

1880
ANL-80-74

*159 9TC
10-28-80*

ANL-80-74

MASTER

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

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

The facilities of Argonne National Laboratory are owned by the United States Government. Under the terms of a contract (W-31-109-Eng-38) among the U. S. Department of Energy, Argonne Universities Association and The University of Chicago, the University employs the staff and operates the Laboratory in accordance with policies and programs formulated, approved and reviewed by the Association.

MEMBERS OF ARGONNE UNIVERSITIES ASSOCIATION

The University of Arizona
Carnegie-Mellon University
Case Western Reserve University
The University of Chicago
University of Cincinnati
Illinois Institute of Technology
University of Illinois
Indiana University
The University of Iowa
Iowa State University

The University of Kansas
Kansas State University
Loyola University of Chicago
Marquette University
The University of Michigan
Michigan State University
University of Minnesota
University of Missouri
Northwestern University
University of Notre Dame

The Ohio State University
Ohio University
The Pennsylvania State University
Purdue University
Saint Louis University
Southern Illinois University
The University of Texas at Austin
Washington University
Wayne State University
The University of Wisconsin-Madison

NOTICE

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government or any agency thereof, nor any of their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

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

Applied Mathematics Division

August 1980

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

[Handwritten signature]

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

TABLE OF CONTENTS

Abstract	5
Preface	5
Acknowledgments	8
CHAPTER 1. Introduction to MINPACK-1	9
1.1 Systems of Nonlinear Equations	9
1.2 Nonlinear Least Squares Problems	9
1.3 Derivative Checking	10
1.4 Algorithmic Paths: Core Subroutines and Easy-to-Use Drivers	10
1.5 MINPACK-1 Subroutines: Systems of Nonlinear Equations	10
1.6 MINPACK-1 Subroutines: Nonlinear Least Squares Problems	12
1.7 Machine-Dependent Constants	13
1.8 MINPACK-1 Internal Subprograms	14
CHAPTER 2. Algorithmic Details	17
2.1 Mathematical Background	17
2.2 Overview of the Algorithms	19
2.3 Convergence Criteria	21
2.4 Approximations to the Jacobian Matrix	26
2.5 Scaling	28
2.6 Subroutine FCN: Calculation of the Function and Jacobian Matrix ..	30
2.7 Constraints	33
2.8 Error Bounds	35
2.9 Printing	43
CHAPTER 3. Notes and References	45
CHAPTER 4. Documentation	49
CHAPTER 5. Program Listings	139

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 (More, 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 More [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 (Iitel)
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, \dots, f_n of the n variables x_1, x_2, \dots, x_n are specified, then MINPACK-1 subroutines can be used to find values for x_1, x_2, \dots, x_n that solve the system of nonlinear equations

$$f_i(x_1, x_2, \dots, x_n) = 0, \quad 1 \leq i \leq n.$$

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

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

1.2 Nonlinear Least Squares Problems

If m functions f_1, f_2, \dots, f_m of the n variables x_1, x_2, \dots, x_n are specified with $m \geq n$, then MINPACK-1 subroutines can be used to find values for x_1, x_2, \dots, 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

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

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

$$f_i(x_1, x_2, \dots, x_n) = 0, \quad 1 \leq i \leq n$$

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, \dots, 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, \dots, f_n and the Jacobian matrix

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

(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.

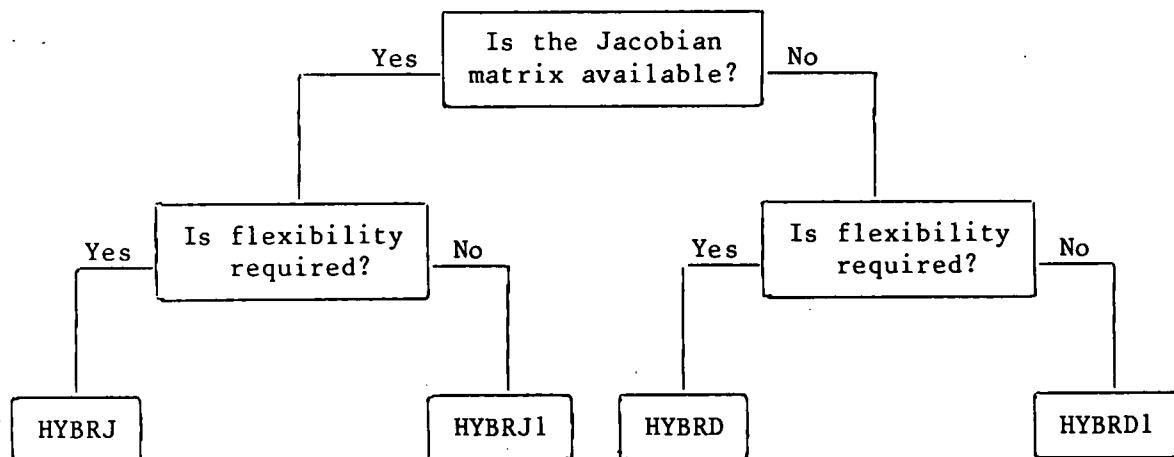


Figure 1
Decision Tree for Systems of Nonlinear Equations

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

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

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, \dots, 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, \dots, f_m and the Jacobian matrix

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

(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, \dots, f_m and the rows of the Jacobian matrix

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

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.

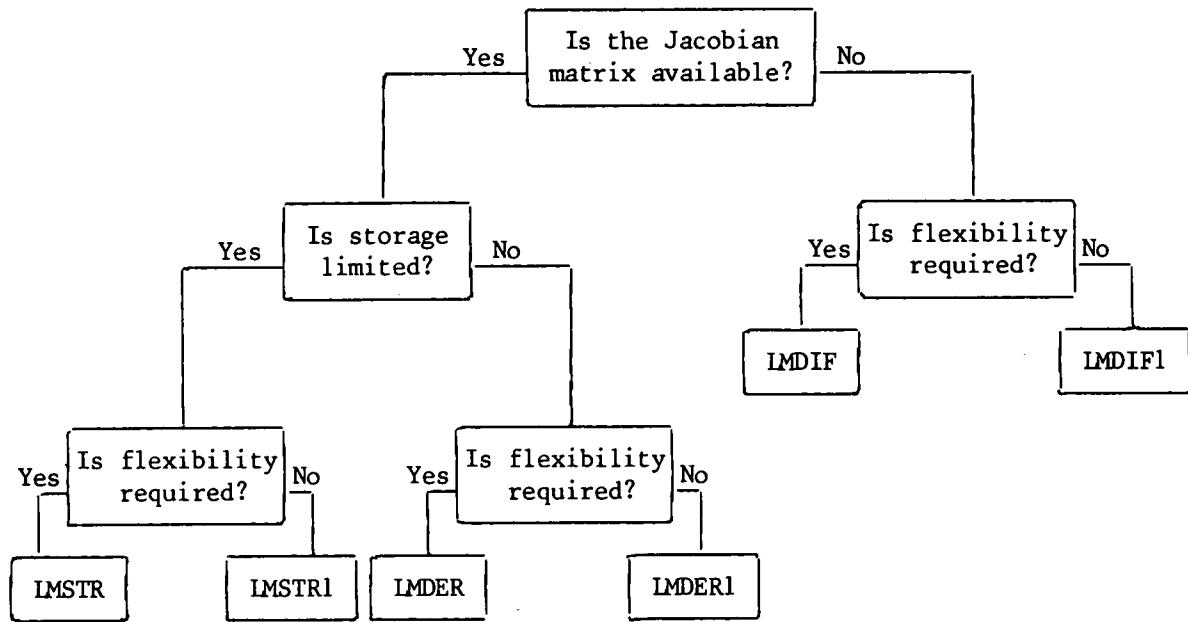


Figure 2
Decision Tree for Nonlinear Least Squares Problems

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:

$$\begin{aligned} &\beta^{1-\lambda}, \text{ the machine precision,} \\ &\beta^{e_{\min}-1}, \text{ the smallest magnitude,} \\ &(1 - \beta^{-\lambda})\beta^{e_{\max}}, \text{ the largest magnitude,} \end{aligned}$$

where λ 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

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

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

$$\begin{pmatrix} A \\ D \end{pmatrix} x \approx \begin{pmatrix} b \\ 0 \end{pmatrix} .$$

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

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

and $\|\mathbf{x}\|$ the Euclidean norm of \mathbf{x} ,

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

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

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

where the component function $f_i: R^n \rightarrow R$ is sometimes called the i -th residual of F . The terminology derives from the fact that a common problem is to fit a model $g(t, \mathbf{x})$ 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 $F: \mathbb{R}^n \rightarrow \mathbb{R}^n$, and a solution vector x^* in \mathbb{R}^n is such that

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

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

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

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 $F: \mathbb{R}^n \rightarrow \mathbb{R}^m$; 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 $f: \mathbb{R}^n \rightarrow \mathbb{R}$, which is the mapping $\nabla f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ 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 $F: R^n \rightarrow R^m$, 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

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

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

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

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

$$(3) \quad \|D(x-x^*)\| \leq \Delta ,$$

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 $\{p: \|Dp\| \leq \Delta\}$. 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

$$(1) \quad \|D(x-x^*)\| \leq XTOL \cdot \|Dx^*\| ,$$

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

$$(2) \quad \|F(x)\| \leq (1 + FTOL) \cdot \|F(x^*)\| ,$$

where FTOL is a second user-supplied tolerance.

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

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

where a_1, a_2, \dots, 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 $XTOL = 10^{-k}$, 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

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

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

$$\Delta \leq \text{XTOL} \cdot \|Dx\|.$$

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 = $(XTOL)^2$ 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

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

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

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

while the (relative) predicted reduction is

$$\text{PRERED} = (\|f\| - \|f+J_p\|)/\|f\| .$$

The F-convergence test then requires that

$$\begin{aligned}\text{PRERED} &\leq \text{FTOL} \\ |\text{ACTRED}| &\leq \text{FTOL} \\ \text{ACTRED} &\leq 2 \cdot \text{PRERED}\end{aligned}$$

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

$$(4) \quad \|f\| \leq AFTOL ,$$

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

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

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

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

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

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

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

$$\min_{D} \{ \| \tilde{J} - J \| : \tilde{J}_p = F(x+p) - F(x) \} ,$$

is given by

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

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

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

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

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

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

$$\text{diag}(\|a_1\|, \|a_2\|, \dots, \|a_n\|) ,$$

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_0 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_0 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

$$\begin{aligned}\tilde{F}(x) &= D_E F(x) , \\ \tilde{x}_0 &= x_0 ,\end{aligned}$$

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 $F: R^n \rightarrow R^m$, 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.

$$\begin{aligned} \tau &= g(x) \\ \text{For } i &= 1, 2, \dots, m \\ f_i(x) &= \tau + h_i(x) . \end{aligned}$$

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

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

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

```

For i = 1,2,...,m
  f_i(x) = 0
  For j = 1,2,...,n
    γ = cos(x_j)
    σ = sin(x_j)
    For i = 1,2,...,m
      f_i(x) = f_i(x) + γα_{ij} + σβ_{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

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

then the rows of the Jacobian matrix are

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

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

As another example, assume that

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

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,...,n
    γ = cos(xj)
    σ = sin(xj)
    For i = 1,2,...,m
         $\frac{\partial f_i(x)}{\partial x_j} = -\sigma \alpha_{ij} + \gamma \beta_{ij} .$ 

```

The previous example illustrates further the possibility of common sub-expressions 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

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

Then the rows of the Jacobian matrix are

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

and the coding of FCN is best done as follows.

```

If FUNCTION EVALUATION then
    τ = g(x)
    Save τ in COMMON
    For i = 1,2,...,m
        fi(x) = τ2 + hi(x)
If JACOBIAN EVALUATION then
    v = ∇g(x)
    For i = 1,2,...,m
        ∇fi(x) = 2τv + ∇hi(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

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

and in FCN define

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

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

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

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

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

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

$$|\rho_i| \leq n^{\frac{1}{2}} \cdot \text{FACTOR} \cdot |\xi_i|,$$

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

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

implies that

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

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

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

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

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

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

$$(3) \quad JP = QR$$

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

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

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 $\epsilon > 0$ and a vector v with $\|v\| = 1$, the problem is to determine a bound σ such that

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

A first order estimate of σ is now

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

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

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

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

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

is used for this purpose.

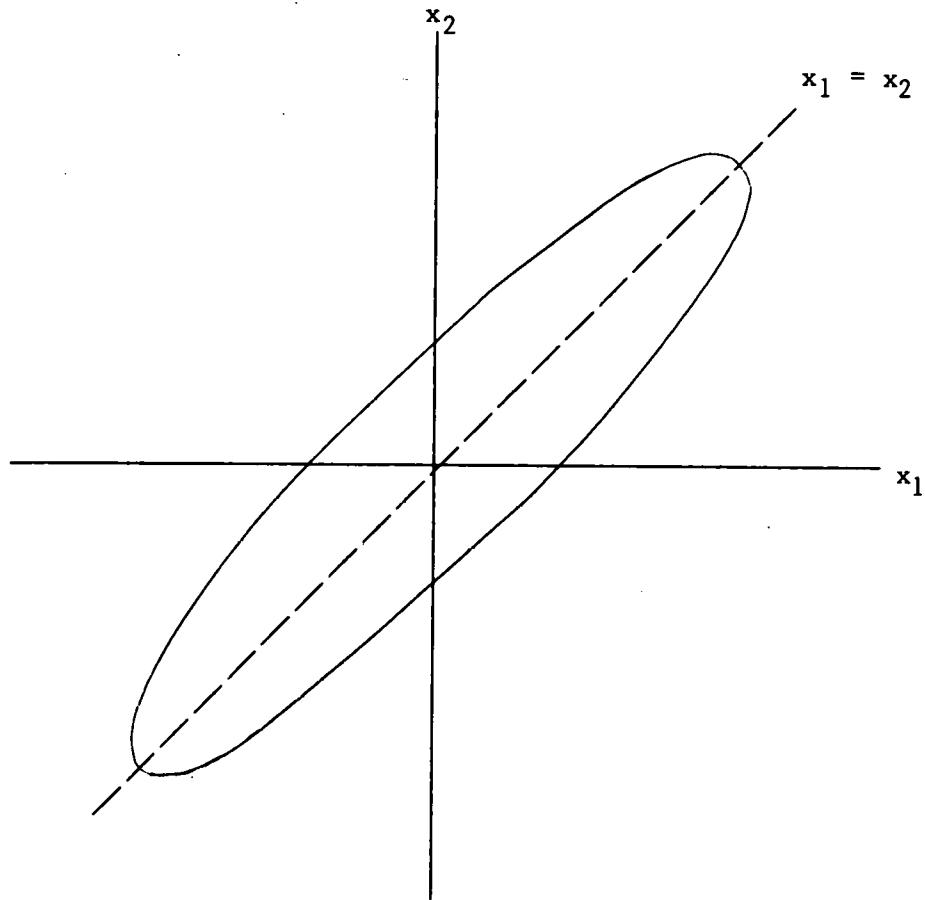


Figure 3

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

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

which is an easy consequence of (3). Subroutine COVAR overwrites R with the upper triangular part of $(R^T R)^{-1}$ 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

$$(5) \quad |r_{kk}| \geq |r_{ij}| , \quad k \leq i \leq j ,$$

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

$$JP_1 = Q_1 R_1$$

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

$$R_1 P_2 = Q_2 R_2$$

where R_2 satisfies (5), and therefore

$$J(P_1 P_2) = (Q_1 Q_2) R_2$$

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,WAL,WA2,WA3)
    DO 40 J = 1, N
        FJAC(J,J) = WAL(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 $P_1 P_2$.

```

SUBROUTINE COVAR(N,R,LDR,IPVT,TOL,WA)          COVR0010
INTEGER N,LDR                                COVR0020
INTEGER IPVT(N)                               COVR0030
DOUBLE PRECISION TOL                          COVR0040
DOUBLE PRECISION R(LDR,N),WA(N)               COVR0050
*****                                           COVR0060
C                                               COVR0070
C                                               COVR0080
C                                               COVR0090
C GIVEN AN M BY N MATRIX A, THE PROBLEM IS TO DETERMINE COVR0100
C THE COVARIANCE MATRIX CORRESPONDING TO A, DEFINED AS COVR0110
C                                               COVR0120
C                                               COVR0130
C                                               COVR0140
C                                               COVR0150
C THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM COVR0160
C IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE COVR0170
C QR FACTORIZATION, WITH COLUMN PIVOTING, OF A. THAT IS, IF COVR0180
C A^P = Q*R, WHERE P IS A PERMUTATION MATRIX, Q HAS ORTHOGONAL COVR0190
C COLUMNS, AND R IS AN UPPER TRIANGULAR MATRIX WITH DIAGONAL COVR0200
C ELEMENTS OF NONINCREASING MAGNITUDE, THEN COVAR EXPECTS COVR0210
C THE FULL UPPER TRIANGLE OF R AND THE PERMUTATION MATRIX P. COVR0220
C THE COVARIANCE MATRIX IS THEN COMPUTED AS COVR0230
C                                               COVR0240
C                                               COVR0250
C                                               COVR0260
C                                               COVR0270
C IF A IS NEARLY RANK DEFICIENT, IT MAY BE DESIRABLE TO COMPUTE COVR0280
C THE COVARIANCE MATRIX CORRESPONDING TO THE LINEARLY INDEPENDENT COVR0290
C COLUMNS OF A. TO DEFINE THE NUMERICAL RANK OF A, COVAR USES COVR0300
C THE TOLERANCE TOL. IF L IS THE LARGEST INTEGER SUCH THAT COVR0310
C COVR0320
C ABS(R(L,L)) .GT. TOL*ABS(R(1,1)) ,          COVR0330
C COVR0340
C THEN COVAR COMPUTES THE COVARIANCE MATRIX CORRESPONDING TO COVR0350
C THE FIRST L COLUMNS OF R. FOR K GREATER THAN L, COLUMN COVR0360
C AND ROW IPVT(K) OF THE COVARIANCE MATRIX ARE SET TO ZERO. COVR0370
C COVR0380
C THE SUBROUTINE STATEMENT IS COVR0390
C COVR0400
C SUBROUTINE COVAR(N,R,LDR,IPVT,TOL,WA)          COVR0410
C COVR0420
C WHERE COVR0430
C COVR0440
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R. COVR0450
C COVR0460
C R IS AN N BY N ARRAY. ON INPUT THE FULL UPPER TRIANGLE MUST COVR0470
C CONTAIN THE FULL UPPER TRIANGLE OF THE MATRIX R. ON OUTPUT COVR0480
C R CONTAINS THE SQUARE SYMMETRIC COVARIANCE MATRIX. COVR0490
C COVR0500
C LDR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N COVR0510
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY R. COVR0520
C COVR0530
C IPVT IS AN INTEGER INPUT ARRAY OF LENGTH N WHICH DEFINES THE COVR0540

```

```

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 *****
C INTEGER I,II,J,JJ,K,KM1,L COVR0710
C LOGICAL SING COVR0720
C DOUBLE PRECISION ONE,TEMP,TOLR,ZERO COVR0730
C 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
C TOLR = TOL*DABS(R(1,1)) COVR0780
C L = 0 COVR0790
C DO 40 K = 1, N COVR0800
C IF (DABS(R(K,K)) .LE. TOLR) GO TO 50 COVR0810
C R(K,K) = ONE/R(K,K) COVR0820
C KM1 = K - 1 COVR0830
C IF (KM1 .LT. 1) GO TO 30 COVR0840
C DO 20 J = 1, KM1 COVR0850
C TEMP = R(K,K)*R(J,K) COVR0860
C R(J,K) = ZERO COVR0870
C DO 10 I = 1, J COVR0880
C 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
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
C 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          COVR1090
80      CONTINUE          COVR1100
     TEMP = R(K,K)        COVR1110
     DO 90 I = 1, K       COVR1120
           R(I,K) = TEMP*R(I,K) COVR1130
90      CONTINUE          COVR1140
100     CONTINUE          COVR1150
110     CONTINUE          COVR1160
C
C      FORM THE FULL LOWER TRIANGLE OF THE COVARIANCE MATRIX    COVR1170
C      IN THE STRICT LOWER TRIANGLE OF R AND IN WA.             COVR1180
C
C      DO 130 J = 1, N          COVR1190
     JJ = IPVT(J)          COVR1200
     SING = J .GT. L        COVR1210
     DO 120 I = 1, J        COVR1220
           IF (SING) R(I,J) = ZERO COVR1230
           II = IPVT(I)          COVR1240
           IF (II .GT. JJ) R(II,JJ) = R(I,J) COVR1250
           IF (II .LT. JJ) R(JJ,II) = R(I,J) COVR1260
120     CONTINUE          COVR1270
     WA(JJ) = R(J,J)        COVR1280
130     CONTINUE          COVR1290
C
C      SYMMETRIZE THE COVARIANCE MATRIX IN R.                  COVR1300
C
C      DO 150 J = 1, N          COVR1310
         DO 140 I = 1, J        COVR1320
           R(I,J) = R(J,I)        COVR1330
140     CONTINUE          COVR1340
         R(J,J) = WA(J)        COVR1350
150     CONTINUE          COVR1360
         RETURN               COVR1370
C
C      LAST CARD OF SUBROUTINE COVAR.                         COVR1380
C
C      END               COVR1390

```

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

Systems of nonlinear equations

HYBRD1, HYBRD, HYBRJ1, HYBRJ

Nonlinear least squares problems

LMDIF1, LMDIF, LMDER1, LMDER, LMSTR1, LMSTR

Derivative checking

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

$$\text{ENORM}(X-XSOL) \leq TOL * \text{ENORM}(XSOL).$$

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 $N \leq 0$, or $TOL < 0.00$, or $LWA < (N*(3*N+13))/2$.

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, QRFAAC, R1MPYQ, R1UPDT

FORTRAN-supplied ... DABS, DMAX1, DMIN1, DSQRT, MINO, 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 $x(1), x(2), \dots, x(9)$, which solve the system of tridiagonal equations

$$\begin{array}{lcl} (3-2*x(1))*x(1) & -2*x(2) & = -1 \\ -x(i-1) + (3-2*x(i))*x(i) & -2*x(i+1) & = -1, \quad i=2-8 \\ & -x(8) + (3-2*x(9))*x(9) & = -1 \end{array}$$

```

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

```
C SUBROUTINE FCN FOR HYBRD1 EXAMPLE.  
C  
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,QT,F,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),QT(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 $(Q \text{ transpose}) * FVEC$.

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

$$\text{ENORM}(D^*(X-XSOL)) \leq XTOL * \text{ENORM}(D^*XSOL).$$

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 $N \leq 0$, or $XTOL < 0.0$, or $\text{MAXFEV} \leq 0$, or $ML < 0$, or $MU < 0$, or $\text{FACTOR} \leq 0.0$, or $LDFJAC < N$, or $LR < (N*(N+1))/2$.

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

$$\begin{aligned} (3-2*x(1))*x(1) & -2*x(2) & = -1 \\ -x(i-1) + (3-2*x(i))*x(i) & -2*x(i+1) & = -1, \quad i=2-8 \\ -x(8) + (3-2*x(9))*x(9) & = -1 \end{aligned}$$

```

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

$$\text{ENORM}(X-XSOL) \leq TOL * \text{ENORM}(XSOL).$$

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 $N \leq 0$, or $LDFJAC < N$, or $TOL < 0.00$, or $LWA < (N*(N+13))/2$.

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

$$\begin{aligned} (3-2*x(1))*x(1) & -2*x(2) & = -1 \\ -x(i-1) + (3-2*x(i))*x(i) & -2*x(i+1) = -1, \quad i=2-8 \\ -x(8) + (3-2*x(9))*x(9) & = -1 \end{aligned}$$

```

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

```

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

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

$$\text{ENORM}(D^*(X-XSOL)) \leq XTOL * \text{ENORM}(D^*XSOL).$$

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 $N \leq 0$, or $LDFJAC < N$, or $XTOL < 0.0$, or $MAXFEV \leq 0$, or $FACTOR \leq 0.0$, or $LR < (N*(N+1))/2$.

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, QRFAAC, R1MPYQ, R1UPDT

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

$$\begin{aligned} (3-2*x(1))*x(1) & -2*x(2) & = -1 \\ -x(i-1) + (3-2*x(i))*x(i) & -2*x(i+1) & = -1, \quad i=2-8 \\ -x(8) + (3-2*x(9))*x(9) & = -1 \end{aligned}$$

```

C ****
C
C DRIVER FOR HYBRJ EXAMPLE.
C DOUBLE PRECISION VERSION
C
C ****
C INTEGER J,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV,LR,NWRITE
C DOUBLE PRECISION XTOL,FACTOR,FNORM
C DOUBLE PRECISION X(9),FVEC(9),FJAC(9,9),DIAG(9),R(45),QTF(9),
C * WA1(9),WA2(9),WA3(9),WA4(9)
C DOUBLE PRECISION ENORM,DPMPPAR
C EXTERNAL FCN
C
C LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
C DATA NWRITE /6/
C
C N = 9
C
C THE FOLLOWING STARTING VALUES PROVIDE A ROUGH SOLUTION.
C
C DO 10 J = 1, 9
C     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(DPMPPAR(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
C      END
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      SUBROUTINE FCN FOR HYBRJ EXAMPLE.

C      INTEGER J,K
C      DOUBLE PRECISION ONE,TEMP,TEMP1,TEMP2,THREE,TWO,ZERO
C      DATA ZERO,ONE,TWO,THREE,FOUR /0.D0,1.D0,2.D0,3.D0,4.D0/
C
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
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

$$\text{ENORM}(FVEC) \leq (1+TOL)*\text{ENORM}(FVECS),$$

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

$$\text{ENORM}(D^*(X-XSOL)) \leq \text{TOL} * \text{ENORM}(D^*XSOL).$$

If this condition is satisfied with $\text{TOL} = 10^{**(-K)}$, 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 $N \leq 0$, or $M < N$, or $LDFJAC < M$, or $\text{TOL} < 0.00$, or $LWA < 5*N+M$.

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

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

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 DRIVER FOR LMDER1 EXAMPLE.
C DOUBLE PRECISION VERSION
C *****
C INTEGER J,M,N,LDFJAC,INFO,LWA,NWRITE
C INTEGER IPVT(3)
C DOUBLE PRECISION TOL,FNORM
C DOUBLE PRECISION X(3),FVEC(15),FJAC(15,3),WA(30)
C DOUBLE PRECISION ENORM,DPMPAR
C EXTERNAL FCN
C
C LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
C DATA NWRITE /6/
C
C M = 15
C N = 3
C
C THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
C X(1) = 1.DO
C X(2) = 1.DO
C X(3) = 1.DO
C
C LDFJAC = 15
C 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
C TOL = DSQRT(DPMPAR(1))
C
C CALL LMDER1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,
C *           INFO,IPVT,WA,LWA)
C FNORM = ENORM(M,FVEC)
C WRITE (NWRITE,1000) FNORM,INFO,(X(J),J=1,N)
C STOP
1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
C *           5X,15H EXIT PARAMETER,16X,I10 //
C *           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 $(Q \text{ transpose})^*FVEC$.

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

$$\text{ENORM}(FVEC) \leq (1+FTOL) * \text{ENORM}(FVECS),$$

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

$$\text{ENORM}(D^*(X-XSOL)) \leq XTOL * \text{ENORM}(D^*XSOL).$$

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, QRSLV

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

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

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 ****
C INTEGER J,M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV,NWRITE
C INTEGER IPVT(3)
C DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR,FNORM
C DOUBLE PRECISION X(3),FVEC(15),FJAC(15,3),DIAG(3),QTF(3),
C *           WA1(3),WA2(3),WA3(3),WA4(15)
C DOUBLE PRECISION ENORM,DPMPAR
C EXTERNAL FCN
C
C LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
C DATA NWRITE /6/
C
C M = 15
C N = 3
C
C THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
C X(1) = 1.D0
C X(2) = 1.D0
C X(3) = 1.D0
C
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
C FTOL = DSQRT(DPMPAR(1))
C XTOL = DSQRT(DPMPAR(1))
C GTOL = 0.D0
C
C MAXFEV = 400
C MODE = 1
C FACTOR = 1.D2
C NPRINT = 0
C
C CALL LMDER(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
C *           MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,
C *           IPVT,QTF,WA1,WA2,WA3,WA4)
C FNORM = ENORM(M,FVEC)
C WRITE (NWRITE,1000) FNORM,NFEV,NJEV,INFO,(X(J),J=1,N)
C STOP
C 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
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
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
C      IF (IFLAG .NE. 0) GO TO 5
C
C      INSERT PRINT STATEMENTS HERE WHEN NPRINT IS POSITIVE.
C
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
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 DMPMPAR(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

$$\text{ENORM}(FVEC) \leq (1+TOL)*\text{ENORM}(FVECS),$$

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

$$\text{ENORM}(D^*(X-XSOL)) \leq \text{TOL} * \text{ENORM}(D^*XSOL).$$

If this condition is satisfied with $\text{TOL} = 10^{**(-K)}$, 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 $N \leq 0$, or $M < N$, or $LDFJAC < N$, or $\text{TOL} < 0.00$, or $LWA < 5*N+M$.

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 ... DMPMPAR, 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

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

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))).
```

DRIVER FOR LMSTR1 EXAMPLE.
DOUBLE PRECISION VERSION

```
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,DMPMPAR
EXTERNAL FCN
```

LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.

DATA NWRITE /6/

M = 15
N = 3

THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.

X(1) = 1.D0
X(2) = 1.D0
X(3) = 1.D0

LDFJAC = 3
LWA = 30

SET TOL TO THE SQUARE ROOT OF THE MACHINE PRECISION.
UNLESS HIGH PRECISION SOLUTIONS ARE REQUIRED,
THIS IS THE RECOMMENDED SETTING.

TOL = DSQRT(DMPMPAR(1))

```
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 SUBROUTINE FCN FOR LMSTR1 EXAMPLE.

C
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
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 LAST CARD OF SUBROUTINE FCN.

C
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

$$\begin{matrix} T & T & T \\ P^T * (JAC^T * JAC) * P^T = R^T * R \end{matrix}$$

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 $(Q \text{ transpose})^*FVEC$.

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

$$\text{ENORM}(FVEC) \leq (1+FTOL)*\text{ENORM}(FVECS),$$

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

$$\text{ENORM}(D^*(X-XSOL)) \leq XTOL * \text{ENORM}(D^*XSOL).$$

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 $N \leq 0$, or $M < N$, or $LDFJAC < N$, or $FTOL < 0.00$, or $XTOL < 0.00$, or $GTOL < 0.00$, or $MAXFEV \leq 0$, or $FACTOR \leq 0.00$.

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

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

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 DRIVER FOR LMSTR EXAMPLE.
C DOUBLE PRECISION VERSION
C *****
C INTEGER J,M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV,NWRITE
C INTEGER IPV(3)
C DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR,FNORM
C DOUBLE PRECISION X(3),FVEC(15),FJAC(3,3),DIAG(3),QTF(3),
C *           WA1(3),WA2(3),WA3(3),WA4(15)
C DOUBLE PRECISION ENORM,DPMPAR
C EXTERNAL FCN
C
C LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
C DATA NWRITE /6/
C
C M = 15
C N = 3
C
C THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
C X(1) = 1.D0
C X(2) = 1.D0
C X(3) = 1.D0
C
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
C FTOL = DSQRT(DPMPAR(1))
C XTOL = DSQRT(DPMPAR(1))
C GTOL = 0.D0
C
C MAXFEV = 400
C MODE = 1
C FACTOR = 1.D2
C NPRINT = 0
C
C CALL LMSTR(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,
C *           MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,
C *           IPV,QTF,WA1,WA2,WA3,WA4)
C FNORM = ENORM(M,FVEC)
C WRITE (NWRITE,1000) FNORM,NFEV,NJEV,INFO,(X(J),J=1,N)
C 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-1,1.34DO,2.1DO,4.39DO/
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

$$\text{ENORM}(\text{FVEC}) \leq (1+\text{TOL}) * \text{ENORM}(\text{FVECS}),$$

where FVECS denotes the functions evaluated at XSOL. If this condition is satisfied with $\text{TOL} = 10^{**(-K)}$, 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

$$\text{ENORM}(D^*(X-XSOL)) \leq \text{TOL} * \text{ENORM}(D^*XSOL).$$

If this condition is satisfied with $\text{TOL} = 10^{**(-K)}$, 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

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

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 DRIVER FOR LMDIF1 EXAMPLE.
C DOUBLE PRECISION VERSION
C

```

C ****
C INTEGER J,M,N,INFO,LWA,NWRITE
C INTEGER IWA(3)
C DOUBLE PRECISION TOL,FNORM
C DOUBLE PRECISION X(3),FVEC(15),WA(75)
C DOUBLE PRECISION ENORM,DMPMPAR
C EXTERNAL FCN
C
C LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
C DATA NWRITE /6/
C
C M = 15
C N = 3
C
C THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
C X(1) = 1.DO
C X(2) = 1.DO
C X(3) = 1.DO
C
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
C TOL = DSQRT(DMPMPAR(1))
C
C CALL LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)
C FNORM = ENORM(M,FVEC)
C WRITE (NWRITE,1000) FNORM,INFO,(X(J),J=1,N)
C 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
C END
C SUBROUTINE FCN(M,N,X,FVEC,IFLAG)
C INTEGER M,N,IFLAG
C DOUBLE PRECISION X(N),FVEC(M)
C
C SUBROUTINE FCN FOR LMDIF1 EXAMPLE.
C
C INTEGER I
C DOUBLE PRECISION TMP1,TMP2,TMP3
C DOUBLE PRECISION Y(15)
C 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-1,1.34DO,2.1DO,4.39DO/
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 THIS PAGE ALREADY
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

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

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 $JAC^T * P^T = Q^T * R$, 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 $(Q^T)^* FVEC$.

