

ANL-80-74

1880

ANL-80-74

159, 97C
10-28-80

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	The University of Kansas	The Ohio State University
Carnegie-Mellon University	Kansas State University	Ohio University
Case Western Reserve University	Loyola University of Chicago	The Pennsylvania State University
The University of Chicago	Marquette University	Purdue University
University of Cincinnati	The University of Michigan	Saint Louis University
Illinois Institute of Technology	Michigan State University	Southern Illinois University
University of Illinois	University of Minnesota	The University of Texas at Austin
Indiana University	University of Missouri	Washington University
The University of Iowa	Northwestern University	Wayne State University
Iowa State University	University of Notre Dame	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

ANL-80-74

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

USER GUIDE FOR MINPACK-1

by

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

Applied Mathematics Division

August 1980

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.

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

Reliability further implies efficient and robust implementations. For example, MINPACK-1 programs access matrices sequentially along columns (rather than rows), since this improves efficiency, especially on paged systems. Also, there are extensive checks on the input parameters, and computations are

formulated to avoid destructive underflows and overflows. Underflows can then be safely ignored; overflows due to the problem should of course be investigated.

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

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

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

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

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

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

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

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

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

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

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

Acknowledgments

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

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

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

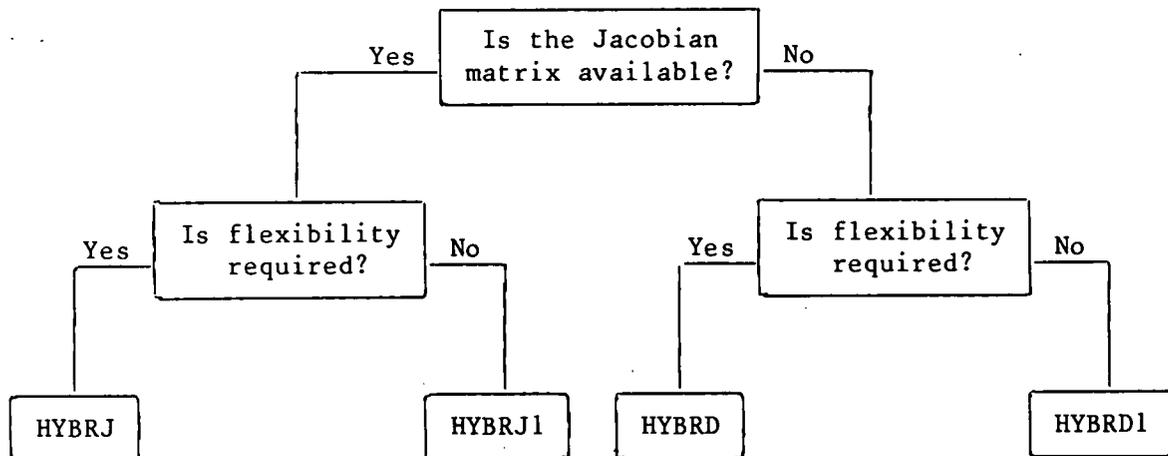


Figure 1
Decision Tree for Systems of Nonlinear Equations

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.

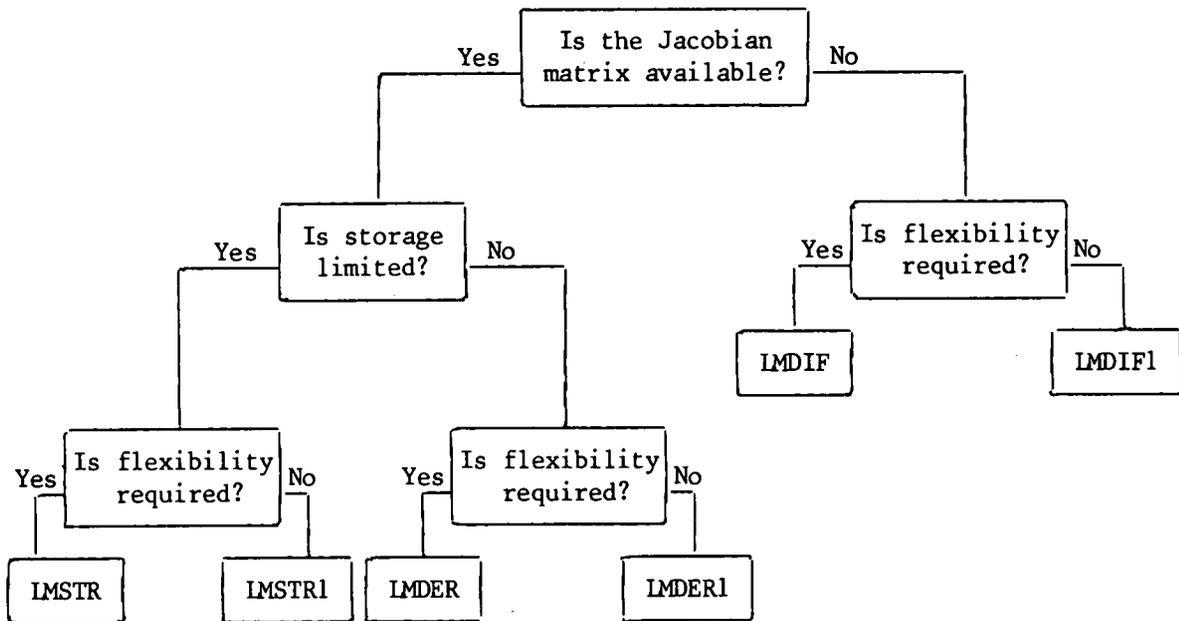


Figure 2
Decision Tree for Nonlinear Least Squares Problems

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:

- β^{1-l} , the machine precision ,
- $\beta^{e_{\min}^{-1}}$, the smallest magnitude ,
- $(1 - \beta^{-l})\beta^{e_{\max}}$, the largest magnitude ,

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

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

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

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 .

RIMPYQ

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

RIUPDT

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 THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

CHAPTER 2

Algorithmic Details

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

2.1 Mathematical Background

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

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

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

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

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

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

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

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 $g(t,x)$ to data y , in which case the f_i are of the form

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

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

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

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

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

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

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

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

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

A related concept is the gradient of a function $f: R^n \rightarrow R$, which is the mapping $\nabla f: R^n \rightarrow R^n$ 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 $m > n$ 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

$$\Delta \leq XTOL \cdot \|Dx\| .$$

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

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

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

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

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

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

On a computer with 15 decimal digits

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

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

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

```

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

```

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

2.4 Approximations to the Jacobian Matrix

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

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

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

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,

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

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

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

is given by

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

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

2.5 Scaling

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

$$\begin{aligned}\tilde{F}(x) &= \alpha F(D_V x) \\ \tilde{x}_0 &= D_V^{-1} x_0,\end{aligned}$$

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

$$(1) \quad \tilde{D}_0 = \alpha D_V D_0,$$

where D_0 and \tilde{D}_0 are the initial scaling matrices of the respective problems defined by F and x_0 and by \tilde{F} and \tilde{x}_0 . 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
    
$$\frac{\partial f_i(x)}{\partial x_j} = -\sigma_{ij} + \gamma\beta_{ij} .$$


```

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

$$\|D_0 p_0\| \leq \Delta_0 ,$$

where D_0 is a diagonal scaling matrix and Δ_0 is a step bound. The contents of D_0 are governed by the parameter MODE. If MODE = 1 then D_0 is internally set, while if MODE = 2 then D_0 is specified by the user through the array DIAG. The step bound Δ_0 is determined from the parameter FACTOR. By definition

$$\Delta_0 = \text{FACTOR} \cdot \|D_0 x_0\| ,$$

unless x_0 is the zero vector, in which case

$$\Delta_0 = \text{FACTOR} .$$

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 $x_0 + p_0$ will be found that belongs to D.

Be aware that the step restriction is on $D_0 p_0$ and not on p_0 directly. A small element of D_0 , which can be set by internal scaling when MODE = 1, may lead to a large component in the correction p_0 . In many cases it is not necessary to control p_0 directly, but if this is desired then MODE = 2 must be used.

When MODE = 2, the contents of D_0 are specified by the user, and this allows direct control of p_0 . If, for example, it is desired to restrict the components of p_0 to small relative corrections of the corresponding components of x_0 (assumed nonzero), then this can be done by setting

$$D_0 = \text{diag} \left(\frac{1}{|\xi_1|}, \frac{1}{|\xi_2|}, \dots, \frac{1}{|\xi_n|} \right) ,$$

where ξ_i is the i -th component of x_0 , and by choosing FACTOR appropriately. To justify this choice, note that p_0 satisfies

$$\|D_0 p_0\| \leq \Delta_0 = \text{FACTOR} \cdot \|D_0 x_0\| ,$$

and that the choice of D_0 guarantees that

$$\|D_0 x_0\| = n^{1/2}.$$

Thus, if ρ_i is the i -th component of p_0 , then

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

which justifies the choice of D_0 .

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

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

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

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

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

$$J e_{p(j)} = Q R e_j ,$$

and hence,

$$\|J e_{p(j)}\| = \|R e_j\| .$$

The following loop uses this relationship to store $\|J e_\ell\|$ 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

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

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

For example, if the level set

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

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.

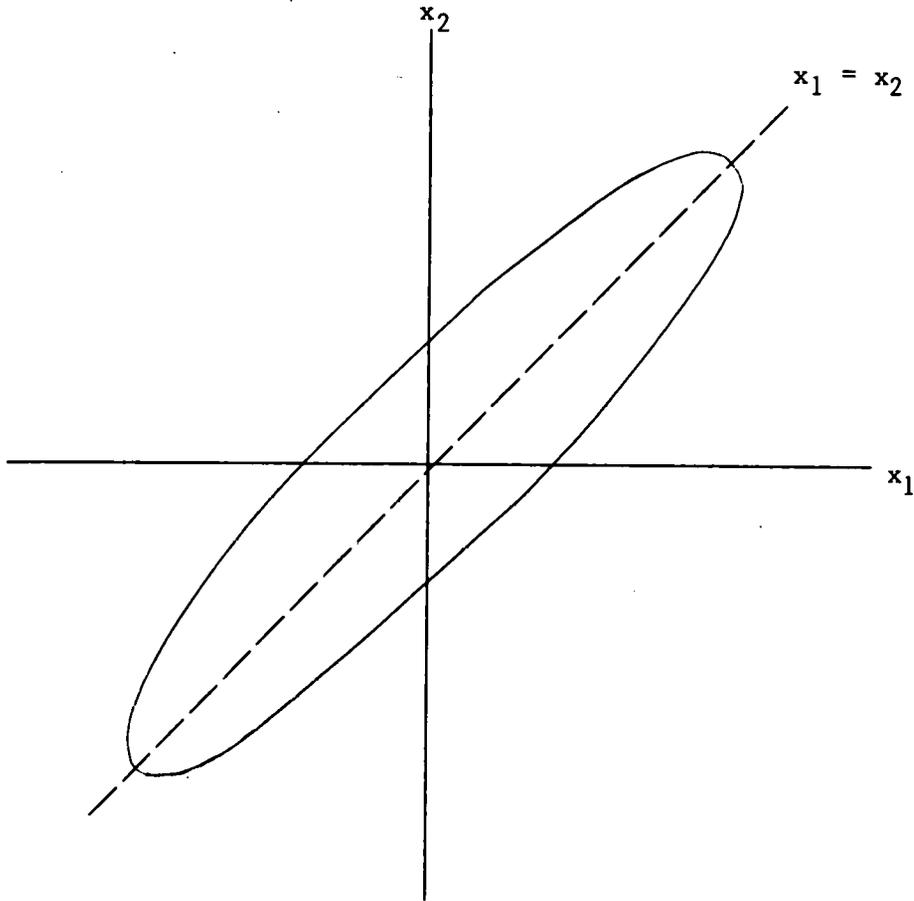


Figure 3

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,WA1,WA2,WA3)
DO 40 J = 1, N
  FJAC(J,J) = WA1(J)
  L = IPVT2(J)
  IPVT2(J) = IPVT1(L)
40  CONTINUE

```

Note that QRFAC sets the contents of the array IPVT2 to define the permutation matrix P_2 , and the final loop in the program segment overwrites IPVT2 to define the permutation matrix $P_1 P_2$.


```

C          PERMUTATION MATRIX P SUCH THAT A*P = Q*R. COLUMN J OF P      COVR0550
C          IS COLUMN IPV(T(J) OF THE IDENTITY MATRIX.                  COVR0560
C                                                                      COVR0570
C          TOL IS A NONNEGATIVE INPUT VARIABLE USED TO DEFINE THE      COVR0580
C          NUMERICAL RANK OF A IN THE MANNER DESCRIBED ABOVE.         COVR0590
C                                                                      COVR0600
C          WA IS A WORK ARRAY OF LENGTH N.                             COVR0610
C                                                                      COVR0620
C          SUBPROGRAMS CALLED                                          COVR0630
C                                                                      COVR0640
C          . FORTRAN-SUPPLIED ... DABS                                COVR0650
C                                                                      COVR0660
C          ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. AUGUST 1980. COVR0670
C          BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE      COVR0680
C                                                                      COVR0690
C          *****                                                  COVR0700
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
C             10          CONTINUE                                    COVR0900
C             20          CONTINUE                                    COVR0910
C             30          CONTINUE                                    COVR0920
C                L = K                                             COVR0930
C             40          CONTINUE                                    COVR0940
C             50          CONTINUE                                    COVR0950
C                                                                      COVR0960
C          FORM THE FULL UPPER TRIANGLE OF THE INVERSE OF (R TRANSPOSE)*R COVR0970
C          IN THE FULL UPPER TRIANGLE OF R.                          COVR0980
C                                                                      COVR0990
C          IF (L .LT. 1) GO TO 110                                   COVR1000
C          DO 100 K = 1, L                                          COVR1010
C             KM1 = K - 1                                           COVR1020
C             IF (KM1 .LT. 1) GO TO 80                               COVR1030
C             DO 70 J = 1, KM1                                       COVR1040
C                TEMP = R(J,K)                                       COVR1050
C                DO 60 I = 1, J                                       COVR1060
C                   R(I,J) = R(I,J) + TEMP*R(I,K)                 COVR1070
C             60          CONTINUE                                    COVR1080
C             70          CONTINUE
C             80          CONTINUE
C             100         CONTINUE

```

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		COVR1170
C	FORM THE FULL LOWER TRIANGLE OF THE COVARIANCE MATRIX	COVR1180
C	IN THE STRICT LOWER TRIANGLE OF R AND IN WA.	COVR1190
C		COVR1200
	DO 130 J = 1, N	COVR1210
	JJ = IPVT(J)	COVR1220
	SING = J .GT. L	COVR1230
	DO 120 I = 1, J	COVR1240
	IF (SING) R(I,J) = ZERO	COVR1250
	II = IPVT(I)	COVR1260
	IF (II .GT. JJ) R(II,JJ) = R(I,J)	COVR1270
	IF (II .LT. JJ) R(JJ,II) = R(I,J)	COVR1280
120	CONTINUE	COVR1290
	WA(JJ) = R(J,J)	COVR1300
130	CONTINUE	COVR1310
C		COVR1320
C	SYMMETRIZE THE COVARIANCE MATRIX IN R.	COVR1330
C		COVR1340
	DO 150 J = 1, N	COVR1350
	DO 140 I = 1, J	COVR1360
	R(I,J) = R(J,I)	COVR1370
140	CONTINUE	COVR1380
	R(J,J) = WA(J)	COVR1390
150	CONTINUE	COVR1400
	RETURN	COVR1410
C		COVR1420
C	LAST CARD OF SUBROUTINE COVAR.	COVR1430
C		COVR1440
	END	COVR1450

2.9 Printing

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

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

```

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

```

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

CHAPTER 3

Notes and References

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

Powell's Hybrid Method

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

The Levenberg-Marquardt Algorithm

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

Derivative Checking

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

MINPACK-1 Internal Subprograms

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

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

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

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

The algorithm used in 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 Applics 13, 117-119.
- Dongarra, J. J., Bunch, J. R., Moler, C. B., and Stewart, G. W. [1979]. LINPACK users' guide, SIAM Publications.

- Fosdick, L. D. [1974]. BRNANL, A Fortran program to identify basic blocks in Fortran programs, University of Colorado, Computer Science report CU-CS-040-74.
- Fox, P. A., Hall, A. D., and Schryer, N. L. [1978]. The PORT mathematical subroutine library, ACM Transactions on Mathematical Software 4, 104-126.
- Garbow, B. S., Hillstrom, K. E., and Moré, J. J. [1980]. Implementation guide for MINPACK-1, Argonne National Laboratory report ANL-80-68.
- Gill, P. E., Golub, G. H., Murray, W., and Saunders, M. A. [1974]. Methods for modifying matrix factorizations, Math. Comp. 28, 505-535.
- Hebden, M. D. [1973]. An algorithm for minimization using exact second derivatives, Atomic Energy Research Establishment report TP 515, Harwell, England.
- Hiebert, K. L. [1979]. A comparison of nonlinear least squares software, Sandia Laboratories report SAND 79-0483, Albuquerque, New Mexico.
- Hiebert, K. L. [1980]. A comparison of software which solves systems of nonlinear equations, Sandia Laboratories report SAND 80-0181, Albuquerque, New Mexico.
- Levenberg, K. [1944]. A method for the solution of certain nonlinear problems in least squares, Quart. Appl. Math. 2, 164-168.
- Marquardt, D. W. [1963]. An algorithm for least-squares estimation of nonlinear parameters, SIAM J. Appl. Math. 11, 431-441.
- Moré, J. J. [1977]. The Levenberg-Marquardt algorithm: Implementation and Theory, Numerical Analysis, G. A. Watson, ed., Lecture Notes in Mathematics 630, Springer-Verlag.
- Moré, J. J., Garbow, B. S., and Hillstrom, K. E. [1978]. Testing unconstrained optimization software, Argonne National Laboratory, Applied Mathematics Division Technical Memorandum 324 (to appear in ACM Transactions on Mathematical Software).
- Powell, M. J. D. [1970]. A hybrid method for nonlinear equations, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach.
- Ryder, B. G. [1974]. The PFORT verifier, Software Practice and Experience 4, 359-377.

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

CHAPTER 4
Documentation

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

Systems of nonlinear equations

HYBRD1, HYBRD, HYBRJ1, HYBRJ

Nonlinear least squares problems

LMDF1, LMDIF, LMDER1, LMDER, LMSTR1, LMSTR

Derivative checking

CHKDER

THIS PAGE
WAS THIS PAGE ALLY
WAS INTENTIONALLY
LEFT BLANK

Documentation for MINPACK subroutine HYBRD1

Double precision version

Argonne National Laboratory

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

March 1980

1. Purpose.

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

2. Subroutine and type statements.

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

3. Parameters.

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

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

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

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

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

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

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

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

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

INFO = 0 Improper input parameters.

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

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

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

INFO = 4 Iteration is not making good progress.

Sections 4 and 5 contain more details about INFO.

WA is a work array of length LWA.

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

4. Successful completion.

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

The test assumes that the functions are reasonably well behaved.

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

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

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

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

5. Unsuccessful completion.

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

Improper input parameters. $INFO$ is set to 0 if $N \leq 0$, 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 $11.5 \cdot (N^2)$ 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 $(3 \cdot N^2 + 17 \cdot N)/2$ double precision storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied FCN

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

FORTTRAN-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 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
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
1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
*          5X,15H EXIT PARAMETER,16X,I10 //
*          5X,27H FINAL APPROXIMATE SOLUTION // (5X,3D15.7))
C
C LAST CARD OF DRIVER FOR HYBRD1 EXAMPLE.
C
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
      INTEGER K
      DOUBLE PRECISION ONE, TEMP, TEMP1, TEMP2, THREE, TWO, ZERO
      DATA ZERO, ONE, TWO, THREE /0.DO, 1.DO, 2.DO, 3.DO/
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. Hillstom, Jorge J. More

March 1980

1. Purpose.

The purpose of HYBRD is to find a zero of a system of N non-linear functions in N variables by a modification of the Powell hybrid method. The user must provide a subroutine which calculates the functions. 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,EPSECN,DIAG,
*           MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,LDFJAC,
*           R,LR,QTF,WA1,WA2,WA3,WA4)
INTEGER N,MAXFEV,ML,MU,MODE,NPRINT,INFO,NFEV,LDFJAC,LR
DOUBLE PRECISION XTOL,EPSECN,FACTOR
DOUBLE PRECISION X(N),FVEC(N),DIAG(N),FJAC(LDFJAC,N),R(LR),QTF(N),
*           WA1(N),WA2(N),WA3(N),WA4(N)
EXTERNAL FCN

```

3. Parameters.

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

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

```

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

```

The value of IFLAG should not be changed by FCN unless the

user wants to terminate execution of HYBRD. In this case set IFLAG to a negative integer.

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

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

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

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

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

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

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

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

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

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

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

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

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

INFO = 0 Improper input parameters.

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

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

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

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

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

Sections 4 and 5 contain more details about INFO.

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

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

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

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

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

QTF is an output array of length N which contains the vector $(Q \text{ transpose}) * FVEC$.

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

$$ENORM(D*(X-XSOL)) \leq XTOL*ENORM(D*XSOL).$$

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

5. Unsuccessful completion.

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

Improper input parameters. INFO is set to 0 if N \leq 0, or XTOL $<$ 0.DO, or MAXFEV \leq 0, or ML $<$ 0, or MU $<$ 0, or FACTOR \leq 0.DO, 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

FORTTRAN-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 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,EPFCN,FACTOR,FNORM
C DOUBLE PRECISION X(9),FVEC(9),DIAG(9),FJAC(9,9),R(45),QTF(9),
* 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.DO
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 EPFCN = 0.DO
C MODE = 2
C DO 20 J = 1, 9
C   DIAG(J) = 1.DO

```

```

20    CONTINUE
    FACTOR = 1.D2
    NPRINT = 0
C
    CALL HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSEFCN,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. Hillstom, Jorge J. More

March 1980

1. Purpose.

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

2. Subroutine and type statements.

```

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

```

3. Parameters.

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

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

```

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

```

The value of IFLAG should not be changed by FCN unless the

user wants to terminate execution of HYBRJ1. In this case set IFLAG to a negative integer.

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

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

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

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

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

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

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

INFO = 0 Improper input parameters.

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

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

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

INFO = 4 Iteration is not making good progress.

Sections 4 and 5 contain more details about INFO.

WA is a work array of length LWA.

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

4. Successful completion.

The accuracy of HYBRJ1 is controlled by the convergence

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

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

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

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

If this condition is satisfied with $\text{TOL} = 10^{*(-K)}$, 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 $\text{LDFJAC} \leq N$, or $\text{TOL} \leq 0$, or $\text{LWA} \leq (N*(N+13))/2$.

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

FORTTRAN-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 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.DO
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 //
*          5X,15H EXIT PARAMETER,16X,I10 //
*          5X,27H FINAL APPROXIMATE SOLUTION // (5X,3D15.7))

```

```

C
C
C
LAST CARD OF DRIVER FOR HYBRJ1 EXAMPLE.

END
SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)
INTEGER N,LDFJAC,IFLAG
DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N)

C
C
C
SUBROUTINE FCN FOR HYBRJ1 EXAMPLE.

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

C
IF (IFLAG .EQ. 2) GO TO 20
DO 10 K = 1, N
  TEMP = (THREE - TWO*X(K))*X(K)
  TEMP1 = ZERO
  IF (K .NE. 1) TEMP1 = X(K-1)
  TEMP2 = ZERO
  IF (K .NE. N) TEMP2 = X(K+1)
  FVEC(K) = TEMP - TEMP1 - TWO*TEMP2 + ONE
10  CONTINUE
  GO TO 50
20 CONTINUE
  DO 40 K = 1, N
    DO 30 J = 1, N
      FJAC(K,J) = ZERO
30  CONTINUE
      FJAC(K,K) = THREE - FOUR*X(K)
      IF (K .NE. 1) FJAC(K,K-1) = -ONE
      IF (K .NE. N) FJAC(K,K+1) = -TWO
40  CONTINUE
50 CONTINUE
  RETURN

C
C
C
LAST CARD OF SUBROUTINE FCN.

END

```

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

FINAL L2 NORM OF THE RESIDUALS 0.1192636D-07

EXIT PARAMETER 1

FINAL APPROXIMATE SOLUTION

```

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

```

Documentation for MINPACK subroutine HYBRJ

Double precision version

Argonne National Laboratory

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

March 1980

1. Purpose.

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

2. Subroutine and type statements.

```

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

```

3. Parameters.

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

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

special calls of FCN with IFLAG = 0 are made.

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

INFO = 0 Improper input parameters.

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

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

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

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

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

Sections 4 and 5 contain more details about INFO.

NEEV 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

$$ENORM(D*(X-XSOL)) \leq XTOL*ENORM(D*XSOL).$$

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

5. Unsuccessful completion.

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

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

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

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

Lack of good progress. HYBRJ searches for a zero of the system by minimizing the sum of the squares of the functions. In so

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

6. Characteristics of the algorithm.

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

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

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

7. Subprograms required.

USER-supplied FCN

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

FORTTRAN-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),
* 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.DO
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 = 1000
C MODE = 2
C DO 20 J = 1, 9
C   DIAG(J) = 1.DO
20 CONTINUE
C FACTOR = 1.D2
C NPRINT = 0
C
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)
C FNORM = ENORM(N,FVEC)
C WRITE (NWRITE,1000) FNORM,NFEV,NJEV,INFO,(X(J),J=1,N)

```

```

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

```

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

FINAL L2 NORM OF THE RESIDUALS 0.1192636D-07

NUMBER OF FUNCTION EVALUATIONS 11

NUMBER OF JACOBIAN EVALUATIONS 1

EXIT PARAMETER 1

FINAL APPROXIMATE SOLUTION

-0.5706545D+00 -0.6816283D+00 -0.7017325D+00

-0.7042129D+00 -0.7013690D+00 -0.6918656D+00

-0.6657920D+00 -0.5960342D+00 -0.4164121D+00

Documentation for MINPACK subroutine LMDER1

Double precision version

Argonne National Laboratory

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

March 1980

1. Purpose.

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

2. Subroutine and type statements.

```

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

```

3. Parameters.

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

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

INFO = 0 Improper input parameters.

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

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

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

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

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

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

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

Sections 4 and 5 contain more details about INFO.

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

WA is a work array of length LWA.

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

4. Successful completion.

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

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

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

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

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

satisfied).

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

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

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 $M \leq N$, or $LDFJAC \leq M$, or $\text{TOL} \leq 0$, or $LWA \leq 5*N+M$.

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 $M \cdot N + 2 \cdot M + 6 \cdot N$ 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

FORTTRAN-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 LMDER1 EXAMPLE.
C DOUBLE PRECISION VERSION
C
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,
*          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 //
*          5X,15H EXIT PARAMETER,16X,I10 //
*          5X,27H FINAL APPROXIMATE SOLUTION // 5X,3D15.7)
C
C LAST CARD OF DRIVER FOR LMDER1 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)

```

```

SUBROUTINE FCN FOR LMDER1 EXAMPLE.

```

```

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/

```

```

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

```

```

LAST CARD OF SUBROUTINE FCN.

```

```

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 THIS PAGE ALLY
WAS INTENTIONALLY
LEFT BLANK

Documentation for MINPACK subroutine LMDER

Double precision version

Argonne National Laboratory

Burton S. Garbow, Kenneth E. Hillstom, 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 $DIAG \cdot X$ if nonzero, or else to FACTOR itself. In most cases FACTOR should lie in the interval (.1,100.). 100. is a generally recommended value.

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

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

- INFO = 0 Improper input parameters.
- INFO = 1 Both actual and predicted relative reductions in the sum of squares are at most FTOL.
- INFO = 2 Relative error between two consecutive iterates is at most XTOL.
- INFO = 3 Conditions for INFO = 1 and INFO = 2 both hold.
- INFO = 4 The cosine of the angle between FVEC and any column of the Jacobian is at most GTOL in absolute value.
- INFO = 5 Number of calls to FCN with IFLAG = 1 has reached MAXFEV.
- INFO = 6 FTOL is too small. No further reduction in the sum of squares is possible.
- INFO = 7 XTOL is too small. No further improvement in the approximate solution X is possible.
- INFO = 8 GTOL is too small. FVEC is orthogonal to the columns of the Jacobian to machine precision.

Sections 4 and 5 contain more details about INFO.

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

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

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

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

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

WA4 is a work array of length M.

4. Successful completion.

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

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

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

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

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

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

If this condition is satisfied with $\text{XTOL} = 10^{**(-K)}$, 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 $\text{MODE} = 1$, 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 ($\text{INFO} = 4$) 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 $N \leq 0$, or $M < N$, or LDFJAC $< M$, or FTOL < 0.00 , or XTOL < 0.00 , or GTOL < 0.00 , or MAXFEV ≤ 0 , or FACTOR ≤ 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 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 $M \cdot N + 2 \cdot M + 6 \cdot N$ 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`

FORTTRAN-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

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

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.DO
C X(2) = 1.DO
C X(3) = 1.DO
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.DO
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
C

LAST CARD OF DRIVER FOR LMDER EXAMPLE.

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
C

SUBROUTINE FCN FOR LMDER EXAMPLE.

