

MASTER

MASTER

ARGONNE NATIONAL LABORATORY

TESTING UNCONSTRAINED OPTIMIZATION SOFTWARE

by

Jorge J. Moré
Burton S. Garbow
Kenneth E. Hillstom



U of C-AUA-USDOE

**APPLIED
MATHEMATICS
DIVISION**

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

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or 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. Mention of commercial products, their manufacturers, or their suppliers in this publication does not imply or connote approval or disapproval of the product by Argonne National Laboratory or the U. S. Department of Energy.

ARGONNE NATIONAL LABORATORY
Argonne, Illinois 60439

NOTICE ~~MIN ONLY~~

PORTIONS OF THIS REPORT ARE ILLEGIBLE. It
has been reproduced from the best available
copy to permit the broadest possible avail-
ability.

TESTING UNCONSTRAINED OPTIMIZATION SOFTWARE *

by

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

Applied Mathematics Division

Technical Memorandum No. 324

July 1978

This report was prepared primarily for internal distribution.

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or 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.

* Work performed under the auspices of the U.S. Department of Energy.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

824

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

TABLE OF CONTENTS

ABSTRACT.	5
1. Introduction.	5
2. The Basic Subroutines	7
3. Test Functions.	9
4. Testing I	26
5. Testing II.	34
References.	39
Appendix 1. Basic Subroutines.	41
Appendix 2. Sample Driver and Interface Function	89
Appendix 3. Sample Data.	93

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

TESTING UNCONSTRAINED OPTIMIZATION SOFTWARE

by

Jorge J. Moré
Burton S. Garbow
Kenneth E. Hillstom

ABSTRACT

Much of the testing of optimization software is inadequate because the number of test functions is small or the starting points are close to the solution. In addition, there has been too much emphasis on measuring the efficiency of the software and not enough on testing reliability and robustness. To address this need, we have produced a relatively large but easy-to-use collection of test functions and designed guidelines for testing the reliability and robustness of unconstrained optimization software.

1. Introduction

When an algorithm is presented in the optimization literature, it has usually been tested on a set of functions. The purpose of this testing is to show that the algorithm works and, indeed, that it works better than other algorithms in the same problem area. In our opinion these claims are usually unwarranted because it is often the case that there are only a small number of test functions, and that the starting points are close to the solution.

Testing an algorithm on a relatively large set of test functions is bothersome because it requires the coding of the functions. This is a tedious and error-prone job that is avoided by many. However, not testing the algorithm on a large number of functions can easily lead the cynical observer to conclude that the algorithm was tuned to particular functions. Even aside from the cynical observer, the algorithm is just not well tested.

It is harder to understand why the standard starting points are usually close to the solution. One possible reason is that the algorithm developer is interested in testing the ability of the algorithm to deal with only one type of problem (e.g., a curved valley), and it is easier to force the algorithm to deal with this problem if the starting point is close to the solution.

Thus, a test function like Rosenbrock's is useful because it tests the ability of the algorithm to follow curved valleys. However, test functions like Rosenbrock's are the exception rather than the rule; other test functions have much more complicated features, and it has been observed that algorithms which succeed from the standard starting points often have problems from points farther away and fail. Hillstom [15] was one of the first to point out the need to test optimization software at non-standard starting points. He proposed using random starting points chosen from a box surrounding the standard starting point. This approach is much more satisfactory, but it tends to produce large amounts of data which can be hard to interpret. Moreover, the use of a random number generator complicates the reproduction of the results at other computing centers.

A final complaint against most of the testing procedures that have appeared in the literature is that there has been too much emphasis on comparing the efficiency of optimization routines and not enough emphasis on testing the reliability and robustness of optimization software -- the ability of a computer program to solve an optimization problem. It is important to measure the efficiency of optimization software, and this can be done, for example, by counting function evaluations or by timing the algorithm. However, either measure has problems, and with the standard starting points it is usually fairly hard to differentiate between similar algorithms (e.g., two quasi-Newton methods) on either count. In contrast, the use of points farther away from the solution will frequently reveal drastic differences in reliability and robustness between the programs, and hence in the number of function evaluations and in the timing of the algorithms.

To deal with the above problems, we have produced a relatively large collection of carefully coded test functions and designed very simple procedures for testing the reliability and robustness of unconstrained optimization software. The heart of our testing procedure is a set of basic subroutines, described in Sections 2 and 3, which define the test functions and the starting points. The attraction of these subroutines lies in their flexibility; with them it is possible to design many different kinds of tests for optimization software. Finally, in Sections 4 and 5 we describe some of the tests that we have been using to measure reliability and robustness.

It should be emphasized that the testing described in this paper is only a beginning and that other tests are necessary. For example, the ability of an algorithm to deal with small tolerances should be tested. However, the testing of Sections 4 and 5 does examine reliability and robustness in ways which other testing procedures have ignored.

2. The Basic Subroutines

Testing of optimization software requires a basic set of subroutines which define the test functions and the starting points. We consider the following three problem areas:

- I. Systems of nonlinear equations. Given $f_i: R^n \rightarrow R$ for $i = 1, \dots, n$, solve

$$f_i(x) = 0, \quad 1 \leq i \leq n, \quad x \in R^n.$$
- II. Nonlinear least squares. Given $f_i: R^n \rightarrow R$ for $i = 1, \dots, m$ with $m \geq n$, solve

$$\min \left\{ \sum_{i=1}^m f_i^2(x) : x \in R^n \right\}.$$
- III. Unconstrained minimization. Given $f: R^n \rightarrow R$, solve

$$\min \{ f(x) : x \in R^n \}.$$

The subroutines which define the test functions and starting points depend on the dimension parameters M and N and on the problem number $NPROB$. We first describe the subroutines for the test functions.

For systems of nonlinear equations, the subroutine

VECFCN(N,X,FVEC,NPROB)

returns in FVEC the vector

$(f_1(x), \dots, f_n(x))$.

In order to prevent gross inefficiencies with solvers which only require one component at a time,

COMFCN(N,K,X,FCNK,NPROB)

returns in FCNK the k-th component $f_k(x)$. For nonlinear least squares

SSQFCN(M,N,X,FVEC,NPROB)

returns in FVEC the vector

$$(f_1(x), \dots, f_m(x)) ,$$

and

SSQJAC(M,N,X,FJAC,LDFJAC,NPROB)

returns in FJAC the Jacobian matrix

$$\frac{\partial f_i(x)}{\partial x_j} , \quad i = 1, \dots, m, \quad j = 1, \dots, n .$$

(The parameter LDFJAC is the leading dimension of the array FJAC as defined in the main program.) For unconstrained minimization

OBJFCN(N,X,F,NPROB)

returns in F the objective function value $f(x)$ and

CRDFCN(N,X,G,NPROB)

returns in G the gradient vector

$$\left(\frac{\partial f(x)}{\partial x_1}, \dots, \frac{\partial f(x)}{\partial x_n} \right) .$$

For each problem area, the starting points are generated by a subroutine

INITPT(N,X,NPROB,FACTOR)

which returns in X the starting point corresponding to the parameters NPROB and FACTOR. If X_s denotes the standard starting point, then X will contain FACTOR* X_s , except that if X_s is the zero vector and FACTOR is not unity, then all the components of X will be set to FACTOR.