$WA1$, $WA2$, 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

$$ENORM(FVEC) \leq (1+FTOL)*ENORM(FVECS),$$

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

$$\text{ENORM}(D^*(X-XSOL)) \leq XTOL * \text{ENORM}(D^*XSOL).$$

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

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

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 *****
C INTEGER J,M,N,MAXFEV,MODE,NPRINT,INFO,NFEV,LDFJAC,NWRITE
C INTEGER IPVT(3)
C DOUBLE PRECISION FTOL,XTOL,GTOL,EPSFCN,FACTOR,FNORM
C DOUBLE PRECISION X(3),FVEC(15),DIAG(3),FJAC(15,3),QTF(3),
C *           WA1(3),WA2(3),WA3(3),WA4(15)
C DOUBLE PRECISION ENORM,DPMPAR
C EXTERNAL FCN
C
C LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
C DATA NWRITE /6/
C
C M = 15
C N = 3
C
C THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
C X(1) = 1.D0
C X(2) = 1.D0
C X(3) = 1.D0
C
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
C FTOL = DSQRT(DPMPAR(1))
C XTOL = DSQRT(DPMPAR(1))
C GTOL = 0.D0
C
C MAXFEV = 800
C EPSFCN = 0.D0
C MODE = 1
C FACTOR = 1.D2
C NPRINT = 0
C
C CALL LMDIF(FCN,M,N,X,FVEC,FTOL,XTOL,GTOL,MAXFEV,EPSFCN,
C *           DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,
C *           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

RECORDED
THIS PAGE
WAS THIS PAGE ALREADY
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

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

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 ****
C INTEGER I,M,N,LDFJAC,MODE,NWRITE
C DOUBLE PRECISION X(3),FVEC(15),FJAC(15,3),XP(3),FVECP(15),
C * ERR(15)
C
C LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
C DATA NWRITE /6/
C
C M = 15
C N = 3
C
C THE FOLLOWING VALUES SHOULD BE SUITABLE FOR
C CHECKING THE JACOBIAN MATRIX.
C
C X(1) = 9.2D-1
C X(2) = 1.3D-1
C X(3) = 5.4D-1
C
C LDFJAC = 15
C
C MODE = 1
C CALL CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
C MODE = 2
C CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,1)
C CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,2)
C CALL FCN(M,N,XP,FVECP,FJAC,LDFJAC,1)
C CALL CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
C
C DO 10 I = 1, M
C     FVECP(I) = FVECP(I) - FVEC(I)
10    CONTINUE
C     WRITE (NWRITE,1000) (FVEC(I),I=1,M)
C     WRITE (NWRITE,2000) (FVECP(I),I=1,M)
C     WRITE (NWRITE,3000) (ERR(I),I=1,M)
C     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
C END
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 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,
*      3.7D-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, DOGLE, ENORM, FDJAC1, FDJAC2, HYBRD, HYBRD1,
HYBRJ, HYBRJ1, LMDER, LMDER1, LMDIF, LMDIF1, LMPAR, LMSTR,
LMSTR1, QFORM, QRFAC, QRSLV, 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)      CHDR0010
  INTEGER M,N,LDFJAC,MODE                                         CHDR0020
  DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),XP(N),FVECP(M),    CHDR0030
*          ERR(M)                                                 CHDR0040
C *****                                                       CHDR0050
C
C SUBROUTINE CHKDER                                         CHDR0060
C
C THIS SUBROUTINE CHECKS THE GRADIENTS OF M NONLINEAR FUNCTIONS   CHDR0070
C IN N VARIABLES, EVALUATED AT A POINT X, FOR CONSISTENCY WITH     CHDR0080
C THE FUNCTIONS THEMSELVES. THE USER MUST CALL CHKDER TWICE,        CHDR0090
C FIRST WITH MODE = 1 AND THEN WITH MODE = 2.                         CHDR0100
C
C MODE = 1. ON INPUT, X MUST CONTAIN THE POINT OF EVALUATION.      CHDR0110
C             ON OUTPUT, XP IS SET TO A NEIGHBORING POINT.            CHDR0120
C
C MODE = 2. ON INPUT, FVEC MUST CONTAIN THE FUNCTIONS AND THE       CHDR0130
C             ROWS OF FJAC MUST CONTAIN THE GRADIENTS                 CHDR0140
C             OF THE RESPECTIVE FUNCTIONS EACH EVALUATED               CHDR0150
C             AT X, AND FVECP MUST CONTAIN THE FUNCTIONS              CHDR0160
C             EVALUATED AT XP.                                         CHDR0170
C             ON OUTPUT, ERR CONTAINS MEASURES OF CORRECTNESS OF     CHDR0180
C             THE RESPECTIVE GRADIENTS.                                CHDR0190
C
C THE SUBROUTINE DOES NOT PERFORM RELIABLY IF CANCELLATION OR      CHDR0200
C ROUNDING ERRORS CAUSE A SEVERE LOSS OF SIGNIFICANCE IN THE       CHDR0210
C EVALUATION OF A FUNCTION. THEREFORE, NONE OF THE COMPONENTS       CHDR0220
C OF X SHOULD BE UNUSUALLY SMALL (IN PARTICULAR, ZERO) OR ANY      CHDR0230
C OTHER VALUE WHICH MAY CAUSE LOSS OF SIGNIFICANCE.                CHDR0240
C
C THE SUBROUTINE STATEMENT IS                                     CHDR0250
C
C SUBROUTINE CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)      CHDR0260
C
C WHERE
C
C   M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER      CHDR0270
C       OF FUNCTIONS.                                              CHDR0280
C
C   N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER      CHDR0290
C       OF VARIABLES.                                             CHDR0300
C
C   X IS AN INPUT ARRAY OF LENGTH N.                               CHDR0310
C
C   FVEC IS AN ARRAY OF LENGTH M. ON INPUT WHEN MODE = 2,           CHDR0320
C       FVEC MUST CONTAIN THE FUNCTIONS EVALUATED AT X.             CHDR0330
C
C   FJAC IS AN M BY N ARRAY. ON INPUT WHEN MODE = 2,              CHDR0340
C       THE ROWS OF FJAC MUST CONTAIN THE GRADIENTS OF             CHDR0350
C       THE RESPECTIVE FUNCTIONS EVALUATED AT X.                  CHDR0360
C
C   LDFJAC IS A POSITIVE INTEGER INPUT PARAMETER NOT LESS THAN M   CHDR0370
C       WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.  CHDR0380
C
C

```

```

C   XP IS AN ARRAY OF LENGTH N. ON OUTPUT WHEN MODE = 1,          CHDR0550-
C   XP IS SET TO A NEIGHBORING POINT OF X.                         CHDR0560
C                                         CHDR0570
C
C   FVECP IS AN ARRAY OF LENGTH M. ON INPUT WHEN MODE = 2,          CHDR0580
C   FVECP MUST CONTAIN THE FUNCTIONS EVALUATED AT XP.             CHDR0590
C                                         CHDR0600
C
C   MODE IS AN INTEGER INPUT VARIABLE SET TO 1 ON THE FIRST CALL    CHDR0610
C   AND 2 ON THE SECOND. OTHER VALUES OF MODE ARE EQUIVALENT      CHDR0620
C   TO MODE = 1.                                              CHDR0630
C                                         CHDR0640
C
C   ERR IS AN ARRAY OF LENGTH M. ON OUTPUT WHEN MODE = 2,           CHDR0650
C   ERR CONTAINS MEASURES OF CORRECTNESS OF THE RESPECTIVE        CHDR0660
C   GRADIENTS. IF THERE IS NO SEVERE LOSS OF SIGNIFICANCE,        CHDR0670
C   THEN IF ERR(I) IS 1.0 THE I-TH GRADIENT IS CORRECT,            CHDR0680
C   WHILE IF ERR(I) IS 0.0 THE I-TH GRADIENT IS INCORRECT.         CHDR0690
C   FOR VALUES OF ERR BETWEEN 0.0 AND 1.0, THE CATEGORIZATION     CHDR0700
C   IS LESS CERTAIN. IN GENERAL, A VALUE OF ERR(I) GREATER        CHDR0710
C   THAN 0.5 INDICATES THAT THE I-TH GRADIENT IS PROBABLY       CHDR0720
C   CORRECT, WHILE A VALUE OF ERR(I) LESS THAN 0.5 INDICATES      CHDR0730
C   THAT THE I-TH GRADIENT IS PROBABLY INCORRECT.                 CHDR0740
C                                         CHDR0750
C
C   SUBPROGRAMS CALLED                                           CHDR0760
C
C   MINPACK SUPPLIED ... DPMPAR                                CHDR0770
C
C   FORTRAN SUPPLIED ... DABS,DLOG10,DSQRT                      CHDR0780
C                                         CHDR0790
C
C   ARGONNE-NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.      CHDR0800
C   BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE        CHDR0810
C
C   *****
C   INTEGER I,J                                              CHDR0820
C   DOUBLE PRECISION EPS,EPSF,EPSLOG,EPSMCH,FACTOR,ONE,TEMP,ZERO  CHDR0830
C   DOUBLE PRECISION DPMPAR                                     CHDR0840
C   DATA FACTOR,ONE,ZERO /1.0D2,1.0D0,0.0D0/                  CHDR0850
C                                         CHDR0860
C   EPSMCH IS THE MACHINE PRECISION.                           CHDR0870
C                                         CHDR0880
C   EPSMCH = DPMPAR(1)                                       CHDR0890
C                                         CHDR0900
C   EPS = DSQRT(EPSMCH)                                      CHDR0910
C                                         CHDR0920
C   IF (MODE .EQ. 2) GO TO 20                                 CHDR0930
C                                         CHDR0940
C   MODE = 1.                                                 CHDR0950
C                                         CHDR0960
C   DO 10 J = 1, N                                         CHDR0970
C   TEMP = EPS*DABS(X(J))                                    CHDR0980
C   IF (TEMP .EQ. ZERO) TEMP = EPS                         CHDR0990
C   XP(J) = X(J) + TEMP                                     CHDR1000
C   10    CONTINUE                                         CHDR1010
C   GO TO 70                                              CHDR1020
C   20    CONTINUE                                         CHDR1030
C
C                                         CHDR1040
C                                         CHDR1050
C                                         CHDR1060
C                                         CHDR1070
C                                         CHDR1080

```

```

C      MODE = 2.                      CHDR1090
C
C      EPSF = FACTOR*EPSMCH          CHDR1100
C      EPSLOG = DLOG10(EPS)          CHDR1110
C      DO 30 I = 1, M               CHDR1120
C          ERR(I) = ZERO            CHDR1130
C      30      CONTINUE              CHDR1140
C          DO 50 J = 1, N             CHDR1150
C              TEMP = DABS(X(J))      CHDR1160
C              IF (TEMP .EQ. ZERO) TEMP = ONE    CHDR1170
C              DO 40 I = 1, M           CHDR1180
C                  ERR(I) = ERR(I) + TEMP*FJAC(I,J) CHDR1190
C      40      CONTINUE              CHDR1200
C      50      CONTINUE              CHDR1210
C          DO 60 I = 1, M           CHDR1220
C              TEMP = ONE            CHDR1230
C              IF (FVEC(I) .NE. ZERO .AND. FVECP(I) .NE. ZERO
C                   .AND. DABS(FVECP(I)-FVEC(I)) .GE. EPSF*DABS(FVEC(I))) CHDR1240
C      *          TEMP = EPS*DABS((FVECP(I)-FVEC(I))/EPS-ERR(I))     CHDR1250
C      *          / (DABS(FVEC(I)) + DABS(FVECP(I)))                 CHDR1260
C          ERR(I) = ONE              CHDR1270
C          IF (TEMP .GT. EPSMCH .AND. TEMP .LT. EPS)                 CHDR1280
C      *          ERR(I) = (DLOG10(TEMP) - EPSLOG)/EPSLOG                CHDR1290
C          IF (TEMP .GE. EPS) ERR(I) = ZERO                         CHDR1300
C      60      CONTINUE              CHDR1310
C      70      CONTINUE              CHDR1320
C
C      RETURN                      CHDR1330
C
C      LAST CARD OF SUBROUTINE CHKDER.          CHDR1340
C
C      END                          CHDR1350

```

```

SUBROUTINE DOGLEG(N,R,LR,DIAG,QTB,DELTA,X,WA1,WA2) DOGL0010
  INTEGER N,LR DOGL0020
  DOUBLE PRECISION DELTA DOGL0030
  DOUBLE PRECISION R(LR),DIAG(N),QTB(N),X(N),WA1(N),WA2(N) DOGL0040
***** DOGL0050
C DOGL0060
C SUBROUTINE DOGLEG DOGL0070
C
C GIVEN AN M BY N MATRIX A, AN N BY N NONSINGULAR DIAGONAL DOGL0080
C MATRIX D, AN M-VECTOR B, AND A POSITIVE NUMBER DELTA, THE DOGL0090
C PROBLEM IS TO DETERMINE THE CONVEX COMBINATION X OF THE DOGL0100
C GAUSS-NEWTON AND SCALED GRADIENT DIRECTIONS THAT MINIMIZES DOGL0110
C (A*X - B) IN THE LEAST SQUARES SENSE, SUBJECT TO THE DOGL0120
C RESTRICTION THAT THE EUCLIDEAN NORM OF D*X BE AT MOST DELTA. DOGL0130
C
C THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM DOGL0140
C IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE DOGL0150
C QR FACTORIZATION OF A. THAT IS, IF A = Q*R, WHERE Q HAS DOGL0160
C ORTHOGONAL COLUMNS AND R IS AN UPPER TRIANGULAR MATRIX, DOGL0170
C THEN DOGLEG EXPECTS THE FULL UPPER TRIANGLE OF R AND DOGL0180
C THE FIRST N COMPONENTS OF (Q TRANSPOSE)*B. DOGL0190
C
C THE SUBROUTINE STATEMENT IS DOGL0200
C
C SUBROUTINE DOGLEG(N,R,LR,DIAG,QTB,DELTA,X,WA1,WA2) DOGL0210
C
C WHERE DOGL0220
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R. DOGL0230
C
C R IS AN INPUT ARRAY OF LENGTH LR WHICH MUST CONTAIN THE UPPER DOGL0240
C TRIANGULAR MATRIX R STORED BY ROWS. DOGL0250
C
C LR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN DOGL0260
C (N*(N+1))/2. DOGL0270
C
C DIAG IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE DOGL0280
C DIAGONAL ELEMENTS OF THE MATRIX D. DOGL0290
C
C QTB IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE FIRST DOGL0300
C N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*B. DOGL0310
C
C DELTA IS A POSITIVE INPUT VARIABLE WHICH SPECIFIES AN UPPER DOGL0320
C BOUND ON THE EUCLIDEAN NORM OF D*X. DOGL0330
C
C X IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE DESIRED DOGL0340
C CONVEX COMBINATION OF THE GAUSS-NEWTON DIRECTION AND THE DOGL0350
C SCALED GRADIENT DIRECTION. DOGL0360
C
C WA1 AND WA2 ARE WORK ARRAYS OF LENGTH N. DOGL0370
C
C SUBPROGRAMS CALLED DOGL0380
C
C MINPACK-SUPPLIED ... DPMPAR,ENORM DOGL0390

```

```

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
C INTEGER I,J,JJ,JP1,K,L DOGL0620
C DOUBLE PRECISION ALPHA,BNORM,EPSMCH,GNORM,ONE,QNORM,SGNORM,SUM, DOGL0630
* TEMP,ZERO DOGL0640
C DOUBLE PRECISION DPMPAR,ENORM DOGL0650
C DATA ONE,ZERO /1.0DO,0.0DO/ DOGL0660
C DOGL0670
C EPSMCH IS THE MACHINE PRECISION. DOGL0680
C DOGL0690
C EPSMCH = DPMPAR(1) DOGL0700
C DOGL0710
C FIRST, CALCULATE THE GAUSS-NEWTON DIRECTION. DOGL0720
C DOGL0730
C JJ = (N*(N + 1))/2 + 1 DOGL0740
C DO 50 K = 1, N DOGL0750
C J = N - K + 1 DOGL0760
C JP1 = J + 1 DOGL0770
C JJ = JJ - K DOGL0780
C L = JJ + 1 DOGL0790
C SUM = ZERO DOGL0800
C IF (N .LT. JP1) GO TO 20 DOGL0810
C DO 10 I = JP1, N DOGL0820
C SUM = SUM + R(L)*X(I) DOGL0830
C 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
C DO 60 J = 1, N DOGL1020
C WA1(J) = ZERO DOGL1030
C 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
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 CALCULATE THE NORM OF THE SCALED GRADIENT AND TEST FOR
C THE SPECIAL CASE IN WHICH THE SCALED GRADIENT IS ZERO.
C
C GNORM = ENORM(N,WA1)
SGNORM = ZERO
ALPHA = DELTA/QNORM
IF (GNORM .EQ. ZERO) GO TO 120

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 TEST WHETHER THE SCALED GRADIENT DIRECTION IS ACCEPTABLE.
C
C ALPHA = ZERO
IF (SGNORM .GE. DELTA) GO TO 120

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 DOGL1630
C DOGL1640
C FORM APPROPRIATE CONVEX COMBINATION OF THE GAUSS-NEWTON DOGL1650
C DIRECTION AND THE SCALED GRADIENT DIRECTION. DOGL1660
C DOGL1670
TEMP = (ONE - ALPHA)*DMIN1(SGNORM,DELTA) DOGL1680
DO 130 J = 1, N DOGL1690
X(J) = TEMP*WA1(J) + ALPHA*X(J) DOGL1700
130 CONTINUE DOGL1710
140 CONTINUE DOGL1720
RETURN DOGL1730
C DOGL1740
C LAST CARD OF SUBROUTINE DOGLE. DOGL1750
C DOGL1760
END DOGL1770
```

```

DOUBLE PRECISION FUNCTION ENORM(N,X) ENRM0010
INTEGER N ENRM0020
DOUBLE PRECISION X(N) ENRM0030
***** ENRM0040
C ENRM0050
C FUNCTION ENORM ENRM0060
C GIVEN AN N-VECTOR X, THIS FUNCTION CALCULATES THE ENRM0070
C EUCLIDEAN NORM OF X. ENRM0080
C THE EUCLIDEAN NORM IS COMPUTED BY ACCUMULATING THE SUM OF ENRM0090
C SQUARES IN THREE DIFFERENT SUMS. THE SUMS OF SQUARES FOR THE ENRM0100
C SMALL AND LARGE COMPONENTS ARE SCALED SO THAT NO OVERFLOWS ENRM0110
C OCCUR. NON-DESTRUCTIVE UNDERFLOWS ARE PERMITTED. UNDERFLOWS ENRM0120
C AND OVERFLOWS DO NOT OCCUR IN THE COMPUTATION OF THE UNSCALED ENRM0130
C SUM OF SQUARES FOR THE INTERMEDIATE COMPONENTS. ENRM0140
C THE DEFINITIONS OF SMALL, INTERMEDIATE AND LARGE COMPONENTS ENRM0150
C DEPEND ON TWO CONSTANTS, RDWARP AND RGIAINT. THE MAIN ENRM0160
C RESTRICTIONS ON THESE CONSTANTS ARE THAT RDWARP**2 NOT ENRM0170
C UNDERFLOW AND RGIAINT**2 NOT OVERFLOW. THE CONSTANTS ENRM0180
C GIVEN HERE ARE SUITABLE FOR EVERY KNOWN COMPUTER. ENRM0190
C THE FUNCTION STATEMENT IS ENRM0200
C DOUBLE PRECISION FUNCTION ENORM(N,X) ENRM0210
C WHERE ENRM0220
C N IS A POSITIVE INTEGER INPUT VARIABLE. ENRM0230
C X IS AN INPUT ARRAY OF LENGTH N. ENRM0240
C SUBPROGRAMS CALLED ENRM0250
C FORTRAN-SUPPLIED ... DABS,DSQRT ENRM0260
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. ENRM0270
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE ENRM0280
C *****
C INTEGER I ENRM0290
C DOUBLE PRECISION AGIANT,FLOATN,ONE,RDWARP,RGIAINT,S1,S2,S3,XABS, ENRM0300
C * X1MAX,X3MAX,ZERO ENRM0310
C DATA ONE,ZERO,RDWARP,RGIAINT /1.0D0,0.0D0,3.834D-20,1.304D19/ ENRM0320
C S1 = ZERO ENRM0330
C S2 = ZERO ENRM0340
C S3 = ZERO ENRM0350
C X1MAX = ZERO ENRM0360
C X3MAX = ZERO ENRM0370
C FLOATN = N ENRM0380
C AGIANT = RGIAINT/FLOATN ENRM0390
C DO 90 I = 1, N ENRM0400
C     XABS = DABS(X(I)) ENRM0410
C     IF (XABS .GT. RDWARP .AND. XABS .LT. AGIANT) GO TO 70 ENRM0420
C

```

```

IF (XABS .LE. RDWARP) GO TO 30 ENRM0550
C
C           SUM FOR LARGE COMPONENTS. ENRM0560
C
C           IF (XABS .LE. X1MAX) GO TO 10 ENRM0570
C               S1 = ONE + S1*(X1MAX/XABS)**2 ENRM0580
C               X1MAX = XABS
C               GO TO 20
10          CONTINUE ENRM0590
C               S1 = S1 + (XABS/X1MAX)**2 ENRM0600
20          CONTINUE ENRM0610
C               GO TO 60 ENRM0620
30          CONTINUE ENRM0630
C
C           SUM FOR SMALL COMPONENTS. ENRM0640
C
C           IF (XABS .LE. X3MAX) GO TO 40 ENRM0650
C               S3 = ONE + S3*(X3MAX/XABS)**2 ENRM0660
C               X3MAX = XABS
C               GO TO 50 ENRM0670
40          CONTINUE ENRM0680
C               IF (XABS .NE. ZERO) S3 = S3 + (XABS/X3MAX)**2 ENRM0690
50          CONTINUE ENRM0700
60          CONTINUE ENRM0710
C               GO TO 80 ENRM0720
70          CONTINUE ENRM0730
C
C           SUM FOR INTERMEDIATE COMPONENTS. ENRM0740
C
C           S2 = S2 + XABS**2 ENRM0750
80          CONTINUE ENRM0760
90          CONTINUE ENRM0770
C
C           CALCULATION OF NORM. ENRM0780
C
C           IF (S1 .EQ. ZERO) GO TO 100 ENRM0790
C               ENORM = X1MAX*DSQRT(S1+(S2/X1MAX)/X1MAX)
C               GO TO 130 ENRM0800
100         CONTINUE ENRM0810
C               IF (S2 .EQ. ZERO) GO TO 110 ENRM0820
C                   IF (S2 .GE. X3MAX) ENRM0830
C                       ENORM = DSQRT(S2*(ONE+(X3MAX/S2)*(X3MAX*S3))) ENRM0840
C                   IF (S2 .LT. X3MAX) ENRM0850
C                       ENORM = DSQRT(X3MAX*((S2/X3MAX)+(X3MAX*S3))) ENRM0860
C                   GO TO 120 ENRM0870
110         CONTINUE ENRM0880
C               ENORM = X3MAX*DSQRT(S3) ENRM0890
120         CONTINUE ENRM0900
130         CONTINUE ENRM0910
C           RETURN ENRM0920
C
C           LAST CARD OF FUNCTION ENORM. ENRM0930
C
END ENRM0940

```

```

SUBROUTINE FDJAC1(FCN,N,X,FVEC,FJAC,LDFJAC,IFLAG,ML,MU,EPSCFN,          FDJ10010
*           WA1,WA2)                                     FDJ10020
INTEGER N,LDFJAC,IFLAG,ML,MU                      FDJ10030
DOUBLE PRECISION EPSCFN                           FDJ10040
DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N),WA1(N),WA2(N)      FDJ10050
*****                                              FDJ10060
C
C SUBROUTINE FDJAC1                               FDJ10070
C
C THIS SUBROUTINE COMPUTES A FORWARD-DIFFERENCE APPROXIMATION     FDJ10100
C TO THE N BY N JACOBIAN MATRIX ASSOCIATED WITH A SPECIFIED        FDJ10110
C PROBLEM OF N FUNCTIONS IN N VARIABLES. IF THE JACOBIAN HAS       FDJ10120
C A BANDED FORM, THEN FUNCTION EVALUATIONS ARE SAVED BY ONLY      FDJ10130
C APPROXIMATING THE NONZERO TERMS.                                FDJ10140
FDJ10150
C
C THE SUBROUTINE STATEMENT IS                         FDJ10160
FDJ10170
C
C SUBROUTINE FDJAC1(FCN,N,X,FVEC,FJAC,LDFJAC,IFLAG,ML,MU,EPSCFN,      FDJ10180
C           WA1,WA2)                                     FDJ10190
C
C WHERE                                              FDJ10200
FDJ10210
FDJ10220
C
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH            FDJ10230
C CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED                  FDJ10240
C IN AN EXTERNAL STATEMENT IN THE USER CALLING                   FDJ10250
C PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.                    FDJ10260
FDJ10270
C
C SUBROUTINE FCN(N,X,FVEC,IFLAG)                            FDJ10280
C INTEGER N,IFLAG                                         FDJ10290
C DOUBLE PRECISION X(N),FVEC(N)                          FDJ10300
C -----
C CALCULATE THE FUNCTIONS AT X AND                         FDJ10320
C RETURN THIS VECTOR IN FVEC.                           FDJ10330
C -----
C RETURN                                              FDJ10340
C END                                                 FDJ10350
FDJ10360
FDJ10370
C
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS        FDJ10380
C THE USER WANTS TO TERMINATE EXECUTION OF FDJAC1.             FDJ10390
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.          FDJ10400
FDJ10410
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER      FDJ10420
C OF FUNCTIONS AND VARIABLES.                                FDJ10430
FDJ10440
C
C X IS AN INPUT ARRAY OF LENGTH N.                           FDJ10450
FDJ10460
C
C FVEC IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE    FDJ10470
C FUNCTIONS EVALUATED AT X.                                FDJ10480
FDJ10490
C
C FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE          FDJ10500
C APPROXIMATION TO THE JACOBIAN MATRIX EVALUATED AT X.        FDJ10510
FDJ10520
C
C LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N   FDJ10530
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.     FDJ10540

```

```

C          FDJ10550
C          FDJ10560
C          FDJ10570
C          FDJ10580
C          FDJ10590
C          FDJ10600
C          FDJ10610
C          FDJ10620
C          FDJ10630
C          FDJ10640
C          FDJ10650
C          FDJ10660
C          FDJ10670
C          FDJ10680
C          FDJ10690
C          FDJ10700
C          FDJ10710
C          FDJ10720
C          FDJ10730
C          FDJ10740
C          FDJ10750
C          FDJ10760
C          FDJ10770
C          FDJ10780
C          FDJ10790
C          FDJ10800
C          FDJ10810
C          FDJ10820
C          FDJ10830
C          FDJ10840
C          FDJ10850
C          FDJ10860
C          FDJ10870
C          FDJ10880
C          FDJ10890
C          FDJ10900
C          FDJ10910
C          FDJ10920
C          FDJ10930
C          FDJ10940
C          FDJ10950
C          FDJ10960
C          FDJ10970
C          FDJ10980
C          FDJ10990
C          FDJ11000
C          FDJ11010
C          FDJ11020
C          FDJ11030
C          FDJ11040
C          FDJ11050
C          FDJ11060
C          FDJ11070
C          FDJ11080
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          *****
C          INTEGER I,J,K,MSUM
C          DOUBLE PRECISION EPS,EPSMCH,H,TEMP,ZERO
C          DOUBLE PRECISION DPMPAR
C          DATA ZERO /0.0D0/
C
C          EPSMCH IS THE MACHINE PRECISION.
C
C          EPSMCH = DPMPAR(1)
C
C          EPS = DSQRT(DMAX1(EPSFCN,EPSMCH))
C          MSUM = ML + MU + 1
C          IF (MSUM .LT. N) GO TO 40
C
C          COMPUTATION OF DENSE APPROXIMATE JACOBIAN.
C
C          DO 20 J = 1, N
C              TEMP = X(J)
C              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) FDJ20010
C INTEGER M,N,LDFJAC,IFLAG FDJ20020
C DOUBLE PRECISION EPSFCN FDJ20030
C DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),WA(M) FDJ20040
C **** FDJ20050
C SUBROUTINE FDJAC2 FDJ20060
C THIS SUBROUTINE COMPUTES A FORWARD-DIFFERENCE APPROXIMATION FDJ20070
C TO THE M BY N JACOBIAN MATRIX ASSOCIATED WITH A SPECIFIED FDJ20080
C PROBLEM OF M FUNCTIONS IN N VARIABLES. FDJ20090
C FDJ20100
C THE SUBROUTINE STATEMENT IS FDJ20110
C FDJ20120
C SUBROUTINE FDJAC2(FCN,M,N,X,FVEC,FJAC,LDFJAC,IFLAG,EPSFCN,WA) FDJ20130
C FDJ20140
C WHERE FDJ20150
C FDJ20160
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH FDJ20170
C CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED FDJ20180
C IN AN EXTERNAL STATEMENT IN THE USER CALLING FDJ20190
C PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS. FDJ20200
C FDJ20210
C FDJ20220
C FDJ20230
C SUBROUTINE FCN(M,N,X,FVEC,IFLAG) FDJ20240
C INTEGER M,N,IFLAG FDJ20250
C DOUBLE PRECISION X(N),FVEC(M) FDJ20260
C -----
C CALCULATE THE FUNCTIONS AT X AND FDJ20270
C RETURN THIS VECTOR IN FVEC. FDJ20280
C -----
C RETURN FDJ20290
C END FDJ20300
C
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS FDJ20310
C THE USER WANTS TO TERMINATE EXECUTION OF FDJAC2. FDJ20320
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER. FDJ20330
C
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER FDJ20340
C OF FUNCTIONS. FDJ20350
C FDJ20360
C FDJ20370
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER FDJ20380
C OF VARIABLES. N MUST NOT EXCEED M. FDJ20390
C FDJ20400
C
C X IS AN INPUT ARRAY OF LENGTH N. FDJ20410
C FDJ20420
C FDJ20430
C
C FVEC IS AN INPUT ARRAY OF LENGTH M WHICH MUST CONTAIN THE FDJ20440
C FUNCTIONS EVALUATED AT X. FDJ20450
C FDJ20460
C FDJ20470
C FDJ20480
C
C FJAC IS AN OUTPUT M BY N ARRAY WHICH CONTAINS THE FDJ20490
C APPROXIMATION TO THE JACOBIAN MATRIX EVALUATED AT X. FDJ20500
C FDJ20510
C
C LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M FDJ20520
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC. FDJ20530
C FDJ20540
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
C      EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE      FDJ20570
C      STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS      FDJ20580
C      APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE           FDJ20590
C      FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS        FDJ20600
C      THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE      FDJ20610
C      ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE          FDJ20620
C      PRECISION.                                                 FDJ20630
C
C      WA IS A WORK ARRAY OF LENGTH M.                                FDJ20640
C
C      SUBPROGRAMS CALLED                                         FDJ20650
C
C      USER-SUPPLIED ..... FCN                                     FDJ20660
C
C      MINPACK-SUPPLIED ... DPMPAR                               FDJ20670
C
C      FORTRAN-SUPPLIED ... DABS,DMAX1,DSQRT                      FDJ20680
C
C      ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.      FDJ20690
C      BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE       FDJ20700
C
C      *****
C      INTEGER I,J                                              FDJ20710
C      DOUBLE PRECISION EPS,EPSMCH,H,TEMP,ZERO                 FDJ20720
C      DOUBLE PRECISION DPMPAR                                FDJ20730
C      DATA ZERO /0.0D0/                                    FDJ20740
C
C      EPSMCH IS THE MACHINE PRECISION.                         FDJ20750
C
C      EPSMCH = DPMPAR(1)                                     FDJ20760
C
C      EPS = DSQRT(DMAX1(EPSFCN,EPSMCH))                   FDJ20770
C      DO 20 J = 1, N                                       FDJ20780
C          TEMP = X(J)                                      FDJ20790
C          H = EPS*DABS(TEMP)                            FDJ20800
C          IF (H .EQ. ZERO) H = EPS                     FDJ20810
C          X(J) = TEMP + H                           FDJ20820
C          CALL FCN(M,N,X,WA,IFLAG)                    FDJ20830
C          IF (IFLAG .LT. 0) GO TO 30                  FDJ20840
C          X(J) = TEMP                                FDJ20850
C          DO 10 I = 1, M                           FDJ20860
C              FJAC(I,J) = (WA(I) - FVEC(I))/H        FDJ20870
C 10          CONTINUE                                 FDJ20880
C 20          CONTINUE                                 FDJ20890
C 30          CONTINUE                                 FDJ20900
C          RETURN                                     FDJ20910
C
C      LAST CARD OF SUBROUTINE FDJAC2.                      FDJ20920
C
C      END                                         FDJ20930

```