```

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
C
C

IF (IFLAG .NE. 0) GO TO 5

INSERT PRINT STATEMENTS HERE WHEN NPRINT IS POSITIVE.

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
C

LAST CARD OF SUBROUTINE FCN.

END

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

FINAL L2 NORM OF THE RESIDUALS 0.9063596D-01

NUMBER OF FUNCTION EVALUATIONS 6

NUMBER OF JACOBIAN EVALUATIONS 5

EXIT PARAMETER 1

FINAL APPROXIMATE SOLUTION

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

Documentation for MINPACK subroutine LMSTR1

Double precision version

Argonne National Laboratory

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

March 1980

1. Purpose.

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

2. Subroutine and type statements.

```

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

```

3. Parameters.

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

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

```

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

```

END

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

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

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

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

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

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

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

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

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

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

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

INFO = 0 Improper input parameters.

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

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

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

INFO = 4 FVEC is orthogonal to the columns of the Jacobian to

machine precision.

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

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

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

Sections 4 and 5 contain more details about INFO.

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

WA is a work array of length LWA.

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

4. Successful completion.

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

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

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

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

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

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

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

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

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 $\text{LDFJAC} < N$, or $\text{TOL} < 0$, or $\text{LWA} < 5*N+M$.

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 $1.5 \cdot (N^2)$ 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 $N^2 + 2 \cdot M + 6 \cdot N$ double precision storage locations and N integer storage locations, in addition to the storage required by the program. There are no internally declared storage arrays.

7. Subprograms required.

USER-supplied FCN

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

FORTTRAN-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

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

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 LMSTR1 EXAMPLE.
C DOUBLE PRECISION VERSION
C
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(3,3), 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.D0
C X(2) = 1.D0
C X(3) = 1.D0
C
C LDFJAC = 3
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 LMSTR1(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,
*          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 //
*          5X,15H EXIT PARAMETER,16X,I10 //
*          5X,27H FINAL APPROXIMATE SOLUTION // 5X,3D15.7)
C

```

```

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

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

FINAL L2 NORM OF THE RESIDUALS  0.9063596D-01

EXIT PARAMETER 1

FINAL APPROXIMATE SOLUTION

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

```

Documentation for MINPACK subroutine LMSTR

Double precision version

Argonne National Laboratory

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

March 1980

1. Purpose.

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

2. Subroutine and type statements.

```

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

```

3. Parameters.

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

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

```

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

```

END

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

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

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

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

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

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

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

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

LDEFJAC 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

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

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

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

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 $N \text{ .LE. } 0$, or $M \text{ .LT. } N$, or $LDFJAC \text{ .LT. } N$, or $FTOL \text{ .LT. } 0.D0$, or $XTOL \text{ .LT. } 0.D0$, or $GTOL \text{ .LT. } 0.D0$, or $MAXFEV \text{ .LE. } 0$, or $FACTOR \text{ .LE. } 0.D0$.

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

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

6. Characteristics of the algorithm.

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

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

Storage. LMSTR requires $N^2 + 2M + 6N$ 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

FORTTRAN-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 LMSTR EXAMPLE.
C DOUBLE PRECISION VERSION
C
C *****
C INTEGER J, M, N, LDFJAC, MAXFEV, MODE, NPRINT, INFO, NFEV, NJEV, NWRITE
C INTEGER IPV T(3)
C DOUBLE PRECISION FTOL, XTOL, GTOL, FACTOR, FNORM
C DOUBLE PRECISION X(3), FVEC(15), FJAC(3, 3), DIAG(3), QTF(3),
*           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.DO
C X(2) = 1.DO
C X(3) = 1.DO
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.DO
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,
*           MAXFEV, DIAG, MODE, FACTOR, NPRINT, INFO, NFEV, NJEV,
*           IPV T, 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
C

LAST CARD OF DRIVER FOR LMSTR EXAMPLE.

```

END
SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)
INTEGER M,N,IFLAG
DOUBLE PRECISION X(N),FVEC(M),FJROW(N)

```

C
C
C

SUBROUTINE FCN FOR LMSTR EXAMPLE.

```

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
C
C

IF (IFLAG .NE. 0) GO TO 5

INSERT PRINT STATEMENTS HERE WHEN NPRINT IS POSITIVE.

```

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
C

LAST CARD OF SUBROUTINE FCN.

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 THIS PAGE ALLY
WAS INTENTIONALLY
LEFT BLANK

Documentation for MINPACK subroutine LMDIF1

Double precision version

Argonne National Laboratory

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

March 1980

1. Purpose.

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

2. Subroutine and type statements.

```

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

```

3. Parameters.

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

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

```

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

```

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

IFLAG to a negative integer.

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

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

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

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

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

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

INFO = 0 Improper input parameters.

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

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

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

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

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

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

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

Sections 4 and 5 contain more details about INFO.

IWA is an integer work array of length N.

WA is a work array of length LWA.

LWA is a positive integer input variable not less than

$M*N+5*N+M.$

4. Successful completion.

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

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

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

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

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

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

$$ENORM(D*(X-XSOL)) \leq TOL*ENORM(D*XSOL).$$

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

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

minimum. Therefore, termination caused by this test (INFO = 4) should be examined carefully.

5. Unsuccessful completion.

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

Improper input parameters. INFO is set to 0 if $N \leq 0$, or $M \leq N$, or $TOL \leq 0$, or $LWA \leq M*N+5*N+M$.

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 $M \cdot N + 2 \cdot M + 6 \cdot N$ 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

FORTTRAN-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

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

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

$$y(i) - (x(1) + u(i)/(v(i) \cdot x(2) + w(i) \cdot x(3))).$$

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

```

C      *****
      INTEGER J,M,N,INFO,LWA,NWRITE
      INTEGER IWA(3)
      DOUBLE PRECISION TOL, FNORM
      DOUBLE PRECISION X(3), FVEC(15), WA(75)
      DOUBLE PRECISION ENORM, DPMPAR
      EXTERNAL FCN

C
C      LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
C      DATA NWRITE /6/

C      M = 15
      N = 3

C
C      THE FOLLOWING STARTING VALUES PROVIDE A ROUGH FIT.
C
      X(1) = 1.DO
      X(2) = 1.DO
      X(3) = 1.DO

C
      LWA = 75

C
C      SET TOL TO THE SQUARE ROOT OF THE MACHINE PRECISION.
      UNLESS HIGH PRECISION SOLUTIONS ARE REQUIRED,
      THIS IS THE RECOMMENDED SETTING.
C
      TOL = DSQRT(DPMPAR(1))

C
      CALL LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)
      FNORM = ENORM(M,FVEC)
      WRITE (NWRITE,1000) FNORM,INFO,(X(J),J=1,N)
      STOP
1000 FORMAT (5X,31H FINAL L2 NORM OF THE RESIDUALS,D15.7 //
*          5X,15H EXIT PARAMETER,16X,I10 //
*          5X,27H FINAL APPROXIMATE SOLUTION // 5X,3D15.7)

C
C      LAST CARD OF DRIVER FOR LMDIF1 EXAMPLE.
C
      END
      SUBROUTINE FCN(M,N,X,FVEC,IFLAG)
      INTEGER M,N,IFLAG
      DOUBLE PRECISION X(N),FVEC(M)

C
C      SUBROUTINE FCN FOR LMDIF1 EXAMPLE.
C
      INTEGER I
      DOUBLE PRECISION TMP1,TMP2,TMP3
      DOUBLE PRECISION Y(15)
      DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),Y(8),
*          Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15)
*          /1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,
*          3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/
C

```

```
DO 10 I = 1, 15
  TMP1 = I
  TMP2 = 16 - I
  TMP3 = TMP1
  IF (I .GT. 8) TMP3 = TMP2
  FVEC(I) = Y(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
10  CONTINUE
  RETURN
```

C
C
C

LAST CARD OF SUBROUTINE FCN.

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
WAS THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

Documentation for MINPACK subroutine LMDIF

Double precision version

Argonne National Laboratory

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

March 1980

1. Purpose.

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

2. Subroutine and type statements.

```

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

```

3. Parameters.

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

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

MODE is an integer input variable. If MODE = 1, the variables will be scaled internally. If MODE = 2, the scaling is

specified by the input DIAG. Other values of MODE are equivalent to MODE = 1.

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

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

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

INFO = 0 Improper input parameters.

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

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

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

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

INFO = 5 Number of calls to FCN has reached or exceeded MAXFEV.

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

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

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

Sections 4 and 5 contain more details about INFO.

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

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

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

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

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

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

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

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

WA4 is a work array of length M.

4. Successful completion.

The accuracy of 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

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

If this condition is satisfied with $\text{XTOL} = 10^{**(-K)}$, 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 $\text{MODE} = 1$, 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 ($\text{INFO} = 4$) 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 $N \leq 0$, or $M < N$, or $\text{LDFJAC} < M$, or $\text{FTOL} < 0.00$, or $\text{XTOL} < 0.00$, or $\text{GTOL} < 0.00$, or $\text{MAXFEV} \leq 0$, or $\text{FACTOR} \leq 0.00$.

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 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 $M \cdot N + 2 \cdot M + 6 \cdot N$ 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

FORTTRAN-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 LMDIF EXAMPLE.
DOUBLE PRECISION VERSION

INTEGER J, M, N, MAXFEV, MODE, NPRINT, INFO, NFEV, LDFJAC, NWRITE
INTEGER IPVT(3)
DOUBLE PRECISION FTOL, XTOL, GTOL, EPSFCN, FACTOR, FNORM
DOUBLE PRECISION X(3), FVEC(15), DIAG(3), FJAC(15, 3), QTF(3),
* WA1(3), WA2(3), WA3(3), WA4(15)
DOUBLE PRECISION ENORM, DPMPAR
EXTERNAL FCN

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 = 15

SET FTOL AND XTOL TO THE SQUARE ROOT OF THE MACHINE PRECISION
AND GTOL TO ZERO. UNLESS HIGH PRECISION SOLUTIONS ARE
REQUIRED, THESE ARE THE RECOMMENDED SETTINGS.

FTOL = DSQRT(DPMPAR(1))
XTOL = DSQRT(DPMPAR(1))
GTOL = 0.D0

MAXFEV = 800
EPSFCN_i = 0.D0
MODE = 1
FACTOR = 1.D2
NPRINT = 0

CALL LMDIF(FCN, M, N, X, FVEC, FTOL, XTOL, GTOL, MAXFEV, EPSFCN,
* DIAG, MODE, FACTOR, NPRINT, INFO, NFEV, FJAC, LDFJAC,
* IPVT, QTF, WA1, WA2, WA3, WA4)

0.8241057D-01 0.1133037D+01 0.2343695D+01

THIS PAGE
FROM THIS PAGE
WAS THIS PAGE ALLY
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

FORTTRAN-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),
* 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-1,3.9D-1,
*   3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/

```

```

C
IF (IFLAG .EQ. 2) GO TO 20
DO 10 I = 1, 15
  TMP1 = I
  TMP2 = 16 - I
  TMP3 = TMP1
  IF (I .GT. 8) TMP3 = TMP2
  FVEC(I) = Y(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
10 CONTINUE
GO TO 40
20 CONTINUE
DO 30 I = 1, 15
  TMP1 = I
  TMP2 = 16 - I

```

```

C
C
C
C
ERROR INTRODUCED INTO NEXT STATEMENT FOR ILLUSTRATION.
CORRECTED STATEMENT SHOULD READ      TMP3 = TMP1 .

```

```

  TMP3 = TMP2
  IF (I .GT. 8) TMP3 = TMP2
  TMP4 = (X(2)*TMP2 + X(3)*TMP3)**2
  FJAC(I,1) = -1.DO
  FJAC(I,2) = TMP1*TMP2/TMP4
  FJAC(I,3) = TMP1*TMP3/TMP4
30 CONTINUE
40 CONTINUE
RETURN

```

```

C
C
C
LAST CARD OF SUBROUTINE FCN.

END

```

Results obtained with different compilers or machines may be different. In particular, the differences FVECP - FVEC are machine dependent.

FVEC

```

-0.1181606D+01 -0.1429655D+01 -0.1606344D+01
-0.1745269D+01 -0.1840654D+01 -0.1921586D+01
-0.1984141D+01 -0.2022537D+01 -0.2468977D+01
-0.2827562D+01 -0.3473582D+01 -0.4437612D+01
-0.6047662D+01 -0.9267761D+01 -0.1891806D+02

```

FVECP - FVEC

```

-0.7724666D-08 -0.3432405D-08 -0.2034843D-09

```

0.2313685D-08	0.4331078D-08	0.5984096D-08
0.7363281D-08	0.8531470D-08	0.1488591D-07
0.2335850D-07	0.3522012D-07	0.5301255D-07
0.8266660D-07	0.1419747D-06	0.3198990D-06

ERR

0.1141397D+00	0.9943516D-01	0.9674474D-01
0.9980447D-01	0.1073116D+00	0.1220445D+00
0.1526814D+00	0.1000000D+01	0.1000000D+01
0.1000000D+01	0.1000000D+01	0.1000000D+01
0.1000000D+01	0.1000000D+01	0.1000000D+01

CHAPTER 5
Program Listings

This chapter contains the double precision version of the MINPACK-1 program listings; both single and double precision versions of the subprograms are available with the MINPACK-1 package. The listings appear in the following (alphanumeric) order:

CHKDER, DOGLEG, ENORM, FDJAC1, FDJAC2, HYBRD, HYBRD1,
HYBRJ, HYBRJ1, LMDER, LMDER1, LMDIF, LMDIF1, LMPAR, LMSTR,
LMSTR1, QFORM, QRFAC, QRSOLV, RWUPDT, RIMPYQ, RIUPDT.

Functions SPMPAR (single precision) and DPMPAR (double precision), which provide the machine-dependent constants, appear at the end.

```

SUBROUTINE CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
INTEGER M,N,LDFJAC,MODE
DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),XP(N),FVECP(M),
*      ERR(M)
C *****
C
C SUBROUTINE CHKDER
C
C THIS SUBROUTINE CHECKS THE GRADIENTS OF M NONLINEAR FUNCTIONS
C IN N VARIABLES, EVALUATED AT A POINT X, FOR CONSISTENCY WITH
C THE FUNCTIONS THEMSELVES. THE USER MUST CALL CHKDER TWICE,
C FIRST WITH MODE = 1 AND THEN WITH MODE = 2.
C
C MODE = 1. ON INPUT, X MUST CONTAIN THE POINT OF EVALUATION.
C ON OUTPUT, XP IS SET TO A NEIGHBORING POINT.
C
C MODE = 2. ON INPUT, FVEC MUST CONTAIN THE FUNCTIONS AND THE
C ROWS OF FJAC MUST CONTAIN THE GRADIENTS
C OF THE RESPECTIVE FUNCTIONS EACH EVALUATED
C AT X, AND FVECP MUST CONTAIN THE FUNCTIONS
C EVALUATED AT XP.
C ON OUTPUT, ERR CONTAINS MEASURES OF CORRECTNESS OF
C THE RESPECTIVE GRADIENTS.
C
C THE SUBROUTINE DOES NOT PERFORM RELIABLY IF CANCELLATION OR
C ROUNDING ERRORS CAUSE A SEVERE LOSS OF SIGNIFICANCE IN THE
C EVALUATION OF A FUNCTION. THEREFORE, NONE OF THE COMPONENTS
C OF X SHOULD BE UNUSUALLY SMALL (IN PARTICULAR, ZERO) OR ANY
C OTHER VALUE WHICH MAY CAUSE LOSS OF SIGNIFICANCE.
C
C THE SUBROUTINE STATEMENT IS
C
C SUBROUTINE CHKDER(M,N,X,FVEC,FJAC,LDFJAC,XP,FVECP,MODE,ERR)
C
C WHERE
C
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C OF FUNCTIONS.
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
C OF VARIABLES.
C
C X IS AN INPUT ARRAY OF LENGTH N.
C
C FVEC IS AN ARRAY OF LENGTH M. ON INPUT WHEN MODE = 2,
C FVEC MUST CONTAIN THE FUNCTIONS EVALUATED AT X.
C
C FJAC IS AN M BY N ARRAY. ON INPUT WHEN MODE = 2,
C THE ROWS OF FJAC MUST CONTAIN THE GRADIENTS OF
C THE RESPECTIVE FUNCTIONS EVALUATED AT X.
C
C LDFJAC IS A POSITIVE INTEGER INPUT PARAMETER NOT LESS THAN M
C WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
CHDR0010
CHDR0020
CHDR0030
CHDR0040
CHDR0050
CHDR0060
CHDR0070
CHDR0080
CHDR0090
CHDR0100
CHDR0110
CHDR0120
CHDR0130
CHDR0140
CHDR0150
CHDR0160
CHDR0170
CHDR0180
CHDR0190
CHDR0200
CHDR0210
CHDR0220
CHDR0230
CHDR0240
CHDR0250
CHDR0260
CHDR0270
CHDR0280
CHDR0290
CHDR0300
CHDR0310
CHDR0320
CHDR0330
CHDR0340
CHDR0350
CHDR0360
CHDR0370
CHDR0380
CHDR0390
CHDR0400
CHDR0410
CHDR0420
CHDR0430
CHDR0440
CHDR0450
CHDR0460
CHDR0470
CHDR0480
CHDR0490
CHDR0500
CHDR0510
CHDR0520
CHDR0530
CHDR0540

```


C	MODE = 2.	CHDR1090
C		CHDR1100
	EPSF = FACTOR*EPSMCH	CHDR1110
	EPSLOG = DLOG10(EPS)	CHDR1120
	DO 30 I = 1, M	CHDR1130
	ERR(I) = ZERO	CHDR1140
30	CONTINUE	CHDR1150
	DO 50 J = 1, N	CHDR1160
	TEMP = DABS(X(J))	CHDR1170
	IF (TEMP .EQ. ZERO) TEMP = ONE	CHDR1180
	DO 40 I = 1, M	CHDR1190
	ERR(I) = ERR(I) + TEMP*FJAC(I,J)	CHDR1200
40	CONTINUE	CHDR1210
50	CONTINUE	CHDR1220
	DO 60 I = 1, M	CHDR1230
	TEMP = ONE	CHDR1240
	IF (FVEC(I) .NE. ZERO .AND. FVECP(I) .NE. ZERO	CHDR1250
*	.AND. DABS(FVECP(I)-FVEC(I)) .GE. EPSF*DABS(FVEC(I)))	CHDR1260
*	TEMP = EPS*DABS((FVECP(I)-FVEC(I))/EPS-ERR(I))	CHDR1270
*	/(DABS(FVEC(I)) + DABS(FVECP(I)))	CHDR1280
	ERR(I) = ONE	CHDR1290
	IF (TEMP .GT. EPSMCH .AND. TEMP .LT. EPS)	CHDR1300
*	ERR(I) = (DLOG10(TEMP) - EPSLOG)/EPSLOG	CHDR1310
	IF (TEMP .GE. EPS) ERR(I) = ZERO	CHDR1320
60	CONTINUE	CHDR1330
70	CONTINUE	CHDR1340
C		CHDR1350
	RETURN	CHDR1360
C		CHDR1370
C	LAST CARD OF SUBROUTINE CHKDER.	CHDR1380
C		CHDR1390
	END	CHDR1400

	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		DOGL0070
C	SUBROUTINE DOGLEG	DOGL0080
C		DOGL0090
C	GIVEN AN M BY N MATRIX A, AN N BY N NONSINGULAR DIAGONAL	DOGL0100
C	MATRIX D, AN M-VECTOR B, AND A POSITIVE NUMBER DELTA, THE	DOGL0110
C	PROBLEM IS TO DETERMINE THE CONVEX COMBINATION X OF THE	DOGL0120
C	GAUSS-NEWTON AND SCALED GRADIENT DIRECTIONS THAT MINIMIZES	DOGL0130
C	(A*X - B) IN THE LEAST SQUARES SENSE, SUBJECT TO THE	DOGL0140
C	RESTRICTION THAT THE EUCLIDEAN NORM OF D*X BE AT MOST DELTA.	DOGL0150
C		DOGL0160
C	THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM	DOGL0170
C	IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE	DOGL0180
C	QR FACTORIZATION OF A. THAT IS, IF $A = Q \cdot R$, WHERE Q HAS	DOGL0190
C	ORTHOGONAL COLUMNS AND R IS AN UPPER TRIANGULAR MATRIX,	DOGL0200
C	THEN DOGLEG EXPECTS THE FULL UPPER TRIANGLE OF R AND	DOGL0210
C	THE FIRST N COMPONENTS OF (Q TRANSPOSE)*B.	DOGL0220
C		DOGL0230
C	THE SUBROUTINE STATEMENT IS	DOGL0240
C		DOGL0250
C	SUBROUTINE DOGLEG(N,R,LR,DIAG,QTB,DELTA,X,WA1,WA2)	DOGL0260
C		DOGL0270
C	WHERE	DOGL0280
C		DOGL0290
C	N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R.	DOGL0300
C		DOGL0310
C	R IS AN INPUT ARRAY OF LENGTH LR WHICH MUST CONTAIN THE UPPER	DOGL0320
C	TRIANGULAR MATRIX R STORED BY ROWS.	DOGL0330
C		DOGL0340
C	LR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN	DOGL0350
C	$(N \cdot (N+1)) / 2$.	DOGL0360
C		DOGL0370
C	DIAG IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE	DOGL0380
C	DIAGONAL ELEMENTS OF THE MATRIX D.	DOGL0390
C		DOGL0400
C	QTB IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE FIRST	DOGL0410
C	N ELEMENTS OF THE VECTOR (Q TRANSPOSE)*B.	DOGL0420
C		DOGL0430
C	DELTA IS A POSITIVE INPUT VARIABLE WHICH SPECIFIES AN UPPER	DOGL0440
C	BOUND ON THE EUCLIDEAN NORM OF D*X.	DOGL0450
C		DOGL0460
C	X IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE DESIRED	DOGL0470
C	CONVEX COMBINATION OF THE GAUSS-NEWTON DIRECTION AND THE	DOGL0480
C	SCALED GRADIENT DIRECTION.	DOGL0490
C		DOGL0500
C	WA1 AND WA2 ARE WORK ARRAYS OF LENGTH N.	DOGL0510
C		DOGL0520
C	SUBPROGRAMS CALLED	DOGL0530
C		DOGL0540
C	MINPACK-SUPPLIED ... DPMPAR,ENORM	