3. Test Functions

Almost all of the test functions that have appeared in the optimization literature are nonlinear least squares. Given a nonlinear least squares problem defined by f_1, \dots, f_m , we can obtain an unconstrained minimization problem by setting

$$(3.1) \quad f(x) = \sum_{i=1}^m f_i^2(x) .$$

If $m = n$, this problem can be posed as the system of nonlinear equations

$$(3.2) \quad f_i(x) = 0 , \quad 1 \leq i \leq n ,$$

and if $m > n$, the optimality conditions for (3.1) lead to the system of nonlinear equations

$$(3.3) \quad \sum_{i=1}^m \left(\frac{\partial f_i(x)}{\partial x_j} \right) f_i(x) = 0 , \quad 1 \leq j \leq n .$$

Note that in general it is inefficient to solve nonlinear least squares problems by general minimization algorithms, since they tend to ignore the structure in (3.1). As far as the nonlinear equations approach is concerned, (3.2) may not have any solutions, while (3.3) will have as a solution any critical point of (3.1). However, for testing purposes, (3.1), (3.2), and (3.3) are valid problems. All of our test functions are formulated for problem area II (nonlinear least squares). The corresponding test function for problem area III (unconstrained minimization) is (3.1), while for problem area I (systems of nonlinear equations), the function is (3.2) if $m = n$ and (3.3) if $m > n$. A given test function may appear in more than one problem area; coding differences among its various versions depend on the particular area. For nonlinear least squares, we need to generate the Jacobian matrix which requires an m by n array, while for unconstrained minimization and systems of equations, this two-dimensional array is not needed.

To define the test functions we have adopted the following general format:

Name of function [reference]

- a) Dimensions
- b) Function definition
- c) Standard starting point (designated x_0)
- d) Minima .

In d) we give the minima of the function (3.1) that we have found, and if convenient, the corresponding minimizer. In a few cases, the minimizer is, for example, of the form $(\alpha, \beta, +\infty)$. This means that

$$\lim_{\gamma \rightarrow +\infty} \nabla f(\alpha, \beta, \gamma) = 0 ,$$

and thus an algorithm may decide that a minimizer is in a neighborhood of (α, β, γ) for some large value of γ .

1) Rosenbrock function [24]

a) $n = 2, \quad m = 2$

b) $f_1(x) = 10(x_2 - x_1^2)$

$f_2(x) = 1 - x_1$

c) $x_0 = (-1.2, 1)$

d) $f = 0$ at $(1, 1)$

2) Freudenstein and Roth function [13]

a) $n = 2, \quad m = 2$

b) $f_1(x) = -13 + x_1 + ((5 - x_2)x_2 - 2)x_2$

$f_2(x) = -29 + x_1 + ((x_2 + 1)x_2 - 14)x_2$

c) $x_0 = (0.5, -2)$

d) $f = 0$ at $(5, 4)$

$f = 48.9842\dots$ at $(11.41\dots, -0.8968\dots)$

3) Powell badly scaled function [22]

a) $n = 2, \quad m = 2$

b) $f_1(x) = 10^4 x_1 x_2 - 1$

$f_2(x) = \exp[-x_1] + \exp[-x_2] - 1.0001$

c) $x_0 = (0,1)$

d) $f = 0$ at $(1.098...10^{-5}, 9.106...)$

4) Brown badly scaled function [unpublished]

a) $n = 2, m = 3$

b) $f_1(x) = x_1 - 10^6$

$f_2(x) = x_2 - 2 \cdot 10^{-6}$

$f_3(x) = x_1 x_2 - 2$

c) $x_0 = (1,1)$

d) $f = 0$ at $(10^6, 2 \cdot 10^{-6})$

5) Beale function [2]

a) $n = 2, m = 3$

b) $f_i(x) = y_i - x_1(1 - x_2^i)$

where $y_1 = 1.5, y_2 = 2.25, y_3 = 2.625$

c) $x_0 = (1,1)$

d) $f = 0$ at $(3,0.5)$

6) Jennrich and Sampson function [16]

a) $n = 2, m \geq n$

b) $f_i(x) = 2 + 2i - (\exp[ix_1] + \exp[ix_2])$

c) $x_0 = (0.3, 0.4)$

d) $f = 124.362... \text{ at } x_1 = x_2 = 0.2578... \text{ for } m = 10$

7) Helical valley function [11]

a) $n = 3, m = 3$

b) $f_1(x) = 10[x_3 - 10\theta(x_1, x_2)]$

$$f_2(x) = 10[(x_1^2 + x_2^2)^{\frac{1}{2}} - 1]$$

$$f_3(x) = x_3$$

where

$$\theta(x_1, x_2) = \begin{cases} \frac{1}{2\pi} \arctan\left(\frac{x_2}{x_1}\right) & \text{if } x_1 > 0 \\ \frac{1}{2\pi} \arctan\left(\frac{x_2}{x_1}\right) + 0.5 & \text{if } x_1 < 0 \end{cases}$$

c) $x_0 = (-1, 0, 0)$

d) $f = 0$ at $(1, 0, 0)$

8) Bard function [1]

a) $n = 3, m = 15$

b)
$$f_i(x) = y_i - \left(x_1 + \frac{u_i}{v_i x_2 + w_i x_3} \right)$$

where $u_i = i$, $v_i = 16 - i$, $w_i = \min(u_i, v_i)$, and

i	y_i
1	0.14
2	0.18
3	0.22
4	0.25
5	0.29
6	0.32
7	0.35
8	0.39
9	0.37
10	0.58
11	0.73
12	0.96
13	1.34
14	2.10
15	4.39

c) $x_0 = (1,1,1)$

d) $f = 8.21487... 10^{-3}$

$f = 17.4286... \text{ at } (0.8406..., -\infty, -\infty)$

9) Gaussian function [unpublished]

a) $n = 3, m = 15$

b) $f_i(x) = x_1 \exp\left[\frac{-x_2(t_i - x_3)^2}{2}\right] - y_i$

where $t_i = (8-i)/2$ and

i	y_i
1,15	0.0009
2,14	0.0044
3,13	0.0175
4,12	0.0540
5,11	0.1295
6,10	0.2420
7,9	0.3521
8	0.3989

c) $x_0 = (0.4,1,0)$

d) $f = 1.12793... 10^{-8}$

10) Meyer function [18]

a) $n = 3, m = 16$

b) $f_i(x) = x_1 \exp\left[\frac{x_2}{(t_i + x_3)}\right] - y_i$

where $t_i = 45+5i$ and

i	y_i	i	y_i
1	34780	9	8261
2	28610	10	7030
3	23650	11	6005
4	19630	12	5147
5	16370	13	4427
6	13720	14	3820
7	11540	15	3307
8	9744	16	2872

c) $x_0 = (0.02, 4000, 250)$

d) $f = 87.9458...$

11) Gulf research and development function [10]

a) $n = 3, \quad n \leq m \leq 100$

b) $f_i(x) = \exp\left[-\frac{|y_i - x_2|^{x_3}}{x_1}\right] - t_i$

where $t_i = i/100$ and

$$y_i = 25 + (-50 \ln(t_i))^{2/3}$$

c) $x_0 = (5, 2.5, 0.15)$

d) $f = 0$ at $(50, 25, 1.5)$

12) Box 3-dimensional function [4]

a) $n = 3, \quad m \geq n$ variable

b) $f_i(x) = \exp[-t_i x_1] - \exp[-t_i x_2] - x_3(\exp[-t_i] - \exp[-10t_i])$

where $t_i = (0.1)i$

c) $x_0 = (0, 10, 20)$

d) $f = 0$ at $(1, 10, 1)$, $(10, 1, -1)$ and wherever $(x_1 = x_2$ and $x_3 = 0)$