```

SUBROUTINE HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,DIAG,          HYBD0010
*           MODE ,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,R,LR,          HYBD0020
*           QTF,WA1,WA2,WA3,WA4)          HYBD0030
INTEGER N,MAXFEV,ML,MU,MODE,NPRINT,INFO,NFEV,LDFJAC,LR          HYBD0040
DOUBLE PRECISION XTOL,EPSFCN,FACTOR          HYBD0050
DOUBLE PRECISION X(N),FVEC(N),DIAG(N),FJAC(LDFJAC,N),R(LR),          HYBD0060
*           QTF(N),WA1(N),WA2(N),WA3(N),WA4(N)          HYBD0070
EXTERNAL FCN          HYBD0080
*****          HYBD0090
C          HYBD0100
C          SUBROUTINE HYBRD          HYBD0110
C          HYBD0120
C          THE PURPOSE OF HYBRD IS TO FIND A ZERO OF A SYSTEM OF          HYBD0130
C          N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION          HYBD0140
C          OF THE POWELL HYBRID METHOD. THE USER MUST PROVIDE A          HYBD0150
C          SUBROUTINE WHICH CALCULATES THE FUNCTIONS. THE JACOBIAN IS          HYBD0160
C          THEN CALCULATED BY A FORWARD-DIFFERENCE APPROXIMATION.          HYBD0170
C          HYBD0180
C          THE SUBROUTINE STATEMENT IS          HYBD0190
C          HYBD0200
C          SUBROUTINE HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,          HYBD0210
C                           DIAG,MODE ,FACTOR,NPRINT,INFO,NFEV,FJAC,          HYBD0220
C                           LDFJAC,R,LR,QTF,WA1,WA2,WA3,WA4)          HYBD0230
C          HYBD0240
C          WHERE          HYBD0250
C          HYBD0260
C          FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH          HYBD0270
C          CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED          HYBD0280
C          IN AN EXTERNAL STATEMENT IN THE USER CALLING          HYBD0290
C          PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.          HYBD0300
C          HYBD0310
C          SUBROUTINE FCN(N,X,FVEC,IFLAG)          HYBD0320
C          INTEGER N,IFLAG          HYBD0330
C          DOUBLE PRECISION X(N),FVEC(N)
C          -----
C          CALCULATE THE FUNCTIONS AT X AND          HYBD0360
C          RETURN THIS VECTOR IN FVEC.          HYBD0370
C          -----
C          RETURN          HYBD0380
C          END          HYBD0390
C          HYBD0400
C          HYBD0410
C          THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS          HYBD0420
C          THE USER WANTS TO TERMINATE EXECUTION OF HYBRD.          HYBD0430
C          IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.          HYBD0440
C          HYBD0450
C          N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER          HYBD0460
C          OF FUNCTIONS AND VARIABLES.          HYBD0470
C          HYBD0480
C          X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN          HYBD0490
C          AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X          HYBD0500
C          CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.          HYBD0510
C          HYBD0520
C          FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS          HYBD0530
C          THE FUNCTIONS EVALUATED AT THE OUTPUT X.          HYBD0540

```

C	XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	HYBD0550
C	OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE	HYBD0560
C	ITERATES IS AT MOST XTOL.	HYBD0570
C	MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION	HYBD0580
C	OCCURS WHEN THE NUMBER OF CALLS TO FCN IS AT LEAST MAXFEV	HYBD0590
C	BY THE END OF AN ITERATION.	HYBD0600
C	ML IS A NONNEGATIVE INTEGER INPUT VARIABLE WHICH SPECIFIES	HYBD0610
C	THE NUMBER OF SUBDIAGONALS WITHIN THE BAND OF THE	HYBD0620
C	JACOBIAN MATRIX. IF THE JACOBIAN IS NOT BANDED, SET	HYBD0630
C	ML TO AT LEAST N - 1.	HYBD0640
C	MU IS A NONNEGATIVE INTEGER INPUT VARIABLE WHICH SPECIFIES	HYBD0650
C	THE NUMBER OF SUPERDIAGONALS WITHIN THE BAND OF THE	HYBD0660
C	JACOBIAN MATRIX. IF THE JACOBIAN IS NOT BANDED, SET	HYBD0670
C	MU TO AT LEAST N - 1.	HYBD0680
C	EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE	HYBD0690
C	STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS	HYBD0700
C	APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE	HYBD0710
C	FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS	HYBD0720
C	THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE	HYBD0730
C	ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE	HYBD0740
C	PRECISION.	HYBD0750
C	DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE	HYBD0760
C	BELOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG	HYBD0770
C	MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS	HYBD0780
C	MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.	HYBD0790
C	MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE	HYBD0800
C	VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,	HYBD0810
C	THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER	HYBD0820
C	VALUES OF MODE ARE EQUIVALENT TO MODE = 1.	HYBD0830
C	FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE	HYBD0840
C	INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF	HYBD0850
C	FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE	HYBD0860
C	TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE	HYBD0870
C	INTERVAL (.1,100.). 100. IS A GENERALLY RECOMMENDED VALUE.	HYBD0880
C	NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED	HYBD0890
C	PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,	HYBD0900
C	FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST	HYBD0910
C	ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND	HYBD0920
C	IMMEDIATELY PRIOR TO RETURN, WITH X AND FVEC AVAILABLE	HYBD0930
C	FOR PRINTING. IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS	HYBD0940
C	OF FCN WITH IFLAG = 0 ARE MADE.	HYBD0950
C	INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS	HYBD0960
C	TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)	HYBD0970
C	VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,	HYBD0980
C		HYBD0990
C		HYBD1000
C		HYBD1010
C		HYBD1020
C		HYBD1030
C		HYBD1040
C		HYBD1050
C		HYBD1060
C		HYBD1070
C		HYBD1080

C INFO IS SET AS FOLLOWS. HYBD1090
C C HYBD1100
C INFO = 0 IMPROPER INPUT PARAMETERS. HYBD1110
C C HYBD1120
C INFO = 1 RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES HYBD1130
C C IS AT MOST XTOL. HYBD1140
C C HYBD1150
C INFO = 2 NUMBER OF CALLS TO FCN HAS REACHED OR EXCEEDED HYBD1160
C C MAXFEV. HYBD1170
C C HYBD1180
C INFO = 3 XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN HYBD1190
C C THE APPROXIMATE SOLUTION X IS POSSIBLE. HYBD1200
C C HYBD1210
C INFO = 4 ITERATION IS NOT MAKING GOOD PROGRESS, AS HYBD1220
C C MEASURED BY THE IMPROVEMENT FROM THE LAST HYBD1230
C C FIVE JACOBIAN EVALUATIONS. HYBD1240
C C HYBD1250
C INFO = 5 ITERATION IS NOT MAKING GOOD PROGRESS, AS HYBD1260
C C MEASURED BY THE IMPROVEMENT FROM THE LAST HYBD1270
C C TEN ITERATIONS. HYBD1280
C C HYBD1290
C NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF HYBD1300
C C CALLS TO FCN. HYBD1310
C C HYBD1320
C FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE HYBD1330
C C ORTHOGONAL MATRIX Q PRODUCED BY THE QR FACTORIZATION HYBD1340
C C OF THE FINAL APPROXIMATE JACOBIAN. HYBD1350
C C HYBD1360
C LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N HYBD1370
C C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC. HYBD1380
C C HYBD1390
C R IS AN OUTPUT ARRAY OF LENGTH LR WHICH CONTAINS THE HYBD1400
C C UPPER TRIANGULAR MATRIX PRODUCED BY THE QR FACTORIZATION HYBD1410
C C OF THE FINAL APPROXIMATE JACOBIAN, STORED ROWWISE. HYBD1420
C C HYBD1430
C LR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN HYBD1440
C C (N*(N+1))/2. HYBD1450
C C HYBD1460
C QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS HYBD1470
C C THE VECTOR (Q TRANSPOSE)*FVEC. HYBD1480
C C HYBD1490
C WA1, WA2, WA3, AND WA4 ARE WORK ARRAYS OF LENGTH N. HYBD1500
C C HYBD1510
C SUBPROGRAMS CALLED HYBD1520
C C HYBD1530
C USER-SUPPLIED FCN HYBD1540
C C HYBD1550
C MINPACK-SUPPLIED ... DOGLEG,DMPMPAR,ENORM,FDJAC1, HYBD1560
C C QFORM,QRFAC,R1MPYQ,R1UPDT HYBD1570
C C HYBD1580
C FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,MIN0,MOD HYBD1590
C C HYBD1600
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. HYBD1610
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE HYBD1620

```

C          HYBD1630
C          HYBD1640
C *****
C          HYBD1650
C          HYBD1660
C          HYBD1670
C          HYBD1680
C          HYBD1690
C          HYBD1700
C          HYBD1710
C          HYBD1720
C          HYBD1730
C          HYBD1740
C          HYBD1750
C          HYBD1760
C          HYBD1770
C          HYBD1780
C          HYBD1790
C          HYBD1800
C          HYBD1810
C          HYBD1820
C          HYBD1830
C          HYBD1840
C          HYBD1850
C          HYBD1860
C          HYBD1870
C          HYBD1880
C          HYBD1890
C          HYBD1900
C          HYBD1910
C          HYBD1920
C          HYBD1930
C          HYBD1940
C          HYBD1950
C          HYBD1960
C          HYBD1970
C          HYBD1980
C          HYBD1990
C          HYBD2000
C          HYBD2010
C          HYBD2020
C          HYBD2030
C          HYBD2040
C          HYBD2050
C          HYBD2060
C          HYBD2070
C          HYBD2080
C          HYBD2090
C          HYBD2100
C          HYBD2110
C          HYBD2120
C          HYBD2130
C          HYBD2140
C          HYBD2150
C          HYBD2160
C
C          *****

INTEGER I,IFLAG,ITER,J,JM1,L,MSUM,NCFAIL,NCSUC,NSLOW1,NSLOW2      HYBD1630
INTEGER IWA(1)                                                       HYBD1640
LOGICAL JEVAL,SING                                                 HYBD1650
DOUBLE PRECISION ACTRED,DELTA,EPSMCH,FNORM,FNORM1,ONE,PNORM,        HYBD1660
*                  PRERED,P1,P5,P001,P0001,RATIO,SUM,TEMP,XNORM,        HYBD1670
*                  ZERO                                         HYBD1680
DOUBLE PRECISION DPMPAR,ENORM                                     HYBD1690
DATA ONE,P1,P5,P001,P0001,ZERO                                    HYBD1700
*                  /1.0D0,1.0D-1,5.0D-1,1.0D-3,1.0D-4,0.0D0/          HYBD1710
C          EPSMCH IS THE MACHINE PRECISION.                         HYBD1720
C          EPSMCH = DPMPAR(1)                                       HYBD1730
C          INFO = 0                                              HYBD1740
C          IFLAG = 0                                             HYBD1750
C          NFEV = 0                                              HYBD1760
C          CHECK THE INPUT PARAMETERS FOR ERRORS.                 HYBD1770
C          IF (N .LE. 0 .OR. XTOL .LT. ZERO .OR. MAXFEV .LE. 0       HYBD1780
*          .OR. ML .LT. 0 .OR. MU .LT. 0 .OR. FACTOR .LE. ZERO       HYBD1790
*          .OR. LDFJAC .LT. N .OR. LR .LT. (N*(N + 1))/2) GO TO 300   HYBD1800
IF (MODE .NE. 2) GO TO 20                                         HYBD1810
DO 10 J = 1, N
    IF (DIAG(J) .LE. ZERO) GO TO 300                           HYBD1820
10    CONTINUE                                         HYBD1830
20    CONTINUE                                         HYBD1840
C          EVALUATE THE FUNCTION AT THE STARTING POINT           HYBD1850
C          AND CALCULATE ITS NORM.                                HYBD1860
C          IFLAG = 1                                            HYBD1870
CALL FCN(N,X,FVEC,IFLAG)                                         HYBD1880
NFEV = 1                                            HYBD1890
IF (IFLAG .LT. 0) GO TO 300                                     HYBD1900
FNORM = ENORM(N,FVEC)                                         HYBD1910
C          DETERMINE THE NUMBER OF CALLS TO FCN NEEDED TO COMPUTE   HYBD1920
C          THE JACOBIAN MATRIX.                                 HYBD1930
C          MSUM = MIN0(ML+MU+1,N)                               HYBD1940
C          INITIALIZE ITERATION COUNTER AND MONITORS.            HYBD1950
C          ITER = 1                                              HYBD1960
NCSUC = 0                                              HYBD1970
NCFAIL = 0                                              HYBD1980
NSLOW1 = 0                                              HYBD1990
NSLOW2 = 0                                              HYBD2000
C          BEGINNING OF THE OUTER LOOP.                          HYBD2010

```

```

C          HYBD2170
30 CONTINUE          HYBD2180
      JEVAL = .TRUE.          HYBD2190
C          HYBD2200
C          HYBD2210
C          HYBD2220
C          HYBD2230
C          HYBD2240
C          HYBD2250
C          HYBD2260
C          HYBD2270
C          HYBD2280
C          HYBD2290
C          HYBD2300
C          HYBD2310
C          HYBD2320
C          HYBD2330
C          HYBD2340
C          HYBD2350
C          HYBD2360
C          HYBD2370
C          HYBD2380
C          HYBD2390
C          HYBD2400
40    CONTINUE          HYBD2410
50    CONTINUE          HYBD2420
C          HYBD2430
C          HYBD2440
C          HYBD2450
C          HYBD2460
C          HYBD2470
C          HYBD2480
C          HYBD2490
C          HYBD2500
C          HYBD2510
C          HYBD2520
C          HYBD2530
C          HYBD2540
C          HYBD2550
C          HYBD2560
C          HYBD2570
C          HYBD2580
C          HYBD2590
C          HYBD2600
C          HYBD2610
C          HYBD2620
C          HYBD2630
C          HYBD2640
C          HYBD2650
C          HYBD2660
C          HYBD2670
C          HYBD2680
C          HYBD2690
100   CONTINUE          HYBD2700
110   CONTINUE

```

C HYBD2170
30 CONTINUE HYBD2180
 JEVAL = .TRUE. HYBD2190
C HYBD2200
C HYBD2210
C HYBD2220
C HYBD2230
C HYBD2240
C HYBD2250
C HYBD2260
C HYBD2270
C HYBD2280
C HYBD2290
C HYBD2300
C HYBD2310
C HYBD2320
C HYBD2330
C HYBD2340
C HYBD2350
C HYBD2360
C HYBD2370
C HYBD2380
C HYBD2390
C HYBD2400
40 CONTINUE HYBD2410
50 CONTINUE HYBD2420
C HYBD2430
C HYBD2440
C HYBD2450
C HYBD2460
C HYBD2470
C HYBD2480
C HYBD2490
C HYBD2500
C HYBD2510
C HYBD2520
C HYBD2530
C HYBD2540
C HYBD2550
C HYBD2560
C HYBD2570
C HYBD2580
C HYBD2590
C HYBD2600
C HYBD2610
C HYBD2620
C HYBD2630
C HYBD2640
C HYBD2650
C HYBD2660
C HYBD2670
C HYBD2680
C HYBD2690
100 CONTINUE HYBD2700
110 CONTINUE

```

120      CONTINUE                                HYBD2710
C
C      COPY THE TRIANGULAR FACTOR OF THE QR FACTORIZATION INTO R.    HYBD2720
C
C      SING = .FALSE.                                HYBD2730
      DO 150 J = 1, N                                HYBD2740
         L = J
         JM1 = J - 1
         IF (JM1 .LT. 1) GO TO 140                HYBD2750
         DO 130 I = 1, JM1
            R(L) = FJAC(I,J)
            L = L + N - I
130      CONTINUE                                HYBD2760
140      CONTINUE                                HYBD2770
         R(L) = WA1(J)
         IF (WA1(J) .EQ. ZERO) SING = .TRUE.       HYBD2780
150      CONTINUE                                HYBD2790
C
C      ACCUMULATE THE ORTHOGONAL FACTOR IN FJAC.   HYBD2800
C
C      CALL QFORM(N,N,FJAC,LDFJAC,WA1)             HYBD2810
C
C      RESCALE IF NECESSARY.                      HYBD2820
C
         IF (MODE .EQ. 2) GO TO 170                HYBD2830
         DO 160 J = 1, N                                HYBD2840
            DIAG(J) = DMAX1(DIAG(J),WA2(J))
160      CONTINUE                                HYBD2850
170      CONTINUE                                HYBD2860
C
C      BEGINNING OF THE INNER LOOP.               HYBD2870
C
180      CONTINUE                                HYBD2880
C
C      IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.   HYBD2890
C
         IF (NPRINT .LE. 0) GO TO 190                HYBD2900
         IFLAG = 0
         IF (MOD(ITER-1,NPRINT) .EQ. 0) CALL FCN(N,X,FVEC,IFLAG)  HYBD2910
         IF (IFLAG .LT. 0) GO TO 300                HYBD2920
190      CONTINUE                                HYBD2930
C
C      DETERMINE THE DIRECTION P.                 HYBD2940
C
         CALL DOGLEG(N,R,LR,DIAG,QTF,DELTA,WA1,WA2,WA3)        HYBD2950
C
C      STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.  HYBD2960
C
         DO 200 J = 1, N                                HYBD2970
            WA1(J) = -WA1(J)
            WA2(J) = X(J) + WA1(J)
            WA3(J) = DIAG(J)*WA1(J)
200      CONTINUE                                HYBD2980
         PNORM = ENORM(N,WA3)                         HYBD2990

```

```

C          HYBD3250
C          ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.  HYBD3260
C          HYBD3270
C          IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)  HYBD3280
C          HYBD3290
C          EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.  HYBD3300
C          HYBD3310
C          IFLAG = 1  HYBD3320
C          CALL FCN(N,WA2,WA4,IFLAG)  HYBD3330
C          NFEV = NFEV + 1  HYBD3340
C          IF (IFLAG .LT. 0) GO TO 300  HYBD3350
C          FNORM1 = ENORM(N,WA4)  HYBD3360
C          HYBD3370
C          COMPUTE THE SCALED ACTUAL REDUCTION.  HYBD3380
C          HYBD3390
C          ACTRED = -ONE  HYBD3400
C          IF (FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2  HYBD3410
C          HYBD3420
C          COMPUTE THE SCALED PREDICTED REDUCTION.  HYBD3430
C          HYBD3440
C          L = 1  HYBD3450
C          DO 220 I = 1, N  HYBD3460
C          SUM = ZERO  HYBD3470
C          DO 210 J = I, N  HYBD3480
C          SUM = SUM + R(L)*WA1(J)  HYBD3490
C          L = L + 1  HYBD3500
210        CONTINUE  HYBD3510
          WA3(I) = QTF(I) + SUM  HYBD3520
220        CONTINUE  HYBD3530
          TEMP = ENORM(N,WA3)  HYBD3540
          PRERED = ZERO  HYBD3550
          IF (TEMP .LT. FNORM) PRERED = ONE - (TEMP/FNORM)**2  HYBD3560
          HYBD3570
C          COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED  HYBD3580
C          REDUCTION.  HYBD3590
C          HYBD3600
C          RATIO = ZERO  HYBD3610
C          IF (PRERED .GT. ZERO) RATIO = ACTRED/PRERED  HYBD3620
C          HYBD3630
C          UPDATE THE STEP BOUND.  HYBD3640
C          HYBD3650
C          IF (RATIO .GE. P1) GO TO 230  HYBD3660
          NCSUC = 0  HYBD3670
          NCFAIL = NCFAIL + 1  HYBD3680
          DELTA = P5*DELTA  HYBD3690
          GO TO 240  HYBD3700
230        CONTINUE  HYBD3710
          NCFAIL = 0  HYBD3720
          NCSUC = NCSUC + 1  HYBD3730
          IF (RATIO .GE. P5 .OR. NCSUC .GT. 1)  HYBD3740
*           DELTA = DMAX1(DELTA,PNORM/P5)  HYBD3750
          IF (DABS(RATIO-ONE) .LE. P1) DELTA = PNORM/P5  HYBD3760
240        CONTINUE  HYBD3770
C          HYBD3780

```

```

C TEST FOR SUCCESSFUL ITERATION. HYBD3790
C IF (RATIO .LT. P0001) GO TO 260 HYBD3800
C SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS. HYBD3820
C HYBD3830
C HYBD3840
DO 250 J = 1, N HYBD3850
X(J) = WA2(J) HYBD3860
WA2(J) = DIAG(J)*X(J) HYBD3870
FVEC(J) = WA4(J) HYBD3880
250 CONTINUE HYBD3890
XNORM = ENORM(N,WA2) HYBD3900
FNORM = FNORM1 HYBD3910
ITER = ITER + 1 HYBD3920
260 CONTINUE HYBD3930
C HYBD3940
C DETERMINE THE PROGRESS OF THE ITERATION. HYBD3950
C HYBD3960
NSLOW1 = NSLOW1 + 1 HYBD3970
IF (ACTRED .GE. P001) NSLOW1 = 0 HYBD3980
IF (JEVAL) NSLOW2 = NSLOW2 + 1 HYBD3990
IF (ACTRED .GE. P1) NSLOW2 = 0 HYBD4000
C HYBD4010
C TEST FOR CONVERGENCE. HYBD4020
C HYBD4030
IF (DELTA .LE. XTOL*XNORM .OR. FNORM .EQ. ZERO) INFO = 1 HYBD4040
IF (INFO .NE. 0) GO TO 300 HYBD4050
C HYBD4060
C TESTS FOR TERMINATION AND STRINGENT TOLERANCES. HYBD4070
C HYBD4080
IF (NFEV .GE. MAXFEV) INFO = 2 HYBD4090
IF (P1*DMAX1(P1*DELTA,PNORM) .LE. EPSMCH*XNORM) INFO = 3 HYBD4100
IF (NSLOW2 .EQ. 5) INFO = 4 HYBD4110
IF (NSLOW1 .EQ. 10) INFO = 5 HYBD4120
IF (INFO .NE. 0) GO TO 300 HYBD4130
C HYBD4140
C CRITERION FOR RECALCULATING JACOBIAN APPROXIMATION HYBD4150
C BY FORWARD DIFFERENCES. HYBD4160
C HYBD4170
IF (NCFAIL .EQ. 2) GO TO 290 HYBD4180
C HYBD4190
C CALCULATE THE RANK ONE MODIFICATION TO THE JACOBIAN HYBD4200
C AND UPDATE QTF IF NECESSARY. HYBD4210
C HYBD4220
DO 280 J = 1, N HYBD4230
SUM = ZERO HYBD4240
DO 270 I = 1, N HYBD4250
SUM = SUM + FJAC(I,J)*WA4(I) HYBD4260
270 CONTINUE HYBD4270
WA2(J) = (SUM - WA3(J))/PNORM HYBD4280
WA1(J) = DIAG(J)*((DIAG(J)*WA1(J))/PNORM) HYBD4290
IF (RATIO .GE. P0001) QTF(J) = SUM HYBD4300
280 CONTINUE HYBD4310
C HYBD4320

```

```

C COMPUTE THE QR FACTORIZATION OF THE UPDATED JACOBIAN. HYBD4330
C
C CALL R1UPDT(N,N,R,LR,WA1,WA2,WA3,SING) HYBD4340
C CALL R1MPYQ(N,N,FJAC,LDFJAC,WA2,WA3) HYBD4350
C CALL R1MPYQ(1,N,QT,1,WA2,WA3) HYBD4360
C
C END OF THE INNER LOOP. HYBD4370
C
C JEVAL = .FALSE. HYBD4380
C GO TO 180 HYBD4390
290 CONTINUE HYBD4400
C
C END OF THE OUTER LOOP. HYBD4410
C
C GO TO 30 HYBD4420
300 CONTINUE HYBD4430
C
C TERMINATION, EITHER NORMAL OR USER IMPOSED. HYBD4440
C
C IF (IFLAG .LT. 0) INFO = IFLAG HYBD4450
IFLAG = 0 HYBD4460
IF (NPRINT .GT. 0) CALL FCN(N,X,FVEC,IFLAG) HYBD4470
RETURN HYBD4480
C
C LAST CARD OF SUBROUTINE HYBRD. HYBD4490
C
C END HYBD4500

```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

SUBROUTINE HYBRD1(FCN,N,X,FVEC,TOL,INFO,WA,LWA)	HYD10010
INTEGER N,INFO,LWA	HYD10020
DOUBLE PRECISION TOL	HYD10030
DOUBLE PRECISION X(N),FVEC(N),WA(LWA)	HYD10040
EXTERNAL FCN	HYD10050
*****	HYD10060
C	HYD10070
C SUBROUTINE HYBRD1	HYD10080
C	HYD10090
C THE PURPOSE OF HYBRD1 IS TO FIND A ZERO OF A SYSTEM OF	HYD10100
C N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION	HYD10110
C OF THE POWELL HYBRID METHOD. THIS IS DONE BY USING THE	HYD10120
C MORE GENERAL NONLINEAR EQUATION SOLVER HYBRD. THE USER	HYD10130
C MUST PROVIDE A SUBROUTINE WHICH CALCULATES THE FUNCTIONS.	HYD10140
C THE JACOBIAN IS THEN CALCULATED BY A FORWARD-DIFFERENCE	HYD10150
C APPROXIMATION.	HYD10160
C	HYD10170
C THE SUBROUTINE STATEMENT IS	HYD10180
C	HYD10190
C SUBROUTINE HYBRD1(FCN,N,X,FVEC,TOL,INFO,WA,LWA)	HYD10200
C	HYD10210
C WHERE	HYD10220
C	HYD10230
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH	HYD10240
C CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED	HYD10250
C IN AN EXTERNAL STATEMENT IN THE USER CALLING	HYD10260
C PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.	HYD10270
C	HYD10280
C SUBROUTINE FCN(N,X,FVEC,IFLAG)	HYD10290
C INTEGER N,IFLAG	HYD10300
C DOUBLE PRECISION X(N),FVEC(N)	HYD10310
C -----	HYD10320
C CALCULATE THE FUNCTIONS AT X AND	HYD10330
C RETURN THIS VECTOR IN FVEC.	HYD10340
C -----	HYD10350
C RETURN	HYD10360
C END	HYD10370
C	HYD10380
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS	HYD10390
C THE USER WANTS TO TERMINATE EXECUTION OF HYBRD1.	HYD10400
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.	HYD10410
C	HYD10420
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	HYD10430
C OF FUNCTIONS AND VARIABLES.	HYD10440
C	HYD10450
C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN	HYD10460
C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X	HYD10470
C CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.	HYD10480
C	HYD10490
C FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS	HYD10500
C THE FUNCTIONS EVALUATED AT THE OUTPUT X.	HYD10510
C	HYD10520
C TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS	HYD10530
C WHEN THE ALGORITHM ESTIMATES THAT THE RELATIVE ERROR	HYD10540

C BETWEEN X AND THE SOLUTION IS AT MOST TOL. HYD10550
 C HYD10560
 C INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS HYD10570
 C TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE) HYD10580
 C VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE, HYD10590
 C INFO IS SET AS FOLLOWS. HYD10600
 C HYD10610
 C INFO = 0 IMPROPER INPUT PARAMETERS. HYD10620
 C HYD10630
 C INFO = 1 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR HYD10640
 C BETWEEN X AND THE SOLUTION IS AT MOST TOL. HYD10650
 C HYD10660
 C INFO = 2 NUMBER OF CALLS TO FCN HAS REACHED OR EXCEEDED HYD10670
 C 200*(N+1). HYD10680
 C HYD10690
 C INFO = 3 TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN HYD10700
 C THE APPROXIMATE SOLUTION X IS POSSIBLE. HYD10710
 C HYD10720
 C INFO = 4 ITERATION IS NOT MAKING GOOD PROGRESS. HYD10730
 C HYD10740
 C WA IS A WORK ARRAY OF LENGTH LWA. HYD10750
 C HYD10760
 C LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN HYD10770
 C (N*(3*N+13))/2. HYD10780
 C HYD10790
 C SUBPROGRAMS CALLED HYD10800
 C HYD10810
 C USER-SUPPLIED FCN HYD10820
 C HYD10830
 C MINPACK-SUPPLIED ... HYBRD HYD10840
 C HYD10850
 C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. HYD10860
 C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE HYD10870
 C HYD10880
 C ***** HYD10890
 C INTEGER INDEX,J,LR,MAXFEV,ML,MODE,MU,NFEV,NPRINT HYD10900
 C DOUBLE PRECISION EPSFCN,FACTOR,ONE,XTOL,ZERO HYD10910
 C DATA FACTOR,ONE,ZERO /1.0D2,1.0D0,0.0D0/ HYD10920
 C INFO = 0 HYD10930
 C HYD10940
 C CHECK THE INPUT PARAMETERS FOR ERRORS. HYD10950
 C HYD10960
 C IF (N .LE. 0 .OR. TOL .LT. ZERO .OR. LWA .LT. (N*(3*N + 13))/2) HYD10970
 * GO TO 20 HYD10980
 C HYD10990
 C CALL HYBRD. HYD11000
 C HYD11010
 C MAXFEV = 200*(N + 1) HYD11020
 C XTOL = TOL HYD11030
 C ML = N - 1 HYD11040
 C MU = N - 1 HYD11050
 C EPSFCN = ZERO HYD11060
 C MODE = 2 HYD11070
 DO 10 J = 1, N HYD11080

WA(J) = ONE	HYD11090
10 CONTINUE	HYD11100
NPRINT = 0	HYD11110
LR = (N*(N + 1))/2	HYD11120
INDEX = 6*N + LR	HYD11130
CALL HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,WA(1),MODE,	HYD11140
* FACTOR,NPRINT,INFO,NFEV,WA(INDEX+1),N,WA(6*N+1),LR,	HYD11150
* WA(N+1),WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))	HYD11160
IF (INFO .EQ. 5) INFO = 4	HYD11170
20 CONTINUE	HYD11180
RETURN	HYD11190
C LAST CARD OF SUBROUTINE HYBRD1.	HYD11200
C	HYD11210
C	HYD11220
END	HYD11230

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```

SUBROUTINE HYBRJ(FCN,N,X,FVEC,FJAC,LDFJAC,XTOL,MAXFEV,DIAG,MODE,      HYBJ0010
*           FACTOR,NPRINT,INFO,NFEV,NJEV,R,LR,QTF,WA1,WA2,      HYBJ0020
*           WA3,WA4)                                              HYBJ0030
INTEGER N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV,LR          HYBJ0040
DOUBLE PRECISION XTOL,FACTOR                                HYBJ0050
DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N),DIAG(N),R(LR),      HYBJ0060
*           QTF(N),WA1(N),WA2(N),WA3(N),WA4(N)                  HYBJ0070
C *****
C
C SUBROUTINE HYBRJ                                         HYBJ0080
C
C THE PURPOSE OF HYBRJ IS TO FIND A ZERO OF A SYSTEM OF      HYBJ0090
C N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION     HYBJ0100
C OF THE POWELL HYBRID METHOD. THE USER MUST PROVIDE A       HYBJ0110
C SUBROUTINE WHICH CALCULATES THE FUNCTIONS AND THE JACOBIAN.   HYBJ0120
C
C THE SUBROUTINE STATEMENT IS                               HYBJ0130
C
C SUBROUTINE HYBRJ(FCN,N,X,FVEC,FJAC,LDFJAC,XTOL,MAXFEV,DIAG,      HYBJ0140
C                   MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,R,LR,QTF,      HYBJ0150
C                   WA1,WA2,WA3,WA4)                                HYBJ0160
C
C WHERE                                                 HYBJ0170
C
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH      HYBJ0180
C CALCULATES THE FUNCTIONS AND THE JACOBIAN. FCN MUST        HYBJ0190
C BE DECLARED IN AN EXTERNAL STATEMENT IN THE USER          HYBJ0200
C CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.        HYBJ0210
C
C SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)                HYBJ0220
C INTEGER N,LDFJAC,IFLAG                                     HYBJ0230
C DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)             HYBJ0240
C -----
C IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND            HYBJ0250
C RETURN THIS VECTOR IN FVEC. DO NOT ALTER FJAC.           HYBJ0260
C IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND            HYBJ0270
C RETURN THIS MATRIX IN FJAC. DO NOT ALTER FVEC.           HYBJ0280
C -----
C RETURN                                               HYBJ0290
C END
C
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS    HYBJ0300
C THE USER WANTS TO TERMINATE EXECUTION OF HYBRJ.           HYBJ0310
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.            HYBJ0320
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER    HYBJ0330
C OF FUNCTIONS AND VARIABLES.                            HYBJ0340
C
C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN        HYBJ0350
C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X     HYBJ0360
C CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.        HYBJ0370
C
C FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS        HYBJ0380
C THE FUNCTIONS EVALUATED AT THE OUTPUT X.                  HYBJ0390

```

C	FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE	HYBJ0550
C	ORTHOGONAL MATRIX Q PRODUCED BY THE QR FACTORIZATION	HYBJ0560
C	OF THE FINAL APPROXIMATE JACOBIAN.	HYBJ0570
C	LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N	HYBJ0580
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.	HYBJ0590
C	XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	HYBJ0600
C	OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE	HYBJ0610
C	ITERATES IS AT MOST XTOL.	HYBJ0620
C	MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION	HYBJ0630
C	OCCURS WHEN THE NUMBER OF CALLS TO FCN WITH IFLAG = 1	HYBJ0640
C	HAS REACHED MAXFEV.	HYBJ0650
C	DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE	HYBJ0660
C	BELLOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG	HYBJ0670
C	MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS	HYBJ0680
C	MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.	HYBJ0690
C	MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE	HYBJ0700
C	VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,	HYBJ0710
C	THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER	HYBJ0720
C	VALUES OF MODE ARE EQUIVALENT TO MODE = 1.	HYBJ0730
C	FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE	HYBJ0740
C	INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF	HYBJ0750
C	FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE	HYBJ0760
C	TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE	HYBJ0770
C	INTERVAL (.1,100.). 100. IS A GENERALLY RECOMMENDED VALUE.	HYBJ0780
C	NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED	HYBJ0790
C	PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,	HYBJ0800
C	FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST	HYBJ0810
C	ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND	HYBJ0820
C	IMMEDIATELY PRIOR TO RETURN, WITH X AND FVEC AVAILABLE	HYBJ0830
C	FOR PRINTING. FVEC AND FJAC SHOULD NOT BE ALTERED.	HYBJ0840
C	IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS OF FCN	HYBJ0850
C	WITH IFLAG = 0 ARE MADE.	HYBJ0860
C	INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS	HYBJ0870
C	TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)	HYBJ0880
C	VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,	HYBJ0890
C	INFO IS SET AS FOLLOWS.	HYBJ0900
C	INFO = 0 IMPROPER INPUT PARAMETERS.	HYBJ0910
C	INFO = 1 RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES	HYBJ0920
C	IS AT MOST XTOL.	HYBJ0930
C	INFO = 2 NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS	HYBJ0940
C	REACHED MAXFEV.	HYBJ0950
C		HYBJ0960
C		HYBJ0970
C		HYBJ0980
C		HYBJ0990
C		HYBJ1000
C		HYBJ1010
C		HYBJ1020
C		HYBJ1030
C		HYBJ1040
C		HYBJ1050
C		HYBJ1060
C		HYBJ1070
C		HYBJ1080

```

C INFO = 3 XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN          HYBJ1090
C THE APPROXIMATE SOLUTION X IS POSSIBLE.                      HYBJ1100
C
C INFO = 4 ITERATION IS NOT MAKING GOOD PROGRESS, AS          HYBJ1110
C MEASURED BY THE IMPROVEMENT FROM THE LAST                  HYBJ1120
C FIVE JACOBIAN EVALUATIONS.                                HYBJ1130
C
C INFO = 5 ITERATION IS NOT MAKING GOOD PROGRESS, AS          HYBJ1140
C MEASURED BY THE IMPROVEMENT FROM THE LAST                  HYBJ1150
C TEN ITERATIONS.                                         HYBJ1160
C
C NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF    HYBJ1170
C CALLS TO FCN WITH IFLAG = 1.                               HYBJ1180
C
C NJEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF    HYBJ1190
C CALLS TO FCN WITH IFLAG = 2.                               HYBJ1200
C
C R IS AN OUTPUT ARRAY OF LENGTH LR WHICH CONTAINS THE      HYBJ1210
C UPPER TRIANGULAR MATRIX PRODUCED BY THE QR FACTORIZATION   HYBJ1220
C OF THE FINAL APPROXIMATE JACOBIAN, STORED ROWWISE.        HYBJ1230
C
C LR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN     HYBJ1240
C (N*(N+1))/2.                                              HYBJ1250
C
C QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS         HYBJ1260
C THE VECTOR (Q TRANSPOSE)*FVEC.                            HYBJ1270
C
C WA1, WA2, WA3, AND WA4 ARE WORK ARRAYS OF LENGTH N.       HYBJ1280
C
C SUBPROGRAMS CALLED
C
C USER-SUPPLIED ..... FCN                                 HYBJ1290
C
C MINPACK-SUPPLIED ... DOGLE,DPMPAR,ENORM,                HYBJ1300
C QFORM,QRFAC,R1MPYQ,R1UPDT                           HYBJ1310
C
C FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,MOD             HYBJ1320
C
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.  HYBJ1330
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE    HYBJ1340
C
C *****
C INTEGER I,IFLAG,ITER,J,JM1,L,NCFAIL,NCSUC,NSLOW1,NSLOW2  HYBJ1350
C INTEGER IWA(1)                                         HYBJ1360
C LOGICAL JEVAL,SING                                     HYBJ1370
C DOUBLE PRECISION ACTRED,DELTA,EPSMCH,FNORM,FNORM1,ONE,PNORM, HYBJ1380
C * PRERED,P1,P5,P001,P0001,RATIO,SUM,TEMP,XNORM,          HYBJ1390
C * ZERO                                                 HYBJ1400
C DOUBLE PRECISION DPMPAR,ENORM                         HYBJ1410
C DATA ONE,P1,P5,P001,P0001,ZERO                      HYBJ1420
C * /1.0D0,1.0D-1,5.0D-1,1.0D-3,1.0D-4,0.0D0/          HYBJ1430
C
C EPSMCH IS THE MACHINE PRECISION.                     HYBJ1440
C
C

