progress.

Timing. The time required by HYBRD to solve a given problem depends on N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by HYBRD is about 11.5*(N**2) to process each call to FCN. Unless FCN can be evaluated quickly, the timing of HYBRD will be strongly influenced by the time spent in FCN.

Storage. HYBRD requires (3*N**2 + 17*N)/2 double precision storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DOGLEG,DPMPAR,ENORM,FDJAC1,
QFORM,QRFAC,R1MPYQ,R1UPDT

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MIN0,MOD

8. References.

M. J. D. Powell, A Hybrid Method for Nonlinear Equations.

Numerical Methods for Nonlinear Algebraic Equations,
P. Rabinowitz, editor. Gordon and Breach, 1970.

9. Example.

The problem is to determine the values of x(1), x(2), ..., x(9), which solve the system of tridiagonal equations

```
(3-2*x(1))*x(1)              -2*x(2)                      = -1
        -x(i-1) + (3-2*x(i))*x(i)             -2*x(i+1) = -1, i=2-8
                             -x(8) + (3-2*x(9))*x(9) = -1
```

```
C     **********
C
C     DRIVER FOR HYBRD EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER J,N,MAXFEV,ML,MU,MODE,NPRINT,INFO,NFEV,LDFJAC,LR,NWRITE
      DOUBLE PRECISION XTOL,EPSFCN,FACTOR,FNORM
      DOUBLE PRECISION X(9),FVEC(9),DIAG(9),FJAC(9,9),R(45),QTF(9),
     *                 WA1(9),WA2(9),WA3(9),WA4(9)
      DOUBLE PRECISION ENORM,DPMPAR
      EXTERNAL FCN
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      N = 9
C
C     THE FOLLOWING STARTING VALUES PROVIDE A ROUGH SOLUTION.
C
      DO 10 J = 1, 9
         X(J) = -1.D0
   10    CONTINUE
C
      LDFJAC = 9
      LR = 45
C
C     SET XTOL TO THE SQUARE ROOT OF THE MACHINE PRECISION.
C     UNLESS HIGH PRECISION SOLUTIONS ARE REQUIRED,
C     THIS IS THE RECOMMENDED SETTING.
C
      XTOL = DSQRT(DPMPAR(1))
C
      MAXFEV = 2000
      ML = 1
      MU = 1
      EPSFCN = 0.D0
      MODE = 2
      DO 20 J = 1, 9
         DIAG(J) = 1.D0
```

```
   20    CONTINUE
      FACTOR = 1.D2
      NPRINT = 0
C
      CALL HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,DIAG,
     *           MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,
     *           R,LR,QTF,WA1,WA2,WA3,WA4)
      FNORM = ENORM(N,FVEC)
      WRITE (NWRITE,1000) FNORM,NFEV,INFO,(X(J),J=1,N)
      STOP
 1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
     *        5X,31H NUMBER OF FUNCTION EVALUATIONS,I10 //
     *        5X,15H EXIT PARAMETER,16X,I10 //
     *        5X,27H FINAL APPROXIMATE SOLUTION // (5X,3D15.7))
C
C     LAST CARD OF DRIVER FOR HYBRD EXAMPLE.
C
      END
      SUBROUTINE FCN(N,X,FVEC,IFLAG)
      INTEGER N,IFLAG
      DOUBLE PRECISION X(N),FVEC(N)
C
C     SUBROUTINE FCN FOR HYBRD EXAMPLE.
C
      INTEGER K
      DOUBLE PRECISION ONE,TEMP,TEMP1,TEMP2,THREE,TWO,ZERO
      DATA ZERO,ONE,TWO,THREE /0.D0,1.D0,2.D0,3.D0/
C
      IF (IFLAG .NE. 0) GO TO 5
C
C     INSERT PRINT STATEMENTS HERE WHEN NPRINT IS POSITIVE.
C
      RETURN
    5 CONTINUE
      DO 10 K = 1, N
         TEMP = (THREE - TWO*X(K))*X(K)
         TEMP1 = ZERO
         IF (K .NE. 1) TEMP1 = X(K-1)
         TEMP2 = ZERO
         IF (K .NE. N) TEMP2 = X(K+1)
         FVEC(K) = TEMP - TEMP1 - TWO*TEMP2 + ONE
   10    CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

Results obtained with different compilers or machines may be slightly different.

```
FINAL L2 NORM OF THE RESIDUALS  0.1192636D-07

NUMBER OF FUNCTION EVALUATIONS        14
```

EXIT PARAMETER 1

FINAL APPROXIMATE SOLUTION

```
-0.5706545D+00 -0.6816283D+00 -0.7017325D+00
-0.7042129D+00 -0.7013690D+00 -0.6918656D+00
-0.6657920D+00 -0.5960342D+00 -0.4164121D+00
```

Documentation for MINPACK subroutine HYBRJ1

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of HYBRJ1 is to find a zero of a system of N non-linear functions in N variables by a modification of the Powell hybrid method. This is done by using the more general nonlinear equation solver HYBRJ. The user must provide a subroutine which calculates the functions and the Jacobian.

2. Subroutine and type statements.

```
SUBROUTINE HYBRJ1(FCN,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,WA,LWA)
INTEGER N,LDFJAC,INFO,LWA
DOUBLE PRECISION TOL
DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N),WA(LWA)
EXTERNAL FCN
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to HYBRJ1 and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from HYBRJ1.

FCN is the name of the user-supplied subroutine which calculates the functions and the Jacobian. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
INTEGER N,LDFJAC,IFLAG
DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)
----------
IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
RETURN THIS VECTOR IN FVEC.  DO NOT ALTER FJAC.
IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND
RETURN THIS MATRIX IN FJAC.  DO NOT ALTER FVEC.
----------
RETURN
END
```

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of HYBRJ1. In this case set IFLAG to a negative integer.

N is a positive integer input variable set to the number of functions and variables.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length N which contains the functions evaluated at the output X.

FJAC is an output N by N array which contains the orthogonal matrix Q produced by the QR factorization of the final approximate Jacobian. Section 6 contains more details about the approximation to the Jacobian.

LDFJAC is a positive integer input variable not less than N which specifies the leading dimension of the array FJAC.

TOL is a nonnegative input variable. Termination occurs when the algorithm estimates that the relative error between X and the solution is at most TOL. Section 4 contains more details about TOL.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0 Improper input parameters.

INFO = 1 Algorithm estimates that the relative error between X and the solution is at most TOL.

INFO = 2 Number of calls to FCN with IFLAG = 1 has reached 100*(N+1).

INFO = 3 TOL is too small. No further improvement in the approximate solution X is possible.

INFO = 4 Iteration is not making good progress.

Sections 4 and 5 contain more details about INFO.

WA is a work array of length LWA.

LWA is a positive integer input variable not less than (N*(N+13))/2.

4. Successful completion.

The accuracy of HYBRJ1 is controlled by the convergence parameter TOL. This parameter is used in a test which makes a comparison between the approximation X and a solution XSOL. HYBRJ1 terminates when the test is satisfied. If TOL is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then HYBRJ1 only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible. Unless high precision solutions are required, the recommended value for TOL is the square root of the machine precision.

The test assumes that the functions and the Jacobian are coded consistently, and that the functions are reasonably well behaved. If these conditions are not satisfied, then HYBRJ1 may incorrectly indicate convergence. The coding of the Jacobian can be checked by the MINPACK subroutine CHKDER. If the Jacobian is coded correctly, then the validity of the answer can be checked, for example, by rerunning HYBRJ1 with a tighter tolerance.

Convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z, then this test attempts to guarantee that

```
ENORM(X-XSOL) .LE. TOL*ENORM(XSOL).
```

If this condition is satisfied with TOL = 10**(-K), then the larger components of X have K significant decimal digits and INFO is set to 1. There is a danger that the smaller components of X may have large relative errors, but the fast rate of convergence of HYBRJ1 usually avoids this possibility.

5. Unsuccessful completion.

Unsuccessful termination of HYBRJ1 can be due to improper input parameters, arithmetic interrupts, an excessive number of function evaluations, or lack of good progress.

Improper input parameters. INFO is set to 0 if N .LE. 0, or LDFJAC .LT. N, or TOL .LT. 0.D0, or LWA .LT. (N*(N+13))/2.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by HYBRJ1. In this case, it may be possible to remedy the situation by not evaluating the functions here, but instead setting the components of FVEC to numbers that exceed those in the initial FVEC, thereby indirectly reducing the step length. The step length can be more directly controlled by using instead HYBRJ, which includes in its calling sequence the step-length- governing parameter FACTOR.

Excessive number of function evaluations. If the number of calls to FCN with IFLAG = 1 reaches 100*(N+1), then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and INFO is set to 2. This situation should be unusual because, as indicated below, lack of good progress is usually diagnosed earlier by HYBRJ1, causing termination with INFO = 4.

Lack of good progress. HYBRJ1 searches for a zero of the system by minimizing the sum of the squares of the functions. In so doing, it can become trapped in a region where the minimum does not correspond to a zero of the system and, in this situation, the iteration eventually fails to make good progress. In particular, this will happen if the system does not have a zero. If the system has a zero, rerunning HYBRJ1 from a different starting point may be helpful.

6. Characteristics of the algorithm.

HYBRJ1 is a modification of the Powell hybrid method. Two of its main characteristics involve the choice of the correction as a convex combination of the Newton and scaled gradient directions, and the updating of the Jacobian by the rank-1 method of Broyden. The choice of the correction guarantees (under reasonable conditions) global convergence for starting points far from the solution and a fast rate of convergence. The Jacobian is calculated at the starting point, but it is not recalculated until the rank-1 method fails to produce satisfactory progress.

Timing. The time required by HYBRJ1 to solve a given problem depends on N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by HYBRJ1 is about 11.5*(N**2) to process each evaluation of the functions (call to FCN with IFLAG = 1) and 1.3*(N**3) to process each evaluation of the Jacobian (call to FCN with IFLAG = 2). Unless FCN can be evaluated quickly, the timing of HYBRJ1 will be strongly influenced by the time spent in FCN.

Storage. HYBRJ1 requires (3*N**2 + 17*N)/2 double precision storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DOGLEG,DPMPAR,ENORM,HYBRJ,
QFORM,QRFAC,R1MPYQ,R1UPDT

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MIN0,MOD

8. References.

M. J. D. Powell, A Hybrid Method for Nonlinear Equations.
Numerical Methods for Nonlinear Algebraic Equations,
P. Rabinowitz, editor. Gordon and Breach, 1970.

9. Example.

The problem is to determine the values of x(1), x(2), ..., x(9), which solve the system of tridiagonal equations

```
(3-2*x(1))*x(1)           -2*x(2)                   = -1
        -x(i-1) + (3-2*x(i))*x(i)         -2*x(i+1) = -1, i=2-8
                          -x(8) + (3-2*x(9))*x(9) = -1
```

```
C     **********
C
C     DRIVER FOR HYBRJ1 EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER J,N,LDFJAC,INFO,LWA,NWRITE
      DOUBLE PRECISION TOL,FNORM
      DOUBLE PRECISION X(9),FVEC(9),FJAC(9,9),WA(99)
      DOUBLE PRECISION ENORM,DPMPAR
      EXTERNAL FCN
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      N = 9
C
C     THE FOLLOWING STARTING VALUES PROVIDE A ROUGH SOLUTION.
C
      DO 10 J = 1, 9
         X(J) = -1.D0
   10    CONTINUE
C
      LDFJAC = 9
      LWA = 99
C
C     SET TOL TO THE SQUARE ROOT OF THE MACHINE PRECISION.
C     UNLESS HIGH PRECISION SOLUTIONS ARE REQUIRED,
C     THIS IS THE RECOMMENDED SETTING.
C
      TOL = DSQRT(DPMPAR(1))
C
      CALL HYBRJ1(FCN,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,WA,LWA)
      FNORM = ENORM(N,FVEC)
      WRITE (NWRITE,1000) FNORM,INFO,(X(J),J=1,N)
      STOP
 1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
     *        5X,15H EXIT PARAMETER,16X,I10 //
     *        5X,27H FINAL APPROXIMATE SOLUTION // (5X,3D15.7))
```

```
C
C     LAST CARD OF DRIVER FOR HYBRJ1 EXAMPLE.
C
      END
      SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
      INTEGER N,LDFJAC,IFLAG
      DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)
C
C     SUBROUTINE FCN FOR HYBRJ1 EXAMPLE.
C
      INTEGER J,K
      DOUBLE PRECISION ONE,TEMP,TEMP1,TEMP2,THREE,TWO,ZERO
      DATA ZERO,ONE,TWO,THREE,FOUR /0.D0,1.D0,2.D0,3.D0,4.D0/
C
      IF (IFLAG .EQ. 2) GO TO 20
      DO 10 K = 1, N
         TEMP = (THREE - TWO*X(K))*X(K)
         TEMP1 = ZERO
         IF (K .NE. 1) TEMP1 = X(K-1)
         TEMP2 = ZERO
         IF (K .NE. N) TEMP2 = X(K+1)
         FVEC(K) = TEMP - TEMP1 - TWO*TEMP2 + ONE
   10    CONTINUE
      GO TO 50
   20 CONTINUE
      DO 40 K = 1, N
         DO 30 J = 1, N
            FJAC(K,J) = ZERO
   30       CONTINUE
         FJAC(K,K) = THREE - FOUR*X(K)
         IF (K .NE. 1) FJAC(K,K-1) = -ONE
         IF (K .NE. N) FJAC(K,K+1) = -TWO
   40    CONTINUE
   50 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

```
Results obtained with different compilers or machines
may be slightly different.

FINAL L2 NORM OF THE RESIDUALS  0.1192636D-07

EXIT PARAMETER                              1

FINAL APPROXIMATE SOLUTION

-0.5706545D+00 -0.6816283D+00 -0.7017325D+00
-0.7042129D+00 -0.7013690D+00 -0.6918656D+00
-0.6657920D+00 -0.5960342D+00 -0.4164121D+00
```

Documentation for MINPACK subroutine HYBRJ

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of HYBRJ is to find a zero of a system of N non-linear functions in N variables by a modification of the Powell hybrid method. The user must provide a subroutine which calculates the functions and the Jacobian.

2. Subroutine and type statements.

```
      SUBROUTINE HYBRJ(FCN,N,X,FVEC,FJAC,LDFJAC,XTOL,MAXFEV,DIAG,
     *                 MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,R,LR,QTF,
     *                 WA1,WA2,WA3,WA4)
      INTEGER N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV,LR
      DOUBLE PRECISION XTOL,FACTOR
      DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N),DIAG(N),R(LR),QTF(N),
     *                 WA1(N),WA2(N),WA3(N),WA4(N)
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to HYBRJ and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from HYBRJ.

FCN is the name of the user-supplied subroutine which calculates the functions and the Jacobian. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
INTEGER N,LDFJAC,IFLAG
DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)
----------
IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
RETURN THIS VECTOR IN FVEC.  DO NOT ALTER FJAC.
IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND
RETURN THIS MATRIX IN FJAC.  DO NOT ALTER FVEC.
----------
RETURN
END
```

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of HYBRJ. In this case set IFLAG to a negative integer.

N is a positive integer input variable set to the number of functions and variables.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length N which contains the functions evaluated at the output X.

FJAC is an output N by N array which contains the orthogonal matrix Q produced by the QR factorization of the final approximate Jacobian. Section 6 contains more details about the approximation to the Jacobian.

LDFJAC is a positive integer input variable not less than N which specifies the leading dimension of the array FJAC.

XTOL is a nonnegative input variable. Termination occurs when the relative error between two consecutive iterates is at most XTOL. Therefore, XTOL measures the relative error desired in the approximate solution. Section 4 contains more details about XTOL.

MAXFEV is a positive integer input variable. Termination occurs when the number of calls to FCN with IFLAG = 1 has reached MAXFEV.

DIAG is an array of length N. If MODE = 1 (see below), DIAG is internally set. If MODE = 2, DIAG must contain positive entries that serve as multiplicative scale factors for the variables.

MODE is an integer input variable. If MODE = 1, the variables will be scaled internally. If MODE = 2, the scaling is specified by the input DIAG. Other values of MODE are equivalent to MODE = 1.

FACTOR is a positive input variable used in determining the initial step bound. This bound is set to the product of FACTOR and the Euclidean norm of DIAG*X if nonzero, or else to FACTOR itself. In most cases FACTOR should lie in the interval (.1,100.). 100. is a generally recommended value.

NPRINT is an integer input variable that enables controlled printing of iterates if it is positive. In this case, FCN is called with IFLAG = 0 at the beginning of the first iteration and every NPRINT iterations thereafter and immediately prior to return, with X and FVEC available for printing. FVEC and FJAC should not be altered. If NPRINT is not positive, no special calls of FCN with IFLAG = 0 are made.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0 Improper input parameters.

INFO = 1 Relative error between two consecutive iterates is at most XTOL.

INFO = 2 Number of calls to FCN with IFLAG = 1 has reached MAXFEV.

INFO = 3 XTOL is too small. No further improvement in the approximate solution X is possible.

INFO = 4 Iteration is not making good progress, as measured by the improvement from the last five Jacobian evaluations.

INFO = 5 Iteration is not making good progress, as measured by the improvement from the last ten iterations.

Sections 4 and 5 contain more details about INFO.

NFEV is an integer output variable set to the number of calls to FCN with IFLAG = 1.

NJEV is an integer output variable set to the number of calls to FCN with IFLAG = 2.

R is an output array of length LR which contains the upper triangular matrix produced by the QR factorization of the final approximate Jacobian, stored rowwise.

LR is a positive integer input variable not less than (N*(N+1))/2.

QTF is an output array of length N which contains the vector (Q transpose)*FVEC.

WA1, WA2, WA3, and WA4 are work arrays of length N.

4. Successful completion.

The accuracy of HYBRJ is controlled by the convergence parameter XTOL. This parameter is used in a test which makes a comparison between the approximation X and a solution XSOL. HYBRJ terminates when the test is satisfied. If the convergence parameter is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then HYBRJ only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible.

The test assumes that the functions and the Jacobian are coded consistently, and that the functions are reasonably well behaved. If these conditions are not satisfied, then HYBRJ may incorrectly indicate convergence. The coding of the Jacobian can be checked by the MINPACK subroutine CHKDER. If the Jacobian is coded correctly, then the validity of the answer can be checked, for example, by rerunning HYBRJ with a tighter tolerance.

Convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z and D is the diagonal matrix whose entries are defined by the array DIAG, then this test attempts to guarantee that

```
ENORM(D*(X-XSOL)) .LE. XTOL*ENORM(D*XSOL).
```

If this condition is satisfied with XTOL = 10**(-K), then the larger components of D*X have K significant decimal digits and INFO is set to 1. There is a danger that the smaller components of D*X may have large relative errors, but the fast rate of convergence of HYBRJ usually avoids this possibility. Unless high precision solutions are required, the recommended value for XTOL is the square root of the machine precision.

5. Unsuccessful completion.

Unsuccessful termination of HYBRJ can be due to improper input parameters, arithmetic interrupts, an excessive number of function evaluations, or lack of good progress.

Improper input parameters. INFO is set to 0 if N .LE. 0, or LDFJAC .LT. N, or XTOL .LT. 0.D0, or MAXFEV .LE. 0, or FACTOR .LE. 0.D0, or LR .LT. (N*(N+1))/2.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by HYBRJ. In this case, it may be possible to remedy the situation by rerunning HYBRJ with a smaller value of FACTOR.

Excessive number of function evaluations. A reasonable value for MAXFEV is 100*(N+1). If the number of calls to FCN with IFLAG = 1 reaches MAXFEV, then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and INFO is set to 2. This situation should be unusual because, as indicated below, lack of good progress is usually diagnosed earlier by HYBRJ, causing termination with INFO = 4 or INFO = 5.

Lack of good progress. HYBRJ searches for a zero of the system by minimizing the sum of the squares of the functions. In so doing, it can become trapped in a region where the minimum does not correspond to a zero of the system and, in this situation, the iteration eventually fails to make good progress. In particular, this will happen if the system does not have a zero. If the system has a zero, rerunning HYBRJ from a different starting point may be helpful.

6. Characteristics of the algorithm.

HYBRJ is a modification of the Powell hybrid method. Two of its main characteristics involve the choice of the correction as a convex combination of the Newton and scaled gradient directions, and the updating of the Jacobian by the rank-1 method of Broyden. The choice of the correction guarantees (under reasonable conditions) global convergence for starting points far from the solution and a fast rate of convergence. The Jacobian is calculated at the starting point, but it is not recalculated until the rank-1 method fails to produce satisfactory progress.

Timing. The time required by HYBRJ to solve a given problem depends on N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by HYBRJ is about 11.5*(N**2) to process each evaluation of the functions (call to FCN with IFLAG = 1) and 1.3*(N**3) to process each evaluation of the Jacobian (call to FCN with IFLAG = 2). Unless FCN can be evaluated quickly, the timing of HYBRJ will be strongly influenced by the time spent in FCN.

Storage. HYBRJ requires (3*N**2 + 17*N)/2 double precision storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DOGLEG,DPMPAR,ENORM,
QFORM,QRFAC,R1MPYQ,R1UPDT

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MIN0,MOD

8. References.

M. J. D. Powell, A Hybrid Method for Nonlinear Equations. Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, editor. Gordon and Breach, 1970.

9. Example.

The problem is to determine the values of x(1), x(2), ..., x(9), which solve the system of tridiagonal equations

```
(3-2*x(1))*x(1)          -2*x(2)                    = -1
        -x(i-1) + (3-2*x(i))*x(i)          -2*x(i+1) = -1, i=2-8
                            -x(8) + (3-2*x(9))*x(9) = -1
```

```
C     **********
C
C     DRIVER FOR HYBRJ EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER J,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV,LR,NWRITE
      DOUBLE PRECISION XTOL,FACTOR,FNORM
      DOUBLE PRECISION X(9),FVEC(9),FJAC(9,9),DIAG(9),R(45),QTF(9),
     *                 WA1(9),WA2(9),WA3(9),WA4(9)
      DOUBLE PRECISION ENORM,DPMPAR
      EXTERNAL FCN
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      N = 9
C
C     THE FOLLOWING STARTING VALUES PROVIDE A ROUGH SOLUTION.
C
      DO 10 J = 1, 9
         X(J) = -1.D0
   10    CONTINUE
C
      LDFJAC = 9
      LR = 45
C
C     SET XTOL TO THE SQUARE ROOT OF THE MACHINE PRECISION.
C     UNLESS HIGH PRECISION SOLUTIONS ARE REQUIRED,
C     THIS IS THE RECOMMENDED SETTING.
C
      XTOL = DSQRT(DPMPAR(1))
C
      MAXFEV = 1000
      MODE = 2
      DO 20 J = 1, 9
         DIAG(J) = 1.D0
   20    CONTINUE
      FACTOR = 1.D2
      NPRINT = 0
C
      CALL HYBRJ(FCN,N,X,FVEC,FJAC,LDFJAC,XTOL,MAXFEV,DIAG,
     *           MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,R,LR,QTF,
     *           WA1,WA2,WA3,WA4)
      FNORM = ENORM(N,FVEC)
      WRITE (NWRITE,1000) FNORM,NFEV,NJEV,INFO,(X(J),J=1,N)
```

```
      STOP
 1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
     *        5X,31H NUMBER OF FUNCTION EVALUATIONS,I10 //
     *        5X,31H NUMBER OF JACOBIAN EVALUATIONS,I10 //
     *        5X,15H EXIT PARAMETER,16X,I10 //
     *        5X,27H FINAL APPROXIMATE SOLUTION // (5X,3D15.7))
C
C     LAST CARD OF DRIVER FOR HYBRJ EXAMPLE.
C
      END
      SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
      INTEGER N,LDFJAC,IFLAG
      DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)
C
C     SUBROUTINE FCN FOR HYBRJ EXAMPLE.
C
      INTEGER J,K
      DOUBLE PRECISION ONE,TEMP,TEMP1,TEMP2,THREE,TWO,ZERO
      DATA ZERO,ONE,TWO,THREE,FOUR /0.D0,1.D0,2.D0,3.D0,4.D0/
C
      IF (IFLAG .NE. 0) GO TO 5
C
C     INSERT PRINT STATEMENTS HERE WHEN NPRINT IS POSITIVE.
C
      RETURN
    5 CONTINUE
      IF (IFLAG .EQ. 2) GO TO 20
      DO 10 K = 1, N
         TEMP = (THREE - TWO*X(K))*X(K)
         TEMP1 = ZERO
         IF (K .NE. 1) TEMP1 = X(K-1)
         TEMP2 = ZERO
         IF (K .NE. N) TEMP2 = X(K+1)
         FVEC(K) = TEMP - TEMP1 - TWO*TEMP2 + ONE
   10    CONTINUE
      GO TO 50
   20 CONTINUE
      DO 40 K = 1, N
         DO 30 J = 1, N
            FJAC(K,J) = ZERO
   30       CONTINUE
         FJAC(K,K) = THREE - FOUR*X(K)
         IF (K .NE. 1) FJAC(K,K-1) = -ONE
         IF (K .NE. N) FJAC(K,K+1) = -TWO
   40    CONTINUE
   50 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

Results obtained with different compilers or machines may be slightly different.

```
FINAL L2 NORM OF THE RESIDUALS  0.1192636D-07

NUMBER OF FUNCTION EVALUATIONS        11

NUMBER OF JACOBIAN EVALUATIONS         1

EXIT PARAMETER                         1

FINAL APPROXIMATE SOLUTION

-0.5706545D+00 -0.6816283D+00 -0.7017325D+00
-0.7042129D+00 -0.7013690D+00 -0.6918656D+00
-0.6657920D+00 -0.5960342D+00 -0.4164121D+00
```

Documentation for MINPACK subroutine LMDER1

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of LMDER1 is to minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. This is done by using the more general least-squares solver LMDER. The user must provide a subroutine which calculates the functions and the Jacobian.

2. Subroutine and type statements.

```
      SUBROUTINE LMDER1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,
     *                  INFO,IPVT,WA,LWA)
      INTEGER M,N,LDFJAC,INFO,LWA
      INTEGER IPVT(N)
      DOUBLE PRECISION TOL
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),WA(LWA)
      EXTERNAL FCN
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to LMDER1 and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from LMDER1.

FCN is the name of the user-supplied subroutine which calculates the functions and the Jacobian. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
INTEGER M,N,LDFJAC,IFLAG
DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)
----------
IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
RETURN THIS VECTOR IN FVEC.  DO NOT ALTER FJAC.
IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND
RETURN THIS MATRIX IN FJAC.  DO NOT ALTER FVEC.
----------
RETURN
END
```

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of LMDER1. In this case set IFLAG to a negative integer.

M is a positive integer input variable set to the number of functions.

N is a positive integer input variable set to the number of variables. N must not exceed M.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length M which contains the functions evaluated at the output X.

FJAC is an output M by N array. The upper N by N submatrix of FJAC contains an upper triangular matrix R with diagonal elements of nonincreasing magnitude such that

$$P^{T}*(JAC^{T}*JAC)*P = R^{T}*R,$$

where P is a permutation matrix and JAC is the final calculated Jacobian. Column j of P is column IPVT(j) (see below) of the identity matrix. The lower trapezoidal part of FJAC contains information generated during the computation of R.

LDFJAC is a positive integer input variable not less than M which specifies the leading dimension of the array FJAC.

TOL is a nonnegative input variable. Termination occurs when the algorithm estimates either that the relative error in the sum of squares is at most TOL or that the relative error between X and the solution is at most TOL. Section 4 contains more details about TOL.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0 Improper input parameters.

INFO = 1 Algorithm estimates that the relative error in the sum of squares is at most TOL.

INFO = 2 Algorithm estimates that the relative error between X and the solution is at most TOL.

INFO = 3 Conditions for INFO = 1 and INFO = 2 both hold.

INFO = 4 FVEC is orthogonal to the columns of the Jacobian to machine precision.

INFO = 5 Number of calls to FCN with IFLAG = 1 has reached 100*(N+1).

INFO = 6 TOL is too small. No further reduction in the sum of squares is possible.

INFO = 7 TOL is too small. No further improvement in the approximate solution X is possible.

Sections 4 and 5 contain more details about INFO.

IPVT is an integer output array of length N. IPVT defines a permutation matrix P such that JAC*P = Q*R, where JAC is the final calculated Jacobian, Q is orthogonal (not stored), and R is upper triangular with diagonal elements of nonincreasing magnitude. Column j of P is column IPVT(j) of the identity matrix.

WA is a work array of length LWA.

LWA is a positive integer input variable not less than 5*N+M.

4. Successful completion.

The accuracy of LMDER1 is controlled by the convergence parameter TOL. This parameter is used in tests which make three types of comparisons between the approximation X and a solution XSOL. LMDER1 terminates when any of the tests is satisfied. If TOL is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then LMDER1 only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible. Unless high precision solutions are required, the recommended value for TOL is the square root of the machine precision.

The tests assume that the functions and the Jacobian are coded consistently, and that the functions are reasonably well behaved. If these conditions are not satisfied, then LMDER1 may incorrectly indicate convergence. The coding of the Jacobian can be checked by the MINPACK subroutine CHKDER. If the Jacobian is coded correctly, then the validity of the answer can be checked, for example, by rerunning LMDER1 with a tighter tolerance.

First convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z, then this test attempts to guarantee that

```
ENORM(FVEC) .LE. (1+TOL)*ENORM(FVECS),
```

where FVECS denotes the functions evaluated at XSOL. If this condition is satisfied with TOL = 10**(-K), then the final residual norm ENORM(FVEC) has K significant decimal digits and INFO is set to 1 (or to 3 if the second test is also satisfied).

Second convergence test. If D is a diagonal matrix (implicitly generated by LMDER1) whose entries contain scale factors for the variables, then this test attempts to guarantee that

ENORM(D*(X-XSOL)) .LE. TOL*ENORM(D*XSOL).

If this condition is satisfied with TOL = 10**(-K), then the larger components of D*X have K significant decimal digits and INFO is set to 2 (or to 3 if the first test is also satisfied). There is a danger that the smaller components of D*X may have large relative errors, but the choice of D is such that the accuracy of the components of X is usually related to their sensitivity.

Third convergence test. This test is satisfied when FVEC is orthogonal to the columns of the Jacobian to machine precision. There is no clear relationship between this test and the accuracy of LMDER1, and furthermore, the test is equally well satisfied at other critical points, namely maximizers and saddle points. Therefore, termination caused by this test (INFO = 4) should be examined carefully.

5. Unsuccessful completion.

Unsuccessful termination of LMDER1 can be due to improper input parameters, arithmetic interrupts, or an excessive number of function evaluations.

Improper input parameters. INFO is set to 0 if N .LE. 0, or M .LT. N, or LDFJAC .LT. M, or TOL .LT. 0.D0, or LWA .LT. 5*N+M.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by LMDER1. In this case, it may be possible to remedy the situation by not evaluating the functions here, but instead setting the components of FVEC to numbers that exceed those in the initial FVEC, thereby indirectly reducing the step length. The step length can be more directly controlled by using instead LMDER, which includes in its calling sequence the step-length- governing parameter FACTOR.

Excessive number of function evaluations. If the number of calls to FCN with IFLAG = 1 reaches 100*(N+1), then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and INFO is set to 5. In this case, it may be helpful to restart LMDER1, thereby forcing it to disregard old (and possibly harmful) information.

6. Characteristics of the algorithm.

LMDER1 is a modification of the Levenberg-Marquardt algorithm. Two of its main characteristics involve the proper use of implicitly scaled variables and an optimal choice for the correction. The use of implicitly scaled variables achieves scale invariance of LMDER1 and limits the size of the correction in any direction where the functions are changing rapidly. The optimal choice of the correction guarantees (under reasonable conditions) global convergence from starting points far from the solution and a fast rate of convergence for problems with small residuals.

Timing. The time required by LMDER1 to solve a given problem depends on M and N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by LMDER1 is about N**3 to process each evaluation of the functions (call to FCN with IFLAG = 1) and M*(N**2) to process each evaluation of the Jacobian (call to FCN with IFLAG = 2). Unless FCN can be evaluated quickly, the timing of LMDER1 will be strongly influenced by the time spent in FCN.

Storage. LMDER1 requires M*N + 2*M + 6*N double precision storage locations and N integer storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DPMPAR,ENORM,LMDER,LMPAR,QRFAC,QRSOLV

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MOD

8. References.

Jorge J. More, The Levenberg-Marquardt Algorithm, Implementation and Theory. Numerical Analysis, G. A. Watson, editor. Lecture Notes in Mathematics 630, Springer-Verlag, 1977.

9. Example.

The problem is to determine the values of x(1), x(2), and x(3) which provide the best fit (in the least squares sense) of

x(1) + u(i)/(v(i)*x(2) + w(i)*x(3)),   i = 1, 15

to the data

y = (0.14,0.18,0.22,0.25,0.29,0.32,0.35,0.39,
     0.37,0.58,0.73,0.96,1.34,2.10,4.39),

where u(i) = i, v(i) = 16 - i, and w(i) = min(u(i),v(i)). The i-th component of FVEC is thus defined by

y(i) - (x(1) + u(i)/(v(i)*x(2) + w(i)*x(3))).

```
C     **********
C
C     DRIVER FOR LMDER1 EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER J,M,N,LDFJAC,INFO,LWA,NWRITE
      INTEGER IPVT(3)
      DOUBLE PRECISION TOL,FNORM
      DOUBLE PRECISION X(3),FVEC(15),FJAC(15,3),WA(30)
      DOUBLE PRECISION ENORM,DPMPAR
      EXTERNAL FCN
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      M = 15
      N = 3
C
C     THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
      X(1) = 1.D0
      X(2) = 1.D0
      X(3) = 1.D0
C
      LDFJAC = 15
      LWA = 30
C
C     SET TOL TO THE SQUARE ROOT OF THE MACHINE PRECISION.
C     UNLESS HIGH PRECISION SOLUTIONS ARE REQUIRED,
C     THIS IS THE RECOMMENDED SETTING.
C
      TOL = DSQRT(DPMPAR(1))
C
      CALL LMDER1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,
     *            INFO,IPVT,WA,LWA)
      FNORM = ENORM(M,FVEC)
      WRITE (NWRITE,1000) FNORM,INFO,(X(J),J=1,N)
      STOP
 1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
     *        5X,15H EXIT PARAMETER,16X,I10 //
     *        5X,27H FINAL APPROXIMATE SOLUTION // 5X,3D15.7)
C
C     LAST CARD OF DRIVER FOR LMDER1 EXAMPLE.
C
```

```
      END
      SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
      INTEGER M,N,LDFJAC,IFLAG
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)
C
C     SUBROUTINE FCN FOR LMDER1 EXAMPLE.
C
      INTEGER I
      DOUBLE PRECISION TMP1,TMP2,TMP3,TMP4
      DOUBLE PRECISION Y(15)
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),
     *     Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15)
     *     /1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,
     *      3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/
C
      IF (IFLAG .EQ. 2) GO TO 20
      DO 10 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         FVEC(I) = Y(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
   10    CONTINUE
      GO TO 40
   20 CONTINUE
      DO 30 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         TMP4 = (X(2)*TMP2 + X(3)*TMP3)**2
         FJAC(I,1) = -1.D0
         FJAC(I,2) = TMP1*TMP2/TMP4
         FJAC(I,3) = TMP1*TMP3/TMP4
   30    CONTINUE
   40 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

Results obtained with different compilers or machines
may be slightly different.

```
FINAL L2 NORM OF THE RESIDUALS  0.9063596D-01

EXIT PARAMETER                           1

FINAL APPROXIMATE SOLUTION

 0.8241058D-01  0.1133037D+01  0.2343695D+01
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Documentation for MINPACK subroutine LMDER

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of LMDER is to minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. The user must provide a subroutine which calculates the functions and the Jacobian.

2. Subroutine and type statements.

```
      SUBROUTINE LMDER(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
     *                 MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,
     *                 IPVT,QTF,WA1,WA2,WA3,WA4)
      INTEGER M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV
      INTEGER IPVT(N)
      DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),DIAG(N),QTF(N),
     *                 WA1(N),WA2(N),WA3(N),WA4(M)
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to LMDER and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from LMDER.