13) Powell singular function [23]

a) $n = 4, \quad m = 4$

b) $f_1(x) = x_1 + 10x_2$

$$f_2(x) = 5^{1/2}(x_3 - x_4)$$

$$f_3(x) = (x_2 - 2x_3)^2$$

$$f_4(x) = 10^{1/2}(x_1 - x_4)^2$$

c) $x_0 = (3, -1, 0, 1)$

d) $f = 0$ at the origin

14) Wood function [9]

a) $n = 4, m = 6$

b) $f_1(x) = 10(x_2 - x_1^2)$

$f_2(x) = 1 - x_1$

$f_3(x) = (90)^{\frac{1}{2}}(x_4 - x_3^2)$

$f_4(x) = 1 - x_3$

$f_5(x) = (10)^{\frac{1}{2}}(x_2 + x_4 - 2)$

$f_6(x) = (10)^{-\frac{1}{2}}(x_2 - x_4)$

c) $x_0 = (-3, -1, -3, -1)$

d) $f = 0$ at $(1, 1, 1, 1)$

15) Kowalik and Osborne function [17]

a) $n = 4, m = 11$

b)
$$f_i(x) = y_i - \frac{x_1(u_i^2 + u_i x_2)}{(u_i^2 + u_i x_3 + x_4)}$$

where

i	y_i	u_i
1	0.1957	4.0000
2	0.1947	2.0000
3	0.1735	1.0000
4	0.1600	0.5000
5	0.0844	0.2500
6	0.0627	0.1670
7	0.0456	0.1250
8	0.0342	0.1000
9	0.0323	0.0833
10	0.0235	0.0714
11	0.0246	0.0625

c) $x_0 = (0.25, 0.39, 0.415, 0.39)$

d) $f = 3.07505 \dots 10^{-4}$

$f = 1.02734 \dots 10^{-3}$ at $(+\infty, -14.07 \dots, -\infty, -\infty)$

16) Brown and Dennis function [5]

a) $n = 4, m \geq n$ variable

b) $f_i(x) = (x_1 + t_i x_2 - \exp[t_i])^2 + (x_3 + x_4 \sin(t_i) - \cos(t_i))^2$
where $t_i = i/5$.

c) $x_0 = (25, 5, -5, -1)$

d) $f = 85822.2...$ if $m = 20$

17) Osborne 1 function [21]

a) $n = 5, m = 33$

b) $f_i(x) = y_i - (x_1 + x_2 \exp[-t_i x_4] + x_3 \exp[-t_i x_5])$
where $t_i = 10(i-1)$ and

i	y_i	i	y_i
1	0.844	18	0.558
2	0.908	19	0.538
3	0.932	20	0.522
4	0.936	21	0.506
5	0.925	22	0.490
6	0.908	23	0.478
7	0.881	24	0.467
8	0.850	25	0.457
9	0.818	26	0.448
10	0.784	27	0.438
11	0.751	28	0.431
12	0.718	29	0.424
13	0.685	30	0.420
14	0.658	31	0.414
15	0.628	32	0.411
16	0.603	33	0.406
17	0.580		

c) $x_0 = (0.5, 1.5, -1, 0.01, 0.02)$

d) $f = 5.46489... \cdot 10^{-5}$

18) Biggs EXP6 function [3]

a) $n = 6, m \geq n$ variable

b) $f_i(x) = x_3 \exp[-t_i x_1] - x_4 \exp[-t_i x_2] + x_6 \exp[-t_i x_5] - y_i$

where $t_i = (0.1)i$ and

$$y_i = \exp[-t_i] - 5\exp[-10t_i] + 3\exp[-4t_i]$$

c) $x_0 = (1, 2, 1, 1, 1, 1)$

d) $f = 5.65565 \dots 10^{-3}$ if $m = 13$

19) Osborne 2 function [21]

a) $n = 11, m = 65$

b)
$$f_i(x) = y_i - (x_1 \exp[-t_i x_5] + x_2 \exp[-(t_i - x_9)^2 x_6] + x_3 \exp[-(t_i - x_{10})^2 x_7] + x_4 \exp[-(t_i - x_{11})^2 x_8])$$

where $t_i = (i-1)/10$ and

i	y_i	i	y_i	i	y_i
1	1.366	23	0.694	45	0.672
2	1.191	24	0.644	46	0.708
3	1.112	25	0.624	47	0.633
4	1.013	26	0.661	48	0.668
5	0.991	27	0.612	49	0.645
6	0.885	28	0.558	50	0.632
7	0.831	29	0.533	51	0.591
8	0.847	30	0.495	52	0.559
9	0.786	31	0.500	53	0.597
10	0.725	32	0.423	54	0.625
11	0.746	33	0.395	55	0.739
12	0.679	34	0.375	56	0.710
13	0.608	35	0.372	57	0.729
14	0.655	36	0.391	58	0.720
15	0.616	37	0.396	59	0.636
16	0.606	38	0.405	60	0.581
17	0.602	39	0.428	61	0.428
18	0.626	40	0.429	62	0.292
19	0.651	41	0.523	63	0.162
20	0.724	42	0.562	64	0.098
21	0.649	43	0.607	65	0.054
22	0.649	44	0.653		

c) $x_0 = (1.3, 0.65, 0.65, 0.7, 0.6, 3, 5, 7, 2, 4.5, 5.5)$

d) $f = 4.01377... \cdot 10^{-2}$

20) Watson function [17]

a) $2 \leq n \leq 31, m = 31$

b)
$$f_i(x) = \sum_{j=2}^n (j-1)x_j t_i^{j-2} - \left(\sum_{j=1}^n x_j t_i^{j-1} \right)^2 - 1$$

where $t_i = i/29, 1 \leq i \leq 29.$

$f_{30}(x) = x_1, f_{31}(x) = x_2 - x_1^2 - 1$

c) $x_0 = (0, \dots, 0)$

d) $f = 2.28767... \cdot 10^{-3}$ if $n = 6$

$f = 1.39976... \cdot 10^{-6}$ if $n = 9$

$f = 4.72238... \cdot 10^{-10}$ if $n = 12$

21) Extended Rosenbrock function [25]

a) n variable but even, $m = n$

b) $f_{2i-1}(x) = 10(x_{2i} - x_{2i-1})$

$f_{2i}(x) = 1 - x_{2i-1}$

c) $x_0 = (\xi_j)$ where $\xi_{2j-1} = -1.2, \xi_{2j} = 1$

d) $f = 0$ at $(1, \dots, 1)$

22) Extended Powell singular function [25]