C		DOGL0550
C	FORTRAN-SUPPLIED ... DABS, DMAX1, DMIN1, DSQRT	DOGL0560
C		DOGL0570
C	ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.	DOGL0580
C	BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE	DOGL0590
C		DOGL0600
C	*****	DOGL0610
	INTEGER I, J, JJ, JP1, K, L	DOGL0620
	DOUBLE PRECISION ALPHA, BNORM, EPSMCH, GNORM, ONE, QNORM, SGNORM, SUM,	DOGL0630
	* TEMP, ZERO	DOGL0640
	DOUBLE PRECISION DPMPAR, ENORM	DOGL0650
	DATA ONE, ZERO /1.0D0, 0.0D0/	DOGL0660
C		DOGL0670
C	EPSMCH IS THE MACHINE PRECISION.	DOGL0680
C		DOGL0690
	EPSMCH = DPMPAR(1)	DOGL0700
C		DOGL0710
C	FIRST, CALCULATE THE GAUSS-NEWTON DIRECTION.	DOGL0720
C		DOGL0730
	JJ = (N*(N + 1))/2 + 1	DOGL0740
	DO 50 K = 1, N	DOGL0750
	J = N - K + 1	DOGL0760
	JP1 = J + 1	DOGL0770
	JJ = JJ - K	DOGL0780
	L = JJ + 1	DOGL0790
	SUM = ZERO	DOGL0800
	IF (N .LT. JP1) GO TO 20	DOGL0810
	DO 10 I = JP1, N	DOGL0820
	SUM = SUM + R(L)*X(I)	DOGL0830
	L = L + 1	DOGL0840
10	CONTINUE	DOGL0850
20	CONTINUE	DOGL0860
	TEMP = R(JJ)	DOGL0870
	IF (TEMP .NE. ZERO) GO TO 40	DOGL0880
	L = J	DOGL0890
	DO 30 I = 1, J	DOGL0900
	TEMP = DMAX1(TEMP, DABS(R(L)))	DOGL0910
	L = L + N - I	DOGL0920
30	CONTINUE	DOGL0930
	TEMP = EPSMCH*TEMP	DOGL0940
	IF (TEMP .EQ. ZERO) TEMP = EPSMCH	DOGL0950
40	CONTINUE	DOGL0960
	X(J) = (QTB(J) - SUM)/TEMP	DOGL0970
50	CONTINUE	DOGL0980
C		DOGL0990
C	TEST WHETHER THE GAUSS-NEWTON DIRECTION IS ACCEPTABLE.	DOGL1000
C		DOGL1010
	DO 60 J = 1, N	DOGL1020
	WA1(J) = ZERO	DOGL1030
	WA2(J) = DIAG(J)*X(J)	DOGL1040
60	CONTINUE	DOGL1050
	QNORM = ENORM(N, WA2)	DOGL1060
	IF (QNORM .LE. DELTA) GO TO 140	DOGL1070
C		DOGL1080

```

C THE GAUSS-NEWTON DIRECTION IS NOT ACCEPTABLE. DOGL1090
C NEXT, CALCULATE THE SCALED GRADIENT DIRECTION. DOGL1100
C DOGL1110
L = 1 DOGL1120
DO 80 J = 1, N DOGL1130
TEMP = QTB(J) DOGL1140
DO 70 I = J, N DOGL1150
WA1(I) = WA1(I) + R(L)*TEMP DOGL1160
L = L + 1 DOGL1170
70 CONTINUE DOGL1180
WA1(J) = WA1(J)/DIAG(J) DOGL1190
80 CONTINUE DOGL1200
DOGL1210
C DOGL1220
C CALCULATE THE NORM OF THE SCALED GRADIENT AND TEST FOR DOGL1230
C THE SPECIAL CASE IN WHICH THE SCALED GRADIENT IS ZERO. DOGL1240
C DOGL1250
GNORM = ENORM(N,WA1) DOGL1260
SGNORM = ZERO DOGL1270
ALPHA = DELTA/QNORM DOGL1280
IF (GNORM .EQ. ZERO) GO TO 120 DOGL1290
DOGL1300
C DOGL1310
C CALCULATE THE POINT ALONG THE SCALED GRADIENT DOGL1320
C AT WHICH THE QUADRATIC IS MINIMIZED. DOGL1330
C DOGL1340
DO 90 J = 1, N DOGL1350
WA1(J) = (WA1(J)/GNORM)/DIAG(J) DOGL1360
90 CONTINUE DOGL1370
L = 1 DOGL1380
DO 110 J = 1, N DOGL1390
SUM = ZERO DOGL1400
DO 100 I = J, N DOGL1410
SUM = SUM + R(L)*WA1(I) DOGL1420
L = L + 1 DOGL1430
100 CONTINUE DOGL1440
WA2(J) = SUM DOGL1450
110 CONTINUE DOGL1460
TEMP = ENORM(N,WA2) DOGL1470
SGNORM = (GNORM/TEMP)/TEMP DOGL1480
DOGL1490
C DOGL1500
C TEST WHETHER THE SCALED GRADIENT DIRECTION IS ACCEPTABLE. DOGL1510
C DOGL1520
ALPHA = ZERO DOGL1530
IF (SGNORM .GE. DELTA) GO TO 120 DOGL1540
DOGL1550
C DOGL1560
C THE SCALED GRADIENT DIRECTION IS NOT ACCEPTABLE. DOGL1570
C FINALLY, CALCULATE THE POINT ALONG THE DOGLEG DOGL1580
C AT WHICH THE QUADRATIC IS MINIMIZED. DOGL1590
C DOGL1600
BNORM = ENORM(N,QTB) DOGL1610
TEMP = (BNORM/GNORM)*(BNORM/QNORM)*(SGNORM/DELTA) DOGL1620
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 DOGLEG.	DOGL1750
C		DOGL1760
	END	DOGL1770

DOUBLE PRECISION FUNCTION ENORM(N,X)

INTEGER N

DOUBLE PRECISION X(N)

FUNCTION ENORM

GIVEN AN N-VECTOR X, THIS FUNCTION CALCULATES THE
EUCLIDEAN NORM OF X.

THE EUCLIDEAN NORM IS COMPUTED BY ACCUMULATING THE SUM OF
SQUARES IN THREE DIFFERENT SUMS. THE SUMS OF SQUARES FOR THE
SMALL AND LARGE COMPONENTS ARE SCALED SO THAT NO OVERFLOWS
OCCUR. NON-DESTRUCTIVE UNDERFLOWS ARE PERMITTED. UNDERFLOWS
AND OVERFLOWS DO NOT OCCUR IN THE COMPUTATION OF THE UNSCALED
SUM OF SQUARES FOR THE INTERMEDIATE COMPONENTS.
THE DEFINITIONS OF SMALL, INTERMEDIATE AND LARGE COMPONENTS
DEPEND ON TWO CONSTANTS, RDWARF AND RGIANT. THE MAIN
RESTRICTIONS ON THESE CONSTANTS ARE THAT RDWARF**2 NOT
UNDERFLOW AND RGIANT**2 NOT OVERFLOW. THE CONSTANTS
GIVEN HERE ARE SUITABLE FOR EVERY KNOWN COMPUTER.

THE FUNCTION STATEMENT IS

DOUBLE PRECISION FUNCTION ENORM(N,X)

WHERE

N IS A POSITIVE INTEGER INPUT VARIABLE.

X IS AN INPUT ARRAY OF LENGTH N.

SUBPROGRAMS CALLED

FORTTRAN-SUPPLIED ... DABS,DSQRT

ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE

INTEGER I

DOUBLE PRECISION AGIANT,FLOATN,ONE,RDWARF,RGIANT,S1,S2,S3,XABS,
* X1MAX,X3MAX,ZERO

DATA ONE,ZERO,RDWARF,RGIANT /1.0D0,0.0D0,3.834D-20,1.304D19/

S1 = ZERO

S2 = ZERO

S3 = ZERO

X1MAX = ZERO

X3MAX = ZERO

FLOATN = N

AGIANT = RGIANT/FLOATN

DO 90 I = 1, N

XABS = DABS(X(I))

IF (XABS .GT. RDWARF .AND. XABS .LT. AGIANT) GO TO 70

ENRM0010

ENRM0020

ENRM0030

ENRM0040

ENRM0050

ENRM0060

ENRM0070

ENRM0080

ENRM0090

ENRM0100

ENRM0110

ENRM0120

ENRM0130

ENRM0140

ENRM0150

ENRM0160

ENRM0170

ENRM0180

ENRM0190

ENRM0200

ENRM0210

ENRM0220

ENRM0230

ENRM0240

ENRM0250

ENRM0260

ENRM0270

ENRM0280

ENRM0290

ENRM0300

ENRM0310

ENRM0320

ENRM0330

ENRM0340

ENRM0350

ENRM0360

ENRM0370

ENRM0380

ENRM0390

ENRM0400

ENRM0410

ENRM0420

ENRM0430

ENRM0440

ENRM0450

ENRM0460

ENRM0470

ENRM0480

ENRM0490

ENRM0500

ENRM0510

ENRM0520

ENRM0530

ENRM0540

	IF (XABS .LE. RDWARF) GO TO 30	ENRM0550
C		ENRM0560
C	SUM FOR LARGE COMPONENTS.	ENRM0570
C		ENRM0580
	IF (XABS .LE. X1MAX) GO TO 10	ENRM0590
	S1 = ONE + S1*(X1MAX/XABS)**2	ENRM0600
	X1MAX = XABS	ENRM0610
	GO TO 20	ENRM0620
10	CONTINUE	ENRM0630
	S1 = S1 + (XABS/X1MAX)**2	ENRM0640
20	CONTINUE	ENRM0650
	GO TO 60	ENRM0660
30	CONTINUE	ENRM0670
		ENRM0680
C		ENRM0690
C	SUM FOR SMALL COMPONENTS.	ENRM0700
C		ENRM0710
	IF (XABS .LE. X3MAX) GO TO 40	ENRM0720
	S3 = ONE + S3*(X3MAX/XABS)**2	ENRM0730
	X3MAX = XABS	ENRM0740
	GO TO 50	ENRM0750
40	CONTINUE	ENRM0760
	IF (XABS .NE. ZERO) S3 = S3 + (XABS/X3MAX)**2	ENRM0770
50	CONTINUE	ENRM0780
60	CONTINUE	ENRM0790
	GO TO 80	ENRM0800
70	CONTINUE	ENRM0810
		ENRM0820
C		ENRM0830
C	SUM FOR INTERMEDIATE COMPONENTS.	ENRM0840
C		ENRM0850
	S2 = S2 + XABS**2	ENRM0860
80	CONTINUE	ENRM0870
90	CONTINUE	ENRM0880
		ENRM0890
C		ENRM0900
C	CALCULATION OF NORM.	ENRM0910
C		ENRM0920
	IF (S1 .EQ. ZERO) GO TO 100	ENRM0930
	ENORM = X1MAX*DSQRT(S1+(S2/X1MAX)/X1MAX)	ENRM0940
	GO TO 130	ENRM0950
100	CONTINUE	ENRM0960
	IF (S2 .EQ. ZERO) GO TO 110	ENRM0970
	IF (S2 .GE. X3MAX)	ENRM0980
	* ENORM = DSQRT(S2*(ONE+(X3MAX/S2)*(X3MAX*S3)))	ENRM0990
	IF (S2 .LT. X3MAX)	ENRM1000
	* ENORM = DSQRT(X3MAX*((S2/X3MAX)+(X3MAX*S3)))	ENRM1010
	GO TO 120	ENRM1020
110	CONTINUE	ENRM1030
	ENORM = X3MAX*DSQRT(S3)	ENRM1040
120	CONTINUE	ENRM1050
130	CONTINUE	ENRM1060
	RETURN	ENRM1070
		ENRM1080
C		ENRM1090
C	LAST CARD OF FUNCTION ENORM.	ENRM1100
C		ENRM1110
	END	ENRM1120

	SUBROUTINE FDJAC1(FCN,N,X,FVEC,FJAC,LDFJAC,IFLAG,ML,MU,EPSFCN,	FDJ10010
	* WA1,WA2)	FDJ10020
	INTEGER N,LDFJAC,IFLAG,ML,MU	FDJ10030
	DOUBLE PRECISION EPSFCN	FDJ10040
	DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N),WA1(N),WA2(N)	FDJ10050
	*****	FDJ10060
C		FDJ10070
C		FDJ10080
C	SUBROUTINE FDJAC1	FDJ10090
C		FDJ10100
C	THIS SUBROUTINE COMPUTES A FORWARD-DIFFERENCE APPROXIMATION	FDJ10110
C	TO THE N BY N JACOBIAN MATRIX ASSOCIATED WITH A SPECIFIED	FDJ10120
C	PROBLEM OF N FUNCTIONS IN N VARIABLES. IF THE JACOBIAN HAS	FDJ10130
C	A BANDED FORM, THEN FUNCTION EVALUATIONS ARE SAVED BY ONLY	FDJ10140
C	APPROXIMATING THE NONZERO TERMS.	FDJ10150
C		FDJ10160
C	THE SUBROUTINE STATEMENT IS	FDJ10170
C		FDJ10180
C	SUBROUTINE FDJAC1(FCN,N,X,FVEC,FJAC,LDFJAC,IFLAG,ML,MU,EPSFCN,	FDJ10190
C	WA1,WA2)	FDJ10200
C		FDJ10210
C	WHERE	FDJ10220
C		FDJ10230
C	FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH	FDJ10240
C	CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED	FDJ10250
C	IN AN EXTERNAL STATEMENT IN THE USER CALLING	FDJ10260
C	PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.	FDJ10270
C		FDJ10280
C	SUBROUTINE FCN(N,X,FVEC,IFLAG)	FDJ10290
C	INTEGER N,IFLAG	FDJ10300
C	DOUBLE PRECISION X(N),FVEC(N)	FDJ10310
C	-----	FDJ10320
C	CALCULATE THE FUNCTIONS AT X AND	FDJ10330
C	RETURN THIS VECTOR IN FVEC.	FDJ10340
C	-----	FDJ10350
C	RETURN	FDJ10360
C	END	FDJ10370
C		FDJ10380
C	THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS	FDJ10390
C	THE USER WANTS TO TERMINATE EXECUTION OF FDJAC1.	FDJ10400
C	IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.	FDJ10410
C		FDJ10420
C	N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	FDJ10430
C	OF FUNCTIONS AND VARIABLES.	FDJ10440
C		FDJ10450
C	X IS AN INPUT ARRAY OF LENGTH N.	FDJ10460
C		FDJ10470
C	FVEC IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE	FDJ10480
C	FUNCTIONS EVALUATED AT X.	FDJ10490
C		FDJ10500
C	FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE	FDJ10510
C	APPROXIMATION TO THE JACOBIAN MATRIX EVALUATED AT X.	FDJ10520
C		FDJ10530
C	LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N	FDJ10540
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.	

C		FDJ10550
C	IFLAG IS AN INTEGER VARIABLE WHICH CAN BE USED TO TERMINATE	FDJ10560
C	THE EXECUTION OF FDJAC1. SEE DESCRIPTION OF FCN.	FDJ10570
C		FDJ10580
C	ML IS A NONNEGATIVE INTEGER INPUT VARIABLE WHICH SPECIFIES	FDJ10590
C	THE NUMBER OF SUBDIAGONALS WITHIN THE BAND OF THE	FDJ10600
C	JACOBIAN MATRIX. IF THE JACOBIAN IS NOT BANDED, SET	FDJ10610
C	ML TO AT LEAST N - 1.	FDJ10620
C		FDJ10630
C	EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE	FDJ10640
C	STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS	FDJ10650
C	APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE	FDJ10660
C	FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS	FDJ10670
C	THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE	FDJ10680
C	ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE	FDJ10690
C	PRECISION.	FDJ10700
C		FDJ10710
C	MU IS A NONNEGATIVE INTEGER INPUT VARIABLE WHICH SPECIFIES	FDJ10720
C	THE NUMBER OF SUPERDIAGONALS WITHIN THE BAND OF THE	FDJ10730
C	JACOBIAN MATRIX. IF THE JACOBIAN IS NOT BANDED, SET	FDJ10740
C	MU TO AT LEAST N - 1.	FDJ10750
C		FDJ10760
C	WA1 AND WA2 ARE WORK ARRAYS OF LENGTH N. IF ML + MU + 1 IS AT	FDJ10770
C	LEAST N, THEN THE JACOBIAN IS CONSIDERED DENSE, AND WA2 IS	FDJ10780
C	NOT REFERENCED.	FDJ10790
C		FDJ10800
C	SUBPROGRAMS CALLED	FDJ10810
C		FDJ10820
C	MINPACK-SUPPLIED ... DPMPAR	FDJ10830
C		FDJ10840
C	FORTRAN-SUPPLIED ... DABS, DMAX1, DSQRT	FDJ10850
C		FDJ10860
C	ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.	FDJ10870
C	BURTON S. GARROW, KENNETH E. HILLSTROM, JORGE J. MORE	FDJ10880
C		FDJ10890
C	*****	FDJ10900
C	INTEGER I, J, K, MSUM	FDJ10910
C	DOUBLE PRECISION EPS, EPSMCH, H, TEMP, ZERO	FDJ10920
C	DOUBLE PRECISION DPMPAR	FDJ10930
C	DATA ZERO /0.0D0/	FDJ10940
C		FDJ10950
C	EPSMCH IS THE MACHINE PRECISION.	FDJ10960
C		FDJ10970
C	EPSMCH = DPMPAR(1)	FDJ10980
C		FDJ10990
C	EPS = DSQRT(DMAX1(EPSFCN, EPSMCH))	FDJ11000
C	MSUM = ML + MU + 1	FDJ11010
C	IF (MSUM .LT. N) GO TO 40	FDJ11020
C		FDJ11030
C	COMPUTATION OF DENSE APPROXIMATE JACOBIAN.	FDJ11040
C		FDJ11050
C	DO 20 J = 1, N	FDJ11060
C	TEMP = X(J)	FDJ11070
C	H = EPS*DABS(TEMP)	FDJ11080

	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
	INTEGER M,N,LDFJAC,IFLAG	FDJ20020
	DOUBLE PRECISION EPSFCN	FDJ20030
	DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),WA(M)	FDJ20040
	*****	FDJ20050
C		FDJ20060
C		FDJ20070
C	SUBROUTINE FDJAC2	FDJ20080
C		FDJ20090
C	THIS SUBROUTINE COMPUTES A FORWARD-DIFFERENCE APPROXIMATION	FDJ20100
C	TO THE M BY N JACOBIAN MATRIX ASSOCIATED WITH A SPECIFIED	FDJ20110
C	PROBLEM OF M FUNCTIONS IN N VARIABLES.	FDJ20120
C		FDJ20130
C	THE SUBROUTINE STATEMENT IS	FDJ20140
C		FDJ20150
C	SUBROUTINE FDJAC2(FCN,M,N,X,FVEC,FJAC,LDFJAC,IFLAG,EPSFCN,WA)	FDJ20160
C		FDJ20170
C	WHERE	FDJ20180
C		FDJ20190
C	FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH	FDJ20200
C	CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED	FDJ20210
C	IN AN EXTERNAL STATEMENT IN THE USER CALLING	FDJ20220
C	PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.	FDJ20230
C		FDJ20240
C	SUBROUTINE FCN(M,N,X,FVEC,IFLAG)	FDJ20250
C	INTEGER M,N,IFLAG	FDJ20260
C	DOUBLE PRECISION X(N),FVEC(M)	FDJ20270
C	-----	FDJ20280
C	CALCULATE THE FUNCTIONS AT X AND	FDJ20290
C	RETURN THIS VECTOR IN FVEC.	FDJ20300
C	-----	FDJ20310
C	RETURN	FDJ20320
C	END	FDJ20330
C		FDJ20340
C	THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS	FDJ20350
C	THE USER WANTS TO TERMINATE EXECUTION OF FDJAC2.	FDJ20360
C	IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.	FDJ20370
C		FDJ20380
C	M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	FDJ20390
C	OF FUNCTIONS.	FDJ20400
C		FDJ20410
C	N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	FDJ20420
C	OF VARIABLES. N MUST NOT EXCEED M.	FDJ20430
C		FDJ20440
C	X IS AN INPUT ARRAY OF LENGTH N.	FDJ20450
C		FDJ20460
C	FVEC IS AN INPUT ARRAY OF LENGTH M WHICH MUST CONTAIN THE	FDJ20470
C	FUNCTIONS EVALUATED AT X.	FDJ20480
C		FDJ20490
C	FJAC IS AN OUTPUT M BY N ARRAY WHICH CONTAINS THE	FDJ20500
C	APPROXIMATION TO THE JACOBIAN MATRIX EVALUATED AT X.	FDJ20510
C		FDJ20520
C	LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M	FDJ20530
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.	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		FDJ20570
C	EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE	FDJ20580
C	STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS	FDJ20590
C	APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE	FDJ20600
C	FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS	FDJ20610
C	THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE	FDJ20620
C	ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE	FDJ20630
C	PRECISION.	FDJ20640
C		FDJ20650
C	WA IS A WORK ARRAY OF LENGTH M.	FDJ20660
C		FDJ20670
C	SUBPROGRAMS CALLED	FDJ20680
C		FDJ20690
C	USER-SUPPLIED FCN	FDJ20700
C		FDJ20710
C	MINPACK-SUPPLIED ... DPMPAR	FDJ20720
C		FDJ20730
C	FORTTRAN-SUPPLIED ... DABS, DMAX1, DSQRT	FDJ20740
C		FDJ20750
C	ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.	FDJ20760
C	BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE	FDJ20770
C		FDJ20780
C	*****	FDJ20790
	INTEGER I, J	FDJ20800
	DOUBLE PRECISION EPS, EPSMCH, H, TEMP, ZERO	FDJ20810
	DOUBLE PRECISION DPMPAR	FDJ20820
	DATA ZERO /0.000/	FDJ20830
C		FDJ20840
C	EPSMCH IS THE MACHINE PRECISION.	FDJ20850
C		FDJ20860
	EPSMCH = DPMPAR(1)	FDJ20870
C		FDJ20880
	EPS = DSQRT(DMAX1(EPSFCN, EPSMCH))	FDJ20890
	DO 20 J = 1, N	FDJ20900
	TEMP = X(J)	FDJ20910
	H = EPS*DABS(TEMP)	FDJ20920
	IF (H .EQ. ZERO) H = EPS	FDJ20930
	X(J) = TEMP + H	FDJ20940
	CALL FCN(M, N, X, WA, IFLAG)	FDJ20950
	IF (IFLAG .LT. 0) GO TO 30	FDJ20960
	X(J) = TEMP	FDJ20970
	DO 10 I = 1, M	FDJ20980
	FJAC(I, J) = (WA(I) - FVEC(I))/H	FDJ20990
10	CONTINUE	FDJ21000
20	CONTINUE	FDJ21010
30	CONTINUE	FDJ21020
	RETURN	FDJ21030
C		FDJ21040
C	LAST CARD OF SUBROUTINE FDJAC2.	FDJ21050
C		FDJ21060
	END	FDJ21070

	SUBROUTINE HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,DIAG,	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		HYBD0110
C	SUBROUTINE HYBRD	HYBD0120
C		HYBD0130
C	THE PURPOSE OF HYBRD IS TO FIND A ZERO OF A SYSTEM OF	HYBD0140
C	N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION	HYBD0150
C	OF THE POWELL HYBRID METHOD. THE USER MUST PROVIDE A	HYBD0160
C	SUBROUTINE WHICH CALCULATES THE FUNCTIONS. THE JACOBIAN IS	HYBD0170
C	THEN CALCULATED BY A FORWARD-DIFFERENCE APPROXIMATION.	HYBD0180
C		HYBD0190
C	THE SUBROUTINE STATEMENT IS	HYBD0200
C		HYBD0210
C	SUBROUTINE HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,	HYBD0220
C	DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,FJAC,	HYBD0230
C	LDFJAC,R,LR,QTF,WA1,WA2,WA3,WA4)	HYBD0240
C		HYBD0250
C	WHERE	HYBD0260
C		HYBD0270
C	FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH	HYBD0280
C	CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED	HYBD0290
C	IN AN EXTERNAL STATEMENT IN THE USER CALLING	HYBD0300
C	PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.	HYBD0310
C		HYBD0320
C	SUBROUTINE FCN(N,X,FVEC,IFLAG)	HYBD0330
C	INTEGER N,IFLAG	HYBD0340
C	DOUBLE PRECISION X(N),FVEC(N)	HYBD0350
C	-----	HYBD0360
C	CALCULATE THE FUNCTIONS AT X AND	HYBD0370
C	RETURN THIS VECTOR IN FVEC.	HYBD0380
C	-----	HYBD0390
C	RETURN	HYBD0400
C	END	HYBD0410
C		HYBD0420
C	THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS	HYBD0430
C	THE USER WANTS TO TERMINATE EXECUTION OF HYBRD.	HYBD0440
C	IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.	HYBD0450
C		HYBD0460
C	N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	HYBD0470
C	OF FUNCTIONS AND VARIABLES.	HYBD0480
C		HYBD0490
C	X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN	HYBD0500
C	AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X	HYBD0510
C	CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.	HYBD0520
C		HYBD0530
C	FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS	HYBD0540
C	THE FUNCTIONS EVALUATED AT THE OUTPUT X.	

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

C	INFO IS SET AS FOLLOWS.	HYBD1090
C		HYBD1100
C	INFO = 0 IMPROPER INPUT PARAMETERS.	HYBD1110
C		HYBD1120
C	INFO = 1 RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES	HYBD1130
C	IS AT MOST XTOL.	HYBD1140
C		HYBD1150
C	INFO = 2 NUMBER OF CALLS TO FCN HAS REACHED OR EXCEEDED	HYBD1160
C	MAXFEV.	HYBD1170
C		HYBD1180
C	INFO = 3 XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN	HYBD1190
C	THE APPROXIMATE SOLUTION X IS POSSIBLE.	HYBD1200
C		HYBD1210
C	INFO = 4 ITERATION IS NOT MAKING GOOD PROGRESS, AS	HYBD1220
C	MEASURED BY THE IMPROVEMENT FROM THE LAST	HYBD1230
C	FIVE JACOBIAN EVALUATIONS.	HYBD1240
C		HYBD1250
C	INFO = 5 ITERATION IS NOT MAKING GOOD PROGRESS, AS	HYBD1260
C	MEASURED BY THE IMPROVEMENT FROM THE LAST	HYBD1270
C	TEN ITERATIONS.	HYBD1280
C		HYBD1290
C	NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF	HYBD1300
C	CALLS TO FCN.	HYBD1310
C		HYBD1320
C	FJAC IS AN OUTPUT N BY N ARRAY WHICH CONTAINS THE	HYBD1330
C	ORTHOGONAL MATRIX Q PRODUCED BY THE QR FACTORIZATION	HYBD1340
C	OF THE FINAL APPROXIMATE JACOBIAN.	HYBD1350
C		HYBD1360
C	LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N	HYBD1370
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.	HYBD1380
C		HYBD1390
C	R IS AN OUTPUT ARRAY OF LENGTH LR WHICH CONTAINS THE	HYBD1400
C	UPPER TRIANGULAR MATRIX PRODUCED BY THE QR FACTORIZATION	HYBD1410
C	OF THE FINAL APPROXIMATE JACOBIAN, STORED ROWWISE.	HYBD1420
C		HYBD1430
C	LR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN	HYBD1440
C	$(N*(N+1))/2$.	HYBD1450
C		HYBD1460
C	QTF IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS	HYBD1470
C	THE VECTOR $(Q \text{ TRANSPOSE}) * FVEC$.	HYBD1480
C		HYBD1490
C	WA1, WA2, WA3, AND WA4 ARE WORK ARRAYS OF LENGTH N.	HYBD1500
C		HYBD1510
C	SUBPROGRAMS CALLED	HYBD1520
C		HYBD1530
C	USER-SUPPLIED FCN	HYBD1540
C		HYBD1550
C	MINPACK-SUPPLIED . . . DOGLEG, DPMPAR, ENORM, FDJAC1,	HYBD1560
C	QFORM, QRFAC, RIMPYQ, RIUPDT	HYBD1570
C		HYBD1580
C	FORTTRAN-SUPPLIED . . . DABS, DMAX1, DMIN1, MINO, MOD	HYBD1590
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
	INTEGER I, IFLAG, ITER, J, JM1, L, MSUM, NCFAIL, NCSUC, NSLOW1, NSLOW2	HYBD1650
	INTEGER IWA(1)	HYBD1660
	LOGICAL JEVAL, SING	HYBD1670
	DOUBLE PRECISION ACTRED, DELTA, EPSMCH, FNORM, FNORM1, ONE, PNORM,	HYBD1680
	* PRERED, P1, P5, P001, P0001, RATIO, SUM, TEMP, XNORM,	HYBD1690
	* ZERO	HYBD1700
	DOUBLE PRECISION DPMPAR, ENORM	HYBD1710
	DATA ONE, P1, P5, P001, P0001, ZERO	HYBD1720
	* /1.0D0, 1.0D-1, 5.0D-1, 1.0D-3, 1.0D-4, 0.0D0/	HYBD1730
C		HYBD1740
C	EPSMCH IS THE MACHINE PRECISION.	HYBD1750
C		HYBD1760
	EPSMCH = DPMPAR(1)	HYBD1770
C		HYBD1780
	INFO = 0	HYBD1790
	IFLAG = 0	HYBD1800
	NFEV = 0	HYBD1810
C		HYBD1820
C	CHECK THE INPUT PARAMETERS FOR ERRORS.	HYBD1830
C		HYBD1840
	IF (N .LE. 0 .OR. XTOL .LT. ZERO .OR. MAXFEV .LE. 0	HYBD1850
	* .OR. ML .LT. 0 .OR. MU .LT. 0 .OR. FACTOR .LE. ZERO	HYBD1860
	* .OR. LDFJAC .LT. N .OR. LR .LT. (N*(N + 1))/2) GO TO 300	HYBD1870
	IF (MODE .NE. 2) GO TO 20	HYBD1880
	DO 10 J = 1, N	HYBD1890
	IF (DIAG(J) .LE. ZERO) GO TO 300	HYBD1900
	10 CONTINUE	HYBD1910
	20 CONTINUE	HYBD1920
C		HYBD1930
C	EVALUATE THE FUNCTION AT THE STARTING POINT	HYBD1940
C	AND CALCULATE ITS NORM.	HYBD1950
C		HYBD1960
	IFLAG = 1	HYBD1970
	CALL FCN(N, X, FVEC, IFLAG)	HYBD1980
	NFEV = 1	HYBD1990
	IF (IFLAG .LT. 0) GO TO 300	HYBD2000
	FNORM = ENORM(N, FVEC)	HYBD2010
C		HYBD2020
C	DETERMINE THE NUMBER OF CALLS TO FCN NEEDED TO COMPUTE	HYBD2030
C	THE JACOBIAN MATRIX.	HYBD2040
C		HYBD2050
	MSUM = MINO(ML+MU+1, N)	HYBD2060
C		HYBD2070
C	INITIALIZE ITERATION COUNTER AND MONITORS.	HYBD2080
C		HYBD2090
	ITER = 1	HYBD2100
	NCSUC = 0	HYBD2110
	NCFAIL = 0	HYBD2120
	NSLOW1 = 0	HYBD2130
	NSLOW2 = 0	HYBD2140
C		HYBD2150
C	BEGINNING OF THE OUTER LOOP.	HYBD2160

C		HYBD2170
	30 CONTINUE	HYBD2180
	JEVAL = .TRUE.	HYBD2190
C		HYBD2200
C	CALCULATE THE JACOBIAN MATRIX.	HYBD2210
C		HYBD2220
	IFLAG = 2	HYBD2230
	CALL FDJAC1(FCN,N,X,FVEC,FJAC,LDFJAC,IFLAG,ML,MU,EPSFCN,WA1,	HYBD2240
	* WA2)	HYBD2250
	NFEV = NFEV + MSUM	HYBD2260
	IF (IFLAG .LT. 0) GO TO 300	HYBD2270
		HYBD2280
C		HYBD2290
C	COMPUTE THE QR FACTORIZATION OF THE JACOBIAN.	HYBD2300
C		HYBD2310
	CALL QRFAC(N,N,FJAC,LDFJAC,.FALSE.,IWA,1,WA1,WA2,WA3)	HYBD2320
C		HYBD2330
C	ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING	HYBD2340
C	TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN.	HYBD2350
C		HYBD2360
	IF (ITER .NE. 1) GO TO 70	HYBD2370
	IF (MODE .EQ. 2) GO TO 50	HYBD2380
	DO 40 J = 1, N	HYBD2390
	DIAG(J) = WA2(J)	HYBD2400
	IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE	HYBD2410
40	CONTINUE	HYBD2420
50	CONTINUE	HYBD2430
		HYBD2440
C	ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X	HYBD2450
C	AND INITIALIZE THE STEP BOUND DELTA.	HYBD2460
C		HYBD2470
	DO 60 J = 1, N	HYBD2480
	WA3(J) = DIAG(J)*X(J)	HYBD2490
60	CONTINUE	HYBD2500
	XNORM = ENORM(N,WA3)	HYBD2510
	DELTA = FACTOR*XNORM	HYBD2520
	IF (DELTA .EQ. ZERO) DELTA = FACTOR	HYBD2530
70	CONTINUE	HYBD2540
		HYBD2550
C	FORM (Q TRANSPOSE)*FVEC AND STORE IN QTF.	HYBD2560
C		HYBD2570
C	DO 80 I = 1, N	HYBD2580
	QTF(I) = FVEC(I)	HYBD2590
80	CONTINUE	HYBD2600
	DO 120 J = 1, N	HYBD2610
	IF (FJAC(J,J) .EQ. ZERO) GO TO 110	HYBD2620
	SUM = ZERO	HYBD2630
	DO 90 I = J, N	HYBD2640
	SUM = SUM + FJAC(I,J)*QTF(I)	HYBD2650
90	CONTINUE	HYBD2660
	TEMP = -SUM/FJAC(J,J)	HYBD2670
	DO 100 I = J, N	HYBD2680
	QTF(I) = QTF(I) + FJAC(I,J)*TEMP	HYBD2690
100	CONTINUE	HYBD2700
110	CONTINUE	

120	CONTINUE	HYBD2710
C		HYBD2720
C	COPY THE TRIANGULAR FACTOR OF THE QR FACTORIZATION INTO R.	HYBD2730
C		HYBD2740
	SING = .FALSE.	HYBD2750
	DO 150 J = 1, N	HYBD2760
	L = J	HYBD2770
	JM1 = J - 1	HYBD2780
	IF (JM1 .LT. 1) GO TO 140	HYBD2790
	DO 130 I = 1, JM1	HYBD2800
	R(L) = FJAC(I,J)	HYBD2810
	L = L + N - I	HYBD2820
130	CONTINUE	HYBD2830
140	CONTINUE	HYBD2840
	R(L) = WA1(J)	HYBD2850
	IF (WA1(J) .EQ. ZERO) SING = .TRUE.	HYBD2860
150	CONTINUE	HYBD2870
C		HYBD2880
C	ACCUMULATE THE ORTHOGONAL FACTOR IN FJAC.	HYBD2890
C		HYBD2900
	CALL QFORM(N,N,FJAC,LDFJAC,WA1)	HYBD2910
C		HYBD2920
C	RESCALE IF NECESSARY.	HYBD2930
C		HYBD2940
	IF (MODE .EQ. 2) GO TO 170	HYBD2950
	DO 160 J = 1, N	HYBD2960
	DIAG(J) = DMAX1(DIAG(J),WA2(J))	HYBD2970
160	CONTINUE	HYBD2980
170	CONTINUE	HYBD2990
C		HYBD3000
C	BEGINNING OF THE INNER LOOP.	HYBD3010
C		HYBD3020
180	CONTINUE	HYBD3030
C		HYBD3040
C	IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.	HYBD3050
C		HYBD3060
	IF (NPRINT .LE. 0) GO TO 190	HYBD3070
	IFLAG = 0	HYBD3080
	IF (MOD(ITER-1,NPRINT) .EQ. 0) CALL FCN(N,X,FVEC,IFLAG)	HYBD3090
	IF (IFLAG .LT. 0) GO TO 300	HYBD3100
190	CONTINUE	HYBD3110
C		HYBD3120
C	DETERMINE THE DIRECTION P.	HYBD3130
C		HYBD3140
	CALL DOGLEG(N,R,LR,DIAG,QTF,DELTA,WA1,WA2,WA3)	HYBD3150
C		HYBD3160
C	STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.	HYBD3170
C		HYBD3180
	DO 200 J = 1, N	HYBD3190
	WA1(J) = -WA1(J)	HYBD3200
	WA2(J) = X(J) + WA1(J)	HYBD3210
	WA3(J) = DIAG(J)*WA1(J)	HYBD3220
200	CONTINUE	HYBD3230
	PNORM = ENORM(N,WA3)	HYBD3240

C		HYBD3250
C	ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.	HYBD3260
C		HYBD3270
	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
	IFLAG = 1	HYBD3320
	CALL FCN(N,WA2,WA4,IFLAG)	HYBD3330
	NFEV = NFEV + 1	HYBD3340
	IF (IFLAG .LT. 0) GO TO 300	HYBD3350
	FNORM1 = ENORM(N,WA4)	HYBD3360
C		HYBD3370
C	COMPUTE THE SCALED ACTUAL REDUCTION.	HYBD3380
C		HYBD3390
	ACTRED = -ONE	HYBD3400
	IF (FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2	HYBD3410
C		HYBD3420
C	COMPUTE THE SCALED PREDICTED REDUCTION.	HYBD3430
C		HYBD3440
	L = 1	HYBD3450
	DO 220 I = 1, N	HYBD3460
	SUM = ZERO	HYBD3470
	DO 210 J = I, N	HYBD3480
	SUM = SUM + R(L)*WA1(J)	HYBD3490
	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
C		HYBD3570
C	COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED	HYBD3580
C	REDUCTION.	HYBD3590
C		HYBD3600
	RATIO = ZERO	HYBD3610
	IF (PRERED .GT. ZERO) RATIO = ACTRED/PRERED	HYBD3620
C		HYBD3630
C	UPDATE THE STEP BOUND.	HYBD3640
C		HYBD3650
	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		HYBD3800
	IF (RATIO .LT. P0001) GO TO 260	HYBD3810
C		HYBD3820
C	SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.	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		HYBD4340
	CALL R1UPDT(N,N,R,LR,WA1,WA2,WA3,SING)	HYBD4350
	CALL R1MPYQ(N,N,FJAC,LDFJAC,WA2,WA3)	HYBD4360
	CALL R1MPYQ(1,N,QTF,1,WA2,WA3)	HYBD4370
C		HYBD4380
C	END OF THE INNER LOOP.	HYBD4390
C		HYBD4400
	JEVAL = .FALSE.	HYBD4410
	GO TO 180	HYBD4420
290	CONTINUE	HYBD4430
C		HYBD4440
C	END OF THE OUTER LOOP.	HYBD4450
C		HYBD4460
	GO TO 30	HYBD4470
300	CONTINUE	HYBD4480
C		HYBD4490
C	TERMINATION, EITHER NORMAL OR USER IMPOSED.	HYBD4500
C		HYBD4510
	IF (IFLAG .LT. 0) INFO = IFLAG	HYBD4520
	IFLAG = 0	HYBD4530
	IF (NPRINT .GT. 0) CALL FCN(N,X,FVEC,IFLAG)	HYBD4540
	RETURN	HYBD4550
C		HYBD4560
C	LAST CARD OF SUBROUTINE HYBRD.	HYBD4570
C		HYBD4580
	END	HYBD4590

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK


```
      WA(J) = ONE
10    CONTINUE
      NPRINT = 0
      LR = (N*(N + 1))/2
      INDEX = 6*N + LR
      CALL HYBRD(FCN,N,X,FVEC,XTOL,MAXFEV,ML,MU,EPSFCN,WA(1),MODE,
*           FACTOR,NPRINT,INFO,NFEV,WA(INDEX+1),N,WA(6*N+1),LR,
*           WA(N+1),WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))
      IF (INFO .EQ. 5) INFO = 4
20    CONTINUE
      RETURN
C
C    LAST CARD OF SUBROUTINE HYBRD1.
C
      END
```

```
HYD11090
HYD11100
HYD11110
HYD11120
HYD11130
HYD11140
HYD11150
HYD11160
HYD11170
HYD11180
HYD11190
HYD11200
HYD11210
HYD11220
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 ***** HYBJ0080
C HYBJ0090
C SUBROUTINE HYBRJ HYBJ0100
C HYBJ0110
C THE PURPOSE OF HYBRJ IS TO FIND A ZERO OF A SYSTEM OF HYBJ0120
C N NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION HYBJ0130
C OF THE POWELL HYBRID METHOD. THE USER MUST PROVIDE A HYBJ0140
C SUBROUTINE WHICH CALCULATES THE FUNCTIONS AND THE JACOBIAN. HYBJ0150
C HYBJ0160
C THE SUBROUTINE STATEMENT IS HYBJ0170
C HYBJ0180
C SUBROUTINE HYBRJ(FCN,N,X,FVEC,FJAC,LDFJAC,XTOL,MAXFEV,DIAG, HYBJ0190
C          MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,R,LR,QTF, HYBJ0200
C          WA1,WA2,WA3,WA4) HYBJ0210
C HYBJ0220
C WHERE HYBJ0230
C HYBJ0240
C FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH HYBJ0250
C CALCULATES THE FUNCTIONS AND THE JACOBIAN. FCN MUST HYBJ0260
C BE DECLARED IN AN EXTERNAL STATEMENT IN THE USER HYBJ0270
C CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS. HYBJ0280
C HYBJ0290
C SUBROUTINE FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG) HYBJ0300
C INTEGER N,LDFJAC,IFLAG HYBJ0310
C DOUBLE PRECISION X(N),FVEC(N),FJAC(LDFJAC,N) HYBJ0320
C ----- HYBJ0330
C IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND HYBJ0340
C RETURN THIS VECTOR IN FVEC. DO NOT ALTER FJAC. HYBJ0350
C IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND HYBJ0360
C RETURN THIS MATRIX IN FJAC. DO NOT ALTER FVEC. HYBJ0370
C ----- HYBJ0380
C RETURN HYBJ0390
C END HYBJ0400
C HYBJ0410
C THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS HYBJ0420
C THE USER WANTS TO TERMINATE EXECUTION OF HYBRJ. HYBJ0430
C IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER. HYBJ0440
C HYBJ0450
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER HYBJ0460
C OF FUNCTIONS AND VARIABLES. HYBJ0470
C HYBJ0480
C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN HYBJ0490
C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X HYBJ0500
C CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR. HYBJ0510
C HYBJ0520
C FVEC IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS HYBJ0530
C THE FUNCTIONS EVALUATED AT THE OUTPUT X. HYBJ0540

```

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

C	EPSMCH = DPMPAR(1)	HYBJ1630
	INFO = 0	HYBJ1640
	IFLAG = 0	HYBJ1650
	NFEV = 0	HYBJ1660
	NJEV = 0	HYBJ1670
C		HYBJ1680
C	CHECK THE INPUT PARAMETERS FOR ERRORS.	HYBJ1690
C		HYBJ1700
	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
C		HYBJ1790
C	EVALUATE THE FUNCTION AT THE STARTING POINT	HYBJ1800
C	AND CALCULATE ITS NORM.	HYBJ1810
C		HYBJ1820
	IFLAG = 1	HYBJ1830
	CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)	HYBJ1840
	NFEV = 1	HYBJ1850
	IF (IFLAG .LT. 0) GO TO 300	HYBJ1860
	FNORM = ENORM(N,FVEC)	HYBJ1870
C		HYBJ1880
C	INITIALIZE ITERATION COUNTER AND MONITORS.	HYBJ1890
C		HYBJ1900
	ITER = 1	HYBJ1910
	NCSUC = 0	HYBJ1920
	NCFAIL = 0	HYBJ1930
	NSLOW1 = 0	HYBJ1940
	NSLOW2 = 0	HYBJ1950
C		HYBJ1960
C	BEGINNING OF THE OUTER LOOP.	HYBJ1970
C		HYBJ1980
	30 CONTINUE	HYBJ1990
	JEVAL = .TRUE.	HYBJ2000
C		HYBJ2010
C	CALCULATE THE JACOBIAN MATRIX.	HYBJ2020
C		HYBJ2030
	IFLAG = 2	HYBJ2040
	CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)	HYBJ2050
	NJEV = NJEV + 1	HYBJ2060
	IF (IFLAG .LT. 0) GO TO 300	HYBJ2070
C		HYBJ2080
C	COMPUTE THE QR FACTORIZATION OF THE JACOBIAN.	HYBJ2090
C		HYBJ2100
	CALL QRFAC(N,N,FJAC,LDFJAC,.FALSE.,IWA,1,WA1,WA2,WA3)	HYBJ2110
C		HYBJ2120
C	ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING	HYBJ2130
C	TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN.	HYBJ2140
C		HYBJ2150
C		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		HYBJ2240
C	ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X	HYBJ2250
C	AND INITIALIZE THE STEP BOUND DELTA.	HYBJ2260
C		HYBJ2270
	DO 60 J = 1, N	HYBJ2280
	WA3(J) = DIAG(J)*X(J)	HYBJ2290
60	CONTINUE	HYBJ2300
	XNORM = ENORM(N,WA3)	HYBJ2310
	DELTA = FACTOR*XNORM	HYBJ2320
	IF (DELTA .EQ. ZERO) DELTA = FACTOR	HYBJ2330
70	CONTINUE	HYBJ2340
C		HYBJ2350
C	FORM (Q TRANSPOSE)*FVEC AND STORE IN QTF.	HYBJ2360
C		HYBJ2370
	DO 80 I = 1, N	HYBJ2380
	QTF(I) = FVEC(I)	HYBJ2390
80	CONTINUE	HYBJ2400
	DO 120 J = 1, N	HYBJ2410
	IF (FJAC(J,J) .EQ. ZERO) GO TO 110	HYBJ2420
	SUM = ZERO	HYBJ2430
	DO 90 I = J, N	HYBJ2440
	SUM = SUM + FJAC(I,J)*QTF(I)	HYBJ2450
90	CONTINUE	HYBJ2460
	TEMP = -SUM/FJAC(J,J)	HYBJ2470
	DO 100 I = J, N	HYBJ2480
	QTF(I) = QTF(I) + FJAC(I,J)*TEMP	HYBJ2490
100	CONTINUE	HYBJ2500
110	CONTINUE	HYBJ2510
120	CONTINUE	HYBJ2520
C		HYBJ2530
C	COPY THE TRIANGULAR FACTOR OF THE QR FACTORIZATION INTO R.	HYBJ2540
C		HYBJ2550
	SING = .FALSE.	HYBJ2560
	DO 150 J = 1, N	HYBJ2570
	L = J	HYBJ2580
	JM1 = J - 1	HYBJ2590
	IF (JM1 .LT. 1) GO TO 140	HYBJ2600
	DO 130 I = 1, JM1	HYBJ2610
	R(L) = FJAC(I,J)	HYBJ2620
	L = L + N - I	HYBJ2630
130	CONTINUE	HYBJ2640
140	CONTINUE	HYBJ2650
	R(L) = WA1(J)	HYBJ2660
	IF (WA1(J) .EQ. ZERO) SING = .TRUE.	HYBJ2670
150	CONTINUE	HYBJ2680
C		HYBJ2690
C	ACCUMULATE THE ORTHOGONAL FACTOR IN FJAC.	HYBJ2700

C	CALL QFORM(N,N,FJAC,LDFJAC,WA1)	HYBJ2710
C		HYBJ2720
C	RESCALE IF NECESSARY.	HYBJ2730
C		HYBJ2740
	IF (MODE .EQ. 2) GO TO 170	HYBJ2750
	DO 160 J = 1, N	HYBJ2760
	DIAG(J) = DMAX1(DIAG(J),WA2(J))	HYBJ2770
160	CONTINUE	HYBJ2780
170	CONTINUE	HYBJ2790
C		HYBJ2800
C	BEGINNING OF THE INNER LOOP.	HYBJ2810
C		HYBJ2820
C	180 CONTINUE	HYBJ2830
C		HYBJ2840
C		HYBJ2850
C	IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.	HYBJ2860
C		HYBJ2870
	IF (NPRINT .LE. 0) GO TO 190	HYBJ2880
	IFLAG = 0	HYBJ2890
	IF (MOD(ITER-1,NPRINT) .EQ. 0)	HYBJ2900
*	CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)	HYBJ2910
	IF (IFLAG .LT. 0) GO TO 300	HYBJ2920
190	CONTINUE	HYBJ2930
C		HYBJ2940
C	DETERMINE THE DIRECTION P.	HYBJ2950
C		HYBJ2960
	CALL DOGLEG(N,R,LR,DIAG,QTF,DELTA,WA1,WA2,WA3)	HYBJ2970
C		HYBJ2980
C	STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.	HYBJ2990
C		HYBJ3000
	DO 200 J = 1, N	HYBJ3010
	WA1(J) = -WA1(J)	HYBJ3020
	WA2(J) = X(J) + WA1(J)	HYBJ3030
	WA3(J) = DIAG(J)*WA1(J)	HYBJ3040
200	CONTINUE	HYBJ3050
	PNORM = ENORM(N,WA3)	HYBJ3060
C		HYBJ3070
C	ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.	HYBJ3080
C		HYBJ3090
	IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)	HYBJ3100
C		HYBJ3110
C	EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.	HYBJ3120
C		HYBJ3130
	IFLAG = 1	HYBJ3140
	CALL FCN(N,WA2,WA4,FJAC,LDFJAC,IFLAG)	HYBJ3150
	NFEV = NFEV + 1	HYBJ3160
	IF (IFLAG .LT. 0) GO TO 300	HYBJ3170
	FNORM1 = ENORM(N,WA4)	HYBJ3180
C		HYBJ3190
C	COMPUTE THE SCALED ACTUAL REDUCTION.	HYBJ3200
C		HYBJ3210
	ACTRED = -ONE	HYBJ3220
	IF (FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2	HYBJ3230
C		HYBJ3240

C	COMPUTE THE SCALED PREDICTED REDUCTION.	HYBJ3250
C		HYBJ3260
	L = 1	HYBJ3270
	DO 220 I = 1, N	HYBJ3280
	SUM = ZERO	HYBJ3290
	DO 210 J = I, N	HYBJ3300
	SUM = SUM + R(L)*WA1(J)	HYBJ3310
	L = L + 1	HYBJ3320
210	CONTINUE	HYBJ3330
	WA3(I) = QTF(I) + SUM	HYBJ3340
220	CONTINUE	HYBJ3350
	TEMP = ENORM(N,WA3)	HYBJ3360
	PRERED = ZERO	HYBJ3370
	IF (TEMP .LT. FNORM) PRERED = ONE - (TEMP/FNORM)**2	HYBJ3380
C		HYBJ3390
C	COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED	HYBJ3400
C	REDUCTION.	HYBJ3410
C		HYBJ3420
	RATIO = ZERO	HYBJ3430
	IF (PRERED .GT. ZERO) RATIO = ACTRED/PRERED	HYBJ3440
C		HYBJ3450
C	UPDATE THE STEP BOUND.	HYBJ3460
C		HYBJ3470
	IF (RATIO .GE. P1) GO TO 230	HYBJ3480
	NCSUC = 0	HYBJ3490
	NCFAIL = NCFail + 1	HYBJ3500
	DELTA = P5*DELTA	HYBJ3510
	GO TO 240	HYBJ3520
230	CONTINUE	HYBJ3530
	NCFail = 0	HYBJ3540
	NCSUC = NCSUC + 1	HYBJ3550
	IF (RATIO .GE. P5 .OR. NCSUC .GT. 1)	HYBJ3560
*	DELTA = DMAX1(DELTA,PNORM/P5)	HYBJ3570
	IF (DABS(RATIO-ONE) .LE. P1) DELTA = PNORM/P5	HYBJ3580
240	CONTINUE	HYBJ3590
C		HYBJ3600
C	TEST FOR SUCCESSFUL ITERATION.	HYBJ3610
C		HYBJ3620
	IF (RATIO .LT. P0001) GO TO 260	HYBJ3630
C		HYBJ3640
C	SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.	HYBJ3650
C		HYBJ3660
	DO 250 J = 1, N	HYBJ3670
	X(J) = WA2(J)	HYBJ3680
	WA2(J) = DIAG(J)*X(J)	HYBJ3690
	FVEC(J) = WA4(J)	HYBJ3700
250	CONTINUE	HYBJ3710
	XNORM = ENORM(N,WA2)	HYBJ3720
	FNORM = FNORM1	HYBJ3730
	ITER = ITER + 1	HYBJ3740
260	CONTINUE	HYBJ3750
C		HYBJ3760
C	DETERMINE THE PROGRESS OF THE ITERATION.	HYBJ3770
C		HYBJ3780

	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	TEST FOR CONVERGENCE.	HYBJ3840
C		HYBJ3850
	IF (DELTA .LE. XTOL*XNORM .OR. FNORM .EQ. ZERO) INFO = 1	HYBJ3860
	IF (INFO .NE. 0) GO TO 300	HYBJ3870
C		HYBJ3880
C	TESTS FOR TERMINATION AND STRINGENT TOLERANCES.	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	CRITERION FOR RECALCULATING JACOBIAN.	HYBJ3970
C		HYBJ3980
	IF (NCFAIL .EQ. 2) GO TO 290	HYBJ3990
C		HYBJ4000
C	CALCULATE THE RANK ONE MODIFICATION TO THE JACOBIAN	HYBJ4010
C	AND UPDATE QTF IF NECESSARY.	HYBJ4020
C		HYBJ4030
	DO 280 J = 1, N	HYBJ4040
	SUM = ZERO	HYBJ4050
	DO 270 I = 1, N	HYBJ4060
	SUM = SUM + FJAC(I,J)*WA4(I)	HYBJ4070
270	CONTINUE	HYBJ4080
	WA2(J) = (SUM - WA3(J))/PNORM	HYBJ4090
	WA1(J) = DIAG(J)*((DIAG(J)*WA1(J))/PNORM)	HYBJ4100
	IF (RATIO .GE. P0001) QTF(J) = SUM	HYBJ4110
280	CONTINUE	HYBJ4120
C		HYBJ4130
C	COMPUTE THE QR FACTORIZATION OF THE UPDATED JACOBIAN.	HYBJ4140
C		HYBJ4150
	CALL RIUPDT(N,N,R,LR,WA1,WA2,WA3,SING)	HYBJ4160
	CALL RIMPYQ(N,N,FJAC,LDFJAC,WA2,WA3)	HYBJ4170
	CALL RIMPYQ(1,N,QTF,1,WA2,WA3)	HYBJ4180
C		HYBJ4190
C	END OF THE INNER LOOP.	HYBJ4200
C		HYBJ4210
	JEVAL = .FALSE.	HYBJ4220
	GO TO 180	HYBJ4230
290	CONTINUE	HYBJ4240
C		HYBJ4250
C	END OF THE OUTER LOOP.	HYBJ4260
C		HYBJ4270
	GO TO 30	HYBJ4280
300	CONTINUE	HYBJ4290
C		HYBJ4300
C	TERMINATION, EITHER NORMAL OR USER IMPOSED.	HYBJ4310
C		HYBJ4320

```
IF (IFLAG .LT. 0) INFO = IFLAG  
IFLAG = 0  
IF (NPRINT .GT. 0) CALL FCN(N,X,FVEC,FJAC,LDFJAC,IFLAG)  
RETURN
```

```
C  
C  
C
```

```
LAST CARD OF SUBROUTINE HYBRJ.
```

```
END
```

```
HYBJ4330  
HYBJ4340  
HYBJ4350  
HYBJ4360  
HYBJ4370  
HYBJ4380  
HYBJ4390  
HYBJ4400
```

THIS PAGE
WAS THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

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

SUBROUTINE HYBRJ1

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.

THE SUBROUTINE STATEMENT IS

SUBROUTINE HYBRJ1(FCN,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,WA,LWA)

WHERE

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

HYJ10010
 HYJ10020
 HYJ10030
 HYJ10040
 HYJ10050
 HYJ10060
 HYJ10070
 HYJ10080
 HYJ10090
 HYJ10100
 HYJ10110
 HYJ10120
 HYJ10130
 HYJ10140
 HYJ10150
 HYJ10160
 HYJ10170
 HYJ10180
 HYJ10190
 HYJ10200
 HYJ10210
 HYJ10220
 HYJ10230
 HYJ10240
 HYJ10250
 HYJ10260
 HYJ10270
 HYJ10280
 HYJ10290
 HYJ10300
 HYJ10310
 HYJ10320
 HYJ10330
 HYJ10340
 HYJ10350
 HYJ10360
 HYJ10370
 HYJ10380
 HYJ10390
 HYJ10400
 HYJ10410
 HYJ10420
 HYJ10430
 HYJ10440
 HYJ10450
 HYJ10460
 HYJ10470
 HYJ10480
 HYJ10490
 HYJ10500
 HYJ10510
 HYJ10520
 HYJ10530
 HYJ10540

C	ORTHOGONAL MATRIX Q PRODUCED BY THE QR FACTORIZATION	HYJ10550
C	OF THE FINAL APPROXIMATE JACOBIAN.	HYJ10560
C		HYJ10570
C	LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N	HYJ10580
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.	HYJ10590
C		HYJ10600
C	TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS	HYJ10610
C	WHEN THE ALGORITHM ESTIMATES THAT THE RELATIVE ERROR	HYJ10620
C	BETWEEN X AND THE SOLUTION IS AT MOST TOL.	HYJ10630
C		HYJ10640
C	INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS	HYJ10650
C	TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)	HYJ10660
C	VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,	HYJ10670
C	INFO IS SET AS FOLLOWS.	HYJ10680
C		HYJ10690
C	INFO = 0 IMPROPER INPUT PARAMETERS.	HYJ10700
C		HYJ10710
C	INFO = 1 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR	HYJ10720
C	BETWEEN X AND THE SOLUTION IS AT MOST TOL.	HYJ10730
C		HYJ10740
C	INFO = 2 NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS	HYJ10750
C	REACHED 100*(N+1).	HYJ10760
C		HYJ10770
C	INFO = 3 TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN	HYJ10780
C	THE APPROXIMATE SOLUTION X IS POSSIBLE.	HYJ10790
C		HYJ10800
C	INFO = 4 ITERATION IS NOT MAKING GOOD PROGRESS.	HYJ10810
C		HYJ10820
C	WA IS A WORK ARRAY OF LENGTH LWA.	HYJ10830
C		HYJ10840
C	LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN	HYJ10850
C	(N*(N+13))/2.	HYJ10860
C		HYJ10870
C	SUBPROGRAMS CALLED	HYJ10880
C		HYJ10890
C	USER-SUPPLIED FCN	HYJ10900
C		HYJ10910
C	MINPACK-SUPPLIED ... HYBRJ	HYJ10920
C		HYJ10930
C	ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.	HYJ10940
C	BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE	HYJ10950
C		HYJ10960
C	*****	HYJ10970
C	INTEGER J,LR,MAXFEV,MODE,NFEV,NJEV,NPRINT	HYJ10980
C	DOUBLE PRECISION FACTOR,ONE,XTOL,ZERO	HYJ10990
C	DATA FACTOR,ONE,ZERO /1.0D2,1.0D0,0.0D0/	HYJ11000
C	INFO = 0	HYJ11010
C		HYJ11020
C	CHECK THE INPUT PARAMETERS FOR ERRORS.	HYJ11030
C		HYJ11040
C	IF (N .LE. 0 .OR. LDFJAC .LT. N .OR. TOL .LT. ZERO	HYJ11050
C	* .OR. LWA .LT. (N*(N + 13))/2) GO TO 20	HYJ11060
C		HYJ11070
C	CALL HYBRJ.	HYJ11080

C	MAXFEV = 100*(N + 1)	HYJ11090
	XTOL = TOL	HYJ11100
	MODE = 2	HYJ11110
	DO 10 J = 1, N	HYJ11120
	WA(J) = ONE	HYJ11130
10	CONTINUE	HYJ11140
	NPRINT = 0	HYJ11150
	LR = (N*(N + 1))/2	HYJ11160
	CALL HYBRJ(FCN,N,X,FVEC,FJAC,LDFJAC,XTOL,MAXFEV,WA(1),MODE,	HYJ11170
	* FACTOR,NPRINT,INFO,NFEV,NJEV,WA(6*N+1),LR,WA(N+1),	HYJ11180
	* WA(2*N+1),WA(3*N+1),WA(4*N+1),WA(5*N+1))	HYJ11190
	IF (INFO .EQ. 5) INFO = 4	HYJ11200
20	CONTINUE	HYJ11210
	RETURN	HYJ11220
C		HYJ11230
C	LAST CARD OF SUBROUTINE HYBRJ1.	HYJ11240
C		HYJ11250
	END	HYJ11260
		HYJ11270

THIS PAGE
WAS THIS PAGE ALLY
WAS INTENTIONALLY
LEFT BLANK