FCN is the name of the user-supplied subroutine which calculates the functions and the Jacobian. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
INTEGER M,N,LDFJAC,IFLAG
DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)
----------
IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
RETURN THIS VECTOR IN FVEC.  DO NOT ALTER FJAC.
IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND
RETURN THIS MATRIX IN FJAC.  DO NOT ALTER FVEC.
----------
RETURN
END
```

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of LMDER. In this case set IFLAG to a negative integer.

M is a positive integer input variable set to the number of functions.

N is a positive integer input variable set to the number of variables. N must not exceed M.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length M which contains the functions evaluated at the output X.

FJAC is an output M by N array. The upper N by N submatrix of FJAC contains an upper triangular matrix R with diagonal elements of nonincreasing magnitude such that

$$P^T*(JAC^T*JAC)*P = R^T*R,$$

where P is a permutation matrix and JAC is the final calculated Jacobian. Column j of P is column IPVT(j) (see below) of the identity matrix. The lower trapezoidal part of FJAC contains information generated during the computation of R.

LDFJAC is a positive integer input variable not less than M which specifies the leading dimension of the array FJAC.

FTOL is a nonnegative input variable. Termination occurs when both the actual and predicted relative reductions in the sum of squares are at most FTOL. Therefore, FTOL measures the relative error desired in the sum of squares. Section 4 contains more details about FTOL.

XTOL is a nonnegative input variable. Termination occurs when the relative error between two consecutive iterates is at most XTOL. Therefore, XTOL measures the relative error desired in the approximate solution. Section 4 contains more details about XTOL.

GTOL is a nonnegative input variable. Termination occurs when the cosine of the angle between FVEC and any column of the Jacobian is at most GTOL in absolute value. Therefore, GTOL measures the orthogonality desired between the function vector and the columns of the Jacobian. Section 4 contains more details about GTOL.

MAXFEV is a positive integer input variable. Termination occurs when the number of calls to FCN with IFLAG = 1 has reached MAXFEV.

DIAG is an array of length N. If MODE = 1 (see below), DIAG is internally set. If MODE = 2, DIAG must contain positive entries that serve as multiplicative scale factors for the variables.

MODE is an integer input variable. If MODE = 1, the variables will be scaled internally. If MODE = 2, the scaling is specified by the input DIAG. Other values of MODE are equivalent to MODE = 1.

FACTOR is a positive input variable used in determining the initial step bound. This bound is set to the product of FACTOR and the Euclidean norm of DIAG*X if nonzero, or else to FACTOR itself. In most cases FACTOR should lie in the interval (.1,100.). 100. is a generally recommended value.

NPRINT is an integer input variable that enables controlled printing of iterates if it is positive. In this case, FCN is called with IFLAG = 0 at the beginning of the first iteration and every NPRINT iterations thereafter and immediately prior to return, with X, FVEC, and FJAC available for printing. FVEC and FJAC should not be altered. If NPRINT is not positive, no special calls of FCN with IFLAG = 0 are made.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0 Improper input parameters.

INFO = 1 Both actual and predicted relative reductions in the sum of squares are at most FTOL.

INFO = 2 Relative error between two consecutive iterates is at most XTOL.

INFO = 3 Conditions for INFO = 1 and INFO = 2 both hold.

INFO = 4 The cosine of the angle between FVEC and any column of the Jacobian is at most GTOL in absolute value.

INFO = 5 Number of calls to FCN with IFLAG = 1 has reached MAXFEV.

INFO = 6 FTOL is too small. No further reduction in the sum of squares is possible.

INFO = 7 XTOL is too small. No further improvement in the approximate solution X is possible.

INFO = 8 GTOL is too small. FVEC is orthogonal to the columns of the Jacobian to machine precision.

Sections 4 and 5 contain more details about INFO.

NFEV is an integer output variable set to the number of calls to FCN with IFLAG = 1.

NJEV is an integer output variable set to the number of calls to FCN with IFLAG = 2.

IPVT is an integer output array of length N. IPVT defines a permutation matrix P such that JAC*P = Q*R, where JAC is the final calculated Jacobian, Q is orthogonal (not stored), and R is upper triangular with diagonal elements of nonincreasing magnitude. Column j of P is column IPVT(j) of the identity matrix.

QTF is an output array of length N which contains the first N elements of the vector (Q transpose)*FVEC.

WA1, WA2, and WA3 are work arrays of length N.

WA4 is a work array of length M.

4. Successful completion.

The accuracy of LMDER is controlled by the convergence parameters FTOL, XTOL, and GTOL. These parameters are used in tests which make three types of comparisons between the approximation X and a solution XSOL. LMDER terminates when any of the tests is satisfied. If any of the convergence parameters is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then LMDER only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible.

The tests assume that the functions and the Jacobian are coded consistently, and that the functions are reasonably well behaved. If these conditions are not satisfied, then LMDER may incorrectly indicate convergence. The coding of the Jacobian can be checked by the MINPACK subroutine CHKDER. If the Jacobian is coded correctly, then the validity of the answer can be checked, for example, by rerunning LMDER with tighter tolerances.

First convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z, then this test attempts to guarantee that

```
ENORM(FVEC) .LE. (1+FTOL)*ENORM(FVECS),
```

where FVECS denotes the functions evaluated at XSOL. If this condition is satisfied with FTOL = 10**(-K), then the final residual norm ENORM(FVEC) has K significant decimal digits and INFO is set to 1 (or to 3 if the second test is also satisfied). Unless high precision solutions are required, the recommended value for FTOL is the square root of the machine precision.

Second convergence test. If D is the diagonal matrix whose entries are defined by the array DIAG, then this test attempts to guarantee that

ENORM(D*(X-XSOL)) .LE. XTOL*ENORM(D*XSOL).

If this condition is satisfied with XTOL = 10**(-K), then the larger components of D*X have K significant decimal digits and INFO is set to 2 (or to 3 if the first test is also satisfied). There is a danger that the smaller components of D*X may have large relative errors, but if MODE = 1, then the accuracy of the components of X is usually related to their sensitivity. Unless high precision solutions are required, the recommended value for XTOL is the square root of the machine precision.

Third convergence test. This test is satisfied when the cosine of the angle between FVEC and any column of the Jacobian at X is at most GTOL in absolute value. There is no clear relationship between this test and the accuracy of LMDER, and furthermore, the test is equally well satisfied at other critical points, namely maximizers and saddle points. Therefore, termination caused by this test (INFO = 4) should be examined carefully. The recommended value for GTOL is zero.

5. Unsuccessful completion.

Unsuccessful termination of LMDER can be due to improper input parameters, arithmetic interrupts, or an excessive number of function evaluations.

Improper input parameters. INFO is set to 0 if N .LE. 0, or M .LT. N, or LDFJAC .LT. M, or FTOL .LT. 0.D0, or XTOL .LT. 0.D0, or GTOL .LT. 0.D0, or MAXFEV .LE. 0, or FACTOR .LE. 0.D0.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by LMDER. In this case, it may be possible to remedy the situation by rerunning LMDER with a smaller value of FACTOR.

Excessive number of function evaluations. A reasonable value for MAXFEV is 100*(N+1). If the number of calls to FCN with IFLAG = 1 reaches MAXFEV, then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and INFO is set to 5. In this case, it may be helpful to restart LMDER with MODE set to 1.

6. Characteristics of the algorithm.

LMDER is a modification of the Levenberg-Marquardt algorithm.

Two of its main characteristics involve the proper use of implicitly scaled variables (if MODE = 1) and an optimal choice for the correction. The use of implicitly scaled variables achieves scale invariance of LMDER and limits the size of the correction in any direction where the functions are changing rapidly. The optimal choice of the correction guarantees (under reasonable conditions) global convergence from starting points far from the solution and a fast rate of convergence for problems with small residuals.

Timing. The time required by LMDER to solve a given problem depends on M and N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by LMDER is about N**3 to process each evaluation of the functions (call to FCN with IFLAG = 1) and M*(N**2) to process each evaluation of the Jacobian (call to FCN with IFLAG = 2). Unless FCN can be evaluated quickly, the timing of LMDER will be strongly influenced by the time spent in FCN.

Storage. LMDER requires M*N + 2*M + 6*N double precision storage locations and N integer storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DPMPAR,ENORM,LMPAR,QRFAC,QRSOLV

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MOD

8. References.

Jorge J. More, The Levenberg-Marquardt Algorithm, Implementation and Theory. Numerical Analysis, G. A. Watson, editor. Lecture Notes in Mathematics 630, Springer-Verlag, 1977.

9. Example.

The problem is to determine the values of x(1), x(2), and x(3) which provide the best fit (in the least squares sense) of

```
x(1) + u(i)/(v(i)*x(2) + w(i)*x(3)),  i = 1, 15
```

to the data

```
y = (0.14,0.18,0.22,0.25,0.29,0.32,0.35,0.39,
     0.37,0.58,0.73,0.96,1.34,2.10,4.39),
```

```
      where u(i) = i, v(i) = 16 - i, and w(i) = min(u(i),v(i)).  The
      i-th component of FVEC is thus defined by

            y(i) - (x(1) + u(i)/(v(i)*x(2) + w(i)*x(3))).

C     **********
C
C     DRIVER FOR LMDER EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER J,M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV,NWRITE
      INTEGER IPVT(3)
      DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR,FNORM
      DOUBLE PRECISION X(3),FVEC(15),FJAC(15,3),DIAG(3),QTF(3),
     *                 WA1(3),WA2(3),WA3(3),WA4(15)
      DOUBLE PRECISION ENORM,DPMPAR
      EXTERNAL FCN
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      M = 15
      N = 3
C
C     THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
      X(1) = 1.D0
      X(2) = 1.D0
      X(3) = 1.D0
C
      LDFJAC = 15
C
C     SET FTOL AND XTOL TO THE SQUARE ROOT OF THE MACHINE PRECISION
C     AND GTOL TO ZERO. UNLESS HIGH PRECISION SOLUTIONS ARE
C     REQUIRED, THESE ARE THE RECOMMENDED SETTINGS.
C
      FTOL = DSQRT(DPMPAR(1))
      XTOL = DSQRT(DPMPAR(1))
      GTOL = 0.D0
C
      MAXFEV = 400
      MODE = 1
      FACTOR = 1.D2
      NPRINT = 0
C
      CALL LMDER(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
     *           MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,
     *           IPVT,QTF,WA1,WA2,WA3,WA4)
      FNORM = ENORM(M,FVEC)
      WRITE (NWRITE,1000) FNORM,NFEV,NJEV,INFO,(X(J),J=1,N)
      STOP
 1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
```

```
     *         5X,31H NUMBER OF FUNCTION EVALUATIONS,I10 //
     *         5X,31H NUMBER OF JACOBIAN EVALUATIONS,I10 //
     *         5X,15H EXIT PARAMETER,16X,I10 //
     *         5X,27H FINAL APPROXIMATE SOLUTION // 5X,3D15.7)
C
C     LAST CARD OF DRIVER FOR LMDER EXAMPLE.
C
      END
      SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
      INTEGER M,N,LDFJAC,IFLAG
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)
C
C     SUBROUTINE FCN FOR LMDER EXAMPLE.
C
      INTEGER I
      DOUBLE PRECISION TMP1,TMP2,TMP3,TMP4
      DOUBLE PRECISION Y(15)
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),
     *     Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15)
     *     /1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,
     *      3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/
C
      IF (IFLAG .NE. 0) GO TO 5
C
C     INSERT PRINT STATEMENTS HERE WHEN NPRINT IS POSITIVE.
C
      RETURN
    5 CONTINUE
      IF (IFLAG .EQ. 2) GO TO 20
      DO 10 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         FVEC(I) = Y(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
   10    CONTINUE
      GO TO 40
   20 CONTINUE
      DO 30 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         TMP4 = (X(2)*TMP2 + X(3)*TMP3)**2
         FJAC(I,1) = -1.D0
         FJAC(I,2) = TMP1*TMP2/TMP4
         FJAC(I,3) = TMP1*TMP3/TMP4
   30    CONTINUE
   40 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

Results obtained with different compilers or machines may be slightly different.

FINAL L2 NORM OF THE RESIDUALS 0.9063596D-01

NUMBER OF FUNCTION EVALUATIONS 6

NUMBER OF JACOBIAN EVALUATIONS 5

EXIT PARAMETER 1

FINAL APPROXIMATE SOLUTION

0.8241058D-01 0.1133037D+01 0.2343695D+01

Documentation for MINPACK subroutine LMSTR1

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of LMSTR1 is to minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm which uses minimal storage. This is done by using the more general least-squares solver LMSTR. The user must provide a subroutine which calculates the functions and the rows of the Jacobian.

2. Subroutine and type statements.

```
      SUBROUTINE LMSTR1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,
     *                  INFO,IPVT,WA,LWA)
      INTEGER M,N,LDFJAC,INFO,LWA
      INTEGER IPVT(N)
      DOUBLE PRECISION TOL
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),WA(LWA)
      EXTERNAL FCN
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to LMSTR1 and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from LMSTR1.

FCN is the name of the user-supplied subroutine which calculates the functions and the rows of the Jacobian. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)
INTEGER M,N,IFLAG
DOUBLE PRECISION X(N),FVEC(M),FJROW(N)
----------
IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
RETURN THIS VECTOR IN FVEC.
IF IFLAG = I CALCULATE THE (I-1)-ST ROW OF THE
JACOBIAN AT X AND RETURN THIS VECTOR IN FJROW.
----------
RETURN
```

END

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of LMSTR1. In this case set IFLAG to a negative integer.

M is a positive integer input variable set to the number of functions.

N is a positive integer input variable set to the number of variables. N must not exceed M.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length M which contains the functions evaluated at the output X.

FJAC is an output N by N array. The upper triangle of FJAC contains an upper triangular matrix R such that

$$P^T*(JAC^T*JAC)*P = R^T*R,$$

where P is a permutation matrix and JAC is the final calculated Jacobian. Column j of P is column IPVT(j) (see below) of the identity matrix. The lower triangular part of FJAC contains information generated during the computation of R.

LDFJAC is a positive integer input variable not less than N which specifies the leading dimension of the array FJAC.

TOL is a nonnegative input variable. Termination occurs when the algorithm estimates either that the relative error in the sum of squares is at most TOL or that the relative error between X and the solution is at most TOL. Section 4 contains more details about TOL.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0 Improper input parameters.

INFO = 1 Algorithm estimates that the relative error in the sum of squares is at most TOL.

INFO = 2 Algorithm estimates that the relative error between X and the solution is at most TOL.

INFO = 3 Conditions for INFO = 1 and INFO = 2 both hold.

INFO = 4 FVEC is orthogonal to the columns of the Jacobian to machine precision.

INFO = 5 Number of calls to FCN with IFLAG = 1 has reached 100*(N+1).

INFO = 6 TOL is too small. No further reduction in the sum of squares is possible.

INFO = 7 TOL is too small. No further improvement in the approximate solution X is possible.

Sections 4 and 5 contain more details about INFO.

IPVT is an integer output array of length N. IPVT defines a permutation matrix P such that JAC*P = Q*R, where JAC is the final calculated Jacobian, Q is orthogonal (not stored), and R is upper triangular. Column j of P is column IPVT(j) of the identity matrix.

WA is a work array of length LWA.

LWA is a positive integer input variable not less than 5*N+M.

4. Successful completion.

The accuracy of LMSTR1 is controlled by the convergence parameter TOL. This parameter is used in tests which make three types of comparisons between the approximation X and a solution XSOL. LMSTR1 terminates when any of the tests is satisfied. If TOL is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then LMSTR1 only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible. Unless high precision solutions are required, the recommended value for TOL is the square root of the machine precision.

The tests assume that the functions and the Jacobian are coded consistently, and that the functions are reasonably well behaved. If these conditions are not satisfied, then LMSTR1 may incorrectly indicate convergence. The coding of the Jacobian can be checked by the MINPACK subroutine CHKDER. If the Jacobian is coded correctly, then the validity of the answer can be checked, for example, by rerunning LMSTR1 with a tighter tolerance.

First convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z, then this test attempts to guarantee that

    ENORM(FVEC) .LE. (1+TOL)*ENORM(FVECS),

where FVECS denotes the functions evaluated at XSOL. If this condition is satisfied with TOL = 10**(-K), then the final residual norm ENORM(FVEC) has K significant decimal digits and

INFO is set to 1 (or to 3 if the second test is also satisfied).

Second convergence test. If D is a diagonal matrix (implicitly generated by LMSTR1) whose entries contain scale factors for the variables, then this test attempts to guarantee that

ENORM(D*(X-XSOL)) .LE. TOL*ENORM(D*XSOL).

If this condition is satisfied with TOL = 10**(-K), then the larger components of D*X have K significant decimal digits and INFO is set to 2 (or to 3 if the first test is also satisfied). There is a danger that the smaller components of D*X may have large relative errors, but the choice of D is such that the accuracy of the components of X is usually related to their sensitivity.

Third convergence test. This test is satisfied when FVEC is orthogonal to the columns of the Jacobian to machine precision. There is no clear relationship between this test and the accuracy of LMSTR1, and furthermore, the test is equally well satisfied at other critical points, namely maximizers and saddle points. Therefore, termination caused by this test (INFO = 4) should be examined carefully.

5. Unsuccessful completion.

Unsuccessful termination of LMSTR1 can be due to improper input parameters, arithmetic interrupts, or an excessive number of function evaluations.

Improper input parameters. INFO is set to 0 if N .LE. 0, or M .LT. N, or LDFJAC .LT. N, or TOL .LT. 0.D0, or LWA .LT. 5*N+M.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by LMSTR1. In this case, it may be possible to remedy the situation by not evaluating the functions here, but instead setting the components of FVEC to numbers that exceed those in the initial FVEC, thereby indirectly reducing the step length. The step length can be more directly controlled by using instead LMSTR, which includes in its calling sequence the step-length- governing parameter FACTOR.

Excessive number of function evaluations. If the number of calls to FCN with IFLAG = 1 reaches 100*(N+1), then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and INFO is set to 5. In this case, it may be helpful to restart LMSTR1, thereby forcing it to disregard old (and possibly harmful) information.

6. Characteristics of the algorithm.

LMSTR1 is a modification of the Levenberg-Marquardt algorithm. Two of its main characteristics involve the proper use of implicitly scaled variables and an optimal choice for the correction. The use of implicitly scaled variables achieves scale invariance of LMSTR1 and limits the size of the correction in any direction where the functions are changing rapidly. The optimal choice of the correction guarantees (under reasonable conditions) global convergence from starting points far from the solution and a fast rate of convergence for problems with small residuals.

Timing. The time required by LMSTR1 to solve a given problem depends on M and N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by LMSTR1 is about N**3 to process each evaluation of the functions (call to FCN with IFLAG = 1) and 1.5*(N**2) to process each row of the Jacobian (call to FCN with IFLAG .GE. 2). Unless FCN can be evaluated quickly, the timing of LMSTR1 will be strongly influenced by the time spent in FCN.

Storage. LMSTR1 requires N**2 + 2*M + 6*N double precision storage locations and N integer storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DPMPAR,ENORM,LMSTR,LMPAR,QRFAC,QRSOLV, RWUPDT

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MOD

8. References.

Jorge J. More, The Levenberg-Marquardt Algorithm, Implementation and Theory. Numerical Analysis, G. A. Watson, editor. Lecture Notes in Mathematics 630, Springer-Verlag, 1977.

9. Example.

The problem is to determine the values of x(1), x(2), and x(3) which provide the best fit (in the least squares sense) of

x(1) + u(i)/(v(i)*x(2) + w(i)*x(3)), i = 1, 15

```
      to the data

            y = (0.14,0.18,0.22,0.25,0.29,0.32,0.35,0.39,
                 0.37,0.58,0.73,0.96,1.34,2.10,4.39),

      where u(i) = i, v(i) = 16 - i, and w(i) = min(u(i),v(i)).  The
      i-th component of FVEC is thus defined by

            y(i) - (x(1) + u(i)/(v(i)*x(2) + w(i)*x(3))).

C     **********
C
C     DRIVER FOR LMSTR1 EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER J,M,N,LDFJAC,INFO,LWA,NWRITE
      INTEGER IPVT(3)
      DOUBLE PRECISION TOL,FNORM
      DOUBLE PRECISION X(3),FVEC(15),FJAC(3,3),WA(30)
      DOUBLE PRECISION ENORM,DPMPAR
      EXTERNAL FCN
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      M = 15
      N = 3
C
C     THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
      X(1) = 1.D0
      X(2) = 1.D0
      X(3) = 1.D0
C
      LDFJAC = 3
      LWA = 30
C
C     SET TOL TO THE SQUARE ROOT OF THE MACHINE PRECISION.
C     UNLESS HIGH PRECISION SOLUTIONS ARE REQUIRED,
C     THIS IS THE RECOMMENDED SETTING.
C
      TOL = DSQRT(DPMPAR(1))
C
      CALL LMSTR1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,
     *            INFO,IPVT,WA,LWA)
      FNORM = ENORM(M,FVEC)
      WRITE (NWRITE,1000) FNORM,INFO,(X(J),J=1,N)
      STOP
 1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
     *        5X,15H EXIT PARAMETER,16X,I10 //
     *        5X,27H FINAL APPROXIMATE SOLUTION // 5X,3D15.7)
C
```

```
C     LAST CARD OF DRIVER FOR LMSTR1 EXAMPLE.

      END
      SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)
      INTEGER M,N,IFLAG
      DOUBLE PRECISION X(N),FVEC(M),FJROW(N)
C
C     SUBROUTINE FCN FOR LMSTR1 EXAMPLE.
C
      INTEGER I
      DOUBLE PRECISION TMP1,TMP2,TMP3,TMP4
      DOUBLE PRECISION Y(15)
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),
     *     Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15)
     *     /1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,
     *      3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/
C
      IF (IFLAG .GE. 2) GO TO 20
      DO 10 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         FVEC(I) = Y(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
   10    CONTINUE
      GO TO 40
   20 CONTINUE
      I = IFLAG - 1
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         TMP4 = (X(2)*TMP2 + X(3)*TMP3)**2
         FJROW(1) = -1.D0
         FJROW(2) = TMP1*TMP2/TMP4
         FJROW(3) = TMP1*TMP3/TMP4
   30    CONTINUE
   40 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

Results obtained with different compilers or machines may be slightly different.

```
FINAL L2 NORM OF THE RESIDUALS  0.9063596D-01

EXIT PARAMETER                             1

FINAL APPROXIMATE SOLUTION

 0.8241058D-01  0.1133037D+01  0.2343695D+01
```

Documentation for MINPACK subroutine LMSTR

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of LMSTR is to minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm which uses minimal storage. The user must provide a subroutine which calculates the functions and the rows of the Jacobian.

2. Subroutine and type statements.

```
 SUBROUTINE LMSTR(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
*                 MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,
*                 IPVT,QTF,WA1,WA2,WA3,WA4)
 INTEGER M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV
 INTEGER IPVT(N)
 DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR
 DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),DIAG(N),QTF(N),
*                 WA1(N),WA2(N),WA3(N),WA4(M)
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to LMSTR and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from LMSTR.

FCN is the name of the user-supplied subroutine which calculates the functions and the rows of the Jacobian. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)
INTEGER M,N,IFLAG
DOUBLE PRECISION X(N),FVEC(M),FJROW(N)
----------
IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
RETURN THIS VECTOR IN FVEC.
IF IFLAG = I CALCULATE THE (I-1)-ST ROW OF THE
JACOBIAN AT X AND RETURN THIS VECTOR IN FJROW.
----------
RETURN
```

END

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of LMSTR. In this case set IFLAG to a negative integer.

M is a positive integer input variable set to the number of functions.

N is a positive integer input variable set to the number of variables. N must not exceed M.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length M which contains the functions evaluated at the output X.

FJAC is an output N by N array. The upper triangle of FJAC contains an upper triangular matrix R such that

$$P^T*(JAC^T*JAC)*P = R^T*R,$$

where P is a permutation matrix and JAC is the final calculated Jacobian. Column j of P is column IPVT(j) (see below) of the identity matrix. The lower triangular part of FJAC contains information generated during the computation of R.

LDFJAC is a positive integer input variable not less than N which specifies the leading dimension of the array FJAC.

FTOL is a nonnegative input variable. Termination occurs when both the actual and predicted relative reductions in the sum of squares are at most FTOL. Therefore, FTOL measures the relative error desired in the sum of squares. Section 4 contains more details about FTOL.

XTOL is a nonnegative input variable. Termination occurs when the relative error between two consecutive iterates is at most XTOL. Therefore, XTOL measures the relative error desired in the approximate solution. Section 4 contains more details about XTOL.

GTOL is a nonnegative input variable. Termination occurs when the cosine of the angle between FVEC and any column of the Jacobian is at most GTOL in absolute value. Therefore, GTOL measures the orthogonality desired between the function vector and the columns of the Jacobian. Section 4 contains more details about GTOL.

MAXFEV is a positive integer input variable. Termination occurs when the number of calls to FCN with IFLAG = 1 has reached

MAXFEV.

DIAG is an array of length N. If MODE = 1 (see below), DIAG is internally set. If MODE = 2, DIAG must contain positive entries that serve as multiplicative scale factors for the variables.

MODE is an integer input variable. If MODE = 1, the variables will be scaled internally. If MODE = 2, the scaling is specified by the input DIAG. Other values of MODE are equivalent to MODE = 1.

FACTOR is a positive input variable used in determining the initial step bound. This bound is set to the product of FACTOR and the Euclidean norm of DIAG*X if nonzero, or else to FACTOR itself. In most cases FACTOR should lie in the interval (.1,100.). 100. is a generally recommended value.

NPRINT is an integer input variable that enables controlled printing of iterates if it is positive. In this case, FCN is called with IFLAG = 0 at the beginning of the first iteration and every NPRINT iterations thereafter and immediately prior to return, with X and FVEC available for printing. If NPRINT is not positive, no special calls of FCN with IFLAG = 0 are made.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0 Improper input parameters.

INFO = 1 Both actual and predicted relative reductions in the sum of squares are at most FTOL.

INFO = 2 Relative error between two consecutive iterates is at most XTOL.

INFO = 3 Conditions for INFO = 1 and INFO = 2 both hold.

INFO = 4 The cosine of the angle between FVEC and any column of the Jacobian is at most GTOL in absolute value.

INFO = 5 Number of calls to FCN with IFLAG = 1 has reached MAXFEV.

INFO = 6 FTOL is too small. No further reduction in the sum of squares is possible.

INFO = 7 XTOL is too small. No further improvement in the approximate solution X is possible.

INFO = 8 GTOL is too small. FVEC is orthogonal to the columns of the Jacobian to machine precision.

Sections 4 and 5 contain more details about INFO.

NFEV is an integer output variable set to the number of calls to FCN with IFLAG = 1.

NJEV is an integer output variable set to the number of calls to FCN with IFLAG = 2.

IPVT is an integer output array of length N. IPVT defines a permutation matrix P such that JAC*P = Q*R, where JAC is the final calculated Jacobian, Q is orthogonal (not stored), and R is upper triangular. Column j of P is column IPVT(j) of the identity matrix.

QTF is an output array of length N which contains the first N elements of the vector (Q transpose)*FVEC.

WA1, WA2, and WA3 are work arrays of length N.

WA4 is a work array of length M.

4. Successful completion.

The accuracy of LMSTR is controlled by the convergence parameters FTOL, XTOL, and GTOL. These parameters are used in tests which make three types of comparisons between the approximation X and a solution XSOL. LMSTR terminates when any of the tests is satisfied. If any of the convergence parameters is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then LMSTR only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible.

The tests assume that the functions and the Jacobian are coded consistently, and that the functions are reasonably well behaved. If these conditions are not satisfied, then LMSTR may incorrectly indicate convergence. The coding of the Jacobian can be checked by the MINPACK subroutine CHKDER. If the Jacobian is coded correctly, then the validity of the answer can be checked, for example, by rerunning LMSTR with tighter tolerances.

First convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z, then this test attempts to guarantee that

```
ENORM(FVEC) .LE. (1+FTOL)*ENORM(FVECS),
```

where FVECS denotes the functions evaluated at XSOL. If this condition is satisfied with FTOL = 10**(-K), then the final residual norm ENORM(FVEC) has K significant decimal digits and INFO is set to 1 (or to 3 if the second test is also satisfied). Unless high precision solutions are required, the recommended value for FTOL is the square root of the machine precision.

Second convergence test. If D is the diagonal matrix whose entries are defined by the array DIAG, then this test attempts to guarantee that

ENORM(D*(X-XSOL)) .LE. XTOL*ENORM(D*XSOL).

If this condition is satisfied with XTOL = 10**(-K), then the larger components of D*X have K significant decimal digits and INFO is set to 2 (or to 3 if the first test is also satisfied). There is a danger that the smaller components of D*X may have large relative errors, but if MODE = 1, then the accuracy of the components of X is usually related to their sensitivity. Unless high precision solutions are required, the recommended value for XTOL is the square root of the machine precision.

Third convergence test. This test is satisfied when the cosine of the angle between FVEC and any column of the Jacobian at X is at most GTOL in absolute value. There is no clear relationship between this test and the accuracy of LMSTR, and furthermore, the test is equally well satisfied at other critical points, namely maximizers and saddle points. Therefore, termination caused by this test (INFO = 4) should be examined carefully. The recommended value for GTOL is zero.

5. Unsuccessful completion.

Unsuccessful termination of LMSTR can be due to improper input parameters, arithmetic interrupts, or an excessive number of function evaluations.

Improper input parameters. INFO is set to 0 if N .LE. 0, or M .LT. N, or LDFJAC .LT. N, or FTOL .LT. 0.D0, or XTOL .LT. 0.D0, or GTOL .LT. 0.D0, or MAXFEV .LE. 0, or FACTOR .LE. 0.D0.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by LMSTR. In this case, it may be possible to remedy the situation by rerunning LMSTR with a smaller value of FACTOR.

Excessive number of function evaluations. A reasonable value for MAXFEV is 100*(N+1). If the number of calls to FCN with IFLAG = 1 reaches MAXFEV, then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and INFO is set to 5. In this case, it may be helpful to restart LMSTR with MODE set to 1.

6. Characteristics of the algorithm.

LMSTR is a modification of the Levenberg-Marquardt algorithm. Two of its main characteristics involve the proper use of implicitly scaled variables (if MODE = 1) and an optimal choice for the correction. The use of implicitly scaled variables achieves scale invariance of LMSTR and limits the size of the correction in any direction where the functions are changing rapidly. The optimal choice of the correction guarantees (under reasonable conditions) global convergence from starting points far from the solution and a fast rate of convergence for problems with small residuals.

Timing. The time required by LMSTR to solve a given problem depends on M and N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by LMSTR is about N**3 to process each evaluation of the functions (call to FCN with IFLAG = 1) and 1.5*(N**2) to process each row of the Jacobian (call to FCN with IFLAG .GE. 2). Unless FCN can be evaluated quickly, the timing of LMSTR will be strongly influenced by the time spent in FCN.

Storage. LMSTR requires N**2 + 2*M + 6*N double precision storage locations and N integer storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DPMPAR,ENORM,LMPAR,QRFAC,QRSOLV,RWUPDT

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MOD

8. References.

Jorge J. More, The Levenberg-Marquardt Algorithm, Implementation and Theory. Numerical Analysis, G. A. Watson, editor. Lecture Notes in Mathematics 630, Springer-Verlag, 1977.

9. Example.

The problem is to determine the values of x(1), x(2), and x(3) which provide the best fit (in the least squares sense) of

```
x(1) + u(i)/(v(i)*x(2) + w(i)*x(3)),  i = 1, 15
```

to the data

```
y = (0.14,0.18,0.22,0.25,0.29,0.32,0.35,0.39,
     0.37,0.58,0.73,0.96,1.34,2.10,4.39),
```

where u(i) = i, v(i) = 16 - i, and w(i) = min(u(i),v(i)). The i-th component of FVEC is thus defined by

y(i) - (x(1) + u(i)/(v(i)*x(2) + w(i)*x(3))).

```
C     **********
C
C     DRIVER FOR LMSTR EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER J,M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV,NWRITE
      INTEGER IPVT(3)
      DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR,FNORM
      DOUBLE PRECISION X(3),FVEC(15),FJAC(3,3),DIAG(3),QTF(3),
     *                 WA1(3),WA2(3),WA3(3),WA4(15)
      DOUBLE PRECISION ENORM,DPMPAR
      EXTERNAL FCN
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      M = 15
      N = 3
C
C     THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
      X(1) = 1.D0
      X(2) = 1.D0
      X(3) = 1.D0
C
      LDFJAC = 3
C
C     SET FTOL AND XTOL TO THE SQUARE ROOT OF THE MACHINE PRECISION
C     AND GTOL TO ZERO. UNLESS HIGH PRECISION SOLUTIONS ARE
C     REQUIRED, THESE ARE THE RECOMMENDED SETTINGS.
C
      FTOL = DSQRT(DPMPAR(1))
      XTOL = DSQRT(DPMPAR(1))
      GTOL = 0.D0
C
      MAXFEV = 400
      MODE = 1
      FACTOR = 1.D2
      NPRINT = 0
C
      CALL LMSTR(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
     *           MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,
     *           IPVT,QTF,WA1,WA2,WA3,WA4)
      FNORM = ENORM(M,FVEC)
      WRITE (NWRITE,1000) FNORM,NFEV,NJEV,INFO,(X(J),J=1,N)
      STOP
 1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
```

```
     *          5X,31H NUMBER OF FUNCTION EVALUATIONS,I10 //
     *          5X,31H NUMBER OF JACOBIAN EVALUATIONS,I10 //
     *          5X,15H EXIT PARAMETER,16X,I10 //
     *          5X,27H FINAL APPROXIMATE SOLUTION // 5X,3D15.7)
C
C     LAST CARD OF DRIVER FOR LMSTR EXAMPLE.
C
      END
      SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)
      INTEGER M,N,IFLAG
      DOUBLE PRECISION X(N),FVEC(M),FJROW(N)
C
C     SUBROUTINE FCN FOR LMSTR EXAMPLE.
C
      INTEGER I
      DOUBLE PRECISION TMP1,TMP2,TMP3,TMP4
      DOUBLE PRECISION Y(15)
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),
     *     Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15)
     *     /1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,
     *      3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/
C
      IF (IFLAG .NE. 0) GO TO 5
C
C     INSERT PRINT STATEMENTS HERE WHEN NPRINT IS POSITIVE.
C
      RETURN
    5 CONTINUE
      IF (IFLAG .GE. 2) GO TO 20
      DO 10 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         FVEC(I) = Y(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
   10    CONTINUE
      GO TO 40
   20 CONTINUE
      I = IFLAG - 1
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         TMP4 = (X(2)*TMP2 + X(3)*TMP3)**2
         FJROW(1) = -1.D0
         FJROW(2) = TMP1*TMP2/TMP4
         FJROW(3) = TMP1*TMP3/TMP4
   30    CONTINUE
   40 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

Results obtained with different compilers or machines may be slightly different.

```
FINAL L2 NORM OF THE RESIDUALS  0.9063596D-01

NUMBER OF FUNCTION EVALUATIONS         6

NUMBER OF JACOBIAN EVALUATIONS         5

EXIT PARAMETER                         1

FINAL APPROXIMATE SOLUTION

 0.8241058D-01  0.1133037D+01  0.2343695D+01
```

THIS PAGE WAS INTENTIONALLY LEFT BLANK

Documentation for MINPACK subroutine LMDIF1

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of LMDIF1 is to minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. This is done by using the more general least-squares solver LMDIF. The user must provide a subroutine which calculates the functions. The Jacobian is then calculated by a forward-difference approximation.

2. Subroutine and type statements.

```
SUBROUTINE LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)
INTEGER M,N,INFO,LWA
INTEGER IWA(N)
DOUBLE PRECISION TOL
DOUBLE PRECISION X(N),FVEC(M),WA(LWA)
EXTERNAL FCN
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to LMDIF1 and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from LMDIF1.

FCN is the name of the user-supplied subroutine which calculates the functions. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
SUBROUTINE FCN(M,N,X,FVEC,IFLAG)
INTEGER M,N,IFLAG
DOUBLE PRECISION X(N),FVEC(M)
----------
CALCULATE THE FUNCTIONS AT X AND
RETURN THIS VECTOR IN FVEC.
----------
RETURN
END
```

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of LMDIF1. In this case set

IFLAG to a negative integer.

M is a positive integer input variable set to the number of functions.

N is a positive integer input variable set to the number of variables. N must not exceed M.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length M which contains the functions evaluated at the output X.

TOL is a nonnegative input variable. Termination occurs when the algorithm estimates either that the relative error in the sum of squares is at most TOL or that the relative error between X and the solution is at most TOL. Section 4 contains more details about TOL.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0 Improper input parameters.

INFO = 1 Algorithm estimates that the relative error in the sum of squares is at most TOL.

INFO = 2 Algorithm estimates that the relative error between X and the solution is at most TOL.

INFO = 3 Conditions for INFO = 1 and INFO = 2 both hold.

INFO = 4 FVEC is orthogonal to the columns of the Jacobian to machine precision.

INFO = 5 Number of calls to FCN has reached or exceeded 200*(N+1).

INFO = 6 TOL is too small. No further reduction in the sum of squares is possible.

INFO = 7 TOL is too small. No further improvement in the approximate solution X is possible.

Sections 4 and 5 contain more details about INFO.

IWA is an integer work array of length N.

WA is a work array of length LWA.

LWA is a positive integer input variable not less than

M*N+5*N+M.

4. Successful completion.

The accuracy of LMDIF1 is controlled by the convergence parameter TOL. This parameter is used in tests which make three types of comparisons between the approximation X and a solution XSOL. LMDIF1 terminates when any of the tests is satisfied. If TOL is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then LMDIF1 only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible. Unless high precision solutions are required, the recommended value for TOL is the square root of the machine precision.

The tests assume that the functions are reasonably well behaved. If this condition is not satisfied, then LMDIF1 may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning LMDIF1 with a tighter tolerance.

First convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z, then this test attempts to guarantee that

```
ENORM(FVEC) .LE. (1+TOL)*ENORM(FVECS),
```

where FVECS denotes the functions evaluated at XSOL. If this condition is satisfied with TOL = 10**(-K), then the final residual norm ENORM(FVEC) has K significant decimal digits and INFO is set to 1 (or to 3 if the second test is also satisfied).

Second convergence test. If D is a diagonal matrix (implicitly generated by LMDIF1) whose entries contain scale factors for the variables, then this test attempts to guarantee that

```
ENORM(D*(X-XSOL)) .LE. TOL*ENORM(D*XSOL).
```

If this condition is satisfied with TOL = 10**(-K), then the larger components of D*X have K significant decimal digits and INFO is set to 2 (or to 3 if the first test is also satisfied). There is a danger that the smaller components of D*X may have large relative errors, but the choice of D is such that the accuracy of the components of X is usually related to their sensitivity.

Third convergence test. This test is satisfied when FVEC is orthogonal to the columns of the Jacobian to machine precision. There is no clear relationship between this test and the accuracy of LMDIF1, and furthermore, the test is equally well satisfied at other critical points, namely maximizers and saddle points. Also, errors in the functions (see below) may result in the test being satisfied at a point not close to the minimum. Therefore, termination caused by this test (INFO = 4) should be examined carefully.

5. Unsuccessful completion.

Unsuccessful termination of LMDIF1 can be due to improper input parameters, arithmetic interrupts, an excessive number of function evaluations, or errors in the functions.

Improper input parameters. INFO is set to 0 if N .LE. 0, or M .LT. N, or TOL .LT. 0.D0, or LWA .LT. M*N+5*N+M.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by LMDIF1. In this case, it may be possible to remedy the situation by not evaluating the functions here, but instead setting the components of FVEC to numbers that exceed those in the initial FVEC, thereby indirectly reducing the step length. The step length can be more directly controlled by using instead LMDIF, which includes in its calling sequence the step-length-governing parameter FACTOR.

Excessive number of function evaluations. If the number of calls to FCN reaches 200*(N+1), then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and INFO is set to 5. In this case, it may be helpful to restart LMDIF1, thereby forcing it to disregard old (and possibly harmful) information.

Errors in the functions. The choice of step length in the forward-difference approximation to the Jacobian assumes that the relative errors in the functions are of the order of the machine precision. If this is not the case, LMDIF1 may fail (usually with INFO = 4). The user should then use LMDIF instead, or one of the programs which require the analytic Jacobian (LMDER1 and LMDER).

6. Characteristics of the algorithm.

LMDIF1 is a modification of the Levenberg-Marquardt algorithm. Two of its main characteristics involve the proper use of implicitly scaled variables and an optimal choice for the correction. The use of implicitly scaled variables achieves scale invariance of LMDIF1 and limits the size of the correction in any direction where the functions are changing rapidly. The optimal choice of the correction guarantees (under reasonable conditions) global convergence from starting points far from the solution and a fast rate of convergence for problems with small residuals.

Timing. The time required by LMDIF1 to solve a given problem depends on M and N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by LMDIF1 is about N**3 to process each evaluation of the functions (one call to FCN) and M*(N**2) to process each approximation to the Jacobian (N calls to FCN). Unless FCN can be evaluated quickly, the timing of LMDIF1 will be strongly influenced by the time spent in FCN.

Storage. LMDIF1 requires M*N + 2*M + 6*N double precision storage locations and N integer storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DPMPAR,ENORM,FDJAC2,LMDIF,LMPAR,
QRFAC,QRSOLV

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MOD

8. References.

Jorge J. More, The Levenberg-Marquardt Algorithm, Implementation and Theory. Numerical Analysis, G. A. Watson, editor. Lecture Notes in Mathematics 630, Springer-Verlag, 1977.