a) n variable but a multiple of 4, $m = n$

b) $f_{4i-3}(x) = x_{4i-3} + 10x_{4i-2}$

$f_{4i-2}(x) = 5^{\frac{1}{2}}(x_{4i-1} - x_{4i})$

$f_{4i-1}(x) = (x_{4i-2} - 2x_{4i-1})^2$

$f_{4i}(x) = 10^{\frac{1}{2}}(x_{4i-3} - x_{4i})^2$

c) $x_0 = (\xi_j)$ where $\xi_{4j-3} = 3$, $\xi_{4j-2} = -1$, $\xi_{4j-1} = 0$, $\xi_{4j} = 1$

d) $f = 0$ at the origin

23) Penalty function I [14]

a) n variable, $m = n+1$

b) $f_i(x) = a^{\frac{1}{2}}(x_i - 1)$, $1 \leq i \leq n$

$$f_{n+1}(x) = \left(\sum_{j=1}^n x_j^2 \right) - \frac{1}{4}$$

where $a = 10^{-5}$

c) $x_0 = (\xi_j)$ where $\xi_j = j$

d) $f = 2.24997 \dots 10^{-5}$ if $n = 4$

$f = 7.08765 \dots 10^{-5}$ if $n = 10$

24) Penalty function II [14]

a) n variable, $m = 2n$

b) $f_1(x) = x_1 - 0.2$

$$f_i(x) = a^{\frac{1}{2}} \left(\exp \left[\frac{x_i}{10} \right] + \exp \left[\frac{x_{i-1}}{10} \right] - y_i \right), \quad 2 \leq i \leq n$$

$$f_i(x) = a^{\frac{1}{2}} \left(\exp \left[\frac{x_{i-n+1}}{10} \right] - \exp \left[\frac{1}{10} \right] \right), \quad n < i < 2n$$

$$f_{2n}(x) = \left(\sum_{j=1}^n (n-j+1) x_j^2 \right) - 1$$

where $a = 10^{-5}$ and $y_i = \exp \left[\frac{i}{10} \right] + \exp \left[\frac{i-1}{10} \right]$.

c) $x_0 = (\frac{1}{2}, \dots, \frac{1}{2})$

d) $f = 9.37629 \dots 10^{-6}$ if $n = 4$

$f = 2.93660 \dots 10^{-4}$ if $n = 10$

25) Variably dimensioned function [unpublished]a) n variable, $m = n+2$ b) $f_i(x) = x_i - 1, \quad i = 1, \dots, n$

$$f_{n+1}(x) = \sum_{j=1}^n j(x_j - 1)$$

$$f_{n+2}(x) = \left(\sum_{j=1}^n j(x_j - 1) \right)^2$$

c) $x_0 = (\xi_j)$ where $\xi_j = 1 - (j/n)$ d) $f = 0$ at $(1, \dots, 1)$ 26) Trigonometric function [25]a) n variable, $m = n$ b) $f_i(x) = n - \sum_{j=1}^n \cos x_j + i(1 - \cos x_i) - \sin x_i$ c) $x_0 = (1/n, \dots, 1/n)$ d) $f = 0$ 27) Brown almost linear function [6]a) n variable, $m = n$ b) $f_i(x) = x_i + \sum_{j=1}^n x_j - (n+1), \quad 1 \leq i < n$

$$f_n(x) = \left(\prod_{j=1}^n x_j \right) - 1$$

c) $x_0 = (1/2, \dots, 1/2)$ d) $f = 0$ at $(\alpha, \dots, \alpha, \alpha^{1-n})$ where α satisfies

$$n\alpha^n - (n+1)\alpha^{n-1} + 1 = 0; \text{ in particular, } \alpha = 1.$$

 $f = 1$ at $(0, \dots, 0, n+1)$

28) Discrete boundary value function [19]

- a) n variable, $m = n$
- b) $f_i(x) = 2x_i - x_{i-1} - x_{i+1} + h^2(x_i + t_i + 1)^3/2$
where $h = 1/(n+1)$, $t_i = ih$, and $x_0 = x_{n+1} = 0$.
- c) $x_0 = (\xi_j)$ where $\xi_j = t_j(t_j - 1)$
- d) $f = 0$

29) Discrete integral function [19]

- a) n variable, $m = n$
- b) $f_i(x) = x_i + h \left[(1-t_i) \sum_{j=1}^i t_j (x_j + t_j + 1)^3 \right. \\ \left. + t_i \sum_{j=i+1}^n (1-t_j) (x_j + t_j + 1)^3 \right] / 2$
where $h = 1/(n+1)$, $t_i = ih$, and $x_0 = x_{n+1} = 0$.
- c) $x_0 = (\xi_j)$ where $\xi_j = t_j(t_j - 1)$
- d) $f = 0$

30) Broyden tridiagonal function [7]

- a) n variable, $m = n$
- b) $f_i(x) = (3-2x_i)x_i - x_{i-1} - 2x_{i+1} + 1$
where $x_0 = x_{n+1} = 0$
- c) $x_0 = (-1, \dots, -1)$
- d) $f = 0$

31) Broyden banded function [8]a) n variable, $m = n$

$$b) \quad f_i(x) = x_i(2+5x_i^2) + 1 - \sum_{j \in J_i} x_j(1+x_j)$$

where $J_i = \{j: j \neq i, \max(1, i-m_\ell) \leq j \leq \min(n, i+m_u)\}$

and $m_\ell = 5, \quad m_u = 1.$

c) $x_0 = (-1, \dots, -1)$ d) $f = 0$ 32) Linear function - full rank [unpublished]a) n variable, $m \geq n$

$$b) \quad f_i(x) = x_i - \frac{2}{m} \left(\sum_{j=1}^n x_j \right) - 1, \quad 1 \leq i \leq n$$

$$f_i(x) = -\frac{2}{m} \left(\sum_{j=1}^n x_j \right) - 1, \quad n < i \leq m$$

c) $x_0 = (1, \dots, 1)$ d) $f = m-n$ at $(-1, \dots, -1)$ 33) Linear function - rank 1 [unpublished]a) n variable, $m \geq n$

$$b) \quad f_i(x) = i \left(\sum_{j=1}^n jx_j \right) - 1$$

c) $x_0 = (1, \dots, 1)$

d) $f = \frac{m(m-1)}{2(2m+1)}$ at any point where $\sum_{j=1}^n jx_j = \frac{3}{2m+1}$

34) Linear function - rank 1 with zero columns and rows [unpublished]

a) n variable, $m \geq n$

b) $f_1(x) = -1, f_m(x) = -1$

$$f_i(x) = (i-1) \left(\sum_{j=2}^{n-1} jx_j \right) - 1, \quad 2 \leq i < m$$

c) $x_0 = (1, \dots, 1)$

d) $f = \frac{m^2 + 3m - 6}{2(2m-3)}$ at any point where $\sum_{j=2}^{m-1} jx_j = \frac{3}{2m-3}$

35) Chebyquad function [12]

a) n variable, $m \geq n$

b) $f_i(x) = \frac{1}{n} \sum_{j=1}^n T_i(x_j) - \int_0^1 T_i(x) dx$

where T_i is the i^{th} Chebyshev polynomial shifted to the interval $[0,1]$ and hence,

$$\int_0^1 T_i(x) dx = 0 \text{ for } i \text{ odd, } \int_0^1 T_i(x) dx = \frac{-1}{(i^2-1)} \text{ for } i \text{ even}$$

c) $x_0 = (\xi_j)$ where $\xi_j = j/(n+1)$

d) $f = 0$ for $1 \leq n \leq 7$ and $n = 9$

$f = 3.51687 \dots 10^{-3}$ for $n = 8$

$f = 6.50395 \dots 10^{-3}$ for $n = 10$

For ease of reference, we list the functions appearing in the three test problem collections. Note that the number in parentheses after the name of the function refers to the number of the function in the main list. Also note that some of the basic subroutines of Section 2 can be used to test algorithms from more than one problem area. For example, GRDFCN effectively defines a collection of nonlinear equation problems and therefore can be used to test nonlinear equation solvers, while SSQFCN and SSQJAC can be used together to test unconstrained minimization algorithms.