```

SUBROUTINE LMDER(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,      LMDR0010
*           MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,      LMDR0020
*           IPVT,QTF,WA1,WA2,WA3,WA4)                          LMDR0030
INTEGER M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV          LMDR0040
INTEGER IPVT(N)                                                LMDR0050
DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR                        LMDR0060
DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),DIAG(N),QTF(N),  LMDR0070
*           WA1(N),WA2(N),WA3(N),WA4(M)                        LMDR0080
*****                                                         LMDR0090
C
C
C SUBROUTINE LMDER                                             LMDR0100
C
C THE PURPOSE OF LMDER IS TO MINIMIZE THE SUM OF THE SQUARES OF  LMDR0110
C M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF     LMDR0120
C THE LEVENBERG-MARQUARDT ALGORITHM. THE USER MUST PROVIDE A   LMDR0130
C SUBROUTINE WHICH CALCULATES THE FUNCTIONS AND THE JACOBIAN.   LMDR0140
C
C THE SUBROUTINE STATEMENT IS                                  LMDR0150
C
C   SUBROUTINE LMDER(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL, LMDR0160
C           MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,          LMDR0170
C           NJEV,IPVT,QTF,WA1,WA2,WA3,WA4)                     LMDR0180
C
C WHERE                                                         LMDR0190
C
C   FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH      LMDR0200
C   CALCULATES THE FUNCTIONS AND THE JACOBIAN. FCN MUST        LMDR0210
C   BE DECLARED IN AN EXTERNAL STATEMENT IN THE USER          LMDR0220
C   CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.        LMDR0230
C
C   SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)               LMDR0240
C   INTEGER M,N,LDFJAC,IFLAG                                   LMDR0250
C   DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)              LMDR0260
C   -----                                                    LMDR0270
C   IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND              LMDR0280
C   RETURN THIS VECTOR IN FVEC. DO NOT ALTER FJAC.            LMDR0290
C   IF IFLAG = 2 CALCULATE THE JACOBIAN AT X AND              LMDR0300
C   RETURN THIS MATRIX IN FJAC. DO NOT ALTER FVEC.            LMDR0310
C   -----                                                    LMDR0320
C   RETURN                                                       LMDR0330
C   END                                                           LMDR0340
C
C   THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS    LMDR0350
C   THE USER WANTS TO TERMINATE EXECUTION OF LMDER.           LMDR0360
C   IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.             LMDR0370
C
C M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER     LMDR0380
C OF FUNCTIONS.                                                 LMDR0390
C
C N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER     LMDR0400
C OF VARIABLES. N MUST NOT EXCEED M.                           LMDR0410
C
C X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN          LMDR0420
C AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X     LMDR0430

```

C	CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.	LMDR0550
C		LMDR0560
C	FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS	LMDR0570
C	THE FUNCTIONS EVALUATED AT THE OUTPUT X.	LMDR0580
C		LMDR0590
C	FJAC IS AN OUTPUT M BY N ARRAY. THE UPPER N BY N SUBMATRIX	LMDR0600
C	OF FJAC CONTAINS AN UPPER TRIANGULAR MATRIX R WITH	LMDR0610
C	DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE SUCH THAT	LMDR0620
C		LMDR0630
C		LMDR0640
C		LMDR0650
C		LMDR0660
C		LMDR0670
C	WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL	LMDR0680
C	CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J)	LMDR0690
C	(SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRAPEZOIDAL	LMDR0700
C	PART OF FJAC CONTAINS INFORMATION GENERATED DURING	LMDR0710
C	THE COMPUTATION OF R.	LMDR0720
C		LMDR0730
C	LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M	LMDR0740
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.	LMDR0750
C		LMDR0760
C	FTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	LMDR0770
C	OCCURS WHEN BOTH THE ACTUAL AND PREDICTED RELATIVE	LMDR0780
C	REDUCTIONS IN THE SUM OF SQUARES ARE AT MOST FTOL.	LMDR0790
C	THEREFORE, FTOL MEASURES THE RELATIVE ERROR DESIRED	LMDR0800
C	IN THE SUM OF SQUARES.	LMDR0810
C		LMDR0820
C	XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	LMDR0830
C	OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE	LMDR0840
C	ITERATES IS AT MOST XTOL. THEREFORE, XTOL MEASURES THE	LMDR0850
C	RELATIVE ERROR DESIRED IN THE APPROXIMATE SOLUTION.	LMDR0860
C		LMDR0870
C	GTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	LMDR0880
C	OCCURS WHEN THE COSINE OF THE ANGLE BETWEEN FVEC AND	LMDR0890
C	ANY COLUMN OF THE JACOBIAN IS AT MOST GTOL IN ABSOLUTE	LMDR0900
C	VALUE. THEREFORE, GTOL MEASURES THE ORTHOGONALITY	LMDR0910
C	DESIRED BETWEEN THE FUNCTION VECTOR AND THE COLUMNS	LMDR0920
C	OF THE JACOBIAN.	LMDR0930
C		LMDR0940
C	MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION	LMDR0950
C	OCCURS WHEN THE NUMBER OF CALLS TO FCN WITH IFLAG = 1	LMDR0960
C	HAS REACHED MAXFEV.	LMDR0970
C		LMDR0980
C	DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE	LMDR0990
C	BELOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG	LMDR1000
C	MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS	LMDR1010
C	MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.	LMDR1020
C		LMDR1030
C	MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE	LMDR1040
C	VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,	LMDR1050
C	THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER	LMDR1060
C	VALUES OF MODE ARE EQUIVALENT TO MODE = 1.	LMDR1070
C		LMDR1080
C	FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE	

C	INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF	LMDR1090
C	FACTOR AND THE EUCLIDEAN NORM OF DIAG^*X IF NONZERO, OR ELSE	LMDR1100
C	TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE	LMDR1110
C	INTERVAL (.1,100.).100. IS A GENERALLY RECOMMENDED VALUE.	LMDR1120
C		LMDR1130
C	NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED	LMDR1140
C	PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,	LMDR1150
C	FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST	LMDR1160
C	ITERATION AND EVERY NPRINT ITERATIONS THEREAFTER AND	LMDR1170
C	IMMEDIATELY PRIOR TO RETURN, WITH X, FVEC, AND FJAC	LMDR1180
C	AVAILABLE FOR PRINTING. FVEC AND FJAC SHOULD NOT BE	LMDR1190
C	ALTERED. IF NPRINT IS NOT POSITIVE, NO SPECIAL CALLS	LMDR1200
C	OF FCN WITH IFLAG = 0 ARE MADE.	LMDR1210
C		LMDR1220
C	INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS	LMDR1230
C	TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)	LMDR1240
C	VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,	LMDR1250
C	INFO IS SET AS FOLLOWS.	LMDR1260
C		LMDR1270
C	INFO = 0 IMPROPER INPUT PARAMETERS.	LMDR1280
C		LMDR1290
C	INFO = 1 BOTH ACTUAL AND PREDICTED RELATIVE REDUCTIONS	LMDR1300
C	IN THE SUM OF SQUARES ARE AT MOST FTOL.	LMDR1310
C		LMDR1320
C	INFO = 2 RELATIVE ERROR BETWEEN TWO CONSECUTIVE ITERATES	LMDR1330
C	IS AT MOST XTOL.	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	LMDR1380
C	COLUMN OF THE JACOBIAN IS AT MOST GTOL IN	LMDR1390
C	ABSOLUTE VALUE.	LMDR1400
C		LMDR1410
C	INFO = 5 NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS	LMDR1420
C	REACHED MAXFEV.	LMDR1430
C		LMDR1440
C	INFO = 6 FTOL IS TOO SMALL. NO FURTHER REDUCTION IN	LMDR1450
C	THE SUM OF SQUARES IS POSSIBLE.	LMDR1460
C		LMDR1470
C	INFO = 7 XTOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN	LMDR1480
C	THE APPROXIMATE SOLUTION X IS POSSIBLE.	LMDR1490
C		LMDR1500
C	INFO = 8 GTOL IS TOO SMALL. FVEC IS ORTHOGONAL TO THE	LMDR1510
C	COLUMNS OF THE JACOBIAN TO MACHINE PRECISION.	LMDR1520
C		LMDR1530
C	NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF	LMDR1540
C	CALLS TO FCN WITH IFLAG = 1.	LMDR1550
C		LMDR1560
C	NJEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF	LMDR1570
C	CALLS TO FCN WITH IFLAG = 2.	LMDR1580
C		LMDR1590
C	IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT	LMDR1600
C	DEFINES A PERMUTATION MATRIX P SUCH THAT $\text{JAC}^*P = Q^*R$,	LMDR1610
C	WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS	LMDR1620

	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
	PAR = ZERO	LMDR2250
	ITER = 1	LMDR2260
C		LMDR2270
C	BEGINNING OF THE OUTER LOOP.	LMDR2280
C		LMDR2290
	30 CONTINUE	LMDR2300
C		LMDR2310
C	CALCULATE THE JACOBIAN MATRIX.	LMDR2320
C		LMDR2330
	IFLAG = 2	LMDR2340
	CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)	LMDR2350
	NJEV = NJEV + 1	LMDR2360
	IF (IFLAG .LT. 0) GO TO 300	LMDR2370
C		LMDR2380
C	IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.	LMDR2390
C		LMDR2400
	IF (NPRINT .LE. 0) GO TO 40	LMDR2410
	IFLAG = 0	LMDR2420
	IF (MOD(ITER-1,NPRINT) .EQ. 0)	LMDR2430
	* CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)	LMDR2440
	IF (IFLAG .LT. 0) GO TO 300	LMDR2450
	40 CONTINUE	LMDR2460
C		LMDR2470
C	COMPUTE THE QR FACTORIZATION OF THE JACOBIAN.	LMDR2480
C		LMDR2490
	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
	IF (ITER .NE. 1) GO TO 80	LMDR2550
	IF (MODE .EQ. 2) GO TO 60	LMDR2560
	DO 50 J = 1, N	LMDR2570
	DIAG(J) = WA2(J)	LMDR2580
	IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE	LMDR2590
	50 CONTINUE	LMDR2600
	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
	DO 70 J = 1, N	LMDR2660
	WA3(J) = DIAG(J)*X(J)	LMDR2670
	70 CONTINUE	LMDR2680
	XNORM = ENORM(N,WA3)	LMDR2690
	DELTA = FACTOR*XNORM	LMDR2700

	IF (DELTA .EQ. ZERO) DELTA = FACTOR	LMDR2710
80	CONTINUE	LMDR2720
C		LMDR2730
C	FORM (Q TRANSPOSE)*FVEC AND STORE THE FIRST N COMPONENTS IN	LMDR2740
C	QTF.	LMDR2750
C		LMDR2760
	DO 90 I = 1, M	LMDR2770
	WA4(I) = FVEC(I)	LMDR2780
90	CONTINUE	LMDR2790
	DO 130 J = 1, N	LMDR2800
	IF (FJAC(J,J) .EQ. ZERO) GO TO 120	LMDR2810
	SUM = ZERO	LMDR2820
	DO 100 I = J, M	LMDR2830
	SUM = SUM + FJAC(I,J)*WA4(I)	LMDR2840
100	CONTINUE	LMDR2850
	TEMP = -SUM/FJAC(J,J)	LMDR2860
	DO 110 I = J, M	LMDR2870
	WA4(I) = WA4(I) + FJAC(I,J)*TEMP	LMDR2880
110	CONTINUE	LMDR2890
120	CONTINUE	LMDR2900
	FJAC(J,J) = WA1(J)	LMDR2910
	QTF(J) = WA4(J)	LMDR2920
130	CONTINUE	LMDR2930
C		LMDR2940
C	COMPUTE THE NORM OF THE SCALED GRADIENT.	LMDR2950
C		LMDR2960
	GNORM = ZERO	LMDR2970
	IF (FNORM .EQ. ZERO) GO TO 170	LMDR2980
	DO 160 J = 1, N	LMDR2990
	L = IPVT(J)	LMDR3000
	IF (WA2(L) .EQ. ZERO) GO TO 150	LMDR3010
	SUM = ZERO	LMDR3020
	DO 140 I = 1, J	LMDR3030
	SUM = SUM + FJAC(I,J)*(QTF(I)/FNORM)	LMDR3040
140	CONTINUE	LMDR3050
	GNORM = DMAX1(GNORM,DABS(SUM/WA2(L)))	LMDR3060
150	CONTINUE	LMDR3070
160	CONTINUE	LMDR3080
170	CONTINUE	LMDR3090
C		LMDR3100
C	TEST FOR CONVERGENCE OF THE GRADIENT NORM.	LMDR3110
C		LMDR3120
	IF (GNORM .LE. GTOL) INFO = 4	LMDR3130
	IF (INFO .NE. 0) GO TO 300	LMDR3140
C		LMDR3150
C	RESCALE IF NECESSARY.	LMDR3160
C		LMDR3170
	IF (MODE .EQ. 2) GO TO 190	LMDR3180
	DO 180 J = 1, N	LMDR3190
	DIAG(J) = DMAX1(DIAG(J),WA2(J))	LMDR3200
180	CONTINUE	LMDR3210
190	CONTINUE	LMDR3220
C		LMDR3230
C	BEGINNING OF THE INNER LOOP.	LMDR3240

C			LMDR3250
200	CONTINUE		LMDR3260
C			LMDR3270
C	DETERMINE THE LEVENBERG-MARQUARDT PARAMETER.		LMDR3280
C			LMDR3290
	CALL LMPAR(N,FJAC,LDFJAC,IPVT,DIAG,QTF,DELTA,PAR,WA1,WA2,		LMDR3300
*	WA3,WA4)		LMDR3310
C			LMDR3320
C	STORE THE DIRECTION P AND X + P. CALCULATE THE NORM OF P.		LMDR3330
C			LMDR3340
	DO 210 J = 1, N		LMDR3350
	WA1(J) = -WA1(J)		LMDR3360
	WA2(J) = X(J) + WA1(J)		LMDR3370
	WA3(J) = DIAG(J)*WA1(J)		LMDR3380
210	CONTINUE		LMDR3390
	PNORM = ENORM(N,WA3)		LMDR3400
C			LMDR3410
C	ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.		LMDR3420
C			LMDR3430
	IF (ITER .EQ. 1) DELTA = DMIN1(DELTA,PNORM)		LMDR3440
C			LMDR3450
C	EVALUATE THE FUNCTION AT X + P AND CALCULATE ITS NORM.		LMDR3460
C			LMDR3470
	IFLAG = 1		LMDR3480
	CALL FCN(M,N,WA2,WA4,FJAC,LDFJAC,IFLAG)		LMDR3490
	NFEV = NFEV + 1		LMDR3500
	IF (IFLAG .LT. 0) GO TO 300		LMDR3510
	FNORM1 = ENORM(M,WA4)		LMDR3520
C			LMDR3530
C	COMPUTE THE SCALED ACTUAL REDUCTION.		LMDR3540
C			LMDR3550
	ACTRED = -ONE		LMDR3560
	IF (P1*FNORM1 .LT. FNORM) ACTRED = ONE - (FNORM1/FNORM)**2		LMDR3570
C			LMDR3580
C	COMPUTE THE SCALED PREDICTED REDUCTION AND		LMDR3590
C	THE SCALED DIRECTIONAL DERIVATIVE.		LMDR3600
C			LMDR3610
	DO 230 J = 1, N		LMDR3620
	WA3(J) = ZERO		LMDR3630
	L = IPVT(J)		LMDR3640
	TEMP = WA1(L)		LMDR3650
	DO 220 I = 1, J		LMDR3660
	WA3(I) = WA3(I) + FJAC(I,J)*TEMP		LMDR3670
220	CONTINUE		LMDR3680
230	CONTINUE		LMDR3690
	TEMP1 = ENORM(N,WA3)/FNORM		LMDR3700
	TEMP2 = (DSQRT(PAR)*PNORM)/FNORM		LMDR3710
	PRERED = TEMP1**2 + TEMP2**2/P5		LMDR3720
	DIRDER = -(TEMP1**2 + TEMP2**2)		LMDR3730
C			LMDR3740
C	COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED		LMDR3750
C	REDUCTION.		LMDR3760
C			LMDR3770
	RATIO = ZERO		LMDR3780

	IF (PRERED .NE. ZERO) RATIO = ACTRED/PRERED	LMDR3790
C		LMDR3800
C	UPDATE THE STEP BOUND.	LMDR3810
C		LMDR3820
	IF (RATIO .GT. P25) GO TO 240	LMDR3830
	IF (ACTRED .GE. ZERO) TEMP = P5	LMDR3840
	IF (ACTRED .LT. ZERO)	LMDR3850
*	TEMP = P5*DIRDER/(DIRDER + P5*ACTRED)	LMDR3860
	IF (P1*FNORM1 .GE. FNORM .OR. TEMP .LT. P1) TEMP = P1	LMDR3870
	DELTA = TEMP*DMIN1(DELTA,PNORM/P1)	LMDR3880
	PAR = PAR/TEMP	LMDR3890
	GO TO 260	LMDR3900
240	CONTINUE	LMDR3910
	IF (PAR .NE. ZERO .AND. RATIO .LT. P75) GO TO 250	LMDR3920
	DELTA = PNORM/P5	LMDR3930
	PAR = P5*PAR	LMDR3940
250	CONTINUE	LMDR3950
260	CONTINUE	LMDR3960
C		LMDR3970
C	TEST FOR SUCCESSFUL ITERATION.	LMDR3980
C		LMDR3990
	IF (RATIO .LT. P0001) GO TO 290	LMDR4000
C		LMDR4010
C	SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.	LMDR4020
C		LMDR4030
	DO 270 J = 1, N	LMDR4040
	X(J) = WA2(J)	LMDR4050
	WA2(J) = DIAG(J)*X(J)	LMDR4060
270	CONTINUE	LMDR4070
	DO 280 I = 1, M	LMDR4080
	FVEC(I) = WA4(I)	LMDR4090
280	CONTINUE	LMDR4100
	XNORM = ENORM(N,WA2)	LMDR4110
	FNORM = FNORM1	LMDR4120
	ITER = ITER + 1	LMDR4130
290	CONTINUE	LMDR4140
C		LMDR4150
C	TESTS FOR CONVERGENCE.	LMDR4160
C		LMDR4170
	IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL	LMDR4180
*	.AND. P5*RATIO .LE. ONE) INFO = 1	LMDR4190
	IF (DELTA .LE. XTOL*XNORM) INFO = 2	LMDR4200
	IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL	LMDR4210
*	.AND. P5*RATIO .LE. ONE .AND. INFO .EQ. 2) INFO = 3	LMDR4220
	IF (INFO .NE. 0) GO TO 300	LMDR4230
C		LMDR4240
C	TESTS FOR TERMINATION AND STRINGENT TOLERANCES.	LMDR4250
C		LMDR4260
	IF (NFEV .GE. MAXFEV) INFO = 5	LMDR4270
	IF (DABS(ACTRED) .LE. EPSMCH .AND. PRERED .LE. EPSMCH	LMDR4280
*	.AND. P5*RATIO .LE. ONE) INFO = 6	LMDR4290
	IF (DELTA .LE. EPSMCH*XNORM) INFO = 7	LMDR4300
	IF (GNORM .LE. EPSMCH) INFO = 8	LMDR4310
	IF (INFO .NE. 0) GO TO 300	LMDR4320

C		LMDR4330
C	END OF THE INNER LOOP. REPEAT IF ITERATION UNSUCCESSFUL.	LMDR4340
C		LMDR4350
C	IF (RATIO .LT. P0001) GO TO 200	LMDR4360
C		LMDR4370
C	END OF THE OUTER LOOP.	LMDR4380
C		LMDR4390
C	GO TO 30	LMDR4400
C	300 CONTINUE	LMDR4410
C		LMDR4420
C	TERMINATION, EITHER NORMAL OR USER IMPOSED.	LMDR4430
C		LMDR4440
C	IF (IFLAG .LT. 0) INFO = IFLAG	LMDR4450
C	IFLAG = 0	LMDR4460
C	IF (NPRINT .GT. 0) CALL FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG).	LMDR4470
C	RETURN	LMDR4480
C		LMDR4490
C	LAST CARD OF SUBROUTINE LMDER.	LMDR4500
C		LMDR4510
C	END	LMDR4520

THIS PAGE
WAS THIS PAGE ALLY
WAS INTENTIONALLY
LEFT BLANK

C		LMR10550
C	FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS	LMR10560
C	THE FUNCTIONS EVALUATED AT THE OUTPUT X.	LMR10570
C		LMR10580
C	FJAC IS AN OUTPUT M BY N ARRAY. THE UPPER N BY N SUBMATRIX	LMR10590
C	OF FJAC CONTAINS AN UPPER TRIANGULAR MATRIX R WITH	LMR10600
C	DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE SUCH THAT	LMR10610
C		LMR10620
C	$P^T (JAC^T * JAC) * P = R^T * R,$	LMR10630
C		LMR10640
C		LMR10650
C	WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL	LMR10660
C	CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J)	LMR10670
C	(SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRAPEZOIDAL	LMR10680
C	PART OF FJAC CONTAINS INFORMATION GENERATED DURING	LMR10690
C	THE COMPUTATION OF R.	LMR10700
C		LMR10710
C	LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M	LMR10720
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.	LMR10730
C		LMR10740
C	TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS	LMR10750
C	WHEN THE ALGORITHM ESTIMATES EITHER THAT THE RELATIVE	LMR10760
C	ERROR IN THE SUM OF SQUARES IS AT MOST TOL OR THAT	LMR10770
C	THE RELATIVE ERROR BETWEEN X AND THE SOLUTION IS AT	LMR10780
C	MOST TOL.	LMR10790
C		LMR10800
C	INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS	LMR10810
C	TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)	LMR10820
C	VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,	LMR10830
C	INFO IS SET AS FOLLOWS.	LMR10840
C		LMR10850
C	INFO = 0 IMPROPER INPUT PARAMETERS.	LMR10860
C		LMR10870
C	INFO = 1 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR	LMR10880
C	IN THE SUM OF SQUARES IS AT MOST TOL.	LMR10890
C		LMR10900
C	INFO = 2 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR	LMR10910
C	BETWEEN X AND THE SOLUTION IS AT MOST TOL.	LMR10920
C		LMR10930
C	INFO = 3 CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.	LMR10940
C		LMR10950
C	INFO = 4 FVEC IS ORTHOGONAL TO THE COLUMNS OF THE	LMR10960
C	JACOBIAN TO MACHINE PRECISION.	LMR10970
C		LMR10980
C	INFO = 5 NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS	LMR10990
C	REACHED 100*(N+1).	LMR11000
C		LMR11010
C	INFO = 6 TOL IS TOO SMALL. NO FURTHER REDUCTION IN	LMR11020
C	THE SUM OF SQUARES IS POSSIBLE.	LMR11030
C		LMR11040
C	INFO = 7 TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN	LMR11050
C	THE APPROXIMATE SOLUTION X IS POSSIBLE.	LMR11060
C		LMR11070
C	IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT	LMR11080


```

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
*****
SUBROUTINE LMDIF

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.

THE SUBROUTINE STATEMENT IS

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)

WHERE

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

```

```

LMDF0010
LMDF0020
LMDF0030
LMDF0040
LMDF0050
LMDF0060
LMDF0070
LMDF0080
LMDF0090
LMDF0100
LMDF0110
LMDF0120
LMDF0130
LMDF0140
LMDF0150
LMDF0160
LMDF0170
LMDF0180
LMDF0190
LMDF0200
LMDF0210
LMDF0220
LMDF0230
LMDF0240
LMDF0250
LMDF0260
LMDF0270
LMDF0280
LMDF0290
LMDF0300
LMDF0310
LMDF0320
LMDF0330
LMDF0340
LMDF0350
LMDF0360
LMDF0370
LMDF0380
LMDF0390
LMDF0400
LMDF0410
LMDF0420
LMDF0430
LMDF0440
LMDF0450
LMDF0460
LMDF0470
LMDF0480
LMDF0490
LMDF0500
LMDF0510
LMDF0520
LMDF0530
LMDF0540

```

C	CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.	LMDF0550
C		LMDF0560
C	FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS	LMDF0570
C	THE FUNCTIONS EVALUATED AT THE OUTPUT X.	LMDF0580
C		LMDF0590
C	FTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	LMDF0600
C	OCCURS WHEN BOTH THE ACTUAL AND PREDICTED RELATIVE	LMDF0610
C	REDUCTIONS IN THE SUM OF SQUARES ARE AT MOST FTOL.	LMDF0620
C	THEREFORE, FTOL MEASURES THE RELATIVE ERROR DESIRED	LMDF0630
C	IN THE SUM OF SQUARES.	LMDF0640
C		LMDF0650
C	XTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	LMDF0660
C	OCCURS WHEN THE RELATIVE ERROR BETWEEN TWO CONSECUTIVE	LMDF0670
C	ITERATES IS AT MOST XTOL. THEREFORE, XTOL MEASURES THE	LMDF0680
C	RELATIVE ERROR DESIRED IN THE APPROXIMATE SOLUTION.	LMDF0690
C		LMDF0700
C	GTOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION	LMDF0710
C	OCCURS WHEN THE COSINE OF THE ANGLE BETWEEN FVEC AND	LMDF0720
C	ANY COLUMN OF THE JACOBIAN IS AT MOST GTOL IN ABSOLUTE	LMDF0730
C	VALUE. THEREFORE, GTOL MEASURES THE ORTHOGONALITY	LMDF0740
C	DESIRED BETWEEN THE FUNCTION VECTOR AND THE COLUMNS	LMDF0750
C	OF THE JACOBIAN.	LMDF0760
C		LMDF0770
C	MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION	LMDF0780
C	OCCURS WHEN THE NUMBER OF CALLS TO FCN IS AT LEAST	LMDF0790
C	MAXFEV BY THE END OF AN ITERATION.	LMDF0800
C		LMDF0810
C	EPSFCN IS AN INPUT VARIABLE USED IN DETERMINING A SUITABLE	LMDF0820
C	STEP LENGTH FOR THE FORWARD-DIFFERENCE APPROXIMATION. THIS	LMDF0830
C	APPROXIMATION ASSUMES THAT THE RELATIVE ERRORS IN THE	LMDF0840
C	FUNCTIONS ARE OF THE ORDER OF EPSFCN. IF EPSFCN IS LESS	LMDF0850
C	THAN THE MACHINE PRECISION, IT IS ASSUMED THAT THE RELATIVE	LMDF0860
C	ERRORS IN THE FUNCTIONS ARE OF THE ORDER OF THE MACHINE	LMDF0870
C	PRECISION.	LMDF0880
C		LMDF0890
C	DIAG IS AN ARRAY OF LENGTH N. IF MODE = 1 (SEE	LMDF0900
C	BELOW), DIAG IS INTERNALLY SET. IF MODE = 2, DIAG	LMDF0910
C	MUST CONTAIN POSITIVE ENTRIES THAT SERVE AS	LMDF0920
C	MULTIPLICATIVE SCALE FACTORS FOR THE VARIABLES.	LMDF0930
C		LMDF0940
C	MODE IS AN INTEGER INPUT VARIABLE. IF MODE = 1, THE	LMDF0950
C	VARIABLES WILL BE SCALED INTERNALLY. IF MODE = 2,	LMDF0960
C	THE SCALING IS SPECIFIED BY THE INPUT DIAG. OTHER	LMDF0970
C	VALUES OF MODE ARE EQUIVALENT TO MODE = 1.	LMDF0980
C		LMDF0990
C	FACTOR IS A POSITIVE INPUT VARIABLE USED IN DETERMINING THE	LMDF1000
C	INITIAL STEP BOUND. THIS BOUND IS SET TO THE PRODUCT OF	LMDF1010
C	FACTOR AND THE EUCLIDEAN NORM OF DIAG*X IF NONZERO, OR ELSE	LMDF1020
C	TO FACTOR ITSELF. IN MOST CASES FACTOR SHOULD LIE IN THE	LMDF1030
C	INTERVAL (.1,100.). 100. IS A GENERALLY RECOMMENDED VALUE.	LMDF1040
C		LMDF1050
C	NPRINT IS AN INTEGER INPUT VARIABLE THAT ENABLES CONTROLLED	LMDF1060
C	PRINTING OF ITERATES IF IT IS POSITIVE. IN THIS CASE,	LMDF1070
C	FCN IS CALLED WITH IFLAG = 0 AT THE BEGINNING OF THE FIRST	LMDF1080

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		LMDF1200
C	INFO = 1 BOTH ACTUAL AND PREDICTED RELATIVE REDUCTIONS	LMDF1210
C	IN THE SUM OF SQUARES ARE AT MOST FTOL.	LMDF1220
C		LMDF1230
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		LMDF1280
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		LMDF1350
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		LMDF1520
C	$P^T (JAC^T * JAC) * P = R^T * R,$	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	LDJFAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M	LMDF1610
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.	LMDF1620

C	EVALUATE THE FUNCTION AT THE STARTING POINT	LMDF2170
C	AND CALCULATE ITS NORM.	LMDF2180
C		LMDF2190
	IFLAG = 1	LMDF2200
	CALL FCN(M,N,X,FVEC,IFLAG)	LMDF2210
	NFEV = 1	LMDF2220
	IF (IFLAG .LT. 0) GO TO 300	LMDF2230
	FNORM = ENORM(M,FVEC)	LMDF2240
C		LMDF2250
C	INITIALIZE LEVENBERG-MARQUARDT PARAMETER AND ITERATION COUNTER.	LMDF2260
C		LMDF2270
	PAR = ZERO	LMDF2280
	ITER = 1	LMDF2290
C		LMDF2300
C	BEGINNING OF THE OUTER LOOP.	LMDF2310
C		LMDF2320
	30 CONTINUE	LMDF2330
C		LMDF2340
C	CALCULATE THE JACOBIAN MATRIX.	LMDF2350
C		LMDF2360
	IFLAG = 2	LMDF2370
	CALL FDJAC2(FCN,M,N,X,FVEC,FJAC,LDFJAC,IFLAG,EPSFCN,WA4)	LMDF2380
	NFEV = NFEV + N	LMDF2390
	IF (IFLAG .LT. 0) GO TO 300	LMDF2400
C		LMDF2410
C	IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.	LMDF2420
C		LMDF2430
	IF (NPRINT .LE. 0) GO TO 40	LMDF2440
	IFLAG = 0	LMDF2450
	IF (MOD(ITER-1,NPRINT) .EQ. 0) CALL FCN(M,N,X,FVEC,IFLAG)	LMDF2460
	IF (IFLAG .LT. 0) GO TO 300	LMDF2470
	40 CONTINUE	LMDF2480
C		LMDF2490
C	COMPUTE THE QR FACTORIZATION OF THE JACOBIAN.	LMDF2500
C		LMDF2510
	CALL QRFAC(M,N,FJAC,LDFJAC,.TRUE.,IPVT,N,WA1,WA2,WA3)	LMDF2520
C		LMDF2530
C	ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING	LMDF2540
C	TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN.	LMDF2550
C		LMDF2560
	IF (ITER .NE. 1) GO TO 80	LMDF2570
	IF (MODE .EQ. 2) GO TO 60	LMDF2580
	DO 50 J = 1, N	LMDF2590
	DIAG(J) = WA2(J)	LMDF2600
	IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE	LMDF2610
	50 CONTINUE	LMDF2620
	60 CONTINUE	LMDF2630
C		LMDF2640
C	ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X	LMDF2650
C	AND INITIALIZE THE STEP BOUND DELTA.	LMDF2660
C		LMDF2670
	DO 70 J = 1, N	LMDF2680
	WA3(J) = DIAG(J)*X(J)	LMDF2690
	70 CONTINUE	LMDF2700

	XNORM = ENORM(N,WA3)	LMDF2710
	DELTA = FACTOR*XNORM	LMDF2720
	IF (DELTA .EQ. ZERO) DELTA = FACTOR	LMDF2730
80	CONTINUE	LMDF2740
C		LMDF2750
C	FORM (Q TRANSPOSE)*FVEC AND STORE THE FIRST N COMPONENTS IN	LMDF2760
C	QTF.	LMDF2770
C		LMDF2780
	DO 90 I = 1, M	LMDF2790
	WA4(I) = FVEC(I)	LMDF2800
90	CONTINUE	LMDF2810
	DO 130 J = 1, N	LMDF2820
	IF (FJAC(J,J) .EQ. ZERO) GO TO 120	LMDF2830
	SUM = ZERO	LMDF2840
	DO 100 I = J, M	LMDF2850
	SUM = SUM + FJAC(I,J)*WA4(I)	LMDF2860
100	CONTINUE	LMDF2870
	TEMP = -SUM/FJAC(J,J)	LMDF2880
	DO 110 I = J, M	LMDF2890
	WA4(I) = WA4(I) + FJAC(I,J)*TEMP	LMDF2900
110	CONTINUE	LMDF2910
120	CONTINUE	LMDF2920
	FJAC(J,J) = WA1(J)	LMDF2930
	QTF(J) = WA4(J)	LMDF2940
130	CONTINUE	LMDF2950
C		LMDF2960
C	COMPUTE THE NORM OF THE SCALED GRADIENT.	LMDF2970
C		LMDF2980
	GNORM = ZERO	LMDF2990
	IF (FNORM .EQ. ZERO) GO TO 170	LMDF3000
	DO 160 J = 1, N	LMDF3010
	L = IPVT(J)	LMDF3020
	IF (WA2(L) .EQ. ZERO) GO TO 150	LMDF3030
	SUM = ZERO	LMDF3040
	DO 140 I = 1, J	LMDF3050
	SUM = SUM + FJAC(I,J)*(QTF(I)/FNORM)	LMDF3060
140	CONTINUE	LMDF3070
	GNORM = DMAX1(GNORM,DABS(SUM/WA2(L)))	LMDF3080
150	CONTINUE	LMDF3090
160	CONTINUE	LMDF3100
170	CONTINUE	LMDF3110
C		LMDF3120
C	TEST FOR CONVERGENCE OF THE GRADIENT NORM.	LMDF3130
C		LMDF3140
	IF (GNORM .LE. GTOL) INFO = 4	LMDF3150
	IF (INFO .NE. 0) GO TO 300	LMDF3160
C		LMDF3170
C	RESCALE IF NECESSARY.	LMDF3180
C		LMDF3190
	IF (MODE .EQ. 2) GO TO 190	LMDF3200
	DO 180 J = 1, N	LMDF3210
	DIAG(J) = DMAX1(DIAG(J),WA2(J))	LMDF3220
180	CONTINUE	LMDF3230
190	CONTINUE	LMDF3240

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

C		RATIO = ZERO	LMDF3790
		IF (PRERED .NE. ZERO) RATIO = ACTRED/PRERED	LMDF3800
C			LMDF3810
C		UPDATE THE STEP BOUND.	LMDF3820
C			LMDF3830
		IF (RATIO .GT. P25) GO TO 240	LMDF3840
		IF (ACTRED .GE. ZERO) TEMP = P5	LMDF3850
		IF (ACTRED .LT. ZERO)	LMDF3860
*		TEMP = P5*DIRDER/(DIRDER + P5*ACTRED)	LMDF3870
		IF (P1*FNORM1 .GE. FNORM .OR. TEMP .LT. P1) TEMP = P1	LMDF3880
		DELTA = TEMP*DMIN1(DELTA,PNORM/P1)	LMDF3890
		PAR = PAR/TEMP	LMDF3900
		GO TO 260	LMDF3910
240		CONTINUE	LMDF3920
		IF (PAR .NE. ZERO .AND. RATIO .LT. P75) GO TO 250	LMDF3930
		DELTA = PNORM/P5	LMDF3940
		PAR = P5*PAR	LMDF3950
250		CONTINUE	LMDF3960
260		CONTINUE	LMDF3970
C			LMDF3980
C		TEST FOR SUCCESSFUL ITERATION.	LMDF3990
C			LMDF4000
		IF (RATIO .LT. P0001) GO TO 290	LMDF4010
C			LMDF4020
C		SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.	LMDF4030
C			LMDF4040
		DO 270 J = 1, N	LMDF4050
		X(J) = WA2(J)	LMDF4060
		WA2(J) = DIAG(J)*X(J)	LMDF4070
270		CONTINUE	LMDF4080
		DO 280 I = 1, M	LMDF4090
		FVEC(I) = WA4(I)	LMDF4100
280		CONTINUE	LMDF4110
		XNORM = ENORM(N,WA2)	LMDF4120
		FNORM = FNORM1	LMDF4130
		ITER = ITER + 1	LMDF4140
290		CONTINUE	LMDF4150
C			LMDF4160
C		TESTS FOR CONVERGENCE.	LMDF4170
C			LMDF4180
		IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL	LMDF4190
*		.AND. P5*RATIO .LE. ONE) INFO = 1	LMDF4200
		IF (DELTA .LE. XTOL*XNORM) INFO = 2	LMDF4210
		IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL	LMDF4220
*		.AND. P5*RATIO .LE. ONE .AND. INFO .EQ. 2) INFO = 3	LMDF4230
		IF (INFO .NE. 0) GO TO 300	LMDF4240
			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
	IF (RATIO .LT. P0001) GO TO 200	LMDF4380
C		LMDF4390
C	END OF THE OUTER LOOP.	LMDF4400
C		LMDF4410
	GO TO 30	LMDF4420
300	CONTINUE	LMDF4430
C		LMDF4440
C	TERMINATION, EITHER NORMAL OR USER IMPOSED.	LMDF4450
C		LMDF4460
	IF (IFLAG .LT. 0) INFO = IFLAG	LMDF4470
	IFLAG = 0	LMDF4480
	IF (NPRINT .GT. 0) CALL FCN(M,N,X,FVEC,IFLAG)	LMDF4490
	RETURN	LMDF4500
C		LMDF4510
C	LAST CARD OF SUBROUTINE LMDIF.	LMDF4520
C		LMDF4530
	END	LMDF4540

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

SUBROUTINE LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)	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
*****	LMF10070
	LMF10080
SUBROUTINE LMDIF1	LMF10090
	LMF10100
THE PURPOSE OF LMDIF1 IS TO MINIMIZE THE SUM OF THE SQUARES OF	LMF10110
M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF THE	LMF10120
LEVENBERG-MARQUARDT ALGORITHM. THIS IS DONE BY USING THE MORE	LMF10130
GENERAL LEAST-SQUARES SOLVER LMDIF. THE USER MUST PROVIDE A	LMF10140
SUBROUTINE WHICH CALCULATES THE FUNCTIONS. THE JACOBIAN IS	LMF10150
THEN CALCULATED BY A FORWARD-DIFFERENCE APPROXIMATION.	LMF10160
	LMF10170
THE SUBROUTINE STATEMENT IS	LMF10180
	LMF10190
SUBROUTINE LMDIF1(FCN,M,N,X,FVEC,TOL,INFO,IWA,WA,LWA)	LMF10200
	LMF10210
WHERE	LMF10220
	LMF10230
FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH	LMF10240
CALCULATES THE FUNCTIONS. FCN MUST BE DECLARED	LMF10250
IN AN EXTERNAL STATEMENT IN THE USER CALLING	LMF10260
PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.	LMF10270
	LMF10280
SUBROUTINE FCN(M,N,X,FVEC,IFLAG)	LMF10290
INTEGER M,N,IFLAG	LMF10300
DOUBLE PRECISION X(N),FVEC(M)	LMF10310
-----	LMF10320
CALCULATE THE FUNCTIONS AT X AND	LMF10330
RETURN THIS VECTOR IN FVEC.	LMF10340
-----	LMF10350
RETURN	LMF10360
END	LMF10370
	LMF10380
THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS	LMF10390
THE USER WANTS TO TERMINATE EXECUTION OF LMDIF1.	LMF10400
IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.	LMF10410
	LMF10420
M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	LMF10430
OF FUNCTIONS.	LMF10440
	LMF10450
N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	LMF10460
OF VARIABLES. N MUST NOT EXCEED M.	LMF10470
	LMF10480
X IS AN ARRAY OF LENGTH N. ON INPUT X MUST CONTAIN	LMF10490
AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X	LMF10500
CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.	LMF10510
	LMF10520
FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS	LMF10530
THE FUNCTIONS EVALUATED AT THE OUTPUT X.	LMF10540

C		LMF10550
C	TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS	LMF10560
C	WHEN THE ALGORITHM ESTIMATES EITHER THAT THE RELATIVE	LMF10570
C	ERROR IN THE SUM OF SQUARES IS AT MOST TOL OR THAT	LMF10580
C	THE RELATIVE ERROR BETWEEN X AND THE SOLUTION IS AT	LMF10590
C	MOST TOL.	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