9. Example.

The problem is to determine the values of x(1), x(2), and x(3) which provide the best fit (in the least squares sense) of

x(1) + u(i)/(v(i)*x(2) + w(i)*x(3)), i = 1, 15

to the data

y = (0.14,0.18,0.22,0.25,0.29,0.32,0.35,0.39,
     0.37,0.58,0.73,0.96,1.34,2.10,4.39),

where u(i) = i, v(i) = 16 - i, and w(i) = min(u(i),v(i)). The i-th component of FVEC is thus defined by

y(i) - (x(1) + u(i)/(v(i)*x(2) + w(i)*x(3))).

```
C     **********
C
C     DRIVER FOR LMDIF1 EXAMPLE.
C     DOUBLE PRECISION VERSION
C
```

```
C       **********
        INTEGER J,M,N,INFO,LWA,NWRITE
        INTEGER IWA(3)
        DOUBLE PRECISION TOL,FNORM
        DOUBLE PRECISION X(3),FVEC(15),WA(75)
        DOUBLE PRECISION ENORM,DPMPAR
        EXTERNAL FCN
C
C       LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
        DATA NWRITE /6/
C
        M = 15
        N = 3
C
C       THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
        X(1) = 1.D0
        X(2) = 1.D0
        X(3) = 1.D0
C
        LWA = 75
C
C       SET TOL TO THE SQUARE ROOT OF THE MACHINE PRECISION.
C       UNLESS HIGH PRECISION SOLUTIONS ARE REQUIRED,
C       THIS IS THE RECOMMENDED SETTING.
C
        TOL = DSQRT(DPMPAR(1))
C
        CALL LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)
        FNORM = ENORM(M,FVEC)
        WRITE (NWRITE,1000) FNORM,INFO,(X(J),J=1,N)
        STOP
 1000   FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
     *          5X,15H EXIT PARAMETER,16X,I10 //
     *          5X,27H FINAL APPROXIMATE SOLUTION // 5X,3D15.7)
C
C       LAST CARD OF DRIVER FOR LMDIF1 EXAMPLE.
C
        END
        SUBROUTINE FCN(M,N,X,FVEC,IFLAG)
        INTEGER M,N,IFLAG
        DOUBLE PRECISION X(N),FVEC(M)
C
C       SUBROUTINE FCN FOR LMDIF1 EXAMPLE.
C
        INTEGER I
        DOUBLE PRECISION TMP1,TMP2,TMP3
        DOUBLE PRECISION Y(15)
        DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),
     *       Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15)
     *       /1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,
     *        3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/
C
```

```
      DO 10 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         FVEC(I) = Y(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
   10    CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

Results obtained with different compilers or machines may be slightly different.

```
FINAL L2 NORM OF THE RESIDUALS  0.9063596D-01

EXIT PARAMETER                         1

FINAL APPROXIMATE SOLUTION

 0.8241057D-01  0.1133037D+01  0.2343695D+01
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Documentation for MINPACK subroutine LMDIF

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of LMDIF is to minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm. The user must provide a subroutine which calculates the functions. The Jacobian is then calculated by a forward-difference approximation.

2. Subroutine and type statements.

```
      SUBROUTINE LMDIF(FCN,M,N,X,FVEC,FTOL,XTOL,GTOL,MAXFEV,EPSFCN,
     *                 DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,
     *                 IPVT,QTF,WA1,WA2,WA3,WA4)
      INTEGER M,N,MAXFEV,MODE,NPRINT,INFO,NFEV,LDFJAC
      INTEGER IPVT(N)
      DOUBLE PRECISION FTOL,XTOL,GTOL,EPSFCN,FACTOR
      DOUBLE PRECISION X(N),FVEC(M),DIAG(N),FJAC(LDFJAC,N),QTF(N),
     *                 WA1(N),WA2(N),WA3(N),WA4(M)
      EXTERNAL FCN
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to LMDIF and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from LMDIF.

FCN is the name of the user-supplied subroutine which calculates the functions. FCN must be declared in an EXTERNAL statement in the user calling program, and should be written as follows.

```
      SUBROUTINE FCN(M,N,X,FVEC,IFLAG)
      INTEGER M,N,IFLAG
      DOUBLE PRECISION X(N),FVEC(M)
      ----------
      CALCULATE THE FUNCTIONS AT X AND
      RETURN THIS VECTOR IN FVEC.
      ----------
      RETURN
      END
```

The value of IFLAG should not be changed by FCN unless the user wants to terminate execution of LMDIF. In this case set IFLAG to a negative integer.

M is a positive integer input variable set to the number of functions.

N is a positive integer input variable set to the number of variables. N must not exceed M.

X is an array of length N. On input X must contain an initial estimate of the solution vector. On output X contains the final estimate of the solution vector.

FVEC is an output array of length M which contains the functions evaluated at the output X.

FTOL is a nonnegative input variable. Termination occurs when both the actual and predicted relative reductions in the sum of squares are at most FTOL. Therefore, FTOL measures the relative error desired in the sum of squares. Section 4 contains more details about FTOL.

XTOL is a nonnegative input variable. Termination occurs when the relative error between two consecutive iterates is at most XTOL. Therefore, XTOL measures the relative error desired in the approximate solution. Section 4 contains more details about XTOL.

GTOL is a nonnegative input variable. Termination occurs when the cosine of the angle between FVEC and any column of the Jacobian is at most GTOL in absolute value. Therefore, GTOL measures the orthogonality desired between the function vector and the columns of the Jacobian. Section 4 contains more details about GTOL.

MAXFEV is a positive integer input variable. Termination occurs when the number of calls to FCN is at least MAXFEV by the end of an iteration.

EPSFCN is an input variable used in determining a suitable step for the forward-difference approximation. This approximation assumes that the relative errors in the functions are of the order of EPSFCN. If EPSFCN is less than the machine precision, it is assumed that the relative errors in the functions are of the order of the machine precision.

DIAG is an array of length N. If MODE = 1 (see below), DIAG is internally set. If MODE = 2, DIAG must contain positive entries that serve as multiplicative scale factors for the variables.

MODE is an integer input variable. If MODE = 1, the variables will be scaled internally. If MODE = 2, the scaling is specified by the input DIAG. Other values of MODE are equivalent to MODE = 1.

FACTOR is a positive input variable used in determining the initial step bound. This bound is set to the product of FACTOR and the Euclidean norm of DIAG*X if nonzero, or else to FACTOR itself. In most cases FACTOR should lie in the interval (.1,100.). 100. is a generally recommended value.

NPRINT is an integer input variable that enables controlled printing of iterates if it is positive. In this case, FCN is called with IFLAG = 0 at the beginning of the first iteration and every NPRINT iterations thereafter and immediately prior to return, with X and FVEC available for printing. If NPRINT is not positive, no special calls of FCN with IFLAG = 0 are made.

INFO is an integer output variable. If the user has terminated execution, INFO is set to the (negative) value of IFLAG. See description of FCN. Otherwise, INFO is set as follows.

INFO = 0	Improper input parameters.
INFO = 1	Both actual and predicted relative reductions in the sum of squares are at most FTOL.
INFO = 2	Relative error between two consecutive iterates is at most XTOL.
INFO = 3	Conditions for INFO = 1 and INFO = 2 both hold.
INFO = 4	The cosine of the angle between FVEC and any column of the Jacobian is at most GTOL in absolute value.
INFO = 5	Number of calls to FCN has reached or exceeded MAXFEV.
INFO = 6	FTOL is too small. No further reduction in the sum of squares is possible.
INFO = 7	XTOL is too small. No further improvement in the approximate solution X is possible.
INFO = 8	GTOL is too small. FVEC is orthogonal to the columns of the Jacobian to machine precision.

Sections 4 and 5 contain more details about INFO.

NFEV is an integer output variable set to the number of calls to FCN.

FJAC is an output M by N array. The upper N by N submatrix of FJAC contains an upper triangular matrix R with diagonal elements of nonincreasing magnitude such that

$$P^T*(JAC^T*JAC)*P = R^T*R,$$

where P is a permutation matrix and JAC is the final calculated Jacobian. Column j of P is column IPVT(j) (see below) of the identity matrix. The lower trapezoidal part of FJAC contains information generated during the computation of R.

LDFJAC is a positive integer input variable not less than M which specifies the leading dimension of the array FJAC.

IPVT is an integer output array of length N. IPVT defines a permutation matrix P such that JAC*P = Q*R, where JAC is the final calculated Jacobian, Q is orthogonal (not stored), and R is upper triangular with diagonal elements of nonincreasing magnitude. Column j of P is column IPVT(j) of the identity matrix.

QTF is an output array of length N which contains the first N elements of the vector (Q transpose)*FVEC.

WA1, WA2, and WA3 are work arrays of length N.

WA4 is a work array of length M.

4. Successful completion.

The accuracy of LMDIF is controlled by the convergence parameters FTOL, XTOL, and GTOL. These parameters are used in tests which make three types of comparisons between the approximation X and a solution XSOL. LMDIF terminates when any of the tests is satisfied. If any of the convergence parameters is less than the machine precision (as defined by the MINPACK function DPMPAR(1)), then LMDIF only attempts to satisfy the test defined by the machine precision. Further progress is not usually possible.

The tests assume that the functions are reasonably well behaved. If this condition is not satisfied, then LMDIF may incorrectly indicate convergence. The validity of the answer can be checked, for example, by rerunning LMDIF with tighter tolerances.

First convergence test. If ENORM(Z) denotes the Euclidean norm of a vector Z, then this test attempts to guarantee that

ENORM(FVEC) .LE. (1+FTOL)*ENORM(FVECS),

where FVECS denotes the functions evaluated at XSOL. If this condition is satisfied with FTOL = 10**(-K), then the final residual norm ENORM(FVEC) has K significant decimal digits and INFO is set to 1 (or to 3 if the second test is also satisfied). Unless high precision solutions are required, the recommended value for FTOL is the square root of the machine precision.

Second convergence test. If D is the diagonal matrix whose entries are defined by the array DIAG, then this test attempts to guarantee that

```
ENORM(D*(X-XSOL)) .LE. XTOL*ENORM(D*XSOL).
```

If this condition is satisfied with XTOL = 10**(-K), then the larger components of D*X have K significant decimal digits and INFO is set to 2 (or to 3 if the first test is also satisfied). There is a danger that the smaller components of D*X may have large relative errors, but if MODE = 1, then the accuracy of the components of X is usually related to their sensitivity. Unless high precision solutions are required, the recommended value for XTOL is the square root of the machine precision.

Third convergence test. This test is satisfied when the cosine of the angle between FVEC and any column of the Jacobian at X is at most GTOL in absolute value. There is no clear relationship between this test and the accuracy of LMDIF, and furthermore, the test is equally well satisfied at other critical points, namely maximizers and saddle points. Therefore, termination caused by this test (INFO = 4) should be examined carefully. The recommended value for GTOL is zero.

5. Unsuccessful completion.

Unsuccessful termination of LMDIF can be due to improper input parameters, arithmetic interrupts, or an excessive number of function evaluations.

Improper input parameters. INFO is set to 0 if N .LE. 0, or M .LT. N, or LDFJAC .LT. M, or FTOL .LT. 0.D0, or XTOL .LT. 0.D0, or GTOL .LT. 0.D0, or MAXFEV .LE. 0, or FACTOR .LE. 0.D0.

Arithmetic interrupts. If these interrupts occur in the FCN subroutine during an early stage of the computation, they may be caused by an unacceptable choice of X by LMDIF. In this case, it may be possible to remedy the situation by rerunning LMDIF with a smaller value of FACTOR.

Excessive number of function evaluations. A reasonable value for MAXFEV is 200*(N+1). If the number of calls to FCN reaches MAXFEV, then this indicates that the routine is converging very slowly as measured by the progress of FVEC, and INFO is set to 5. In this case, it may be helpful to restart LMDIF with MODE set to 1.

6. Characteristics of the algorithm.

LMDIF is a modification of the Levenberg-Marquardt algorithm. Two of its main characteristics involve the proper use of implicitly scaled variables (if MODE = 1) and an optimal choice for the correction. The use of implicitly scaled variables achieves scale invariance of LMDIF and limits the size of the correction in any direction where the functions are changing rapidly. The optimal choice of the correction guarantees (under reasonable conditions) global convergence from starting points far from the solution and a fast rate of convergence for problems with small residuals.

Timing. The time required by LMDIF to solve a given problem depends on M and N, the behavior of the functions, the accuracy requested, and the starting point. The number of arithmetic operations needed by LMDIF is about N**3 to process each evaluation of the functions (one call to FCN) and M*(N**2) to process each approximation to the Jacobian (N calls to FCN). Unless FCN can be evaluated quickly, the timing of LMDIF will be strongly influenced by the time spent in FCN.

Storage. LMDIF requires M*N + 2*M + 6*N double precision storage locations and N integer storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied ...... FCN

MINPACK-supplied ... DPMPAR,ENORM,FDJAC2,LMPAR,QRFAC,QRSOLV

FORTRAN-supplied ... DABS,DMAX1,DMIN1,DSQRT,MOD

8. References.

Jorge J. More, The Levenberg-Marquardt Algorithm, Implementation and Theory. Numerical Analysis, G. A. Watson, editor. Lecture Notes in Mathematics 630, Springer-Verlag, 1977.

9. Example.

The problem is to determine the values of x(1), x(2), and x(3) which provide the best fit (in the least squares sense) of

```
x(1) + u(i)/(v(i)*x(2) + w(i)*x(3)),  i = 1, 15
```

to the data

```
        y = (0.14,0.18,0.22,0.25,0.29,0.32,0.35,0.39,
             0.37,0.58,0.73,0.96,1.34,2.10,4.39),

      where u(i) = i, v(i) = 16 - i, and w(i) = min(u(i),v(i)).  The
      i-th component of FVEC is thus defined by

            y(i) - (x(1) + u(i)/(v(i)*x(2) + w(i)*x(3))).

C     **********
C
C     DRIVER FOR LMDIF EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER J,M,N,MAXFEV,MODE,NPRINT,INFO,NFEV,LDFJAC,NWRITE
      INTEGER IPVT(3)
      DOUBLE PRECISION FTOL,XTOL,GTOL,EPSFCN,FACTOR,FNORM
      DOUBLE PRECISION X(3),FVEC(15),DIAG(3),FJAC(15,3),QTF(3),
     *                 WA1(3),WA2(3),WA3(3),WA4(15)
      DOUBLE PRECISION ENORM,DPMPAR
      EXTERNAL FCN
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      M = 15
      N = 3
C
C     THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
      X(1) = 1.D0
      X(2) = 1.D0
      X(3) = 1.D0
C
      LDFJAC = 15
C
C     SET FTOL AND XTOL TO THE SQUARE ROOT OF THE MACHINE PRECISION
C     AND GTOL TO ZERO. UNLESS HIGH PRECISION SOLUTIONS ARE
C     REQUIRED, THESE ARE THE RECOMMENDED SETTINGS.
C
      FTOL = DSQRT(DPMPAR(1))
      XTOL = DSQRT(DPMPAR(1))
      GTOL = 0.D0
C
      MAXFEV = 800
      EPSFCN = 0.D0
      MODE = 1
      FACTOR = 1.D2
      NPRINT = 0
C
      CALL LMDIF(FCN,M,N,X,FVEC,FTOL,XTOL,GTOL,MAXFEV,EPSFCN,
     *           DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,
     *           IPVT,QTF,WA1,WA2,WA3,WA4)
```

```
      FNORM = ENORM(M,FVEC)
      WRITE (NWRITE,1000) FNORM,NFEV,INFO,(X(J),J=1,N)
      STOP
 1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
     *        5X,31H NUMBER OF FUNCTION EVALUATIONS,I10 //
     *        5X,15H EXIT PARAMETER,16X,I10 //
     *        5X,27H FINAL APPROXIMATE SOLUTION // 5X,3D15.7)
C
C     LAST CARD OF DRIVER FOR LMDIF EXAMPLE.
C
      END
      SUBROUTINE FCN(M,N,X,FVEC,IFLAG)
      INTEGER M,N,IFLAG
      DOUBLE PRECISION X(N),FVEC(M)
C
C     SUBROUTINE FCN FOR LMDIF EXAMPLE.
C
      INTEGER I
      DOUBLE PRECISION TMP1,TMP2,TMP3
      DOUBLE PRECISION Y(15)
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),
     *     Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15)
     *     /1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,
     *      3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/
C
      IF (IFLAG .NE. 0) GO TO 5
C
C     INSERT PRINT STATEMENTS HERE WHEN NPRINT IS POSITIVE.
C
      RETURN
    5 CONTINUE
      DO 10 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         FVEC(I) = Y(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
   10    CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

Results obtained with different compilers or machines may be slightly different.

```
FINAL L2 NORM OF THE RESIDUALS  0.9063596D-01

NUMBER OF FUNCTION EVALUATIONS        21

EXIT PARAMETER                         1

FINAL APPROXIMATE SOLUTION
```

0.8241057D-01   0.1133037D+01   0.2343695D+01

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Documentation for MINPACK subroutine CHKDER

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstrom, Jorge J. More

March 1980

1. Purpose.

The purpose of CHKDER is to check the gradients of M nonlinear functions in N variables, evaluated at a point X, for consistency with the functions themselves. The user must call CHKDER twice, first with MODE = 1 and then with MODE = 2.

2. Subroutine and type statements.

```
      SUBROUTINE CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
      INTEGER M,N,LDFJAC,MODE
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),XP(N),FVECP(M),
     *                 ERR(M)
```

3. Parameters.

Parameters designated as input parameters must be specified on entry to CHKDER and are not changed on exit, while parameters designated as output parameters need not be specified on entry and are set to appropriate values on exit from CHKDER.

M is a positive integer input variable set to the number of functions.

N is a positive integer input variable set to the number of variables.

X is an input array of length N.

FVEC is an array of length M. On input when MODE = 2, FVEC must contain the functions evaluated at X.

FJAC is an M by N array. On input when MODE = 2, the rows of FJAC must contain the gradients of the respective functions evaluated at X.

LDFJAC is a positive integer input variable not less than M which specifies the leading dimension of the array FJAC.

XP is an array of length N. On output when MODE = 1, XP is set to a neighboring point of X.

FVECP is an array of length M. On input when MODE = 2, FVECP must contain the functions evaluated at XP.

MODE is an integer input variable set to 1 on the first call and 2 on the second. Other values of MODE are equivalent to MODE = 1.

ERR is an array of length M. On output when MODE = 2, ERR contains measures of correctness of the respective gradients. If there is no severe loss of significance, then if ERR(I) is 1.0 the I-th gradient is correct, while if ERR(I) is 0.0 the I-th gradient is incorrect. For values of ERR between 0.0 and 1.0, the categorization is less certain. In general, a value of ERR(I) greater than 0.5 indicates that the I-th gradient is probably correct, while a value of ERR(I) less than 0.5 indicates that the I-th gradient is probably incorrect.

4. Successful completion.

CHKDER usually guarantees that if ERR(I) is 1.0, then the I-th gradient at X is consistent with the I-th function. This suggests that the input X be such that consistency of the gradient at X implies consistency of the gradient at all points of interest. If all the components of X are distinct and the fractional part of each one has two nonzero digits, then X is likely to be a satisfactory choice.

If ERR(I) is not 1.0 but is greater than 0.5, then the I-th gradient is probably consistent with the I-th function (the more so the larger ERR(I) is), but the conditions for ERR(I) to be 1.0 have not been completely satisfied. In this case, it is recommended that CHKDER be rerun with other input values of X. If ERR(I) is always greater than 0.5, then the I-th gradient is consistent with the I-th function.

5. Unsuccessful completion.

CHKDER does not perform reliably if cancellation or rounding errors cause a severe loss of significance in the evaluation of a function. Therefore, none of the components of X should be unusually small (in particular, zero) or any other value which may cause loss of significance. The relative differences between corresponding elements of FVECP and FVEC should be at least two orders of magnitude greater than the machine precision (as defined by the MINPACK function DPMPAR(1)). If there is a severe loss of significance in the evaluation of the I-th function, then ERR(I) may be 0.0 and yet the I-th gradient could be correct.

If ERR(I) is not 0.0 but is less than 0.5, then the I-th gradient is probably not consistent with the I-th function (the more so the smaller ERR(I) is), but the conditions for ERR(I) to be 0.0 have not been completely satisfied. In this case, it is recommended that CHKDER be rerun with other input values of X. If ERR(I) is always less than 0.5 and if there is no severe loss of significance, then the I-th gradient is not consistent with the I-th function.

6. Characteristics of the algorithm.

CHKDER checks the I-th gradient for consistency with the I-th function by computing a forward-difference approximation along a suitably chosen direction and comparing this approximation with the user-supplied gradient along the same direction. The principal characteristic of CHKDER is its invariance to changes in scale of the variables or functions.

Timing. The time required by CHKDER depends only on M and N. The number of arithmetic operations needed by CHKDER is about N when MODE = 1 and M*N when MODE = 2.

Storage. CHKDER requires M*N + 3*M + 2*N double precision storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

MINPACK-supplied ... DPMPAR

FORTRAN-supplied ... DABS,DLOG10,DSQRT

8. References.

None.

9. Example.

This example checks the Jacobian matrix for the problem that determines the values of x(1), x(2), and x(3) which provide the best fit (in the least squares sense) of

```
x(1) + u(i)/(v(i)*x(2) + w(i)*x(3)),   i = 1, 15
```

to the data

```
y = (0.14,0.18,0.22,0.25,0.29,0.32,0.35,0.39,
     0.37,0.58,0.73,0.96,1.34,2.10,4.39),
```

where u(i) = i, v(i) = 16 - i, and w(i) = min(u(i),v(i)). The i-th component of FVEC is thus defined by

```
y(i) - (x(1) + u(i)/(v(i)*x(2) + w(i)*x(3))).
```

```
C     **********
C
C     DRIVER FOR CHKDER EXAMPLE.
C     DOUBLE PRECISION VERSION
C
C     **********
      INTEGER I,M,N,LDFJAC,MODE,NWRITE
      DOUBLE PRECISION X(3),FVEC(15),FJAC(15,3),XP(3),FVECP(15),
     *                 ERR(15)
C
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NWRITE /6/
C
      M = 15
      N = 3
C
C     THE FOLLOWING VALUES SHOULD BE SUITABLE FOR
C     CHECKING THE JACOBIAN MATRIX.
C
      X(1) = 9.2D-1
      X(2) = 1.3D-1
      X(3) = 5.4D-1
C
      LDFJAC = 15
C
      MODE = 1
      CALL CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
      MODE = 2
      CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,1)
      CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,2)
      CALL FCN(M,N,XP,FVECP,FJAC,LDFJAC,1)
      CALL CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
C
      DO 10 I = 1, M
         FVECP(I) = FVECP(I) - FVEC(I)
   10    CONTINUE
      WRITE (NWRITE,1000) (FVEC(I),I=1,M)
      WRITE (NWRITE,2000) (FVECP(I),I=1,M)
      WRITE (NWRITE,3000) (ERR(I),I=1,M)
      STOP
 1000 FORMAT (/5X,5H FVEC // (5X,3D15.7))
 2000 FORMAT (/5X,13H FVECP - FVEC // (5X,3D15.7))
 3000 FORMAT (/5X,4H ERR // (5X,3D15.7))
C
C     LAST CARD OF DRIVER FOR CHKDER EXAMPLE.
C
      END
      SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
      INTEGER M,N,LDFJAC,IFLAG
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)
C
C     SUBROUTINE FCN FOR CHKDER EXAMPLE.
C
```

```
      INTEGER I
      DOUBLE PRECISION TMP1,TMP2,TMP3,TMP4
      DOUBLE PRECISION Y(15)
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),
     *     Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15)
     *     /1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,
     *      3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/
C
      IF (IFLAG .EQ. 2) GO TO 20
      DO 10 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         FVEC(I) = Y(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
   10    CONTINUE
      GO TO 40
   20 CONTINUE
      DO 30 I = 1, 15
         TMP1 = I
         TMP2 = 16 - I
C
C        ERROR INTRODUCED INTO NEXT STATEMENT FOR ILLUSTRATION.
C        CORRECTED STATEMENT SHOULD READ    TMP3 = TMP1 .
C
         TMP3 = TMP2
         IF (I .GT. 8) TMP3 = TMP2
         TMP4 = (X(2)*TMP2 + X(3)*TMP3)**2
         FJAC(I,1) = -1.D0
         FJAC(I,2) = TMP1*TMP2/TMP4
         FJAC(I,3) = TMP1*TMP3/TMP4
   30    CONTINUE
   40 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE FCN.
C
      END
```

Results obtained with different compilers or machines may be different. In particular, the differences FVECP - FVEC are machine dependent.

```
FVEC

-0.1181606D+01 -0.1429655D+01 -0.1606344D+01
-0.1745269D+01 -0.1840654D+01 -0.1921586D+01
-0.1984141D+01 -0.2022537D+01 -0.2468977D+01
-0.2827562D+01 -0.3473582D+01 -0.4437612D+01
-0.6047662D+01 -0.9267761D+01 -0.1891806D+02

FVECP - FVEC

-0.7724666D-08 -0.3432405D-08 -0.2034843D-09
```

0.2313685D-08	0.4331078D-08	0.5984096D-08
0.7363281D-08	0.8531470D-08	0.1488591D-07
0.2335850D-07	0.3522012D-07	0.5301255D-07
0.8266660D-07	0.1419747D-06	0.3198990D-06

ERR

0.1141397D+00	0.9943516D-01	0.9674474D-01
0.9980447D-01	0.1073116D+00	0.1220445D+00
0.1526814D+00	0.1000000D+01	0.1000000D+01
0.1000000D+01	0.1000000D+01	0.1000000D+01
0.1000000D+01	0.1000000D+01	0.1000000D+01

CHAPTER 5
Program Listings

This chapter contains the double precision version of the MINPACK-1 program listings; both single and double precision versions of the subprograms are available with the MINPACK-1 package. The listings appear in the following (alphanumeric) order:

> CHKDER, DOGLEG, ENORM, FDJAC1, FDJAC2, HYBRD, HYBRD1, HYBRJ, HYBRJ1, LMDER, LMDER1, LMDIF, LMDIF1, LMPAR, LMSTR, LMSTR1, QFORM, QRFAC, QRSOLV, RWUPDT, R1MPYQ, R1UPDT.