```

```

EPSMCH = DPMPAR(1) HYBJ1630
C
INFO = 0 HYBJ1640
IFLAG = 0 HYBJ1650
NFEV = 0 HYBJ1660
NJEV = 0 HYBJ1670
HYBJ1680
HYBJ1690
C
C CHECK THE INPUT PARAMETERS FOR ERRORS. HYBJ1700
C
IF (N .LE. 0 .OR. LDFJAC .LT. N .OR. XTOL .LT. ZERO HYBJ1710
* .OR. MAXFEV .LE. 0 .OR. FACTOR .LE. ZERO HYBJ1720
* .OR. LR .LT. (N*(N + 1))/2) GO TO 300 HYBJ1730
IF (MODE .NE. 2) GO TO 20 HYBJ1740
DO 10 J = 1, N HYBJ1750
IF (DIAG(J) .LE. ZERO) GO TO 300 HYBJ1760
10 CONTINUE HYBJ1770
20 CONTINUE HYBJ1780
HYBJ1790
C
C EVALUATE THE FUNCTION AT THE STARTING POINT HYBJ1800
C AND CALCULATE ITS NORM. HYBJ1810
HYBJ1820
HYBJ1830
C
IFLAG = 1 HYBJ1840
CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG) HYBJ1850
NFEV = 1 HYBJ1860
IF (IFLAG .LT. 0) GO TO 300 HYBJ1870
FNORM = ENORM(N,FVEC) HYBJ1880
C
C INITIALIZE ITERATION COUNTER AND MONITORS. HYBJ1890
HYBJ1900
HYBJ1910
C
ITER = 1 HYBJ1920
NCSUC = 0 HYBJ1930
NCFAIL = 0 HYBJ1940
NSLOW1 = 0 HYBJ1950
NSLOW2 = 0 HYBJ1960
C
C BEGINNING OF THE OUTER LOOP. HYBJ1970
HYBJ1980
HYBJ1990
C
30 CONTINUE HYBJ2000
JEVAL = .TRUE. HYBJ2010
C
C CALCULATE THE JACOBIAN MATRIX. HYBJ2020
HYBJ2030
HYBJ2040
C
IFLAG = 2 HYBJ2050
CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG) HYBJ2060
NJEV = NJEV + 1 HYBJ2070
IF (IFLAG .LT. 0) GO TO 300 HYBJ2080
C
C COMPUTE THE QR FACTORIZATION OF THE JACOBIAN. HYBJ2090
HYBJ2100
HYBJ2110
C
CALL QRFAC(N,N,FJAC,LDFJAC,.FALSE.,IWA,1,WA1,WA2,WA3) HYBJ2120
C
C ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING HYBJ2130
TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN. HYBJ2140
HYBJ2150
HYBJ2160

```

```

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
C      ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X   HYBJ2240
C      AND INITIALIZE THE STEP BOUND DELTA.                      HYBJ2250
C
C      DO 60 J = 1, N                                         HYBJ2260
C          WA3(J) = DIAG(J)*X(J)                            HYBJ2270
60      CONTINUE                                         HYBJ2280
XNORM = ENORM(N,WA3)                                     HYBJ2290
DELTA = FACTOR*XNORM                                     HYBJ2300
IF (DELTA .EQ. ZERO) DELTA = FACTOR                     HYBJ2310
70      CONTINUE                                         HYBJ2320
C
C      FORM (Q TRANSPOSE)*FVEC AND STORE IN QTF.           HYBJ2330
C
C      DO 80 I = 1, N                                         HYBJ2340
C          QTF(I) = FVEC(I)                                 HYBJ2350
80      CONTINUE                                         HYBJ2360
DO 120 J = 1, N                                         HYBJ2370
    IF (FJAC(J,J) .EQ. ZERO) GO TO 110                  HYBJ2380
    SUM = ZERO                                         HYBJ2390
    DO 90 I = J, N                                     HYBJ2400
        SUM = SUM + FJAC(I,J)*QTF(I)                   HYBJ2410
90      CONTINUE                                         HYBJ2420
    TEMP = -SUM/FJAC(J,J)                           HYBJ2430
    DO 100 I = J, N                                    HYBJ2440
        QTF(I) = QTF(I) + FJAC(I,J)*TEMP             HYBJ2450
100     CONTINUE                                         HYBJ2460
110     CONTINUE                                         HYBJ2470
120     CONTINUE                                         HYBJ2480
C
C      COPY THE TRIANGULAR FACTOR OF THE QR FACTORIZATION INTO R.  HYBJ2490
C
SING = .FALSE.                                         HYBJ2500
DO 150 J = 1, N                                         HYBJ2510
    L = J                                              HYBJ2520
    JM1 = J - 1                                         HYBJ2530
    IF (JM1 .LT. 1) GO TO 140                         HYBJ2540
    DO 130 I = 1, JM1                                  HYBJ2550
        R(L) = FJAC(I,J)                               HYBJ2560
        L = L + N - I                                HYBJ2570
130      CONTINUE                                         HYBJ2580
140      CONTINUE                                         HYBJ2590
        R(L) = WA1(J)                                HYBJ2600
        IF (WA1(J) .EQ. ZERO) SING = .TRUE.            HYBJ2610
150      CONTINUE                                         HYBJ2620
C
C      ACCUMULATE THE ORTHOGONAL FACTOR IN FJAC.          HYBJ2630

```

```

C          CALL QFORM(N,N,FJAC,LDFJAC,WA1)          HYBJ2710
C          RESCALE IF NECESSARY.                  HYBJ2720
C
C          IF (MODE .EQ. 2) GO TO 170            HYBJ2730
DO 160 J = 1, N
    DIAG(J) = DMAX1(DIAG(J),WA2(J))          HYBJ2740
160    CONTINUE                                HYBJ2750
170    CONTINUE                                HYBJ2760
C          BEGINNING OF THE INNER LOOP.        HYBJ2770
C
C          180    CONTINUE                            HYBJ2780
C
C          IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES. HYBJ2790
C
C          IF (NPRINT .LE. 0) GO TO 190          HYBJ2800
IFLAG = 0
    IF (MOD(ITER-1,NPRINT) .EQ. 0)          HYBJ2810
        CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG) HYBJ2820
    IF (IFLAG .LT. 0) GO TO 300            HYBJ2830
190    CONTINUE                                HYBJ2840
C
C          DETERMINE THE DIRECTION P.          HYBJ2850
C
C          CALL DOGLEG(N,R,LR,DIAG,QTF,DELTA,WA1,WA2,WA3) HYBJ2860
C
C          STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P. HYBJ2870
C
C          DO 200 J = 1, N                      HYBJ2940
    WA1(J) = -WA1(J)                        HYBJ2950
    WA2(J) = X(J) + WA1(J)                  HYBJ2960
    WA3(J) = DIAG(J)*WA1(J)                HYBJ2970
200    CONTINUE                                HYBJ2980
C          PNORM = ENORM(N,WA3)                HYBJ2990
C
C          ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND. HYBJ3000
C
C          IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM) HYBJ3010
C
C          EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM. HYBJ3020
C
C          IFLAG = 1                            HYBJ3030
CALL FCN(N,WA2,WA4,FJAC,LDFJAC,IFLAG) HYBJ3040
NFEV = NFEV + 1                         HYBJ3050
    IF (IFLAG .LT. 0) GO TO 300            HYBJ3060
FNORM1 = ENORM(N,WA4)                   HYBJ3070
C
C          COMPUTE THE SCALED ACTUAL REDUCTION. HYBJ3080
C
C          ACTRED = -ONE                      HYBJ3090
IF (FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2 HYBJ3100
C
C          HYBJ3110
C          HYBJ3120
C          HYBJ3130
C          HYBJ3140
C          HYBJ3150
C          HYBJ3160
C          HYBJ3170
C          HYBJ3180
C          HYBJ3190
C          HYBJ3200
C          HYBJ3210
C          HYBJ3220
C          HYBJ3230
C          HYBJ3240

```

```

C COMPUTE THE SCALED PREDICTED REDUCTION.          HYBJ3250
C
C L = 1                                         HYBJ3260
DO 220 I = 1, N                               HYBJ3270
      SUM = ZERO                                HYBJ3280
      DO 210 J = I, N                           HYBJ3290
          SUM = SUM + R(L)*WA1(J)                HYBJ3300
          L = L + 1                             HYBJ3310
210      CONTINUE                                HYBJ3320
          WA3(I) = QTF(I) + SUM                  HYBJ3330
220      CONTINUE                                HYBJ3340
          TEMP = ENORM(N,WA3)                   HYBJ3350
          PRERED = ZERO                         HYBJ3360
          IF (TEMP .LT. FNORM) PRERED = ONE - (TEMP/FNORM)**2 HYBJ3370
C
C COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED HYBJ3380
C REDUCTION.                                     HYBJ3390
C
C RATIO = ZERO                                 HYBJ3400
IF (PRERED .GT. ZERO) RATIO = ACTRED/PRERED    HYBJ3410
C
C UPDATE THE STEP BOUND.                      HYBJ3420
C
C IF (RATIO .GE. P1) GO TO 230                HYBJ3430
      NCSUC = 0                                HYBJ3440
      NCFAIL = NCFAIL + 1                     HYBJ3450
      DELTA = P5*DELTA                         HYBJ3460
      GO TO 240                                HYBJ3470
230      CONTINUE                                HYBJ3480
      NCFAIL = 0                                HYBJ3490
      NCSUC = NCSUC + 1                        HYBJ3500
      IF (RATIO .GE. P5 .OR. NCSUC .GT. 1)     HYBJ3510
      *      DELTA = DMAX1(DELTA,PNORM/P5)       HYBJ3520
      IF (DABS(RATIO-ONE) .LE. P1) DELTA = PNORM/P5 HYBJ3530
240      CONTINUE                                HYBJ3540
C
C TEST FOR SUCCESSFUL ITERATION.               HYBJ3550
C
C IF (RATIO .LT. P0001) GO TO 260             HYBJ3560
C
C SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS. HYBJ3570
C
C DO 250 J = 1, N                            HYBJ3580
      X(J) = WA2(J)                          HYBJ3590
      WA2(J) = DIAG(J)*X(J)                  HYBJ3600
      FVEC(J) = WA4(J)                      HYBJ3610
250      CONTINUE                                HYBJ3620
      XNORM = ENORM(N,WA2)                  HYBJ3630
      FNORM = FNORM1                         HYBJ3640
      ITER = ITER + 1                      HYBJ3650
260      CONTINUE                                HYBJ3660
C
C DETERMINE THE PROGRESS OF THE ITERATION.    HYBJ3670
C
C

```

```

NSLOW1 = NSLOW1 + 1                                HYBJ3790
IF (ACTRED .GE. P001) NSLOW1 = 0                  HYBJ3800
IF (JEVAL) NSLOW2 = NSLOW2 + 1                  HYBJ3810
IF (ACTRED .GE. P1) NSLOW2 = 0                  HYBJ3820
C                                               HYBJ3830
C                                               HYBJ3840
C                                               HYBJ3850
TEST FOR CONVERGENCE.
C                                               HYBJ3860
C                                               HYBJ3870
C                                               HYBJ3880
TESTS FOR TERMINATION AND STRINGENT TOLERANCES.
C                                               HYBJ3890
C                                               HYBJ3900
IF (NFEV .GE. MAXFEV) INFO = 2                HYBJ3910
IF (P1*DMAX1(P1*DELTA,PNORM) .LE. EPSMCH*XNORM) INFO = 3 HYBJ3920
IF (NSLOW2 .EQ. 5) INFO = 4                  HYBJ3930
IF (NSLOW1 .EQ. 10) INFO = 5                 HYBJ3940
IF (INFO .NE. 0) GO TO 300                  HYBJ3950
C                                               HYBJ3960
C                                               HYBJ3970
C                                               HYBJ3980
CRITERION FOR RECALCULATING JACOBIAN.
C                                               HYBJ3990
C                                               HYBJ4000
C                                               HYBJ4010
C                                               HYBJ4020
C                                               HYBJ4030
CALCULATE THE RANK ONE MODIFICATION TO THE JACOBIAN
AND UPDATE QTF IF NECESSARY.
C                                               HYBJ4040
C                                               HYBJ4050
DO 280 J = 1, N
    SUM = ZERO
    DO 270 I = 1, N
        SUM = SUM + FJAC(I,J)*WA4(I)
        CONTINUE
        WA2(J) = (SUM - WA3(J))/PNORM
        WA1(J) = DIAG(J)*((DIAG(J)*WA1(J))/PNORM)
        IF (RATIO .GE. P0001) QTF(J) = SUM
    CONTINUE
270
280
C                                               HYBJ4060
C                                               HYBJ4070
C                                               HYBJ4080
C                                               HYBJ4090
C                                               HYBJ4100
C                                               HYBJ4110
C                                               HYBJ4120
C                                               HYBJ4130
COMPUTE THE QR FACTORIZATION OF THE UPDATED JACOBIAN.
C                                               HYBJ4140
C                                               HYBJ4150
CALL R1UPDT(N,N,R,LR,WA1,WA2,WA3,SING)          HYBJ4160
CALL R1MPYQ(N,N,FJAC,LDFJAC,WA2,WA3)            HYBJ4170
CALL R1MPYQ(1,N,QTF,1,WA2,WA3)                  HYBJ4180
C                                               HYBJ4190
C                                               HYBJ4200
C                                               HYBJ4210
END OF THE INNER LOOP.
C                                               HYBJ4220
C                                               HYBJ4230
JEVAL = .FALSE.
GO TO 180
290
CONTINUE
C                                               HYBJ4240
C                                               HYBJ4250
C                                               HYBJ4260
C                                               HYBJ4270
END OF THE OUTER LOOP.
C                                               HYBJ4280
C                                               HYBJ4290
C                                               HYBJ4300
C                                               HYBJ4310
C                                               HYBJ4320
GO TO 30
300
CONTINUE
TERMINATION, EITHER NORMAL OR USER IMPOSED.

```

```
IF (IFLAG .LT. 0) INFO = IFLAG          HYBJ4330
IFLAG = 0                                HYBJ4340
IF (NPRINT .GT. 0) CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)  HYBJ4350
RETURN                                    HYBJ4360
                                         HYBJ4370
C                                         HYBJ4380
C                                         HYBJ4390
C                                         HYBJ4400
END
```

THIS PAGE
WAS THIS PAGE ALONE
WAS INTENTIONALLY
LEFT BLANK

```

SUBROUTINE HYBRJ1(FCN,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,WA,LWA)          HYJ10010
C INTEGER N,LDFJAC,INFO,LWA                                         HYJ10020
C DOUBLE PRECISION TOL                                         HYJ10030
C DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N),WA(LWA)           HYJ10040
C EXTERNAL FCN                                                 HYJ10050
C *****
C SUBROUTINE HYBRJ1                                         HYJ10060
C
C THE PURPOSE OF HYBRJ1 IS TO FIND A ZERO OF A SYSTEM OF             HYJ10070
C N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION            HYJ10080
C OF THE POWELL HYBRID METHOD. THIS IS DONE BY USING THE             HYJ10090
C MORE GENERAL NONLINEAR EQUATION SOLVER HYBRJ. THE USER            HYJ10100
C MUST PROVIDE A SUBROUTINE WHICH CALCULATES THE FUNCTIONS          HYJ10110
C AND THE JACOBIAN.                                              HYJ10120
C
C THE SUBROUTINE STATEMENT IS                                     HYJ10130
C
C SUBROUTINE HYBRJ1(FCN,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,WA,LWA)          HYJ10140
C
C WHERE                                                 HYJ10150
C
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH           HYJ10160
C CALCULATES THE FUNCTIONS AND THE JACOBIAN. FCN MUST             HYJ10170
C BE DECLARED IN AN EXTERNAL STATEMENT IN THE USER               HYJ10180
C CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.           HYJ10190
C
C SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)                   HYJ10200
C INTEGER N,LDFJAC,IFLAG                                         HYJ10210
C DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)                 HYJ10220
C -----
C IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND                  HYJ10230
C RETURN THIS VECTOR IN FVEC. DO NOT ALTER FJAC.                HYJ10240
C IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND                 HYJ10250
C RETURN THIS MATRIX IN FJAC. DO NOT ALTER FVEC.              HYJ10260
C -----
C RETURN                                               HYJ10270
C END                                                 HYJ10280
C
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS        HYJ10290
C THE USER WANTS TO TERMINATE EXECUTION OF HYBRJ1.             HYJ10300
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.          HYJ10310
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER       HYJ10320
C OF FUNCTIONS AND VARIABLES.                                HYJ10330
C
C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN           HYJ10340
C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X      HYJ10350
C CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.        HYJ10360
C
C FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS          HYJ10370
C THE FUNCTIONS EVALUATED AT THE OUTPUT X.                    HYJ10380
C
C FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE          HYJ10390

```

```

C ORTHOGONAL MATRIX Q PRODUCED BY THE QR FACTORIZATION          HYJ10550
C OF THE FINAL APPROXIMATE JACOBIAN.                         HYJ10560
C
C LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N      HYJ10570
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.     HYJ10580
C
C TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS        HYJ10600
C WHEN THE ALGORITHM ESTIMATES THAT THE RELATIVE ERROR         HYJ10610
C BETWEEN X AND THE SOLUTION IS AT MOST TOL.                  HYJ10620
C
C INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS           HYJ10630
C TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)            HYJ10640
C VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,           HYJ10650
C INFO IS SET AS FOLLOWS.                                     HYJ10660
C
C     INFO = 0    IMPROPER INPUT PARAMETERS.                   HYJ10670
C
C     INFO = 1    ALGORITHM ESTIMATES THAT THE RELATIVE ERROR   HYJ10680
C                  BETWEEN X AND THE SOLUTION IS AT MOST TOL.     HYJ10690
C
C     INFO = 2    NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS       HYJ10700
C                  REACHED 100*(N+1).                           HYJ10710
C
C     INFO = 3    TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN   HYJ10720
C                  THE APPROXIMATE SOLUTION X IS POSSIBLE.        HYJ10730
C
C     INFO = 4    ITERATION IS NOT MAKING GOOD PROGRESS.        HYJ10740
C
C WA IS A WORK ARRAY OF LENGTH LWA.                            HYJ10750
C
C LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN        HYJ10760
C (N*(N+13))/2.                                              HYJ10770
C
C SUBPROGRAMS CALLED                                         HYJ10780
C
C     USER-SUPPLIED ..... FCN                                HYJ10790
C
C     MINPACK-SUPPLIED ... HYBRJ                            HYJ10800
C
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.    HYJ10810
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE      HYJ10820
C
C *****
C INTEGER J,LR,MAXFEV,MODE,NFEV,NJEV,NPRINT                HYJ10830
C DOUBLE PRECISION FACTOR,ONE,XTOL,ZERO                   HYJ10840
C DATA FACTOR,ONE,ZERO /1.0D2,1.0D0,0.0D0/                 HYJ10850
C INFO = 0                                                 HYJ10860
C
C CHECK THE INPUT PARAMETERS FOR ERRORS.                  HYJ10870
C
C IF (N .LE. 0 .OR. LDFJAC .LT. N .OR. TOL .LT. ZERO      HYJ10880
C *      .OR. LWA .LT. (N*(N + 13))/2) GO TO 20             HYJ10890
C
C CALL HYBRJ.                                            HYJ10900

```

```

C                               HYJ11090
MAXFEV = 100*(N + 1)          HYJ11100
XTOL = TOL                     HYJ11110
MODE = 2                        HYJ11120
DO 10 J = 1, N                 HYJ11130
      WA(J) = ONE                HYJ11140
10   CONTINUE                   HYJ11150
      NPRINT = 0                 HYJ11160
      LR = (N*(N + 1))/2          HYJ11170
      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))    HYJ11180
*                         HYJ11190
*                         HYJ11200
      IF (INFO .EQ. 5) INFO = 4    HYJ11210
20   CONTINUE                   HYJ11220
      RETURN                      HYJ11230
C                               HYJ11240
C                               LAST CARD OF SUBROUTINE HYBRJ1.    HYJ11250
C                               HYJ11260
C                               END                           HYJ11270

```

THIS PAGE
WAS THIS PAGE ALREADY
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 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 FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS LMDR0560
C THE FUNCTIONS EVALUATED AT THE OUTPUT X. LMDR0570
C 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
$$P^T * (JAC^T * JAC) * P = R^T * R,$$
 LMDR0640
C 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 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. LMDR1090
 C LMDR1100
 C LMDR1110
 C LMDR1120
 C LMDR1130
 C 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. LMDR1140
 C LMDR1150
 C LMDR1160
 C LMDR1170
 C LMDR1180
 C LMDR1190
 C LMDR1200
 C LMDR1210
 C LMDR1220
 C 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. LMDR1230
 C LMDR1240
 C LMDR1250
 C LMDR1260
 C LMDR1270
 C INFO = 0 IMPROPER INPUT PARAMETERS. LMDR1280
 C LMDR1290
 C INFO = 1 BOTH ACTUAL AND PREDICTED RELATIVE REDUCTIONS IN THE SUM OF SQUARES ARE AT MOST FTOL. LMDR1300
 C LMDR1310
 C LMDR1320
 C INFO = 2 RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES IS AT MOST XTOL. LMDR1330
 C LMDR1340
 C LMDR1350
 C INFO = 3 CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD. LMDR1360
 C LMDR1370
 C INFO = 4 THE COSINE OF THE ANGLE BETWEEN FVEC AND ANY COLUMN OF THE JACOBIAN IS AT MOST GTOL IN ABSOLUTE VALUE. LMDR1380
 C LMDR1390
 C LMDR1400
 C LMDR1410
 C INFO = 5 NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS REACHED MAXFEV. LMDR1420
 C LMDR1430
 C LMDR1440
 C INFO = 6 FTOL IS TOO SMALL. NO FURTHER REDUCTION IN THE SUM OF SQUARES IS POSSIBLE. LMDR1450
 C LMDR1460
 C LMDR1470
 C INFO = 7 XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN THE APPROXIMATE SOLUTION X IS POSSIBLE. LMDR1480
 C LMDR1490
 C LMDR1500
 C INFO = 8 GTOL IS TOO SMALL. FVEC IS ORTHOGONAL TO THE COLUMNS OF THE JACOBIAN TO MACHINE PRECISION. LMDR1510
 C LMDR1520
 C LMDR1530
 C NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF CALLS TO FCN WITH IFLAG = 1. LMDR1540
 C LMDR1550
 C LMDR1560
 C NJEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF CALLS TO FCN WITH IFLAG = 2. LMDR1570
 C LMDR1580
 C LMDR1590
 C 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 LMDR1600
 C LMDR1610
 C LMDR1620

```

C ORTHOGONAL (NOT STORED), AND R IS UPPER TRIANGULAR      LMDR1630
C WITH DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE.      LMDR1640
C COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.  LMDR1650
C
C QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS      LMDR1660
C THE FIRST N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*FVEC.  LMDR1670
C
C WA1, WA2, AND WA3 ARE WORK ARRAYS OF LENGTH N.          LMDR1680
C
C WA4 IS A WORK ARRAY OF LENGTH M.                         LMDR1690
C
C SUBPROGRAMS CALLED                                     LMDR1700
C
C USER-SUPPLIED ..... FCN                               LMDR1710
C
C MINPACK-SUPPLIED ... DPMPAR,ENORM,LMPAR,QRFAC        LMDR1720
C
C FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,DSQRT,MOD       LMDR1730
C
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. LMDR1740
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE   LMDR1750
C
C *****
C INTEGER I,IFLAG,ITER,J,L                                LMDR1760
C DOUBLE PRECISION ACTRED,DELTA,DIRDER,EPSMCH,FNORM,FNORM1,GNORM, LMDR1770
C *           ONE,PAR,PNORM,PRERED,P1,P5,P25,P75,P0001,RATIO, LMDR1780
C *           SUM,TEMP,TEMP1,TEMP2,XNORM,ZERO                LMDR1790
C DOUBLE PRECISION DPMPAR,ENORM                          LMDR1800
C DATA ONE,P1,P5,P25,P75,P0001,ZERO                  LMDR1810
C *           /1.0D0,1.0D-1,5.0D-1,2.5D-1,7.5D-1,1.0D-4,0.0D0/ LMDR1820
C
C EPSMCH IS THE MACHINE PRECISION.                      LMDR1830
C
C EPSMCH = DPMPAR(1)                                    LMDR1840
C
C INFO = 0                                              LMDR1850
C IFLAG = 0                                             LMDR1860
C NFEV = 0                                              LMDR1870
C NJEV = 0                                              LMDR1880
C
C CHECK THE INPUT PARAMETERS FOR ERRORS.               LMDR1890
C
C IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. M      LMDR1900
C *           .OR. FTOL .LT. ZERO .OR. XTOL .LT. ZERO .OR. GTOL .LT. ZERO LMDR1910
C *           .OR. MAXFEV .LE. 0 .OR. FACTOR .LE. ZERO) GO TO 300 LMDR1920
C IF (MODE .NE. 2) GO TO 20                            LMDR1930
C DO 10 J = 1, N                                       LMDR1940
C     IF (DIAG(J) .LE. ZERO) GO TO 300                 LMDR1950
C 10 CONTINUE                                         LMDR1960
C 20 CONTINUE                                         LMDR1970
C
C EVALUATE THE FUNCTION AT THE STARTING POINT        LMDR1980
C AND CALCULATE ITS NORM.                           LMDR1990
C
C

```

```

IFLAG = 1                                LMDR2170
CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)   LMDR2180
NFEV = 1                                 LMDR2190
IF (IFLAG .LT. 0) GO TO 300               LMDR2200
FNORM = ENORM(M,FVEC)                   LMDR2210
C                                         LMDR2220
C   INITIALIZE LEVENBERG-MARQUARDT PARAMETER AND ITERATION COUNTER. LMDR2230
C                                         LMDR2240
C   PAR = ZERO                            LMDR2250
C   ITER = 1                             LMDR2260
C                                         LMDR2270
C   BEGINNING OF THE OUTER LOOP.          LMDR2280
C                                         LMDR2290
C   30 CONTINUE                           LMDR2300
C                                         LMDR2310
C   CALCULATE THE JACOBIAN MATRIX.        LMDR2320
C                                         LMDR2330
C   IFLAG = 2                                LMDR2340
C   CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG) LMDR2350
C   NJEV = NJEV + 1                         LMDR2360
C   IF (IFLAG .LT. 0) GO TO 300             LMDR2370
C                                         LMDR2380
C   IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES. LMDR2390
C                                         LMDR2400
C   IF (NPRINT .LE. 0) GO TO 40            LMDR2410
C   IFLAG = 0                               LMDR2420
C   IF (MOD(ITER-1,NPRINT) .EQ. 0)          LMDR2430
*    CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG) LMDR2440
C   IF (IFLAG .LT. 0) GO TO 300             LMDR2450
C   40 CONTINUE                           LMDR2460
C                                         LMDR2470
C   COMPUTE THE QR FACTORIZATION OF THE JACOBIAN. LMDR2480
C                                         LMDR2490
C   CALL QRFAC(M,N,FJAC,LDFJAC,.TRUE.,IPVT,N,WA1,WA2,WA3) LMDR2500
C                                         LMDR2510
C   ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING LMDR2520
C   TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN. LMDR2530
C                                         LMDR2540
C   IF (ITER .NE. 1) GO TO 80              LMDR2550
C   IF (MODE .EQ. 2) GO TO 60              LMDR2560
C   DO 50 J = 1, N                        LMDR2570
C     DIAG(J) = WA2(J)                   LMDR2580
C     IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE LMDR2590
C   50 CONTINUE                           LMDR2600
C   60 CONTINUE                           LMDR2610
C                                         LMDR2620
C   ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X LMDR2630
C   AND INITIALIZE THE STEP BOUND DELTA.    LMDR2640
C                                         LMDR2650
C   DO 70 J = 1, N                        LMDR2660
C     WA3(J) = DIAG(J)*X(J)              LMDR2670
C   70 CONTINUE                           LMDR2680
C   XNORM = ENORM(N,WA3)                 LMDR2690
C   DELTA = FACTOR*XNORM                 LMDR2700

```

```

      IF (DELTA .EQ. ZERO) DELTA = FACTOR          LMDR2710
80    CONTINUE                                     LMDR2720
C
C      FORM (Q TRANSPOSE)*FVEC AND STORE THE FIRST N COMPONENTS IN   LMDR2730
C      QTF.                                         LMDR2740
C
C      DO 90 I = 1, M                           LMDR2750
         WA4(I) = FVEC(I)                      LMDR2760
90    CONTINUE                                     LMDR2770
      DO 130 J = 1, N                         LMDR2780
         IF (FJAC(J,J) .EQ. ZERO) GO TO 120     LMDR2790
         SUM = ZERO                            LMDR2800
         DO 100 I = J, M                      LMDR2810
            SUM = SUM + FJAC(I,J)*WA4(I)       LMDR2820
100   CONTINUE                                     LMDR2830
         TEMP = -SUM/FJAC(J,J)                 LMDR2840
         DO 110 I = J, M                      LMDR2850
            WA4(I) = WA4(I) + FJAC(I,J)*TEMP   LMDR2860
110   CONTINUE                                     LMDR2870
120   CONTINUE                                     LMDR2880
         FJAC(J,J) = WA1(J)                  LMDR2890
         QTF(J) = WA4(J)                     LMDR2900
130   CONTINUE                                     LMDR2910
C
C      COMPUTE THE NORM OF THE SCALED GRADIENT.        LMDR2920
C
C      GNORM = ZERO                                LMDR2930
         IF (FNORM .EQ. ZERO) GO TO 170          LMDR2940
         DO 160 J = 1, N                          LMDR2950
            L = IPVT(J)                         LMDR2960
            IF (WA2(L) .EQ. ZERO) GO TO 150        LMDR2970
            SUM = ZERO                           LMDR2980
            DO 140 I = 1, J                      LMDR2990
               SUM = SUM + FJAC(I,J)*(QTF(I)/FNORM)  LMDR3000
140   CONTINUE                                     LMDR3010
         GNORM = DMAX1(GNORM,DABS(SUM/WA2(L)))  LMDR3020
150   CONTINUE                                     LMDR3030
160   CONTINUE                                     LMDR3040
170   CONTINUE                                     LMDR3050
C
C      TEST FOR CONVERGENCE OF THE CRADIENT NORM.      LMDR3060
C
C      IF (GNORM .LE. GTOL) INFO = 4                LMDR3070
         IF (INFO .NE. 0) GO TO 300                LMDR3080
C
C      RESCALE IF NECESSARY.                      LMDR3090
C
         IF (MODE .EQ. 2) GO TO 190                LMDR3100
         DO 180 J = 1, N                          LMDR3110
            DIAG(J) = DMAX1(DIAG(J),WA2(J))       LMDR3120
180   CONTINUE                                     LMDR3130
190   CONTINUE                                     LMDR3140
C
C      BEGINNING OF THE INNER LOOP.                LMDR3150

```

```

C      200    CONTINUE
C
C          DETERMINE THE LEVENBERG-MARQUARDT PARAMETER.
C
C          CALL LMPAR(N,FJAC,LDFJAC,IPVT,DIAG,QTF,DELTA,PAR,WA1,WA2,
C          *                  WA3,WA4)
C
C          STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.
C
C          DO 210 J = 1, N
C                 WA1(J) = -WA1(J)
C                 WA2(J) = X(J) + WA1(J)
C                 WA3(J) = DIAG(J)*WA1(J)
C
C          210    CONTINUE
C          PNORM = ENORM(N,WA3)
C
C          ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.
C
C          IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)
C
C          EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.
C
C          IFLAG = 1
C          CALL FCN(M,N,WA2,WA4,FJAC,LDFJAC,IFLAG)
C          NFEV = NFEV + 1
C          IF (IFLAG .LT. 0) GO TO 300
C          FNORM1 = ENORM(M,WA4)
C
C          COMPUTE THE SCALED ACTUAL REDUCTION.
C
C          ACTRED = -ONE
C          IF (P1*FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2
C
C          COMPUTE THE SCALED PREDICTED REDUCTION AND
C          THE SCALED DIRECTIONAL DERIVATIVE.
C
C          DO 230 J = 1, N
C                 WA3(J) = ZERO
C                 L = IPVT(J)
C                 TEMP = WA1(L)
C                 DO 220 I = 1, J
C                         WA3(I) = WA3(I) + FJAC(I,J)*TEMP
C
C          220    CONTINUE
C
C          230    CONTINUE
C          TEMP1 = ENORM(N,WA3)/FNORM
C          TEMP2 = (DSQRT(PAR)*PNORM)/FNORM
C          PRERED = TEMP1**2 + TEMP2**2/P5
C          DIRDER = -(TEMP1**2 + TEMP2**2)
C
C          COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED
C          REDUCTION.
C
C          RATIO = ZERO

```

```

C       IF (PRERED .NE. ZERO) RATIO = ACTRED/PRERED          LMDR3790
C
C       UPDATE THE STEP BOUND.                                LMDR3800
C
C       IF (RATIO .GT. P25) GO TO 240                         LMDR3810
C           IF (ACTRED .GE. ZERO) TEMP = P5                  LMDR3820
C           IF (ACTRED .LT. ZERO)                           LMDR3830
*             TEMP = P5*DIRDER/(DIRDER + P5*ACTRED)        LMDR3840
C           IF (P1*FNORM1 .GE. FNORM .OR. TEMP .LT. P1) TEMP = P1  LMDR3850
C           DELTA = TEMP*DMIN1(DELTA,PNORM/P1)              LMDR3860
C           PAR = PAR/TEMP                                  LMDR3870
C           GO TO 260                                     LMDR3880
240     CONTINUE                                              LMDR3890
C           IF (PAR .NE. ZERO .AND. RATIO .LT. P75) GO TO 250  LMDR3900
C           DELTA = PNORM/P5                               LMDR3910
C           PAR = P5*PAR                                 LMDR3920
250     CONTINUE                                              LMDR3930
260     CONTINUE                                              LMDR3940
C
C       TEST FOR SUCCESSFUL ITERATION.                      LMDR3950
C
C       IF (RATIO .LT. P0001) GO TO 290                  LMDR3960
C
C       SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS. LMDR3970
C
C       DO 270 J = 1, N                                    LMDR3980
C           X(J) = WA2(J)                                LMDR3990
C           WA2(J) = DIAG(J)*X(J)                          LMDR4000
270     CONTINUE                                              LMDR4010
C       DO 280 I = 1, M                                    LMDR4020
C           FVEC(I) = WA4(I)                                LMDR4030
280     CONTINUE                                              LMDR4040
C           XNORM = ENORM(N,WA2)                            LMDR4050
C           FNORM = FNORM1                               LMDR4060
C           ITER = ITER + 1                                LMDR4070
290     CONTINUE                                              LMDR4080
C
C       TESTS FOR CONVERGENCE.                            LMDR4090
C
C       IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL    LMDR4100
*         .AND. P5*RATIO .LE. ONE) INFO = 1               LMDR4110
C       IF (DELTA .LE. XTOL*XNORM) INFO = 2            LMDR4120
C       IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL    LMDR4130
*         .AND. P5*RATIO .LE. ONE .AND. INFO .EQ. 2) INFO = 3  LMDR4140
C       IF (INFO .NE. 0) GO TO 300                     LMDR4150
C
C       TESTS FOR TERMINATION AND STRINGENT TOLERANCES.   LMDR4160
C
C       IF (NFEV .GE. MAXFEV) INFO = 5                  LMDR4170
C       IF (DABS(ACTRED) .LE. EPSMCH .AND. PRERED .LE. EPSMCH  LMDR4180
*         .AND. P5*RATIO .LE. ONE) INFO = 6            LMDR4190
C       IF (DELTA .LE. EPSMCH*XNORM) INFO = 7          LMDR4200
C       IF (GNORM .LE. EPSMCH) INFO = 8                LMDR4210
C       IF (INFO .NE. 0) GO TO 300                     LMDR4220
C
C       IF (NFEV .GE. MAXFEV) INFO = 5                  LMDR4230
C       IF (DABS(ACTRED) .LE. EPSMCH .AND. PRERED .LE. EPSMCH  LMDR4240
*         .AND. P5*RATIO .LE. ONE) INFO = 6            LMDR4250
C       IF (DELTA .LE. EPSMCH*XNORM) INFO = 7          LMDR4260
C       IF (GNORM .LE. EPSMCH) INFO = 8                LMDR4270
C       IF (INFO .NE. 0) GO TO 300                     LMDR4280