```
C          INFO = 0                                LMF11090
C          CHECK THE INPUT PARAMETERS FOR ERRORS.  LMF11100
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          CALL LMDIF.                              LMF11130
C          MAXFEV = 200*(N + 1)                    LMF11140
C          FTOL = TOL                              LMF11150
C          XTOL = TOL                              LMF11160
C          GTOL = ZERO                             LMF11170
C          EPSFCN = ZERO                           LMF11180
C          MODE = 1                                LMF11190
C          NPRINT = 0                              LMF11200
C          MP5N = M + 5*N                          LMF11210
C          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
C          IF (INFO .EQ. 8) INFO = 4               LMF11250
10 CONTINUE                                       LMF11260
C          RETURN                                   LMF11270
C          LAST CARD OF SUBROUTINE LMDIF1.        LMF11280
C          END                                     LMF11290
C          LMF11300
C          LMF11310
C          LMF11320
C          LMF11330
C          LMF11340
C          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)
*****
SUBROUTINE LMPAR

GIVEN AN M BY N MATRIX A, AN N BY N NONSINGULAR DIAGONAL
MATRIX D, AN M-VECTOR B, AND A POSITIVE NUMBER DELTA,
THE PROBLEM IS TO DETERMINE A VALUE FOR THE PARAMETER
PAR SUCH THAT IF X SOLVES THE SYSTEM

      A*X = B ,      SQRT(PAR)*D*X = 0 ,

IN THE LEAST SQUARES SENSE, AND DXNORM IS THE EUCLIDEAN
NORM OF D*X, THEN EITHER PAR IS ZERO AND

      (DXNORM-DELTA) .LE. 0.1*DELTA ,

OR PAR IS POSITIVE AND

      ABS(DXNORM-DELTA) .LE. 0.1*DELTA .

THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM
IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE
QR FACTORIZATION, WITH COLUMN PIVOTING, OF A. THAT IS, IF
A*P = Q*R, WHERE P IS A PERMUTATION MATRIX, Q HAS ORTHOGONAL
COLUMNS, AND R IS AN UPPER TRIANGULAR MATRIX WITH DIAGONAL
ELEMENTS OF NONINCREASING MAGNITUDE, THEN LMPAR EXPECTS
THE FULL UPPER TRIANGLE OF R, THE PERMUTATION MATRIX P,
AND THE FIRST N COMPONENTS OF (Q TRANSPOSE)*B. ON OUTPUT
LMPAR ALSO PROVIDES AN UPPER TRIANGULAR MATRIX S SUCH THAT

      T      T      T
      P *(A *A + PAR*D*D)*P = S *S .

S IS EMPLOYED WITHIN LMPAR AND MAY BE OF SEPARATE INTEREST.

ONLY A FEW ITERATIONS ARE GENERALLY NEEDED FOR CONVERGENCE
OF THE ALGORITHM. IF, HOWEVER, THE LIMIT OF 10 ITERATIONS
IS REACHED, THEN THE OUTPUT PAR WILL CONTAIN THE BEST
VALUE OBTAINED SO FAR.

THE SUBROUTINE STATEMENT IS

      SUBROUTINE LMPAR(N,R,LDR,IPVT,DIAG,QTB,DELTA,PAR,X,SDIAG,
          WA1,WA2)