Functions SPMPAR (single precision) and DPMPAR (double precision), which provide the machine-dependent constants, appear at the end.

```
      SUBROUTINE CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
      INTEGER M,N,LDFJAC,MODE
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),XP(N),FVECP(M),
     *                 ERR(M)
C     **********
C
C     SUBROUTINE CHKDER
C
C     THIS SUBROUTINE CHECKS THE GRADIENTS OF M NONLINEAR FUNCTIONS
C     IN N VARIABLES, EVALUATED AT A POINT X, FOR CONSISTENCY WITH
C     THE FUNCTIONS THEMSELVES. THE USER MUST CALL CHKDER TWICE,
C     FIRST WITH MODE = 1 AND THEN WITH MODE = 2.
C
C     MODE = 1. ON INPUT, X MUST CONTAIN THE POINT OF EVALUATION.
C               ON OUTPUT, XP IS SET TO A NEIGHBORING POINT.
C
C     MODE = 2. ON INPUT, FVEC MUST CONTAIN THE FUNCTIONS AND THE
C                         ROWS OF FJAC MUST CONTAIN THE GRADIENTS
C                         OF THE RESPECTIVE FUNCTIONS EACH EVALUATED
C                         AT X, AND FVECP MUST CONTAIN THE FUNCTIONS
C                         EVALUATED AT XP.
C               ON OUTPUT, ERR CONTAINS MEASURES OF CORRECTNESS OF
C                          THE RESPECTIVE GRADIENTS.
C
C     THE SUBROUTINE DOES NOT PERFORM RELIABLY IF CANCELLATION OR
C     ROUNDING ERRORS CAUSE A SEVERE LOSS OF SIGNIFICANCE IN THE
C     EVALUATION OF A FUNCTION. THEREFORE, NONE OF THE COMPONENTS
C     OF X SHOULD BE UNUSUALLY SMALL (IN PARTICULAR, ZERO) OR ANY
C     OTHER VALUE WHICH MAY CAUSE LOSS OF SIGNIFICANCE.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
C
C     WHERE
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF VARIABLES.
C
C       X IS AN INPUT ARRAY OF LENGTH N.
C
C       FVEC IS AN ARRAY OF LENGTH M. ON INPUT WHEN MODE = 2,
C         FVEC MUST CONTAIN THE FUNCTIONS EVALUATED AT X.
C
C       FJAC IS AN M BY N ARRAY. ON INPUT WHEN MODE = 2,
C         THE ROWS OF FJAC MUST CONTAIN THE GRADIENTS OF
C         THE RESPECTIVE FUNCTIONS EVALUATED AT X.
C
C       LDFJAC IS A POSITIVE INTEGER INPUT PARAMETER NOT LESS THAN M
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
C
```

```
C       XP IS AN ARRAY OF LENGTH N. ON OUTPUT WHEN MODE = 1,
C         XP IS SET TO A NEIGHBORING POINT OF X.
C
C       FVECP IS AN ARRAY OF LENGTH M. ON INPUT WHEN MODE = 2,
C         FVECP MUST CONTAIN THE FUNCTIONS EVALUATED AT XP.
C
C       MODE IS AN INTEGER INPUT VARIABLE SET TO 1 ON THE FIRST CALL
C         AND 2 ON THE SECOND. OTHER VALUES OF MODE ARE EQUIVALENT
C         TO MODE = 1.
C
C       ERR IS AN ARRAY OF LENGTH M. ON OUTPUT WHEN MODE = 2,
C         ERR CONTAINS MEASURES OF CORRECTNESS OF THE RESPECTIVE
C         GRADIENTS. IF THERE IS NO SEVERE LOSS OF SIGNIFICANCE,
C         THEN IF ERR(I) IS 1.0 THE I-TH GRADIENT IS CORRECT,
C         WHILE IF ERR(I) IS 0.0 THE I-TH GRADIENT IS INCORRECT.
C         FOR VALUES OF ERR BETWEEN 0.0 AND 1.0, THE CATEGORIZATION
C         IS LESS CERTAIN. IN GENERAL, A VALUE OF ERR(I) GREATER
C         THAN 0.5 INDICATES THAT THE I-TH GRADIENT IS PROBABLY
C         CORRECT, WHILE A VALUE OF ERR(I) LESS THAN 0.5 INDICATES
C         THAT THE I-TH GRADIENT IS PROBABLY INCORRECT.
C
C     SUBPROGRAMS CALLED
C
C       MINPACK SUPPLIED ... DPMPAR
C
C       FORTRAN SUPPLIED ... DABS,DLOG10,DSQRT
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,J
      DOUBLE PRECISION EPS,EPSF,EPSLOG,EPSMCH,FACTOR,ONE,TEMP,ZERO
      DOUBLE PRECISION DPMPAR
      DATA FACTOR,ONE,ZERO /1.0D2,1.0D0,0.0D0/
C
C     EPSMCH IS THE MACHINE PRECISION.
C
      EPSMCH = DPMPAR(1)
C
      EPS = DSQRT(EPSMCH)
C
      IF (MODE .EQ. 2) GO TO 20
C
C        MODE = 1.
C
         DO 10 J = 1, N
            TEMP = EPS*DABS(X(J))
            IF (TEMP .EQ. ZERO) TEMP = EPS
            XP(J) = X(J) + TEMP
   10       CONTINUE
         GO TO 70
   20 CONTINUE
C
```

```
C        MODE = 2.
C
         EPSF = FACTOR*EPSMCH
         EPSLOG = DLOG10(EPS)
         DO 30 I = 1, M
            ERR(I) = ZERO
   30       CONTINUE
         DO 50 J = 1, N
            TEMP = DABS(X(J))
            IF (TEMP .EQ. ZERO) TEMP = ONE
            DO 40 I = 1, M
               ERR(I) = ERR(I) + TEMP*FJAC(I,J)
   40          CONTINUE
   50       CONTINUE
         DO 60 I = 1, M
            TEMP = ONE
            IF (FVEC(I) .NE. ZERO .AND. FVECP(I) .NE. ZERO
     *          .AND. DABS(FVECP(I)-FVEC(I)) .GE. EPSF*DABS(FVEC(I)))
     *         TEMP = EPS*DABS((FVECP(I)-FVEC(I))/EPS-ERR(I))
     *                /(DABS(FVEC(I)) + DABS(FVECP(I)))
            ERR(I) = ONE
            IF (TEMP .GT. EPSMCH .AND. TEMP .LT. EPS)
     *         ERR(I) = (DLOG10(TEMP) - EPSLOG)/EPSLOG
            IF (TEMP .GE. EPS) ERR(I) = ZERO
   60       CONTINUE
   70 CONTINUE
C
      RETURN
C
C     LAST CARD OF SUBROUTINE CHKDER.
C
      END
```

```
      SUBROUTINE DOGLEG(N,R,LR,DIAG,QTB,DELTA,X,WA1,WA2)
      INTEGER N,LR
      DOUBLE PRECISION DELTA
      DOUBLE PRECISION R(LR),DIAG(N),QTB(N),X(N),WA1(N),WA2(N)
C     **********
C
C     SUBROUTINE DOGLEG
C
C     GIVEN AN M BY N MATRIX A, AN N BY N NONSINGULAR DIAGONAL
C     MATRIX D, AN M-VECTOR B, AND A POSITIVE NUMBER DELTA, THE
C     PROBLEM IS TO DETERMINE THE CONVEX COMBINATION X OF THE
C     GAUSS-NEWTON AND SCALED GRADIENT DIRECTIONS THAT MINIMIZES
C     (A*X - B) IN THE LEAST SQUARES SENSE, SUBJECT TO THE
C     RESTRICTION THAT THE EUCLIDEAN NORM OF D*X BE AT MOST DELTA.
C
C     THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM
C     IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE
C     QR FACTORIZATION OF A. THAT IS, IF A = Q*R, WHERE Q HAS
C     ORTHOGONAL COLUMNS AND R IS AN UPPER TRIANGULAR MATRIX,
C     THEN DOGLEG EXPECTS THE FULL UPPER TRIANGLE OF R AND
C     THE FIRST N COMPONENTS OF (Q TRANSPOSE)*B.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE DOGLEG(N,R,LR,DIAG,QTB,DELTA,X,WA1,WA2)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R.
C
C       R IS AN INPUT ARRAY OF LENGTH LR WHICH MUST CONTAIN THE UPPER
C         TRIANGULAR MATRIX R STORED BY ROWS.
C
C       LR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN
C         (N*(N+1))/2.
C
C       DIAG IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE
C         DIAGONAL ELEMENTS OF THE MATRIX D.
C
C       QTB IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE FIRST
C         N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*B.
C
C       DELTA IS A POSITIVE INPUT VARIABLE WHICH SPECIFIES AN UPPER
C         BOUND ON THE EUCLIDEAN NORM OF D*X.
C
C       X IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE DESIRED
C         CONVEX COMBINATION OF THE GAUSS-NEWTON DIRECTION AND THE
C         SCALED GRADIENT DIRECTION.
C
C       WA1 AND WA2 ARE WORK ARRAYS OF LENGTH N.
C
C     SUBPROGRAMS CALLED
C
C       MINPACK-SUPPLIED ... DPMPAR,ENORM
```

```
C                                                                       DOGL0550
C        FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,DSQRT                    DOGL0560
C                                                                       DOGL0570
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.         DOGL0580
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE             DOGL0590
C                                                                       DOGL0600
C     **********                                                        DOGL0610
      INTEGER I,J,JJ,JP1,K,L                                            DOGL0620
      DOUBLE PRECISION ALPHA,BNORM,EPSMCH,GNORM,ONE,QNORM,SGNORM,SUM,   DOGL0630
     *                 TEMP,ZERO                                        DOGL0640
      DOUBLE PRECISION DPMPAR,ENORM                                     DOGL0650
      DATA ONE,ZERO /1.0D0,0.0D0/                                       DOGL0660
C                                                                       DOGL0670
C     EPSMCH IS THE MACHINE PRECISION.                                  DOGL0680
C                                                                       DOGL0690
      EPSMCH = DPMPAR(1)                                                DOGL0700
C                                                                       DOGL0710
C     FIRST, CALCULATE THE GAUSS-NEWTON DIRECTION.                      DOGL0720
C                                                                       DOGL0730
      JJ = (N*(N + 1))/2 + 1                                            DOGL0740
      DO 50 K = 1, N                                                    DOGL0750
         J = N - K + 1                                                  DOGL0760
         JP1 = J + 1                                                    DOGL0770
         JJ = JJ - K                                                    DOGL0780
         L = JJ + 1                                                     DOGL0790
         SUM = ZERO                                                     DOGL0800
         IF (N .LT. JP1) GO TO 20                                       DOGL0810
         DO 10 I = JP1, N                                               DOGL0820
            SUM = SUM + R(L)*X(I)                                       DOGL0830
            L = L + 1                                                   DOGL0840
   10       CONTINUE                                                    DOGL0850
   20    CONTINUE                                                       DOGL0860
         TEMP = R(JJ)                                                   DOGL0870
         IF (TEMP .NE. ZERO) GO TO 40                                   DOGL0880
         L = J                                                          DOGL0890
         DO 30 I = 1, J                                                 DOGL0900
            TEMP = DMAX1(TEMP,DABS(R(L)))                               DOGL0910
            L = L + N - I                                               DOGL0920
   30       CONTINUE                                                    DOGL0930
         TEMP = EPSMCH*TEMP                                             DOGL0940
         IF (TEMP .EQ. ZERO) TEMP = EPSMCH                              DOGL0950
   40    CONTINUE                                                       DOGL0960
         X(J) = (QTB(J) - SUM)/TEMP                                     DOGL0970
   50    CONTINUE                                                       DOGL0980
C                                                                       DOGL0990
C     TEST WHETHER THE GAUSS-NEWTON DIRECTION IS ACCEPTABLE.            DOGL1000
C                                                                       DOGL1010
      DO 60 J = 1, N                                                    DOGL1020
         WA1(J) = ZERO                                                  DOGL1030
         WA2(J) = DIAG(J)*X(J)                                          DOGL1040
   60    CONTINUE                                                       DOGL1050
      QNORM = ENORM(N,WA2)                                              DOGL1060
      IF (QNORM .LE. DELTA) GO TO 140                                   DOGL1070
C                                                                       DOGL1080
```

```
C     THE GAUSS-NEWTON DIRECTION IS NOT ACCEPTABLE.
C     NEXT, CALCULATE THE SCALED GRADIENT DIRECTION.
C
      L = 1
      DO 80 J = 1, N
         TEMP = QTB(J)
         DO 70 I = J, N
            WA1(I) = WA1(I) + R(L)*TEMP
            L = L + 1
   70       CONTINUE
         WA1(J) = WA1(J)/DIAG(J)
   80    CONTINUE
C
C     CALCULATE THE NORM OF THE SCALED GRADIENT AND TEST FOR
C     THE SPECIAL CASE IN WHICH THE SCALED GRADIENT IS ZERO.
C
      GNORM = ENORM(N,WA1)
      SGNORM = ZERO
      ALPHA = DELTA/QNORM
      IF (GNORM .EQ. ZERO) GO TO 120
C
C     CALCULATE THE POINT ALONG THE SCALED GRADIENT
C     AT WHICH THE QUADRATIC IS MINIMIZED.
C
      DO 90 J = 1, N
         WA1(J) = (WA1(J)/GNORM)/DIAG(J)
   90    CONTINUE
      L = 1
      DO 110 J = 1, N
         SUM = ZERO
         DO 100 I = J, N
            SUM = SUM + R(L)*WA1(I)
            L = L + 1
  100       CONTINUE
         WA2(J) = SUM
  110    CONTINUE
      TEMP = ENORM(N,WA2)
      SGNORM = (GNORM/TEMP)/TEMP
C
C     TEST WHETHER THE SCALED GRADIENT DIRECTION IS ACCEPTABLE.
C
      ALPHA = ZERO
      IF (SGNORM .GE. DELTA) GO TO 120
C
C     THE SCALED GRADIENT DIRECTION IS NOT ACCEPTABLE.
C     FINALLY, CALCULATE THE POINT ALONG THE DOGLEG
C     AT WHICH THE QUADRATIC IS MINIMIZED.
C
      BNORM = ENORM(N,QTB)
      TEMP = (BNORM/GNORM)*(BNORM/QNORM)*(SGNORM/DELTA)
      TEMP = TEMP - (DELTA/QNORM)*(SGNORM/DELTA)**2
     *       + DSQRT((TEMP-(DELTA/QNORM))**2
     *               +(ONE-(DELTA/QNORM)**2)*(ONE-(SGNORM/DELTA)**2))
      ALPHA = ((DELTA/QNORM)*(ONE - (SGNORM/DELTA)**2))/TEMP
```

```
  120 CONTINUE
C
C     FORM APPROPRIATE CONVEX COMBINATION OF THE GAUSS-NEWTON
C     DIRECTION AND THE SCALED GRADIENT DIRECTION.
C
      TEMP = (ONE - ALPHA)*DMIN1(SGNORM,DELTA)
      DO 130 J = 1, N
         X(J) = TEMP*WA1(J) + ALPHA*X(J)
  130    CONTINUE
  140 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE DOGLEG.
C
      END
```

```
      DOUBLE PRECISION FUNCTION ENORM(N,X)
      INTEGER N
      DOUBLE PRECISION X(N)
C     **********
C
C     FUNCTION ENORM
C
C     GIVEN AN N-VECTOR X, THIS FUNCTION CALCULATES THE
C     EUCLIDEAN NORM OF X.
C
C     THE EUCLIDEAN NORM IS COMPUTED BY ACCUMULATING THE SUM OF
C     SQUARES IN THREE DIFFERENT SUMS. THE SUMS OF SQUARES FOR THE
C     SMALL AND LARGE COMPONENTS ARE SCALED SO THAT NO OVERFLOWS
C     OCCUR. NON-DESTRUCTIVE UNDERFLOWS ARE PERMITTED. UNDERFLOWS
C     AND OVERFLOWS DO NOT OCCUR IN THE COMPUTATION OF THE UNSCALED
C     SUM OF SQUARES FOR THE INTERMEDIATE COMPONENTS.
C     THE DEFINITIONS OF SMALL, INTERMEDIATE AND LARGE COMPONENTS
C     DEPEND ON TWO CONSTANTS, RDWARF AND RGIANT. THE MAIN
C     RESTRICTIONS ON THESE CONSTANTS ARE THAT RDWARF**2 NOT
C     UNDERFLOW AND RGIANT**2 NOT OVERFLOW. THE CONSTANTS
C     GIVEN HERE ARE SUITABLE FOR EVERY KNOWN COMPUTER.
C
C     THE FUNCTION STATEMENT IS
C
C       DOUBLE PRECISION FUNCTION ENORM(N,X)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE.
C
C       X IS AN INPUT ARRAY OF LENGTH N.
C
C     SUBPROGRAMS CALLED
C
C       FORTRAN-SUPPLIED ... DABS,DSQRT
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I
      DOUBLE PRECISION AGIANT,FLOATN,ONE,RDWARF,RGIANT,S1,S2,S3,XABS,
     *                 X1MAX,X3MAX,ZERO
      DATA ONE,ZERO,RDWARF,RGIANT /1.0D0,0.0D0,3.834D-20,1.304D19/
      S1 = ZERO
      S2 = ZERO
      S3 = ZERO
      X1MAX = ZERO
      X3MAX = ZERO
      FLOATN = N
      AGIANT = RGIANT/FLOATN
      DO 90 I = 1, N
         XABS = DABS(X(I))
         IF (XABS .GT. RDWARF .AND. XABS .LT. AGIANT) GO TO 70
```

```
            IF (XABS .LE. RDWARF) GO TO 30
C
C              SUM FOR LARGE COMPONENTS.
C
               IF (XABS .LE. X1MAX) GO TO 10
                  S1 = ONE + S1*(X1MAX/XABS)**2
                  X1MAX = XABS
                  GO TO 20
   10          CONTINUE
                  S1 = S1 + (XABS/X1MAX)**2
   20          CONTINUE
               GO TO 60
   30       CONTINUE
C
C              SUM FOR SMALL COMPONENTS.
C
               IF (XABS .LE. X3MAX) GO TO 40
                  S3 = ONE + S3*(X3MAX/XABS)**2
                  X3MAX = XABS
                  GO TO 50
   40          CONTINUE
                  IF (XABS .NE. ZERO) S3 = S3 + (XABS/X3MAX)**2
   50          CONTINUE
   60       CONTINUE
            GO TO 80
   70    CONTINUE
C
C           SUM FOR INTERMEDIATE COMPONENTS.
C
            S2 = S2 + XABS**2
   80    CONTINUE
   90    CONTINUE
C
C     CALCULATION OF NORM.
C
      IF (S1 .EQ. ZERO) GO TO 100
         ENORM = X1MAX*DSQRT(S1+(S2/X1MAX)/X1MAX)
         GO TO 130
  100 CONTINUE
         IF (S2 .EQ. ZERO) GO TO 110
            IF (S2 .GE. X3MAX)
     *         ENORM = DSQRT(S2*(ONE+(X3MAX/S2)*(X3MAX*S3)))
            IF (S2 .LT. X3MAX)
     *         ENORM = DSQRT(X3MAX*((S2/X3MAX)+(X3MAX*S3)))
            GO TO 120
  110    CONTINUE
            ENORM = X3MAX*DSQRT(S3)
  120    CONTINUE
  130 CONTINUE
      RETURN
C
C     LAST CARD OF FUNCTION ENORM.
C
      END
```

```
      SUBROUTINE FDJAC1(FCN,N,X,FVEC,FJAC,LDFJAC,IFLAG,ML,MU,EPSFCN,
     *                  WA1,WA2)
      INTEGER N,LDFJAC,IFLAG,ML,MU
      DOUBLE PRECISION EPSFCN
      DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N),WA1(N),WA2(N)
C     **********
C
C     SUBROUTINE FDJAC1
C
C     THIS SUBROUTINE COMPUTES A FORWARD-DIFFERENCE APPROXIMATION
C     TO THE N BY N JACOBIAN MATRIX ASSOCIATED WITH A SPECIFIED
C     PROBLEM OF N FUNCTIONS IN N VARIABLES. IF THE JACOBIAN HAS
C     A BANDED FORM, THEN FUNCTION EVALUATIONS ARE SAVED BY ONLY
C     APPROXIMATING THE NONZERO TERMS.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE FDJAC1(FCN,N,X,FVEC,FJAC,LDFJAC,IFLAG,ML,MU,EPSFCN,
C                         WA1,WA2)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED
C         IN AN EXTERNAL STATEMENT IN THE USER CALLING
C         PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(N,X,FVEC,IFLAG)
C         INTEGER N,IFLAG
C         DOUBLE PRECISION X(N),FVEC(N)
C         ----------
C         CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC.
C         ----------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF FDJAC1.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS AND VARIABLES.
C
C       X IS AN INPUT ARRAY OF LENGTH N.
C
C       FVEC IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE
C         FUNCTIONS EVALUATED AT X.
C
C       FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE
C         APPROXIMATION TO THE JACOBIAN MATRIX EVALUATED AT X.
C
C       LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
```

```
C
C        IFLAG IS AN INTEGER VARIABLE WHICH CAN BE USED TO TERMINATE
C          THE EXECUTION OF FDJAC1. SEE DESCRIPTION OF FCN.
C
C        ML IS A NONNEGATIVE INTEGER INPUT VARIABLE WHICH SPECIFIES
C          THE NUMBER OF SUBDIAGONALS WITHIN THE BAND OF THE
C          JACOBIAN MATRIX. IF THE JACOBIAN IS NOT BANDED, SET
C          ML TO AT LEAST N - 1.
C
C        EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE
C          STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS
C          APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE
C          FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS
C          THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE
C          ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE
C          PRECISION.
C
C        MU IS A NONNEGATIVE INTEGER INPUT VARIABLE WHICH SPECIFIES
C          THE NUMBER OF SUPERDIAGONALS WITHIN THE BAND OF THE
C          JACOBIAN MATRIX. IF THE JACOBIAN IS NOT BANDED, SET
C          MU TO AT LEAST N - 1.
C
C        WA1 AND WA2 ARE WORK ARRAYS OF LENGTH N. IF ML + MU + 1 IS AT
C          LEAST N, THEN THE JACOBIAN IS CONSIDERED DENSE, AND WA2 IS
C          NOT REFERENCED.
C
C     SUBPROGRAMS CALLED
C
C       MINPACK-SUPPLIED ... DPMPAR
C
C       FORTRAN-SUPPLIED ... DABS,DMAX1,DSQRT
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,J,K,MSUM
      DOUBLE PRECISION EPS,EPSMCH,H,TEMP,ZERO
      DOUBLE PRECISION DPMPAR
      DATA ZERO /0.0D0/
C
C     EPSMCH IS THE MACHINE PRECISION.
C
      EPSMCH = DPMPAR(1)
C
      EPS = DSQRT(DMAX1(EPSFCN,EPSMCH))
      MSUM = ML + MU + 1
      IF (MSUM .LT. N) GO TO 40
C
C        COMPUTATION OF DENSE APPROXIMATE JACOBIAN.
C
         DO 20 J = 1, N
            TEMP = X(J)
            H = EPS*DABS(TEMP)
```

```
            IF (H .EQ. ZERO) H = EPS                                      FDJ11090
            X(J) = TEMP + H                                               FDJ11100
            CALL FCN(N,X,WA1,IFLAG)                                       FDJ11110
            IF (IFLAG .LT. 0) GO TO 30                                    FDJ11120
            X(J) = TEMP                                                   FDJ11130
            DO 10 I = 1, N                                                FDJ11140
               FJAC(I,J) = (WA1(I) - FVEC(I))/H                           FDJ11150
   10          CONTINUE                                                   FDJ11160
   20       CONTINUE                                                      FDJ11170
   30    CONTINUE                                                         FDJ11180
         GO TO 110                                                        FDJ11190
   40 CONTINUE                                                            FDJ11200
C                                                                         FDJ11210
C        COMPUTATION OF BANDED APPROXIMATE JACOBIAN.                      FDJ11220
C                                                                         FDJ11230
         DO 90 K = 1, MSUM                                                FDJ11240
            DO 60 J = K, N, MSUM                                          FDJ11250
               WA2(J) = X(J)                                              FDJ11260
               H = EPS*DABS(WA2(J))                                       FDJ11270
               IF (H .EQ. ZERO) H = EPS                                   FDJ11280
               X(J) = WA2(J) + H                                          FDJ11290
   60          CONTINUE                                                   FDJ11300
            CALL FCN(N,X,WA1,IFLAG)                                       FDJ11310
            IF (IFLAG .LT. 0) GO TO 100                                   FDJ11320
            DO 80 J = K, N, MSUM                                          FDJ11330
               X(J) = WA2(J)                                              FDJ11340
               H = EPS*DABS(WA2(J))                                       FDJ11350
               IF (H .EQ. ZERO) H = EPS                                   FDJ11360
               DO 70 I = 1, N                                             FDJ11370
                  FJAC(I,J) = ZERO                                        FDJ11380
                  IF (I .GE. J - MU .AND. I .LE. J + ML)                  FDJ11390
     *               FJAC(I,J) = (WA1(I) - FVEC(I))/H                     FDJ11400
   70             CONTINUE                                                FDJ11410
   80          CONTINUE                                                   FDJ11420
   90       CONTINUE                                                      FDJ11430
  100    CONTINUE                                                         FDJ11440
  110 CONTINUE                                                            FDJ11450
      RETURN                                                              FDJ11460
C                                                                         FDJ11470
C     LAST CARD OF SUBROUTINE FDJAC1.                                     FDJ11480
C                                                                         FDJ11490
      END                                                                 FDJ11500
```

```
      SUBROUTINE FDJAC2(FCN,M,N,X,FVEC,FJAC,LDFJAC,IFLAG,EPSFCN,WA)
      INTEGER M,N,LDFJAC,IFLAG
      DOUBLE PRECISION EPSFCN
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),WA(M)
C     **********
C
C     SUBROUTINE FDJAC2
C
C     THIS SUBROUTINE COMPUTES A FORWARD-DIFFERENCE APPROXIMATION
C     TO THE M BY N JACOBIAN MATRIX ASSOCIATED WITH A SPECIFIED
C     PROBLEM OF M FUNCTIONS IN N VARIABLES.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE FDJAC2(FCN,M,N,X,FVEC,FJAC,LDFJAC,IFLAG,EPSFCN,WA)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED
C         IN AN EXTERNAL STATEMENT IN THE USER CALLING
C         PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(M,N,X,FVEC,IFLAG)
C         INTEGER M,N,IFLAG
C         DOUBLE PRECISION X(N),FVEC(M)
C         ----------
C         CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC.
C         ----------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF FDJAC2.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF VARIABLES. N MUST NOT EXCEED M.
C
C       X IS AN INPUT ARRAY OF LENGTH N.
C
C       FVEC IS AN INPUT ARRAY OF LENGTH M WHICH MUST CONTAIN THE
C         FUNCTIONS EVALUATED AT X.
C
C       FJAC IS AN OUTPUT M BY N ARRAY WHICH CONTAINS THE
C         APPROXIMATION TO THE JACOBIAN MATRIX EVALUATED AT X.
C
C       LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
C
```

```
C        IFLAG IS AN INTEGER VARIABLE WHICH CAN BE USED TO TERMINATE    FDJ20550
C          THE EXECUTION OF FDJAC2. SEE DESCRIPTION OF FCN.             FDJ20560
C                                                                       FDJ20570
C        EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE     FDJ20580
C          STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS   FDJ20590
C          APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE        FDJ20600
C          FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS      FDJ20610
C          THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE  FDJ20620
C          ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE      FDJ20630
C          PRECISION.                                                   FDJ20640
C                                                                       FDJ20650
C        WA IS A WORK ARRAY OF LENGTH M.                                FDJ20660
C                                                                       FDJ20670
C      SUBPROGRAMS CALLED                                               FDJ20680
C                                                                       FDJ20690
C        USER-SUPPLIED ...... FCN                                       FDJ20700
C                                                                       FDJ20710
C        MINPACK-SUPPLIED ... DPMPAR                                    FDJ20720
C                                                                       FDJ20730
C        FORTRAN-SUPPLIED ... DABS,DMAX1,DSQRT                          FDJ20740
C                                                                       FDJ20750
C      ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.        FDJ20760
C      BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE            FDJ20770
C                                                                       FDJ20780
C      **********                                                       FDJ20790
      INTEGER I,J                                                       FDJ20800
      DOUBLE PRECISION EPS,EPSMCH,H,TEMP,ZERO                           FDJ20810
      DOUBLE PRECISION DPMPAR                                           FDJ20820
      DATA ZERO /0.0D0/                                                 FDJ20830
C                                                                       FDJ20840
C     EPSMCH IS THE MACHINE PRECISION.                                  FDJ20850
C                                                                       FDJ20860
      EPSMCH = DPMPAR(1)                                                FDJ20870
C                                                                       FDJ20880
      EPS = DSQRT(DMAX1(EPSFCN,EPSMCH))                                 FDJ20890
      DO 20 J = 1, N                                                    FDJ20900
         TEMP = X(J)                                                    FDJ20910
         H = EPS*DABS(TEMP)                                             FDJ20920
         IF (H .EQ. ZERO) H = EPS                                       FDJ20930
         X(J) = TEMP + H                                                FDJ20940
         CALL FCN(M,N,X,WA,IFLAG)                                       FDJ20950
         IF (IFLAG .LT. 0) GO TO 30                                     FDJ20960
         X(J) = TEMP                                                    FDJ20970
         DO 10 I = 1, M                                                 FDJ20980
            FJAC(I,J) = (WA(I) - FVEC(I))/H                             FDJ20990
   10       CONTINUE                                                    FDJ21000
   20    CONTINUE                                                       FDJ21010
   30 CONTINUE                                                          FDJ21020
      RETURN                                                            FDJ21030
C                                                                       FDJ21040
C     LAST CARD OF SUBROUTINE FDJAC2.                                   FDJ21050
C                                                                       FDJ21060
      END                                                               FDJ21070
```

```
      SUBROUTINE HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,DIAG,
     *                 MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,R,LR,
     *                 QTF,WA1,WA2,WA3,WA4)
      INTEGER N,MAXFEV,ML,MU,MODE,NPRINT,INFO,NFEV,LDFJAC,LR
      DOUBLE PRECISION XTOL,EPSFCN,FACTOR
      DOUBLE PRECISION X(N),FVEC(N),DIAG(N),FJAC(LDFJAC,N),R(LR),
     *                 QTF(N),WA1(N),WA2(N),WA3(N),WA4(N)
      EXTERNAL FCN
C     **********
C
C     SUBROUTINE HYBRD
C
C     THE PURPOSE OF HYBRD IS TO FIND A ZERO OF A SYSTEM OF
C     N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION
C     OF THE POWELL HYBRID METHOD. THE USER MUST PROVIDE A
C     SUBROUTINE WHICH CALCULATES THE FUNCTIONS. THE JACOBIAN IS
C     THEN CALCULATED BY A FORWARD-DIFFERENCE APPROXIMATION.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,
C                        DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,
C                        LDFJAC,R,LR,QTF,WA1,WA2,WA3,WA4)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED
C         IN AN EXTERNAL STATEMENT IN THE USER CALLING
C         PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(N,X,FVEC,IFLAG)
C         INTEGER N,IFLAG
C         DOUBLE PRECISION X(N),FVEC(N)
C         ----------
C         CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC.
C         ---------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF HYBRD.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS AND VARIABLES.
C
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
C         CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.
C
C       FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS
C         THE FUNCTIONS EVALUATED AT THE OUTPUT X.
```

```
C
C       XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION
C         OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE
C         ITERATES IS AT MOST XTOL.
C
C       MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION
C         OCCURS WHEN THE NUMBER OF CALLS TO FCN IS AT LEAST MAXFEV
C         BY THE END OF AN ITERATION.
C
C       ML IS A NONNEGATIVE INTEGER INPUT VARIABLE WHICH SPECIFIES
C         THE NUMBER OF SUBDIAGONALS WITHIN THE BAND OF THE
C         JACOBIAN MATRIX. IF THE JACOBIAN IS NOT BANDED, SET
C         ML TO AT LEAST N - 1.
C
C       MU IS A NONNEGATIVE INTEGER INPUT VARIABLE WHICH SPECIFIES
C         THE NUMBER OF SUPERDIAGONALS WITHIN THE BAND OF THE
C         JACOBIAN MATRIX. IF THE JACOBIAN IS NOT BANDED, SET
C         MU TO AT LEAST N - 1.
C
C       EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE
C         STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS
C         APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE
C         FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS
C         THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE
C         ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE
C         PRECISION.
C
C       DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE
C         BELOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG
C         MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS
C         MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.
C
C       MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE
C         VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,
C         THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER
C         VALUES OF MODE ARE EQUIVALENT TO MODE = 1.
C
C       FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE
C         INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF
C         FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE
C         TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE
C         INTERVAL (.1,100.). 100. IS A GENERALLY RECOMMENDED VALUE.
C
C       NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED
C         PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,
C         FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST
C         ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND
C         IMMEDIATELY PRIOR TO RETURN, WITH X AND FVEC AVAILABLE
C         FOR PRINTING. IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS
C         OF FCN WITH IFLAG = 0 ARE MADE.
C
C       INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS
C         TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)
C         VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,
```

```
C         INFO IS SET AS FOLLOWS.
C
C         INFO = 0   IMPROPER INPUT PARAMETERS.
C
C         INFO = 1   RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES
C                    IS AT MOST XTOL.
C
C         INFO = 2   NUMBER OF CALLS TO FCN HAS REACHED OR EXCEEDED
C                    MAXFEV.
C
C         INFO = 3   XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN
C                    THE APPROXIMATE SOLUTION X IS POSSIBLE.
C
C         INFO = 4   ITERATION IS NOT MAKING GOOD PROGRESS, AS
C                    MEASURED BY THE IMPROVEMENT FROM THE LAST
C                    FIVE JACOBIAN EVALUATIONS.
C
C         INFO = 5   ITERATION IS NOT MAKING GOOD PROGRESS, AS
C                    MEASURED BY THE IMPROVEMENT FROM THE LAST
C                    TEN ITERATIONS.
C
C       NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF
C         CALLS TO FCN.
C
C       FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE
C         ORTHOGONAL MATRIX Q PRODUCED BY THE QR FACTORIZATION
C         OF THE FINAL APPROXIMATE JACOBIAN.
C
C       LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
C
C       R IS AN OUTPUT ARRAY OF LENGTH LR WHICH CONTAINS THE
C         UPPER TRIANGULAR MATRIX PRODUCED BY THE QR FACTORIZATION
C         OF THE FINAL APPROXIMATE JACOBIAN, STORED ROWWISE.
C
C       LR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN
C         (N*(N+1))/2.
C
C       QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS
C         THE VECTOR (Q TRANSPOSE)*FVEC.
C
C       WA1, WA2, WA3, AND WA4 ARE WORK ARRAYS OF LENGTH N.
C
C     SUBPROGRAMS CALLED
C
C       USER-SUPPLIED ...... FCN
C
C       MINPACK-SUPPLIED ... DOGLEG,DPMPAR,ENORM,FDJAC1,
C                            QFORM,QRFAC,R1MPYQ,R1UPDT
C
C       FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,MIN0,MOD
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
```

```
C
C     **********
      INTEGER I,IFLAG,ITER,J,JM1,L,MSUM,NCFAIL,NCSUC,NSLOW1,NSLOW2
      INTEGER IWA(1)
      LOGICAL JEVAL,SING
      DOUBLE PRECISION ACTRED,DELTA,EPSMCH,FNORM,FNORM1,ONE,PNORM,
     *                 PRERED,P1,P5,P001,P0001,RATIO,SUM,TEMP,XNORM,
     *                 ZERO
      DOUBLE PRECISION DPMPAR,ENORM
      DATA ONE,P1,P5,P001,P0001,ZERO
     *     /1.0D0,1.0D-1,5.0D-1,1.0D-3,1.0D-4,0.0D0/
C
C     EPSMCH IS THE MACHINE PRECISION.
C
      EPSMCH = DPMPAR(1)
C
      INFO = 0
      IFLAG = 0
      NFEV = 0
C
C     CHECK THE INPUT PARAMETERS FOR ERRORS.
C
      IF (N .LE. 0 .OR. XTOL .LT. ZERO .OR. MAXFEV .LE. 0
     *    .OR. ML .LT. 0 .OR. MU .LT. 0 .OR. FACTOR .LE. ZERO
     *    .OR. LDFJAC .LT. N .OR. LR .LT. (N*(N + 1))/2) GO TO 300
      IF (MODE .NE. 2) GO TO 20
      DO 10 J = 1, N
         IF (DIAG(J) .LE. ZERO) GO TO 300
   10    CONTINUE
   20 CONTINUE
C
C     EVALUATE THE FUNCTION AT THE STARTING POINT
C     AND CALCULATE ITS NORM.
C
      IFLAG = 1
      CALL FCN(N,X,FVEC,IFLAG)
      NFEV = 1
      IF (IFLAG .LT. 0) GO TO 300
      FNORM = ENORM(N,FVEC)
C
C     DETERMINE THE NUMBER OF CALLS TO FCN NEEDED TO COMPUTE
C     THE JACOBIAN MATRIX.
C
      MSUM = MIN0(ML+MU+1,N)
C
C     INITIALIZE ITERATION COUNTER AND MONITORS.
C
      ITER = 1
      NCSUC = 0
      NCFAIL = 0
      NSLOW1 = 0
      NSLOW2 = 0
C
C     BEGINNING OF THE OUTER LOOP.
```

```
C
   30 CONTINUE
         JEVAL = .TRUE.
C
C        CALCULATE THE JACOBIAN MATRIX.
C
         IFLAG = 2
         CALL FDJAC1(FCN,N,X,FVEC,FJAC,LDFJAC,IFLAG,ML,MU,EPSFCN,WA1,
     *               WA2)
         NFEV = NFEV + MSUM
         IF (IFLAG .LT. 0) GO TO 300
C
C        COMPUTE THE QR FACTORIZATION OF THE JACOBIAN.
C
         CALL QRFAC(N,N,FJAC,LDFJAC,.FALSE.,IWA,1,WA1,WA2,WA3)
C
C        ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING
C        TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN.
C
         IF (ITER .NE. 1) GO TO 70
         IF (MODE .EQ. 2) GO TO 50
         DO 40 J = 1, N
            DIAG(J) = WA2(J)
            IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE
   40       CONTINUE
   50    CONTINUE
C
C        ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X
C        AND INITIALIZE THE STEP BOUND DELTA.
C
         DO 60 J = 1, N
            WA3(J) = DIAG(J)*X(J)
   60       CONTINUE
         XNORM = ENORM(N,WA3)
         DELTA = FACTOR*XNORM
         IF (DELTA .EQ. ZERO) DELTA = FACTOR
   70    CONTINUE
C
C        FORM (Q TRANSPOSE)*FVEC AND STORE IN QTF.
C
         DO 80 I = 1, N
            QTF(I) = FVEC(I)
   80       CONTINUE
         DO 120 J = 1, N
            IF (FJAC(J,J) .EQ. ZERO) GO TO 110
            SUM = ZERO
            DO 90 I = J, N
               SUM = SUM + FJAC(I,J)*QTF(I)
   90          CONTINUE
            TEMP = -SUM/FJAC(J,J)
            DO 100 I = J, N
               QTF(I) = QTF(I) + FJAC(I,J)*TEMP
  100          CONTINUE
  110       CONTINUE
```

```
  120       CONTINUE
C
C        COPY THE TRIANGULAR FACTOR OF THE QR FACTORIZATION INTO R.
C
         SING = .FALSE.
         DO 150 J = 1, N
            L = J
            JM1 = J - 1
            IF (JM1 .LT. 1) GO TO 140
            DO 130 I = 1, JM1
               R(L) = FJAC(I,J)
               L = L + N - I
  130          CONTINUE
  140       CONTINUE
            R(L) = WA1(J)
            IF (WA1(J) .EQ. ZERO) SING = .TRUE.
  150       CONTINUE
C
C        ACCUMULATE THE ORTHOGONAL FACTOR IN FJAC.
C
         CALL QFORM(N,N,FJAC,LDFJAC,WA1)
C
C        RESCALE IF NECESSARY.
C
         IF (MODE .EQ. 2) GO TO 170
         DO 160 J = 1, N
            DIAG(J) = DMAX1(DIAG(J),WA2(J))
  160       CONTINUE
  170    CONTINUE
C
C        BEGINNING OF THE INNER LOOP.
C
  180    CONTINUE
C
C           IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.
C
            IF (NPRINT .LE. 0) GO TO 190
            IFLAG = 0
            IF (MOD(ITER-1,NPRINT) .EQ. 0) CALL FCN(N,X,FVEC,IFLAG)
            IF (IFLAG .LT. 0) GO TO 300
  190       CONTINUE
C
C           DETERMINE THE DIRECTION P.
C
            CALL DOGLEG(N,R,LR,DIAG,QTF,DELTA,WA1,WA2,WA3)
C
C           STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.
C
            DO 200 J = 1, N
               WA1(J) = -WA1(J)
               WA2(J) = X(J) + WA1(J)
               WA3(J) = DIAG(J)*WA1(J)
  200          CONTINUE
            PNORM = ENORM(N,WA3)
```

```
C
C        ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.
C
         IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)
C
C        EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.
C
         IFLAG = 1
         CALL FCN(N,WA2,WA4,IFLAG)
         NFEV = NFEV + 1
         IF (IFLAG .LT. 0) GO TO 300
         FNORM1 = ENORM(N,WA4)
C
C        COMPUTE THE SCALED ACTUAL REDUCTION.
C
         ACTRED = -ONE
         IF (FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2
C
C        COMPUTE THE SCALED PREDICTED REDUCTION.
C
         L = 1
         DO 220 I = 1, N
            SUM = ZERO
            DO 210 J = I, N
               SUM = SUM + R(L)*WA1(J)
               L = L + 1
  210          CONTINUE
            WA3(I) = QTF(I) + SUM
  220       CONTINUE
         TEMP = ENORM(N,WA3)
         PRERED = ZERO
         IF (TEMP .LT. FNORM) PRERED = ONE - (TEMP/FNORM)**2
C
C        COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED
C        REDUCTION.
C
         RATIO = ZERO
         IF (PRERED .GT. ZERO) RATIO = ACTRED/PRERED
C
C        UPDATE THE STEP BOUND.
C
         IF (RATIO .GE. P1) GO TO 230
            NCSUC = 0
            NCFAIL = NCFAIL + 1
            DELTA = P5*DELTA
            GO TO 240
  230    CONTINUE
            NCFAIL = 0
            NCSUC = NCSUC + 1
            IF (RATIO .GE. P5 .OR. NCSUC .GT. 1)
     *         DELTA = DMAX1(DELTA,PNORM/P5)
            IF (DABS(RATIO-ONE) .LE. P1) DELTA = PNORM/P5
  240    CONTINUE
C
```

```
C          TEST FOR SUCCESSFUL ITERATION.
C
           IF (RATIO .LT. P0001) GO TO 260
C
C          SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.
C
           DO 250 J = 1, N
              X(J) = WA2(J)
              WA2(J) = DIAG(J)*X(J)
              FVEC(J) = WA4(J)
  250         CONTINUE
           XNORM = ENORM(N,WA2)
           FNORM = FNORM1
           ITER = ITER + 1
  260      CONTINUE
C
C          DETERMINE THE PROGRESS OF THE ITERATION.
C
           NSLOW1 = NSLOW1 + 1
           IF (ACTRED .GE. P001) NSLOW1 = 0
           IF (JEVAL) NSLOW2 = NSLOW2 + 1
           IF (ACTRED .GE. P1) NSLOW2 = 0
C
C          TEST FOR CONVERGENCE.
C
           IF (DELTA .LE. XTOL*XNORM .OR. FNORM .EQ. ZERO) INFO = 1
           IF (INFO .NE. 0) GO TO 300
C
C          TESTS FOR TERMINATION AND STRINGENT TOLERANCES.
C
           IF (NFEV .GE. MAXFEV) INFO = 2
           IF (P1*DMAX1(P1*DELTA,PNORM) .LE. EPSMCH*XNORM) INFO = 3
           IF (NSLOW2 .EQ. 5) INFO = 4
           IF (NSLOW1 .EQ. 10) INFO = 5
           IF (INFO .NE. 0) GO TO 300
C
C          CRITERION FOR RECALCULATING JACOBIAN APPROXIMATION
C          BY FORWARD DIFFERENCES.
C
           IF (NCFAIL .EQ. 2) GO TO 290
C
C          CALCULATE THE RANK ONE MODIFICATION TO THE JACOBIAN
C          AND UPDATE QTF IF NECESSARY.
C
           DO 280 J = 1, N
              SUM = ZERO
              DO 270 I = 1, N
                 SUM = SUM + FJAC(I,J)*WA4(I)
  270            CONTINUE
              WA2(J) = (SUM - WA3(J))/PNORM
              WA1(J) = DIAG(J)*((DIAG(J)*WA1(J))/PNORM)
              IF (RATIO .GE. P0001) QTF(J) = SUM
  280         CONTINUE
C
```

```
C           COMPUTE THE QR FACTORIZATION OF THE UPDATED JACOBIAN.           HYBD4330
C                                                                           HYBD4340
            CALL R1UPDT(N,N,R,LR,WA1,WA2,WA3,SING)                          HYBD4350
            CALL R1MPYQ(N,N,FJAC,LDFJAC,WA2,WA3)                            HYBD4360
            CALL R1MPYQ(1,N,QTF,1,WA2,WA3)                                  HYBD4370
C                                                                           HYBD4380
C           END OF THE INNER LOOP.                                          HYBD4390
C                                                                           HYBD4400
            JEVAL = .FALSE.                                                 HYBD4410
            GO TO 180                                                       HYBD4420
  290    CONTINUE                                                           HYBD4430
C                                                                           HYBD4440
C        END OF THE OUTER LOOP.                                             HYBD4450
C                                                                           HYBD4460
         GO TO 30                                                           HYBD4470
  300 CONTINUE                                                              HYBD4480
C                                                                           HYBD4490
C     TERMINATION, EITHER NORMAL OR USER IMPOSED.                           HYBD4500
C                                                                           HYBD4510
      IF (IFLAG .LT. 0) INFO = IFLAG                                        HYBD4520
      IFLAG = 0                                                             HYBD4530
      IF (NPRINT .GT. 0) CALL FCN(N,X,FVEC,IFLAG)                           HYBD4540
      RETURN                                                                HYBD4550
C                                                                           HYBD4560
C     LAST CARD OF SUBROUTINE HYBRD.                                        HYBD4570
C                                                                           HYBD4580
      END                                                                   HYBD4590
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      SUBROUTINE HYBRD1(FCN,N,X,FVEC,TOL,INFO,WA,LWA)
      INTEGER N,INFO,LWA
      DOUBLE PRECISION TOL
      DOUBLE PRECISION X(N),FVEC(N),WA(LWA)
      EXTERNAL FCN
C     **********
C
C     SUBROUTINE HYBRD1
C
C     THE PURPOSE OF HYBRD1 IS TO FIND A ZERO OF A SYSTEM OF
C     N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION
C     OF THE POWELL HYBRID METHOD. THIS IS DONE BY USING THE
C     MORE GENERAL NONLINEAR EQUATION SOLVER HYBRD. THE USER
C     MUST PROVIDE A SUBROUTINE WHICH CALCULATES THE FUNCTIONS.
C     THE JACOBIAN IS THEN CALCULATED BY A FORWARD-DIFFERENCE
C     APPROXIMATION.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE HYBRD1(FCN,N,X,FVEC,TOL,INFO,WA,LWA)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED
C         IN AN EXTERNAL STATEMENT IN THE USER CALLING
C         PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(N,X,FVEC,IFLAG)
C         INTEGER N,IFLAG
C         DOUBLE PRECISION X(N),FVEC(N)
C         ----------
C         CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC.
C         ---------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF HYBRD1.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS AND VARIABLES.
C
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
C         CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.
C
C       FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS
C         THE FUNCTIONS EVALUATED AT THE OUTPUT X.
C
C       TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS
C         WHEN THE ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
```

```
C         BETWEEN X AND THE SOLUTION IS AT MOST TOL.
C
C       INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS
C         TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)
C         VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,
C         INFO IS SET AS FOLLOWS.
C
C         INFO = 0   IMPROPER INPUT PARAMETERS.
C
C         INFO = 1   ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
C                    BETWEEN X AND THE SOLUTION IS AT MOST TOL.
C
C         INFO = 2   NUMBER OF CALLS TO FCN HAS REACHED OR EXCEEDED
C                    200*(N+1).
C
C         INFO = 3   TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN
C                    THE APPROXIMATE SOLUTION X IS POSSIBLE.
C
C         INFO = 4   ITERATION IS NOT MAKING GOOD PROGRESS.
C
C       WA IS A WORK ARRAY OF LENGTH LWA.
C
C       LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN
C         (N*(3*N+13))/2.
C
C     SUBPROGRAMS CALLED
C
C       USER-SUPPLIED ...... FCN
C
C       MINPACK-SUPPLIED ... HYBRD
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER INDEX,J,LR,MAXFEV,ML,MODE,MU,NFEV,NPRINT
      DOUBLE PRECISION EPSFCN,FACTOR,ONE,XTOL,ZERO
      DATA FACTOR,ONE,ZERO /1.0D2,1.0D0,0.0D0/
      INFO = 0
C
C     CHECK THE INPUT PARAMETERS FOR ERRORS.
C
      IF (N .LE. 0 .OR. TOL .LT. ZERO .OR. LWA .LT. (N*(3*N + 13))/2)
     *   GO TO 20
C
C     CALL HYBRD.
C
      MAXFEV = 200*(N + 1)
      XTOL = TOL
      ML = N - 1
      MU = N - 1
      EPSFCN = ZERO
      MODE = 2
      DO 10 J = 1, N
```

```
            WA(J) = ONE
   10       CONTINUE
         NPRINT = 0
         LR = (N*(N + 1))/2
         INDEX = 6*N + LR
         CALL HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,WA(1),MODE,
     *              FACTOR,NPRINT,INFO,NFEV,WA(INDEX+1),N,WA(6*N+1),LR,
     *              WA(N+1),WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))
         IF (INFO .EQ. 5) INFO = 4
   20 CONTINUE
         RETURN
C
C     LAST CARD OF SUBROUTINE HYBRD1.
C
         END
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      SUBROUTINE HYBRJ(FCN,N,X,FVEC,FJAC,LDFJAC,XTOL,MAXFEV,DIAG,MODE,
     *                 FACTOR,NPRINT,INFO,NFEV,NJEV,R,LR,QTF,WA1,WA2,
     *                 WA3,WA4)
      INTEGER N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV,LR
      DOUBLE PRECISION XTOL,FACTOR
      DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N),DIAG(N),R(LR),
     *                 QTF(N),WA1(N),WA2(N),WA3(N),WA4(N)
C     **********
C
C     SUBROUTINE HYBRJ
C
C     THE PURPOSE OF HYBRJ IS TO FIND A ZERO OF A SYSTEM OF
C     N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION
C     OF THE POWELL HYBRID METHOD. THE USER MUST PROVIDE A
C     SUBROUTINE WHICH CALCULATES THE FUNCTIONS AND THE JACOBIAN.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE HYBRJ(FCN,N,X,FVEC,FJAC,LDFJAC,XTOL,MAXFEV,DIAG,
C                        MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,R,LR,QTF,
C                        WA1,WA2,WA3,WA4)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS AND THE JACOBIAN. FCN MUST
C         BE DECLARED IN AN EXTERNAL STATEMENT IN THE USER
C         CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
C         INTEGER N,LDFJAC,IFLAG
C         DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)
C         ----------
C         IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC. DO NOT ALTER FJAC.
C         IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND
C         RETURN THIS MATRIX IN FJAC. DO NOT ALTER FVEC.
C         ---------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF HYBRJ.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS AND VARIABLES.
C
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
C         CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.
C
C       FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS
C         THE FUNCTIONS EVALUATED AT THE OUTPUT X.
```

```
C                                                                       HYBJ0550
C       FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE               HYBJ0560
C         ORTHOGONAL MATRIX Q PRODUCED BY THE QR FACTORIZATION          HYBJ0570
C         OF THE FINAL APPROXIMATE JACOBIAN.                            HYBJ0580
C                                                                       HYBJ0590
C       LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N     HYBJ0600
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.      HYBJ0610
C                                                                       HYBJ0620
C       XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION               HYBJ0630
C         OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE        HYBJ0640
C         ITERATES IS AT MOST XTOL.                                     HYBJ0650
C                                                                       HYBJ0660
C       MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION        HYBJ0670
C         OCCURS WHEN THE NUMBER OF CALLS TO FCN WITH IFLAG = 1         HYBJ0680
C         HAS REACHED MAXFEV.                                           HYBJ0690
C                                                                       HYBJ0700
C       DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE                  HYBJ0710
C         BELOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG             HYBJ0720
C         MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS                   HYBJ0730
C         MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.               HYBJ0740
C                                                                       HYBJ0750
C       MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE             HYBJ0760
C         VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,             HYBJ0770
C         THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER             HYBJ0780
C         VALUES OF MODE ARE EQUIVALENT TO MODE = 1.                    HYBJ0790
C                                                                       HYBJ0800
C       FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE     HYBJ0810
C         INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF       HYBJ0820
C         FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE   HYBJ0830
C         TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE      HYBJ0840
C         INTERVAL (.1,100.). 100. IS A GENERALLY RECOMMENDED VALUE.    HYBJ0850
C                                                                       HYBJ0860
C       NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED     HYBJ0870
C         PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,         HYBJ0880
C         FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST    HYBJ0890
C         ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND          HYBJ0900
C         IMMEDIATELY PRIOR TO RETURN, WITH X AND FVEC AVAILABLE        HYBJ0910
C         FOR PRINTING. FVEC AND FJAC SHOULD NOT BE ALTERED.            HYBJ0920
C         IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS OF FCN            HYBJ0930
C         WITH IFLAG = 0 ARE MADE.                                      HYBJ0940
C                                                                       HYBJ0950
C       INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS             HYBJ0960
C         TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)           HYBJ0970
C         VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,            HYBJ0980
C         INFO IS SET AS FOLLOWS.                                       HYBJ0990
C                                                                       HYBJ1000
C         INFO = 0   IMPROPER INPUT PARAMETERS.                         HYBJ1010
C                                                                       HYBJ1020
C         INFO = 1   RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES    HYBJ1030
C                    IS AT MOST XTOL.                                   HYBJ1040
C                                                                       HYBJ1050
C         INFO = 2   NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS          HYBJ1060
C                    REACHED MAXFEV.                                    HYBJ1070
C                                                                       HYBJ1080
```

```
C        INFO = 3   XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN       HYBJ1090
C                   THE APPROXIMATE SOLUTION X IS POSSIBLE.            HYBJ1100
C                                                                      HYBJ1110
C        INFO = 4   ITERATION IS NOT MAKING GOOD PROGRESS, AS          HYBJ1120
C                   MEASURED BY THE IMPROVEMENT FROM THE LAST          HYBJ1130
C                   FIVE JACOBIAN EVALUATIONS.                         HYBJ1140
C                                                                      HYBJ1150
C        INFO = 5   ITERATION IS NOT MAKING GOOD PROGRESS, AS          HYBJ1160
C                   MEASURED BY THE IMPROVEMENT FROM THE LAST          HYBJ1170
C                   TEN ITERATIONS.                                    HYBJ1180
C                                                                      HYBJ1190
C       NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF        HYBJ1200
C         CALLS TO FCN WITH IFLAG = 1.                                 HYBJ1210
C                                                                      HYBJ1220
C       NJEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF        HYBJ1230
C         CALLS TO FCN WITH IFLAG = 2.                                 HYBJ1240
C                                                                      HYBJ1250
C       R IS AN OUTPUT ARRAY OF LENGTH LR WHICH CONTAINS THE           HYBJ1260
C         UPPER TRIANGULAR MATRIX PRODUCED BY THE QR FACTORIZATION     HYBJ1270
C         OF THE FINAL APPROXIMATE JACOBIAN, STORED ROWWISE.           HYBJ1280
C                                                                      HYBJ1290
C       LR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN          HYBJ1300
C         (N*(N+1))/2.                                                 HYBJ1310
C                                                                      HYBJ1320
C       QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS              HYBJ1330
C         THE VECTOR (Q TRANSPOSE)*FVEC.                               HYBJ1340
C                                                                      HYBJ1350
C       WA1, WA2, WA3, AND WA4 ARE WORK ARRAYS OF LENGTH N.            HYBJ1360
C                                                                      HYBJ1370
C     SUBPROGRAMS CALLED                                               HYBJ1380
C                                                                      HYBJ1390
C       USER-SUPPLIED ...... FCN                                       HYBJ1400
C                                                                      HYBJ1410
C       MINPACK-SUPPLIED ... DOGLEG,DPMPAR,ENORM,                      HYBJ1420
C                            QFORM,QRFAC,R1MPYQ,R1UPDT                 HYBJ1430
C                                                                      HYBJ1440
C       FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,MOD                      HYBJ1450
C                                                                      HYBJ1460
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.        HYBJ1470
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE            HYBJ1480
C                                                                      HYBJ1490
C     **********                                                       HYBJ1500
      INTEGER I,IFLAG,ITER,J,JM1,L,NCFAIL,NCSUC,NSLOW1,NSLOW2          HYBJ1510
      INTEGER IWA(1)                                                   HYBJ1520
      LOGICAL JEVAL,SING                                               HYBJ1530
      DOUBLE PRECISION ACTRED,DELTA,EPSMCH,FNORM,FNORM1,ONE,PNORM,     HYBJ1540
     *                 PRERED,P1,P5,P001,P0001,RATIO,SUM,TEMP,XNORM,   HYBJ1550
     *                 ZERO                                            HYBJ1560
      DOUBLE PRECISION DPMPAR,ENORM                                    HYBJ1570
      DATA ONE,P1,P5,P001,P0001,ZERO                                   HYBJ1580
     *     /1.0D0,1.0D-1,5.0D-1,1.0D-3,1.0D-4,0.0D0/                   HYBJ1590
C                                                                      HYBJ1600
C     EPSMCH IS THE MACHINE PRECISION.                                 HYBJ1610
C                                                                      HYBJ1620
```

```
      EPSMCH = DPMPAR(1)
C
      INFO = 0
      IFLAG = 0
      NFEV = 0
      NJEV = 0
C
C     CHECK THE INPUT PARAMETERS FOR ERRORS.
C
      IF (N .LE. 0 .OR. LDFJAC .LT. N .OR. XTOL .LT. ZERO
     *    .OR. MAXFEV .LE. 0 .OR. FACTOR .LE. ZERO
     *    .OR. LR .LT. (N*(N + 1))/2) GO TO 300
      IF (MODE .NE. 2) GO TO 20
      DO 10 J = 1, N
         IF (DIAG(J) .LE. ZERO) GO TO 300
   10    CONTINUE
   20 CONTINUE
C
C     EVALUATE THE FUNCTION AT THE STARTING POINT
C     AND CALCULATE ITS NORM.
C
      IFLAG = 1
      CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
      NFEV = 1
      IF (IFLAG .LT. 0) GO TO 300
      FNORM = ENORM(N,FVEC)
C
C     INITIALIZE ITERATION COUNTER AND MONITORS.
C
      ITER = 1
      NCSUC = 0
      NCFAIL = 0
      NSLOW1 = 0
      NSLOW2 = 0
C
C     BEGINNING OF THE OUTER LOOP.
C
   30 CONTINUE
         JEVAL = .TRUE.
C
C        CALCULATE THE JACOBIAN MATRIX.
C
         IFLAG = 2
         CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
         NJEV = NJEV + 1
         IF (IFLAG .LT. 0) GO TO 300
C
C        COMPUTE THE QR FACTORIZATION OF THE JACOBIAN.
C
         CALL QRFAC(N,N,FJAC,LDFJAC,.FALSE.,IWA,1,WA1,WA2,WA3)
C
C        ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING
C        TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN.
C
```

```
      IF (ITER .NE. 1) GO TO 70                                         HYBJ2170
      IF (MODE .EQ. 2) GO TO 50                                         HYBJ2180
      DO 40 J = 1, N                                                    HYBJ2190
         DIAG(J) = WA2(J)                                               HYBJ2200
         IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE                            HYBJ2210
   40    CONTINUE                                                       HYBJ2220
   50 CONTINUE                                                          HYBJ2230
C                                                                       HYBJ2240
C        ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X     HYBJ2250
C        AND INITIALIZE THE STEP BOUND DELTA.                           HYBJ2260
C                                                                       HYBJ2270
      DO 60 J = 1, N                                                    HYBJ2280
         WA3(J) = DIAG(J)*X(J)                                          HYBJ2290
   60    CONTINUE                                                       HYBJ2300
      XNORM = ENORM(N,WA3)                                              HYBJ2310
      DELTA = FACTOR*XNORM                                              HYBJ2320
      IF (DELTA .EQ. ZERO) DELTA = FACTOR                               HYBJ2330
   70 CONTINUE                                                          HYBJ2340
C                                                                       HYBJ2350
C        FORM (Q TRANSPOSE)*FVEC AND STORE IN QTF.                      HYBJ2360
C                                                                       HYBJ2370
      DO 80 I = 1, N                                                    HYBJ2380
         QTF(I) = FVEC(I)                                               HYBJ2390
   80    CONTINUE                                                       HYBJ2400
      DO 120 J = 1, N                                                   HYBJ2410
         IF (FJAC(J,J) .EQ. ZERO) GO TO 110                             HYBJ2420
         SUM = ZERO                                                     HYBJ2430
         DO 90 I = J, N                                                 HYBJ2440
            SUM = SUM + FJAC(I,J)*QTF(I)                                HYBJ2450
   90       CONTINUE                                                    HYBJ2460
         TEMP = -SUM/FJAC(J,J)                                          HYBJ2470
         DO 100 I = J, N                                                HYBJ2480
            QTF(I) = QTF(I) + FJAC(I,J)*TEMP                            HYBJ2490
  100       CONTINUE                                                    HYBJ2500
  110    CONTINUE                                                       HYBJ2510
  120    CONTINUE                                                       HYBJ2520
C                                                                       HYBJ2530
C        COPY THE TRIANGULAR FACTOR OF THE QR FACTORIZATION INTO R.     HYBJ2540
C                                                                       HYBJ2550
      SING = .FALSE.                                                    HYBJ2560
      DO 150 J = 1, N                                                   HYBJ2570
         L = J                                                          HYBJ2580
         JM1 = J - 1                                                    HYBJ2590
         IF (JM1 .LT. 1) GO TO 140                                      HYBJ2600
         DO 130 I = 1, JM1                                              HYBJ2610
            R(L) = FJAC(I,J)                                            HYBJ2620
            L = L + N - I                                               HYBJ2630
  130       CONTINUE                                                    HYBJ2640
  140    CONTINUE                                                       HYBJ2650
         R(L) = WA1(J)                                                  HYBJ2660
         IF (WA1(J) .EQ. ZERO) SING = .TRUE.                            HYBJ2670
  150    CONTINUE                                                       HYBJ2680
C                                                                       HYBJ2690
C        ACCUMULATE THE ORTHOGONAL FACTOR IN FJAC.                      HYBJ2700
```

```
C
      CALL QFORM(N,N,FJAC,LDFJAC,WA1)
C
C     RESCALE IF NECESSARY.
C
      IF (MODE .EQ. 2) GO TO 170
      DO 160 J = 1, N
         DIAG(J) = DMAX1(DIAG(J),WA2(J))
  160    CONTINUE
  170 CONTINUE
C
C     BEGINNING OF THE INNER LOOP.
C
  180 CONTINUE
C
C        IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.
C
         IF (NPRINT .LE. 0) GO TO 190
         IFLAG = 0
         IF (MOD(ITER-1,NPRINT) .EQ. 0)
     *      CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
         IF (IFLAG .LT. 0) GO TO 300
  190    CONTINUE
C
C        DETERMINE THE DIRECTION P.
C
         CALL DOGLEG(N,R,LR,DIAG,QTF,DELTA,WA1,WA2,WA3)
C
C        STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.
C
         DO 200 J = 1, N
            WA1(J) = -WA1(J)
            WA2(J) = X(J) + WA1(J)
            WA3(J) = DIAG(J)*WA1(J)
  200       CONTINUE
         PNORM = ENORM(N,WA3)
C
C        ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.
C
         IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)
C
C        EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.
C
         IFLAG = 1
         CALL FCN(N,WA2,WA4,FJAC,LDFJAC,IFLAG)
         NFEV = NFEV + 1
         IF (IFLAG .LT. 0) GO TO 300
         FNORM1 = ENORM(N,WA4)
C
C        COMPUTE THE SCALED ACTUAL REDUCTION.
C
         ACTRED = -ONE
         IF (FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2
C
```

```
C
C        COMPUTE THE SCALED PREDICTED REDUCTION.
C
         L = 1
         DO 220 I = 1, N
            SUM = ZERO
            DO 210 J = I, N
               SUM = SUM + R(L)*WA1(J)
               L = L + 1
  210          CONTINUE
            WA3(I) = QTF(I) + SUM
  220       CONTINUE
         TEMP = ENORM(N,WA3)
         PRERED = ZERO
         IF (TEMP .LT. FNORM) PRERED = ONE - (TEMP/FNORM)**2
C
C        COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED
C        REDUCTION.
C
         RATIO = ZERO
         IF (PRERED .GT. ZERO) RATIO = ACTRED/PRERED
C
C        UPDATE THE STEP BOUND.
C
         IF (RATIO .GE. P1) GO TO 230
            NCSUC = 0
            NCFAIL = NCFAIL + 1
            DELTA = P5*DELTA
            GO TO 240
  230    CONTINUE
            NCFAIL = 0
            NCSUC = NCSUC + 1
            IF (RATIO .GE. P5 .OR. NCSUC .GT. 1)
     *         DELTA = DMAX1(DELTA,PNORM/P5)
            IF (DABS(RATIO-ONE) .LE. P1) DELTA = PNORM/P5
  240    CONTINUE
C
C        TEST FOR SUCCESSFUL ITERATION.
C
         IF (RATIO .LT. P0001) GO TO 260
C
C        SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.
C
         DO 250 J = 1, N
            X(J) = WA2(J)
            WA2(J) = DIAG(J)*X(J)
            FVEC(J) = WA4(J)
  250       CONTINUE
         XNORM = ENORM(N,WA2)
         FNORM = FNORM1
         ITER = ITER + 1
  260    CONTINUE
C
C        DETERMINE THE PROGRESS OF THE ITERATION.
C
```

```
      NSLOW1 = NSLOW1 + 1
      IF (ACTRED .GE. P001) NSLOW1 = 0
      IF (JEVAL) NSLOW2 = NSLOW2 + 1
      IF (ACTRED .GE. P1) NSLOW2 = 0
C
C        TEST FOR CONVERGENCE.
C
      IF (DELTA .LE. XTOL*XNORM .OR. FNORM .EQ. ZERO) INFO = 1
      IF (INFO .NE. 0) GO TO 300
C
C        TESTS FOR TERMINATION AND STRINGENT TOLERANCES.
C
      IF (NFEV .GE. MAXFEV) INFO = 2
      IF (P1*DMAX1(P1*DELTA,PNORM) .LE. EPSMCH*XNORM) INFO = 3
      IF (NSLOW2 .EQ. 5) INFO = 4
      IF (NSLOW1 .EQ. 10) INFO = 5
      IF (INFO .NE. 0) GO TO 300
C
C        CRITERION FOR RECALCULATING JACOBIAN.
C
      IF (NCFAIL .EQ. 2) GO TO 290
C
C        CALCULATE THE RANK ONE MODIFICATION TO THE JACOBIAN
C        AND UPDATE QTF IF NECESSARY.
C
      DO 280 J = 1, N
         SUM = ZERO
         DO 270 I = 1, N
            SUM = SUM + FJAC(I,J)*WA4(I)
  270       CONTINUE
         WA2(J) = (SUM - WA3(J))/PNORM
         WA1(J) = DIAG(J)*((DIAG(J)*WA1(J))/PNORM)
         IF (RATIO .GE. P0001) QTF(J) = SUM
  280    CONTINUE
C
C        COMPUTE THE QR FACTORIZATION OF THE UPDATED JACOBIAN.
C
      CALL R1UPDT(N,N,R,LR,WA1,WA2,WA3,SING)
      CALL R1MPYQ(N,N,FJAC,LDFJAC,WA2,WA3)
      CALL R1MPYQ(1,N,QTF,1,WA2,WA3)
C
C        END OF THE INNER LOOP.
C
      JEVAL = .FALSE.
      GO TO 180
  290 CONTINUE
C
C        END OF THE OUTER LOOP.
C
      GO TO 30
  300 CONTINUE
C
C     TERMINATION, EITHER NORMAL OR USER IMPOSED.
C
```

```
      IF (IFLAG .LT. 0) INFO = IFLAG
      IFLAG = 0
      IF (NPRINT .GT. 0) CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
      RETURN
C
C     LAST CARD OF SUBROUTINE HYBRJ.
C
      END
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      SUBROUTINE HYBRJ1(FCN,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,WA,LWA)
      INTEGER N,LDFJAC,INFO,LWA
      DOUBLE PRECISION TOL
      DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N),WA(LWA)
      EXTERNAL FCN
C     **********
C
C     SUBROUTINE HYBRJ1
C
C     THE PURPOSE OF HYBRJ1 IS TO FIND A ZERO OF A SYSTEM OF
C     N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION
C     OF THE POWELL HYBRID METHOD. THIS IS DONE BY USING THE
C     MORE GENERAL NONLINEAR EQUATION SOLVER HYBRJ. THE USER
C     MUST PROVIDE A SUBROUTINE WHICH CALCULATES THE FUNCTIONS
C     AND THE JACOBIAN.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE HYBRJ1(FCN,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,WA,LWA)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS AND THE JACOBIAN. FCN MUST
C         BE DECLARED IN AN EXTERNAL STATEMENT IN THE USER
C         CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
C         INTEGER N,LDFJAC,IFLAG
C         DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)
C         ----------
C         IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC. DO NOT ALTER FJAC.
C         IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND
C         RETURN THIS MATRIX IN FJAC. DO NOT ALTER FVEC.
C         ---------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF HYBRJ1.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS AND VARIABLES.
C
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
C         CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.
C
C       FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS
C         THE FUNCTIONS EVALUATED AT THE OUTPUT X.
C
C       FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE
```

```
C         ORTHOGONAL MATRIX Q PRODUCED BY THE QR FACTORIZATION
C         OF THE FINAL APPROXIMATE JACOBIAN.
C
C       LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
C
C       TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS
C         WHEN THE ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
C         BETWEEN X AND THE SOLUTION IS AT MOST TOL.
C
C       INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS
C         TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)
C         VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,
C         INFO IS SET AS FOLLOWS.
C
C         INFO = 0   IMPROPER INPUT PARAMETERS.
C
C         INFO = 1   ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
C                    BETWEEN X AND THE SOLUTION IS AT MOST TOL.
C
C         INFO = 2   NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS
C                    REACHED 100*(N+1).
C
C         INFO = 3   TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN
C                    THE APPROXIMATE SOLUTION X IS POSSIBLE.
C
C         INFO = 4   ITERATION IS NOT MAKING GOOD PROGRESS.
C
C       WA IS A WORK ARRAY OF LENGTH LWA.
C
C       LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN
C         (N*(N+13))/2.
C
C     SUBPROGRAMS CALLED
C
C       USER-SUPPLIED ...... FCN
C
C       MINPACK-SUPPLIED ... HYBRJ
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER J,LR,MAXFEV,MODE,NFEV,NJEV,NPRINT
      DOUBLE PRECISION FACTOR,ONE,XTOL,ZERO
      DATA FACTOR,ONE,ZERO /1.0D2,1.0D0,0.0D0/
      INFO = 0
C
C     CHECK THE INPUT PARAMETERS FOR ERRORS.
C
      IF (N .LE. 0 .OR. LDFJAC .LT. N .OR. TOL .LT. ZERO
     *    .OR. LWA .LT. (N*(N + 13))/2) GO TO 20
C
C     CALL HYBRJ.
```

```
C
      MAXFEV = 100*(N + 1)
      XTOL = TOL
      MODE = 2
      DO 10 J = 1, N
         WA(J) = ONE
   10    CONTINUE
      NPRINT = 0
      LR = (N*(N + 1))/2
      CALL HYBRJ(FCN,N,X,FVEC,FJAC,LDFJAC,XTOL,MAXFEV,WA(1),MODE,
     *           FACTOR,NPRINT,INFO,NFEV,NJEV,WA(6*N+1),LR,WA(N+1),
     *           WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))
      IF (INFO .EQ. 5) INFO = 4
   20 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE HYBRJ1.
C
      END
```