```

C END OF THE INNER LOOP. REPEAT IF ITERATION UNSUCCESSFUL. LMDR4330
C LMDR4340
C LMDR4350
C LMDR4360
C LMDR4370
C LMDR4380
C LMDR4390
C LMDR4400
C LMDR4410
C LMDR4420
C LMDR4430
C LMDR4440
C LMDR4450
C LMDR4460
C LMDR4470
C LMDR4480
C LMDR4490
C LMDR4500
C LMDR4510
C LMDR4520
C IF (RATIO .LT. P0001) GO TO 200
C END OF THE OUTER LOOP.
C GO TO 30
300 CONTINUE
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 LAST CARD OF SUBROUTINE LMDER.
C END

THIS PAGE
WAS THIS PAGE ALONE
WAS INTENTIONALLY
LEFT BLANK

```

SUBROUTINE LMDER1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,IPVT,WA,      LMR10010
*          LWA)                                              LMR10020
INTEGER M,N,LDFJAC,INFO,LWA                                         LMR10030
INTEGER IPVT(N)                                              LMR10040
DOUBLE PRECISION TOL                                         LMR10050
DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),WA(LWA)           LMR10060
EXTERNAL FCN                                              LMR10070
*****                                                       LMR10080
C                                                       LMR10090
C SUBROUTINE LMDER1                                         LMR10100
C                                                       LMR10110
C THE PURPOSE OF LMDER1 IS TO MINIMIZE THE SUM OF THE SQUARES OF   LMR10120
C M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF THE   LMR10130
C LEVENBERG-MARQUARDT ALGORITHM. THIS IS DONE BY USING THE MORE    LMR10140
C GENERAL LEAST-SQUARES SOLVER LMDER. THE USER MUST PROVIDE A     LMR10150
C SUBROUTINE WHICH CALCULATES THE FUNCTIONS AND THE JACOBIAN.       LMR10160
C                                                       LMR10170
C THE SUBROUTINE STATEMENT IS                                     LMR10180
C                                                       LMR10190
C SUBROUTINE LMDER1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,      LMR10200
C                   IPVT,WA,LWA)                                         LMR10210
C                                                       LMR10220
C WHERE                                                       LMR10230
C                                                       LMR10240
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH            LMR10250
C CALCULATES THE FUNCTIONS AND THE JACOBIAN. FCN MUST             LMR10260
C BE DECLARED IN AN EXTERNAL STATEMENT IN THE USER                LMR10270
C CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.             LMR10280
C                                                       LMR10290
C SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)                 LMR10300
C INTEGER M,N,LDFJAC,IFLAG                                         LMR10310
C DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)                  LMR10320
C -----
C IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND                  LMR10340
C RETURN THIS VECTOR IN FVEC. DO NOT ALTER FJAC.                  LMR10350
C IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND                  LMR10360
C RETURN THIS MATRIX IN FJAC. DO NOT ALTER FVEC.                  LMR10370
C -----
C RETURN                                              LMR10380
C END                                                 LMR10390
C                                                       LMR10400
C                                                       LMR10410
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS        LMR10420
C THE USER WANTS TO TERMINATE EXECUTION OF LMDER1.              LMR10430
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.                 LMR10440
C                                                       LMR10450
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER      LMR10460
C OF FUNCTIONS.                                              LMR10470
C                                                       LMR10480
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER      LMR10490
C OF VARIABLES. N MUST NOT EXCEED M.                           LMR10500
C                                                       LMR10510
C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN           LMR10520
C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X      LMR10530
C CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.          LMR10540

```

C FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS LMR10550
 C THE FUNCTIONS EVALUATED AT THE OUTPUT X. LMR10560
 C FJAC IS AN OUTPUT M BY N ARRAY. THE UPPER N BY N SUBMATRIX LMR10570
 C OF FJAC CONTAINS AN UPPER TRIANGULAR MATRIX R WITH LMR10580
 C DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE SUCH THAT LMR10590
 C C T T T
 C P *(JAC *JAC)*P = R *R, LMR10600
 C WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL LMR10610
 C CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J) LMR10620
 C (SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRAPEZOIDAL LMR10630
 C PART OF FJAC CONTAINS INFORMATION GENERATED DURING LMR10640
 C THE COMPUTATION OF R. LMR10650
 C LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M LMR10660
 C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC. LMR10670
 C TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS LMR10680
 C WHEN THE ALGORITHM ESTIMATES EITHER THAT THE RELATIVE LMR10690
 C ERROR IN THE SUM OF SQUARES IS AT MOST TOL OR THAT LMR10700
 C THE RELATIVE ERROR BETWEEN X AND THE SOLUTION IS AT LMR10710
 C MOST TOL. LMR10720
 C INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS LMR10730
 C TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE) LMR10740
 C VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE, LMR10750
 C INFO IS SET AS FOLLOWS. LMR10760
 C C INFO = 0 IMPROPER INPUT PARAMETERS. LMR10770
 C C INFO = 1 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR LMR10780
 C IN THE SUM OF SQUARES IS AT MOST TOL. LMR10790
 C C INFO = 2 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR LMR10800
 C BETWEEN X AND THE SOLUTION IS AT MOST TOL. LMR10810
 C C INFO = 3 CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD. LMR10820
 C C INFO = 4 FVEC IS ORTHOGONAL TO THE COLUMNS OF THE LMR10830
 C JACOBIAN TO MACHINE PRECISION. LMR10840
 C C INFO = 5 NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS LMR10850
 C REACHED 100*(N+1). LMR10860
 C C INFO = 6 TOL IS TOO SMALL. NO FURTHER REDUCTION IN LMR10870
 C THE SUM OF SQUARES IS POSSIBLE. LMR10880
 C C INFO = 7 TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN LMR10890
 C THE APPROXIMATE SOLUTION X IS POSSIBLE. LMR10900
 C C IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT LMR10910

```

C      DEFINES A PERMUTATION MATRIX P SUCH THAT JAC*P = Q*R,          LMR11090
C      WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS               LMR11100
C      ORTHOGONAL (NOT STORED), AND R IS UPPER TRIANGULAR            LMR11110
C      WITH DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE.           LMR11120
C      COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.       LMR11130
C                                         LMR11140
C      WA IS A WORK ARRAY OF LENGTH LWA.                            LMR11150
C                                         LMR11160
C      LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN 5*N+M. LMR11170
C                                         LMR11180
C      SUBPROGRAMS CALLED                                         LMR11190
C                                         LMR11200
C      USER-SUPPLIED ..... FCN                                 LMR11210
C                                         LMR11220
C      MINPACK-SUPPLIED ... LMDER                           LMR11230
C                                         LMR11240
C      ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.   LMR11250
C      BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE        LMR11260
C                                         LMR11270
C *****                         LMR11280
C      INTEGER MAXFEV,MODE,NFEV,NJEV,NPRINT                  LMR11290
C      DOUBLE PRECISION FACTOR,FTOL,GTOL,XTOL,ZERO           LMR11300
C      DATA FACTOR,ZERO /1.0D2,0.0D0/                         LMR11310
C      INFO = 0                                              LMR11320
C                                         LMR11330
C      CHECK THE INPUT PARAMETERS FOR ERRORS.                 LMR11340
C                                         LMR11350
C      IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. M .OR. TOL .LT. ZERO LMR11360
C      *     .OR. LWA .LT. 5*N + M) GO TO 10                   LMR11370
C                                         LMR11380
C      CALL LMDER.                                         LMR11390
C                                         LMR11400
C      MAXFEV = 100*(N + 1)                                LMR11410
C      FTOL = TOL                                         LMR11420
C      XTOL = TOL                                         LMR11430
C      GTOL = ZERO                                         LMR11440
C      MODE = 1                                           LMR11450
C      NPRINT = 0                                         LMR11460
C      CALL LMDER(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,MAXFEV, LMR11470
C      *           WA(1),MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,IPVT,WA(N+1), LMR11480
C      *           WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))          LMR11490
C      IF (INFO .EQ. 8) INFO = 4                           LMR11500
10    CONTINUE                                         LMR11510
      RETURN                                         LMR11520
C                                         LMR11530
C      LAST CARD OF SUBROUTINE LMDER1.                    LMR11540
C                                         LMR11550
      END                                         LMR11560

```

```

SUBROUTINE LMDIF(FCN,M,N,X,FVEC,FTOL,XTOL,GTOL,MAXFEV,EPSFCN,          LMDF0010
*           DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,          LMDF0020
*           IPVT,QTF,WA1,WA2,WA3,WA4)          LMDF0030
INTEGER M,N,MAXFEV,MODE,NPRINT,INFO,NFEV,LDFJAC          LMDF0040
INTEGER IPVT(N)          LMDF0050
DOUBLE PRECISION FTOL,XTOL,GTOL,EPSFCN,FACTOR          LMDF0060
DOUBLE PRECISION X(N),FVEC(M),DIAG(N),FJAC(LDFJAC,N),QTF(N),          LMDF0070
*           WA1(N),WA2(N),WA3(N),WA4(M)          LMDF0080
EXTERNAL FCN          LMDF0090
*****          LMDF0100
*****          LMDF0110
*****          LMDF0120
*****          LMDF0130
C SUBROUTINE LMDIF          LMDF0140
C
C THE PURPOSE OF LMDIF IS TO MINIMIZE THE SUM OF THE SQUARES OF          LMDF0150
C M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF          LMDF0160
C THE LEVENBERG-MARQUARDT ALGORITHM. THE USER MUST PROVIDE A          LMDF0170
C SUBROUTINE WHICH CALCULATES THE FUNCTIONS. THE JACOBIAN IS          LMDF0180
C THEN CALCULATED BY A FORWARD-DIFFERENCE APPROXIMATION.          LMDF0190
C
C THE SUBROUTINE STATEMENT IS          LMDF0200
C
C SUBROUTINE LMDIF(FCN,M,N,X,FVEC,FTOL,XTOL,GTOL,MAXFEV,EPSFCN,          LMDF0220
C           DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,          LMDF0230
C           LDFJAC,IPVT,QTF,WA1,WA2,WA3,WA4)          LMDF0240
C
C WHERE          LMDF0250
C
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH          LMDF0280
C CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED          LMDF0290
C IN AN EXTERNAL STATEMENT IN THE USER CALLING          LMDF0300
C PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.          LMDF0310
C
C SUBROUTINE FCN(M,N,X,FVEC,IFLAG)          LMDF0330
C INTEGER M,N,IFLAG          LMDF0340
C DOUBLE PRECISION X(N),FVEC(M)          LMDF0350
C -----
C CALCULATE THE FUNCTIONS AT X AND          LMDF0360
C RETURN THIS VECTOR IN FVEC.          LMDF0370
C -----
C RETURN          LMDF0380
C END          LMDF0390
C
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS          LMDF0430
C THE USER WANTS TO TERMINATE EXECUTION OF LMDIF.          LMDF0440
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.          LMDF0450
C
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER          LMDF0460
C OF FUNCTIONS.          LMDF0470
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER          LMDF0480
C OF VARIABLES. N MUST NOT EXCEED M.          LMDF0490
C
C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN          LMDF0500
C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X          LMDF0510
C

```

C	CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.	LMDF0550
C	FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS	LMDF0560
C	THE FUNCTIONS EVALUATED AT THE OUTPUT X.	LMDF0570
C	FTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	LMDF0580
C	OCCURS WHEN BOTH THE ACTUAL AND PREDICTED RELATIVE	LMDF0590
C	REDUCTIONS IN THE SUM OF SQUARES ARE AT MOST FTOL.	LMDF0600
C	THEREFORE, FTOL MEASURES THE RELATIVE ERROR DESIRED	LMDF0610
C	IN THE SUM OF SQUARES.	LMDF0620
C	XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	LMDF0630
C	OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE	LMDF0640
C	ITERATES IS AT MOST XTOL. THEREFORE, XTOL MEASURES THE	LMDF0650
C	RELATIVE ERROR DESIRED IN THE APPROXIMATE SOLUTION.	LMDF0660
C	GTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	LMDF0670
C	OCCURS WHEN THE COSINE OF THE ANGLE BETWEEN FVEC AND	LMDF0680
C	ANY COLUMN OF THE JACOBIAN IS AT MOST GTOL IN ABSOLUTE	LMDF0690
C	VALUE. THEREFORE, GTOL MEASURES THE ORTHOGONALITY	LMDF0700
C	DESIRED BETWEEN THE FUNCTION VECTOR AND THE COLUMNS	LMDF0710
C	OF THE JACOBIAN.	LMDF0720
C	MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION	LMDF0730
C	OCCURS WHEN THE NUMBER OF CALLS TO FCN IS AT LEAST	LMDF0740
C	MAXFEV BY THE END OF AN ITERATION.	LMDF0750
C	EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE	LMDF0760
C	STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS	LMDF0770
C	APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE	LMDF0780
C	FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS	LMDF0790
C	THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE	LMDF0800
C	ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE	LMDF0810
C	PRECISION.	LMDF0820
C	DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE	LMDF0830
C	BELLOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG	LMDF0840
C	MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS	LMDF0850
C	MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.	LMDF0860
C	MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE	LMDF0870
C	VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,	LMDF0880
C	THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER	LMDF0890
C	VALUES OF MODE ARE EQUIVALENT TO MODE = 1.	LMDF0900
C	FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE	LMDF0910
C	INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF	LMDF0920
C	FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE	LMDF0930
C	TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE	LMDF0940
C	INTERVAL (.1,100.). 100. IS A GENERALLY RECOMMENDED VALUE.	LMDF0950
C	NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED	LMDF0960
C	PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,	LMDF0970
C	FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST	LMDF0980

C ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND LMDF1090
C IMMEDIATELY PRIOR TO RETURN, WITH X AND FVEC AVAILABLE LMDF1100
C FOR PRINTING. IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS LMDF1110
C OF FCN WITH IFLAG = 0 ARE MADE. LMDF1120
C LMDF1130

C INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS LMDF1140
C TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE) LMDF1150
C VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE, LMDF1160
C INFO IS SET AS FOLLOWS. LMDF1170
C LMDF1180

C INFO = 0 IMPROPER INPUT PARAMETERS. LMDF1190

C INFO = 1 BOTH ACTUAL AND PREDICTED RELATIVE REDUCTIONS LMDF1210
C IN THE SUM OF SQUARES ARE AT MOST FTOL. LMDF1220

C INFO = 2 RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES LMDF1240
C IS AT MOST XTOL. LMDF1250
C LMDF1260

C INFO = 3 CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD. LMDF1270

C INFO = 4 THE COSINE OF THE ANGLE BETWEEN FVEC AND ANY LMDF1290
C COLUMN OF THE JACOBIAN IS AT MOST GTOL IN LMDF1300
C ABSOLUTE VALUE. LMDF1310
C LMDF1320

C INFO = 5 NUMBER OF CALLS TO FCN HAS REACHED OR LMDF1330
C EXCEEDED MAXFEV. LMDF1340

C INFO = 6 FTOL IS TOO SMALL. NO FURTHER REDUCTION IN LMDF1360
C THE SUM OF SQUARES IS POSSIBLE. LMDF1370
C LMDF1380

C INFO = 7 XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN LMDF1390
C THE APPROXIMATE SOLUTION X IS POSSIBLE. LMDF1400
C LMDF1410

C INFO = 8 GTOL IS TOO SMALL. FVEC IS ORTHOGONAL TO THE LMDF1420
C COLUMNS OF THE JACOBIAN TO MACHINE PRECISION. LMDF1430
C LMDF1440

C NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF LMDF1450
C CALLS TO FCN. LMDF1460
C LMDF1470

C FJAC IS AN OUTPUT M BY N ARRAY. THE UPPER N BY N SUBMATRIX LMDF1480
C OF FJAC CONTAINS AN UPPER TRIANGULAR MATRIX R WITH LMDF1490
C DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE SUCH THAT LMDF1500
C LMDF1510

C
$$P^T * (JAC^T * JAC) * P^T = R^T * R,$$
 LMDF1520
C LMDF1530
C LMDF1540

C WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL LMDF1550
C CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J) LMDF1560
C (SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRAPEZOIDAL LMDF1570
C PART OF FJAC CONTAINS INFORMATION GENERATED DURING LMDF1580
C THE COMPUTATION OF R. LMDF1590
C LMDF1600

C LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M LMDF1610
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC. LMDF1620

```

C IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT          LMDF1630
C DEFINES A PERMUTATION MATRIX P SUCH THAT JAC*P = Q*R,      LMDF1640
C WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS          LMDF1650
C ORTHOGONAL (NOT STORED), AND R IS UPPER TRIANGULAR        LMDF1660
C WITH DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE.       LMDF1670
C COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.   LMDF1680
C
C QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS          LMDF1690
C THE FIRST N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*FVEC.    LMDF1700
C
C WA1, WA2, AND WA3 ARE WORK ARRAYS OF LENGTH N.           LMDF1710
C
C WA4 IS A WORK ARRAY OF LENGTH M.                          LMDF1720
C
C SUBPROGRAMS CALLED
C
C USER-SUPPLIED ..... FCN                                LMDF1730
C
C MINPACK-SUPPLIED ... DPMPAR,ENORM,FDJAC2,LMPAR,QRFAC    LMDF1740
C
C FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,DSQRT,MOD         LMDF1750
C
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. LMDF1760
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE     LMDF1770
C
C *****
C
C INTEGER I,IFLAG,ITER,J,L                               LMDF1780
C DOUBLE PRECISION ACTRED,DELTA,DIRDER,EPSMCH,FNORM,FNORM1,GNORM,LMDF1790
C *          ONE,PAR,PNORM,PRERED,P1,P5,P25,P75,P0001,RATIO,LMDF1800
C *          SUM,TEMP,TEMP1,TEMP2,XNORM,ZERO                LMDF1810
C DOUBLE PRECISION DPMPAR,ENORM                         LMDF1820
C DATA ONE,P1,P5,P25,P75,P0001,ZERO                  LMDF1830
C *          /1.0D0,1.0D-1,5.0D-1,2.5D-1,7.5L-1,1.0D-4,...0D0/ LMDF1840
C
C EPSMCH IS THE MACHINE PRECISION.                   LMDF1850
C
C EPSMCH = DPMPAR(1)                                 LMDF1860
C
C INFO = 0                                         LMDF1870
C IFLAG = 0                                         LMDF1880
C NFEV = 0                                         LMDF1890
C
C CHECK THE INPUT PARAMETERS FOR ERRORS.            LMDF1900
C
C IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. M      LMDF1910
C *          .OR. FTOL .LT. ZERO .OR. XTOL .LT. ZERO .OR. GTOL .LT. ZERO LMDF1920
C *          .OR. MAXFEV .LE. 0 .OR. FACTOR .LE. ZERO) GO TO 300 LMDF1930
C IF (MODE .NE. 2) GO TO 20                           LMDF1940
C DO 10 J = 1, N
C     IF (DIAG(J) .LE. ZERO) GO TO 300               LMDF1950
10    CONTINUE                                         LMDF1960
20    CONTINUE                                         LMDF1970
C

```

```

C EVALUATE THE FUNCTION AT THE STARTING POINT           LMDF2170
C AND CALCULATE ITS NORM.                            LMDF2180
C
C IFLAG = 1                                         LMDF2190
C CALL FCN(M,N,X,FVEC,IFLAG)                         LMDF2200
C NFEV = 1                                           LMDF2210
C IF (IFLAG .LT. 0) GO TO 300                         LMDF2220
C FNORM = ENORM(M,FVEC)                             LMDF2230
C
C INITIALIZE LEVENBERG-MARQUARDT PARAMETER AND ITERATION COUNTER. LMDF2240
C
C PAR = ZERO                                         LMDF2250
C ITER = 1                                           LMDF2260
C
C BEGINNING OF THE OUTER LOOP.                      LMDF2270
C
C 30 CONTINUE                                         LMDF2280
C
C CALCULATE THE JACOBIAN MATRIX.                   LMDF2290
C
C IFLAG = 2                                         LMDF2300
C CALL FDJAC2(FCN,M,N,X,FVEC,FJAC,LDFJAC,IFLAG,EPSFCN,WA4) LMDF2310
C NFEV = NFEV + N                                 LMDF2320
C IF (IFLAG .LT. 0) GO TO 300                         LMDF2330
C
C IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES. LMDF2340
C
C IF (NPRINT .LE. 0) GO TO 40                         LMDF2350
C IFLAG = 0                                         LMDF2360
C IF (MOD(ITER-1,NPRINT) .EQ. 0) CALL FCN(M,N,X,FVEC,IFLAG) LMDF2370
C IF (IFLAG .LT. 0) GO TO 300                         LMDF2380
C
C 40 CONTINUE                                         LMDF2390
C
C COMPUTE THE QR FACTORIZATION OF THE JACOBIAN.    LMDF2400
C
C CALL QRFAC(M,N,FJAC,LDFJAC,.TRUE.,IPVT,N,WA1,WA2,WA3) LMDF2410
C
C ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING LMDF2420
C TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN. LMDF2430
C
C IF (ITER .NE. 1) GO TO 80                         LMDF2440
C IF (MODE .EQ. 2) GO TO 60                         LMDF2450
C DO 50 J = 1, N                                     LMDF2460
C     DIAG(J) = WA2(J)                               LMDF2470
C     IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE          LMDF2480
C
C 50 CONTINUE                                         LMDF2490
C
C 60 CONTINUE                                         LMDF2500
C
C ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X LMDF2510
C AND INITIALIZE THE STEP BOUND DELTA.               LMDF2520
C
C DO 70 J = 1, N                                     LMDF2530
C     WA3(J) = DIAG(J)*X(J)                         LMDF2540
C
C 70 CONTINUE                                         LMDF2550

```

```

XNORM = ENORM(N,WA3)                                LMDF2710
DELTA = FACTOR*XNORM                                LMDF2720
IF (DELTA .EQ. ZERO) DELTA = FACTOR                LMDF2730
80  CONTINUE                                         LMDF2740
C
C      FORM (Q TRANSPOSE)*FVEC AND STORE THE FIRST N COMPONENTS IN    LMDF2750
C      QTF.                                                       LMDF2760
C
DO 90 I = 1, M                                     LMDF2780
  WA4(I) = FVEC(I)                                 LMDF2790
90  CONTINUE                                         LMDF2800
DO 130 J = 1, N                                    LMDF2810
  IF (FJAC(J,J) .EQ. ZERO) GO TO 120             LMDF2820
  SUM = ZERO                                       LMDF2830
  DO 100 I = J, M                                 LMDF2840
    SUM = SUM + FJAC(I,J)*WA4(I)                  LMDF2850
100  CONTINUE                                         LMDF2860
  TEMP = -SUM/FJAC(J,J)                           LMDF2870
  DO 110 I = J, M                                 LMDF2880
    WA4(I) = WA4(I) + FJAC(I,J)*TEMP            LMDF2890
110  CONTINUE                                         LMDF2900
120  CONTINUE                                         LMDF2910
  FJAC(J,J) = WA1(J)                            LMDF2920
  QTF(J) = WA4(J)                               LMDF2930
130  CONTINUE                                         LMDF2940
C
C      COMPUTE THE NORM OF THE SCALED GRADIENT.          LMDF2950
C
GNORM = ZERO                                         LMDF2960
IF (FNORM .EQ. ZERO) GO TO 170                     LMDF2970
DO 160 J = 1, N                                    LMDF2980
  L = IPVT(J)                                     LMDF2990
  IF (WA2(L) .EQ. ZERO) GO TO 150                 LMDF3000
  SUM = ZERO                                       LMDF3010
  DO 140 I = 1, J                                 LMDF3020
    SUM = SUM + FJAC(I,J)*(QTF(I)/FNORM)        LMDF3030
140  CONTINUE                                         LMDF3040
  GNORM = DMAX1(GNORM,DABS(SUM/WA2(L)))         LMDF3050
150  CONTINUE                                         LMDF3060
160  CONTINUE                                         LMDF3070
170  CONTINUE                                         LMDF3080
C
C      TEST FOR CONVERGENCE OF THE GRADIENT NORM.       LMDF3090
C
IF (GNORM .LE. GTOL) INFO = 4                      LMDF3100
IF (INFO .NE. 0) GO TO 300                         LMDF3110
C
C      RESCALE IF NECESSARY.                          LMDF3120
C
IF (MODE .EQ. 2) GO TO 190                         LMDF3130
DO 180 J = 1, N                                    LMDF3140
  DIAG(J) = DMAX1(DIAG(J),WA2(J))                LMDF3150
180  CONTINUE                                         LMDF3160
190  CONTINUE                                         LMDF3170
                                         LMDF3180
                                         LMDF3190
                                         LMDF3200
                                         LMDF3210
                                         LMDF3220
                                         LMDF3230
                                         LMDF3240

```

```

C                                LMDF3250
C      BEGINNING OF THE INNER LOOP.   LMDF3260
C                                LMDF3270
C
200    CONTINUE                  LMDF3280
C                                LMDF3290
C      DETERMINE THE LEVENBERG-MARQUARDT PARAMETER.   LMDF3300
C                                LMDF3310
C
*       CALL LMPAR(N,FJAC,LDFJAC,IPVT,DIAG,QTF,DELTA,PAR,WA1,WA2,
          WA3,WA4)                  LMDF3320
C                                LMDF3330
C      STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.   LMDF3340
C                                LMDF3350
C
DO 210 J = 1, N                LMDF3360
  WA1(J) = -WA1(J)              LMDF3370
  WA2(J) = X(J) + WA1(J)        LMDF3380
  WA3(J) = DIAG(J)*WA1(J)        LMDF3390
210    CONTINUE                  LMDF3400
  PNORM = ENORM(N,WA3)          LMDF3410
C                                LMDF3420
C      ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.   LMDF3430
C                                LMDF3440
C
IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)   LMDF3450
C                                LMDF3460
C      EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.   LMDF3470
C                                LMDF3480
C
IFLAG = 1                      LMDF3490
CALL FCN(M,N,WA2,WA4,IFLAG)    LMDF3500
NFEV = NFEV + 1                LMDF3510
IF (IFLAG .LT. 0) GO TO 300    LMDF3520
FNORM1 = ENORM(M,WA4)          LMDF3530
C                                LMDF3540
C      COMPUTE THE SCALED ACTUAL REDUCTION.   LMDF3550
C                                LMDF3560
C
ACTRED = -ONE                  LMDF3570
IF (P1*FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2   LMDF3580
C                                LMDF3590
C      COMPUTE THE SCALED PREDICTED REDUCTION AND   LMDF3600
C      THE SCALED DIRECTIONAL DERIVATIVE.   LMDF3610
C                                LMDF3620
C
DO 230 J = 1, N                LMDF3630
  WA3(J) = ZERO                LMDF3640
  L = IPVT(J)                  LMDF3650
  TEMP = WA1(L)                LMDF3660
  DO 220 I = 1, J                LMDF3670
    WA3(I) = WA3(I) + FJAC(I,J)*TEMP   LMDF3680
220    CONTINUE                  LMDF3690
230    CONTINUE                  LMDF3700
  TEMP1 = ENORM(N,WA3)/FNORM    LMDF3710
  TEMP2 = (DSQRT(PAR)*PNORM)/FNORM   LMDF3720
  PRERED = TEMP1**2 + TEMP2**2/P5   LMDF3730
  DIRDER = -(TEMP1**2 + TEMP2**2)   LMDF3740
C                                LMDF3750
C      COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED   LMDF3760
C      REDUCTION.   LMDF3770
C                                LMDF3780

```

```

C          RATIO = ZERO          LMDF3790
          IF (PRERED .NE. ZERO) RATIO = ACTRED/PRERED  LMDF3800
C          UPDATE THE STEP BOUND.          LMDF3810
C          LMDF3820
C          LMDF3830
C          LMDF3840
IF (RATIO .GT. P25) GO TO 240          LMDF3850
  IF (ACTRED .GE. ZERO) TEMP = P5          LMDF3860
  IF (ACTRED .LT. ZERO)          LMDF3870
    TEMP = P5*DIRDER/(DIRDER + P5*ACTRED)          LMDF3880
    IF (P1*FNORM1 .GE. FNORM .OR. TEMP .LT. P1) TEMP = P1          LMDF3890
    DELTA = TEMP*DMIN1(DELTA,PNORM/P1)          LMDF3900
    PAR = PAR/TEMP          LMDF3910
    GO TO 260          LMDF3920
240      CONTINUE          LMDF3930
  IF (PAR .NE. ZERO .AND. RATIO .LT. P75) GO TO 250          LMDF3940
  DELTA = PNORM/P5          LMDF3950
  PAR = P5*PAR          LMDF3960
250      CONTINUE          LMDF3970
260      CONTINUE          LMDF3980
C          LMDF3990
C          TEST FOR SUCCESSFUL ITERATION.          LMDF4000
C          LMDF4010
IF (RATIO .LT. P0001) GO TO 290          LMDF4020
C          LMDF4030
C          SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.          LMDF4040
C          LMDF4050
DO 270 J = 1, N          LMDF4060
  X(J) = WA2(J)          LMDF4070
  WA2(J) = DIAG(J)*X(J)          LMDF4080
270      CONTINUE          LMDF4090
DO 280 I = 1, M          LMDF4100
  FVEC(I) = WA4(I)          LMDF4110
280      CONTINUE          LMDF4120
  XNORM = ENORM(N,WA2)          LMDF4130
  FNORM = FNORM1          LMDF4140
  ITER = ITER + 1          LMDF4150
290      CONTINUE          LMDF4160
C          LMDF4170
C          TESTS FOR CONVERGENCE.          LMDF4180
C          LMDF4190
  IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL          LMDF4200
    .AND. P5*RATIO .LE. ONE) INFO = 1          LMDF4210
  IF (DELTA .LE. XTOL*XNORM) INFO = 2          LMDF4220
  IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL          LMDF4230
    .AND. P5*RATIO .LE. ONE .AND. INFO .EQ. 2) INFO = 3          LMDF4240
  IF (INFO .NE. 0) GO TO 300          LMDF4250
C          LMDF4260
C          TESTS FOR TERMINATION AND STRINGENT TOLERANCES.          LMDF4270
C          LMDF4280
  IF (NFEV .GE. MAXFEV) INFO = 5          LMDF4290
  IF (DABS(ACTRED) .LE. EPSMCH .AND. PRERED .LE. EPSMCH          LMDF4300
    .AND. P5*RATIO .LE. ONE) INFO = 6          LMDF4310
  IF (DELTA .LE. EPSMCH*XNORM) INFO = 7          LMDF4320

```

```
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
C                                         IF (RATIO .LT. P0001) GO TO 200      LMDF4380
C                                         LMDF4390
C                                         END OF THE OUTER LOOP.      LMDF4400
C                                         LMDF4410
C                                         GO TO 30                      LMDF4420
300 CONTINUE                           LMDF4430
C                                         LMDF4440
C                                         TERMINATION, EITHER NORMAL OR USER IMPOSED. LMDF4450
C                                         LMDF4460
C                                         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 THIS PAGE ALONE
WAS INTENTIONALLY
LEFT BLANK

```

SUBROUTINE LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)          LMF10010
INTEGER M,N,INFO,LWA                                         LMF10020
INTEGER IWA(N)                                              LMF10030
DOUBLE PRECISION TOL                                         LMF10040
DOUBLE PRECISION X(N),FVEC(M),WA(LWA)                         LMF10050
EXTERNAL FCN                                                 LMF10060
*****
C
C SUBROUTINE LMDIF1                                         LMF10070
C
C THE PURPOSE OF LMDIF1 IS TO MINIMIZE THE SUM OF THE SQUARES OF   LMF10080
C M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF THE   LMF10090
C LEVENBERG-MARQUARDT ALGORITHM. THIS IS DONE BY USING THE MORE   LMF10100
C GENERAL LEAST-SQUARES SOLVER LMDIF. THE USER MUST PROVIDE A   LMF10110
C SUBROUTINE WHICH CALCULATES THE FUNCTIONS. THE JACOBIAN IS   LMF10120
C THEN CALCULATED BY A FORWARD-DIFFERENCE APPROXIMATION.        LMF10130
C
C THE SUBROUTINE STATEMENT IS                               LMF10140
C
C SUBROUTINE LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)          LMF10150
C
C WHERE                                                       LMF10160
C
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH          LMF10170
C CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED                LMF10180
C IN AN EXTERNAL STATEMENT IN THE USER CALLING                 LMF10190
C PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.                  LMF10200
C
C SUBROUTINE FCN(M,N,X,FVEC,IFLAG)                            LMF10210
C INTEGER M,N,IFLAG                                         LMF10220
C DOUBLE PRECISION X(N),FVEC(M)                                LMF10230
C -----
C CALCULATE THE FUNCTIONS AT X AND                           LMF10240
C RETURN THIS VECTOR IN FVEC.                                LMF10250
C -----
C RETURN                                               LMF10260
C END                                                 LMF10270
C
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS      LMF10280
C THE USER WANTS TO TERMINATE EXECUTION OF LMDIF1.            LMF10290
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.             LMF10300
C
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER    LMF10310
C OF FUNCTIONS.                                              LMF10320
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER    LMF10330
C OF VARIABLES. N MUST NOT EXCEED M.                          LMF10340
C
C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN          LMF10350
C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X     LMF10360
C CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.         LMF10370
C
C FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS          LMF10380
C THE FUNCTIONS EVALUATED AT THE OUTPUT X.                    LMF10390

```

C TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS LMF10550
 C WHEN THE ALGORITHM ESTIMATES EITHER THAT THE RELATIVE LMF10560
 C ERROR IN THE SUM OF SQUARES IS AT MOST TOL OR THAT LMF10570
 C THE RELATIVE ERROR BETWEEN X AND THE SOLUTION IS AT LMF10580
 C MOST TOL. LMF10590
 C LMF10600
 C LMF10610
 C INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS LMF10620
 C TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE) LMF10630
 C VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE, LMF10640
 C INFO IS SET AS FOLLOWS. LMF10650
 C LMF10660
 C INFO = 0 IMPROPER INPUT PARAMETERS. LMF10670
 C LMF10680
 C INFO = 1 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR LMF10690
 C IN THE SUM OF SQUARES IS AT MOST TOL. LMF10700
 C LMF10710
 C INFO = 2 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR LMF10720
 C BETWEEN X AND THE SOLUTION IS AT MOST TOL. LMF10730
 C LMF10740
 C INFO = 3 CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD. LMF10750
 C LMF10760
 C INFO = 4 FVEC IS ORTHOGONAL TO THE COLUMNS OF THE LMF10770
 C JACOBIAN TO MACHINE PRECISION. LMF10780
 C LMF10790
 C INFO = 5 NUMBER OF CALLS TO FCN HAS REACHED OR LMF10800
 C EXCEEDED 200*(N+1). LMF10810
 C LMF10820
 C INFO = 6 TOL IS TOO SMALL. NO FURTHER REDUCTION IN LMF10830
 C THE SUM OF SQUARES IS POSSIBLE. LMF10840
 C LMF10850
 C INFO = 7 TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN LMF10860
 C THE APPROXIMATE SOLUTION X IS POSSIBLE. LMF10870
 C LMF10880
 C IWA IS AN INTEGER WORK ARRAY OF LENGTH N. LMF10890
 C LMF10900
 C WA IS A WORK ARRAY OF LENGTH LWA. LMF10910
 C LMF10920
 C LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN LMF10930
 C M*N+5*N+M. LMF10940
 C LMF10950
 C SUBPROGRAMS CALLED LMF10960
 C LMF10970
 C USER-SUPPLIED FCN LMF10980
 C LMF10990
 C MINPACK-SUPPLIED ... LMDIF LMF11000
 C LMF11010
 C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. LMF11020
 C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE LMF11030
 C LMF11040
 C ***** LMF11050
 C INTEGER MAXFEV, MODE, MP5N, NFEV, NPRINT LMF11060
 C DOUBLE PRECISION EPSFCN, FACTOR, FTOL, GTOL, XTOL, ZERO LMF11070
 C DATA FACTOR, ZERO /1.0D2, 0.0D0/ LMF11080