WHERE

```

```

LMPR0010
LMPR0020
LMPR0030
LMPR0040
LMPR0050
LMPR0060
LMPR0070
LMPR0080
LMPR0090
LMPR0100
LMPR0110
LMPR0120
LMPR0130
LMPR0140
LMPR0150
LMPR0160
LMPR0170
LMPR0180
LMPR0190
LMPR0200
LMPR0210
LMPR0220
LMPR0230
LMPR0240
LMPR0250
LMPR0260
LMPR0270
LMPR0280
LMPR0290
LMPR0300
LMPR0310
LMPR0320
LMPR0330
LMPR0340
LMPR0350
LMPR0360
LMPR0370
LMPR0380
LMPR0390
LMPR0400
LMPR0410
LMPR0420
LMPR0430
LMPR0440
LMPR0450
LMPR0460
LMPR0470
LMPR0480
LMPR0490
LMPR0500
LMPR0510
LMPR0520
LMPR0530
LMPR0540

```

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 $(Q \text{ TRANSPOSE}) * B$.	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 $A * X = B$, $\text{SQRT}(\text{PAR}) * D * X = 0$,	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, DWARF, 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, 0.0D0/	LMPR1060
C		LMPR1070
C	DWARF IS THE SMALLEST POSITIVE MAGNITUDE.	LMPR1080

C		LMPR1090
	DWARF = DPMPAR(2)	LMPR1100
C		LMPR1110
C	COMPUTE AND STORE IN X THE GAUSS-NEWTON DIRECTION. IF THE	LMPR1120
C	JACOBIAN IS RANK-DEFICIENT, OBTAIN A LEAST SQUARES SOLUTION.	LMPR1130
C		LMPR1140
	NSING = N	LMPR1150
	DO 10 J = 1, N	LMPR1160
	WA1(J) = QTB(J)	LMPR1170
	IF (R(J,J) .EQ. ZERO .AND. NSING .EQ. N) NSING = J - 1	LMPR1180
	IF (NSING .LT. N) WA1(J) = ZERO	LMPR1190
10	CONTINUE	LMPR1200
	IF (NSING .LT. 1) GO TO 50	LMPR1210
	DO 40 K = 1, NSING	LMPR1220
	J = NSING - K + 1	LMPR1230
	WA1(J) = WA1(J)/R(J,J)	LMPR1240
	TEMP = WA1(J)	LMPR1250
	JM1 = J - 1	LMPR1260
	IF (JM1 .LT. 1) GO TO 30	LMPR1270
	DO 20 I = 1, JM1	LMPR1280
	WA1(I) = WA1(I) - R(I,J)*TEMP	LMPR1290
20	CONTINUE	LMPR1300
30	CONTINUE	LMPR1310
40	CONTINUE	LMPR1320
50	CONTINUE	LMPR1330
	DO 60 J = 1, N	LMPR1340
	L = IPVT(J)	LMPR1350
	X(L) = WA1(J)	LMPR1360
60	CONTINUE	LMPR1370
		LMPR1380
C		LMPR1390
C	INITIALIZE THE ITERATION COUNTER.	LMPR1400
C	EVALUATE THE FUNCTION AT THE ORIGIN, AND TEST	LMPR1410
C	FOR ACCEPTANCE OF THE GAUSS-NEWTON DIRECTION.	LMPR1420
C		LMPR1430
	ITER = 0	LMPR1440
	DO 70 J = 1, N	LMPR1450
	WA2(J) = DIAG(J)*X(J)	LMPR1460
70	CONTINUE	LMPR1470
	DXNORM = ENORM(N,WA2)	LMPR1480
	FP = DXNORM - DELTA	LMPR1490
	IF (FP .LE. P1*DELTA) GO TO 220	LMPR1500
		LMPR1510
C		LMPR1520
C	IF THE JACOBIAN IS NOT RANK DEFICIENT, THE NEWTON	LMPR1530
C	STEP PROVIDES A LOWER BOUND, PARL, FOR THE ZERO OF	LMPR1540
C	THE FUNCTION. OTHERWISE SET THIS BOUND TO ZERO.	LMPR1550
C		LMPR1560
	PARL = ZERO	LMPR1570
	IF (NSING .LT. N) GO TO 120	LMPR1580
	DO 80 J = 1, N	LMPR1590
	L = IPVT(J)	LMPR1600
	WA1(J) = DIAG(L)*(WA2(L)/DXNORM)	LMPR1610
80	CONTINUE	LMPR1620
	DO 110 J = 1, N	
	SUM = ZERO	

	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(DWARF,POO1*PARU)	LMPR2030
	TEMP = DSQRT(PAR)	LMPR2040
	DO 160 J = 1, N	LMPR2050
	WA1(J) = TEMP*DIAG(J)	LMPR2060
160	CONTINUE	LMPR2070
	CALL QRSOLV(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 PARL	LMPR2170
C	IS ZERO OR THE NUMBER OF ITERATIONS HAS REACHED 10.	LMPR2180
C		LMPR2190
	IF (DABS(FP) .LE. P1*DELTA	LMPR2200
*	.OR. PARL .EQ. ZERO .AND. FP .LE. TEMP	LMPR2210
*	.AND. TEMP .LT. ZERO .OR. ITER .EQ. 10) GO TO 220	LMPR2220
C		LMPR2230
C	COMPUTE THE NEWTON CORRECTION.	LMPR2240
C		LMPR2250
	DO 180 J = 1, N	LMPR2260
	L = IPVT(J)	LMPR2270
	WA1(J) = DIAG(L)*(WA2(L)/DXNORM)	LMPR2280
180	CONTINUE	LMPR2290
	DO 210 J = 1, N	LMPR2300
	WA1(J) = WA1(J)/SDIAG(J)	LMPR2310
	TEMP = WA1(J)	LMPR2320
	JP1 = J + 1	LMPR2330
	IF (N .LT. JP1) GO TO 200	LMPR2340
	DO 190 I = JP1, N	LMPR2350
	WA1(I) = WA1(I) - R(I,J)*TEMP	LMPR2360
190	CONTINUE	LMPR2370
200	CONTINUE	LMPR2380
210	CONTINUE	LMPR2390
	TEMP = ENORM(N,WA1)	LMPR2400
	PARC = ((FP/DELTA)/TEMP)/TEMP	LMPR2410
C		LMPR2420
C	DEPENDING ON THE SIGN OF THE FUNCTION, UPDATE PARL OR PARU.	LMPR2430
C		LMPR2440
	IF (FP .GT. ZERO) PARL = DMAX1(PARL,PAR)	LMPR2450
	IF (FP .LT. ZERO) PARU = DMIN1(PARU,PAR)	LMPR2460
C		LMPR2470
C	COMPUTE AN IMPROVED ESTIMATE FOR PAR.	LMPR2480
C		LMPR2490
	PAR = DMAX1(PARL,PAR+PARC)	LMPR2500
C		LMPR2510
C	END OF AN ITERATION.	LMPR2520
C		LMPR2530
	GO TO 150	LMPR2540
220	CONTINUE	LMPR2550
C		LMPR2560
C	TERMINATION.	LMPR2570
C		LMPR2580
	IF (ITER .EQ. 0) PAR = ZERO	LMPR2590
	RETURN	LMPR2600
C		LMPR2610
C	LAST CARD OF SUBROUTINE LMPAR.	LMPR2620
C		LMPR2630
	END	LMPR2640

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

	SUBROUTINE LMSTR(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,	LMSR0010
*	MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,NJEV,	LMSR0020
*	IPVT,QTF,WA1,WA2,WA3,WA4)	LMSR0030
	INTEGER M,N,LDFJAC,MAXFEV,MODE,NPRINT,INFO,NFEV,NJEV	LMSR0040
	INTEGER IPVT(N)	LMSR0050
	LOGICAL SING	LMSR0060
	DOUBLE PRECISION FTOL,XTOL,GTOL,FACTOR	LMSR0070
	DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N),DIAG(N),QTF(N),	LMSR0080
*	WA1(N),WA2(N),WA3(N),WA4(M)	LMSR0090
	*****	LMSR0100
C		LMSR0110
C		LMSR0120
C	SUBROUTINE LMSTR	LMSR0130
C		LMSR0140
C	THE PURPOSE OF LMSTR IS TO MINIMIZE THE SUM OF THE SQUARES OF	LMSR0150
C	M NONLINEAR FUNCTIONS IN N VARIABLES BY A MODIFICATION OF	LMSR0160
C	THE LEVENBERG-MARQUARDT ALGORITHM WHICH USES MINIMAL STORAGE.	LMSR0170
C	THE USER MUST PROVIDE A SUBROUTINE WHICH CALCULATES THE	LMSR0180
C	FUNCTIONS AND THE ROWS OF THE JACOBIAN.	LMSR0190
C		LMSR0200
C	THE SUBROUTINE STATEMENT IS	LMSR0210
C		LMSR0220
C	SUBROUTINE LMSTR(FCN,M,N,X,FVEC,FJAC,LDFJAC,FTOL,XTOL,GTOL,	LMSR0230
C	MAXFEV,DIAG,MODE,FACTOR,NPRINT,INFO,NFEV,	LMSR0240
C	NJEV,IPVT,QTF,WA1,WA2,WA3,WA4)	LMSR0250
C		LMSR0260
C	WHERE	LMSR0270
C		LMSR0280
C	FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH	LMSR0290
C	CALCULATES THE FUNCTIONS AND THE ROWS OF THE JACOBIAN.	LMSR0300
C	FCN MUST BE DECLARED IN AN EXTERNAL STATEMENT IN THE	LMSR0310
C	USER CALLING PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.	LMSR0320
C		LMSR0330
C	SUBROUTINE FCN(M,N,X,FVEC,FJROW,IFLAG)	LMSR0340
C	INTEGER M,N,IFLAG	LMSR0350
C	DOUBLE PRECISION X(N),FVEC(M),FJROW(N)	LMSR0360
C	-----	LMSR0370
C	IF IFLAG = 1 CALCULATE THE FUNCTIONS AT X AND	LMSR0380
C	RETURN THIS VECTOR IN FVEC.	LMSR0390
C	IF IFLAG = I CALCULATE THE (I-1)-ST ROW OF THE	LMSR0400
C	JACOBIAN AT X AND RETURN THIS VECTOR IN FJROW.	LMSR0410
C	-----	LMSR0420
C	RETURN	LMSR0430
C	END	LMSR0440
C		LMSR0450
C	THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS	LMSR0460
C	THE USER WANTS TO TERMINATE EXECUTION OF LMSTR.	LMSR0470
C	IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.	LMSR0480
C		LMSR0490
C	M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	LMSR0500
C	OF FUNCTIONS.	LMSR0510
C		LMSR0520
C	N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	LMSR0530
C	OF VARIABLES. N MUST NOT EXCEED M.	LMSR0540

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	T T T	LMSR0650
C	$P *(JAC *JAC)*P = R *R,$	LMSR0660
C		LMSR0670
C	WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL	LMSR0680
C	CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPVT(J)	LMSR0690
C	(SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRIANGULAR	LMSR0700
C	PART OF FJAC CONTAINS INFORMATION GENERATED DURING	LMSR0710
C	THE COMPUTATION OF R.	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 $\text{DIAG} * X$ 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 $\text{IFLAG} = 0$ 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 $\text{IFLAG} = 0$ 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 $\text{IFLAG} = 1$ 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 $\text{IFLAG} = 1$.	LMSR1550
C		LMSR1560
C	NJEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF	LMSR1570
C	CALLS TO FCN WITH $\text{IFLAG} = 2$.	LMSR1580
C		LMSR1590
C	IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPVT	LMSR1600
C	DEFINES A PERMUTATION MATRIX P SUCH THAT $\text{JAC} * P = Q * R$,	LMSR1610
C	WHERE JAC IS THE FINAL CALCULATED JACOBIAN, Q IS	LMSR1620

	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
C		LMSR2220
C	INITIALIZE LEVENBERG-MARQUARDT PARAMETER AND ITERATION COUNTER.	LMSR2230
C		LMSR2240
	PAR = ZERO	LMSR2250
	ITER = 1	LMSR2260
C		LMSR2270
C	BEGINNING OF THE OUTER LOOP.	LMSR2280
C		LMSR2290
	30 CONTINUE	LMSR2300
C		LMSR2310
C	IF REQUESTED, CALL FCN TO ENABLE PRINTING OF ITERATES.	LMSR2320
C		LMSR2330
	IF (NPRINT .LE. 0) GO TO 40	LMSR2340
	IFLAG = 0	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
	40 CONTINUE	LMSR2380
C		LMSR2390
C	COMPUTE THE QR FACTORIZATION OF THE JACOBIAN MATRIX	LMSR2400
C	CALCULATED ONE ROW AT A TIME, WHILE SIMULTANEOUSLY	LMSR2410
C	FORMING (Q TRANSPOSE)*FVEC AND STORING THE FIRST	LMSR2420
C	N COMPONENTS IN QTF.	LMSR2430
C		LMSR2440
	DO 60 J = 1, N	LMSR2450
	QTF(J) = ZERO	LMSR2460
	DO 50 I = 1, N	LMSR2470
	FJAC(I,J) = ZERO	LMSR2480
	50 CONTINUE	LMSR2490
	60 CONTINUE	LMSR2500
	IFLAG = 2	LMSR2510
	DO 70 I = 1, M	LMSR2520
	CALL FCN(M,N,X,FVEC,WA3,IFLAG)	LMSR2530
	IF (IFLAG .LT. 0) GO TO 340	LMSR2540
	TEMP = FVEC(I)	LMSR2550
	CALL RWUPDT(N,FJAC,LDFJAC,WA3,QTF,TEMP,WA1,WA2)	LMSR2560
	IFLAG = IFLAG + 1	LMSR2570
	70 CONTINUE	LMSR2580
	NJEV = NJEV + 1	LMSR2590
C		LMSR2600
C	IF THE JACOBIAN IS RANK DEFICIENT, CALL QRFAC TO	LMSR2610
C	REORDER ITS COLUMNS AND UPDATE THE COMPONENTS OF QTF.	LMSR2620
C		LMSR2630
	SING = .FALSE.	LMSR2640
	DO 80 J = 1, N	LMSR2650
	IF (FJAC(J,J) .EQ. ZERO) SING = .TRUE.	LMSR2660
	IPVT(J) = J	LMSR2670
	WA2(J) = ENORM(J,FJAC(1,J))	LMSR2680
	80 CONTINUE	LMSR2690
	IF (.NOT.SING) GO TO 130	LMSR2700

	CALL QRFAC(N,N,FJAC,LDFJAC,.TRUE.,IPVT,N,WA1,WA2,WA3)	LMSR2710
	DO 120 J = 1, N	LMSR2720
	IF (FJAC(J,J) .EQ. ZERO) GO TO 110	LMSR2730
	SUM = ZERO	LMSR2740
	DO 90 I = J, N	LMSR2750
	SUM = SUM + FJAC(I,J)*QTF(I)	LMSR2760
90	CONTINUE	LMSR2770
	TEMP = -SUM/FJAC(J,J)	LMSR2780
	DO 100 I = J, N	LMSR2790
	QTF(I) = QTF(I) + FJAC(I,J)*TEMP	LMSR2800
100	CONTINUE	LMSR2810
110	CONTINUE	LMSR2820
	FJAC(J,J) = WA1(J)	LMSR2830
120	CONTINUE	LMSR2840
130	CONTINUE	LMSR2850
C		LMSR2860
C	ON THE FIRST ITERATION AND IF MODE IS 1, SCALE ACCORDING	LMSR2870
C	TO THE NORMS OF THE COLUMNS OF THE INITIAL JACOBIAN.	LMSR2880
C		LMSR2890
	IF (ITER .NE. 1) GO TO 170	LMSR2900
	IF (MODE .EQ. 2) GO TO 150	LMSR2910
	DO 140 J = 1, N	LMSR2920
	DIAG(J) = WA2(J)	LMSR2930
	IF (WA2(J) .EQ. ZERO) DIAG(J) = ONE	LMSR2940
140	CONTINUE	LMSR2950
150	CONTINUE	LMSR2960
C		LMSR2970
C	ON THE FIRST ITERATION, CALCULATE THE NORM OF THE SCALED X	LMSR2980
C	AND INITIALIZE THE STEP BOUND DELTA.	LMSR2990
C		LMSR3000
	DO 160 J = 1, N	LMSR3010
	WA3(J) = DIAG(J)*X(J)	LMSR3020
160	CONTINUE	LMSR3030
	XNORM = ENORM(N,WA3)	LMSR3040
	DELTA = FACTOR*XNORM	LMSR3050
	IF (DELTA .EQ. ZERO) DELTA = FACTOR	LMSR3060
170	CONTINUE	LMSR3070
C		LMSR3080
C	COMPUTE THE NORM OF THE SCALED GRADIENT.	LMSR3090
C		LMSR3100
	GNORM = ZERO	LMSR3110
	IF (FNORM .EQ. ZERO) GO TO 210	LMSR3120
	DO 200 J = 1, N	LMSR3130
	L = IPVT(J)	LMSR3140
	IF (WA2(L) .EQ. ZERO) GO TO 190	LMSR3150
	SUM = ZERO	LMSR3160
	DO 180 I = 1, J	LMSR3170
	SUM = SUM + FJAC(I,J)*(QTF(I)/FNORM)	LMSR3180
180	CONTINUE	LMSR3190
	GNORM = DMAX1(GNORM,DABS(SUM/WA2(L)))	LMSR3200
190	CONTINUE	LMSR3210
200	CONTINUE	LMSR3220
210	CONTINUE	LMSR3230
C		LMSR3240

C	TEST FOR CONVERGENCE OF THE GRADIENT NORM.	LMSR3250
C		LMSR3260
	IF (GNORM .LE. GTOL) INFO = 4	LMSR3270
	IF (INFO .NE. 0) GO TO 340	LMSR3280
C		LMSR3290
C	RESCALE IF NECESSARY.	LMSR3300
C		LMSR3310
	IF (MODE .EQ. 2) GO TO 230	LMSR3320
	DO 220 J = 1, N	LMSR3330
	DIAG(J) = DMAX1(DIAG(J),WA2(J))	LMSR3340
220	CONTINUE	LMSR3350
230	CONTINUE	LMSR3360
C		LMSR3370
C	BEGINNING OF THE INNER LOOP.	LMSR3380
C		LMSR3390
240	CONTINUE	LMSR3400
C		LMSR3410
C	DETERMINE THE LEVENBERG-MARQUARDT PARAMETER.	LMSR3420
C		LMSR3430
	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
	DO 250 J = 1, N	LMSR3490
	WA1(J) = -WA1(J)	LMSR3500
	WA2(J) = X(J) + WA1(J)	LMSR3510
	WA3(J) = DIAG(J)*WA1(J)	LMSR3520
250	CONTINUE	LMSR3530
	PNORM = ENORM(N,WA3)	LMSR3540
C		LMSR3550
C	ON THE FIRST ITERATION, ADJUST THE INITIAL STEP BOUND.	LMSR3560
C		LMSR3570
	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
	IFLAG = 1	LMSR3620
	CALL FCN(M,N,WA2,WA4,WA3,IFLAG)	LMSR3630
	NFEV = NFEV + 1	LMSR3640
	IF (IFLAG .LT. 0) GO TO 340	LMSR3650
	FNORM1 = ENORM(M,WA4)	LMSR3660
C		LMSR3670
C	COMPUTE THE SCALED ACTUAL REDUCTION.	LMSR3680
C		LMSR3690
	ACTRED = -ONE	LMSR3700
	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
	DO 270 J = 1, N	LMSR3760
	WA3(J) = ZERO	LMSR3770
	L = IPVT(J)	LMSR3780

	TEMP = WA1(L)	LMSR3790
	DO 260 I = 1, J	LMSR3800
	WA3(I) = WA3(I) + FJAC(I, J)*TEMP	LMSR3810
260	CONTINUE	LMSR3820
270	CONTINUE	LMSR3830
	TEMP1 = ENORM(N, WA3)/FNORM	LMSR3840
	TEMP2 = (DSQRT(PAR)*PNORM)/FNORM	LMSR3850
	PRERED = TEMP1**2 + TEMP2**2/P5	LMSR3860
	DIRDER = -(TEMP1**2 + TEMP2**2)	LMSR3870
C		LMSR3880
C	COMPUTE THE RATIO OF THE ACTUAL TO THE PREDICTED	LMSR3890
C	REDUCTION.	LMSR3900
C		LMSR3910
	RATIO = ZERO	LMSR3920
	IF (PRERED .NE. ZERO) RATIO = ACTRED/PRERED	LMSR3930
C		LMSR3940
C	UPDATE THE STEP BOUND.	LMSR3950
C		LMSR3960
	IF (RATIO .GT. P25) GO TO 280	LMSR3970
	IF (ACTRED .GE. ZERO) TEMP = P5	LMSR3980
	IF (ACTRED .LT. ZERO)	LMSR3990
*	TEMP = P5*DIRDER/(DIRDER + P5*ACTRED)	LMSR4000
	IF (P1*FNORM1 .GE. FNORM .OR. TEMP .LT. P1) TEMP = P1	LMSR4010
	DELTA = TEMP*DMIN1(DELTA, PNORM/P1)	LMSR4020
	PAR = PAR/TEMP	LMSR4030
	GO TO 300	LMSR4040
280	CONTINUE	LMSR4050
	IF (PAR .NE. ZERO .AND. RATIO .LT. P75) GO TO 290	LMSR4060
	DELTA = PNORM/P5	LMSR4070
	PAR = P5*PAR	LMSR4080
290	CONTINUE	LMSR4090
300	CONTINUE	LMSR4100
C		LMSR4110
C	TEST FOR SUCCESSFUL ITERATION.	LMSR4120
C		LMSR4130
	IF (RATIO .LT. P0001) GO TO 330	LMSR4140
C		LMSR4150
C	SUCCESSFUL ITERATION. UPDATE X, FVEC, AND THEIR NORMS.	LMSR4160
C		LMSR4170
	DO 310 J = 1, N	LMSR4180
	X(J) = WA2(J)	LMSR4190
	WA2(J) = DIAG(J)*X(J)	LMSR4200
310	CONTINUE	LMSR4210
	DO 320 I = 1, M	LMSR4220
	FVEC(I) = WA4(I)	LMSR4230
320	CONTINUE	LMSR4240
	XNORM = ENORM(N, WA2)	LMSR4250
	FNORM = FNORM1	LMSR4260
	ITER = ITER + 1	LMSR4270
330	CONTINUE	LMSR4280
C		LMSR4290
C	TESTS FOR CONVERGENCE.	LMSR4300
C		LMSR4310
	IF (DABS(ACTRED) .LE. FTOL .AND. PRERED .LE. FTOL	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	LMSR4350
	* .AND. P5*RATIO .LE. ONE .AND. INFO .EQ. 2) INFO = 3	LMSR4360
	IF (INFO .NE. 0) GO TO 340	LMSR4370
C		LMSR4380
C	TESTS FOR TERMINATION AND STRINGENT TOLERANCES.	LMSR4390
C		LMSR4400
	IF (NFEV .GE. MAXFEV) INFO = 5	LMSR4410
	IF (DABS(ACTRED) .LE. EPSMCH .AND. PRERED .LE. EPSMCH	LMSR4420
	* .AND. P5*RATIO .LE. ONE) INFO = 6	LMSR4430
	IF (DELTA .LE. EPSMCH*XNORM) INFO = 7	LMSR4440
	IF (GNORM .LE. EPSMCH) INFO = 8	LMSR4450
	IF (INFO .NE. 0) GO TO 340	LMSR4460
C		LMSR4470
C	END OF THE INNER LOOP. REPEAT IF ITERATION UNSUCCESSFUL.	LMSR4480
C		LMSR4490
	IF (RATIO .LT. P0001) GO TO 240	LMSR4500
C		LMSR4510
C	END OF THE OUTER LOOP.	LMSR4520
C		LMSR4530
	GO TO 30	LMSR4540
	340 CONTINUE	LMSR4550
C		LMSR4560
C	TERMINATION, EITHER NORMAL OR USER IMPOSED.	LMSR4570
C		LMSR4580
	IF (IFLAG .LT. 0) INFO = IFLAG	LMSR4590
	IFLAG = 0	LMSR4600
	IF (NPRINT .GT. 0) CALL FCN(M,N,X,FVEC,WA3,IFLAG)	LMSR4610
	RETURN	LMSR4620
C		LMSR4630
C	LAST CARD OF SUBROUTINE LMSTR.	LMSR4640
C		LMSR4650
	END	LMSR4660

THIS PAGE
WAS THIS PAGE IALLY
WAS INTENTIONALLY
LEFT BLANK

C	CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.	LMS10550
C		LMS10560
C	FVEC IS AN OUTPUT ARRAY OF LENGTH M WHICH CONTAINS	LMS10570
C	THE FUNCTIONS EVALUATED AT THE OUTPUT X.	LMS10580
C		LMS10590
C	FJAC IS AN OUTPUT N BY N ARRAY. THE UPPER TRIANGLE OF FJAC	LMS10600
C	CONTAINS AN UPPER TRIANGULAR MATRIX R SUCH THAT	LMS10610
C		LMS10620
C	$P^T * (JAC * JAC) * P = R * R,$	LMS10630
C		LMS10640
C		LMS10650
C	WHERE P IS A PERMUTATION MATRIX AND JAC IS THE FINAL	LMS10660
C	CALCULATED JACOBIAN. COLUMN J OF P IS COLUMN IPV T(J)	LMS10670
C	(SEE BELOW) OF THE IDENTITY MATRIX. THE LOWER TRIANGULAR	LMS10680
C	PART OF FJAC CONTAINS INFORMATION GENERATED DURING	LMS10690
C	THE COMPUTATION OF R.	LMS10700
C		LMS10710
C	LDFJAC IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N	LMS10720
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.	LMS10730
C		LMS10740
C	TOL IS A NONNEGATIVE INPUT VARIABLE. TERMINATION OCCURS	LMS10750
C	WHEN THE ALGORITHM ESTIMATES EITHER THAT THE RELATIVE	LMS10760
C	ERROR IN THE SUM OF SQUARES IS AT MOST TOL OR THAT	LMS10770
C	THE RELATIVE ERROR BETWEEN X AND THE SOLUTION IS AT	LMS10780
C	MOST TOL.	LMS10790
C		LMS10800
C	INFO IS AN INTEGER OUTPUT VARIABLE. IF THE USER HAS	LMS10810
C	TERMINATED EXECUTION, INFO IS SET TO THE (NEGATIVE)	LMS10820
C	VALUE OF IFLAG. SEE DESCRIPTION OF FCN. OTHERWISE,	LMS10830
C	INFO IS SET AS FOLLOWS.	LMS10840
C		LMS10850
C	INFO = 0 IMPROPER INPUT PARAMETERS.	LMS10860
C		LMS10870
C	INFO = 1 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR	LMS10880
C	IN THE SUM OF SQUARES IS AT MOST TOL.	LMS10890
C		LMS10900
C	INFO = 2 ALGORITHM ESTIMATES THAT THE RELATIVE ERROR	LMS10910
C	BETWEEN X AND THE SOLUTION IS AT MOST TOL.	LMS10920
C		LMS10930
C	INFO = 3 CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.	LMS10940
C		LMS10950
C	INFO = 4 FVEC IS ORTHOGONAL TO THE COLUMNS OF THE	LMS10960
C	JACOBIAN TO MACHINE PRECISION.	LMS10970
C		LMS10980
C	INFO = 5 NUMBER OF CALLS TO FCN WITH IFLAG = 1 HAS	LMS10990
C	REACHED 100*(N+1).	LMS11000
C		LMS11010
C	INFO = 6 TOL IS TOO SMALL. NO FURTHER REDUCTION IN	LMS11020
C	THE SUM OF SQUARES IS POSSIBLE.	LMS11030
C		LMS11040
C	INFO = 7 TOL IS TOO SMALL. NO FURTHER IMPROVEMENT IN	LMS11050
C	THE APPROXIMATE SOLUTION X IS POSSIBLE.	LMS11060
C		LMS11070
C	IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH N. IPV T	LMS11080

THIS PAGE
WAS THIS PAGE
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		QFRM0050
C		QFRM0060
C	SUBROUTINE QFORM	QFRM0070
C		QFRM0080
C	THIS SUBROUTINE PROCEEDS FROM THE COMPUTED QR FACTORIZATION OF	QFRM0090
C	AN M BY N MATRIX A TO ACCUMULATE THE M BY M ORTHOGONAL MATRIX	QFRM0100
C	Q FROM ITS FACTORED FORM.	QFRM0110
C		QFRM0120
C	THE SUBROUTINE STATEMENT IS	QFRM0130
C		QFRM0140
C	SUBROUTINE QFORM(M,N,Q,LDQ,WA)	QFRM0150
C		QFRM0160
C	WHERE	QFRM0170
C		QFRM0180
C	M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	QFRM0190
C	OF ROWS OF A AND THE ORDER OF Q.	QFRM0200
C		QFRM0210
C	N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER	QFRM0220
C	OF COLUMNS OF A.	QFRM0230
C		QFRM0240
C	Q IS AN M BY M ARRAY. ON INPUT THE FULL LOWER TRAPEZOID IN	QFRM0250
C	THE FIRST MIN(M,N) COLUMNS OF Q CONTAINS THE FACTORED FORM.	QFRM0260
C	ON OUTPUT Q HAS BEEN ACCUMULATED INTO A SQUARE MATRIX.	QFRM0270
C		QFRM0280
C	LDQ IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M	QFRM0290
C	WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY Q.	QFRM0300
C		QFRM0310
C	WA IS A WORK ARRAY OF LENGTH M.	QFRM0320
C		QFRM0330
C	SUBPROGRAMS CALLED	QFRM0340
C		QFRM0350
C	FORTRAN-SUPPLIED ... MINO	QFRM0360
C		QFRM0370
C	ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.	QFRM0380
C	BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE	QFRM0390
C		QFRM0400
C	*****	QFRM0410
C	INTEGER I,J,JM1,K,L,MINMN,NP1	QFRM0420
C	DOUBLE PRECISION ONE,SUM,TEMP,ZERO	QFRM0430
C	DATA ONE,ZERO /1.0D0,0.0D0/	QFRM0440
C		QFRM0450
C	ZERO OUT UPPER TRIANGLE OF Q IN THE FIRST MIN(M,N) COLUMNS.	QFRM0460
C		QFRM0470
C	MINMN = MINO(M,N)	QFRM0480
C	IF (MINMN .LT. 2) GO TO 30	QFRM0490
C	DO 20 J = 2, MINMN	QFRM0500
C	JM1 = J - 1	QFRM0510
C	DO 10 I = 1, JM1	QFRM0520
C	Q(I,J) = ZERO	QFRM0530
C	CONTINUE	QFRM0540
10	CONTINUE	
20	CONTINUE	

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)
 INTEGER M,N,LDA,LIPVT
 INTEGER IPVT(LIPVT)
 LOGICAL PIVOT
 DOUBLE PRECISION A(LDA,N),RDIAG(N),ACNORM(N),WA(N)

SUBROUTINE QRFAC

THIS SUBROUTINE USES HOUSEHOLDER TRANSFORMATIONS WITH COLUMN
 PIVOTING (OPTIONAL) TO COMPUTE A QR FACTORIZATION OF THE
 M BY N MATRIX A. THAT IS, QRFAC DETERMINES AN ORTHOGONAL
 MATRIX Q, A PERMUTATION MATRIX P, AND AN UPPER TRAPEZOIDAL
 MATRIX R WITH DIAGONAL ELEMENTS OF NONINCREASING MAGNITUDE,
 SUCH THAT $A^*P = Q^*R$. THE HOUSEHOLDER TRANSFORMATION FOR
 COLUMN K, $K = 1, 2, \dots, \text{MIN}(M, N)$, IS OF THE FORM

$$I - (1/U(K))^*U^*U$$

WHERE U HAS ZEROS IN THE FIRST K-1 POSITIONS. THE FORM OF
 THIS TRANSFORMATION AND THE METHOD OF PIVOTING FIRST
 APPEARED IN THE CORRESPONDING LINPACK SUBROUTINE.

THE SUBROUTINE STATEMENT IS

SUBROUTINE QRFAC(M,N,A,LDA,PIVOT,IPVT,LIPVT,RDIAG,ACNORM,WA)

WHERE

M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
 OF ROWS OF A.

N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
 OF COLUMNS OF A.

A IS AN M BY N ARRAY. ON INPUT A CONTAINS THE MATRIX FOR
 WHICH THE QR FACTORIZATION IS TO BE COMPUTED. ON OUTPUT
 THE STRICT UPPER TRAPEZOIDAL PART OF A CONTAINS THE STRICT
 UPPER TRAPEZOIDAL PART OF R, AND THE LOWER TRAPEZOIDAL
 PART OF A CONTAINS A FACTORED FORM OF Q (THE NON-TRIVIAL
 ELEMENTS OF THE U VECTORS DESCRIBED ABOVE).

LDA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M
 WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY A.

PIVOT IS A LOGICAL INPUT VARIABLE. IF PIVOT IS SET TRUE,
 THEN COLUMN PIVOTING IS ENFORCED. IF PIVOT IS SET FALSE,
 THEN NO COLUMN PIVOTING IS DONE.

IPVT IS AN INTEGER OUTPUT ARRAY OF LENGTH LIPVT. IPVT
 DEFINES THE PERMUTATION MATRIX P SUCH THAT $A^*P = Q^*R$.
 COLUMN J OF P IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.
 IF PIVOT IS FALSE, IPVT IS NOT REFERENCED.

QRFA0010
 QRFA0020
 QRFA0030
 QRFA0040
 QRFA0050
 QRFA0060
 QRFA0070
 QRFA0080
 QRFA0090
 QRFA0100
 QRFA0110
 QRFA0120
 QRFA0130
 QRFA0140
 QRFA0150
 QRFA0160
 QRFA0170
 QRFA0180
 QRFA0190
 QRFA0200
 QRFA0210
 QRFA0220
 QRFA0230
 QRFA0240
 QRFA0250
 QRFA0260
 QRFA0270
 QRFA0280
 QRFA0290
 QRFA0300
 QRFA0310
 QRFA0320
 QRFA0330
 QRFA0340
 QRFA0350
 QRFA0360
 QRFA0370
 QRFA0380
 QRFA0390
 QRFA0400
 QRFA0410
 QRFA0420
 QRFA0430
 QRFA0440
 QRFA0450
 QRFA0460
 QRFA0470
 QRFA0480
 QRFA0490
 QRFA0500
 QRFA0510
 QRFA0520
 QRFA0530
 QRFA0540

C		QRFA0550
C	LIPVT IS A POSITIVE INTEGER INPUT VARIABLE. IF PIVOT IS FALSE,	QRFA0560
C	THEN LIPVT MAY BE AS SMALL AS 1. IF PIVOT IS TRUE, THEN	QRFA0570
C	LIPVT MUST BE AT LEAST N.	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	FORTTRAN-SUPPLIED ... DMAX1,DSQRT,MINO	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	*****	QRFA0800
C	INTEGER I,J,JP1,K,KMAX,MINMN	QRFA0810
C	DOUBLE PRECISION AJNORM,EPSMCH,ONE,PO5,SUM,TEMP,ZERO	QRFA0820
C	DOUBLE PRECISION DPMPAR,ENORM	QRFA0830
C	DATA ONE,PO5,ZERO /1.0D0,5.0D-2,0.0D0/	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 = MINO(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)	QRFA1130
	A(I,J) = A(I,KMAX)	QRFA1140
	A(I,KMAX) = TEMP	QRFA1150
30	CONTINUE	QRFA1160
	RDIAG(KMAX) = RDIAG(J)	QRFA1170
	WA(KMAX) = WA(J)	QRFA1180
	K = IPVT(J)	QRFA1190
	IPVT(J) = IPVT(KMAX)	QRFA1200
	IPVT(KMAX) = K	QRFA1210
40	CONTINUE	QRFA1220
C		QRFA1230
C	COMPUTE THE HOUSEHOLDER TRANSFORMATION TO REDUCE THE	QRFA1240
C	J-TH COLUMN OF A TO A MULTIPLE OF THE J-TH UNIT VECTOR.	QRFA1250
C		QRFA1260
	AJNORM = ENORM(M-J+1,A(J,J))	QRFA1270
	IF (AJNORM .EQ. ZERO) GO TO 100	QRFA1280
	IF (A(J,J) .LT. ZERO) AJNORM = -AJNORM	QRFA1290
	DO 50 I = J, M	QRFA1300
	A(I,J) = A(I,J)/AJNORM	QRFA1310
50	CONTINUE	QRFA1320
	A(J,J) = A(J,J) + ONE	QRFA1330
C		QRFA1340
C	APPLY THE TRANSFORMATION TO THE REMAINING COLUMNS	QRFA1350
C	AND UPDATE THE NORMS.	QRFA1360
C		QRFA1370
	JP1 = J + 1	QRFA1380
	IF (N .LT. JP1) GO TO 100	QRFA1390
	DO 90 K = JP1, N	QRFA1400
	SUM = ZERO	QRFA1410
	DO 60 I = J, M	QRFA1420
	SUM = SUM + A(I,J)*A(I,K)	QRFA1430
60	CONTINUE	QRFA1440
	TEMP = SUM/A(J,J)	QRFA1450
	DO 70 I = J, M	QRFA1460
	A(I,K) = A(I,K) - TEMP*A(I,J)	QRFA1470
70	CONTINUE	QRFA1480
	IF (.NOT.PIVOT .OR. RDIAG(K) .EQ. ZERO) GO TO 80	QRFA1490
	TEMP = A(J,K)/RDIAG(K)	QRFA1500
	RDIAG(K) = RDIAG(K)*DSQRT(DMAX1(ZERO,ONE-TEMP**2))	QRFA1510
	IF (PO5*(RDIAG(K)/WA(K))**2 .GT. EPSMCH) GO TO 80	QRFA1520
	RDIAG(K) = ENORM(M-J,A(JP1,K))	QRFA1530
	WA(K) = RDIAG(K)	QRFA1540
80	CONTINUE	QRFA1550
90	CONTINUE	QRFA1560
100	CONTINUE	QRFA1570
	RDIAG(J) = -AJNORM	QRFA1580
110	CONTINUE	QRFA1590
	RETURN	QRFA1600
C		QRFA1610
C	LAST CARD OF SUBROUTINE QRFAC.	QRFA1620

```

SUBROUTINE QRSOLV(N,R,LDR,IPVT,DIAG,QTB,X,SDIAG,WA)
INTEGER N,LDR
INTEGER IPVT(N)
DOUBLE PRECISION R(LDR,N),DIAG(N),QTB(N),X(N),SDIAG(N),WA(N)
*****
C
C
C
SUBROUTINE QRSOLV
C
C
C
GIVEN AN M BY N MATRIX A, AN N BY N DIAGONAL MATRIX D,
C
C
AND AN M-VECTOR B, THE PROBLEM IS TO DETERMINE AN X WHICH
C
C
SOLVES THE SYSTEM
C
C

$$A * X = B , \quad D * X = 0 ,$$

C
C
IN THE LEAST SQUARES SENSE.
C
C
THIS SUBROUTINE COMPLETES THE SOLUTION OF THE PROBLEM
C
C
IF IT IS PROVIDED WITH THE NECESSARY INFORMATION FROM THE
C
C
QR FACTORIZATION, WITH COLUMN PIVOTING, OF A. THAT IS, IF
C
C
 $A * P = Q * R$ , WHERE P IS A PERMUTATION MATRIX, Q HAS ORTHOGONAL
C
C
COLUMNS, AND R IS AN UPPER TRIANGULAR MATRIX WITH DIAGONAL
C
C
ELEMENTS OF NONINCREASING MAGNITUDE, THEN QRSOLV EXPECTS
C
C
THE FULL UPPER TRIANGLE OF R, THE PERMUTATION MATRIX P,
C
C
AND THE FIRST N COMPONENTS OF (Q TRANSPOSE)*B. THE SYSTEM
C
C
 $A * X = B, D * X = 0$ , IS THEN EQUIVALENT TO
C
C

$$R^T * Z = Q^T * B , \quad P^T * D * P * Z = 0 ,$$

C
C
WHERE  $X = P * Z$ . IF THIS SYSTEM DOES NOT HAVE FULL RANK,
C
C
THEN A LEAST SQUARES SOLUTION IS OBTAINED. ON OUTPUT QRSOLV
C
C
ALSO PROVIDES AN UPPER TRIANGULAR MATRIX S SUCH THAT
C
C

$$P^T * (A * A + D * D) * P = S * S .$$

C
C
S IS COMPUTED WITHIN QRSOLV AND MAY BE OF SEPARATE INTEREST.
C
C
THE SUBROUTINE STATEMENT IS
C
C
SUBROUTINE QRSOLV(N,R,LDR,IPVT,DIAG,QTB,X,SDIAG,WA)
C
C
WHERE
C
C
N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE ORDER OF R.
C
C
R IS AN N BY N ARRAY. ON INPUT THE FULL UPPER TRIANGLE
C
C
MUST CONTAIN THE FULL UPPER TRIANGLE OF THE MATRIX R.
C
C
ON OUTPUT THE FULL UPPER TRIANGLE IS UNALTERED, AND THE
C
C
STRICT LOWER TRIANGLE CONTAINS THE STRICT UPPER TRIANGLE
C
C
(TRANPOSED) OF THE UPPER TRIANGULAR MATRIX S.
C
C
LDR IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N
C
C
WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY R.

```

QRSL0010
QRSL0020
QRSL0030
QRSL0040
QRSL0050
QRSL0060
QRSL0070
QRSL0080
QRSL0090
QRSL0100
QRSL0110
QRSL0120
QRSL0130
QRSL0140
QRSL0150
QRSL0160
QRSL0170
QRSL0180
QRSL0190
QRSL0200
QRSL0210
QRSL0220
QRSL0230
QRSL0240
QRSL0250
QRSL0260
QRSL0270
QRSL0280
QRSL0290
QRSL0300
QRSL0310
QRSL0320
QRSL0330
QRSL0340
QRSL0350
QRSL0360
QRSL0370
QRSL0380
QRSL0390
QRSL0400
QRSL0410
QRSL0420
QRSL0430
QRSL0440
QRSL0450
QRSL0460
QRSL0470
QRSL0480
QRSL0490
QRSL0500
QRSL0510
QRSL0520
QRSL0530
QRSL0540

C		QRSL0550
C	IPVT IS AN INTEGER INPUT ARRAY OF LENGTH N WHICH DEFINES THE	QRSL0560
C	PERMUTATION MATRIX P SUCH THAT $A * P = Q * R$. COLUMN J OF P	QRSL0570
C	IS COLUMN IPVT(J) OF THE IDENTITY MATRIX.	QRSL0580
C		QRSL0590
C	DIAG IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE	QRSL0600
C	DIAGONAL ELEMENTS OF THE MATRIX D.	QRSL0610
C		QRSL0620
C	QTB IS AN INPUT ARRAY OF LENGTH N WHICH MUST CONTAIN THE FIRST	QRSL0630
C	N ELEMENTS OF THE VECTOR $(Q \text{ TRANSPOSE}) * B$.	QRSL0640
C		QRSL0650
C	X IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE LEAST	QRSL0660
C	SQUARES SOLUTION OF THE SYSTEM $A * X = B$, $D * X = 0$.	QRSL0670
C		QRSL0680
C	SDIAG IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE	QRSL0690
C	DIAGONAL ELEMENTS OF THE UPPER TRIANGULAR MATRIX S.	QRSL0700
C		QRSL0710
C	WA IS A WORK ARRAY OF LENGTH N.	QRSL0720
C		QRSL0730
C	SUBPROGRAMS CALLED	QRSL0740
C		QRSL0750
C	FORTRAN-SUPPLIED ... DABS,DSQRT	QRSL0760
C		QRSL0770
C	ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.	QRSL0780
C	BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE	QRSL0790
C		QRSL0800
C	*****	QRSL0810
C	INTEGER I,J,JP1,K,KP1,L,NSING	QRSL0820
C	DOUBLE PRECISION COS,COTAN,P5,P25,QTBPJ,SIN,SUM,TAN,TEMP,ZERO	QRSL0830
C	DATA P5,P25,ZERO /5.0D-1,2.5D-1,0.0D0/	QRSL0840
C		QRSL0850
C	COPY R AND $(Q \text{ TRANSPOSE}) * B$ TO PRESERVE INPUT AND INITIALIZE S.	QRSL0860
C	IN PARTICULAR, SAVE THE DIAGONAL ELEMENTS OF R IN X.	QRSL0870
C		QRSL0880
C	DO 20 J = 1, N	QRSL0890
C	DO 10 I = J, N	QRSL0900
C	R(I,J) = R(J,I)	QRSL0910
C	10 CONTINUE	QRSL0920
C	X(J) = R(J,J)	QRSL0930
C	WA(J) = QTB(J)	QRSL0940
C	20 CONTINUE	QRSL0950
C		QRSL0960
C	ELIMINATE THE DIAGONAL MATRIX D USING A GIVENS ROTATION.	QRSL0970
C		QRSL0980
C	DO 100 J = 1, N	QRSL0990
C		QPFL1000
C	PREPARE THE ROW OF D TO BE ELIMINATED, LOCATING THE	QRSL1010
C	DIAGONAL ELEMENT USING P FROM THE QR FACTORIZATION.	QRSL1020
C		QRSL1030
C	L = IPVT(J)	QRSL1040
C	IF (DIAG(L) .EQ. ZERO) GO TO 90	QRSL1050
C	DO 30 K = J, N	QRSL1060
C	SDIAG(K) = ZERO	QRSL1070
C	30 CONTINUE	QRSL1080