THIS PAGE WAS INTENTIONALLY LEFT BLANK

```
      SUBROUTINE LMDER(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
     *                 MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,
     *                 IPVT,QTF,WA1,WA2,WA3,WA4)
      INTEGER M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV
      INTEGER IPVT(N)
      DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),DIAG(N),QTF(N),
     *                 WA1(N),WA2(N),WA3(N),WA4(M)
C     **********
C
C     SUBROUTINE LMDER
C
C     THE PURPOSE OF LMDER IS TO MINIMIZE THE SUM OF THE SQUARES OF
C     M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF
C     THE LEVENBERG-MARQUARDT ALGORITHM. THE USER MUST PROVIDE A
C     SUBROUTINE WHICH CALCULATES THE FUNCTIONS AND THE JACOBIAN.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE LMDER(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
C                        MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,
C                        NJEV,IPVT,QTF,WA1,WA2,WA3,WA4)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS AND THE JACOBIAN. FCN MUST
C         BE DECLARED IN AN EXTERNAL STATEMENT IN THE USER
C         CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
C         INTEGER M,N,LDFJAC,IFLAG
C         DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)
C         ----------
C         IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC. DO NOT ALTER FJAC.
C         IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND
C         RETURN THIS MATRIX IN FJAC. DO NOT ALTER FVEC.
C         ----------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF LMDER.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF VARIABLES. N MUST NOT EXCEED M.
C
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
```

```
C           CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.             LMDR0550
C                                                                           LMDR0560
C         FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS                LMDR0570
C           THE FUNCTIONS EVALUATED AT THE OUTPUT X.                        LMDR0580
C                                                                           LMDR0590
C         FJAC IS AN OUTPUT M BY N ARRAY. THE UPPER N BY N SUBMATRIX        LMDR0600
C           OF FJAC CONTAINS AN UPPER TRIANGULAR MATRIX R WITH              LMDR0610
C           DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE SUCH THAT          LMDR0620
C                                                                           LMDR0630
C                  T     T           T                                      LMDR0640
C                 P *(JAC *JAC)*P = R *R,                                   LMDR0650
C                                                                           LMDR0660
C           WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL            LMDR0670
C           CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J)            LMDR0680
C           (SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRAPEZOIDAL       LMDR0690
C           PART OF FJAC CONTAINS INFORMATION GENERATED DURING              LMDR0700
C           THE COMPUTATION OF R.                                           LMDR0710
C                                                                           LMDR0720
C         LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M       LMDR0730
C           WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.        LMDR0740
C                                                                           LMDR0750
C         FTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION                 LMDR0760
C           OCCURS WHEN BOTH THE ACTUAL AND PREDICTED RELATIVE              LMDR0770
C           REDUCTIONS IN THE SUM OF SQUARES ARE AT MOST FTOL.              LMDR0780
C           THEREFORE, FTOL MEASURES THE RELATIVE ERROR DESIRED             LMDR0790
C           IN THE SUM OF SQUARES.                                          LMDR0800
C                                                                           LMDR0810
C         XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION                 LMDR0820
C           OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE          LMDR0830
C           ITERATES IS AT MOST XTOL. THEREFORE, XTOL MEASURES THE          LMDR0840
C           RELATIVE ERROR DESIRED IN THE APPROXIMATE SOLUTION.             LMDR0850
C                                                                           LMDR0860
C         GTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION                 LMDR0870
C           OCCURS WHEN THE COSINE OF THE ANGLE BETWEEN FVEC AND            LMDR0880
C           ANY COLUMN OF THE JACOBIAN IS AT MOST GTOL IN ABSOLUTE          LMDR0890
C           VALUE. THEREFORE, GTOL MEASURES THE ORTHOGONALITY               LMDR0900
C           DESIRED BETWEEN THE FUNCTION VECTOR AND THE COLUMNS             LMDR0910
C           OF THE JACOBIAN.                                                LMDR0920
C                                                                           LMDR0930
C         MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION          LMDR0940
C           OCCURS WHEN THE NUMBER OF CALLS TO FCN WITH IFLAG = 1           LMDR0950
C           HAS REACHED MAXFEV.                                             LMDR0960
C                                                                           LMDR0970
C         DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE                    LMDR0980
C           BELOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG               LMDR0990
C           MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS                     LMDR1000
C           MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.                 LMDR1010
C                                                                           LMDR1020
C         MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE               LMDR1030
C           VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,               LMDR1040
C           THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER               LMDR1050
C           VALUES OF MODE ARE EQUIVALENT TO MODE = 1.                      LMDR1060
C                                                                           LMDR1070
C         FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE       LMDR1080
```

```
C         INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF
C         FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE
C         TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE
C         INTERVAL (.1,100.).100. IS A GENERALLY RECOMMENDED VALUE.
C
C       NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED
C         PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,
C         FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST
C         ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND
C         IMMEDIATELY PRIOR TO RETURN, WITH X, FVEC, AND FJAC
C         AVAILABLE FOR PRINTING. FVEC AND FJAC SHOULD NOT BE
C         ALTERED. IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS
C         OF FCN WITH IFLAG = 0 ARE MADE.
C
C       INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS
C         TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)
C         VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,
C         INFO IS SET AS FOLLOWS.
C
C         INFO = 0  IMPROPER INPUT PARAMETERS.
C
C         INFO = 1  BOTH ACTUAL AND PREDICTED RELATIVE REDUCTIONS
C                   IN THE SUM OF SQUARES ARE AT MOST FTOL.
C
C         INFO = 2  RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES
C                   IS AT MOST XTOL.
C
C         INFO = 3  CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.
C
C         INFO = 4  THE COSINE OF THE ANGLE BETWEEN FVEC AND ANY
C                   COLUMN OF THE JACOBIAN IS AT MOST GTOL IN
C                   ABSOLUTE VALUE.
C
C         INFO = 5  NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS
C                   REACHED MAXFEV.
C
C         INFO = 6  FTOL IS TOO SMALL. NO FURTHER REDUCTION IN
C                   THE SUM OF SQUARES IS POSSIBLE.
C
C         INFO = 7  XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN
C                   THE APPROXIMATE SOLUTION X IS POSSIBLE.
C
C         INFO = 8  GTOL IS TOO SMALL. FVEC IS ORTHOGONAL TO THE
C                   COLUMNS OF THE JACOBIAN TO MACHINE PRECISION.
C
C       NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF
C         CALLS TO FCN WITH IFLAG = 1.
C
C       NJEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF
C         CALLS TO FCN WITH IFLAG = 2.
C
C       IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT
C         DEFINES A PERMUTATION MATRIX P SUCH THAT JAC*P = Q*R,
C         WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS
```

```
C          ORTHOGONAL (NOT STORED), AND R IS UPPER TRIANGULAR
C          WITH DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE.
C          COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.
C
C        QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS
C          THE FIRST N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*FVEC.
C
C        WA1, WA2, AND WA3 ARE WORK ARRAYS OF LENGTH N.
C
C        WA4 IS A WORK ARRAY OF LENGTH M.
C
C      SUBPROGRAMS CALLED
C
C        USER-SUPPLIED ...... FCN
C
C        MINPACK-SUPPLIED ... DPMPAR,ENORM,LMPAR,QRFAC
C
C        FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,DSQRT,MOD
C
C      ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C      BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C      **********
       INTEGER I,IFLAG,ITER,J,L
       DOUBLE PRECISION ACTRED,DELTA,DIRDER,EPSMCH,FNORM,FNORM1,GNORM,
     *                  ONE,PAR,PNORM,PRERED,P1,P5,P25,P75,P0001,RATIO,
     *                  SUM,TEMP,TEMP1,TEMP2,XNORM,ZERO
       DOUBLE PRECISION DPMPAR,ENORM
       DATA ONE,P1,P5,P25,P75,P0001,ZERO
     *     /1.0D0,1.0D-1,5.0D-1,2.5D-1,7.5D-1,1.0D-4,0.0D0/
C
C      EPSMCH IS THE MACHINE PRECISION.
C
       EPSMCH = DPMPAR(1)
C
       INFO = 0
       IFLAG = 0
       NFEV = 0
       NJEV = 0
C
C      CHECK THE INPUT PARAMETERS FOR ERRORS.
C
       IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. M
     *     .OR. FTOL .LT. ZERO .OR. XTOL .LT. ZERO .OR. GTOL .LT. ZERO
     *     .OR. MAXFEV .LE. 0 .OR. FACTOR .LE. ZERO) GO TO 300
       IF (MODE .NE. 2) GO TO 20
       DO 10 J = 1, N
          IF (DIAG(J) .LE. ZERO) GO TO 300
    10    CONTINUE
    20 CONTINUE
C
C      EVALUATE THE FUNCTION AT THE STARTING POINT
C      AND CALCULATE ITS NORM.
C
```

```
      IFLAG = 1
      CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
      NFEV = 1
      IF (IFLAG .LT. 0) GO TO 300
      FNORM = ENORM(M,FVEC)
C
C     INITIALIZE LEVENBERG-MARQUARDT PARAMETER AND ITERATION COUNTER.
C
      PAR = ZERO
      ITER = 1
C
C     BEGINNING OF THE OUTER LOOP.
C
   30 CONTINUE
C
C        CALCULATE THE JACOBIAN MATRIX.
C
         IFLAG = 2
         CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
         NJEV = NJEV + 1
         IF (IFLAG .LT. 0) GO TO 300
C
C        IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.
C
         IF (NPRINT .LE. 0) GO TO 40
         IFLAG = 0
         IF (MOD(ITER-1,NPRINT) .EQ. 0)
     *      CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
         IF (IFLAG .LT. 0) GO TO 300
   40    CONTINUE
C
C        COMPUTE THE QR FACTORIZATION OF THE JACOBIAN.
C
         CALL QRFAC(M,N,FJAC,LDFJAC,.TRUE.,IPVT,N,WA1,WA2,WA3)
C
C        ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING
C        TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN.
C
         IF (ITER .NE. 1) GO TO 80
         IF (MODE .EQ. 2) GO TO 60
         DO 50 J = 1, N
            DIAG(J) = WA2(J)
            IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE
   50       CONTINUE
   60    CONTINUE
C
C        ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X
C        AND INITIALIZE THE STEP BOUND DELTA.
C
         DO 70 J = 1, N
            WA3(J) = DIAG(J)*X(J)
   70       CONTINUE
         XNORM = ENORM(N,WA3)
         DELTA = FACTOR*XNORM
```

```
         IF (DELTA .EQ. ZERO) DELTA = FACTOR
   80    CONTINUE
C
C        FORM (Q TRANSPOSE)*FVEC AND STORE THE FIRST N COMPONENTS IN
C        QTF.
C
         DO 90 I = 1, M
            WA4(I) = FVEC(I)
   90       CONTINUE
         DO 130 J = 1, N
            IF (FJAC(J,J) .EQ. ZERO) GO TO 120
            SUM = ZERO
            DO 100 I = J, M
               SUM = SUM + FJAC(I,J)*WA4(I)
  100          CONTINUE
            TEMP = -SUM/FJAC(J,J)
            DO 110 I = J, M
               WA4(I) = WA4(I) + FJAC(I,J)*TEMP
  110          CONTINUE
  120       CONTINUE
            FJAC(J,J) = WA1(J)
            QTF(J) = WA4(J)
  130       CONTINUE
C
C        COMPUTE THE NORM OF THE SCALED GRADIENT.
C
         GNORM = ZERO
         IF (FNORM .EQ. ZERO) GO TO 170
         DO 160 J = 1, N
            L = IPVT(J)
            IF (WA2(L) .EQ. ZERO) GO TO 150
            SUM = ZERO
            DO 140 I = 1, J
               SUM = SUM + FJAC(I,J)*(QTF(I)/FNORM)
  140          CONTINUE
            GNORM = DMAX1(GNORM,DABS(SUM/WA2(L)))
  150       CONTINUE
  160       CONTINUE
  170    CONTINUE
C
C        TEST FOR CONVERGENCE OF THE GRADIENT NORM.
C
         IF (GNORM .LE. GTOL) INFO = 4
         IF (INFO .NE. 0) GO TO 300
C
C        RESCALE IF NECESSARY.
C
         IF (MODE .EQ. 2) GO TO 190
         DO 180 J = 1, N
            DIAG(J) = DMAX1(DIAG(J),WA2(J))
  180       CONTINUE
  190    CONTINUE
C
C        BEGINNING OF THE INNER LOOP.
```

```
C
  200    CONTINUE
C
C           DETERMINE THE LEVENBERG-MARQUARDT PARAMETER.
C
         CALL LMPAR(N,FJAC,LDFJAC,IPVT,DIAG,QTF,DELTA,PAR,WA1,WA2,
     *              WA3,WA4)
C
C           STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.
C
         DO 210 J = 1, N
            WA1(J) = -WA1(J)
            WA2(J) = X(J) + WA1(J)
            WA3(J) = DIAG(J)*WA1(J)
  210       CONTINUE
         PNORM = ENORM(N,WA3)
C
C           ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.
C
         IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)
C
C           EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.
C
         IFLAG = 1
         CALL FCN(M,N,WA2,WA4,FJAC,LDFJAC,IFLAG)
         NFEV = NFEV + 1
         IF (IFLAG .LT. 0) GO TO 300
         FNORM1 = ENORM(M,WA4)
C
C           COMPUTE THE SCALED ACTUAL REDUCTION.
C
         ACTRED = -ONE
         IF (P1*FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2
C
C           COMPUTE THE SCALED PREDICTED REDUCTION AND
C           THE SCALED DIRECTIONAL DERIVATIVE.
C
         DO 230 J = 1, N
            WA3(J) = ZERO
            L = IPVT(J)
            TEMP = WA1(L)
            DO 220 I = 1, J
               WA3(I) = WA3(I) + FJAC(I,J)*TEMP
  220          CONTINUE
  230       CONTINUE
         TEMP1 = ENORM(N,WA3)/FNORM
         TEMP2 = (DSQRT(PAR)*PNORM)/FNORM
         PRERED = TEMP1**2 + TEMP2**2/P5
         DIRDER = -(TEMP1**2 + TEMP2**2)
C
C           COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED
C           REDUCTION.
C
         RATIO = ZERO
```

```
      IF (PRERED .NE. ZERO) RATIO = ACTRED/PRERED
C
C        UPDATE THE STEP BOUND.
C
      IF (RATIO .GT. P25) GO TO 240
         IF (ACTRED .GE. ZERO) TEMP = P5
         IF (ACTRED .LT. ZERO)
     *      TEMP = P5*DIRDER/(DIRDER + P5*ACTRED)
         IF (P1*FNORM1 .GE. FNORM .OR. TEMP .LT. P1) TEMP = P1
         DELTA = TEMP*DMIN1(DELTA,PNORM/P1)
         PAR = PAR/TEMP
         GO TO 260
  240 CONTINUE
         IF (PAR .NE. ZERO .AND. RATIO .LT. P75) GO TO 250
         DELTA = PNORM/P5
         PAR = P5*PAR
  250    CONTINUE
  260 CONTINUE
C
C        TEST FOR SUCCESSFUL ITERATION.
C
      IF (RATIO .LT. P0001) GO TO 290
C
C        SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.
C
      DO 270 J = 1, N
         X(J) = WA2(J)
         WA2(J) = DIAG(J)*X(J)
  270    CONTINUE
      DO 280 I = 1, M
         FVEC(I) = WA4(I)
  280    CONTINUE
      XNORM = ENORM(N,WA2)
      FNORM = FNORM1
      ITER = ITER + 1
  290 CONTINUE
C
C        TESTS FOR CONVERGENCE.
C
      IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL
     *    .AND. P5*RATIO .LE. ONE) INFO = 1
      IF (DELTA .LE. XTOL*XNORM) INFO = 2
      IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL
     *    .AND. P5*RATIO .LE. ONE .AND. INFO .EQ. 2) INFO = 3
      IF (INFO .NE. 0) GO TO 300
C
C        TESTS FOR TERMINATION AND STRINGENT TOLERANCES.
C
      IF (NFEV .GE. MAXFEV) INFO = 5
      IF (DABS(ACTRED) .LE. EPSMCH .AND. PRERED .LE. EPSMCH
     *    .AND. P5*RATIO .LE. ONE) INFO = 6
      IF (DELTA .LE. EPSMCH*XNORM) INFO = 7
      IF (GNORM .LE. EPSMCH) INFO = 8
      IF (INFO .NE. 0) GO TO 300
```

```
C
C        END OF THE INNER LOOP. REPEAT IF ITERATION UNSUCCESSFUL.
C
         IF (RATIO .LT. P0001) GO TO 200
C
C     END OF THE OUTER LOOP.
C
      GO TO 30
  300 CONTINUE
C
C     TERMINATION, EITHER NORMAL OR USER IMPOSED.
C
      IF (IFLAG .LT. 0) INFO = IFLAG
      IFLAG = 0
      IF (NPRINT .GT. 0) CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
      RETURN
C
C     LAST CARD OF SUBROUTINE LMDER.
C
      END
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      SUBROUTINE LMDER1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,IPVT,WA,
     *                  LWA)
      INTEGER M,N,LDFJAC,INFO,LWA
      INTEGER IPVT(N)
      DOUBLE PRECISION TOL
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),WA(LWA)
      EXTERNAL FCN
C     **********
C
C     SUBROUTINE LMDER1
C
C     THE PURPOSE OF LMDER1 IS TO MINIMIZE THE SUM OF THE SQUARES OF
C     M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF THE
C     LEVENBERG-MARQUARDT ALGORITHM. THIS IS DONE BY USING THE MORE
C     GENERAL LEAST-SQUARES SOLVER LMDER. THE USER MUST PROVIDE A
C     SUBROUTINE WHICH CALCULATES THE FUNCTIONS AND THE JACOBIAN.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE LMDER1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,
C                         IPVT,WA,LWA)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS AND THE JACOBIAN. FCN MUST
C         BE DECLARED IN AN EXTERNAL STATEMENT IN THE USER
C         CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
C         INTEGER M,N,LDFJAC,IFLAG
C         DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)
C         ----------
C         IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC. DO NOT ALTER FJAC.
C         IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND
C         RETURN THIS MATRIX IN FJAC. DO NOT ALTER FVEC.
C         ----------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF LMDER1.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF VARIABLES. N MUST NOT EXCEED M.
C
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
C         CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.
```

```
C
C       FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS
C         THE FUNCTIONS EVALUATED AT THE OUTPUT X.
C
C       FJAC IS AN OUTPUT M BY N ARRAY. THE UPPER N BY N SUBMATRIX
C         OF FJAC CONTAINS AN UPPER TRIANGULAR MATRIX R WITH
C         DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE SUCH THAT
C
C                T     T           T
C               P *(JAC *JAC)*P = R *R,
C
C         WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL
C         CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J)
C         (SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRAPEZOIDAL
C         PART OF FJAC CONTAINS INFORMATION GENERATED DURING
C         THE COMPUTATION OF R.
C
C       LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
C
C       TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS
C         WHEN THE ALGORITHM ESTIMATES EITHER THAT THE RELATIVE
C         ERROR IN THE SUM OF SQUARES IS AT MOST TOL OR THAT
C         THE RELATIVE ERROR BETWEEN X AND THE SOLUTION IS AT
C         MOST TOL.
C
C       INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS
C         TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)
C         VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,
C         INFO IS SET AS FOLLOWS.
C
C         INFO = 0  IMPROPER INPUT PARAMETERS.
C
C         INFO = 1  ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
C                   IN THE SUM OF SQUARES IS AT MOST TOL.
C
C         INFO = 2  ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
C                   BETWEEN X AND THE SOLUTION IS AT MOST TOL.
C
C         INFO = 3  CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.
C
C         INFO = 4  FVEC IS ORTHOGONAL TO THE COLUMNS OF THE
C                   JACOBIAN TO MACHINE PRECISION.
C
C         INFO = 5  NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS
C                   REACHED 100*(N+1).
C
C         INFO = 6  TOL IS TOO SMALL. NO FURTHER REDUCTION IN
C                   THE SUM OF SQUARES IS POSSIBLE.
C
C         INFO = 7  TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN
C                   THE APPROXIMATE SOLUTION X IS POSSIBLE.
C
C       IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT
```

```
C          DEFINES A PERMUTATION MATRIX P SUCH THAT JAC*P = Q*R,
C          WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS
C          ORTHOGONAL (NOT STORED), AND R IS UPPER TRIANGULAR
C          WITH DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE.
C          COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.
C
C        WA IS A WORK ARRAY OF LENGTH LWA.
C
C        LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN 5*N+M.
C
C      SUBPROGRAMS CALLED
C
C        USER-SUPPLIED ...... FCN
C
C        MINPACK-SUPPLIED ... LMDER
C
C      ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C      BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C      **********
       INTEGER MAXFEV,MODE,NFEV,NJEV,NPRINT
       DOUBLE PRECISION FACTOR,FTOL,GTOL,XTOL,ZERO
       DATA FACTOR,ZERO /1.0D2,0.0D0/
       INFO = 0
C
C      CHECK THE INPUT PARAMETERS FOR ERRORS.
C
       IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. M .OR. TOL .LT. ZERO
      *    .OR. LWA .LT. 5*N + M) GO TO 10
C
C      CALL LMDER.
C
       MAXFEV = 100*(N + 1)
       FTOL = TOL
       XTOL = TOL
       GTOL = ZERO
       MODE = 1
       NPRINT = 0
       CALL LMDER(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,MAXFEV,
      *           WA(1),MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,IPVT,WA(N+1),
      *           WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))
       IF (INFO .EQ. 8) INFO = 4
    10 CONTINUE
       RETURN
C
C      LAST CARD OF SUBROUTINE LMDER1.
C
       END
```

```
      SUBROUTINE LMDIF(FCN,M,N,X,FVEC,FTOL,XTOL,GTOL,MAXFEV,EPSFCN,
     *                 DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,
     *                 IPVT,QTF,WA1,WA2,WA3,WA4)
      INTEGER M,N,MAXFEV,MODE,NPRINT,INFO,NFEV,LDFJAC
      INTEGER IPVT(N)
      DOUBLE PRECISION FTOL,XTOL,GTOL,EPSFCN,FACTOR
      DOUBLE PRECISION X(N),FVEC(M),DIAG(N),FJAC(LDFJAC,N),QTF(N),
     *                 WA1(N),WA2(N),WA3(N),WA4(M)
      EXTERNAL FCN
C     **********
C
C     SUBROUTINE LMDIF
C
C     THE PURPOSE OF LMDIF IS TO MINIMIZE THE SUM OF THE SQUARES OF
C     M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF
C     THE LEVENBERG-MARQUARDT ALGORITHM. THE USER MUST PROVIDE A
C     SUBROUTINE WHICH CALCULATES THE FUNCTIONS. THE JACOBIAN IS
C     THEN CALCULATED BY A FORWARD-DIFFERENCE APPROXIMATION.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE LMDIF(FCN,M,N,X,FVEC,FTOL,XTOL,GTOL,MAXFEV,EPSFCN,
C                        DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,
C                        LDFJAC,IPVT,QTF,WA1,WA2,WA3,WA4)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED
C         IN AN EXTERNAL STATEMENT IN THE USER CALLING
C         PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(M,N,X,FVEC,IFLAG)
C         INTEGER M,N,IFLAG
C         DOUBLE PRECISION X(N),FVEC(M)
C         ----------
C         CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC.
C         ----------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF LMDIF.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF VARIABLES. N MUST NOT EXCEED M.
C
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
```

```
C         CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.
C
C       FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS
C         THE FUNCTIONS EVALUATED AT THE OUTPUT X.
C
C       FTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION
C         OCCURS WHEN BOTH THE ACTUAL AND PREDICTED RELATIVE
C         REDUCTIONS IN THE SUM OF SQUARES ARE AT MOST FTOL.
C         THEREFORE, FTOL MEASURES THE RELATIVE ERROR DESIRED
C         IN THE SUM OF SQUARES.
C
C       XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION
C         OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE
C         ITERATES IS AT MOST XTOL. THEREFORE, XTOL MEASURES THE
C         RELATIVE ERROR DESIRED IN THE APPROXIMATE SOLUTION.
C
C       GTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION
C         OCCURS WHEN THE COSINE OF THE ANGLE BETWEEN FVEC AND
C         ANY COLUMN OF THE JACOBIAN IS AT MOST GTOL IN ABSOLUTE
C         VALUE. THEREFORE, GTOL MEASURES THE ORTHOGONALITY
C         DESIRED BETWEEN THE FUNCTION VECTOR AND THE COLUMNS
C         OF THE JACOBIAN.
C
C       MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION
C         OCCURS WHEN THE NUMBER OF CALLS TO FCN IS AT LEAST
C         MAXFEV BY THE END OF AN ITERATION.
C
C       EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE
C         STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS
C         APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE
C         FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS
C         THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE
C         ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE
C         PRECISION.
C
C       DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE
C         BELOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG
C         MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS
C         MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.
C
C       MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE
C         VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,
C         THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER
C         VALUES OF MODE ARE EQUIVALENT TO MODE = 1.
C
C       FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE
C         INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF
C         FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE
C         TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE
C         INTERVAL (.1,100.). 100. IS A GENERALLY RECOMMENDED VALUE.
C
C       NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED
C         PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,
C         FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST
```

```
C         ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND
C         IMMEDIATELY PRIOR TO RETURN, WITH X AND FVEC AVAILABLE
C         FOR PRINTING. IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS
C         OF FCN WITH IFLAG = 0 ARE MADE.
C
C       INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS
C         TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)
C         VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,
C         INFO IS SET AS FOLLOWS.
C
C         INFO = 0  IMPROPER INPUT PARAMETERS.
C
C         INFO = 1  BOTH ACTUAL AND PREDICTED RELATIVE REDUCTIONS
C                   IN THE SUM OF SQUARES ARE AT MOST FTOL.
C
C         INFO = 2  RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES
C                   IS AT MOST XTOL.
C
C         INFO = 3  CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.
C
C         INFO = 4  THE COSINE OF THE ANGLE BETWEEN FVEC AND ANY
C                   COLUMN OF THE JACOBIAN IS AT MOST GTOL IN
C                   ABSOLUTE VALUE.
C
C         INFO = 5  NUMBER OF CALLS TO FCN HAS REACHED OR
C                   EXCEEDED MAXFEV.
C
C         INFO = 6  FTOL IS TOO SMALL. NO FURTHER REDUCTION IN
C                   THE SUM OF SQUARES IS POSSIBLE.
C
C         INFO = 7  XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN
C                   THE APPROXIMATE SOLUTION X IS POSSIBLE.
C
C         INFO = 8  GTOL IS TOO SMALL. FVEC IS ORTHOGONAL TO THE
C                   COLUMNS OF THE JACOBIAN TO MACHINE PRECISION.
C
C       NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF
C         CALLS TO FCN.
C
C       FJAC IS AN OUTPUT M BY N ARRAY. THE UPPER N BY N SUBMATRIX
C         OF FJAC CONTAINS AN UPPER TRIANGULAR MATRIX R WITH
C         DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE SUCH THAT
C
C                T     T           T
C               P *(JAC *JAC)*P = R *R,
C
C         WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL
C         CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J)
C         (SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRAPEZOIDAL
C         PART OF FJAC CONTAINS INFORMATION GENERATED DURING
C         THE COMPUTATION OF R.
C
C       LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
```

```
C
C       IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT
C         DEFINES A PERMUTATION MATRIX P SUCH THAT JAC*P = Q*R,
C         WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS
C         ORTHOGONAL (NOT STORED), AND R IS UPPER TRIANGULAR
C         WITH DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE.
C         COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.
C
C       QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS
C         THE FIRST N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*FVEC.
C
C       WA1, WA2, AND WA3 ARE WORK ARRAYS OF LENGTH N.
C
C       WA4 IS A WORK ARRAY OF LENGTH M.
C
C     SUBPROGRAMS CALLED
C
C       USER-SUPPLIED ...... FCN
C
C       MINPACK-SUPPLIED ... DPMPAR,ENORM,FDJAC2,LMPAR,QRFAC
C
C       FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,DSQRT,MOD
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,IFLAG,ITER,J,L
      DOUBLE PRECISION ACTRED,DELTA,DIRDER,EPSMCH,FNORM,FNORM1,GNORM,
     *                 ONE,PAR,PNORM,PRERED,P1,P5,P25,P75,P0001,RATIO,
     *                 SUM,TEMP,TEMP1,TEMP2,XNORM,ZERO
      DOUBLE PRECISION DPMPAR,ENORM
      DATA ONE,P1,P5,P25,P75,P0001,ZERO
     *     /1.0D0,1.0D-1,5.0D-1,2.5D-1,7.5D-1,1.0D-4,0.0D0/
C
C     EPSMCH IS THE MACHINE PRECISION.
C
      EPSMCH = DPMPAR(1)
C
      INFO = 0
      IFLAG = 0
      NFEV = 0
C
C     CHECK THE INPUT PARAMETERS FOR ERRORS.
C
      IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. M
     *    .OR. FTOL .LT. ZERO .OR. XTOL .LT. ZERO .OR. GTOL .LT. ZERO
     *    .OR. MAXFEV .LE. 0 .OR. FACTOR .LE. ZERO) GO TO 300
      IF (MODE .NE. 2) GO TO 20
      DO 10 J = 1, N
         IF (DIAG(J) .LE. ZERO) GO TO 300
   10    CONTINUE
   20 CONTINUE
C
```

```
C     EVALUATE THE FUNCTION AT THE STARTING POINT
C     AND CALCULATE ITS NORM.
C
      IFLAG = 1
      CALL FCN(M,N,X,FVEC,IFLAG)
      NFEV = 1
      IF (IFLAG .LT. 0) GO TO 300
      FNORM = ENORM(M,FVEC)
C
C     INITIALIZE LEVENBERG-MARQUARDT PARAMETER AND ITERATION COUNTER.
C
      PAR = ZERO
      ITER = 1
C
C     BEGINNING OF THE OUTER LOOP.
C
   30 CONTINUE
C
C        CALCULATE THE JACOBIAN MATRIX.
C
         IFLAG = 2
         CALL FDJAC2(FCN,M,N,X,FVEC,FJAC,LDFJAC,IFLAG,EPSFCN,WA4)
         NFEV = NFEV + N
         IF (IFLAG .LT. 0) GO TO 300
C
C        IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.
C
         IF (NPRINT .LE. 0) GO TO 40
         IFLAG = 0
         IF (MOD(ITER-1,NPRINT) .EQ. 0) CALL FCN(M,N,X,FVEC,IFLAG)
         IF (IFLAG .LT. 0) GO TO 300
   40    CONTINUE
C
C        COMPUTE THE QR FACTORIZATION OF THE JACOBIAN.
C
         CALL QRFAC(M,N,FJAC,LDFJAC,.TRUE.,IPVT,N,WA1,WA2,WA3)
C
C        ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING
C        TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN.
C
         IF (ITER .NE. 1) GO TO 80
         IF (MODE .EQ. 2) GO TO 60
         DO 50 J = 1, N
            DIAG(J) = WA2(J)
            IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE
   50       CONTINUE
   60    CONTINUE
C
C        ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X
C        AND INITIALIZE THE STEP BOUND DELTA.
C
         DO 70 J = 1, N
            WA3(J) = DIAG(J)*X(J)
   70       CONTINUE
```

```
      XNORM = ENORM(N,WA3)
      DELTA = FACTOR*XNORM
      IF (DELTA .EQ. ZERO) DELTA = FACTOR
   80 CONTINUE
C
C     FORM (Q TRANSPOSE)*FVEC AND STORE THE FIRST N COMPONENTS IN
C     QTF.
C
      DO 90 I = 1, M
         WA4(I) = FVEC(I)
   90    CONTINUE
      DO 130 J = 1, N
         IF (FJAC(J,J) .EQ. ZERO) GO TO 120
         SUM = ZERO
         DO 100 I = J, M
            SUM = SUM + FJAC(I,J)*WA4(I)
  100       CONTINUE
         TEMP = -SUM/FJAC(J,J)
         DO 110 I = J, M
            WA4(I) = WA4(I) + FJAC(I,J)*TEMP
  110       CONTINUE
  120    CONTINUE
         FJAC(J,J) = WA1(J)
         QTF(J) = WA4(J)
  130    CONTINUE
C
C     COMPUTE THE NORM OF THE SCALED GRADIENT.
C
      GNORM = ZERO
      IF (FNORM .EQ. ZERO) GO TO 170
      DO 160 J = 1, N
         L = IPVT(J)
         IF (WA2(L) .EQ. ZERO) GO TO 150
         SUM = ZERO
         DO 140 I = 1, J
            SUM = SUM + FJAC(I,J)*(QTF(I)/FNORM)
  140       CONTINUE
         GNORM = DMAX1(GNORM,DABS(SUM/WA2(L)))
  150    CONTINUE
  160    CONTINUE
  170 CONTINUE
C
C     TEST FOR CONVERGENCE OF THE GRADIENT NORM.
C
      IF (GNORM .LE. GTOL) INFO = 4
      IF (INFO .NE. 0) GO TO 300
C
C     RESCALE IF NECESSARY.
C
      IF (MODE .EQ. 2) GO TO 190
      DO 180 J = 1, N
         DIAG(J) = DMAX1(DIAG(J),WA2(J))
  180    CONTINUE
  190 CONTINUE
```

```
C
C     BEGINNING OF THE INNER LOOP.
C
  200 CONTINUE
C
C        DETERMINE THE LEVENBERG-MARQUARDT PARAMETER.
C
         CALL LMPAR(N,FJAC,LDFJAC,IPVT,DIAG,QTF,DELTA,PAR,WA1,WA2,
     *              WA3,WA4)
C
C        STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.
C
         DO 210 J = 1, N
            WA1(J) = -WA1(J)
            WA2(J) = X(J) + WA1(J)
            WA3(J) = DIAG(J)*WA1(J)
  210       CONTINUE
         PNORM = ENORM(N,WA3)
C
C        ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.
C
         IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)
C
C        EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.
C
         IFLAG = 1
         CALL FCN(M,N,WA2,WA4,IFLAG)
         NFEV = NFEV + 1
         IF (IFLAG .LT. 0) GO TO 300
         FNORM1 = ENORM(M,WA4)
C
C        COMPUTE THE SCALED ACTUAL REDUCTION.
C
         ACTRED = -ONE
         IF (P1*FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2
C
C        COMPUTE THE SCALED PREDICTED REDUCTION AND
C        THE SCALED DIRECTIONAL DERIVATIVE.
C
         DO 230 J = 1, N
            WA3(J) = ZERO
            L = IPVT(J)
            TEMP = WA1(L)
            DO 220 I = 1, J
               WA3(I) = WA3(I) + FJAC(I,J)*TEMP
  220          CONTINUE
  230       CONTINUE
         TEMP1 = ENORM(N,WA3)/FNORM
         TEMP2 = (DSQRT(PAR)*PNORM)/FNORM
         PRERED = TEMP1**2 + TEMP2**2/P5
         DIRDER = -(TEMP1**2 + TEMP2**2)
C
C        COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED
C        REDUCTION.
```

```
C
      RATIO = ZERO
      IF (PRERED .NE. ZERO) RATIO = ACTRED/PRERED
C
C        UPDATE THE STEP BOUND.
C
      IF (RATIO .GT. P25) GO TO 240
         IF (ACTRED .GE. ZERO) TEMP = P5
         IF (ACTRED .LT. ZERO)
     *      TEMP = P5*DIRDER/(DIRDER + P5*ACTRED)
         IF (P1*FNORM1 .GE. FNORM .OR. TEMP .LT. P1) TEMP = P1
         DELTA = TEMP*DMIN1(DELTA,PNORM/P1)
         PAR = PAR/TEMP
         GO TO 260
  240 CONTINUE
         IF (PAR .NE. ZERO .AND. RATIO .LT. P75) GO TO 250
         DELTA = PNORM/P5
         PAR = P5*PAR
  250    CONTINUE
  260 CONTINUE
C
C        TEST FOR SUCCESSFUL ITERATION.
C
      IF (RATIO .LT. P0001) GO TO 290
C
C        SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.
C
      DO 270 J = 1, N
         X(J) = WA2(J)
         WA2(J) = DIAG(J)*X(J)
  270    CONTINUE
      DO 280 I = 1, M
         FVEC(I) = WA4(I)
  280    CONTINUE
      XNORM = ENORM(N,WA2)
      FNORM = FNORM1
      ITER = ITER + 1
  290 CONTINUE
C
C        TESTS FOR CONVERGENCE.
C
      IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL
     *    .AND. P5*RATIO .LE. ONE) INFO = 1
      IF (DELTA .LE. XTOL*XNORM) INFO = 2
      IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL
     *    .AND. P5*RATIO .LE. ONE .AND. INFO .EQ. 2) INFO = 3
      IF (INFO .NE. 0) GO TO 300
C
C        TESTS FOR TERMINATION AND STRINGENT TOLERANCES.
C
      IF (NFEV .GE. MAXFEV) INFO = 5
      IF (DABS(ACTRED) .LE. EPSMCH .AND. PRERED .LE. EPSMCH
     *    .AND. P5*RATIO .LE. ONE) INFO = 6
      IF (DELTA .LE. EPSMCH*XNORM) INFO = 7
```

```
            IF (GNORM .LE. EPSMCH) INFO = 8                           LMDF4330
            IF (INFO .NE. 0) GO TO 300                                LMDF4340
C                                                                     LMDF4350
C           END OF THE INNER LOOP. REPEAT IF ITERATION UNSUCCESSFUL.  LMDF4360
C                                                                     LMDF4370
            IF (RATIO .LT. P0001) GO TO 200                           LMDF4380
C                                                                     LMDF4390
C        END OF THE OUTER LOOP.                                       LMDF4400
C                                                                     LMDF4410
         GO TO 30                                                     LMDF4420
  300 CONTINUE                                                        LMDF4430
C                                                                     LMDF4440
C     TERMINATION, EITHER NORMAL OR USER IMPOSED.                     LMDF4450
C                                                                     LMDF4460
      IF (IFLAG .LT. 0) INFO = IFLAG                                  LMDF4470
      IFLAG = 0                                                       LMDF4480
      IF (NPRINT .GT. 0) CALL FCN(M,N,X,FVEC,IFLAG)                   LMDF4490
      RETURN                                                          LMDF4500
C                                                                     LMDF4510
C     LAST CARD OF SUBROUTINE LMDIF.                                  LMDF4520
C                                                                     LMDF4530
      END                                                             LMDF4540
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      SUBROUTINE LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)
      INTEGER M,N,INFO,LWA
      INTEGER IWA(N)
      DOUBLE PRECISION TOL
      DOUBLE PRECISION X(N),FVEC(M),WA(LWA)
      EXTERNAL FCN
C     **********
C
C     SUBROUTINE LMDIF1
C
C     THE PURPOSE OF LMDIF1 IS TO MINIMIZE THE SUM OF THE SQUARES OF
C     M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF THE
C     LEVENBERG-MARQUARDT ALGORITHM. THIS IS DONE BY USING THE MORE
C     GENERAL LEAST-SQUARES SOLVER LMDIF. THE USER MUST PROVIDE A
C     SUBROUTINE WHICH CALCULATES THE FUNCTIONS. THE JACOBIAN IS
C     THEN CALCULATED BY A FORWARD-DIFFERENCE APPROXIMATION.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED
C         IN AN EXTERNAL STATEMENT IN THE USER CALLING
C         PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(M,N,X,FVEC,IFLAG)
C         INTEGER M,N,IFLAG
C         DOUBLE PRECISION X(N),FVEC(M)
C         ----------
C         CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC.
C         ----------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF LMDIF1.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF VARIABLES. N MUST NOT EXCEED M.
C
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
C         CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.
C
C       FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS
C         THE FUNCTIONS EVALUATED AT THE OUTPUT X.
```

```
C
C       TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS
C         WHEN THE ALGORITHM ESTIMATES EITHER THAT THE RELATIVE
C         ERROR IN THE SUM OF SQUARES IS AT MOST TOL OR THAT
C         THE RELATIVE ERROR BETWEEN X AND THE SOLUTION IS AT
C         MOST TOL.
C
C       INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS
C         TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)
C         VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,
C         INFO IS SET AS FOLLOWS.
C
C         INFO = 0  IMPROPER INPUT PARAMETERS.
C
C         INFO = 1  ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
C                   IN THE SUM OF SQUARES IS AT MOST TOL.
C
C         INFO = 2  ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
C                   BETWEEN X AND THE SOLUTION IS AT MOST TOL.
C
C         INFO = 3  CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.
C
C         INFO = 4  FVEC IS ORTHOGONAL TO THE COLUMNS OF THE
C                   JACOBIAN TO MACHINE PRECISION.
C
C         INFO = 5  NUMBER OF CALLS TO FCN HAS REACHED OR
C                   EXCEEDED 200*(N+1).
C
C         INFO = 6  TOL IS TOO SMALL. NO FURTHER REDUCTION IN
C                   THE SUM OF SQUARES IS POSSIBLE.
C
C         INFO = 7  TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN
C                   THE APPROXIMATE SOLUTION X IS POSSIBLE.
C
C       IWA IS AN INTEGER WORK ARRAY OF LENGTH N.
C
C       WA IS A WORK ARRAY OF LENGTH LWA.
C
C       LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN
C         M*N+5*N+M.
C
C     SUBPROGRAMS CALLED
C
C       USER-SUPPLIED ...... FCN
C
C       MINPACK-SUPPLIED ... LMDIF
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER MAXFEV,MODE,MP5N,NFEV,NPRINT
      DOUBLE PRECISION EPSFCN,FACTOR,FTOL,GTOL,XTOL,ZERO
      DATA FACTOR,ZERO /1.0D2,0.0D0/
```

```
      INFO = 0
C
C     CHECK THE INPUT PARAMETERS FOR ERRORS.
C
      IF (N .LE. 0 .OR. M .LT. N .OR. TOL .LT. ZERO
     *    .OR. LWA .LT. M*N + 5*N + M) GO TO 10
C
C     CALL LMDIF.
C
      MAXFEV = 200*(N + 1)
      FTOL = TOL
      XTOL = TOL
      GTOL = ZERO
      EPSFCN = ZERO
      MODE = 1
      NPRINT = 0
      MP5N = M + 5*N
      CALL LMDIF(FCN,M,N,X,FVEC,FTOL,XTOL,GTOL,MAXFEV,EPSFCN,WA(1),
     *           MODE,FACTOR,NPRINT,INFO,NFEV,WA(MP5N+1),M,IWA,
     *           WA(N+1),WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))
      IF (INFO .EQ. 8) INFO = 4
   10 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE LMDIF1.
C
      END
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      SUBROUTINE LMPAR(N,R,LDR,IPVT,DIAG,QTB,DELTA,PAR,X,SDIAG,WA1,
     *                 WA2)
      INTEGER N,LDR
      INTEGER IPVT(N)
      DOUBLE PRECISION DELTA,PAR
      DOUBLE PRECISION R(LDR,N),DIAG(N),QTB(N),X(N),SDIAG(N),WA1(N),
     *                 WA2(N)
C     **********
C
C     SUBROUTINE LMPAR
C
C     GIVEN AN M BY N MATRIX A, AN N BY N NONSINGULAR DIAGONAL
C     MATRIX D, AN M-VECTOR B, AND A POSITIVE NUMBER DELTA,
C     THE PROBLEM IS TO DETERMINE A VALUE FOR THE PARAMETER
C     PAR SUCH THAT IF X SOLVES THE SYSTEM
C
C           A*X = B ,     SQRT(PAR)*D*X = 0 ,
C
C     IN THE LEAST SQUARES SENSE, AND DXNORM IS THE EUCLIDEAN
C     NORM OF D*X, THEN EITHER PAR IS ZERO AND
C
C           (DXNORM-DELTA) .LE. 0.1*DELTA ,
C
C     OR PAR IS POSITIVE AND
C
C           ABS(DXNORM-DELTA) .LE. 0.1*DELTA .
C
C     THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM
C     IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE
C     QR FACTORIZATION, WITH COLUMN PIVOTING, OF A. THAT IS, IF
C     A*P = Q*R, WHERE P IS A PERMUTATION MATRIX, Q HAS ORTHOGONAL
C     COLUMNS, AND R IS AN UPPER TRIANGULAR MATRIX WITH DIAGONAL
C     ELEMENTS OF NONINCREASING MAGNITUDE, THEN LMPAR EXPECTS
C     THE FULL UPPER TRIANGLE OF R, THE PERMUTATION MATRIX P,
C     AND THE FIRST N COMPONENTS OF (Q TRANSPOSE)*B. ON OUTPUT
C     LMPAR ALSO PROVIDES AN UPPER TRIANGULAR MATRIX S SUCH THAT
C
C            T   T                   T
C           P *(A *A + PAR*D*D)*P = S *S .
C
C     S IS EMPLOYED WITHIN LMPAR AND MAY BE OF SEPARATE INTEREST.
C
C     ONLY A FEW ITERATIONS ARE GENERALLY NEEDED FOR CONVERGENCE
C     OF THE ALGORITHM. IF, HOWEVER, THE LIMIT OF 10 ITERATIONS
C     IS REACHED, THEN THE OUTPUT PAR WILL CONTAIN THE BEST
C     VALUE OBTAINED SO FAR.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE LMPAR(N,R,LDR,IPVT,DIAG,QTB,DELTA,PAR,X,SDIAG,
C                        WA1,WA2)
C
C     WHERE
C
```

```
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R.
C
C       R IS AN N BY N ARRAY. ON INPUT THE FULL UPPER TRIANGLE
C         MUST CONTAIN THE FULL UPPER TRIANGLE OF THE MATRIX R.
C         ON OUTPUT THE FULL UPPER TRIANGLE IS UNALTERED, AND THE
C         STRICT LOWER TRIANGLE CONTAINS THE STRICT UPPER TRIANGLE
C         (TRANSPOSED) OF THE UPPER TRIANGULAR MATRIX S.
C
C       LDR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY R.
C
C       IPVT IS AN INTEGER INPUT ARRAY OF LENGTH N WHICH DEFINES THE
C         PERMUTATION MATRIX P SUCH THAT A*P = Q*R. COLUMN J OF P
C         IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.
C
C       DIAG IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE
C         DIAGONAL ELEMENTS OF THE MATRIX D.
C
C       QTB IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE FIRST
C         N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*B.
C
C       DELTA IS A POSITIVE INPUT VARIABLE WHICH SPECIFIES AN UPPER
C         BOUND ON THE EUCLIDEAN NORM OF D*X.
C
C       PAR IS A NONNEGATIVE VARIABLE. ON INPUT PAR CONTAINS AN
C         INITIAL ESTIMATE OF THE LEVENBERG-MARQUARDT PARAMETER.
C         ON OUTPUT PAR CONTAINS THE FINAL ESTIMATE.
C
C       X IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE LEAST
C         SQUARES SOLUTION OF THE SYSTEM A*X = B, SQRT(PAR)*D*X = 0,
C         FOR THE OUTPUT PAR.
C
C       SDIAG IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE
C         DIAGONAL ELEMENTS OF THE UPPER TRIANGULAR MATRIX S.
C
C       WA1 AND WA2 ARE WORK ARRAYS OF LENGTH N.
C
C     SUBPROGRAMS CALLED
C
C       MINPACK-SUPPLIED ... DPMPAR,ENORM,QRSOLV
C
C       FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,DSQRT
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,ITER,J,JM1,JP1,K,L,NSING
      DOUBLE PRECISION DXNORM,DWARF,FP,GNORM,PARC,PARL,PARU,P1,P001,
     *                 SUM,TEMP,ZERO
      DOUBLE PRECISION DPMPAR,ENORM
      DATA P1,P001,ZERO /1.0D-1,1.0D-3,0.0D0/
C
C     DWARF IS THE SMALLEST POSITIVE MAGNITUDE.
```

```
C
      DWARF = DPMPAR(2)
C
C     COMPUTE AND STORE IN X THE GAUSS-NEWTON DIRECTION. IF THE
C     JACOBIAN IS RANK-DEFICIENT, OBTAIN A LEAST SQUARES SOLUTION.
C
      NSING = N
      DO 10 J = 1, N
         WA1(J) = QTB(J)
         IF (R(J,J) .EQ. ZERO .AND. NSING .EQ. N) NSING = J - 1
         IF (NSING .LT. N) WA1(J) = ZERO
   10    CONTINUE
      IF (NSING .LT. 1) GO TO 50
      DO 40 K = 1, NSING
         J = NSING - K + 1
         WA1(J) = WA1(J)/R(J,J)
         TEMP = WA1(J)
         JM1 = J - 1
         IF (JM1 .LT. 1) GO TO 30
         DO 20 I = 1, JM1
            WA1(I) = WA1(I) - R(I,J)*TEMP
   20       CONTINUE
   30    CONTINUE
   40    CONTINUE
   50 CONTINUE
      DO 60 J = 1, N
         L = IPVT(J)
         X(L) = WA1(J)
   60    CONTINUE
C
C     INITIALIZE THE ITERATION COUNTER.
C     EVALUATE THE FUNCTION AT THE ORIGIN, AND TEST
C     FOR ACCEPTANCE OF THE GAUSS-NEWTON DIRECTION.
C
      ITER = 0
      DO 70 J = 1, N
         WA2(J) = DIAG(J)*X(J)
   70    CONTINUE
      DXNORM = ENORM(N,WA2)
      FP = DXNORM - DELTA
      IF (FP .LE. P1*DELTA) GO TO 220
C
C     IF THE JACOBIAN IS NOT RANK DEFICIENT, THE NEWTON
C     STEP PROVIDES A LOWER BOUND, PARL, FOR THE ZERO OF
C     THE FUNCTION. OTHERWISE SET THIS BOUND TO ZERO.
C
      PARL = ZERO
      IF (NSING .LT. N) GO TO 120
      DO 80 J = 1, N
         L = IPVT(J)
         WA1(J) = DIAG(L)*(WA2(L)/DXNORM)
   80    CONTINUE
      DO 110 J = 1, N
         SUM = ZERO
```

```
         JM1 = J - 1
         IF (JM1 .LT. 1) GO TO 100
         DO 90 I = 1, JM1
            SUM = SUM + R(I,J)*WA1(I)
   90       CONTINUE
  100    CONTINUE
         WA1(J) = (WA1(J) - SUM)/R(J,J)
  110    CONTINUE
      TEMP = ENORM(N,WA1)
      PARL = ((FP/DELTA)/TEMP)/TEMP
  120 CONTINUE
C
C     CALCULATE AN UPPER BOUND, PARU, FOR THE ZERO OF THE FUNCTION.
C
      DO 140 J = 1, N
         SUM = ZERO
         DO 130 I = 1, J
            SUM = SUM + R(I,J)*QTB(I)
  130       CONTINUE
         L = IPVT(J)
         WA1(J) = SUM/DIAG(L)
  140    CONTINUE
      GNORM = ENORM(N,WA1)
      PARU = GNORM/DELTA
      IF (PARU .EQ. ZERO) PARU = DWARF/DMIN1(DELTA,P1)
C
C     IF THE INPUT PAR LIES OUTSIDE OF THE INTERVAL (PARL,PARU),
C     SET PAR TO THE CLOSER ENDPOINT.
C
      PAR = DMAX1(PAR,PARL)
      PAR = DMIN1(PAR,PARU)
      IF (PAR .EQ. ZERO) PAR = GNORM/DXNORM
C
C     BEGINNING OF AN ITERATION.
C
  150 CONTINUE
         ITER = ITER + 1
C
C        EVALUATE THE FUNCTION AT THE CURRENT VALUE OF PAR.
C
         IF (PAR .EQ. ZERO) PAR = DMAX1(DWARF,P001*PARU)
         TEMP = DSQRT(PAR)
         DO 160 J = 1, N
            WA1(J) = TEMP*DIAG(J)
  160       CONTINUE
         CALL QRSOLV(N,R,LDR,IPVT,WA1,QTB,X,SDIAG,WA2)
         DO 170 J = 1, N
            WA2(J) = DIAG(J)*X(J)
  170       CONTINUE
         DXNORM = ENORM(N,WA2)
         TEMP = FP
         FP = DXNORM - DELTA
C
C        IF THE FUNCTION IS SMALL ENOUGH, ACCEPT THE CURRENT VALUE
```

```
C        OF PAR. ALSO TEST FOR THE EXCEPTIONAL CASES WHERE PARL
C        IS ZERO OR THE NUMBER OF ITERATIONS HAS REACHED 10.
C
         IF (DABS(FP) .LE. P1*DELTA
     *       .OR. PARL .EQ. ZERO .AND. FP .LE. TEMP
     *            .AND. TEMP .LT. ZERO .OR. ITER .EQ. 10) GO TO 220
C
C        COMPUTE THE NEWTON CORRECTION.
C
         DO 180 J = 1, N
            L = IPVT(J)
            WA1(J) = DIAG(L)*(WA2(L)/DXNORM)
  180       CONTINUE
         DO 210 J = 1, N
            WA1(J) = WA1(J)/SDIAG(J)
            TEMP = WA1(J)
            JP1 = J + 1
            IF (N .LT. JP1) GO TO 200
            DO 190 I = JP1, N
               WA1(I) = WA1(I) - R(I,J)*TEMP
  190          CONTINUE
  200       CONTINUE
  210       CONTINUE
         TEMP = ENORM(N,WA1)
         PARC = ((FP/DELTA)/TEMP)/TEMP
C
C        DEPENDING ON THE SIGN OF THE FUNCTION, UPDATE PARL OR PARU.
C
         IF (FP .GT. ZERO) PARL = DMAX1(PARL,PAR)
         IF (FP .LT. ZERO) PARU = DMIN1(PARU,PAR)
C
C        COMPUTE AN IMPROVED ESTIMATE FOR PAR.
C
         PAR = DMAX1(PARL,PAR+PARC)
C
C        END OF AN ITERATION.
C
         GO TO 150
  220 CONTINUE
C
C     TERMINATION.
C
      IF (ITER .EQ. 0) PAR = ZERO
      RETURN
C
C     LAST CARD OF SUBROUTINE LMPAR.
C
      END
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      SUBROUTINE LMSTR(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
     *                 MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,
     *                 IPVT,QTF,WA1,WA2,WA3,WA4)
      INTEGER M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV
      INTEGER IPVT(N)
      LOGICAL SING
      DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),DIAG(N),QTF(N),
     *                 WA1(N),WA2(N),WA3(N),WA4(M)
C     **********
C
C     SUBROUTINE LMSTR
C
C     THE PURPOSE OF LMSTR IS TO MINIMIZE THE SUM OF THE SQUARES OF
C     M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF
C     THE LEVENBERG-MARQUARDT ALGORITHM WHICH USES MINIMAL STORAGE.
C     THE USER MUST PROVIDE A SUBROUTINE WHICH CALCULATES THE
C     FUNCTIONS AND THE ROWS OF THE JACOBIAN.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE LMSTR(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
C                        MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,
C                        NJEV,IPVT,QTF,WA1,WA2,WA3,WA4)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS AND THE ROWS OF THE JACOBIAN.
C         FCN MUST BE DECLARED IN AN EXTERNAL STATEMENT IN THE
C         USER CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)
C         INTEGER M,N,IFLAG
C         DOUBLE PRECISION X(N),FVEC(M),FJROW(N)
C         ----------
C         IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC.
C         IF IFLAG = I CALCULATE THE (I-1)-ST ROW OF THE
C         JACOBIAN AT X AND RETURN THIS VECTOR IN FJROW.
C         ----------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF LMSTR.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF VARIABLES. N MUST NOT EXCEED M.
C
```

```
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
C         CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.
C
C       FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS
C         THE FUNCTIONS EVALUATED AT THE OUTPUT X.
C
C       FJAC IS AN OUTPUT N BY N ARRAY. THE UPPER TRIANGLE OF FJAC
C         CONTAINS AN UPPER TRIANGULAR MATRIX R SUCH THAT
C
C                T     T           T
C               P *(JAC *JAC)*P = R *R,
C
C         WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL
C         CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J)
C         (SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRIANGULAR
C         PART OF FJAC CONTAINS INFORMATION GENERATED DURING
C         THE COMPUTATION OF R.
C
C       LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
C
C       FTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION
C         OCCURS WHEN BOTH THE ACTUAL AND PREDICTED RELATIVE
C         REDUCTIONS IN THE SUM OF SQUARES ARE AT MOST FTOL.
C         THEREFORE, FTOL MEASURES THE RELATIVE ERROR DESIRED
C         IN THE SUM OF SQUARES.
C
C       XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION
C         OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE
C         ITERATES IS AT MOST XTOL. THEREFORE, XTOL MEASURES THE
C         RELATIVE ERROR DESIRED IN THE APPROXIMATE SOLUTION.
C
C       GTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION
C         OCCURS WHEN THE COSINE OF THE ANGLE BETWEEN FVEC AND
C         ANY COLUMN OF THE JACOBIAN IS AT MOST GTOL IN ABSOLUTE
C         VALUE. THEREFORE, GTOL MEASURES THE ORTHOGONALITY
C         DESIRED BETWEEN THE FUNCTION VECTOR AND THE COLUMNS
C         OF THE JACOBIAN.
C
C       MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION
C         OCCURS WHEN THE NUMBER OF CALLS TO FCN WITH IFLAG = 1
C         HAS REACHED MAXFEV.
C
C       DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE
C         BELOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG
C         MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS
C         MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.
C
C       MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE
C         VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,
C         THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER
C         VALUES OF MODE ARE EQUIVALENT TO MODE = 1.
C
```

```
C       FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE
C         INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF
C         FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE
C         TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE
C         INTERVAL (.1,100.). 100. IS A GENERALLY RECOMMENDED VALUE.
C
C       NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED
C         PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,
C         FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST
C         ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND
C         IMMEDIATELY PRIOR TO RETURN, WITH X AND FVEC AVAILABLE
C         FOR PRINTING. IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS
C         OF FCN WITH IFLAG = 0 ARE MADE.
C
C       INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS
C         TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)
C         VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,
C         INFO IS SET AS FOLLOWS.
C
C         INFO = 0  IMPROPER INPUT PARAMETERS.
C
C         INFO = 1  BOTH ACTUAL AND PREDICTED RELATIVE REDUCTIONS
C                   IN THE SUM OF SQUARES ARE AT MOST FTOL.
C
C         INFO = 2  RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES
C                   IS AT MOST XTOL.
C
C         INFO = 3  CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.
C
C         INFO = 4  THE COSINE OF THE ANGLE BETWEEN FVEC AND ANY
C                   COLUMN OF THE JACOBIAN IS AT MOST GTOL IN
C                   ABSOLUTE VALUE.
C
C         INFO = 5  NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS
C                   REACHED MAXFEV.
C
C         INFO = 6  FTOL IS TOO SMALL. NO FURTHER REDUCTION IN
C                   THE SUM OF SQUARES IS POSSIBLE.
C
C         INFO = 7  XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN
C                   THE APPROXIMATE SOLUTION X IS POSSIBLE.
C
C         INFO = 8  GTOL IS TOO SMALL. FVEC IS ORTHOGONAL TO THE
C                   COLUMNS OF THE JACOBIAN TO MACHINE PRECISION.
C
C       NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF
C         CALLS TO FCN WITH IFLAG = 1.
C
C       NJEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF
C         CALLS TO FCN WITH IFLAG = 2.
C
C       IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT
C         DEFINES A PERMUTATION MATRIX P SUCH THAT JAC*P = Q*R,
C         WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS
```

```
C          ORTHOGONAL (NOT STORED), AND R IS UPPER TRIANGULAR.
C          COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.
C
C        QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS
C          THE FIRST N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*FVEC.
C
C        WA1, WA2, AND WA3 ARE WORK ARRAYS OF LENGTH N.
C
C        WA4 IS A WORK ARRAY OF LENGTH M.
C
C      SUBPROGRAMS CALLED
C
C        USER-SUPPLIED ...... FCN
C
C        MINPACK-SUPPLIED ... DPMPAR,ENORM,LMPAR,QRFAC,RWUPDT
C
C        FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,DSQRT,MOD
C
C      ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C      BURTON S. GARBOW, DUDLEY V. GOETSCHEL, KENNETH E. HILLSTROM,
C      JORGE J. MORE
C
C      **********
       INTEGER I,IFLAG,ITER,J,L
       DOUBLE PRECISION ACTRED,DELTA,DIRDER,EPSMCH,FNORM,FNORM1,GNORM,
      *                 ONE,PAR,PNORM,PRERED,P1,P5,P25,P75,P0001,RATIO,
      *                 SUM,TEMP,TEMP1,TEMP2,XNORM,ZERO
       DOUBLE PRECISION DPMPAR,ENORM
       DATA ONE,P1,P5,P25,P75,P0001,ZERO
      *     /1.0D0,1.0D-1,5.0D-1,2.5D-1,7.5D-1,1.0D-4,0.0D0/
C
C      EPSMCH IS THE MACHINE PRECISION.
C
       EPSMCH = DPMPAR(1)
C
       INFO = 0
       IFLAG = 0
       NFEV = 0
       NJEV = 0
C
C      CHECK THE INPUT PARAMETERS FOR ERRORS.
C
       IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. N
      *    .OR. FTOL .LT. ZERO .OR. XTOL .LT. ZERO .OR. GTOL .LT. ZERO
      *    .OR. MAXFEV .LE. 0 .OR. FACTOR .LE. ZERO) GO TO 340
       IF (MODE .NE. 2) GO TO 20
       DO 10 J = 1, N
          IF (DIAG(J) .LE. ZERO) GO TO 340
    10    CONTINUE
    20 CONTINUE
C
C      EVALUATE THE FUNCTION AT THE STARTING POINT
C      AND CALCULATE ITS NORM.
C
```

```
      IFLAG = 1
      CALL FCN(M,N,X,FVEC,WA3,IFLAG)
      NFEV = 1
      IF (IFLAG .LT. 0) GO TO 340
      FNORM = ENORM(M,FVEC)
C
C     INITIALIZE LEVENBERG-MARQUARDT PARAMETER AND ITERATION COUNTER.
C
      PAR = ZERO
      ITER = 1
C
C     BEGINNING OF THE OUTER LOOP.
C
   30 CONTINUE
C
C        IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.
C
         IF (NPRINT .LE. 0) GO TO 40
         IFLAG = 0
         IF (MOD(ITER-1,NPRINT) .EQ. 0) CALL FCN(M,N,X,FVEC,WA3,IFLAG)
         IF (IFLAG .LT. 0) GO TO 340
   40    CONTINUE
C
C        COMPUTE THE QR FACTORIZATION OF THE JACOBIAN MATRIX
C        CALCULATED ONE ROW AT A TIME, WHILE SIMULTANEOUSLY
C        FORMING (Q TRANSPOSE)*FVEC AND STORING THE FIRST
C        N COMPONENTS IN QTF.
C
         DO 60 J = 1, N
            QTF(J) = ZERO
            DO 50 I = 1, N
               FJAC(I,J) = ZERO
   50          CONTINUE
   60       CONTINUE
         IFLAG = 2
         DO 70 I = 1, M
            CALL FCN(M,N,X,FVEC,WA3,IFLAG)
            IF (IFLAG .LT. 0) GO TO 340
            TEMP = FVEC(I)
            CALL RWUPDT(N,FJAC,LDFJAC,WA3,QTF,TEMP,WA1,WA2)
            IFLAG = IFLAG + 1
   70       CONTINUE
         NJEV = NJEV + 1
C
C        IF THE JACOBIAN IS RANK DEFICIENT, CALL QRFAC TO
C        REORDER ITS COLUMNS AND UPDATE THE COMPONENTS OF QTF.
C
         SING = .FALSE.
         DO 80 J = 1, N
            IF (FJAC(J,J) .EQ. ZERO) SING = .TRUE.
            IPVT(J) = J
            WA2(J) = ENORM(J,FJAC(1,J))
   80       CONTINUE
         IF (.NOT.SING) GO TO 130
```

```
      CALL QRFAC(N,N,FJAC,LDFJAC,.TRUE.,IPVT,N,WA1,WA2,WA3)
      DO 120 J = 1, N
         IF (FJAC(J,J) .EQ. ZERO) GO TO 110
         SUM = ZERO
         DO 90 I = J, N
            SUM = SUM + FJAC(I,J)*QTF(I)
   90       CONTINUE
         TEMP = -SUM/FJAC(J,J)
         DO 100 I = J, N
            QTF(I) = QTF(I) + FJAC(I,J)*TEMP
  100       CONTINUE
  110    CONTINUE
         FJAC(J,J) = WA1(J)
  120    CONTINUE
  130 CONTINUE
C
C        ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING
C        TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN.
C
      IF (ITER .NE. 1) GO TO 170
      IF (MODE .EQ. 2) GO TO 150
      DO 140 J = 1, N
         DIAG(J) = WA2(J)
         IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE
  140    CONTINUE
  150 CONTINUE
C
C        ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X
C        AND INITIALIZE THE STEP BOUND DELTA.
C
      DO 160 J = 1, N
         WA3(J) = DIAG(J)*X(J)
  160    CONTINUE
      XNORM = ENORM(N,WA3)
      DELTA = FACTOR*XNORM
      IF (DELTA .EQ. ZERO) DELTA = FACTOR
  170 CONTINUE
C
C        COMPUTE THE NORM OF THE SCALED GRADIENT.
C
      GNORM = ZERO
      IF (FNORM .EQ. ZERO) GO TO 210
      DO 200 J = 1, N
         L = IPVT(J)
         IF (WA2(L) .EQ. ZERO) GO TO 190
         SUM = ZERO
         DO 180 I = 1, J
            SUM = SUM + FJAC(I,J)*(QTF(I)/FNORM)
  180       CONTINUE
         GNORM = DMAX1(GNORM,DABS(SUM/WA2(L)))
  190    CONTINUE
  200    CONTINUE
  210 CONTINUE
C
```

```
C        TEST FOR CONVERGENCE OF THE GRADIENT NORM.
C
         IF (GNORM .LE. GTOL) INFO = 4
         IF (INFO .NE. 0) GO TO 340
C
C        RESCALE IF NECESSARY.
C
         IF (MODE .EQ. 2) GO TO 230
         DO 220 J = 1, N
            DIAG(J) = DMAX1(DIAG(J),WA2(J))
  220       CONTINUE
  230    CONTINUE
C
C        BEGINNING OF THE INNER LOOP.
C
  240    CONTINUE
C
C           DETERMINE THE LEVENBERG-MARQUARDT PARAMETER.
C
            CALL LMPAR(N,FJAC,LDFJAC,IPVT,DIAG,QTF,DELTA,PAR,WA1,WA2,
     *                 WA3,WA4)
C
C           STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.
C
            DO 250 J = 1, N
               WA1(J) = -WA1(J)
               WA2(J) = X(J) + WA1(J)
               WA3(J) = DIAG(J)*WA1(J)
  250          CONTINUE
            PNORM = ENORM(N,WA3)
C
C           ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.
C
            IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)
C
C           EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.
C
            IFLAG = 1
            CALL FCN(M,N,WA2,WA4,WA3,IFLAG)
            NFEV = NFEV + 1
            IF (IFLAG .LT. 0) GO TO 340
            FNORM1 = ENORM(M,WA4)
C
C           COMPUTE THE SCALED ACTUAL REDUCTION.
C
            ACTRED = -ONE
            IF (P1*FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2
C
C           COMPUTE THE SCALED PREDICTED REDUCTION AND
C           THE SCALED DIRECTIONAL DERIVATIVE.
C
            DO 270 J = 1, N
               WA3(J) = ZERO
               L = IPVT(J)
```

```
               TEMP = WA1(L)
               DO 260 I = 1, J
                  WA3(I) = WA3(I) + FJAC(I,J)*TEMP
  260             CONTINUE
  270          CONTINUE
            TEMP1 = ENORM(N,WA3)/FNORM
            TEMP2 = (DSQRT(PAR)*PNORM)/FNORM
            PRERED = TEMP1**2 + TEMP2**2/P5
            DIRDER = -(TEMP1**2 + TEMP2**2)
C
C           COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED
C           REDUCTION.
C
            RATIO = ZERO
            IF (PRERED .NE. ZERO) RATIO = ACTRED/PRERED
C
C           UPDATE THE STEP BOUND.
C
            IF (RATIO .GT. P25) GO TO 280
               IF (ACTRED .GE. ZERO) TEMP = P5
               IF (ACTRED .LT. ZERO)
     *            TEMP = P5*DIRDER/(DIRDER + P5*ACTRED)
               IF (P1*FNORM1 .GE. FNORM .OR. TEMP .LT. P1) TEMP = P1
               DELTA = TEMP*DMIN1(DELTA,PNORM/P1)
               PAR = PAR/TEMP
               GO TO 300
  280       CONTINUE
               IF (PAR .NE. ZERO .AND. RATIO .LT. P75) GO TO 290
               DELTA = PNORM/P5
               PAR = P5*PAR
  290          CONTINUE
  300       CONTINUE
C
C           TEST FOR SUCCESSFUL ITERATION.
C
            IF (RATIO .LT. P0001) GO TO 330
C
C           SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.
C
            DO 310 J = 1, N
               X(J) = WA2(J)
               WA2(J) = DIAG(J)*X(J)
  310          CONTINUE
            DO 320 I = 1, M
               FVEC(I) = WA4(I)
  320          CONTINUE
            XNORM = ENORM(N,WA2)
            FNORM = FNORM1
            ITER = ITER + 1
  330       CONTINUE
C
C           TESTS FOR CONVERGENCE.
C
            IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL
```

```
     *        .AND. P5*RATIO .LE. ONE) INFO = 1
            IF (DELTA .LE. XTOL*XNORM) INFO = 2
            IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL
     *        .AND. P5*RATIO .LE. ONE .AND. INFO .EQ. 2) INFO = 3
            IF (INFO .NE. 0) GO TO 340
C
C           TESTS FOR TERMINATION AND STRINGENT TOLERANCES.
C
            IF (NFEV .GE. MAXFEV) INFO = 5
            IF (DABS(ACTRED) .LE. EPSMCH .AND. PRERED .LE. EPSMCH
     *        .AND. P5*RATIO .LE. ONE) INFO = 6
            IF (DELTA .LE. EPSMCH*XNORM) INFO = 7
            IF (GNORM .LE. EPSMCH) INFO = 8
            IF (INFO .NE. 0) GO TO 340
C
C           END OF THE INNER LOOP. REPEAT IF ITERATION UNSUCCESSFUL.
C
            IF (RATIO .LT. P0001) GO TO 240
C
C        END OF THE OUTER LOOP.
C
         GO TO 30
  340 CONTINUE
C
C     TERMINATION, EITHER NORMAL OR USER IMPOSED.
C
      IF (IFLAG .LT. 0) INFO = IFLAG
      IFLAG = 0
      IF (NPRINT .GT. 0) CALL FCN(M,N,X,FVEC,WA3,IFLAG)
      RETURN
C
C     LAST CARD OF SUBROUTINE LMSTR.
C
      END
```