Systems of nonlinear equations

1. Rosenbrock function (1)
2. Powell singular function (13)
3. Powell badly scaled function (3)
4. Wood function (14)
5. Helical valley function (7)
6. Watson function (20)
7. Chebyquad function (35)
8. Brown almost-linear function (27)
9. Discrete boundary value function (28)
10. Discrete integral equation function (29)
11. Trigonometric function (26)
12. Variably dimensioned function (25)
13. Broyden tridiagonal function (30)
14. Broyden banded function (31)

Nonlinear least squares

1. Linear function - full rank (32)
2. Linear function - rank 1 (33)
3. Linear function - rank 1 with zero columns and rows (34)
4. Rosenbrock function (1)
5. Helical valley function (7)
6. Powell singular function (13)
7. Freudenstein and Roth function (2)
8. Bard function (8)
9. Kowalik and Osborne function (15)
10. Meyer function (10)
11. Watson function (20)
12. Box 3-dimensional function (12)
13. Jennrich and Sampson function (6)
14. Brown and Dennis function (16)
15. Chebyquad function (35)
16. Brown almost-linear function (27)
17. Osborne 1 function (17)
18. Osborne 2 function (19)

Unconstrained Minimization

1. Helical valley function (7)
2. Biggs EXP6 function (18)
3. Gaussian function (9)
4. Powell badly scaled function (3)
5. Box 3-dimensional function (12)
6. Variably dimensioned function (25)
7. Watson function (20)
8. Penalty function I (23)
9. Penalty function II (24)
10. Brown badly scaled function (4)
11. Brown and Dennis function (16)
12. Gulf research and development function (11)
13. Trigonometric function (26)
14. Extended Rosenbrock function (21)
15. Extended Powell singular function (22)
16. Beale function (5)
17. Wood function (14)
18. Chebyquad function (35)

4. Testing I

With the basic subroutines and the test functions described in Sections 2 and 3, we have the tools for testing unconstrained nonlinear optimization algorithms. In this section we would like to mention some of the possible tests that can be carried out.

Suppose, for example, that we want to test a nonlinear least squares algorithm SOLVER on a given test function. This can be done by the following program outline.

```
(4.1)      EXTERNAL FCN
            READ ( , ) NPROB,N,M,NTRIES
            FACTOR = 1.0
            DO K = 1,NTRIES
            |   CALL INITPT(N,X,NPROB,FACTOR)
            |   CALL SOLVER(FCN,M,N,X,...)
            |   FACTOR = 10.0*FACTOR
```

The choice of the integer NTRIES depends on the function defined by NPROB, and on how stringently we want to test SOLVER. If the function contains rapidly growing sub-functions such as exponentials, then NTRIES = 1 is probably all that should be allowed. For other functions, NTRIES = 3 may be a reasonable setting; this tests SOLVER with starting vectors of x_s , $10x_s$, and $100x_s$ where x_s is the standard starting vector. The vectors x_s and $100x_s$ are regarded as being close to and far away from the solution, respectively; it is not unusual for algorithms to succeed with x_s but to fail with $100x_s$.

In (4.1), SOLVER calls an interface subroutine FCN. The calling sequence for FCN should be identical to the calling sequence of the function subroutine in SOLVER; its main purpose is to call the testing functions with the appropriate value of problem number. For example, if the calling sequence of the function subroutine in SOLVER is

```
FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG) ,
```

then the body of FCN could be

```
COMMON /REFNUM/ NPROB,NFEV,NJEV
IF IFLAG = 1
|   CALL SSQFCN(M,N,X,FVEC,NPROB)
|   NFEV = NFEV+1
IF IFLAG = 2
|   CALL SSQJAC(M,N,X,FJAC,LDFJAC,NPROB)
|   NJEV = NJEV+1
```

Note that the COMMON block REFNUM transmits the variable NPROB and provides counters for the number of function and Jacobian evaluations required by SOLVER.

Nothing that has been said is intrinsic to the nonlinear least squares problem; the same type of driver can be used for nonlinear equations or unconstrained minimization. We emphasize that the test results provided by (4.1) can be quite revealing if NTRIES is set properly. For example, to compare the choices of scaling strategy, Table 1 was presented in [20]. In this table "FC" means failure to converge within 1000 function evaluations.

Table 1

PROBLEM	SCALING	x_s		$10x_s$		$100x_s$	
		NFEV	NJEV	NFEV	NJEV	NFEV	NJEV
1	Initial	12	9	34	29	FC	FC
	Adaptive	11	8	20	15	19	16
	Continuous	12	9	14	12	176	141
2	Initial	19	17	81	71	365	315
	Adaptive	18	16	79	71	348	307
	Continuous	18	16	63	54	FC	FC
3	Initial	8	7	37	36	14	13
	Adaptive	8	7	37	36	14	13
	Continuous	8	7	FC	FC	FC	FC
4	Initial	268	242	423	400	FC	FC
	Adaptive	268	242	57	47	229	207
	Continuous	FC	FC	FC	FC	FC	FC

It is clear from this table that the adaptive scaling strategy is best in these four examples, and that we could not have reached this conclusion if we had only considered the standard starting points.

We have shown how to use the basic subroutines to test different versions of the same algorithm, and in this case comparisons are straightforward. However, these subroutines will inevitably be used to test and compare different algorithms. Comparisons are then more difficult because the two algorithms will usually have different stopping criteria, and it may not be immediately clear how much of the success of the algorithm is due to its stopping criteria. However, the effect of the stopping criteria can be measured by running the

program with different tolerances or by looking at the progress of the iteration.

To illustrate the use of the basic subroutines in the testing of algorithms, consider two nonlinear least squares subroutines NLSQ1 and NLSQ2. The names have been changed to protect the innocent, but it should be realized that the development of each of these codes has received considerable attention; both of them appear in optimization libraries. These subroutines have an output parameter which indicates the status of the computation, and in Tables 2 and 3 we have used the parameter INFO to report this information. If the subroutine claims success then INFO is set to 1, and otherwise it is set to 0.