```

INFO = 0                                LMF11090
C
C      CHECK THE INPUT PARAMETERS FOR ERRORS.          LMF11100
C
C      IF (N .LE. 0 .OR. M .LT. N .OR. TOL .LT. ZERO    LMF11110
*      .OR. LWA .LT. M*N + 5*N + M) GO TO 10          LMF11120
C
C      CALL LMDIF.                                     LMF11130
C
MAXFEV = 200*(N + 1)                      LMF11140
FTOL = TOL                               LMF11150
XTOL = TOL                               LMF11160
GTOL = ZERO                             LMF11170
EPSFCN = ZERO                           LMF11180
MODE = 1                                 LMF11190
NPRINT = 0                               LMF11200
MP5N = M + 5*N                           LMF11210
CALL LMDIF(FCN,M,N,X,FVEC,FTOL,XTOL,GTOL,MAXFEV,EPSFCN,WA(1),   LMF11220
*           MODE,FACTOR,NPRINT,INFO,NFEV,WA(MP5N+1),M,IWA,        LMF11230
*           WA(N+1),WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))     LMF11240
IF (INFO .EQ. 8) INFO = 4                 LMF11250
10 CONTINUE                               LMF11260
      RETURN                                LMF11270
C
C      LAST CARD OF SUBROUTINE LMDIF1.          LMF11280
C
END                                     LMF11290
                                         LMF11300
                                         LMF11310
                                         LMF11320
                                         LMF11330
                                         LMF11340
                                         LMF11350

```

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 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 N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R. LMPR0550
 C LMPR0560
 C R IS AN N BY N ARRAY. ON INPUT THE FULL UPPER TRIANGLE LMPR0570
 C MUST CONTAIN THE FULL UPPER TRIANGLE OF THE MATRIX R. LMPR0580
 C ON OUTPUT THE FULL UPPER TRIANGLE IS UNALTERED, AND THE LMPR0590
 C STRICT LOWER TRIANGLE CONTAINS THE STRICT UPPER TRIANGLE LMPR0600
 C (TRANSPOSED) OF THE UPPER TRIANGULAR MATRIX S. LMPR0610
 C LMPR0620
 C LDR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N LMPR0630
 C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY R. LMPR0640
 C LMPR0650
 C IPVT IS AN INTEGER INPUT ARRAY OF LENGTH N WHICH DEFINES THE LMPR0660
 C PERMUTATION MATRIX P SUCH THAT $A^*P = Q^*R$. COLUMN J OF P LMPR0670
 C IS COLUMN IPVT(J) OF THE IDENTITY MATRIX. LMPR0680
 C LMPR0690
 C DIAG IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE LMPR0700
 C DIAGONAL ELEMENTS OF THE MATRIX D. LMPR0710
 C LMPR0720
 C QTB IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE FIRST LMPR0730
 C N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*B. LMPR0740
 C LMPR0750
 C DELTA IS A POSITIVE INPUT VARIABLE WHICH SPECIFIES AN UPPER LMPR0760
 C BOUND ON THE EUCLIDEAN NORM OF D*X. LMPR0770
 C LMPR0780
 C PAR IS A NONNEGATIVE VARIABLE. ON INPUT PAR CONTAINS AN LMPR0790
 C INITIAL ESTIMATE OF THE LEVENBERG-MARQUARDT PARAMETER. LMPR0800
 C ON OUTPUT PAR CONTAINS THE FINAL ESTIMATE. LMPR0810
 C LMPR0820
 C X IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE LEAST LMPR0830
 C SQUARES SOLUTION OF THE SYSTEM $A^*X = B$, $SQRT(PAR)^*D^*X = 0$, LMPR0840
 C FOR THE OUTPUT PAR. LMPR0850
 C LMPR0860
 C SDIAG IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE LMPR0870
 C DIAGONAL ELEMENTS OF THE UPPER TRIANGULAR MATRIX S. LMPR0880
 C LMPR0890
 C WA1 AND WA2 ARE WORK ARRAYS OF LENGTH N. LMPR0900
 C LMPR0910
 C SUBPROGRAMS CALLED LMPR0920
 C LMPR0930
 C MINPACK-SUPPLIED ... DPMPAR,ENORM,QRSOLV LMPR0940
 C LMPR0950
 C FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,DSQRT LMPR0960
 C LMPR0970
 C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. LMPR0980
 C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE LMPR0990
 C LMPR1000
 C ***** LMPR1010
 C INTEGER I,ITER,J,JM1,JP1,K,L,NSING LMPR1020
 C DOUBLE PRECISION DXNORM,DWARP,FP,GNORM,PARC,PARL,PARU,P1,P001, LMPR1030
 C * SUM,TEMP,ZERO LMPR1040
 C DOUBLE PRECISION DPMPAR,ENORM LMPR1050
 C DATA P1,P001,ZERO /1.0D-1,1.0D-3,0.0D/ LMPR1060
 C LMPR1070
 C DWARP IS THE SMALLEST POSITIVE MAGNITUDE. LMPR1080

```

C          DWARF = DPMPAR(2)                                LMPR1090
C          COMPUTE AND STORE IN X THE GAUSS-NEWTON DIRECTION. IF THE   LMPR1100
C          JACOBIAN IS RANK-DEFICIENT, OBTAIN A LEAST SQUARES SOLUTION. LMPR1110
C          NSING = N                                         LMPR1120
C          DO 10 J = 1, N                                     LMPR1130
C              WA1(J) = QTB(J)                               LMPR1140
C              IF (R(J,J) .EQ. ZERO .AND. NSING .EQ. N) NSING = J - 1 LMPR1150
C              IF (NSING .LT. N) WA1(J) = ZERO             LMPR1160
10        CONTINUE                                         LMPR1170
        IF (NSING .LT. 1) GO TO 50                         LMPR1180
        DO 40 K = 1, NSING                                 LMPR1190
            J = NSING - K + 1                            LMPR1200
            WA1(J) = WA1(J)/R(J,J)                      LMPR1210
            TEMP = WA1(J)                                LMPR1220
            JM1 = J - 1                                  LMPR1230
            IF (JM1 .LT. 1) GO TO 30                     LMPR1240
            DO 20 I = 1, JM1                           LMPR1250
                WA1(I) = WA1(I) - R(I,J)*TEMP           LMPR1260
20        CONTINUE                                         LMPR1270
30        CONTINUE                                         LMPR1280
40        CONTINUE                                         LMPR1290
50        CONTINUE                                         LMPR1300
        DO 60 J = 1, N                                     LMPR1310
            L = IPVT(J)                                LMPR1320
            X(L) = WA1(J)                                LMPR1330
60        CONTINUE                                         LMPR1340
C          INITIALIZE THE ITERATION COUNTER.               LMPR1350
C          EVALUATE THE FUNCTION AT THE ORIGIN, AND TEST      LMPR1360
C          FOR ACCEPTANCE OF THE GAUSS-NEWTON DIRECTION.    LMPR1370
C          LMPR1380
        ITER = 0                                         LMPR1390
        DO 70 J = 1, N                                     LMPR1400
            WA2(J) = DIAG(J)*X(J)                        LMPR1410
70        CONTINUE                                         LMPR1420
        DXNORM = ENORM(N,WA2)                          LMPR1430
        FP = DXNORM - DELTA                           LMPR1440
        IF (FP .LE. P1*DELTA) GO TO 220             LMPR1450
C          IF THE JACOBIAN IS NOT RANK DEFICIENT, THE NEWTON   LMPR1460
C          STEP PROVIDES A LOWER BOUND, PARL, FOR THE ZERO OF   LMPR1470
C          THE FUNCTION. OTHERWISE SET THIS BOUND TO ZERO.       LMPR1480
C          LMPR1490
        PARL = ZERO                                       LMPR1500
        IF (NSING .LT. N) GO TO 120                     LMPR1510
        DO 80 J = 1, N                                     LMPR1520
            L = IPVT(J)                                LMPR1530
            WA1(J) = DIAG(L)*(WA2(L)/DXNORM)           LMPR1540
80        CONTINUE                                         LMPR1550
        DO 110 J = 1, N                                    LMPR1560
            SUM = ZERO                                LMPR1570

```

```

JM1 = J - 1 LMPR1630
IF (JM1 .LT. 1) GO TO 100 LMPR1640
DO 90 I = 1, JM1 LMPR1650
    SUM = SUM + R(I,J)*WA1(I) LMPR1660
90    CONTINUE LMPR1670
100   CONTINUE LMPR1680
    WA1(J) = (WA1(J) - SUM)/R(J,J) LMPR1690
110   CONTINUE LMPR1700
    TEMP = ENORM(N,WA1) LMPR1710
    PARL = ((FP/DELTA)/TEMP)/TEMP LMPR1720
120   CONTINUE LMPR1730
C LMPR1740
C CALCULATE AN UPPER BOUND, PARU, FOR THE ZERO OF THE FUNCTION. LMPR1750
C LMPR1760
DO 140 J = 1, N LMPR1770
    SUM = ZERO LMPR1780
    DO 130 I = 1, J LMPR1790
        SUM = SUM + R(I,J)*QTB(I) LMPR1800
130    CONTINUE LMPR1810
    L = IPVT(J) LMPR1820
    WA1(J) = SUM/DIAG(L) LMPR1830
140    CONTINUE LMPR1840
    GNORM = ENORM(N,WA1) LMPR1850
    PARU = GNORM/DELTA LMPR1860
    IF (PARU .EQ. ZERO) PARU = DWARF/DMIN1(DELTA,P1) LMPR1870
C LMPR1880
C IF THE INPUT PAR LIES OUTSIDE OF THE INTERVAL (PARL,PARU), LMPR1890
C SET PAR TO THE CLOSER ENDPOINT. LMPR1900
C LMPR1910
    PAR = DMAX1(PAR,PARL) LMPR1920
    PAR = DMIN1(PAR,PARU) LMPR1930
    IF (PAR .EQ. ZERO) PAR = GNORM/DXNORM LMPR1940
C LMPR1950
C BEGINNING OF AN ITERATION. LMPR1960
C LMPR1970
150 CONTINUE LMPR1980
    ITER = ITER + 1 LMPR1990
C LMPR2000
C EVALUATE THE FUNCTION AT THE CURRENT VALUE OF PAR. LMPR2010
C LMPR2020
    IF (PAR .EQ. ZERO) PAR = DMAX1(DWARP,P001*PARU) LMPR2030
    TEMP = DSQRT(PAR) LMPR2040
    DO 160 J = 1, N LMPR2050
        WA1(J) = TEMP*DIAG(J) LMPR2060
160    CONTINUE LMPR2070
    CALL QRSLV(N,R,LDR,IPVT,WA1,QTB,X,SDIAG,WA2) LMPR2080
    DO 170 J = 1, N LMPR2090
        WA2(J) = DIAG(J)*X(J) LMPR2100
170    CONTINUE LMPR2110
    DXNORM = ENORM(N,WA2) LMPR2120
    TEMP = FP LMPR2130
    FP = DXNORM - DELTA LMPR2140
C LMPR2150
C IF THE FUNCTION IS SMALL ENOUGH, ACCEPT THE CURRENT VALUE LMPR2160

```

```

C OF PAR. ALSO TEST FOR THE EXCEPTIONAL CASES WHERE PARM
C IS ZERO OR THE NUMBER OF ITERATIONS HAS REACHED 10.
C
C IF (DABS(FP) .LE. P1*DELTA
*   .OR. PARM .EQ. ZERO .AND. FP .LE. TEMP
*   .AND. TEMP .LT. ZERO .OR. ITER .EQ. 10) GO TO 220
C
C COMPUTE THE NEWTON CORRECTION.
C
C DO 180 J = 1, N
180   L = IPVT(J)
      WA1(J) = DIAG(L)*(WA2(L)/DXNORM)
      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 DEPENDING ON THE SIGN OF THE FUNCTION, UPDATE PARM OR PARU.
C
C IF (FP .GT. ZERO) PARM = DMAX1(PARM,PAR)
C IF (FP .LT. ZERO) PARU = DMIN1(PARU,PAR)

C COMPUTE AN IMPROVED ESTIMATE FOR PAR.
C
C PAR = DMAX1(PARM,PAR+PARC)
C
C END OF AN ITERATION.
C
C GO TO 150
220 CONTINUE

C TERMINATION.
C
C IF (ITER .EQ. 0) PAR = ZERO.
C RETURN

C LAST CARD OF SUBROUTINE LMPAR.
C
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)                                LMSR0010
INTEGER M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV               LMSR0020
INTEGER IPVT(N)                                                 LMSR0030
LOGICAL SING                                              LMSR0040
DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR                         LMSR0050
DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),DIAG(N),QTF(N),
*                  WA1(N),WA2(N),WA3(N),WA4(M)                           LMSR0060
*****                                         LMSR0070
C
C SUBROUTINE LMSTR                                         LMSR0080
C
C THE PURPOSE OF LMSTR IS TO MINIMIZE THE SUM OF THE SQUARES OF   LMSR0090
C M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF      LMSR0100
C THE LEVENBERG-MARQUARDT ALGORITHM WHICH USES MINIMAL STORAGE.    LMSR0110
C THE USER MUST PROVIDE A SUBROUTINE WHICH CALCULATES THE        LMSR0120
C FUNCTIONS AND THE ROWS OF THE JACOBIAN.                         LMSR0130
C
C THE SUBROUTINE STATEMENT IS                                 LMSR0140
C
C SUBROUTINE LMSTR(FCN,M,N,X,FVEC,FJAC,LDEJAC,FTOL,XTOL,GTOL,
C                   MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,
C                   NJEV,IPVT,QTF,WA1,WA2,WA3,WA4)                      LMSR0150
C
C WHERE                                              LMSR0160
C
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH          LMSR0170
C CALCULATES THE FUNCTIONS AND THE ROWS OF THE JACOBIAN.        LMSR0180
C FCN MUST BE DECLARED IN AN EXTERNAL STATEMENT IN THE          LMSR0190
C USER CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.       LMSR0200
C
C SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)                         LMSR0210
C INTEGER M,N,IFLAG                                         LMSR0220
C DOUBLE PRECISION X(N),FVEC(M),FJROW(N)                         LMSR0230
C -----
C IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND                LMSR0240
C RETURN THIS VECTOR IN FVEC.                                     LMSR0250
C IF IFLAG = I CALCULATE THE (I-1)-ST ROW OF THE                LMSR0260
C JACOBIAN AT X AND RETURN THIS VECTOR IN FJROW.              LMSR0270
C -----
C RETURN                                              LMSR0280
C END                                                 LMSR0290
C
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS        LMSR0300
C THE USER WANTS TO TERMINATE EXECUTION OF LMSTR.             LMSR0310
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.            LMSR0320
C
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER      LMSR0330
C OF FUNCTIONS.                                              LMSR0340
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER      LMSR0350
C OF VARIABLES. N MUST NOT EXCEED M.                          LMSR0360
C
C

```

C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN LMSR0550
 C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X LMSR0560
 C CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR. LMSR0570
 C LMSR0580

C FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS LMSR0590
 C THE FUNCTIONS EVALUATED AT THE OUTPUT X. LMSR0600
 C LMSR0610

C FJAC IS AN OUTPUT N BY N ARRAY. THE UPPER TRIANGLE OF FJAC LMSR0620
 C CONTAINS AN UPPER TRIANGULAR MATRIX R SUCH THAT LMSR0630
 C LMSR0640

C
$$\begin{matrix} T & T & T \\ P^T & (JAC^T)^T & P^T \\ & = & R^T \end{matrix}$$
, LMSR0650
 C WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL LMSR0660
 C CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J) LMSR0670
 C (SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRIANGULAR LMSR0680
 C PART OF FJAC CONTAINS INFORMATION GENERATED DURING LMSR0690
 C THE COMPUTATION OF R. LMSR0700
 C LMSR0710
 C LMSR0720
 C LMSR0730

C LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N LMSR0740
 C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC. LMSR0750
 C LMSR0760

C FTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION LMSR0770
 C OCCURS WHEN BOTH THE ACTUAL AND PREDICTED RELATIVE LMSR0780
 C REDUCTIONS IN THE SUM OF SQUARES ARE AT MOST FTOL. LMSR0790
 C THEREFORE, FTOL MEASURES THE RELATIVE ERROR DESIRED LMSR0800
 C IN THE SUM OF SQUARES. LMSR0810
 C LMSR0820

C XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION LMSR0830
 C OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE LMSR0840
 C ITERATES IS AT MOST XTOL. THEREFORE, XTOL MEASURES THE LMSR0850
 C RELATIVE ERROR DESIRED IN THE APPROXIMATE SOLUTION. LMSR0860
 C LMSR0870

C GTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION LMSR0880
 C OCCURS WHEN THE COSINE OF THE ANGLE BETWEEN FVEC AND LMSR0890
 C ANY COLUMN OF THE JACOBIAN IS AT MOST GTOL IN ABSOLUTE LMSR0900
 C VALUE. THEREFORE, GTOL MEASURES THE ORTHOGONALITY LMSR0910
 C DESIRED BETWEEN THE FUNCTION VECTOR AND THE COLUMNS LMSR0920
 C OF THE JACOBIAN. LMSR0930
 C LMSR0940

C MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION LMSR0950
 C OCCURS WHEN THE NUMBER OF CALLS TO FCN WITH IFLAG = 1 LMSR0960
 C HAS REACHED MAXFEV. LMSR0970
 C LMSR0980

C DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE LMSR0990
 C BELOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG LMSR1000
 C MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS LMSR1010
 C MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES. LMSR1020
 C LMSR1030

C MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE LMSR1040
 C VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2, LMSR1050
 C THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER LMSR1060
 C VALUES OF MODE ARE EQUIVALENT TO MODE = 1. LMSR1070
 C LMSR1080

C FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE LMSR1090
 C INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF LMSR1100
 C FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE LMSR1110
 C TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE LMSR1120
 C INTERVAL (.1,100.). 100. IS A GENERALLY RECOMMENDED VALUE. LMSR1130
 C LMSR1140

C NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED LMSR1150
 C PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE, LMSR1160
 C FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST LMSR1170
 C ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND LMSR1180
 C IMMEDIATELY PRIOR TO RETURN, WITH X AND FVEC AVAILABLE LMSR1190
 C FOR PRINTING. IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS LMSR1200
 C OF FCN WITH IFLAG = 0 ARE MADE. LMSR1210
 C LMSR1220

C INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS LMSR1230
 C TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE) LMSR1240
 C VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE, LMSR1250
 C INFO IS SET AS FOLLOWS. LMSR1260
 C LMSR1270

C INFO = 0 IMPROPER INPUT PARAMETERS. LMSR1280
 C LMSR1290

C INFO = 1 BOTH ACTUAL AND PREDICTED RELATIVE REDUCTIONS LMSR1300
 C IN THE SUM OF SQUARES ARE AT MOST FTOL. LMSR1310
 C LMSR1320

C INFO = 2 RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES LMSR1330
 C IS AT MOST XTOL. LMSR1340
 C LMSR1350

C INFO = 3 CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD. LMSR1360
 C LMSR1370

C INFO = 4 THE COSINE OF THE ANGLE BETWEEN FVEC AND ANY LMSR1380
 C COLUMN OF THE JACOBIAN IS AT MOST GTOL IN LMSR1390
 C ABSOLUTE VALUE. LMSR1400
 C LMSR1410

C INFO = 5 NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS LMSR1420
 C REACHED MAXFEV. LMSR1430
 C LMSR1440

C INFO = 6 FTOL IS TOO SMALL. NO FURTHER REDUCTION IN LMSR1450
 C THE SUM OF SQUARES IS POSSIBLE. LMSR1460
 C LMSR1470

C INFO = 7 XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN LMSR1480
 C THE APPROXIMATE SOLUTION X IS POSSIBLE. LMSR1490
 C LMSR1500

C INFO = 8 GTOL IS TOO SMALL. FVEC IS ORTHOGONAL TO THE LMSR1510
 C COLUMNS OF THE JACOBIAN TO MACHINE PRECISION. LMSR1520
 C LMSR1530

C NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF LMSR1540
 C CALLS TO FCN WITH IFLAG = 1. LMSR1550
 C LMSR1560

C NJEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF LMSR1570
 C CALLS TO FCN WITH IFLAG = 2. LMSR1580
 C LMSR1590

C IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT LMSR1600
 C DEFINES A PERMUTATION MATRIX P SUCH THAT JAC*P = Q*R, LMSR1610
 C WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS LMSR1620

```

C ORTHOGONAL (NOT STORED), AND R IS UPPER TRIANGULAR.          LMSR1630
C COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.      LMSR1640
C
C QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS           LMSR1650
C   THE FIRST N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*FVEC.    LMSR1660
C
C WA1, WA2, AND WA3 ARE WORK ARRAYS OF LENGTH N.            LMSR1670
C
C WA4 IS A WORK ARRAY OF LENGTH M.                          LMSR1680
C
C SUBPROGRAMS CALLED                                     LMSR1690
C
C   USER-SUPPLIED ..... FCN                            LMSR1700
C
C   MINPACK-SUPPLIED ... DPMPAR,ENORM,LMPAR,QRFAC,RWUPDT LMSR1710
C
C   FORTRAN-SUPPLIED ... DABS,DMAX1,DMIN1,DSQRT,MOD     LMSR1720
C
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. LMSR1730
C BURTON S. GARBOW, DUDLEY V. GOETSCHEL, KENNETH E. HILLSTROM, LMSR1740
C JORGE J. MORE                                         LMSR1750
C
C *****
C INTEGER I,IFLAG,ITER,J,L                                LMSR1760
C DOUBLE PRECISION ACTRED,DELTA,DIRDER,EPSMCH,FNORM,FNORM1,GNORM, LMSR1770
C *          ONE,PAR,PNORM,PRERED,P1,P5,P25,P75,P0001,RATIO, LMSR1780
C *          SUM,TEMP,TEMP1,TEMP2,XNORM,ZERO                LMSR1790
C DOUBLE PRECISION DPMPAR,ENORM                         LMSR1800
C DATA ONE,P1,P5,P25,P75,P0001,ZERO                  LMSR1810
C *          /1.0D0,1.0D-1,5.0D-1,2.5D-1,7.5D-1,1.0D-4,0.0D0/ LMSR1820
C
C EPSMCH IS THE MACHINE PRECISION.                      LMSR1830
C
C EPSMCH = DPMPAR(1)                                    LMSR1840
C
C INFO = 0                                            LMSR1850
C IFLAG = 0                                           LMSR1860
C NFEV = 0                                            LMSR1870
C NJEV = 0                                            LMSR1880
C
C CHECK THE INPUT PARAMETERS FOR ERRORS.             LMSR1890
C
C IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. N      LMSR1900
C *          .OR. FTOL .LT. ZERO .OR. XTOL .LT. ZERO .OR. GTOL .LT. ZERO LMSR1910
C *          .OR. MAXFEV .LE. 0 .OR. FACTOR .LE. ZERO) GO TO 340 LMSR1920
C IF (MODE .NE. 2) GO TO 20                           LMSR1930
C DO 10 J = 1, N                                      LMSR1940
C   IF (DIAG(J) .LE. ZERO) GO TO 340                 LMSR1950
C 10 CONTINUE                                         LMSR1960
C 20 CONTINUE                                         LMSR1970
C
C EVALUATE THE FUNCTION AT THE STARTING POINT       LMSR1980
C AND CALCULATE ITS NORM.                           LMSR1990
C
C

```

```

IFLAG = 1                                LMSR2170
CALL FCN(M,N,X,FVEC,WA3,IFLAG)          LMSR2180
NFEV = 1                                 LMSR2190
IF (IFLAG .LT. 0) GO TO 340             LMSR2200
FNORM = ENORM(M,FVEC)                  LMSR2210
LMSR2220
C
C   INITIALIZE LEVENBERG-MARQUARDT PARAMETER AND ITERATION COUNTER. LMSR2230
C
C   PAR = ZERO                           LMSR2240
ITER = 1                                LMSR2250
LMSR2260
C
C   BEGINNING OF THE OUTER LOOP.        LMSR2270
LMSR2280
LMSR2290
C
30 CONTINUE                               LMSR2300
LMSR2310
C
C   IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.      LMSR2320
C
C   IF (NPRINT .LE. 0) GO TO 40           LMSR2330
IFLAG = 0                                LMSR2340
LMSR2350
IF (MOD(ITER-1,NPRINT) .EQ. 0) CALL FCN(M,N,X,FVEC,WA3,IFLAG) LMSR2360
IF (IFLAG .LT. 0) GO TO 340             LMSR2370
LMSR2380
40 CONTINUE                               LMSR2390
LMSR2400
C
C   COMPUTE THE QR FACTORIZATION OF THE JACOBIAN MATRIX      LMSR2410
C   CALCULATED ONE ROW AT A TIME, WHILE SIMULTANEOUSLY      LMSR2420
C   FORMING (Q TRANSPOSE)*FVEC AND STORING THE FIRST      LMSR2430
C   N COMPONENTS IN QTF.                                LMSR2440
LMSR2450
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      LMSR2590
REORDER ITS COLUMNS AND UPDATE THE COMPONENTS OF QTF.      LMSR2600
LMSR2610
LMSR2620
LMSR2630
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
LMSR2640
LMSR2650
LMSR2660
LMSR2670
LMSR2680
LMSR2690
LMSR2700

```

```

CALL QRFAC(N,N,FJAC,LDFJAC,.TRUE.,IPVT,N,WA1,WA2,WA3)          LMSR2710
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.          LMSR3250
C                                                 LMSR3260
C IF (GNORM .LE. GTOL) INFO = 4                      LMSR3270
C IF (INFO .NE. 0) GO TO 340                         LMSR3280
C                                                 LMSR3290
C RESCALE IF NECESSARY.                            LMSR3300
C                                                 LMSR3310
C IF (MODE .EQ. 2) GO TO 230                         LMSR3320
C DO 220 J = 1, N                                     LMSR3330
C     DIAG(J) = DMAX1(DIAG(J),WA2(J))                LMSR3340
220     CONTINUE                                         LMSR3350
C 230     CONTINUE                                         LMSR3360
C                                                 LMSR3370
C BEGINNING OF THE INNER LOOP.                      LMSR3380
C                                                 LMSR3390
C 240     CONTINUE                                         LMSR3400
C                                                 LMSR3410
C DETERMINE THE LEVENBERG-MARQUARDT PARAMETER.    LMSR3420
C                                                 LMSR3430
C CALL LMPAR(N,FJAC,LDFJAC,IPVT,DIAG,QTF,DELTA,PAR,WA1,WA2,   LMSR3440
*          WA3,WA4)                                     LMSR3450
C                                                 LMSR3460
C STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P. LMSR3470
C                                                 LMSR3480
C DO 250 J = 1, N                                     LMSR3490
C     WA1(J) = -WA1(J)                                LMSR3500
C     WA2(J) = X(J) + WA1(J)                           LMSR3510
C     WA3(J) = DIAG(J)*WA1(J)                          LMSR3520
250     CONTINUE                                         LMSR3530
C     PNORM = ENORM(N,WA3)                            LMSR3540
C                                                 LMSR3550
C ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND. LMSR3560
C                                                 LMSR3570
C IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)        LMSR3580
C                                                 LMSR3590
C EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM. LMSR3600
C                                                 LMSR3610
C IFLAG = 1                                           LMSR3620
C CALL FCN(M,N,WA2,WA4,WA3,IFLAG)                  LMSR3630
C NFEV = NFEV + 1                                    LMSR3640
C IF (IFLAG .LT. 0) GO TO 340                      LMSR3650
C FNORM1 = ENORM(M,WA4)                            LMSR3660
C                                                 LMSR3670
C COMPUTE THE SCALED ACTUAL REDUCTION.            LMSR3680
C                                                 LMSR3690
C ACTRED = -ONE                                     LMSR3700
C IF (P1*FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2 LMSR3710
C                                                 LMSR3720
C COMPUTE THE SCALED PREDICTED REDUCTION AND      LMSR3730
C THE SCALED DIRECTIONAL DERIVATIVE.             LMSR3740
C                                                 LMSR3750
C DO 270 J = 1, N                                     LMSR3760
C     WA3(J) = ZERO                                 LMSR3770
C     L = IPVT(J)                                  LMSR3780

```

```

        TEMP = WA1(L)                                LMSR3790
        DO 260 I = 1, J                            LMSR3800
          WA3(I) = WA3(I) + FJAC(I,J)*TEMP
        CONTINUE                                     LMSR3810
260      CONTINUE                                     LMSR3820
270      TEMP1 = ENORM(N,WA3)/FNORM
        TEMP2 = (DSQRT(PAR)*PNORM)/FNORM
        PRERED = TEMP1**2 + TEMP2**2/P5
        DIRDER = -(TEMP1**2 + TEMP2**2)             LMSR3830
C
C      COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED
C      REDUCTION.                                 LMSR3840
C
C      RATIO = ZERO                               LMSR3850
C      IF (PRERED .NE. ZERO) RATIO = ACTRED/PRERED
C
C      UPDATE THE STEP BOUND.                    LMSR3860
C
C      IF (RATIO .GT. P25) GO TO 280
        IF (ACTRED .GE. ZERO) TEMP = P5
        IF (ACTRED .LT. ZERO)
*          TEMP = P5*DIRDER/(DIRDER + P5*ACTRED)   LMSR3870
          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                                     LMSR3880
          IF (PAR .NE. ZERO .AND. RATIO .LT. P75) GO TO 290
          DELTA = PNORM/P5
          PAR = P5*PAR
290      CONTINUE                                     LMSR3890
300      CONTINUE                                     LMSR3900
C
C      TEST FOR SUCCESSFUL ITERATION.            LMSR3910
C
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)                                LMSR3920
          WA2(J) = DIAG(J)*X(J)
310      CONTINUE                                     LMSR3930
        DO 320 I = 1, M
          FVEC(I) = WA4(I)                                LMSR3940
320      CONTINUE                                     LMSR3950
        XNORM = ENORM(N,WA2)
        FNORM = FNORM1
        ITER = ITER + 1
330      CONTINUE                                     LMSR3960
C
C      TESTS FOR CONVERGENCE.                      LMSR3970
C
        IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL
          LMSR3980
          LMSR3990
          LMSR4000
          LMSR4010
          LMSR4020
          LMSR4030
          LMSR4040
          LMSR4050
          LMSR4060
          LMSR4070
          LMSR4080
          LMSR4090
          LMSR4100
          LMSR4110
          LMSR4120
          LMSR4130
          LMSR4140
          LMSR4150
          LMSR4160
          LMSR4170
          LMSR4180
          LMSR4190
          LMSR4200
          LMSR4210
          LMSR4220
          LMSR4230
          LMSR4240
          LMSR4250
          LMSR4260
          LMSR4270
          LMSR4280
          LMSR4290
          LMSR4300
          LMSR4310
          LMSR4320

```