THIS PAGE WAS INTENTIONALLY LEFT BLANK

```
      SUBROUTINE LMSTR1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,IPVT,WA,
     *                  LWA)
      INTEGER M,N,LDFJAC,INFO,LWA
      INTEGER IPVT(N)
      DOUBLE PRECISION TOL
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),WA(LWA)
      EXTERNAL FCN
C     **********
C
C     SUBROUTINE LMSTR1
C
C     THE PURPOSE OF LMSTR1 IS TO MINIMIZE THE SUM OF THE SQUARES OF
C     M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF
C     THE LEVENBERG-MARQUARDT ALGORITHM WHICH USES MINIMAL STORAGE.
C     THIS IS DONE BY USING THE MORE GENERAL LEAST-SQUARES SOLVER
C     LMSTR. THE USER MUST PROVIDE A SUBROUTINE WHICH CALCULATES
C     THE FUNCTIONS AND THE ROWS OF THE JACOBIAN.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE LMSTR1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,
C                         IPVT,WA,LWA)
C
C     WHERE
C
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH
C         CALCULATES THE FUNCTIONS AND THE ROWS OF THE JACOBIAN.
C         FCN MUST BE DECLARED IN AN EXTERNAL STATEMENT IN THE
C         USER CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.
C
C         SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)
C         INTEGER M,N,IFLAG
C         DOUBLE PRECISION X(N),FVEC(M),FJROW(N)
C         ----------
C         IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND
C         RETURN THIS VECTOR IN FVEC.
C         IF IFLAG = I CALCULATE THE (I-1)-ST ROW OF THE
C         JACOBIAN AT X AND RETURN THIS VECTOR IN FJROW.
C         ----------
C         RETURN
C         END
C
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS
C         THE USER WANTS TO TERMINATE EXECUTION OF LMSTR1.
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF FUNCTIONS.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF VARIABLES. N MUST NOT EXCEED M.
C
C       X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X
```

```
C        CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.
C
C      FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS
C        THE FUNCTIONS EVALUATED AT THE OUTPUT X.
C
C      FJAC IS AN OUTPUT N BY N ARRAY. THE UPPER TRIANGLE OF FJAC
C        CONTAINS AN UPPER TRIANGULAR MATRIX R SUCH THAT
C
C               T     T           T
C              P *(JAC *JAC)*P = R *R,
C
C        WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL
C        CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J)
C        (SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRIANGULAR
C        PART OF FJAC CONTAINS INFORMATION GENERATED DURING
C        THE COMPUTATION OF R.
C
C      LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C        WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
C
C      TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS
C        WHEN THE ALGORITHM ESTIMATES EITHER THAT THE RELATIVE
C        ERROR IN THE SUM OF SQUARES IS AT MOST TOL OR THAT
C        THE RELATIVE ERROR BETWEEN X AND THE SOLUTION IS AT
C        MOST TOL.
C
C      INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS
C        TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)
C        VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,
C        INFO IS SET AS FOLLOWS.
C
C        INFO = 0  IMPROPER INPUT PARAMETERS.
C
C        INFO = 1  ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
C                  IN THE SUM OF SQUARES IS AT MOST TOL.
C
C        INFO = 2  ALGORITHM ESTIMATES THAT THE RELATIVE ERROR
C                  BETWEEN X AND THE SOLUTION IS AT MOST TOL.
C
C        INFO = 3  CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.
C
C        INFO = 4  FVEC IS ORTHOGONAL TO THE COLUMNS OF THE
C                  JACOBIAN TO MACHINE PRECISION.
C
C        INFO = 5  NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS
C                  REACHED 100*(N+1).
C
C        INFO = 6  TOL IS TOO SMALL. NO FURTHER REDUCTION IN
C                  THE SUM OF SQUARES IS POSSIBLE.
C
C        INFO = 7  TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN
C                  THE APPROXIMATE SOLUTION X IS POSSIBLE.
C
C      IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT
```

```
C         DEFINES A PERMUTATION MATRIX P SUCH THAT JAC*P = Q*R,
C         WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS
C         ORTHOGONAL (NOT STORED), AND R IS UPPER TRIANGULAR.
C         COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.
C
C       WA IS A WORK ARRAY OF LENGTH LWA.
C
C       LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN 5*N+M.
C
C     SUBPROGRAMS CALLED
C
C       USER-SUPPLIED ...... FCN
C
C       MINPACK-SUPPLIED ... LMSTR
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, DUDLEY V. GOETSCHEL, KENNETH E. HILLSTROM,
C     JORGE J. MORE
C
C     **********
      INTEGER MAXFEV,MODE,NFEV,NJEV,NPRINT
      DOUBLE PRECISION FACTOR,FTOL,GTOL,XTOL,ZERO
      DATA FACTOR,ZERO /1.0D2,0.0D0/
      INFO = 0
C
C     CHECK THE INPUT PARAMETERS FOR ERRORS.
C
      IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. N .OR. TOL .LT. ZERO
     *    .OR. LWA .LT. 5*N + M) GO TO 10
C
C     CALL LMSTR.
C
      MAXFEV = 100*(N + 1)
      FTOL = TOL
      XTOL = TOL
      GTOL = ZERO
      MODE = 1
      NPRINT = 0
      CALL LMSTR(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,MAXFEV,
     *           WA(1),MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,IPVT,WA(N+1),
     *           WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))
      IF (INFO .EQ. 8) INFO = 4
   10 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE LMSTR1.
C
      END
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      SUBROUTINE QFORM(M,N,Q,LDQ,WA)
      INTEGER M,N,LDQ
      DOUBLE PRECISION Q(LDQ,M),WA(M)
C     **********
C
C     SUBROUTINE QFORM
C
C     THIS SUBROUTINE PROCEEDS FROM THE COMPUTED QR FACTORIZATION OF
C     AN M BY N MATRIX A TO ACCUMULATE THE M BY M ORTHOGONAL MATRIX
C     Q FROM ITS FACTORED FORM.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE QFORM(M,N,Q,LDQ,WA)
C
C     WHERE
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF ROWS OF A AND THE ORDER OF Q.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF COLUMNS OF A.
C
C       Q IS AN M BY M ARRAY. ON INPUT THE FULL LOWER TRAPEZOID IN
C         THE FIRST MIN(M,N) COLUMNS OF Q CONTAINS THE FACTORED FORM.
C         ON OUTPUT Q HAS BEEN ACCUMULATED INTO A SQUARE MATRIX.
C
C       LDQ IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY Q.
C
C       WA IS A WORK ARRAY OF LENGTH M.
C
C     SUBPROGRAMS CALLED
C
C       FORTRAN-SUPPLIED ... MIN0
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,J,JM1,K,L,MINMN,NP1
      DOUBLE PRECISION ONE,SUM,TEMP,ZERO
      DATA ONE,ZERO /1.0D0,0.0D0/
C
C     ZERO OUT UPPER TRIANGLE OF Q IN THE FIRST MIN(M,N) COLUMNS.
C
      MINMN = MIN0(M,N)
      IF (MINMN .LT. 2) GO TO 30
      DO 20 J = 2, MINMN
         JM1 = J - 1
         DO 10 I = 1, JM1
            Q(I,J) = ZERO
   10       CONTINUE
   20    CONTINUE
```

```
   30 CONTINUE
C
C     INITIALIZE REMAINING COLUMNS TO THOSE OF THE IDENTITY MATRIX.
C
      NP1 = N + 1
      IF (M .LT. NP1) GO TO 60
      DO 50 J = NP1, M
         DO 40 I = 1, M
            Q(I,J) = ZERO
   40       CONTINUE
         Q(J,J) = ONE
   50    CONTINUE
   60 CONTINUE
C
C     ACCUMULATE Q FROM ITS FACTORED FORM.
C
      DO 120 L = 1, MINMN
         K = MINMN - L + 1
         DO 70 I = K, M
            WA(I) = Q(I,K)
            Q(I,K) = ZERO
   70       CONTINUE
         Q(K,K) = ONE
         IF (WA(K) .EQ. ZERO) GO TO 110
         DO 100 J = K, M
            SUM = ZERO
            DO 80 I = K, M
               SUM = SUM + Q(I,J)*WA(I)
   80          CONTINUE
            TEMP = SUM/WA(K)
            DO 90 I = K, M
               Q(I,J) = Q(I,J) - TEMP*WA(I)
   90          CONTINUE
  100       CONTINUE
  110    CONTINUE
  120    CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE QFORM.
C
      END
```

```
      SUBROUTINE QRFAC(M,N,A,LDA,PIVOT,IPVT,LIPVT,RDIAG,ACNORM,WA)
      INTEGER M,N,LDA,LIPVT
      INTEGER IPVT(LIPVT)
      LOGICAL PIVOT
      DOUBLE PRECISION A(LDA,N),RDIAG(N),ACNORM(N),WA(N)
C     **********
C
C     SUBROUTINE QRFAC
C
C     THIS SUBROUTINE USES HOUSEHOLDER TRANSFORMATIONS WITH COLUMN
C     PIVOTING (OPTIONAL) TO COMPUTE A QR FACTORIZATION OF THE
C     M BY N MATRIX A. THAT IS, QRFAC DETERMINES AN ORTHOGONAL
C     MATRIX Q, A PERMUTATION MATRIX P, AND AN UPPER TRAPEZOIDAL
C     MATRIX R WITH DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE,
C     SUCH THAT A*P = Q*R. THE HOUSEHOLDER TRANSFORMATION FOR
C     COLUMN K, K = 1,2,...,MIN(M,N), IS OF THE FORM
C
C                            T
C           I - (1/U(K))*U*U
C
C     WHERE U HAS ZEROS IN THE FIRST K-1 POSITIONS. THE FORM OF
C     THIS TRANSFORMATION AND THE METHOD OF PIVOTING FIRST
C     APPEARED IN THE CORRESPONDING LINPACK SUBROUTINE.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE QRFAC(M,N,A,LDA,PIVOT,IPVT,LIPVT,RDIAG,ACNORM,WA)
C
C     WHERE
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF ROWS OF A.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF COLUMNS OF A.
C
C       A IS AN M BY N ARRAY. ON INPUT A CONTAINS THE MATRIX FOR
C         WHICH THE QR FACTORIZATION IS TO BE COMPUTED. ON OUTPUT
C         THE STRICT UPPER TRAPEZOIDAL PART OF A CONTAINS THE STRICT
C         UPPER TRAPEZOIDAL PART OF R, AND THE LOWER TRAPEZOIDAL
C         PART OF A CONTAINS A FACTORED FORM OF Q (THE NON-TRIVIAL
C         ELEMENTS OF THE U VECTORS DESCRIBED ABOVE).
C
C       LDA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY A.
C
C       PIVOT IS A LOGICAL INPUT VARIABLE. IF PIVOT IS SET TRUE,
C         THEN COLUMN PIVOTING IS ENFORCED. IF PIVOT IS SET FALSE,
C         THEN NO COLUMN PIVOTING IS DONE.
C
C       IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH LIPVT. IPVT
C         DEFINES THE PERMUTATION MATRIX P SUCH THAT A*P = Q*R.
C         COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.
C         IF PIVOT IS FALSE, IPVT IS NOT REFERENCED.
```

```
C
C        LIPVT IS A POSITIVE INTEGER INPUT VARIABLE. IF PIVOT IS FALSE,
C          THEN LIPVT MAY BE AS SMALL AS 1. IF PIVOT IS TRUE, THEN
C          LIPVT MUST BE AT LEAST N.
C
C        RDIAG IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE
C          DIAGONAL ELEMENTS OF R.
C
C        ACNORM IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE
C          NORMS OF THE CORRESPONDING COLUMNS OF THE INPUT MATRIX A.
C          IF THIS INFORMATION IS NOT NEEDED, THEN ACNORM CAN COINCIDE
C          WITH RDIAG.
C
C        WA IS A WORK ARRAY OF LENGTH N. IF PIVOT IS FALSE, THEN WA
C          CAN COINCIDE WITH RDIAG.
C
C      SUBPROGRAMS CALLED
C
C        MINPACK-SUPPLIED ... DPMPAR,ENORM
C
C        FORTRAN-SUPPLIED ... DMAX1,DSQRT,MIN0
C
C      ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C      BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,J,JP1,K,KMAX,MINMN
      DOUBLE PRECISION AJNORM,EPSMCH,ONE,P05,SUM,TEMP,ZERO
      DOUBLE PRECISION DPMPAR,ENORM
      DATA ONE,P05,ZERO /1.0D0,5.0D-2,0.0D0/
C
C     EPSMCH IS THE MACHINE PRECISION.
C
      EPSMCH = DPMPAR(1)
C
C     COMPUTE THE INITIAL COLUMN NORMS AND INITIALIZE SEVERAL ARRAYS.
C
      DO 10 J = 1, N
         ACNORM(J) = ENORM(M,A(1,J))
         RDIAG(J) = ACNORM(J)
         WA(J) = RDIAG(J)
         IF (PIVOT) IPVT(J) = J
   10    CONTINUE
C
C     REDUCE A TO R WITH HOUSEHOLDER TRANSFORMATIONS.
C
      MINMN = MIN0(M,N)
      DO 110 J = 1, MINMN
         IF (.NOT.PIVOT) GO TO 40
C
C        BRING THE COLUMN OF LARGEST NORM INTO THE PIVOT POSITION.
C
         KMAX = J
         DO 20 K = J, N
```

```
            IF (RDIAG(K) .GT. RDIAG(KMAX)) KMAX = K
   20       CONTINUE
         IF (KMAX .EQ. J) GO TO 40
         DO 30 I = 1, M
            TEMP = A(I,J)
            A(I,J) = A(I,KMAX)
            A(I,KMAX) = TEMP
   30       CONTINUE
         RDIAG(KMAX) = RDIAG(J)
         WA(KMAX) = WA(J)
         K = IPVT(J)
         IPVT(J) = IPVT(KMAX)
         IPVT(KMAX) = K
   40    CONTINUE
C
C        COMPUTE THE HOUSEHOLDER TRANSFORMATION TO REDUCE THE
C        J-TH COLUMN OF A TO A MULTIPLE OF THE J-TH UNIT VECTOR.
C
         AJNORM = ENORM(M-J+1,A(J,J))
         IF (AJNORM .EQ. ZERO) GO TO 100
         IF (A(J,J) .LT. ZERO) AJNORM = -AJNORM
         DO 50 I = J, M
            A(I,J) = A(I,J)/AJNORM
   50       CONTINUE
         A(J,J) = A(J,J) + ONE
C
C        APPLY THE TRANSFORMATION TO THE REMAINING COLUMNS
C        AND UPDATE THE NORMS.
C
         JP1 = J + 1
         IF (N .LT. JP1) GO TO 100
         DO 90 K = JP1, N
            SUM = ZERO
            DO 60 I = J, M
               SUM = SUM + A(I,J)*A(I,K)
   60          CONTINUE
            TEMP = SUM/A(J,J)
            DO 70 I = J, M
               A(I,K) = A(I,K) - TEMP*A(I,J)
   70          CONTINUE
            IF (.NOT.PIVOT .OR. RDIAG(K) .EQ. ZERO) GO TO 80
            TEMP = A(J,K)/RDIAG(K)
            RDIAG(K) = RDIAG(K)*DSQRT(DMAX1(ZERO,ONE-TEMP**2))
            IF (P05*(RDIAG(K)/WA(K))**2 .GT. EPSMCH) GO TO 80
            RDIAG(K) = ENORM(M-J,A(JP1,K))
            WA(K) = RDIAG(K)
   80       CONTINUE
   90       CONTINUE
  100    CONTINUE
         RDIAG(J) = -AJNORM
  110    CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE QRFAC.
```

```
      SUBROUTINE QRSOLV(N,R,LDR,IPVT,DIAG,QTB,X,SDIAG,WA)
      INTEGER N,LDR
      INTEGER IPVT(N)
      DOUBLE PRECISION R(LDR,N),DIAG(N),QTB(N),X(N),SDIAG(N),WA(N)
C     **********
C
C     SUBROUTINE QRSOLV
C
C     GIVEN AN M BY N MATRIX A, AN N BY N DIAGONAL MATRIX D,
C     AND AN M-VECTOR B, THE PROBLEM IS TO DETERMINE AN X WHICH
C     SOLVES THE SYSTEM
C
C           A*X = B ,     D*X = 0 ,
C
C     IN THE LEAST SQUARES SENSE.
C
C     THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM
C     IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE
C     QR FACTORIZATION, WITH COLUMN PIVOTING, OF A. THAT IS, IF
C     A*P = Q*R, WHERE P IS A PERMUTATION MATRIX, Q HAS ORTHOGONAL
C     COLUMNS, AND R IS AN UPPER TRIANGULAR MATRIX WITH DIAGONAL
C     ELEMENTS OF NONINCREASING MAGNITUDE, THEN QRSOLV EXPECTS
C     THE FULL UPPER TRIANGLE OF R, THE PERMUTATION MATRIX P,
C     AND THE FIRST N COMPONENTS OF (Q TRANSPOSE)*B. THE SYSTEM
C     A*X = B, D*X = 0, IS THEN EQUIVALENT TO
C
C                  T        T
C           R*Z = Q *B ,  P *D*P*Z = 0 ,
C
C     WHERE X = P*Z. IF THIS SYSTEM DOES NOT HAVE FULL RANK,
C     THEN A LEAST SQUARES SOLUTION IS OBTAINED. ON OUTPUT QRSOLV
C     ALSO PROVIDES AN UPPER TRIANGULAR MATRIX S SUCH THAT
C
C            T   T               T
C           P *(A *A + D*D)*P = S *S .
C
C     S IS COMPUTED WITHIN QRSOLV AND MAY BE OF SEPARATE INTEREST.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE QRSOLV(N,R,LDR,IPVT,DIAG,QTB,X,SDIAG,WA)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R.
C
C       R IS AN N BY N ARRAY. ON INPUT THE FULL UPPER TRIANGLE
C         MUST CONTAIN THE FULL UPPER TRIANGLE OF THE MATRIX R.
C         ON OUTPUT THE FULL UPPER TRIANGLE IS UNALTERED, AND THE
C         STRICT LOWER TRIANGLE CONTAINS THE STRICT UPPER TRIANGLE
C         (TRANSPOSED) OF THE UPPER TRIANGULAR MATRIX S.
C
C       LDR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY R.
```

```
C                                                                       QRSL0550
C        IPVT IS AN INTEGER INPUT ARRAY OF LENGTH N WHICH DEFINES THE   QRSL0560
C          PERMUTATION MATRIX P SUCH THAT A*P = Q*R. COLUMN J OF P      QRSL0570
C          IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.                    QRSL0580
C                                                                       QRSL0590
C        DIAG IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE      QRSL0600
C          DIAGONAL ELEMENTS OF THE MATRIX D.                           QRSL0610
C                                                                       QRSL0620
C        QTB IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE FIRST QRSL0630
C          N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*B.                    QRSL0640
C                                                                       QRSL0650
C        X IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE LEAST      QRSL0660
C          SQUARES SOLUTION OF THE SYSTEM A*X = B, D*X = 0.             QRSL0670
C                                                                       QRSL0680
C        SDIAG IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE        QRSL0690
C          DIAGONAL ELEMENTS OF THE UPPER TRIANGULAR MATRIX S.          QRSL0700
C                                                                       QRSL0710
C        WA IS A WORK ARRAY OF LENGTH N.                                QRSL0720
C                                                                       QRSL0730
C     SUBPROGRAMS CALLED                                                QRSL0740
C                                                                       QRSL0750
C        FORTRAN-SUPPLIED ... DABS,DSQRT                                QRSL0760
C                                                                       QRSL0770
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.         QRSL0780
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE             QRSL0790
C                                                                       QRSL0800
C     **********                                                        QRSL0810
      INTEGER I,J,JP1,K,KP1,L,NSING                                     QRSL0820
      DOUBLE PRECISION COS,COTAN,P5,P25,QTBPJ,SIN,SUM,TAN,TEMP,ZERO     QRSL0830
      DATA P5,P25,ZERO /5.0D-1,2.5D-1,0.0D0/                            QRSL0840
C                                                                       QRSL0850
C     COPY R AND (Q TRANSPOSE)*B TO PRESERVE INPUT AND INITIALIZE S.    QRSL0860
C     IN PARTICULAR, SAVE THE DIAGONAL ELEMENTS OF R IN X.              QRSL0870
C                                                                       QRSL0880
      DO 20 J = 1, N                                                    QRSL0890
         DO 10 I = J, N                                                 QRSL0900
            R(I,J) = R(J,I)                                             QRSL0910
   10       CONTINUE                                                    QRSL0920
         X(J) = R(J,J)                                                  QRSL0930
         WA(J) = QTB(J)                                                 QRSL0940
   20    CONTINUE                                                       QRSL0950
C                                                                       QRSL0960
C     ELIMINATE THE DIAGONAL MATRIX D USING A GIVENS ROTATION.          QRSL0970
C                                                                       QRSL0980
      DO 100 J = 1, N                                                   QRSL0990
C                                                                       QRSL1000
C        PREPARE THE ROW OF D TO BE ELIMINATED, LOCATING THE            QRSL1010
C        DIAGONAL ELEMENT USING P FROM THE QR FACTORIZATION.            QRSL1020
C                                                                       QRSL1030
         L = IPVT(J)                                                    QRSL1040
         IF (DIAG(L) .EQ. ZERO) GO TO 90                                QRSL1050
         DO 30 K = J, N                                                 QRSL1060
            SDIAG(K) = ZERO                                             QRSL1070
   30       CONTINUE                                                    QRSL1080
```

```
         SDIAG(J) = DIAG(L)
C
C        THE TRANSFORMATIONS TO ELIMINATE THE ROW OF D
C        MODIFY ONLY A SINGLE ELEMENT OF (Q TRANSPOSE)*B
C        BEYOND THE FIRST N, WHICH IS INITIALLY ZERO.
C
         QTBPJ = ZERO
         DO 80 K = J, N
C
C           DETERMINE A GIVENS ROTATION WHICH ELIMINATES THE
C           APPROPRIATE ELEMENT IN THE CURRENT ROW OF D.
C
            IF (SDIAG(K) .EQ. ZERO) GO TO 70
            IF (DABS(R(K,K)) .GE. DABS(SDIAG(K))) GO TO 40
               COTAN = R(K,K)/SDIAG(K)
               SIN = P5/DSQRT(P25+P25*COTAN**2)
               COS = SIN*COTAN
               GO TO 50
   40       CONTINUE
               TAN = SDIAG(K)/R(K,K)
               COS = P5/DSQRT(P25+P25*TAN**2)
               SIN = COS*TAN
   50       CONTINUE
C
C           COMPUTE THE MODIFIED DIAGONAL ELEMENT OF R AND
C           THE MODIFIED ELEMENT OF ((Q TRANSPOSE)*B,0).
C
            R(K,K) = COS*R(K,K) + SIN*SDIAG(K)
            TEMP = COS*WA(K) + SIN*QTBPJ
            QTBPJ = -SIN*WA(K) + COS*QTBPJ
            WA(K) = TEMP
C
C           ACCUMULATE THE TRANFORMATION IN THE ROW OF S.
C
            KP1 = K + 1
            IF (N .LT. KP1) GO TO 70
            DO 60 I = KP1, N
               TEMP = COS*R(I,K) + SIN*SDIAG(I)
               SDIAG(I) = -SIN*R(I,K) + COS*SDIAG(I)
               R(I,K) = TEMP
   60          CONTINUE
   70       CONTINUE
   80       CONTINUE
   90    CONTINUE
C
C        STORE THE DIAGONAL ELEMENT OF S AND RESTORE
C        THE CORRESPONDING DIAGONAL ELEMENT OF R.
C
         SDIAG(J) = R(J,J)
         R(J,J) = X(J)
  100    CONTINUE
C
C     SOLVE THE TRIANGULAR SYSTEM FOR Z. IF THE SYSTEM IS
C     SINGULAR, THEN OBTAIN A LEAST SQUARES SOLUTION.
```

```
C
      NSING = N
      DO 110 J = 1, N
         IF (SDIAG(J) .EQ. ZERO .AND. NSING .EQ. N) NSING = J - 1
         IF (NSING .LT. N) WA(J) = ZERO
  110    CONTINUE
      IF (NSING .LT. 1) GO TO 150
      DO 140 K = 1, NSING
         J = NSING - K + 1
         SUM = ZERO
         JP1 = J + 1
         IF (NSING .LT. JP1) GO TO 130
         DO 120 I = JP1, NSING
            SUM = SUM + R(I,J)*WA(I)
  120       CONTINUE
  130    CONTINUE
         WA(J) = (WA(J) - SUM)/SDIAG(J)
  140    CONTINUE
  150 CONTINUE
C
C     PERMUTE THE COMPONENTS OF Z BACK TO COMPONENTS OF X.
C
      DO 160 J = 1, N
         L = IPVT(J)
         X(L) = WA(J)
  160    CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE QRSOLV.
C
      END
```

```
      SUBROUTINE RWUPDT(N,R,LDR,W,B,ALPHA,COS,SIN)
      INTEGER N,LDR
      DOUBLE PRECISION ALPHA
      DOUBLE PRECISION R(LDR,N),W(N),B(N),COS(N),SIN(N)
C     **********
C
C     SUBROUTINE RWUPDT
C
C     GIVEN AN N BY N UPPER TRIANGULAR MATRIX R, THIS SUBROUTINE
C     COMPUTES THE QR DECOMPOSITION OF THE MATRIX FORMED WHEN A ROW
C     IS ADDED TO R. IF THE ROW IS SPECIFIED BY THE VECTOR W, THEN
C     RWUPDT DETERMINES AN ORTHOGONAL MATRIX Q SUCH THAT WHEN THE
C     N+1 BY N MATRIX COMPOSED OF R AUGMENTED BY W IS PREMULTIPLIED
C     BY (Q TRANSPOSE), THE RESULTING MATRIX IS UPPER TRAPEZOIDAL.
C     THE MATRIX (Q TRANSPOSE) IS THE PRODUCT OF N TRANSFORMATIONS
C
C           G(N)*G(N-1)* ... *G(1)
C
C     WHERE G(I) IS A GIVENS ROTATION IN THE (I,N+1) PLANE WHICH
C     ELIMINATES ELEMENTS IN THE (N+1)-ST PLANE. RWUPDT ALSO
C     COMPUTES THE PRODUCT (Q TRANSPOSE)*C WHERE C IS THE
C     (N+1)-VECTOR (B,ALPHA). Q ITSELF IS NOT ACCUMULATED, RATHER
C     THE INFORMATION TO RECOVER THE G ROTATIONS IS SUPPLIED.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE RWUPDT(N,R,LDR,W,B,ALPHA,COS,SIN)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R.
C
C       R IS AN N BY N ARRAY. ON INPUT THE UPPER TRIANGULAR PART OF
C         R MUST CONTAIN THE MATRIX TO BE UPDATED. ON OUTPUT R
C         CONTAINS THE UPDATED TRIANGULAR MATRIX.
C
C       LDR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY R.
C
C       W IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE ROW
C         VECTOR TO BE ADDED TO R.
C
C       B IS AN ARRAY OF LENGTH N. ON INPUT B MUST CONTAIN THE
C         FIRST N ELEMENTS OF THE VECTOR C. ON OUTPUT B CONTAINS
C         THE FIRST N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*C.
C
C       ALPHA IS A VARIABLE. ON INPUT ALPHA MUST CONTAIN THE
C         (N+1)-ST ELEMENT OF THE VECTOR C. ON OUTPUT ALPHA CONTAINS
C         THE (N+1)-ST ELEMENT OF THE VECTOR (Q TRANSPOSE)*C.
C
C       COS IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE
C         COSINES OF THE TRANSFORMING GIVENS ROTATIONS.
C
C       SIN IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE
```

```
C        SINES OF THE TRANSFORMING GIVENS ROTATIONS.                    RWUP0550
C                                                                       RWUP0560
C     SUBPROGRAMS CALLED                                                RWUP0570
C                                                                       RWUP0580
C       FORTRAN-SUPPLIED ... DABS,DSQRT                                 RWUP0590
C                                                                       RWUP0600
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.         RWUP0610
C     BURTON S. GARBOW, DUDLEY V. GOETSCHEL, KENNETH E. HILLSTROM,      RWUP0620
C     JORGE J. MORE                                                     RWUP0630
C                                                                       RWUP0640
C     **********                                                        RWUP0650
      INTEGER I,J,JM1                                                   RWUP0660
      DOUBLE PRECISION COTAN,ONE,P5,P25,ROWJ,TAN,TEMP,ZERO              RWUP0670
      DATA ONE,P5,P25,ZERO /1.0D0,5.0D-1,2.5D-1,0.0D0/                  RWUP0680
C                                                                       RWUP0690
      DO 60 J = 1, N                                                    RWUP0700
         ROWJ = W(J)                                                    RWUP0710
         JM1 = J - 1                                                    RWUP0720
C                                                                       RWUP0730
C        APPLY THE PREVIOUS TRANSFORMATIONS TO                          RWUP0740
C        R(I,J), I=1,2,...,J-1, AND TO W(J).                            RWUP0750
C                                                                       RWUP0760
         IF (JM1 .LT. 1) GO TO 20                                       RWUP0770
         DO 10 I = 1, JM1                                               RWUP0780
            TEMP = COS(I)*R(I,J) + SIN(I)*ROWJ                          RWUP0790
            ROWJ = -SIN(I)*R(I,J) + COS(I)*ROWJ                         RWUP0800
            R(I,J) = TEMP                                               RWUP0810
   10       CONTINUE                                                    RWUP0820
   20    CONTINUE                                                       RWUP0830
C                                                                       RWUP0840
C        DETERMINE A GIVENS ROTATION WHICH ELIMINATES W(J).             RWUP0850
C                                                                       RWUP0860
         COS(J) = ONE                                                   RWUP0870
         SIN(J) = ZERO                                                  RWUP0880
         IF (ROWJ .EQ. ZERO) GO TO 50                                   RWUP0890
         IF (DABS(R(J,J)) .GE. DABS(ROWJ)) GO TO 30                     RWUP0900
            COTAN = R(J,J)/ROWJ                                         RWUP0910
            SIN(J) = P5/DSQRT(P25+P25*COTAN**2)                         RWUP0920
            COS(J) = SIN(J)*COTAN                                       RWUP0930
            GO TO 40                                                    RWUP0940
   30    CONTINUE                                                       RWUP0950
            TAN = ROWJ/R(J,J)                                           RWUP0960
            COS(J) = P5/DSQRT(P25+P25*TAN**2)                           RWUP0970
            SIN(J) = COS(J)*TAN                                         RWUP0980
   40    CONTINUE                                                       RWUP0990
C                                                                       RWUP1000
C        APPLY THE CURRENT TRANSFORMATION TO R(J,J), B(J), AND ALPHA.   RWUP1010
C                                                                       RWUP1020
         R(J,J) = COS(J)*R(J,J) + SIN(J)*ROWJ                           RWUP1030
         TEMP = COS(J)*B(J) + SIN(J)*ALPHA                              RWUP1040
         ALPHA = -SIN(J)*B(J) + COS(J)*ALPHA                            RWUP1050
         B(J) = TEMP                                                    RWUP1060
   50    CONTINUE                                                       RWUP1070
   60    CONTINUE                                                       RWUP1080
```

```
      RETURN                                                          RWUP1090
C                                                                     RWUP1100
C     LAST CARD OF SUBROUTINE RWUPDT.                                 RWUP1110
C                                                                     RWUP1120
      END                                                             RWUP1130
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      SUBROUTINE R1MPYQ(M,N,A,LDA,V,W)
      INTEGER M,N,LDA
      DOUBLE PRECISION A(LDA,N),V(N),W(N)
C     **********
C
C     SUBROUTINE R1MPYQ
C
C     GIVEN AN M BY N MATRIX A, THIS SUBROUTINE COMPUTES A*Q WHERE
C     Q IS THE PRODUCT OF 2*(N - 1) TRANSFORMATIONS
C
C           GV(N-1)*...*GV(1)*GW(1)*...*GW(N-1)
C
C     AND GV(I), GW(I) ARE GIVENS ROTATIONS IN THE (I,N) PLANE WHICH
C     ELIMINATE ELEMENTS IN THE I-TH AND N-TH PLANES, RESPECTIVELY.
C     Q ITSELF IS NOT GIVEN, RATHER THE INFORMATION TO RECOVER THE
C     GV, GW ROTATIONS IS SUPPLIED.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE R1MPYQ(M,N,A,LDA,V,W)
C
C     WHERE
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF ROWS OF A.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF COLUMNS OF A.
C
C       A IS AN M BY N ARRAY. ON INPUT A MUST CONTAIN THE MATRIX
C         TO BE POSTMULTIPLIED BY THE ORTHOGONAL MATRIX Q
C         DESCRIBED ABOVE. ON OUTPUT A*Q HAS REPLACED A.
C
C       LDA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY A.
C
C       V IS AN INPUT ARRAY OF LENGTH N. V(I) MUST CONTAIN THE
C         INFORMATION NECESSARY TO RECOVER THE GIVENS ROTATION GV(I)
C         DESCRIBED ABOVE.
C
C       W IS AN INPUT ARRAY OF LENGTH N. W(I) MUST CONTAIN THE
C         INFORMATION NECESSARY TO RECOVER THE GIVENS ROTATION GW(I)
C         DESCRIBED ABOVE.
C
C     SUBROUTINES CALLED
C
C       FORTRAN-SUPPLIED ... DABS,DSQRT
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,J,NMJ,NM1
      DOUBLE PRECISION COS,ONE,SIN,TEMP
```

```
      DATA ONE /1.0D0/
C
C     APPLY THE FIRST SET OF GIVENS ROTATIONS TO A.
C
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 50
      DO 20 NMJ = 1, NM1
         J = N - NMJ
         IF (DABS(V(J)) .GT. ONE) COS = ONE/V(J)
         IF (DABS(V(J)) .GT. ONE) SIN = DSQRT(ONE-COS**2)
         IF (DABS(V(J)) .LE. ONE) SIN = V(J)
         IF (DABS(V(J)) .LE. ONE) COS = DSQRT(ONE-SIN**2)
         DO 10 I = 1, M
            TEMP = COS*A(I,J) - SIN*A(I,N)
            A(I,N) = SIN*A(I,J) + COS*A(I,N)
            A(I,J) = TEMP
   10       CONTINUE
   20    CONTINUE
C
C     APPLY THE SECOND SET OF GIVENS ROTATIONS TO A.
C
      DO 40 J = 1, NM1
         IF (DABS(W(J)) .GT. ONE) COS = ONE/W(J)
         IF (DABS(W(J)) .GT. ONE) SIN = DSQRT(ONE-COS**2)
         IF (DABS(W(J)) .LE. ONE) SIN = W(J)
         IF (DABS(W(J)) .LE. ONE) COS = DSQRT(ONE-SIN**2)
         DO 30 I = 1, M
            TEMP = COS*A(I,J) + SIN*A(I,N)
            A(I,N) = -SIN*A(I,J) + COS*A(I,N)
            A(I,J) = TEMP
   30       CONTINUE
   40    CONTINUE
   50 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE R1MPYQ.
C
      END
```

```
      SUBROUTINE R1UPDT(M,N,S,LS,U,V,W,SING)
      INTEGER M,N,LS
      LOGICAL SING
      DOUBLE PRECISION S(LS),U(M),V(N),W(M)
C     **********
C
C     SUBROUTINE R1UPDT
C
C     GIVEN AN M BY N LOWER TRAPEZOIDAL MATRIX S, AN M-VECTOR U,
C     AND AN N-VECTOR V, THE PROBLEM IS TO DETERMINE AN
C     ORTHOGONAL MATRIX Q SUCH THAT
C
C                   T
C             (S + U*V )*Q
C
C     IS AGAIN LOWER TRAPEZOIDAL.
C
C     THIS SUBROUTINE DETERMINES Q AS THE PRODUCT OF 2*(N - 1)
C     TRANSFORMATIONS
C
C           GV(N-1)*...*GV(1)*GW(1)*...*GW(N-1)
C
C     WHERE GV(I), GW(I) ARE GIVENS ROTATIONS IN THE (I,N) PLANE
C     WHICH ELIMINATE ELEMENTS IN THE I-TH AND N-TH PLANES,
C     RESPECTIVELY. Q ITSELF IS NOT ACCUMULATED, RATHER THE
C     INFORMATION TO RECOVER THE GV, GW ROTATIONS IS RETURNED.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE R1UPDT(M,N,S,LS,U,V,W,SING)
C
C     WHERE
C
C       M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF ROWS OF S.
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C         OF COLUMNS OF S. N MUST NOT EXCEED M.
C
C       S IS AN ARRAY OF LENGTH LS. ON INPUT S MUST CONTAIN THE LOWER
C         TRAPEZOIDAL MATRIX S STORED BY COLUMNS. ON OUTPUT S CONTAINS
C         THE LOWER TRAPEZOIDAL MATRIX PRODUCED AS DESCRIBED ABOVE.
C
C       LS IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN
C         (N*(2*M-N+1))/2.
C
C       U IS AN INPUT ARRAY OF LENGTH M WHICH MUST CONTAIN THE
C         VECTOR U.
C
C       V IS AN ARRAY OF LENGTH N. ON INPUT V MUST CONTAIN THE VECTOR
C         V. ON OUTPUT V(I) CONTAINS THE INFORMATION NECESSARY TO
C         RECOVER THE GIVENS ROTATION GV(I) DESCRIBED ABOVE.
C
C       W IS AN OUTPUT ARRAY OF LENGTH M. W(I) CONTAINS INFORMATION
```

```
C           NECESSARY TO RECOVER THE GIVENS ROTATION GW(I) DESCRIBED
C           ABOVE.
C
C         SING IS A LOGICAL OUTPUT VARIABLE. SING IS SET TRUE IF ANY
C           OF THE DIAGONAL ELEMENTS OF THE OUTPUT S ARE ZERO. OTHERWISE
C           SING IS SET FALSE.
C
C       SUBPROGRAMS CALLED
C
C         MINPACK-SUPPLIED ... DPMPAR
C
C         FORTRAN-SUPPLIED ... DABS,DSQRT
C
C       ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C       BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE,
C       JOHN L. NAZARETH
C
C       **********
        INTEGER I,J,JJ,L,NMJ,NM1
        DOUBLE PRECISION COS,COTAN,GIANT,ONE,P5,P25,SIN,TAN,TAU,TEMP,
     *                   ZERO
        DOUBLE PRECISION DPMPAR
        DATA ONE,P5,P25,ZERO /1.0D0,5.0D-1,2.5D-1,0.0D0/
C
C       GIANT IS THE LARGEST MAGNITUDE.
C
        GIANT = DPMPAR(3)
C
C       INITIALIZE THE DIAGONAL ELEMENT POINTER.
C
        JJ = (N*(2*M - N + 1))/2 - (M - N)
C
C       MOVE THE NONTRIVIAL PART OF THE LAST COLUMN OF S INTO W.
C
        L = JJ
        DO 10 I = N, M
           W(I) = S(L)
           L = L + 1
   10      CONTINUE
C
C       ROTATE THE VECTOR V INTO A MULTIPLE OF THE N-TH UNIT VECTOR
C       IN SUCH A WAY THAT A SPIKE IS INTRODUCED INTO W.
C
        NM1 = N - 1
        IF (NM1 .LT. 1) GO TO 70
        DO 60 NMJ = 1, NM1
           J = N - NMJ
           JJ = JJ - (M - J + 1)
           W(J) = ZERO
           IF (V(J) .EQ. ZERO) GO TO 50
C
C          DETERMINE A GIVENS ROTATION WHICH ELIMINATES THE
C          J-TH ELEMENT OF V.
C
```

```
         IF (DABS(V(N)) .GE. DABS(V(J))) GO TO 20
            COTAN = V(N)/V(J)
            SIN = P5/DSQRT(P25+P25*COTAN**2)
            COS = SIN*COTAN
            TAU = ONE
            IF (DABS(COS)*GIANT .GT. ONE) TAU = ONE/COS
            GO TO 30
   20    CONTINUE
            TAN = V(J)/V(N)
            COS = P5/DSQRT(P25+P25*TAN**2)
            SIN = COS*TAN
            TAU = SIN
   30    CONTINUE
C
C        APPLY THE TRANSFORMATION TO V AND STORE THE INFORMATION
C        NECESSARY TO RECOVER THE GIVENS ROTATION.
C
         V(N) = SIN*V(J) + COS*V(N)
         V(J) = TAU
C
C        APPLY THE TRANSFORMATION TO S AND EXTEND THE SPIKE IN W.
C
         L = JJ
         DO 40 I = J, M
            TEMP = COS*S(L) - SIN*W(I)
            W(I) = SIN*S(L) + COS*W(I)
            S(L) = TEMP
            L = L + 1
   40       CONTINUE
   50    CONTINUE
   60    CONTINUE
   70 CONTINUE
C
C     ADD THE SPIKE FROM THE RANK 1 UPDATE TO W.
C
      DO 80 I = 1, M
         W(I) = W(I) + V(N)*U(I)
   80    CONTINUE
C
C     ELIMINATE THE SPIKE.
C
      SING = .FALSE.
      IF (NM1 .LT. 1) GO TO 140
      DO 130 J = 1, NM1
         IF (W(J) .EQ. ZERO) GO TO 120
C
C        DETERMINE A GIVENS ROTATION WHICH ELIMINATES THE
C        J-TH ELEMENT OF THE SPIKE.
C
         IF (DABS(S(JJ)) .GE. DABS(W(J))) GO TO 90
            COTAN = S(JJ)/W(J)
            SIN = P5/DSQRT(P25+P25*COTAN**2)
            COS = SIN*COTAN
            TAU = ONE
```

```
            IF (DABS(COS)*GIANT .GT. ONE) TAU = ONE/COS
            GO TO 100
   90    CONTINUE
            TAN = W(J)/S(JJ)
            COS = P5/DSQRT(P25+P25*TAN**2)
            SIN = COS*TAN
            TAU = SIN
  100    CONTINUE
C
C        APPLY THE TRANSFORMATION TO S AND REDUCE THE SPIKE IN W.
C
         L = JJ
         DO 110 I = J, M
            TEMP = COS*S(L) + SIN*W(I)
            W(I) = -SIN*S(L) + COS*W(I)
            S(L) = TEMP
            L = L + 1
  110       CONTINUE
C
C        STORE THE INFORMATION NECESSARY TO RECOVER THE
C        GIVENS ROTATION.
C
         W(J) = TAU
  120    CONTINUE
C
C        TEST FOR ZERO DIAGONAL ELEMENTS IN THE OUTPUT S.
C
         IF (S(JJ) .EQ. ZERO) SING = .TRUE.
         JJ = JJ + (M - J + 1)
  130    CONTINUE
  140 CONTINUE
C
C     MOVE W BACK INTO THE LAST COLUMN OF THE OUTPUT S.
C
      L = JJ
      DO 150 I = N, M
         S(L) = W(I)
         L = L + 1
  150    CONTINUE
      IF (S(JJ) .EQ. ZERO) SING = .TRUE.
      RETURN
C
C     LAST CARD OF SUBROUTINE R1UPDT.
C
      END
```

```
      REAL FUNCTION SPMPAR(I)
      INTEGER I
C     **********
C
C     FUNCTION SPMPAR
C
C     THIS FUNCTION PROVIDES SINGLE PRECISION MACHINE PARAMETERS
C     WHEN THE APPROPRIATE SET OF DATA STATEMENTS IS ACTIVATED (BY
C     REMOVING THE C FROM COLUMN 1) AND ALL OTHER DATA STATEMENTS ARE
C     RENDERED INACTIVE. MOST OF THE PARAMETER VALUES WERE OBTAINED
C     FROM THE CORRESPONDING BELL LABORATORIES PORT LIBRARY FUNCTION.
C
C     THE FUNCTION STATEMENT IS
C
C       REAL FUNCTION SPMPAR(I)
C
C     WHERE
C
C       I IS AN INTEGER INPUT VARIABLE SET TO 1, 2, OR 3 WHICH
C         SELECTS THE DESIRED MACHINE PARAMETER. IF THE MACHINE HAS
C         T BASE B DIGITS AND ITS SMALLEST AND LARGEST EXPONENTS ARE
C         EMIN AND EMAX, RESPECTIVELY, THEN THESE PARAMETERS ARE
C
C         SPMPAR(1) = B**(1 - T), THE MACHINE PRECISION,
C
C         SPMPAR(2) = B**(EMIN - 1), THE SMALLEST MAGNITUDE,
C
C         SPMPAR(3) = B**EMAX*(1 - B**(-T)), THE LARGEST MAGNITUDE.
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER MCHEPS(2)
      INTEGER MINMAG(2)
      INTEGER MAXMAG(2)
      REAL RMACH(3)
      EQUIVALENCE (RMACH(1),MCHEPS(1))
      EQUIVALENCE (RMACH(2),MINMAG(1))
      EQUIVALENCE (RMACH(3),MAXMAG(1))
C
C     MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
C     THE AMDAHL 470/V6, THE ICL 2900, THE ITEL AS/6,
C     THE XEROX SIGMA 5/7/9 AND THE SEL SYSTEMS 85/86.
C
      DATA RMACH(1) / Z3C100000 /
      DATA RMACH(2) / Z00100000 /
      DATA RMACH(3) / Z7FFFFFFF /
C
C     MACHINE CONSTANTS FOR THE HONEYWELL 600/6000 SERIES.
C
C     DATA RMACH(1) / O716400000000 /
C     DATA RMACH(2) / O402400000000 /
C     DATA RMACH(3) / O376777777777 /
```

```
C                                                                       SPPR0550
C     MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.                   SPPR0560
C                                                                       SPPR0570
C     DATA RMACH(1) / 16414000000000000000B /                           SPPR0580
C     DATA RMACH(2) / 00014000000000000000B /                           SPPR0590
C     DATA RMACH(3) / 37767777777777777777B /                           SPPR0600
C                                                                       SPPR0610
C     MACHINE CONSTANTS FOR THE PDP-10 (KA OR KI PROCESSOR).            SPPR0620
C                                                                       SPPR0630
C     DATA RMACH(1) / "147400000000 /                                   SPPR0640
C     DATA RMACH(2) / "000400000000 /                                   SPPR0650
C     DATA RMACH(3) / "377777777777 /                                   SPPR0660
C                                                                       SPPR0670
C     MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING               SPPR0680
C     32-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).                 SPPR0690
C                                                                       SPPR0700
C     DATA MCHEPS(1) /  889192448 /                                     SPPR0710
C     DATA MINMAG(1) /    8388608 /                                     SPPR0720
C     DATA MAXMAG(1) / 2147483647 /                                     SPPR0730
C                                                                       SPPR0740
C     DATA RMACH(1) / 006500000000 /                                    SPPR0750
C     DATA RMACH(2) / 000040000000 /                                    SPPR0760
C     DATA RMACH(3) / 017777777777 /                                    SPPR0770
C                                                                       SPPR0780
C     MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING               SPPR0790
C     16-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).                 SPPR0800
C                                                                       SPPR0810
C     DATA MCHEPS(1),MCHEPS(2) / 13568,     0 /                         SPPR0820
C     DATA MINMAG(1),MINMAG(2) /   128,     0 /                         SPPR0830
C     DATA MAXMAG(1),MAXMAG(2) / 32767,    -1 /                         SPPR0840
C                                                                       SPPR0850
C     DATA MCHEPS(1),MCHEPS(2) / 0032400, 0000000 /                     SPPR0860
C     DATA MINMAG(1),MINMAG(2) / 0000200, 0000000 /                     SPPR0870
C     DATA MAXMAG(1),MAXMAG(2) / 0077777, 0177777 /                     SPPR0880
C                                                                       SPPR0890
C     MACHINE CONSTANTS FOR THE BURROUGHS 5700/6700/7700 SYSTEMS.       SPPR0900
C                                                                       SPPR0910
C     DATA RMACH(1) / O1301000000000000 /                               SPPR0920
C     DATA RMACH(2) / O1771000000000000 /                               SPPR0930
C     DATA RMACH(3) / O0777777777777777 /                               SPPR0940
C                                                                       SPPR0950
C     MACHINE CONSTANTS FOR THE BURROUGHS 1700 SYSTEM.                  SPPR0960
C                                                                       SPPR0970
C     DATA RMACH(1) / Z4EA800000 /                                      SPPR0980
C     DATA RMACH(2) / Z400800000 /                                      SPPR0990
C     DATA RMACH(3) / Z5FFFFFFFF /                                      SPPR1000
C                                                                       SPPR1010
C     MACHINE CONSTANTS FOR THE UNIVAC 1100 SERIES.                     SPPR1020
C                                                                       SPPR1030
C     DATA RMACH(1) / O147400000000 /                                   SPPR1040
C     DATA RMACH(2) / O000400000000 /                                   SPPR1050
C     DATA RMACH(3) / O377777777777 /                                   SPPR1060
C                                                                       SPPR1070
C     MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/200.             SPPR1080
```

```
C
C     NOTE - IT MAY BE APPROPRIATE TO INCLUDE THE FOLLOWING CARD -
C     STATIC RMACH(3)
C
C     DATA MINMAG/20K,0/,MAXMAG/77777K,177777K/
C     DATA MCHEPS/36020K,0/
C
C     MACHINE CONSTANTS FOR THE HARRIS 220.
C
C     DATA MCHEPS(1),MCHEPS(2) / '20000000, '00000353 /
C     DATA MINMAG(1),MINMAG(2) / '20000000, '00000201 /
C     DATA MAXMAG(1),MAXMAG(2) / '37777777, '00000177 /
C
C     MACHINE CONSTANTS FOR THE CRAY-1.
C
C     DATA RMACH(1) / 0377224000000000000000B /
C     DATA RMACH(2) / 0200034000000000000000B /
C     DATA RMACH(3) / 0577777777777777777776B /
C
C     MACHINE CONSTANTS FOR THE PRIME 400.
C
C     DATA MCHEPS(1) / :10000000153 /
C     DATA MINMAG(1) / :10000000000 /
C     DATA MAXMAG(1) / :17777777777 /
C
      SPMPAR = RMACH(I)
      RETURN
C
C     LAST CARD OF FUNCTION SPMPAR.
C
      END
```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```
      DOUBLE PRECISION FUNCTION DPMPAR(I)
      INTEGER I
C     **********
C
C     FUNCTION DPMPAR
C
C     THIS FUNCTION PROVIDES DOUBLE PRECISION MACHINE PARAMETERS
C     WHEN THE APPROPRIATE SET OF DATA STATEMENTS IS ACTIVATED (BY
C     REMOVING THE C FROM COLUMN 1) AND ALL OTHER DATA STATEMENTS ARE
C     RENDERED INACTIVE. MOST OF THE PARAMETER VALUES WERE OBTAINED
C     FROM THE CORRESPONDING BELL LABORATORIES PORT LIBRARY FUNCTION.
C
C     THE FUNCTION STATEMENT IS
C
C       DOUBLE PRECISION FUNCTION DPMPAR(I)
C
C     WHERE
C
C       I IS AN INTEGER INPUT VARIABLE SET TO 1, 2, OR 3 WHICH
C         SELECTS THE DESIRED MACHINE PARAMETER. IF THE MACHINE HAS
C         T BASE B DIGITS AND ITS SMALLEST AND LARGEST EXPONENTS ARE
C         EMIN AND EMAX, RESPECTIVELY, THEN THESE PARAMETERS ARE
C
C         DPMPAR(1) = B**(1 - T), THE MACHINE PRECISION,
C
C         DPMPAR(2) = B**(EMIN - 1), THE SMALLEST MAGNITUDE,
C
C         DPMPAR(3) = B**EMAX*(1 - B**(-T)), THE LARGEST MAGNITUDE.
C
C     ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER MCHEPS(4)
      INTEGER MINMAG(4)
      INTEGER MAXMAG(4)
      DOUBLE PRECISION DMACH(3)
      EQUIVALENCE (DMACH(1),MCHEPS(1))
      EQUIVALENCE (DMACH(2),MINMAG(1))
      EQUIVALENCE (DMACH(3),MAXMAG(1))
C
C     MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
C     THE AMDAHL 470/V6, THE ICL 2900, THE ITEL AS/6,
C     THE XEROX SIGMA 5/7/9 AND THE SEL SYSTEMS 85/86.
C
      DATA MCHEPS(1),MCHEPS(2) / Z34100000, Z00000000 /
      DATA MINMAG(1),MINMAG(2) / Z00100000, Z00000000 /
      DATA MAXMAG(1),MAXMAG(2) / Z7FFFFFFF, ZFFFFFFFF /
C
C     MACHINE CONSTANTS FOR THE HONEYWELL 600/6000 SERIES.
C
C     DATA MCHEPS(1),MCHEPS(2) / O606400000000, O000000000000 /
C     DATA MINMAG(1),MINMAG(2) / O402400000000, O000000000000 /
C     DATA MAXMAG(1),MAXMAG(2) / O376777777777, O777777777777 /
```