We first ran these algorithms with the standard starting points; the results are shown in Tables 2 and 3. The following points are worthy of mention:

- (a) There are three problems (10,14,17) in which NLSQ2 required more than 100 function evaluations. On each of these problems NLSQ1 required fewer function evaluations.
- (b) For problem 15 with $n = 1$, the standard starting point is a critical point. NLSQ1 claimed success on this problem while NLSQ2 classified this problem as a possible failure.
- (c) The results for problem 16 with $n = 40$ are not comparable because the algorithms converged to different local minima.
- (d) A look at the progress of the iteration shows that both algorithms were converging at the same rate on problem 6, but differences in convergence criteria caused NLSQ1 to work much harder.
- (e) Problems 2 and 3 are rank-deficient linear problems, and the differences in performance can be traced to the fact that NLSQ1 uses orthogonal transformations to solve the linear least squares subproblems, while NLSQ2 uses Cholesky decomposition on the normal equations.
- (f) On the remainder of the problems both algorithms required only a small number of function evaluations (less than 50).

Table 2

SUMMARY OF 28 CALIS TC NLSQ1

NPROB	N	M	NFEV	NJEV	INFO	FINAL L2 NORM
1	5	10	3	2	1	0.2236068D 01
1	5	50	3	2	1	0.6708204D 01
2	5	10	3	2	1	0.1463850D 01
2	5	50	3	2	1	0.3482630D 01
3	5	10	3	2	1	0.1909727D 01
3	5	50	3	2	1	0.3691729D 01
4	2	2	18	14	1	0.0
5	3	3	12	9	1	0.9195638D-32
6	4	4	68	62	1	0.9523448D-35
7	2	2	17	10	1	0.6998875D 01
8	3	15	7	6	1	0.9063596D-01
9	4	11	23	21	1	0.1753584D-01
10	3	16	136	120	1	0.9377945D 01
11	6	31	9	8	1	0.4782959D-01
11	9	31	9	8	1	0.1183115D-02
11	12	31	10	9	1	0.2173104D-04
12	3	10	8	7	1	0.7211110D-16
13	2	10	25	14	1	0.1115178D 02
14	4	20	315	282	1	0.2929543D 03
15	1	8	1	1	1	0.1886238D 01
15	8	8	44	24	1	0.5930324D-01
15	9	9	11	8	1	0.3304872D-15
15	10	10	24	14	1	0.8064710D-01
16	10	10	17	15	1	0.8987408D-15
16	30	30	20	15	1	0.2170133D-14
16	40	40	19	14	1	0.1254229D-12
17	5	33	19	16	1	0.7392493D-02
18	11	65	18	14	1	0.2003440D 00

Table 3

SUMMARY OF 28 CALLS TO NLSQ2

NPROB	N	M	NFEV	NJEV	INFO	FINAL L2 NORM
1	5	10	3	2	1	0.2236068D 01
1	5	50	3	2	1	0.6708204D 01
2	5	10	11	10	1	0.1463850D 01
2	5	50	11	10	1	0.3482630D 01
3	5	10	13	12	1	0.1909727D 01
3	5	50	13	12	1	0.3691729D 01
4	2	2	18	14	1	0.0
5	3	3	12	9	1	0.3731651D-22
6	4	4	23	22	1	0.7212634D-12
7	2	2	17	15	1	0.6998875D 01
8	3	15	7	6	1	0.9063596D-01
9	4	11	18	15	1	0.1753584D-01
10	3	16	174	133	1	0.9377945D 01
11	6	31	10	9	1	0.4782959D-01
11	9	31	6	5	1	0.1183115D-02
11	12	31	7	6	1	0.2173104D-04
12	3	10	7	6	1	0.1804112D-15
13	2	10	17	9	1	0.1115178D 02
14	4	20	377	325	1	0.2929543D 03
15	1	8	1	1	0	0.1886238D 01
15	8	8	31	21	1	0.5930324D-01
15	9	9	10	7	1	0.1168522D-07
15	10	10	16	11	1	0.8064710D-01
16	10	10	15	9	1	0.1606452D-12
16	30	30	33	14	1	0.3021128D-10
16	40	40	8	4	1	0.1000000D 01
17	5	33	167	117	1	0.7392493D-02
18	11	65	15	13	1	0.2003440D 00

The conclusion from Tables 2 and 3 is that although the use of standard starting points reveals some differences, none of these differences are significant. This is not the case when NLSQ1 and NLSQ2 are run on the full set of starting points. These results appear in Tables 4 and 5, and the main differences are now as follows:

- (a) NLSQ1 only fails (failure is identified by the size of the final ℓ_2 norm) on problem 10 while NLSQ2 fails three times -- once on problem 5 and twice on problem 10. Moreover, for both failures on problem 10, the INFO value of NLSQ2 incorrectly claims success.
- (b) Although this information does not appear in the tables, NLSQ1 does not generate any overflows while NLSQ2 produces overflows on problem 16 with $n = 10$ and 30 . The overflows for $n = 30$ are generated by the function subroutine and occur on the first iteration; they are due to a large initial step. The overflows for $n = 10$ are generated by NLSQ2 and occur towards the middle of the iteration.
- (c) On all of the problems where NTRIES was set to 3 (problems 4, 5, 6, 7, 8, 9, 10, 11, 14, 15 with $n = 1$, 16 with $n = 10$), the differences in performance between NLSQ1 and NLSQ2 are most pronounced for the farthest starting point, and here NLSQ1 is clearly superior to NLSQ2. For the standard starting point the algorithms perform very similarly, while for the intermediate starting point NLSQ1 seems to perform slightly better than NLSQ2. These observations are also based on a detailed examination of the progress of the iteration. These results show that Tables 4 and 5 are not unduly influenced by the stopping criteria. The only exceptions occur when the problem has a continuum of solutions, and in these cases (problems 8 and 9 where the final ℓ_2 norms are 4.174... and 0.03205..., respectively), the convergence criteria of NLSQ2 are clearly inadequate.

It should now be clear that on the basis of the above testing, NLSQ1 is a better piece of software than NLSQ2. Again we point out that the development of NLSQ1 and NLSQ2 received considerable attention; had this not been the case, then our testing would have uncovered more drastic differences.