```

*           .AND. P5*RATIO .LE. ONE) INFO = 1          LMSR4330
IF (DELTA .LE. XTOL*XNORM) INFO = 2          LMSR4340
IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL
*           .AND. P5*RATIO .LE. ONE .AND. INFO .EQ. 2) INFO = 3          LMSR4350
IF (INFO .NE. 0) GO TO 340          LMSR4360
C
C           TESTS FOR TERMINATION AND STRINGENT TOLERANCES.          LMSR4370
C
IF (NFEV .GE. MAXFEV) INFO = 5          LMSR4380
IF (DABS(ACTRED) .LE. EPSMCH .AND. PRERED .LE. EPSMCH
*           .AND. P5*RATIO .LE. ONE) INFO = 6          LMSR4390
IF (DELTA .LE. EPSMCH*XNORM) INFO = 7          LMSR4400
IF (GNORM .LE. EPSMCH) INFO = 8          LMSR4410
IF (INFO .NE. 0) GO TO 340          LMSR4420
C
C           END OF THE INNER LOOP. REPEAT IF ITERATION UNSUCCESSFUL.          LMSR4430
C
IF (RATIO .LT. P0001) GO TO 240          LMSR4440
C
C           END OF THE OUTER LOOP.          LMSR4450
C
GO TO 30          LMSR4460
340 CONTINUE          LMSR4470
C
C           TERMINATION, EITHER NORMAL OR USER IMPOSED.          LMSR4480
C
IF (IFLAG .LT. 0) INFO = IFLAG          LMSR4490
IFLAG = 0          LMSR4500
IF (NPRINT .GT. 0) CALL FCN(M,N,X,FVEC,WA3,IFLAG)
RETURN          LMSR4510
C
C           LAST CARD OF SUBROUTINE LMSTR.          LMSR4520
C
END          LMSR4530

```

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```

SUBROUTINE LMSTR1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,IPVT,WA,      LMS10010
*          LWA)                                              LMS10020
INTEGER M,N,LDFJAC,INFO,LWA                                         LMS10030
INTEGER IPVT(N)                                               LMS10040
DOUBLE PRECISION TOL                                         LMS10050
DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),WA(LWA)           LMS10060
EXTERNAL FCN                                              LMS10070
*****                                                       LMS10080
C
C
C SUBROUTINE LMSTR1                                         LMS10090
C
C THE PURPOSE OF LMSTR1 IS TO MINIMIZE THE SUM OF THE SQUARES OF    LMS10100
C M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF        LMS10110
C THE LEVENBERG-MARQUARDT ALGORITHM WHICH USES MINIMAL STORAGE.     LMS10120
C THIS IS DONE BY USING THE MORE GENERAL LEAST-SQUARES SOLVER       LMS10130
C LMSTR. THE USER MUST PROVIDE A SUBROUTINE WHICH CALCULATES        LMS10140
C THE FUNCTIONS AND THE ROWS OF THE JACOBIAN.                         LMS10150
C
C THE SUBROUTINE STATEMENT IS                                     LMS10160
C
C   SUBROUTINE LMSTR1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,          LMS10210
C                     IPVT,WA,LWA)                                     LMS10220
C
C WHERE                                                       LMS10230
C
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH            LMS10240
C CALCULATES THE FUNCTIONS AND THE ROWS OF THE JACOBIAN.          LMS10250
C FCN MUST BE DECLARED IN AN EXTERNAL STATEMENT IN THE           LMS10260
C USER CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.         LMS10270
C
C SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)                           LMS10280
C INTEGER M,N,IFLAG                                         LMS10290
C DOUBLE PRECISION X(N),FVEC(M),FJROW(N)                         LMS10300
C -----
C IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND                  LMS10310
C RETURN THIS VECTOR IN FVEC.                                    LMS10320
C IF IFLAG = I CALCULATE THE (I-1)-ST ROW OF THE                LMS10330
C JACOBIAN AT X AND RETURN THIS VECTOR IN FJROW.               LMS10340
C -----
C RETURN                                              LMS10350
C END                                                 LMS10360
C
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS        LMS10370
C THE USER WANTS TO TERMINATE EXECUTION OF LMSTR1.             LMS10380
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.                 LMS10390
C
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER        LMS10400
C OF FUNCTIONS.                                              LMS10410
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER        LMS10420
C OF VARIABLES. N MUST NOT EXCEED M.                            LMS10430
C
C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN            LMS10440
C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X       LMS10450
C
C

```

C CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR. LMS10550
C FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS LMS10560
C THE FUNCTIONS EVALUATED AT THE OUTPUT X. LMS10570
C FJAC IS AN OUTPUT N BY N ARRAY. THE UPPER TRIANGLE OF FJAC LMS10580
C CONTAINS AN UPPER TRIANGULAR MATRIX R SUCH THAT LMS10590
C
C T T
C P *(JAC *JAC)*P = R *R, LMS10600
C
C WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL LMS10610
C CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPV(J) LMS10620
C (SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRIANGULAR LMS10630
C PART OF FJAC CONTAINS INFORMATION GENERATED DURING LMS10640
C THE COMPUTATION OF R. LMS10650
C
C LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N LMS10660
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC. LMS10670
C
C TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS LMS10680
C WHEN THE ALGORITHM ESTIMATES EITHER THAT THE RELATIVE LMS10690
C ERROR IN THE SUM OF SQUARES IS AT MOST TOL OR THAT LMS10700
C THE RELATIVE ERROR BETWEEN X AND THE SOLUTION IS AT LMS10710
C MOST TOL. LMS10720
C
C INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS LMS10730
C TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE) LMS10740
C VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE, LMS10750
C INFO IS SET AS FOLLOWS. LMS10760
C
C INFO = 0 IMPROPER INPUT PARAMETERS. LMS10770
C
C INFO = 1 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR LMS10780
C IN THE SUM OF SQUARES IS AT MOST TOL. LMS10790
C
C INFO = 2 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR LMS10800
C BETWEEN X AND THE SOLUTION IS AT MOST TOL. LMS10810
C
C INFO = 3 CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD. LMS10820
C
C INFO = 4 FVEC IS ORTHOGONAL TO THE COLUMNS OF THE LMS10830
C JACOBIAN TO MACHINE PRECISION. LMS10840
C
C INFO = 5 NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS LMS10850
C REACHED 100*(N+1). LMS10860
C
C INFO = 6 TOL IS TOO SMALL. NO FURTHER REDUCTION IN LMS10870
C THE SUM OF SQUARES IS POSSIBLE. LMS10880
C
C INFO = 7 TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN LMS10890
C THE APPROXIMATE SOLUTION X IS POSSIBLE. LMS10900
C
C IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT LMS10910

```

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      *****
C      INTEGER MAXFEV,MODE,NFEV,NJEV,NPRINT
C      DOUBLE PRECISION FACTOR,FTOL,GTOL,XTOL,ZERO
C      DATA FACTOR,ZERO /1.0D2,0.0D0/
C      INFO = 0
C
C      CHECK THE INPUT PARAMETERS FOR ERRORS.
C
C      IF (N .LE. 0 .OR. M .LT. N .OR. LDFJAC .LT. N .OR. TOL .LT. ZERO
C      *     .OR. LWA .LT. 5*N + M) GO TO 10
C
C      CALL LMSTR.
C
C      MAXFEV = 100*(N + 1)
C      FTOL = TOL
C      XTOL = TOL
C      GTOL = ZERO
C      MODE = 1
C      NPRINT = 0
C      CALL LMSTR(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,MAXFEV,
C      *             WA(1),MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,IPVT,WA(N+1),
C      *             WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))
C      IF (INFO .EQ. 8) INFO = 4
10    CONTINUE
      RETURN
C
C      LAST CARD OF SUBROUTINE LMSTR1.
C
C      END

```

THIS PAGE
WAS THIS PAGE ALONE
WAS INTENTIONALLY
LEFT BLANK

```

SUBROUTINE QFORM(M,N,Q,LDQ,WA) QFRM0010
  INTEGER M,N,LDQ QFRM0020
  DOUBLE PRECISION Q(LDQ,M),WA(M) QFRM0030
***** QFRM0040
C   SUBROUTINE QFORM QFRM0050
C
C THIS SUBROUTINE PROCEEDS FROM THE COMPUTED QR FACTORIZATION OF QFRM0060
C AN M BY N MATRIX A TO ACCUMULATE THE M BY M ORTHOGONAL MATRIX QFRM0070
C Q FROM ITS FACTORED FORM. QFRM0080
C
C THE SUBROUTINE STATEMENT IS QFRM0090
C
C   SUBROUTINE QFORM(M,N,Q,LDQ,WA) QFRM0100
C
C WHERE QFRM0110
C
C   M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER QFRM0120
C   OF ROWS OF A AND THE ORDER OF Q. QFRM0130
C
C   N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER QFRM0140
C   OF COLUMNS OF A. QFRM0150
C
C   Q IS AN M BY M ARRAY. ON INPUT THE FULL LOWER TRAPEZOID IN QFRM0160
C   THE FIRST MIN(M,N) COLUMNS OF Q CONTAINS THE FACTORED FORM. QFRM0170
C   ON OUTPUT Q HAS BEEN ACCUMULATED INTO A SQUARE MATRIX. QFRM0180
C
C   LDQ IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M QFRM0190
C   WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY Q. QFRM0200
C
C   WA IS A WORK ARRAY OF LENGTH M. QFRM0210
C
C SUBPROGRAMS CALLED QFRM0220
C
C   FORTRAN-SUPPLIED ... MINO QFRM0230
C
C   ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. QFRM0240
C   BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE QFRM0250
C
C *****
C   INTEGER I,J,JM1,K,L,MINMN,NP1 QFRM0260
C   DOUBLE PRECISION ONE,SUM,TEMP,ZERO QFRM0270
C   DATA ONE,ZERO /1.0D0,0.0D0/ QFRM0280
C
C   ZERO OUT UPPER TRIANGLE OF Q IN THE FIRST MIN(M,N) COLUMNS. QFRM0290
C
C   MINMN = MINO(M,N) QFRM0300
C   IF (MINMN .LT. 2) GO TO 30 QFRM0310
C   DO 20 J = 2, MINMN QFRM0320
C     JM1 = J - 1 QFRM0330
C     DO 10 I = 1, JM1 QFRM0340
C       Q(I,J) = ZERO QFRM0350
C 10    CONTINUE QFRM0360
C 20    CONTINUE QFRM0370

```

```

30 CONTINUE QFRM0550
C QFRM0560
C INITIALIZE REMAINING COLUMNS TO THOSE OF THE IDENTITY MATRIX. QFRM0570
C QFRM0580
NP1 = N + 1 QFRM0590
IF (M .LT. NP1) GO TO 60 QFRM0600
DO 50 J = NP1, M QFRM0610
   DO 40 I = 1, M QFRM0620
      Q(I,J) = ZERO QFRM0630
40   CONTINUE QFRM0640
      Q(J,J) = ONE QFRM0650
50   CONTINUE QFRM0660
60 CONTINUE QFRM0670
C QFRM0680
C ACCUMULATE Q FROM ITS FACTORED FORM. QFRM0690
C QFRM0700
DO 120 L = 1, MINMN QFRM0710
   K = MINMN - L + 1 QFRM0720
   DO 70 I = K, M QFRM0730
      WA(I) = Q(I,K) QFRM0740
      Q(I,K) = ZERO QFRM0750
70   CONTINUE QFRM0760
      Q(K,K) = ONE QFRM0770
      IF (WA(K) .EQ. ZERO) GO TO 110 QFRM0780
      DO 100 J = K, M QFRM0790
         SUM = ZERO QFRM0800
         DO 80 I = K, M QFRM0810
            SUM = SUM + Q(I,J)*WA(I) QFRM0820
80   CONTINUE QFRM0830
      TEMP = SUM/WA(K) QFRM0840
      DO 90 I = K, M QFRM0850
         Q(I,J) = Q(I,J) - TEMP*WA(I) QFRM0860
90   CONTINUE QFRM0870
100  CONTINUE QFRM0880
110  CONTINUE QFRM0890
120  CONTINUE QFRM0900
RETURN QFRM0910
C QFRM0920
C LAST CARD OF SUBROUTINE QFORM. QFRM0930
C QFRM0940
END QFRM0950

```

SUBROUTINE QRFAC(M,N,A,LDA,PIVOT,IPVT,LIPVT,RDIAG,ACNORM,WA)	QRFA0010
INTEGER M,N,LDA,LIPVT	QRFA0020
INTEGER IPVT(LIPVT)	QRFA0030
LOGICAL PIVOT	QRFA0040
DOUBLE PRECISION A(LDA,N),RDIAG(N),ACNORM(N),WA(N)	QRFA0050
*****	QRFA0060
C SUBROUTINE QRFAC	QRFA0070
C THIS SUBROUTINE USES HOUSEHOLDER TRANSFORMATIONS WITH COLUMN	QRFA0080
C PIVOTING (OPTIONAL) TO COMPUTE A QR FACTORIZATION OF THE	QRFA0090
C M BY N MATRIX A. THAT IS, QRFAC DETERMINES AN ORTHOGONAL	QRFA0100
C MATRIX Q, A PERMUTATION MATRIX P, AND AN UPPER TRAPEZOIDAL	QRFA0110
C MATRIX R WITH DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE,	QRFA0120
C SUCH THAT $A^*P = Q^*R$. THE HOUSEHOLDER TRANSFORMATION FOR	QRFA0130
C COLUMN K, $K = 1, 2, \dots, \min(M, N)$, IS OF THE FORM	QRFA0140
C	QRFA0150
C T	QRFA0160
C I - (1/U(K)) ^T U ^T U	QRFA0170
C WHERE U HAS ZEROS IN THE FIRST K-1 POSITIONS. THE FORM OF	QRFA0180
C THIS TRANSFORMATION AND THE METHOD OF PIVOTING FIRST	QRFA0190
C APPEARED IN THE CORRESPONDING LINPACK SUBROUTINE.	QRFA0200
C THE SUBROUTINE STATEMENT IS	QRFA0210
C SUBROUTINE QRFAC(M,N,A,LDA,PIVOT,IPVT,LIPVT,RDIAG,ACNORM,WA)	QRFA0220
C WHERE	QRFA0230
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	QRFA0240
C OF ROWS OF A.	QRFA0250
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	QRFA0260
C OF COLUMNS OF A.	QRFA0270
C A IS AN M BY N ARRAY. ON INPUT A CONTAINS THE MATRIX FOR	QRFA0280
C WHICH THE QR FACTORIZATION IS TO BE COMPUTED. ON OUTPUT	QRFA0290
C THE STRICT UPPER TRAPEZOIDAL PART OF A CONTAINS THE STRICT	QRFA0300
C UPPER TRAPEZOIDAL PART OF R, AND THE LOWER TRAPEZOIDAL	QRFA0310
C PART OF A CONTAINS A FACTORED FORM OF Q (THE NON-TRIVIAL	QRFA0320
C ELEMENTS OF THE U VECTORS DESCRIBED ABOVE).	QRFA0330
C LDA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M	QRFA0340
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY A.	QRFA0350
C PIVOT IS A LOGICAL INPUT VARIABLE. IF PIVOT IS SET TRUE,	QRFA0360
C THEN COLUMN PIVOTING IS ENFORCED. IF PIVOT IS SET FALSE,	QRFA0370
C THEN NO COLUMN PIVOTING IS DONE.	QRFA0380
C IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH LIPVT. IPVT	QRFA0390
C DEFINES THE PERMUTATION MATRIX P SUCH THAT $A^*P = Q^*R$.	QRFA0400
C COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.	QRFA0410
C IF PIVOT IS FALSE, IPVT IS NOT REFERENCED.	QRFA0420
C	QRFA0430
C	QRFA0440
C	QRFA0450
C	QRFA0460
C	QRFA0470
C	QRFA0480
C	QRFA0490
C	QRFA0500
C	QRFA0510
C	QRFA0520
C	QRFA0530
C	QRFA0540

```

C LIPVT IS A POSITIVE INTEGER INPUT VARIABLE. IF PIVOT IS FALSE, QRFA0550
C THEN LIPVT MAY BE AS SMALL AS 1. IF PIVOT IS TRUE, THEN QRFA0560
C LIPVT MUST BE AT LEAST N. QRFA0570
C QRFA0580
C QRFA0590
C RDIAG IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE QRFA0600
C DIAGONAL ELEMENTS OF R. QRFA0610
C QRFA0620
C ACNORM IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE QRFA0630
C NORMS OF THE CORRESPONDING COLUMNS OF THE INPUT MATRIX A. QRFA0640
C IF THIS INFORMATION IS NOT NEEDED, THEN ACNORM CAN COINCIDE QRFA0650
C WITH RDIAG. QRFA0660
C QRFA0670
C WA IS A WORK ARRAY OF LENGTH N. IF PIVOT IS FALSE, THEN WA QRFA0680
C CAN COINCIDE WITH RDIAG. QRFA0690
C QRFA0700
C SUBPROGRAMS CALLED QRFA0710
C QRFA0720
C MINPACK-SUPPLIED ... DPMPAR,ENORM QRFA0730
C QRFA0740
C FORTRAN-SUPPLIED ... DMAX1,DSQRT,MIN0 QRFA0750
C QRFA0760
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. QRFA0770
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE QRFA0780
C QRFA0790
C *****
C INTEGER I,J,JP1,K,KMAX,MINMN QRFA0800
C DOUBLE PRECISION AJNORM,EPSMCH,ONE,P05,SUM,TEMP,ZERO QRFA0810
C DOUBLE PRECISION DPMPAR,ENORM QRFA0820
C DATA ONE,P05,ZERO /1.0D0,5.0D-2,0.0D0/ QRFA0830
C QRFA0840
C QRFA0850
C EPSMCH IS THE MACHINE PRECISION. QRFA0860
C QRFA0870
C EPSMCH = DPMPAR(1) QRFA0880
C QRFA0890
C COMPUTE THE INITIAL COLUMN NORMS AND INITIALIZE SEVERAL ARRAYS. QRFA0900
C QRFA0910
C DO 10 J = 1, N QRFA0920
C   ACNORM(J) = ENORM(M,A(1,J)) QRFA0930
C   RDIAG(J) = ACNORM(J) QRFA0940
C   WA(J) = RDIAG(J) QRFA0950
C   IF (PIVOT) IPVT(J) = J QRFA0960
C 10  CONTINUE QRFA0970
C QRFA0980
C REDUCE A TO R WITH HOUSEHOLDER TRANSFORMATIONS. QRFA0990
C QRFA1000
C MINMN = MIN0(M,N) QRFA1010
C DO 110 J = 1, MINMN QRFA1020
C   IF (.NOT.PIVOT) GO TO 40 QRFA1030
C QRFA1040
C BRING THE COLUMN OF LARGEST NORM INTO THE PIVOT POSITION. QRFA1050
C QRFA1060
C KMAX = J QRFA1070
C DO 20 K = J, N QRFA1080

```

```

      IF (RDIAG(K) .GT. RDIAG(KMAX)) KMAX = K           QRFA1090
20    CONTINUE                                         QRFA1100
      IF (KMAX .EQ. J) GO TO 40                         QRFA1110
      DO 30 I = 1, M                                     QRFA1120
          TEMP = A(I,J)
          A(I,J) = A(I,KMAX)
          A(I,KMAX) = TEMP
30    CONTINUE                                         QRFA1130
      RDIAG(KMAX) = RDIAG(J)                           QRFA1140
      WA(KMAX) = WA(J)                                 QRFA1150
      K = IPVT(J)                                    QRFA1160
      IPVT(J) = IPVT(KMAX)
      IPVT(KMAX) = K
40    CONTINUE                                         QRFA1170
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))                     QRFA1230
      IF (AJNORM .EQ. ZERO) GO TO 100                  QRFA1240
      IF (A(J,J) .LT. ZERO) AJNORM = -AJNORM          QRFA1250
      DO 50 I = J, M                                   QRFA1260
          A(I,J) = A(I,J)/AJNORM                      QRFA1270
50    CONTINUE                                         QRFA1280
      A(J,J) = A(J,J) + ONE                          QRFA1290
C
C APPLY THE TRANSFORMATION TO THE REMAINING COLUMNS
C AND UPDATE THE NORMS.
C
      JP1 = J + 1                                     QRFA1320
      IF (N .LT. JP1) GO TO 100                      QRFA1330
      DO 90 K = JP1, N                               QRFA1340
          SUM = ZERO
          DO 60 I = J, M
              SUM = SUM + A(I,J)*A(I,K)
60    CONTINUE                                         QRFA1350
          TEMP = SUM/A(J,J)
          DO 70 I = J, M
              A(I,K) = A(I,K) - TEMP*A(I,J)
70    CONTINUE                                         QRFA1360
      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                                         QRFA1370
90    CONTINUE                                         QRFA1380
100   CONTINUE                                         QRFA1390
      RDIAG(J) = -AJNORM                            QRFA1400
110   CONTINUE                                         QRFA1410
      RETURN                                           QRFA1420
C
C LAST CARD OF SUBROUTINE QRFAC.                   QRFA1430

```

```

SUBROUTINE QRSOLV(N,R,LDR,IPVT,DIAG,QTB,X,SDIAG,WA) QRSL0010
INTEGER N,LDR QRSL0020
INTEGER IPVT(N) QRSL0030
DOUBLE PRECISION R(LDR,N),DIAG(N),QTB(N),X(N),SDIAG(N),WA(N) QRSL0040
***** QRSL0050
C QRSL0060
C SUBROUTINE QRSOLV QRSL0070
C QRSL0080
C GIVEN AN M BY N MATRIX A, AN N BY N DIAGONAL MATRIX D, QRSL0090
C AND AN M-VECTOR B, THE PROBLEM IS TO DETERMINE AN X WHICH QRSL0100
C SOLVES THE SYSTEM QRSL0110
C QRSL0120
C A*X = B , QRSL0130
C D*X = 0 , QRSL0140
C IN THE LEAST SQUARES SENSE. QRSL0150
C QRSL0160
C THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM QRSL0170
C IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE QRSL0180
C QR FACTORIZATION, WITH COLUMN PIVOTING, OF A. THAT IS, IF QRSL0190
C A*P = Q*R, WHERE P IS A PERMUTATION MATRIX, Q HAS ORTHOGONAL QRSL0200
C COLUMNS, AND R IS AN UPPER TRIANGULAR MATRIX WITH DIAGONAL QRSL0210
C ELEMENTS OF NONINCREASING MAGNITUDE, THEN QRSOLV EXPECTS QRSL0220
C THE FULL UPPER TRIANGLE OF R, THE PERMUTATION MATRIX P, QRSL0230
C AND THE FIRST N COMPONENTS OF (Q TRANSPOSE)*B. THE SYSTEM QRSL0240
C A*X = B, D*X = 0, IS THEN EQUIVALENT TO QRSL0250
C QRSL0260
C T T
C R*Z = Q *B , P *D*P*Z = 0 , QRSL0270
C QRSL0280
C QRSL0290
C WHERE X = P*Z. IF THIS SYSTEM DOES NOT HAVE FULL RANK, QRSL0300
C THEN A LEAST SQUARES SOLUTION IS OBTAINED. ON OUTPUT QRSOLV QRSL0310
C ALSO PROVIDES AN UPPER TRIANGULAR MATRIX S SUCH THAT QRSL0320
C QRSL0330
C T T T
C P *(A *A + D*D)*P = S *S . QRSL0340
C QRSL0350
C QRSL0360
C S IS COMPUTED WITHIN QRSOLV AND MAY BE OF SEPARATE INTEREST. QRSL0370
C QRSL0380
C THE SUBROUTINE STATEMENT IS QRSL0390
C QRSL0400
C SUBROUTINE QRSOLV(N,R,LDR,IPVT,DIAG,QTB,X,SDIAG,WA) QRSL0410
C QRSL0420
C WHERE QRSL0430
C QRSL0440
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R. QRSL0450
C QRSL0460
C R IS AN N BY N ARRAY. ON INPUT THE FULL UPPER TRIANGLE QRSL0470
C MUST CONTAIN THE FULL UPPER TRIANGLE OF THE MATRIX R. QRSL0480
C ON OUTPUT THE FULL UPPER TRIANGLE IS UNALTERED, AND THE QRSL0490
C STRICT LOWER TRIANGLE CONTAINS THE STRICT UPPER TRIANGLE QRSL0500
C (TRANSPOSED) OF THE UPPER TRIANGULAR MATRIX S. QRSL0510
C QRSL0520
C LDR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N QRSL0530
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY R. QRSL0540

```

```

C IPVT IS AN INTEGER INPUT ARRAY OF LENGTH N WHICH DEFINES THE QRSL0550
C PERMUTATION MATRIX P SUCH THAT A*P = Q*R. COLUMN J OF P QRSL0560
C IS COLUMN IPVT(J) OF THE IDENTITY MATRIX. QRSL0570
C
C DIAG IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE QRSL0580
C DIAGONAL ELEMENTS OF THE MATRIX D. QRSL0590
C
C QTB IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE FIRST QRSL0600
C N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*B. QRSL0610
C
C X IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE LEAST QRSL0620
C SQUARES SOLUTION OF THE SYSTEM A*X = B, D*X = 0. QRSL0630
C
C SDIAG IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE QRSL0640
C DIAGONAL ELEMENTS OF THE UPPER TRIANGULAR MATRIX S. QRSL0650
C
C WA IS A WORK ARRAY OF LENGTH N. QRSL0660
C
C SUBPROGRAMS CALLED QRSL0670
C
C FORTRAN-SUPPLIED ... DABS,DSQRT QRSL0680
C
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. QRSL0690
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE QRSL0700
C
C *****
C INTEGER I,J,JP1,K,KP1,L,NSING QRSL0710
C DOUBLE PRECISION COS,COTAN,P5,P25,QTBPJ,SIN,SUM,TAN,TEMP,ZERO QRSL0720
C DATA P5,P25,ZERO /5.0D-1,2.5D-1,0.0D0/ QRSL0730
C
C COPY R AND (Q TRANSPOSE)*B TO PRESERVE INPUT AND INITIALIZE S. QRSL0740
C IN PARTICULAR, SAVE THE DIAGONAL ELEMENTS OF R IN X. QRSL0750
C
C DO 20 J = 1, N QRSL0760
C   DO 10 I = J, N QRSL0770
C     R(I,J) = R(J,I) QRSL0780
C   10  CONTINUE QRSL0790
C     X(J) = R(J,J) QRSL0800
C     WA(J) = QTBPJ(J) QRSL0810
C   20  CONTINUE QRSL0820
C
C ELIMINATE THE DIAGONAL MATRIX D USING A GIVENS ROTATION. QRSL0830
C
C DO 100 J = 1, N QRSL0840
C
C   PREPARE THE ROW OF D TO BE ELIMINATED, LOCATING THE QRSL0850
C   DIAGONAL ELEMENT USING P FROM THE QR FACTORIZATION. QRSL0860
C
C   L = IPVT(J) QRSL0870
C   IF (DIAG(L) .EQ. ZERO) GO TO 90 QRSL0880
C   DO 30 K = J, N QRSL0890
C     SDIAG(K) = ZERO QRSL0900
C   30  CONTINUE QRSL0910

```

```

C SDIAG(J) = DIAG(L) QRSL1090
C THE TRANSFORMATIONS TO ELIMINATE THE ROW OF D QRSL1100
C MODIFY ONLY A SINGLE ELEMENT OF (Q TRANSPOSE)*B QRSL1110
C BEYOND THE FIRST N, WHICH IS INITIALLY ZERO. QRSL1120
C
C QTBPJ = ZERO QRSL1130
DO 80 K = J, N QRSL1140
C
C DETERMINE A GIVENS ROTATION WHICH ELIMINATES THE QRSL1150
C APPROPRIATE ELEMENT IN THE CURRENT ROW OF D. QRSL1160
C
C IF (SDIAG(K) .EQ. ZERO) GO TO 70 QRSL1170
IF (DABS(R(K,K)) .GE. DABS(SDIAG(K))) GO TO 40 QRSL1180
COTAN = R(K,K)/SDIAG(K) QRSL1190
SIN = P5/DSQRT(P25+P25*COTAN**2) QRSL1200
COS = SIN*COTAN QRSL1210
GO TO 50 QRSL1220
40 CONTINUE QRSL1230
TAN = SDIAG(K)/R(K,K) QRSL1240
COS = P5/DSQRT(P25+P25*TAN**2) QRSL1250
SIN = COS*TAN QRSL1260
50 CONTINUE QRSL1270
C COMPUTE THE MODIFIED DIAGONAL ELEMENT OF R AND QRSL1280
C THE MODIFIED ELEMENT OF ((Q TRANSPOSE)*B,0). QRSL1290
C
C R(K,K) = COS*R(K,K) + SIN*SDIAG(K) QRSL1300
TEMP = COS*WA(K) + SIN*QTBPJ QRSL1310
QTBPJ = -SIN*WA(K) + COS*QTBPJ QRSL1320
WA(K) = TEMP QRSL1330
C ACCUMULATE THE TRANFORMATION IN THE ROW OF S. QRSL1340
C
C KP1 = K + 1 QRSL1350
IF (N .LT. KP1) GO TO 70 QRSL1360
DO 60 I = KP1, N QRSL1370
TEMP = COS*R(I,K) + SIN*SDIAG(I) QRSL1380
SDIAG(I) = -SIN*R(I,K) + COS*SDIAG(I) QRSL1390
R(I,K) = TEMP QRSL1400
60 CONTINUE QRSL1410
70 CONTINUE QRSL1420
80 CONTINUE QRSL1430
90 CONTINUE QRSL1440
C STORE THE DIAGONAL ELEMENT OF S AND RESTORE QRSL1450
C THE CORRESPONDING DIAGONAL ELEMENT OF R. QRSL1460
C
C SDIAG(J) = R(J,J) QRSL1470
R(J,J) = X(J) QRSL1480
100 CONTINUE QRSL1490
C SOLVE THE TRIANGULAR SYSTEM FOR Z. IF THE SYSTEM IS QRSL1500
C SINGULAR, THEN OBTAIN A LEAST SQUARES SOLUTION. QRSL1510
C
C QRSL1520
C QRSL1530
C QRSL1540
C QRSL1550
C QRSL1560
C QRSL1570
C QRSL1580
C QRSL1590
C QRSL1600
C QRSL1610
C QRSL1620

```

```

C
NSING = N                               QRSL1630
DO 110 J = 1, N                         QRSL1640
  IF (SDIAG(J) .EQ. ZERO .AND. NSING .EQ. N) NSING = J - 1
  IF (NSING .LT. N) WA(J) = ZERO
110  CONTINUE                             QRSL1650
    IF (NSING .LT. 1) GO TO 150
    DO 140 K = 1, NSING                  QRSL1660
      J = NSING - K + 1                 QRSL1670
      SUM = ZERO                         QRSL1680
      JP1 = J + 1                        QRSL1690
      IF (NSING .LT. JP1) GO TO 130
      DO 120 I = JP1, NSING              QRSL1700
        SUM = SUM + R(I,J)*WA(I)
120  CONTINUE                             QRSL1710
130  CONTINUE                             QRSL1720
  WA(J) = (WA(J) - SUM)/SDIAG(J)         QRSL1730
140  CONTINUE                             QRSL1740
150 CONTINUE                             QRSL1750
QRSL1760
C
C      PERMUTE THE COMPONENTS OF Z BACK TO COMPONENTS OF X.
C
DO 160 J = 1, N                         QRSL1770
  L = IPVT(J)                           QRSL1780
  X(L) = WA(J)                          QRSL1790
160  CONTINUE                             QRSL1800
RETURN                                  QRSL1810
QRSL1820
C
C      LAST CARD OF SUBROUTINE QRSOLV.
C
END                                     QRSL1830
QRSL1840
QRSL1850
QRSL1860
QRSL1870
QRSL1880
QRSL1890
QRSL1900
QRSL1910
QRSL1920
QRSL1930

```

```

SUBROUTINE RWUPDT(N,R,LDR,W,B,ALPHA,COS,SIN) RWUP0010
  INTEGER N,LDR RWUP0020
  DOUBLE PRECISION ALPHA RWUP0030
  DOUBLE PRECISION R(LDR,N),W(N),B(N),COS(N),SIN(N) RWUP0040
***** RWUP0050
C SUBROUTINE RWUPDT RWUP0060
C GIVEN AN N BY N UPPER TRIANGULAR MATRIX R, THIS SUBROUTINE RWUP0070
C COMPUTES THE QR DECOMPOSITION OF THE MATRIX FORMED WHEN A ROW RWUP0080
C IS ADDED TO R. IF THE ROW IS SPECIFIED BY THE VECTOR W, THEN RWUP0090
C RWUPDT DETERMINES AN ORTHOGONAL MATRIX Q SUCH THAT WHEN THE RWUP0100
C N+1 BY N MATRIX COMPOSED OF R AUGMENTED BY W IS PREMULTIPLIED RWUP0110
C BY (Q TRANSPOSE), THE RESULTING MATRIX IS UPPER TRAPEZOIDAL. RWUP0120
C THE MATRIX (Q TRANSPOSE) IS THE PRODUCT OF N TRANSFORMATIONS RWUP0130
C RWUP0140
C G(N)*G(N-1)* ... *G(1) RWUP0150
C WHERE G(I) IS A GIVENS ROTATION IN THE (I,N+1) PLANE WHICH RWUP0160
C ELIMINATES ELEMENTS IN THE (N+1)-ST PLANE. RWUPDT ALSO RWUP0170
C COMPUTES THE PRODUCT (Q TRANSPOSE)*C WHERE C IS THE RWUP0180
C (N+1)-VECTOR (B,ALPHA). Q ITSELF IS NOT ACCUMULATED, RATHER RWUP0190
C THE INFORMATION TO RECOVER THE G ROTATIONS IS SUPPLIED. RWUP0200
C THE SUBROUTINE STATEMENT IS RWUP0210
C SUBROUTINE RWUPDT(N,R,LDR,W,B,ALPHA,COS,SIN) RWUP0220
C WHERE RWUP0230
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R. RWUP0240
C RWUP0250
C R IS AN N BY N ARRAY. ON INPUT THE UPPER TRIANGULAR PART OF RWUP0260
C R MUST CONTAIN THE MATRIX TO BE UPDATED. ON OUTPUT R RWUP0270
C CONTAINS THE UPDATED TRIANGULAR MATRIX. RWUP0280
C RWUP0290
C LDR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N RWUP0300
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY R. RWUP0310
C RWUP0320
C W IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE ROW RWUP0330
C VECTOR TO BE ADDED TO R. RWUP0340
C RWUP0350
C B IS AN ARRAY OF LENGTH N. ON INPUT B MUST CONTAIN THE RWUP0360
C FIRST N ELEMENTS OF THE VECTOR C. ON OUTPUT B CONTAINS RWUP0370
C THE FIRST N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*C. RWUP0380
C RWUP0390
C ALPHA IS A VARIABLE. ON INPUT ALPHA MUST CONTAIN THE RWUP0400
C (N+1)-ST ELEMENT OF THE VECTOR C. ON OUTPUT ALPHA CONTAINS RWUP0410
C THE (N+1)-ST ELEMENT OF THE VECTOR (Q TRANSPOSE)*C. RWUP0420
C RWUP0430
C COS IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE RWUP0440
C COSINES OF THE TRANSFORMING GIVENS ROTATIONS. RWUP0450
C RWUP0460
C SIN IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE RWUP0470
C RWUP0480
C RWUP0490
C RWUP0500
C RWUP0510
C RWUP0520
C RWUP0530
C RWUP0540