```
C                                                                       DPPR0550
C     MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.                   DPPR0560
C                                                                       DPPR0570
C     DATA MCHEPS(1) / 15614000000000000000B /                          DPPR0580
C     DATA MCHEPS(2) / 15010000000000000000B /                          DPPR0590
C                                                                       DPPR0600
C     DATA MINMAG(1) / 00604000000000000000B /                          DPPR0610
C     DATA MINMAG(2) / 00000000000000000000B /                          DPPR0620
C                                                                       DPPR0630
C     DATA MAXMAG(1) / 37767777777777777777B /                          DPPR0640
C     DATA MAXMAG(2) / 37167777777777777777B /                          DPPR0650
C                                                                       DPPR0660
C     MACHINE CONSTANTS FOR THE PDP-10 (KA PROCESSOR).                  DPPR0670
C                                                                       DPPR0680
C     DATA MCHEPS(1),MCHEPS(2) / "114400000000, "000000000000 /         DPPR0690
C     DATA MINMAG(1),MINMAG(2) / "033400000000, "000000000000 /         DPPR0700
C     DATA MAXMAG(1),MAXMAG(2) / "377777777777, "344777777777 /         DPPR0710
C                                                                       DPPR0720
C     MACHINE CONSTANTS FOR THE PDP-10 (KI PROCESSOR).                  DPPR0730
C                                                                       DPPR0740
C     DATA MCHEPS(1),MCHEPS(2) / "104400000000, "000000000000 /         DPPR0750
C     DATA MINMAG(1),MINMAG(2) / "000400000000, "000000000000 /         DPPR0760
C     DATA MAXMAG(1),MAXMAG(2) / "377777777777, "377777777777 /         DPPR0770
C                                                                       DPPR0780
C     MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING               DPPR0790
C     32-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).                 DPPR0800
C                                                                       DPPR0810
C     DATA MCHEPS(1),MCHEPS(2) /  620756992,           0 /              DPPR0820
C     DATA MINMAG(1),MINMAG(2) /    8388608,           0 /              DPPR0830
C     DATA MAXMAG(1),MAXMAG(2) / 2147483647,          -1 /              DPPR0840
C                                                                       DPPR0850
C     DATA MCHEPS(1),MCHEPS(2) / 004500000000, 000000000000 /           DPPR0860
C     DATA MINMAG(1),MINMAG(2) / 000040000000, 000000000000 /           DPPR0870
C     DATA MAXMAG(1),MAXMAG(2) / 017777777777, 037777777777 /           DPPR0880
C                                                                       DPPR0890
C     MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING               DPPR0900
C     16-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).                 DPPR0910
C                                                                       DPPR0920
C     DATA MCHEPS(1),MCHEPS(2) /   9472,      0 /                       DPPR0930
C     DATA MCHEPS(3),MCHEPS(4) /      0,      0 /                       DPPR0940
C                                                                       DPPR0950
C     DATA MINMAG(1),MINMAG(2) /    128,      0 /                       DPPR0960
C     DATA MINMAG(3),MINMAG(4) /      0,      0 /                       DPPR0970
C                                                                       DPPR0980
C     DATA MAXMAG(1),MAXMAG(2) /  32767,     -1 /                       DPPR0990
C     DATA MAXMAG(3),MAXMAG(4) /     -1,     -1 /                       DPPR1000
C                                                                       DPPR1010
C     DATA MCHEPS(1),MCHEPS(2) / 0022400, 0000000 /                     DPPR1020
C     DATA MCHEPS(3),MCHEPS(4) / 0000000, 0000000 /                     DPPR1030
C                                                                       DPPR1040
C     DATA MINMAG(1),MINMAG(2) / 0000200, 0000000 /                     DPPR1050
C     DATA MINMAG(3),MINMAG(4) / 0000000, 0000000 /                     DPPR1060
C                                                                       DPPR1070
C     DATA MAXMAG(1),MAXMAG(2) / 0077777, 0177777 /                     DPPR1080
```