Table 4

SUMMARY OF 54 CALLS TC NLSQ1

NPROB	N	M	NFEV	NJEV	INFO	FINAL L2 NORM
1	5	10	3	2	1	0.2236068D 01
1	5	50	3	2	1	0.6708204D 01
2	5	10	3	2	1	0.1463850D 01
2	5	50	3	2	1	0.3482630D 01
3	5	10	3	2	1	0.1909727D 01
3	5	50	3	2	1	0.3691729D 01
4	2	2	18	14	1	0.0
4	2	2	8	5	1	0.0
4	2	2	6	4	1	0.1394700D-15
5	3	3	12	9	1	0.9195638D-32
5	3	3	21	16	1	0.1197349D-34
5	3	3	19	16	1	0.7062250D-29
6	4	4	68	62	1	0.9523448D-35
6	4	4	62	61	1	0.9545825D-33
6	4	4	69	65	1	0.1429468D-32
7	2	2	17	10	1	0.6998875D 01
7	2	2	22	13	1	0.6998875D 01
7	2	2	25	17	1	0.6998875D 01
8	3	15	7	6	1	0.9063596D-01
8	3	15	50	49	1	0.4174769D 01
8	3	15	28	27	1	0.4174769D 01
9	4	11	23	21	1	0.1753584D-01
9	4	11	93	85	1	0.3205219D-01
9	4	11	353	312	1	0.1753584D-01
10	3	16	136	120	1	0.9377945D 01
10	3	16	800	652	0	0.7156159D 03
10	3	16	279	245	1	0.9377945D 01
11	6	31	9	8	1	0.4782959D-01
11	6	31	15	14	1	0.4782959D-01
11	6	31	16	15	1	0.4782959D-01
11	9	31	9	8	1	0.1183115D-02
11	9	31	19	15	1	0.1183115D-02
11	9	31	18	15	1	0.1183115D-02
11	12	31	10	9	1	0.2173104D-04
11	12	31	14	12	1	0.2173104D-04
11	12	31	34	28	1	0.2173104D-04
12	3	10	8	7	1	0.7211110D-16
13	2	10	25	14	1	0.1115178D 02
14	4	20	315	282	1	0.2929543D 03
14	4	20	73	61	1	0.2929543D 03
14	4	20	328	300	1	0.2929543D 03
15	1	8	1	1	1	0.1886238D 01
15	1	8	30	29	1	0.1884248D 01
15	1	8	48	47	1	0.1884248D 01
15	8	8	44	24	1	0.5930324D-01
15	9	9	11	8	1	0.3304872D-15
15	10	10	24	14	1	0.8064710D-01
16	10	10	17	15	1	0.8987408D-15
16	10	10	13	8	1	0.1708998D-14
16	10	10	44	42	1	0.5623502D-15
16	30	30	20	15	1	0.2170133D-14
16	40	40	19	14	1	0.1254229D-12
17	5	33	19	16	1	0.7392493D-02
18	11	65	18	14	1	0.2003440D 00

Table 5

SUMMARY OF 54 CALLS TO NLSQ2

NPROB	N	M	NFEV	NJEV	INFO	FINAL L2 NORM
1	5	10	3	2	1	0.2236068D 01
1	5	50	3	2	1	0.6708204D 01
2	5	10	11	10	1	0.1463850D 01
2	5	50	11	10	1	0.3482630D 01
3	5	10	13	12	1	0.1909727D 01
3	5	50	13	12	1	0.3691729D 01
4	2	2	18	14	1	0.0
4	2	2	6	4	1	0.0
4	2	2	6	4	1	0.0
5	3	3	12	9	1	0.3731651D-22
5	3	3	34	27	1	0.2734634D-17
5	3	3	800	685	0	0.4494176D 03
6	4	4	23	22	1	0.7212634D-12
6	4	4	26	25	1	0.1126973D-11
6	4	4	29	28	1	0.1760897D-11
7	2	2	17	15	1	0.6998875D 01
7	2	2	16	14	1	0.6998875D 01
7	2	2	28	26	1	0.6998875D 01
8	3	15	7	6	1	0.9063596D-01
8	3	15	148	50	1	0.4174769D 01
8	3	15	61	6	1	0.4174769D 01
9	4	11	18	15	1	0.1753584D-01
9	4	11	122	95	1	0.3205219D-01
9	4	11	470	382	1	0.1753584D-01
10	3	16	174	133	1	0.9377945D 01
10	3	16	43	13	1	0.3765455D 05
10	3	16	16	2	1	0.6237599D 05
11	6	31	10	9	1	0.4782959D-01
11	6	31	16	15	1	0.4782959D-01
11	6	31	19	18	1	0.4782959D-01
11	9	31	6	5	1	0.1183115D-02
11	9	31	13	12	1	0.1183115D-02
11	9	31	43	31	1	0.1183115D-02
11	12	31	7	6	1	0.2173104D-04
11	12	31	36	21	1	0.2173104D-04
11	12	31	47	31	1	0.2173104D-04
12	3	10	7	6	1	0.1804112D-15
13	2	10	17	9	1	0.1115178D 02
14	4	20	377	325	1	0.2929543D 03
14	4	20	824	686	1	0.2929543D 03
14	4	20	890	760	1	0.2929543D 03
15	1	8	1	1	0	0.1886238D 01
15	1	8	29	28	1	0.1884248D 01
15	1	8	56	55	1	0.1884248D 01
15	8	8	31	21	1	0.5930324D-01
15	9	9	10	7	1	0.1168522D-07
15	10	10	16	11	1	0.8064710D-01
16	10	10	15	9	1	0.1606452D-12
16	10	10	22	18	1	0.3501853D-14
16	10	10	637	570	1	0.4630529D-10
16	30	30	33	14	1	0.3021128D-10
16	40	40	8	4	1	0.1000000D 01
17	5	33	167	117	1	0.7392493D-02
18	11	65	15	13	1	0.2003440D 00

5. Testing II

The test functions defined in Section 3 represent a basic set; in order to further test optimization software, it is desirable to modify this basic set to yield related problems. For example, consider the nonlinear least squares problem defined by a function \hat{F} which is related to a function F from the basic set by the change of scale

$$(5.1) \quad \begin{aligned} \hat{F}(x) &= \alpha F(\sum x) \\ \hat{x}_0 &= \sum^{-1} x_0 \end{aligned}$$

where α is a positive scalar and \sum is a diagonal matrix with positive entries.

A very desirable attribute of an optimization algorithm is scale invariance. This requires that for the above problems the algorithm should generate iterates which satisfy

$$\hat{x}_k = \sum^{-1} x_k, \quad k > 0.$$

If an algorithm is scale invariant, it need not perform well on a problem; however, its performance will not change with the scaling of the problem. On the other hand, the performance of a scale dependent algorithm usually deteriorates when it is applied to a badly scaled function \hat{F} .

For unconstrained minimization, the change of scale analogous to (5.1) is

$$\hat{f}(x) = \alpha f(\sum x).$$

If f comes from our basic set, the minimum of \hat{f} is still nonnegative, so it may also be worthwhile to choose β so that

$$\hat{f}(x) = \alpha f(\sum x) + \beta$$

has a negative minimum. For nonlinear equations, it is interesting to consider the more general change of scale

$$(5.2) \quad \hat{F}(x) = \sum_1 F(\sum_2 x)$$

where both \sum_1 and \sum_2 are diagonal matrices with positive entries.

It is very easy to arrange the above tests by suitable modifications of the interface function FCN. For example, for (5.1) the body of FCN would be

```

DO      J = 1,N
      Z(J) = SIGMA(J)*X(J)

IF      IFLAG = 1
      CALL SSQFCN(M,N,Z,FVEC,NPROB)
      DO      I = 1,M
      FVEC(I) = ALPHA*FVEC(I)

IF      IFLAG = 2
      CALL SSQJAC(M,N,Z,FJAC,LDFJAC,NPROB)
      DO      J = 1,N
      DO      I = 1,M
      FJAC(I,J) = ALPHA*FJAC(I,J)*SIGMA(J)

```

In the above program outline, we assume that FCN has assigned storage space to the one-dimensional arrays Z and SIGMA. The elements of SIGMA can either be generated once and passed to FCN via COMMON, or they can be generated each time FCN is called. We have found that setting

$$(5.3) \quad \text{SIGMA}(J) = 10 ** \left[\frac{5(2j-n-1)}{(n-1)} \right]$$

(if $n = 1$ no scaling is performed) is adequate for investigating the scaling properties of algorithms.