```

```

C      SINES OF THE TRANSFORMING GIVENS ROTATIONS.          RWUP0550
C
C      SUBPROGRAMS CALLED                               RWUP0560
C
C      FORTRAN-SUPPLIED ... DABS,DSQRT                 RWUP0570
C
C      ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.   RWUP0580
C      BURTON S. GARBOW, DUDLEY V. GOETSCHEL, KENNETH E. HILLSTROM,   RWUP0590
C      JORGE J. MORE                                     RWUP0600
C
C      *****
C      INTEGER I,J,JM1                                 RWUP0610
C      DOUBLE PRECISION COTAN,ONE,P5,P25,ROWJ,TAN,TEMP,ZERO   RWUP0620
C      DATA ONE,P5,P25,ZERO /1.0D0,5.0D-1,2.5D-1,0.0D0/   RWUP0630
C
C      DO 60 J = 1, N                                RWUP0640
C          ROWJ = W(J)                               RWUP0650
C          JM1 = J - 1                             RWUP0660
C
C      APPLY THE PREVIOUS TRANSFORMATIONS TO        RWUP0670
C      R(I,J), I=1,2,...,J-1, AND TO W(J).         RWUP0680
C
C      IF (JM1 .LT. 1) GO TO 20                      RWUP0690
C      DO 10 I = 1, JM1                         RWUP0700
C          TEMP = COS(I)*R(I,J) + SIN(I)*ROWJ    RWUP0710
C          ROWJ = -SIN(I)*R(I,J) + COS(I)*ROWJ   RWUP0720
C          R(I,J) = TEMP                          RWUP0730
C      10    CONTINUE                           RWUP0740
C      20    CONTINUE                           RWUP0750
C
C      DETERMINE A GIVENS ROTATION WHICH ELIMINATES W(J).   RWUP0760
C
C      COS(J) = ONE                                RWUP0770
C      SIN(J) = ZERO                               RWUP0780
C      IF (ROWJ .EQ. ZERO) GO TO 50                RWUP0790
C      IF (DABS(R(J,J)) .GE. DABS(ROWJ)) GO TO 30   RWUP0800
C          COTAN = R(J,J)/ROWJ                   RWUP0810
C          SIN(J) = P5/DSQRT(P25+P25*COTAN**2)   RWUP0820
C          COS(J) = SIN(J)*COTAN                RWUP0830
C          GO TO 40                                RWUP0840
C      30    CONTINUE                           RWUP0850
C          TAN = ROWJ/R(J,J)                     RWUP0860
C          COS(J) = P5/DSQRT(P25+P25*TAN**2)   RWUP0870
C          SIN(J) = COS(J)*TAN                  RWUP0880
C      40    CONTINUE                           RWUP0890
C
C      APPLY THE CURRENT TRANSFORMATION TO R(J,J), B(J), AND ALPHA.   RWUP0900
C
C      R(J,J) = COS(J)*R(J,J) + SIN(J)*ROWJ   RWUP0910
C      TEMP = COS(J)*B(J) + SIN(J)*ALPHA     RWUP0920
C      ALPHA = -SIN(J)*B(J) + COS(J)*ALPHA   RWUP0930
C      B(J) = TEMP                            RWUP0940
C      50    CONTINUE                           RWUP0950
C      60    CONTINUE                           RWUP0960

```

RETURN RWUP1090
C RWUP1100
C LAST CARD OF SUBROUTINE RWUPDT. RWUP1110
C RWUP1120
END RWUP1130

DO NOT FOLD
THIS PAGE
**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

SUBROUTINE R1MPYQ(M,N,A,LDA,V,W)	R1MQ0010
INTEGER M,N,LDA	R1MQ0020
DOUBLE PRECISION A(LDA,N),V(N),W(N)	R1MQ0030
*****	R1MQ0040
C	R1MQ0050
C SUBROUTINE R1MPYQ	R1MQ0060
C	R1MQ0070
C GIVEN AN M BY N MATRIX A, THIS SUBROUTINE COMPUTES A*Q WHERE	R1MQ0080
C Q IS THE PRODUCT OF 2*(N - 1) TRANSFORMATIONS	R1MQ0090
C	R1MQ0100
C GV(N-1)*...*GV(1)*GW(1)*...*GW(N-1)	R1MQ0110
C	R1MQ0120
C AND GV(I), GW(I) ARE GIVENS ROTATIONS IN THE (I,N) PLANE WHICH	R1MQ0130
C ELIMINATE ELEMENTS IN THE I-TH AND N-TH PLANES, RESPECTIVELY.	R1MQ0140
C Q ITSELF IS NOT GIVEN, RATHER THE INFORMATION TO RECOVER THE	R1MQ0150
C GV, GW ROTATIONS IS SUPPLIED.	R1MQ0160
C	R1MQ0170
C THE SUBROUTINE STATEMENT IS	R1MQ0180
C	R1MQ0190
C SUBROUTINE R1MPYQ(M,N,A,LDA,V,W)	R1MQ0200
C	R1MQ0210
C WHERE	R1MQ0220
C	R1MQ0230
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	R1MQ0240
C OF ROWS OF A.	R1MQ0250
C	R1MQ0260
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	R1MQ0270
C OF COLUMNS OF A.	R1MQ0280
C	R1MQ0290
C A IS AN M BY N ARRAY. ON INPUT A MUST CONTAIN THE MATRIX	R1MQ0300
C TO BE POSTMULTIPLIED BY THE ORTHOGONAL MATRIX Q	R1MQ0310
C DESCRIBED ABOVE. ON OUTPUT A*Q HAS REPLACED A.	R1MQ0320
C	R1MQ0330
C LDA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M	R1MQ0340
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY A.	R1MQ0350
C	R1MQ0360
C V IS AN INPUT ARRAY OF LENGTH N. V(I) MUST CONTAIN THE	R1MQ0370
C INFORMATION NECESSARY TO RECOVER THE GIVENS ROTATION GV(I)	R1MQ0380
C DESCRIBED ABOVE.	R1MQ0390
C	R1MQ0400
C W IS AN INPUT ARRAY OF LENGTH N: W(I) MUST CONTAIN THE	R1MQ0410
C INFORMATION NECESSARY TO RECOVER THE GIVENS ROTATION GW(I)	R1MQ0420
C DESCRIBED ABOVE.	R1MQ0430
C	R1MQ0440
C SUBROUTINES CALLED	R1MQ0450
C	R1MQ0460
C FORTRAN-SUPPLIED ... DABS, DSQRT	R1MQ0470
C	R1MQ0480
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.	R1MQ0490
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE	R1MQ0500
C	R1MQ0510
C *****	R1MQ0520
C INTEGER I,J,NMJ,NM1	R1MQ0530
C DOUBLE PRECISION COS,ONE,SIN,TEMP	R1MQ0540

```

DATA ONE /1.0D0/ R1MQ0550
C R1MQ0560
C APPLY THE FIRST SET OF GIVENS ROTATIONS TO A. R1MQ0570
C R1MQ0580
NM1 = N - 1 R1MQ0590
IF (NM1 .LT. 1) GO TO 50 R1MQ0600
DO 20 NMJ = 1, NM1 R1MQ0610
   J = N - NMJ R1MQ0620
   IF (DABS(V(J)) .GT. ONE) COS = ONE/V(J) R1MQ0630
   IF (DABS(V(J)) .GT. ONE) SIN = DSQRT(ONE-COS**2) R1MQ0640
   IF (DABS(V(J)) .LE. ONE) SIN = V(J) R1MQ0650
   IF (DABS(V(J)) .LE. ONE) COS = DSQRT(ONE-SIN**2) R1MQ0660
   DO 10 I = 1, M R1MQ0670
      TEMP = COS*A(I,J) - SIN*A(I,N) R1MQ0680
      A(I,N) = SIN*A(I,J) + COS*A(I,N) R1MQ0690
      A(I,J) = TEMP R1MQ0700
10    CONTINUE R1MQ0710
20    CONTINUE R1MQ0720
C R1MQ0730
C APPLY THE SECOND SET OF GIVENS ROTATIONS TO A. R1MQ0740
C R1MQ0750
DO 40 J = 1, NM1 R1MQ0760
   IF (DABS(W(J)) .GT. ONE) COS = ONE/W(J) R1MQ0770
   IF (DABS(W(J)) .GT. ONE) SIN = DSQRT(ONE-COS**2) R1MQ0780
   IF (DABS(W(J)) .LE. ONE) SIN = W(J) R1MQ0790
   IF (DABS(W(J)) .LE. ONE) COS = DSQRT(ONE-SIN**2) R1MQ0800
   DO 30 I = 1, M R1MQ0810
      TEMP = COS*A(I,J) + SIN*A(I,N) R1MQ0820
      A(I,N) = -SIN*A(I,J) + COS*A(I,N) R1MQ0830
      A(I,J) = TEMP R1MQ0840
30    CONTINUE R1MQ0850
40    CONTINUE R1MQ0860
50 CONTINUE R1MQ0870
RETURN R1MQ0880
C R1MQ0890
C LAST CARD OF SUBROUTINE R1MPYQ. R1MQ0900
C R1MQ0910
END R1MQ0920

```

```

SUBROUTINE R1UPDT(M,N,S,LS,U,V,W,SING)          R1UP0010
INTEGER M,N,LS                                  R1UP0020
LOGICAL SING                                     R1UP0030
DOUBLE PRECISION S(LS),U(M),V(N),W(M)          R1UP0040
*****                                           R1UP0050
C                                               R1UP0060
C                                               R1UP0070
C                                               R1UP0080
C                                               R1UP0090
C                                               R1UP0100
C                                               R1UP0110
C                                               R1UP0120
C                                               R1UP0130
C                                               R1UP0140
C                                               R1UP0150
C IS AGAIN LOWER TRAPEZOIDAL.                  R1UP0160
C                                               R1UP0170
C THIS SUBROUTINE DETERMINES Q AS THE PRODUCT OF 2*(N - 1) R1UP0180
C TRANSFORMATIONS                               R1UP0190
C                                               R1UP0200
C                                               R1UP0210
C                                               R1UP0220
C WHERE GV(I), GW(I) ARE GIVENS ROTATIONS IN THE (I,N) PLANE R1UP0230
C WHICH ELIMINATE ELEMENTS IN THE I-TH AND N-TH PLANES,      R1UP0240
C RESPECTIVELY. Q ITSELF IS NOT ACCUMULATED, RATHER THE      R1UP0250
C INFORMATION TO RECOVER THE GV, GW ROTATIONS IS RETURNED.    R1UP0260
C                                               R1UP0270
C THE SUBROUTINE STATEMENT IS                  R1UP0280
C                                               R1UP0290
C SUBROUTINE R1UPDT(M,N,S,LS,U,V,W,SING)          R1UP0300
C WHERE                                         R1UP0310
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER R1UP0320
C OF ROWS OF S.                                 R1UP0330
C                                               R1UP0340
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER R1UP0350
C OF COLUMNS OF S. N MUST NOT EXCEED M.        R1UP0360
C                                               R1UP0370
C S IS AN ARRAY OF LENGTH LS. ON INPUT S MUST CONTAIN THE LOWER R1UP0380
C TRAPEZOIDAL MATRIX S STORED BY COLUMNS. ON OUTPUT S CONTAINS R1UP0390
C THE LOWER TRAPEZOIDAL MATRIX PRODUCED AS DESCRIBED ABOVE.   R1UP0400
C                                               R1UP0410
C LS IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN     R1UP0420
C (N*(2*M-N+1))/2.                                R1UP0430
C                                               R1UP0440
C U IS AN INPUT ARRAY OF LENGTH M WHICH MUST CONTAIN THE     R1UP0450
C VECTOR U.                                       R1UP0460
C                                               R1UP0470
C V IS AN ARRAY OF LENGTH N. ON INPUT V MUST CONTAIN THE VECTOR R1UP0480
C V. ON OUTPUT V(I) CONTAINS THE INFORMATION NECESSARY TO     R1UP0490
C RECOVER THE GIVENS ROTATION GV(I) DESCRIBED ABOVE.         R1UP0500
C                                               R1UP0510
C W IS AN OUTPUT ARRAY OF LENGTH M. W(I) CONTAINS INFORMATION R1UP0520
C                                         R1UP0530
C                                         R1UP0540

```

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 *****
 C INTEGER I,J,JJ,L,NMJ,NM1
 C DOUBLE PRECISION COS,COTAN,GIANT,ONE,P5,P25,SIN,TAN,TAU,TEMP,
 * ZERO
 C DOUBLE PRECISION DPMPAR
 C DATA ONE,P5,P25,ZERO /1.0D0,5.0D-1,2.5D-1,0.0D0/
 C
 C GIANT IS THE LARGEST MAGNITUDE.
 C
 C GIANT = DPMPAR(3)
 C
 C INITIALIZE THE DIAGONAL ELEMENT POINTER.
 C
 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
 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
 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.

```

IF (DABS(V(N)) .GE. DABS(V(J))) GO TO 20 R1UP1090
COTAN = V(N)/V(J) R1UP1100
SIN = P5/DSQRT(P25+P25*COTAN**2) R1UP1110
COS = SIN*COTAN R1UP1120
TAU = ONE R1UP1130
IF (DABS(COS)*GIANT .GT. ONE) TAU = ONE/COS R1UP1140
GO TO 30 R1UP1150
20 CONTINUE R1UP1160
TAN = V(J)/V(N) R1UP1170
COS = P5/DSQRT(P25+P25*TAN**2) R1UP1180
SIN = COS*TAN R1UP1190
TAU = SIN R1UP1200
30 CONTINUE R1UP1210
C R1UP1220
C APPLY THE TRANSFORMATION TO V AND STORE THE INFORMATION R1UP1230
C NECESSARY TO RECOVER THE GIVENS ROTATION. R1UP1240
C R1UP1250
V(N) = SIN*V(J) + COS*V(N) R1UP1260
V(J) = TAU R1UP1270
C R1UP1280
C APPLY THE TRANSFORMATION TO S AND EXTEND THE SPIKE IN W. R1UP1290
C R1UP1300
L = JJ R1UP1310
DO 40 I = J, M R1UP1320
TEMP = COS*S(L) - SIN*W(I) R1UP1330
W(I) = SIN*S(L) + COS*W(I) R1UP1340
S(L) = TEMP R1UP1350
L = L + 1 R1UP1360
40 CONTINUE R1UP1370
50 CONTINUE R1UP1380
60 CONTINUE R1UP1390
70 CONTINUE R1UP1400
C R1UP1410
C ADD THE SPIKE FROM THE RANK 1 UPDATE TO W. R1UP1420
C R1UP1430
DO 80 I = 1, M R1UP1440
W(I) = W(I) + V(N)*U(I) R1UP1450
80 CONTINUE R1UP1460
C R1UP1470
C ELIMINATE THE SPIKE. R1UP1480
C R1UP1490
SING = .FALSE. R1UP1500
IF (NM1 .LT. 1) GO TO 140 R1UP1510
DO 130 J = 1, NM1 R1UP1520
IF (W(J) .EQ. ZERO) GO TO 120 R1UP1530
C R1UP1540
C DETERMINE A GIVENS ROTATION WHICH ELIMINATES THE R1UP1550
C J-TH ELEMENT OF THE SPIKE. R1UP1560
C R1UP1570
IF (DABS(S(JJ)) .GE. DABS(W(J))) GO TO 90 R1UP1580
COTAN = S(JJ)/W(J) R1UP1590
SIN = P5/DSQRT(P25+P25*COTAN**2) R1UP1600
COS = SIN*COTAN R1UP1610
TAU = ONE R1UP1620

```

```

      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) SPPR0010
C INTEGER I SPPR0020
C **** SPPR0030
C
C FUNCTION SPMPAR SPPR0040
C
C THIS FUNCTION PROVIDES SINGLE PRECISION MACHINE PARAMETERS SPPR0050
C WHEN THE APPROPRIATE SET OF DATA STATEMENTS IS ACTIVATED (BY SPPR0060
C REMOVING THE C FROM COLUMN 1) AND ALL OTHER DATA STATEMENTS ARE SPPR0070
C RENDERED INACTIVE. MOST OF THE PARAMETER VALUES WERE OBTAINED SPPR0080
C FROM THE CORRESPONDING BELL LABORATORIES PORT LIBRARY FUNCTION. SPPR0090
C
C THE FUNCTION STATEMENT IS SPPR0100
C
C REAL FUNCTION SPMPAR(I) SPPR0110
C
C WHERE SPPR0120
C
C I IS AN INTEGER INPUT VARIABLE SET TO 1, 2, OR 3 WHICH SPPR0130
C SELECTS THE DESIRED MACHINE PARAMETER. IF THE MACHINE HAS SPPR0140
C T BASE B DIGITS AND ITS SMALLEST AND LARGEST EXPONENTS ARE SPPR0150
C EMIN AND EMAX, RESPECTIVELY, THEN THESE PARAMETERS ARE SPPR0160
C
C SPMPAR(1) = B** (1 - T), THE MACHINE PRECISION, SPPR0170
C
C SPMPAR(2) = B** (EMIN - 1), THE SMALLEST MAGNITUDE, SPPR0180
C
C SPMPAR(3) = B** EMAX*(1 - B** (-T)), THE LARGEST MAGNITUDE. SPPR0190
C
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. SPPR0200
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE SPPR0210
C
C **** SPPR0220
C INTEGER MCHEPS(2) SPPR0230
C INTEGER MINMAG(2) SPPR0240
C INTEGER MAXMAG(2) SPPR0250
C REAL RMACH(3) SPPR0260
C EQUIVALENCE (RMACH(1),MCHEPS(1)) SPPR0270
C EQUIVALENCE (RMACH(2),MINMAG(1)) SPPR0280
C EQUIVALENCE (RMACH(3),MAXMAG(1)) SPPR0290
C
C MACHINE CONSTANTS FOR THE IBM 360/370 SERIES, SPPR0300
C THE AMDAHL 470/V6, THE ICL 2900, THE ITAL AS/6, SPPR0310
C THE XEROX SIGMA 5/7/9 AND THE SEL SYSTEMS 85/86. SPPR0320
C
C DATA RMACH(1) / Z3C100000 / SPPR0330
C DATA RMACH(2) / Z00100000 / SPPR0340
C DATA RMACH(3) / Z7FFFFFF / SPPR0350
C
C MACHINE CONSTANTS FOR THE HONEYWELL 600/6000 SERIES. SPPR0360
C
C DATA RMACH(1) / 0716400000000 / SPPR0370
C DATA RMACH(2) / 0402400000000 / SPPR0380
C DATA RMACH(3) / 0376777777777 / SPPR0390

```

```

C          SPPR0550
C          SPPR0560
C          SPPR0570
C          SPPR0580
C          SPPR0590
C          SPPR0600
C          SPPR0610
C          SPPR0620
C          SPPR0630
C          SPPR0640
C          SPPR0650
C          SPPR0660
C          SPPR0670
C          SPPR0680
C          SPPR0690
C          SPPR0700
C          SPPR0710
C          SPPR0720
C          SPPR0730
C          SPPR0740
C          SPPR0750
C          SPPR0760
C          SPPR0770
C          SPPR0780
C          SPPR0790
C          SPPR0800
C          SPPR0810
C          SPPR0820
C          SPPR0830
C          SPPR0840
C          SPPR0850
C          SPPR0860
C          SPPR0870
C          SPPR0880
C          SPPR0890
C          SPPR0900
C          SPPR0910
C          SPPR0920
C          SPPR0930
C          SPPR0940
C          SPPR0950
C          SPPR0960
C          SPPR0970
C          SPPR0980
C          SPPR0990
C          SPPR1000
C          SPPR1010
C          SPPR1020
C          SPPR1030
C          SPPR1040
C          SPPR1050
C          SPPR1060
C          SPPR1070
C          SPPR1080

C MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.
C
C DATA RMACH(1) / 16414000000000000000B /
C DATA RMACH(2) / 0001400000000000000B /
C DATA RMACH(3) / 3776777777777777777B /
C
C MACHINE CONSTANTS FOR THE PDP-10 (KA OR KI PROCESSOR).
C
C DATA RMACH(1) / "147400000000 /
C DATA RMACH(2) / "000400000000 /
C DATA RMACH(3) / "377777777777 /
C
C MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING
C 32-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).
C
C DATA MCHEPS(1) / 889192448 /
C DATA MINMAG(1) / 8388608 /
C DATA MAXMAG(1) / 2147483647 /
C
C DATA RMACH(1) / 006500000000 /
C DATA RMACH(2) / 000040000000 /
C DATA RMACH(3) / 017777777777 /
C
C MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING
C 16-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).
C
C DATA MCHEPS(1),MCHEPS(2) / 13568,      0 /
C DATA MINMAG(1),MINMAG(2) / 128,        0 /
C DATA MAXMAG(1),MAXMAG(2) / 32767,     -1 /
C
C DATA MCHEPS(1),MCHEPS(2) / 0032400, 0000000 /
C DATA MINMAG(1),MINMAG(2) / 0000200, 0000000 /
C DATA MAXMAG(1),MAXMAG(2) / 0077777, 0177777 /
C
C MACHINE CONSTANTS FOR THE BURROUGHS 5700/6700/7700 SYSTEMS.
C
C DATA RMACH(1) / 01301000000000000000 /
C DATA RMACH(2) / 01771000000000000000 /
C DATA RMACH(3) / 00777777777777777777 /
C
C MACHINE CONSTANTS FOR THE BURROUGHS 1700 SYSTEM.
C
C DATA RMACH(1) / Z4EA800000 /
C DATA RMACH(2) / Z400800000 /
C DATA RMACH(3) / Z5FFFFFFF /
C
C MACHINE CONSTANTS FOR THE UNIVAC 1100 SERIES.
C
C DATA RMACH(1) / 0147400000000 /
C DATA RMACH(2) / 0000400000000 /
C DATA RMACH(3) / 0377777777777 /
C
C MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/200.
C

```

```

C SPPR1090
C NOTE - IT MAY BE APPROPRIATE TO INCLUDE THE FOLLOWING CARD -
C STATIC RMACH(3) SPPR1100
C SPPR1110
C SPPR1120
C DATA MINMAG/20K,0/ ,MAXMAG/77777K,177777K/ SPPR1130
C DATA MCHEPS/36020K,0/ SPPR1140
C SPPR1150
C MACHINE CONSTANTS FOR THE HARRIS 220. SPPR1160
C SPPR1170
C DATA MCHEPS(1),MCHEPS(2) / '20000000, '00000353 / SPPR1180
C DATA MINMAG(1),MINMAG(2) / '20000000, '00000201 / SPPR1190
C DATA MAXMAG(1),MAXMAG(2) / '37777777, '00000177 / SPPR1200
C SPPR1210
C MACHINE CONSTANTS FOR THE CRAY-1. SPPR1220
C SPPR1230
C DATA RMACH(1) / 0377224000000000000000B / SPPR1240
C DATA RMACH(2) / 0200034000000000000000B / SPPR1250
C DATA RMACH(3) / 057777777777777777776B / SPPR1260
C SPPR1270
C MACHINE CONSTANTS FOR THE PRIME 400. SPPR1280
C SPPR1290
C DATA MCHEPS(1) / :10000000153 / SPPR1300
C DATA MINMAG(1) / :10000000000 / SPPR1310
C DATA MAXMAG(1) / :17777777777 / SPPR1320
C SPPR1330
C SPMPAR = RMACH(I)
C RETURN SPPR1340
C LAST CARD OF FUNCTION SPMPAR. SPPR1350
C SPPR1360
C END SPPR1370
C SPPR1380
C SPPR1390

```

WHY
DO YOU
THINK THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

```

DOUBLE PRECISION FUNCTION DPMPAR(I) DPPR0010
INTEGER I DPPR0020
C **** DPPR0030
C FUNCTION DPMPAR DPPR0040
C
C THIS FUNCTION PROVIDES DOUBLE PRECISION MACHINE PARAMETERS DPPR0050
C WHEN THE APPROPRIATE SET OF DATA STATEMENTS IS ACTIVATED (BY DPPR0060
C REMOVING THE C FROM COLUMN 1) AND ALL OTHER DATA STATEMENTS ARE DPPR0070
C RENDERED INACTIVE. MOST OF THE PARAMETER VALUES WERE OBTAINED DPPR0080
C FROM THE CORRESPONDING BELL LABORATORIES PORT LIBRARY FUNCTION. DPPR0090
C
C THE FUNCTION STATEMENT IS DPPR0100
C
C DOUBLE PRECISION FUNCTION DPMPAR(I) DPPR0110
C
C WHERE DPPR0120
C
C I IS AN INTEGER INPUT VARIABLE SET TO 1, 2, OR 3 WHICH DPPR0130
C SELECTS THE DESIRED MACHINE PARAMETER. IF THE MACHINE HAS DPPR0140
C T BASE B DIGITS AND ITS SMALLEST AND LARGEST EXPONENTS ARE DPPR0150
C EMIN AND EMAX, RESPECTIVELY, THEN THESE PARAMETERS ARE DPPR0160
C
C DPMPAR(1) = B**(1 - T), THE MACHINE PRECISION, DPPR0170
C
C DPMPAR(2) = B**EMIN - 1, THE SMALLEST MAGNITUDE, DPPR0180
C
C DPMPAR(3) = B**EMAX*(1 - B**(-T)), THE LARGEST MAGNITUDE. DPPR0190
C
C ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980. DPPR0200
C BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE DPPR0210
C
C **** DPPR0220
C INTEGER MCHEPS(4) DPPR0230
C INTEGER MINMAG(4) DPPR0240
C INTEGER MAXMAG(4) DPPR0250
C DOUBLE PRECISION DMACH(3) DPPR0260
C EQUIVALENCE (DMACH(1),MCHEPS(1)) DPPR0270
C EQUIVALENCE (DMACH(2),MINMAG(1)) DPPR0280
C EQUIVALENCE (DMACH(3),MAXMAG(1)) DPPR0290
C
C MACHINE CONSTANTS FOR THE IBM 360/370 SERIES, DPPR0300
C THE AMDAHL 470/V6, THE ICL 2900, THE ITEL AS/6, DPPR0310
C THE XEROX SIGMA 5/7/9 AND THE SEL SYSTEMS 85/86. DPPR0320
C
C DATA MCHEPS(1),MCHEPS(2) / Z34100000, Z00000000 / DPPR0330
C DATA MINMAG(1),MINMAG(2) / Z00100000, Z00000000 / DPPR0340
C DATA MAXMAG(1),MAXMAG(2) / Z7FFFFFF, ZFFFFFF / DPPR0350
C
C MACHINE CONSTANTS FOR THE HONEYWELL 600/6000 SERIES. DPPR0360
C
C DATA MCHEPS(1),MCHEPS(2) / 0606400000000, 0000000000000 / DPPR0370
C DATA MINMAG(1),MINMAG(2) / 0402400000000, 0000000000000 / DPPR0380
C DATA MAXMAG(1),MAXMAG(2) / 0376777777777, 0777777777777 / DPPR0390

```

```

C          DPPR0550
C          DPPR0560
C          DPPR0570
C          DPPR0580
C          DPPR0590
C          DPPR0600
C          DPPR0610
C          DPPR0620
C          DPPR0630
C          DPPR0640
C          DPPR0650
C          DPPR0660
C          DPPR0670
C          DPPR0680
C          DPPR0690
C          DPPR0700
C          DPPR0710
C          DPPR0720
C          DPPR0730
C          DPPR0740
C          DPPR0750
C          DPPR0760
C          DPPR0770
C          DPPR0780
C          DPPR0790
C          DPPR0800
C          DPPR0810
C          DPPR0820
C          DPPR0830
C          DPPR0840
C          DPPR0850
C          DPPR0860
C          DPPR0870
C          DPPR0880
C          DPPR0890
C          DPPR0900
C          DPPR0910
C          DPPR0920
C          DPPR0930
C          DPPR0940
C          DPPR0950
C          DPPR0960
C          DPPR0970
C          DPPR0980
C          DPPR0990
C          DPPR1000
C          DPPR1010
C          DPPR1020
C          DPPR1030
C          DPPR1040
C          DPPR1050
C          DPPR1060
C          DPPR1070
C          DPPR1080

C MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.

C DATA MCHEPS(1) / 1561400000000000000B /
C DATA MCHEPS(2) / 1501000000000000000B /
C
C DATA MINMAG(1) / 0060400000000000000B /
C DATA MINMAG(2) / 0000000000000000000B /
C
C DATA MAXMAG(1) / 3776777777777777777B /
C DATA MAXMAG(2) / 3716777777777777777B /
C
C MACHINE CONSTANTS FOR THE PDP-10 (KA PROCESSOR).

C DATA MCHEPS(1),MCHEPS(2) / "114400000000, "000000000000 /
C DATA MINMAG(1),MINMAG(2) / "033400000000, "000000000000 /
C DATA MAXMAG(1),MAXMAG(2) / "377777777777, "344777777777 /
C
C MACHINE CONSTANTS FOR THE PDP-10 (KI PROCESSOR).

C DATA MCHEPS(1),MCHEPS(2) / "104400000000, "000000000000 /
C DATA MINMAG(1),MINMAG(2) / "000400000000, "000000000000 /
C DATA MAXMAG(1),MAXMAG(2) / "377777777777, "377777777777 /
C
C MACHINE CONSTANTS FOR THE PDP-11 FORTAN SUPPORTING
C 32-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).

C DATA MCHEPS(1),MCHEPS(2) / 620756992,           0 /
C DATA MINMAG(1),MINMAG(2) /     8388608,           0 /
C DATA MAXMAG(1),MAXMAG(2) / 2147483647,          -1 /
C
C DATA MCHEPS(1),MCHEPS(2) / 004500000000, 000000000000 /
C DATA MINMAG(1),MINMAG(2) / 000040000000, 000000000000 /
C DATA MAXMAG(1),MAXMAG(2) / 017777777777, 037777777777 /
C
C MACHINE CONSTANTS FOR THE PDP-11 FORTAN SUPPORTING
C 16-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).

C DATA MCHEPS(1),MCHEPS(2) /    9472,      0 /
C DATA MCHEPS(3),MCHEPS(4) /      0,       0 /
C
C DATA MINMAG(1),MINMAG(2) /     128,      0 /
C DATA MINMAG(3),MINMAG(4) /      0,       0 /
C
C DATA MAXMAG(1),MAXMAG(2) /   32767,     -1 /
C DATA MAXMAG(3),MAXMAG(4) /     -1,     -1 /
C
C DATA MCHEPS(1),MCHEPS(2) / 0022400, 0000000 /
C DATA MCHEPS(3),MCHEPS(4) / 0000000, 0000000 /
C
C DATA MINMAG(1),MINMAG(2) / 0000200, 0000000 /
C DATA MINMAG(3),MINMAG(4) / 0000000, 0000000 /
C
C DATA MAXMAG(1),MAXMAG(2) / 0077777, 0177777 /

```

```

C DATA MAXMAG(3),MAXMAG(4) / 0177777, 0177777 /
C DPPR1090
C
C MACHINE CONSTANTS FOR THE BURROUGHS 6700/7700 SYSTEMS.
C DPPR1100
C DPPR1110
C DPPR1120
C DATA MCHEPS(1) / 0145100000000000 /
C DPPR1130
C DATA MCHEPS(2) / 0000000000000000 /
C DPPR1140
C DPPR1150
C DATA MINMAG(1) / 0177100000000000 /
C DPPR1160
C DATA MINMAG(2) / 0777000000000000 /
C DPPR1170
C DPPR1180
C DATA MAXMAG(1) / 0077777777777777 /
C DPPR1190
C DATA MAXMAG(2) / 0777777777777777 /
C DPPR1200
C DPPR1210
C MACHINE CONSTANTS FOR THE BURROUGHS 5700 SYSTEM.
C DPPR1220
C DPPR1230
C DATA MCHEPS(1) / 0145100000000000 /
C DPPR1240
C DATA MCHEPS(2) / 0000000000000000 /
C DPPR1250
C DPPR1260
C DATA MINMAG(1) / 0177100000000000 /
C DPPR1270
C DATA MINMAG(2) / 0000000000000000 /
C DPPR1280
C DPPR1290
C DATA MAXMAG(1) / 0077777777777777 /
C DPPR1300
C DATA MAXMAG(2) / 0000777777777777 /
C DPPR1310
C DPPR1320
C MACHINE CONSTANTS FOR THE BURROUGHS 1700 SYSTEM.
C DPPR1330
C DPPR1340
C DATA MCHEPS(1) / ZCC6800000 /
C DPPR1350
C DATA MCHEPS(2) / Z000000000 /
C DPPR1360
C DPPR1370
C DATA MINMAG(1) / ZC00800000 /
C DPPR1380
C DATA MINMAG(2) / Z000000000 /
C DPPR1390
C DPPR1400
C DATA MAXMAG(1) / ZFFFFFFF /
C DPPR1410
C DATA MAXMAG(2) / ZFFFFFFF /
C DPPR1420
C DPPR1430
C MACHINE CONSTANTS FOR THE UNIVAC 1100 SERIES.
C DPPR1440
C DPPR1450
C DATA MCHEPS(1),MCHEPS(2) / 0170640000000, 0000000000000000 /
C DPPR1460
C DATA MINMAG(1),MINMAG(2) / 0000040000000, 0000000000000000 /
C DPPR1470
C DATA MAXMAG(1),MAXMAG(2) / 0377777777777, 0777777777777777 /
C DPPR1480
C DPPR1490
C MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/200.
C DPPR1500
C DPPR1510
C NOTE - IT MAY BE APPROPRIATE TO INCLUDE THE FOLLOWING CARD -
C DPPR1520
C STATIC DMACH(3)
C DPPR1530
C DPPR1540
C DATA MINMAG/20K,3*0/,MAXMAG/77777K,3*177777K/
C DPPR1550
C DATA MCHEPS/32020K,3*0/
C DPPR1560
C DPPR1570
C MACHINE CONSTANTS FOR THE HARRIS 220.
C DPPR1580
C DPPR1590
C DATA MCHEPS(1),MCHEPS(2) / '20000000, '00000334 /
C DPPR1600
C DATA MINMAG(1),MINMAG(2) / '20000000, '00000201 /
C DPPR1610
C DATA MAXMAG(1),MAXMAG(2) / '37777777, '37777577 /
C DPPR1620

```

```

C          DPPR1630
C MACHINE CONSTANTS FOR THE CRAY-1.      DPPR1640
C          DPPR1650
C          DPPR1660
C DATA MCHEPS(1) / 0376424000000000000000B /
C DATA MCHEPS(2) / 0000000000000000000000B /
C          DPPR1670
C          DPPR1680
C DATA MINMAG(1) / 0200034000000000000000B /
C DATA MINMAG(2) / 0000000000000000000000B /
C          DPPR1690
C          DPPR1700
C          DPPR1710
C DATA MAXMAG(1) / 0577777777777777777777B /
C DATA MAXMAG(2) / 0000007777777777777776B /
C          DPPR1720
C          DPPR1730
C          DPPR1740
C MACHINE CONSTANTS FOR THE PRIME 400.    DPPR1750
C          DPPR1760
C          DPPR1770
C DATA MCHEPS(1),MCHEPS(2) / :1000000000, :00000000123 /
C DATA MINMAG(1),MINMAG(2) / :1000000000, :00000100000 /
C DATA MAXMAG(1),MAXMAG(2) / :17777777777, :37777677776 /
C          DPPR1780
C          DPPR1790
C          DPPR1800
C          DPPR1810
C          DPPR1820
C          DPPR1830
C          DPPR1840
C          DPPR1850
C          DPPR1860
C DPMPAR = DMACH(I)
C RETURN
C LAST CARD OF FUNCTION DPMPAR.
C END

```

Distribution for ANL-80-74Internal:

J. M. Boyle	J. J. Moreé
W. J. Cody	National Energy Software Center (100)
T. F. Coleman	D. M. Pahis
W. R. Cowell	L. M. Phebus (2)
J. J. Dongarra	G. W. Pieper
B. S. Garbow (100)	R. C. Raffenetti
G. T. Garvey	R. J. Royston
K. E. Hillstrom	D. C. Sorensen
A. B. Krissiunas	B. T. Smith
J. N. Lyness	ANL Contract File
P. C. Messina	ANL Libraries
M. Minkoff	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)