```

SDIAG(J) = DIAG(L)
C
C THE TRANSFORMATIONS TO ELIMINATE THE ROW OF D
C MODIFY ONLY A SINGLE ELEMENT OF (Q TRANSPOSE)*B
C BEYOND THE FIRST N, WHICH IS INITIALLY ZERO.
C
QTBPJ = ZERO
DO 80 K = J, N
C
C DETERMINE A GIVENS ROTATION WHICH ELIMINATES THE
C APPROPRIATE ELEMENT IN THE CURRENT ROW OF D.
C
IF (SDIAG(K) .EQ. ZERO) GO TO 70
IF (DABS(R(K,K)) .GE. DABS(SDIAG(K))) GO TO 40
COTAN = R(K,K)/SDIAG(K)
SIN = P5/DSQRT(P25+P25*COTAN**2)
COS = SIN*COTAN
GO TO 50
40 CONTINUE
TAN = SDIAG(K)/R(K,K)
COS = P5/DSQRT(P25+P25*TAN**2)
SIN = COS*TAN
50 CONTINUE
C
C COMPUTE THE MODIFIED DIAGONAL ELEMENT OF R AND
C THE MODIFIED ELEMENT OF ((Q TRANSPOSE)*B,0).
C
R(K,K) = COS*R(K,K) + SIN*SDIAG(K)
TEMP = COS*WA(K) + SIN*QTBPJ
QTBPJ = -SIN*WA(K) + COS*QTBPJ
WA(K) = TEMP
C
C ACCUMULATE THE TRANSFORMATION IN THE ROW OF S.
C
KP1 = K + 1
IF (N .LT. KP1) GO TO 70
DO 60 I = KP1, N
TEMP = COS*R(I,K) + SIN*SDIAG(I)
SDIAG(I) = -SIN*R(I,K) + COS*SDIAG(I)
R(I,K) = TEMP
60 CONTINUE
70 CONTINUE
80 CONTINUE
90 CONTINUE
C
C STORE THE DIAGONAL ELEMENT OF S AND RESTORE
C THE CORRESPONDING DIAGONAL ELEMENT OF R.
C
SDIAG(J) = R(J,J)
R(J,J) = X(J)
100 CONTINUE
C
C SOLVE THE TRIANGULAR SYSTEM FOR Z. IF THE SYSTEM IS
C SINGULAR, THEN OBTAIN A LEAST SQUARES SOLUTION.

```

QRSL1090
QRSL1100
QRSL1110
QRSL1120
QRSL1130
QRSL1140
QRSL1150
QRSL1160
QRSL1170
QRSL1180
QRSL1190
QRSL1200
QRSL1210
QRSL1220
QRSL1230
QRSL1240
QRSL1250
QRSL1260
QRSL1270
QRSL1280
QRSL1290
QRSL1300
QRSL1310
QRSL1320
QRSL1330
QRSL1340
QRSL1350
QRSL1360
QRSL1370
QRSL1380
QRSL1390
QRSL1400
QRSL1410
QRSL1420
QRSL1430
QRSL1440
QRSL1450
QRSL1460
QRSL1470
QRSL1480
QRSL1490
QRSL1500
QRSL1510
QRSL1520
QRSL1530
QRSL1540
QRSL1550
QRSL1560
QRSL1570
QRSL1580
QRSL1590
QRSL1600
QRSL1610
QRSL1620

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

C	SINES OF THE TRANSFORMING GIVENS ROTATIONS.	RWUP0550
C		RWUP0560
C	SUBPROGRAMS CALLED	RWUP0570
C		RWUP0580
C	FORTRAN-SUPPLIED ... DABS,DSQRT	RWUP0590
C		RWUP0600
C	ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.	RWUP0610
C	BURTON S. GARBOW, DUDLEY V. GOETSCHEL, KENNETH E. HILLSTROM,	RWUP0620
C	JORGE J. MORE	RWUP0630
C		RWUP0640
C	*****	RWUP0650
	INTEGER I,J,JM1	RWUP0660
	DOUBLE PRECISION COTAN,ONE,P5,P25,ROWJ,TAN,TEMP,ZERO	RWUP0670
	DATA ONE,P5,P25,ZERO /1.0D0,5.0D-1,2.5D-1,0.0D0/	RWUP0680
C		RWUP0690
	DO 60 J = 1, N	RWUP0700
	ROWJ = W(J)	RWUP0710
	JM1 = J - 1	RWUP0720
C		RWUP0730
C	APPLY THE PREVIOUS TRANSFORMATIONS TO	RWUP0740
C	R(I,J), I=1,2,...,J-1, AND TO W(J).	RWUP0750
C		RWUP0760
	IF (JM1 .LT. 1) GO TO 20	RWUP0770
	DO 10 I = 1, JM1	RWUP0780
	TEMP = COS(I)*R(I,J) + SIN(I)*ROWJ	RWUP0790
	ROWJ = -SIN(I)*R(I,J) + COS(I)*ROWJ	RWUP0800
	R(I,J) = TEMP	RWUP0810
10	CONTINUE	RWUP0820
20	CONTINUE	RWUP0830
C		RWUP0840
C	DETERMINE A GIVENS ROTATION WHICH ELIMINATES W(J).	RWUP0850
C		RWUP0860
	COS(J) = ONE	RWUP0870
	SIN(J) = ZERO	RWUP0880
	IF (ROWJ .EQ. ZERO) GO TO 50	RWUP0890
	IF (DABS(R(J,J)) .GE. DABS(ROWJ)) GO TO 30	RWUP0900
	COTAN = R(J,J)/ROWJ	RWUP0910
	SIN(J) = P5/DSQRT(P25+P25*COTAN**2)	RWUP0920
	COS(J) = SIN(J)*COTAN	RWUP0930
	GO TO 40	RWUP0940
30	CONTINUE	RWUP0950
	TAN = ROWJ/R(J,J)	RWUP0960
	COS(J) = P5/DSQRT(P25+P25*TAN**2)	RWUP0970
	SIN(J) = COS(J)*TAN	RWUP0980
40	CONTINUE	RWUP0990
C		RWUP1000
C	APPLY THE CURRENT TRANSFORMATION TO R(J,J), B(J), AND ALPHA.	RWUP1010
C		RWUP1020
	R(J,J) = COS(J)*R(J,J) + SIN(J)*ROWJ	RWUP1030
	TEMP = COS(J)*B(J) + SIN(J)*ALPHA	RWUP1040
	ALPHA = -SIN(J)*B(J) + COS(J)*ALPHA	RWUP1050
	B(J) = TEMP	RWUP1060
50	CONTINUE	RWUP1070
60	CONTINUE	RWUP1080

C
C
C
RETURN
LAST CARD OF SUBROUTINE RWUPDT.
END

RWUP1090
RWUP1100
RWUP1110
RWUP1120
RWUP1130

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

SUBROUTINE R1MPYQ(M,N,A,LDA,V,W)
 INTEGER M,N,LDA
 DOUBLE PRECISION A(LDA,N),V(N),W(N)

SUBROUTINE R1MPYQ

GIVEN AN M BY N MATRIX A, THIS SUBROUTINE COMPUTES $A*Q$ WHERE
 Q IS THE PRODUCT OF $2*(N - 1)$ TRANSFORMATIONS

$$GV(N-1)*...*GV(1)*GW(1)*...*GW(N-1)$$

AND $GV(I)$, $GW(I)$ ARE GIVENS ROTATIONS IN THE (I,N) PLANE WHICH
 ELIMINATE ELEMENTS IN THE I -TH AND N -TH PLANES, RESPECTIVELY.
 Q ITSELF IS NOT GIVEN, RATHER THE INFORMATION TO RECOVER THE
 GV , GW ROTATIONS IS SUPPLIED.

THE SUBROUTINE STATEMENT IS

SUBROUTINE R1MPYQ(M,N,A,LDA,V,W)

WHERE

M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
 OF ROWS OF A.

N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
 OF COLUMNS OF A.

A IS AN M BY N ARRAY. ON INPUT A MUST CONTAIN THE MATRIX
 TO BE POSTMULTIPLIED BY THE ORTHOGONAL MATRIX Q
 DESCRIBED ABOVE. ON OUTPUT $A*Q$ HAS REPLACED A.

LDA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN M
 WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY A.

V IS AN INPUT ARRAY OF LENGTH N. $V(I)$ MUST CONTAIN THE
 INFORMATION NECESSARY TO RECOVER THE GIVENS ROTATION $GV(I)$
 DESCRIBED ABOVE.

W IS AN INPUT ARRAY OF LENGTH N: $W(I)$ MUST CONTAIN THE
 INFORMATION NECESSARY TO RECOVER THE GIVENS ROTATION $GW(I)$
 DESCRIBED ABOVE.

SUBROUTINES CALLED

FORTRAN-SUPPLIED ... DABS,DSQRT

ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
 BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE

INTEGER I,J,NMJ,NM1
 DOUBLE PRECISION COS,ONE,SIN,TEMP

R1MQ0010
 R1MQ0020
 R1MQ0030
 R1MQ0040
 R1MQ0050
 R1MQ0060
 R1MQ0070
 R1MQ0080
 R1MQ0090
 R1MQ0100
 R1MQ0110
 R1MQ0120
 R1MQ0130
 R1MQ0140
 R1MQ0150
 R1MQ0160
 R1MQ0170
 R1MQ0180
 R1MQ0190
 R1MQ0200
 R1MQ0210
 R1MQ0220
 R1MQ0230
 R1MQ0240
 R1MQ0250
 R1MQ0260
 R1MQ0270
 R1MQ0280
 R1MQ0290
 R1MQ0300
 R1MQ0310
 R1MQ0320
 R1MQ0330
 R1MQ0340
 R1MQ0350
 R1MQ0360
 R1MQ0370
 R1MQ0380
 R1MQ0390
 R1MQ0400
 R1MQ0410
 R1MQ0420
 R1MQ0430
 R1MQ0440
 R1MQ0450
 R1MQ0460
 R1MQ0470
 R1MQ0480
 R1MQ0490
 R1MQ0500
 R1MQ0510
 R1MQ0520
 R1MQ0530
 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)
 INTEGER M,N,LS
 LOGICAL SING
 DOUBLE PRECISION S(LS),U(M),V(N),W(M)

SUBROUTINE R1UPDT

GIVEN AN M BY N LOWER TRAPEZOIDAL MATRIX S, AN M-VECTOR U,
 AND AN N-VECTOR V, THE PROBLEM IS TO DETERMINE AN
 ORTHOGONAL MATRIX Q SUCH THAT

$$(S + U^T V)^T Q$$

IS AGAIN LOWER TRAPEZOIDAL.

THIS SUBROUTINE DETERMINES Q AS THE PRODUCT OF $2*(N - 1)$
 TRANSFORMATIONS

$$GV(N-1)*...*GV(1)*GW(1)*...*GW(N-1)$$

WHERE GV(I), GW(I) ARE GIVENS ROTATIONS IN THE (I,N) PLANE
 WHICH ELIMINATE ELEMENTS IN THE I-TH AND N-TH PLANES,
 RESPECTIVELY. Q ITSELF IS NOT ACCUMULATED, RATHER THE
 INFORMATION TO RECOVER THE GV, GW ROTATIONS IS RETURNED.

THE SUBROUTINE STATEMENT IS

SUBROUTINE R1UPDT(M,N,S,LS,U,V,W,SING)

WHERE

M IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
 OF ROWS OF S.

N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER
 OF COLUMNS OF S. N MUST NOT EXCEED M.

S IS AN ARRAY OF LENGTH LS. ON INPUT S MUST CONTAIN THE LOWER
 TRAPEZOIDAL MATRIX S STORED BY COLUMNS. ON OUTPUT S CONTAINS
 THE LOWER TRAPEZOIDAL MATRIX PRODUCED AS DESCRIBED ABOVE.

LS IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN
 $(N*(2*M-N+1))/2$.

U IS AN INPUT ARRAY OF LENGTH M WHICH MUST CONTAIN THE
 VECTOR U.

V IS AN ARRAY OF LENGTH N. ON INPUT V MUST CONTAIN THE VECTOR
 V. ON OUTPUT V(I) CONTAINS THE INFORMATION NECESSARY TO
 RECOVER THE GIVENS ROTATION GV(I) DESCRIBED ABOVE.

W IS AN OUTPUT ARRAY OF LENGTH M. W(I) CONTAINS INFORMATION

R1UP0010
 R1UP0020
 R1UP0030
 R1UP0040
 R1UP0050
 R1UP0060
 R1UP0070
 R1UP0080
 R1UP0090
 R1UP0100
 R1UP0110
 R1UP0120
 R1UP0130
 R1UP0140
 R1UP0150
 R1UP0160
 R1UP0170
 R1UP0180
 R1UP0190
 R1UP0200
 R1UP0210
 R1UP0220
 R1UP0230
 R1UP0240
 R1UP0250
 R1UP0260
 R1UP0270
 R1UP0280
 R1UP0290
 R1UP0300
 R1UP0310
 R1UP0320
 R1UP0330
 R1UP0340
 R1UP0350
 R1UP0360
 R1UP0370
 R1UP0380
 R1UP0390
 R1UP0400
 R1UP0410
 R1UP0420
 R1UP0430
 R1UP0440
 R1UP0450
 R1UP0460
 R1UP0470
 R1UP0480
 R1UP0490
 R1UP0500
 R1UP0510
 R1UP0520
 R1UP0530
 R1UP0540

	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	R1UP1630
	GO TO 100	R1UP1640
90	CONTINUE	R1UP1650
	TAN = W(J)/S(JJ)	R1UP1660
	COS = P5/DSQRT(P25+P25*TAN**2)	R1UP1670
	SIN = COS*TAN	R1UP1680
	TAU = SIN	R1UP1690
100	CONTINUE	R1UP1700
C		R1UP1710
C	APPLY THE TRANSFORMATION TO S AND REDUCE THE SPIKE IN W.	R1UP1720
C		R1UP1730
	L = JJ	R1UP1740
	DO 110 I = J, M	R1UP1750
	TEMP = COS*S(L) + SIN*W(I)	R1UP1760
	W(I) = -SIN*S(L) + COS*W(I)	R1UP1770
	S(L) = TEMP	R1UP1780
	L = L + 1	R1UP1790
110	CONTINUE	R1UP1800
C		R1UP1810
C	STORE THE INFORMATION NECESSARY TO RECOVER THE	R1UP1820
C	GIVENS ROTATION.	R1UP1830
C		R1UP1840
	W(J) = TAU	R1UP1850
120	CONTINUE	R1UP1860
C		R1UP1870
C	TEST FOR ZERO DIAGONAL ELEMENTS IN THE OUTPUT S.	R1UP1880
C		R1UP1890
	IF (S(JJ) .EQ. ZERO) SING = .TRUE.	R1UP1900
	JJ = JJ + (M - J + 1)	R1UP1910
130	CONTINUE	R1UP1920
140	CONTINUE	R1UP1930
C		R1UP1940
C	MOVE W BACK INTO THE LAST COLUMN OF THE OUTPUT S.	R1UP1950
C		R1UP1960
	L = JJ	R1UP1970
	DO 150 I = N, M	R1UP1980
	S(L) = W(I)	R1UP1990
	L = L + 1	R1UP2000
150	CONTINUE	R1UP2010
	IF (S(JJ) .EQ. ZERO) SING = .TRUE.	R1UP2020
	RETURN	R1UP2030
C		R1UP2040
C	LAST CARD OF SUBROUTINE R1UPDT.	R1UP2050
C		R1UP2060
	END	R1UP2070

REAL FUNCTION SPMPAR(I)	SPPR0010
INTEGER I	SPPR0020
*****	SPPR0030
C	SPPR0040
C	SPPR0050
FUNCTION SPMPAR	SPPR0060
C	SPPR0070
C	SPPR0080
C	SPPR0090
C	SPPR0100
C	SPPR0110
C	SPPR0120
C	SPPR0130
C	SPPR0140
C	SPPR0150
C	SPPR0160
C	SPPR0170
C	SPPR0180
C	SPPR0190
C	SPPR0200
C	SPPR0210
C	SPPR0220
C	SPPR0230
C	SPPR0240
C	SPPR0250
C	SPPR0260
C	SPPR0270
C	SPPR0280
C	SPPR0290
C	SPPR0300
C	SPPR0310
C	SPPR0320
C	SPPR0330
C	SPPR0340
C	SPPR0350
C	SPPR0360
C	SPPR0370
C	SPPR0380
C	SPPR0390
C	SPPR0400
C	SPPR0410
C	SPPR0420
C	SPPR0430
C	SPPR0440
C	SPPR0450
C	SPPR0460
C	SPPR0470
C	SPPR0480
C	SPPR0490
C	SPPR0500
C	SPPR0510
C	SPPR0520
C	SPPR0530
C	SPPR0540

THIS FUNCTION PROVIDES SINGLE PRECISION MACHINE PARAMETERS
 WHEN THE APPROPRIATE SET OF DATA STATEMENTS IS ACTIVATED (BY
 REMOVING THE C FROM COLUMN 1) AND ALL OTHER DATA STATEMENTS ARE
 RENDERED INACTIVE. MOST OF THE PARAMETER VALUES WERE OBTAINED
 FROM THE CORRESPONDING BELL LABORATORIES PORT LIBRARY FUNCTION.
 THE FUNCTION STATEMENT IS
 REAL FUNCTION SPMPAR(I)
 WHERE
 I IS AN INTEGER INPUT VARIABLE SET TO 1, 2, OR 3 WHICH
 SELECTS THE DESIRED MACHINE PARAMETER. IF THE MACHINE HAS
 T BASE B DIGITS AND ITS SMALLEST AND LARGEST EXPONENTS ARE
 EMIN AND EMAX, RESPECTIVELY, THEN THESE PARAMETERS ARE
 $SPMPAR(1) = B^{*(1 - T)}$, THE MACHINE PRECISION,
 $SPMPAR(2) = B^{*(EMIN - 1)}$, THE SMALLEST MAGNITUDE,
 $SPMPAR(3) = B^{*EMAX*(1 - B^{*(-T)})}$, THE LARGEST MAGNITUDE.
 ARGONNE NATIONAL LABORATORY. MINPACK PROJECT. MARCH 1980.
 BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE

 INTEGER MCHEPS(2)
 INTEGER MINMAG(2)
 INTEGER MAXMAG(2)
 REAL RMACH(3)
 EQUIVALENCE (RMACH(1),MCHEPS(1))
 EQUIVALENCE (RMACH(2),MINMAG(1))
 EQUIVALENCE (RMACH(3),MAXMAG(1))
 MACHINE CONSTANTS FOR THE IBM 360/370 SERIES,
 THE AMDAHL 470/V6, THE ICL 2900, THE ITEL AS/6,
 THE XEROX SIGMA 5/7/9 AND THE SEL SYSTEMS 85/86.
 DATA RMACH(1) / Z3C100000 /
 DATA RMACH(2) / Z00100000 /
 DATA RMACH(3) / Z7FFFFFFF /
 MACHINE CONSTANTS FOR THE HONEYWELL 600/6000 SERIES.
 DATA RMACH(1) / 0716400000000 /
 DATA RMACH(2) / 0402400000000 /
 DATA RMACH(3) / 037677777777 /

C		SPPR0550
C	MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.	SPPR0560
C		SPPR0570
C	DATA RMACH(1) / 1641400000000000000B /	SPPR0580
C	DATA RMACH(2) / 0001400000000000000B /	SPPR0590
C	DATA RMACH(3) / 3776777777777777777B /	SPPR0600
C		SPPR0610
C	MACHINE CONSTANTS FOR THE PDP-10 (KA OR KI PROCESSOR).	SPPR0620
C		SPPR0630
C	DATA RMACH(1) / "147400000000 /	SPPR0640
C	DATA RMACH(2) / "000400000000 /	SPPR0650
C	DATA RMACH(3) / "377777777777 /	SPPR0660
C		SPPR0670
C	MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING	SPPR0680
C	32-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).	SPPR0690
C		SPPR0700
C	DATA MCHEPS(1) / 889192448 /	SPPR0710
C	DATA MINMAG(1) / 8388608 /	SPPR0720
C	DATA MAXMAG(1) / 2147483647 /	SPPR0730
C		SPPR0740
C	DATA RMACH(1) / 006500000000 /	SPPR0750
C	DATA RMACH(2) / 000040000000 /	SPPR0760
C	DATA RMACH(3) / 017777777777 /	SPPR0770
C		SPPR0780
C	MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING	SPPR0790
C	16-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).	SPPR0800
C		SPPR0810
C	DATA MCHEPS(1),MCHEPS(2) / 13568, 0 /	SPPR0820
C	DATA MINMAG(1),MINMAG(2) / 128, 0 /	SPPR0830
C	DATA MAXMAG(1),MAXMAG(2) / 32767, -1 /	SPPR0840
C		SPPR0850
C	DATA MCHEPS(1),MCHEPS(2) / 0032400, 0000000 /	SPPR0860
C	DATA MINMAG(1),MINMAG(2) / 0000200, 0000000 /	SPPR0870
C	DATA MAXMAG(1),MAXMAG(2) / 0077777, 0177777 /	SPPR0880
C		SPPR0890
C	MACHINE CONSTANTS FOR THE BURROUGHS 5700/6700/7700 SYSTEMS.	SPPR0900
C		SPPR0910
C	DATA RMACH(1) / 013010000000000000 /	SPPR0920
C	DATA RMACH(2) / 017710000000000000 /	SPPR0930
C	DATA RMACH(3) / 007777777777777777 /	SPPR0940
C		SPPR0950
C	MACHINE CONSTANTS FOR THE BURROUGHS 1700 SYSTEM.	SPPR0960
C		SPPR0970
C	DATA RMACH(1) / Z4EA800000 /	SPPR0980
C	DATA RMACH(2) / Z400800000 /	SPPR0990
C	DATA RMACH(3) / Z5FFFFFFF /	SPPR1000
C		SPPR1010
C	MACHINE CONSTANTS FOR THE UNIVAC 1100 SERIES.	SPPR1020
C		SPPR1030
C	DATA RMACH(1) / 0147400000000 /	SPPR1040
C	DATA RMACH(2) / 0000400000000 /	SPPR1050
C	DATA RMACH(3) / 0377777777777 /	SPPR1060
C		SPPR1070
C	MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/200.	SPPR1080

C		SPPR1090
C	NOTE - IT MAY BE APPROPRIATE TO INCLUDE THE FOLLOWING CARD -	SPPR1100
C	STATIC RMACH(3)	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) / 037722400000000000000000B /	SPPR1240
C	DATA RMACH(2) / 020003400000000000000000B /	SPPR1250
C	DATA RMACH(3) / 05777777777777777777776B /	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) / :1777777777 /	SPPR1320
C		SPPR1330
C	SPMPAR = RMACH(I)	SPPR1340
	RETURN	SPPR1350
C		SPPR1360
C	LAST CARD OF FUNCTION SPMPAR.	SPPR1370
C		SPPR1380
	END	SPPR1390

C		DPPR0550
C	MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES.	DPPR0560
C		DPPR0570
C	DATA MCHEPS(1) / 15614000000000000000B /	DPPR0580
C	DATA MCHEPS(2) / 15010000000000000000B /	DPPR0590
C		DPPR0600
C	DATA MINMAG(1) / 00604000000000000000B /	DPPR0610
C	DATA MINMAG(2) / 00000000000000000000B /	DPPR0620
C		DPPR0630
C	DATA MAXMAG(1) / 3776777777777777777B /	DPPR0640
C	DATA MAXMAG(2) / 3716777777777777777B /	DPPR0650
C		DPPR0660
C	MACHINE CONSTANTS FOR THE PDP-10 (KA PROCESSOR).	DPPR0670
C		DPPR0680
C	DATA MCHEPS(1),MCHEPS(2) / "114400000000, "000000000000 /	DPPR0690
C	DATA MINMAG(1),MINMAG(2) / "033400000000, "000000000000 /	DPPR0700
C	DATA MAXMAG(1),MAXMAG(2) / "377777777777, "344777777777 /	DPPR0710
C		DPPR0720
C	MACHINE CONSTANTS FOR THE PDP-10 (KI PROCESSOR).	DPPR0730
C		DPPR0740
C	DATA MCHEPS(1),MCHEPS(2) / "104400000000, "000000000000 /	DPPR0750
C	DATA MINMAG(1),MINMAG(2) / "000400000000, "000000000000 /	DPPR0760
C	DATA MAXMAG(1),MAXMAG(2) / "377777777777, "377777777777 /	DPPR0770
C		DPPR0780
C	MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING	DPPR0790
C	32-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).	DPPR0800
C		DPPR0810
C	DATA MCHEPS(1),MCHEPS(2) / 620756992, 0 /	DPPR0820
C	DATA MINMAG(1),MINMAG(2) / 8388608, 0 /	DPPR0830
C	DATA MAXMAG(1),MAXMAG(2) / 2147483647, -1 /	DPPR0840
C		DPPR0850
C	DATA MCHEPS(1),MCHEPS(2) / 004500000000, 000000000000 /	DPPR0860
C	DATA MINMAG(1),MINMAG(2) / 000040000000, 000000000000 /	DPPR0870
C	DATA MAXMAG(1),MAXMAG(2) / 017777777777, 037777777777 /	DPPR0880
C		DPPR0890
C	MACHINE CONSTANTS FOR THE PDP-11 FORTRAN SUPPORTING	DPPR0900
C	16-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).	DPPR0910
C		DPPR0920
C	DATA MCHEPS(1),MCHEPS(2) / 9472, 0 /	DPPR0930
C	DATA MCHEPS(3),MCHEPS(4) / 0, 0 /	DPPR0940
C		DPPR0950
C	DATA MINMAG(1),MINMAG(2) / 128, 0 /	DPPR0960
C	DATA MINMAG(3),MINMAG(4) / 0, 0 /	DPPR0970
C		DPPR0980
C	DATA MAXMAG(1),MAXMAG(2) / 32767, -1 /	DPPR0990
C	DATA MAXMAG(3),MAXMAG(4) / -1, -1 /	DPPR1000
C		DPPR1010
C	DATA MCHEPS(1),MCHEPS(2) / 0022400, 0000000 /	DPPR1020
C	DATA MCHEPS(3),MCHEPS(4) / 0000000, 0000000 /	DPPR1030
C		DPPR1040
C	DATA MINMAG(1),MINMAG(2) / 0000200, 0000000 /	DPPR1050
C	DATA MINMAG(3),MINMAG(4) / 0000000, 0000000 /	DPPR1060
C		DPPR1070
C	DATA MAXMAG(1),MAXMAG(2) / 0077777, 0177777 /	DPPR1080

C	DATA MAXMAG(3),MAXMAG(4) / 0177777, 0177777 /	DPPR1090
C		DPPR1100
C	MACHINE CONSTANTS FOR THE BURROUGHS 6700/7700 SYSTEMS.	DPPR1110
C		DPPR1120
C	DATA MCHEPS(1) / 01451000000000000 /	DPPR1130
C	DATA MCHEPS(2) / 00000000000000000 /	DPPR1140
C		DPPR1150
C	DATA MINMAG(1) / 01771000000000000 /	DPPR1160
C	DATA MINMAG(2) / 07770000000000000 /	DPPR1170
C		DPPR1180
C	DATA MAXMAG(1) / 00777777777777777 /	DPPR1190
C	DATA MAXMAG(2) / 07777777777777777 /	DPPR1200
C		DPPR1210
C	MACHINE CONSTANTS FOR THE BURROUGHS 5700 SYSTEM.	DPPR1220
C		DPPR1230
C	DATA MCHEPS(1) / 01451000000000000 /	DPPR1240
C	DATA MCHEPS(2) / 00000000000000000 /	DPPR1250
C		DPPR1260
C	DATA MINMAG(1) / 01771000000000000 /	DPPR1270
C	DATA MINMAG(2) / 00000000000000000 /	DPPR1280
C		DPPR1290
C	DATA MAXMAG(1) / 00777777777777777 /	DPPR1300
C	DATA MAXMAG(2) / 00007777777777777 /	DPPR1310
C		DPPR1320
C	MACHINE CONSTANTS FOR THE BURROUGHS 1700 SYSTEM.	DPPR1330
C		DPPR1340
C	DATA MCHEPS(1) / ZCC6800000 /	DPPR1350
C	DATA MCHEPS(2) / Z000000000 /	DPPR1360
C		DPPR1370
C	DATA MINMAG(1) / ZC00800000 /	DPPR1380
C	DATA MINMAG(2) / Z000000000 /	DPPR1390
C		DPPR1400
C	DATA MAXMAG(1) / ZDFFFFFFF /	DPPR1410
C	DATA MAXMAG(2) / ZFFFFFFF /	DPPR1420
C		DPPR1430
C	MACHINE CONSTANTS FOR THE UNIVAC 1100 SERIES.	DPPR1440
C		DPPR1450
C	DATA MCHEPS(1),MCHEPS(2) / 0170640000000, 0000000000000 /	DPPR1460
C	DATA MINMAG(1),MINMAG(2) / 0000040000000, 0000000000000 /	DPPR1470
C	DATA MAXMAG(1),MAXMAG(2) / 0377777777777, 0777777777777 /	DPPR1480
C		DPPR1490
C	MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/200.	DPPR1500
C		DPPR1510
C	NOTE - IT MAY BE APPROPRIATE TO INCLUDE THE FOLLOWING CARD -	DPPR1520
C	STATIC DMACH(3)	DPPR1530
C		DPPR1540
C	DATA MINMAG/20K,3*0/,MAXMAG/77777K,3*177777K/	DPPR1550
C	DATA MCHEPS/32020K,3*0/	DPPR1560
C		DPPR1570
C	MACHINE CONSTANTS FOR THE HARRIS 220.	DPPR1580
C		DPPR1590
C	DATA MCHEPS(1),MCHEPS(2) / '20000000, '00000334 /	DPPR1600
C	DATA MINMAG(1),MINMAG(2) / '20000000, '00000201 /	DPPR1610
C	DATA MAXMAG(1),MAXMAG(2) / '37777777, '37777577 /	DPPR1620

C		DPPR1630
C	MACHINE CONSTANTS FOR THE CRAY-1.	DPPR1640
C		DPPR1650
C	DATA MCHEPS(1) / 037642400000000000000000B /	DPPR1660
C	DATA MCHEPS(2) / 000000000000000000000000B /	DPPR1670
C		DPPR1680
C	DATA MINMAG(1) / 020003400000000000000000B /	DPPR1690
C	DATA MINMAG(2) / 000000000000000000000000B /	DPPR1700
C		DPPR1710
C	DATA MAXMAG(1) / 057777777777777777777777B /	DPPR1720
C	DATA MAXMAG(2) / 0000007777777777777777776B /	DPPR1730
C		DPPR1740
C	MACHINE CONSTANTS FOR THE PRIME 400.	DPPR1750
C		DPPR1760
C	DATA MCHEPS(1),MCHEPS(2) / :10000000000, :00000000123 /	DPPR1770
C	DATA MINMAG(1),MINMAG(2) / :10000000000, :00000100000 /	DPPR1780
C	DATA MAXMAG(1),MAXMAG(2) / :17777777777, :37777677776 /	DPPR1790
C		DPPR1800
	DPMPAR = DMACH(I)	DPPR1810
	RETURN	DPPR1820
C		DPPR1830
C	LAST CARD OF FUNCTION DPMPAR.	DPPR1840
C		DPPR1850
	END	DPPR1860

Distribution for ANL-80-74Internal:

J. M. Boyle	J. J. Moré
W. J. Cody	National Energy Software Center (100)
T. F. Coleman	D. M. Pabis
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. Krisciunas	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)