```
C     DATA MAXMAG(3),MAXMAG(4) / 0177777, 0177777 /                    DPPR1090
C                                                                      DPPR1100
C     MACHINE CONSTANTS FOR THE BURROUGHS 6700/7700 SYSTEMS.           DPPR1110
C                                                                      DPPR1120
C     DATA MCHEPS(1) / 01451000000000000 /                             DPPR1130
C     DATA MCHEPS(2) / 00000000000000000 /                             DPPR1140
C                                                                      DPPR1150
C     DATA MINMAG(1) / 01771000000000000 /                             DPPR1160
C     DATA MINMAG(2) / 07770000000000000 /                             DPPR1170
C                                                                      DPPR1180
C     DATA MAXMAG(1) / 00777777777777777 /                             DPPR1190
C     DATA MAXMAG(2) / 07777777777777777 /                             DPPR1200
C                                                                      DPPR1210
C     MACHINE CONSTANTS FOR THE BURROUGHS 5700 SYSTEM.                 DPPR1220
C                                                                      DPPR1230
C     DATA MCHEPS(1) / 01451000000000000 /                             DPPR1240
C     DATA MCHEPS(2) / 00000000000000000 /                             DPPR1250
C                                                                      DPPR1260
C     DATA MINMAG(1) / 01771000000000000 /                             DPPR1270
C     DATA MINMAG(2) / 00000000000000000 /                             DPPR1280
C                                                                      DPPR1290
C     DATA MAXMAG(1) / 00777777777777777 /                             DPPR1300
C     DATA MAXMAG(2) / 00007777777777777 /                             DPPR1310
C                                                                      DPPR1320
C     MACHINE CONSTANTS FOR THE BURROUGHS 1700 SYSTEM.                 DPPR1330
C                                                                      DPPR1340
C     DATA MCHEPS(1) / ZCC6800000 /                                    DPPR1350
C     DATA MCHEPS(2) / Z000000000 /                                    DPPR1360
C                                                                      DPPR1370
C     DATA MINMAG(1) / ZC00800000 /                                    DPPR1380
C     DATA MINMAG(2) / Z000000000 /                                    DPPR1390
C                                                                      DPPR1400
C     DATA MAXMAG(1) / ZDFFFFFFFF /                                    DPPR1410
C     DATA MAXMAG(2) / ZFFFFFFFFF /                                    DPPR1420
C                                                                      DPPR1430
C     MACHINE CONSTANTS FOR THE UNIVAC 1100 SERIES.                    DPPR1440
C                                                                      DPPR1450
C     DATA MCHEPS(1),MCHEPS(2) / 0170640000000, 0000000000000 /        DPPR1460
C     DATA MINMAG(1),MINMAG(2) / 0000040000000, 0000000000000 /        DPPR1470
C     DATA MAXMAG(1),MAXMAG(2) / 0377777777777, 0777777777777 /        DPPR1480
C                                                                      DPPR1490
C     MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/200.             DPPR1500
C                                                                      DPPR1510
C     NOTE - IT MAY BE APPROPRIATE TO INCLUDE THE FOLLOWING CARD -     DPPR1520
C     STATIC DMACH(3)                                                  DPPR1530
C                                                                      DPPR1540
C     DATA MINMAG/20K,3*0/,MAXMAG/77777K,3*177777K/                    DPPR1550
C     DATA MCHEPS/32020K,3*0/                                          DPPR1560
C                                                                      DPPR1570
C     MACHINE CONSTANTS FOR THE HARRIS 220.                            DPPR1580
C                                                                      DPPR1590
C     DATA MCHEPS(1),MCHEPS(2) / '20000000, '00000334 /                DPPR1600
C     DATA MINMAG(1),MINMAG(2) / '20000000, '00000201 /                DPPR1610
C     DATA MAXMAG(1),MAXMAG(2) / '37777777, '37777577 /                DPPR1620
```

```
C                                                                       DPPR1630
C     MACHINE CONSTANTS FOR THE CRAY-1.                                 DPPR1640
C                                                                       DPPR1650
C     DATA MCHEPS(1) / 0376424000000000000000B /                        DPPR1660
C     DATA MCHEPS(2) / 0000000000000000000000B /                        DPPR1670
C                                                                       DPPR1680
C     DATA MINMAG(1) / 0200034000000000000000B /                        DPPR1690
C     DATA MINMAG(2) / 0000000000000000000000B /                        DPPR1700
C                                                                       DPPR1710
C     DATA MAXMAG(1) / 0577777777777777777777B /                        DPPR1720
C     DATA MAXMAG(2) / 0000007777777777777776B /                        DPPR1730
C                                                                       DPPR1740
C     MACHINE CONSTANTS FOR THE PRIME 400.                              DPPR1750
C                                                                       DPPR1760
C     DATA MCHEPS(1),MCHEPS(2) / :10000000000, :00000000123 /           DPPR1770
C     DATA MINMAG(1),MINMAG(2) / :10000000000, :00000100000 /           DPPR1780
C     DATA MAXMAG(1),MAXMAG(2) / :17777777777, :37777677776 /           DPPR1790
C                                                                       DPPR1800
      DPMPAR = DMACH(I)                                                 DPPR1810
      RETURN                                                            DPPR1820
C                                                                       DPPR1830
C     LAST CARD OF FUNCTION DPMPAR.                                     DPPR1840
C                                                                       DPPR1850
      END                                                               DPPR1860
```

Distribution for ANL-80-74

Internal:

J. M. Boyle
W. J. Cody
T. F. Coleman
W. R. Cowell
J. J. Dongarra
B. S. Garbow (100)
G. T. Garvey
K. E. Hillstrom
A. B. Krisciunas
J. N. Lyness
P. C. Messina
M. Minkoff
J. J. Moré
National Energy Software Center (100)
D. M. Pahis
L. M. Phebus (2)
G. W. Pieper
R. C. Raffenetti
R. J. Royston
D. C. Sorensen
B. T. Smith
ANL Contract File
ANL Libraries
TIS Files (6)

External:

DOE-TIC, for distribution per UC-32 (183)
Manager, Chicago Operations and Regional Office, DOE-CORO
Chief, Office of Patent Counsel, DOE-CORO
President, Argonne Universities Association
Applied Mathematics Division Review Committee:
  G. Estrin, U. California, Los Angeles
  W. M. Gentleman, U. Waterloo
  J. M. Ortega, U. Virginia
  E. N. Pinson, Bell Telephone Labs.
  S. Rosen, Purdue U.
  M. F. Wheeler, Rice U.
  D. M. Young, Jr., U. Texas at Austin
International Mathematical and Statistical Libraries (100)

☆ U.S. GOVERNMENT PRINTING OFFICE: 1980 — 651-118/22