To illustrate the type of results that can be obtained, consider two subroutines for the solution of systems of nonlinear equations, NEQ1 and NEQ2. As in Section 4, we have selected these two subroutines (with names changed) from optimization libraries.

We first ran these algorithms with the standard starting points; the results are shown in Tables 6 and 7. It is not our intention to compare these results very carefully, but the following points are worthy of mention:

- (a) NEQ2 fails on problem 6 with $n = 9$ and quits near the solution of problem 2, while NEQ1 succeeds on both problems.
- (b) Problem 7 with $n = 8$ is a system of nonlinear equations with no solution, and thus both algorithms fail.
- (c) NEQ2 quits near the solution of problem 8 with $n = 40$, while NEQ1 finds a point that minimizes the sum of squares which is not a solution to the system of nonlinear equations.

Table 6: SUMMARY OF 22 CALLS TC NEQ1

NPROB	N	NFEV	INFO	FINAL L2 NORM
1	2	24	1	0.1051242D-11
2	4	32	1	0.5279897D-10
3	2	182	1	0.1151521D-09
4	4	94	1	0.3993570D-10
5	3	27	1	0.2753458D-12
6	6	95	1	0.9830624D-10
6	9	135	1	0.1307264D-10
7	5	16	1	0.2630178D-10
7	6	28	1	0.1470389D-12
7	7	23	1	0.3074985D-10
7	8	114	0	0.7483098D-01
7	9	52	1	0.6368168D-11
8	10	31	1	0.9049180D-14
8	30	74	1	0.1094541D-11
0	40	102	0	0.1000000D 01
9	10	15	1	0.1697678D-10
10	1	6	1	0.8548717D-13
10	10	15	1	0.5422021D-10
11	10	44	1	0.9272253D-10
12	10	55	1	0.1722142D-11
13	10	23	1	0.7622868D-10
14	10	33	1	0.8251833D-10

Table 7: SUMMARY OF 22 CALLS TC NEQ2

NPROB	N	NFEV	INFO	FINAL L2 NORM
1	2	24	1	0.0
2	4	89	0	0.3879041D-09
3	2	89	1	0.3630099D-10
4	4	33	1	0.3147609D-11
5	3	34	1	0.1238056D-10
6	6	42	1	0.1118730D-10
6	9	600	0	0.2094271D 00
7	5	16	1	0.1981472D-12
7	6	35	1	0.7459022D-10
7	7	28	1	0.2546015D-11
7	8	139	0	0.5933494D-01
7	9	34	1	0.4694295D-10
8	10	29	1	0.1763058D-10
8	30	184	1	0.2126396D-12
8	40	451	0	0.2813878D-04
9	10	33	1	0.8672105D-10
10	1	6	1	0.8548717D-13
10	10	16	1	0.3420128D-11
11	10	42	1	0.3280180D-10
12	10	69	1	0.8435982D-13
13	10	25	1	0.5306915D-11
14	10	34	1	0.7919650D-10

These results seem to favor NEQ1, but they are far from conclusive.

We next ran these algorithms on the scaled problem (5.2) where \sum_1 is the identity matrix and \sum_2 is chosen by (5.3); the results are shown in Tables 8 and 9. It is now clear that NEQ1 is much less susceptible to changes in scale than NEQ2 and is thus the superior routine. We might add that the tests on the full set of starting points do not change this conclusion.

To close this section we note that the routines NLSQ1 and NLSQ2 compared in Section 4 are both invariant with respect to scale changes, and thus the tests of this section would not affect their relative performance.

Table 8: SUMMARY OF 22 CALIS TC NEQ1

NPRCB	N	NFEV	INFO	FINAL L2 NORM
1	2	24	1	0.2779025D-14
2	4	32	1	0.5050454D-10
3	2	29	0	0.1014940D-03
4	4	148	1	0.2333514D-10
5	3	45	1	0.5030085D-14
6	6	41	1	0.7532181D-12
6	9	57	1	0.8618547D-12
7	5	22	1	0.8699149D-10
7	6	29	1	0.2819654D-11
7	7	30	1	0.2639084D-08
7	8	55	0	0.1495160D 00
7	9	43	0	0.1416533D 00
8	10	33	0	0.9882763D 00
8	30	101	1	0.8347604D 02
8	40	204	1	0.1000000D 01
9	10	15	1	0.3535204D-10
10	1	6	1	0.8548717D-13
10	10	16	1	0.2355356D-12
11	10	31	0	0.8411753D-01
12	10	31	0	0.2240213D 07
13	10	23	1	0.4465230D-08
14	10	29	1	0.4091723D-06

Table 9: SUMMARY OF 22 CALIS TC NEQ2

NPRCB	N	NFEV	INFO	FINAL L2 NORM
1	2	39	0	0.1977266D 01
2	4	55	0	0.8848524D 01
3	2	37	0	0.9997400D 00
4	4	56	0	0.6190943D 04
5	3	12	0	0.4975108D 01
6	6	114	0	0.6368151D 01
6	9	107	0	0.2261702D 02
7	5	54	0	0.2015743D 00
7	6	61	0	0.1675853D 00
7	7	71	0	0.2078739D 00
7	8	72	0	0.1595835D 00
7	9	77	0	0.1493451D 00
8	10	80	0	0.1142024D 01
8	30	180	0	0.1094029D 01
8	40	274	0	0.1118047D 01
9	10	66	0	0.3517726D-01
10	1	6	1	0.8548717D-13
10	10	66	0	0.2495601D 00
11	10	86	0	0.6825777D-01
12	10	53	0	0.3289782D 01
13	10	129	0	0.3500787D 01
14	10	89	0	0.1675228D 02

References

1. Bard, Y., Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems, *SIAM J. Numer. Anal.*, 7 (1970), 157-186.
2. Beale, E. M. L., On an Iterative Method of Finding a Local Minimum of a Function of More Than One Variable, Technical Report No. 25, Statistical Techniques Research Group, Princeton University.
3. Biggs, M. C., Minimization Algorithms Making Use of Non-Quadratic Properties of the Objective Function, *J. Inst. Maths Applies*, 8 (1971), 315-327.
4. Box, M. J., A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems, *The Computer Journal*, 9 (1966), 67-77.
5. Brown, K. M. and Dennis, J. E., New Computational Algorithms for Minimizing a Sum of Squares of Nonlinear Functions, Yale University, Department of Computer Science Report No. 71-6 (March 1971).
6. Brown, K. M., A Quadratically Convergent Newton-like Method Based upon Gaussian Elimination, *SIAM J. Numer. Anal.*, 6 (1969), 560-569.
7. Broyden, C. G., A Class of Methods for Solving Nonlinear Simultaneous Equations, *Math. Comp.*, 19 (1965), 577-593.
8. Broyden, C. G., The Convergence of an Algorithm for Solving Sparse Nonlinear Systems, *Math. Comp.*, 25 (1971), 285-294.
9. Colville, A. R., A Comparative Study of Nonlinear Programming Codes, IBM New York Scientific Center Report 320-2949 (1968).
10. Cox, R. A., Comparison of the Performance of Seven Optimization Algorithms on Twelve Unconstrained Optimization Problems, Pittsburgh Gulf Research and Development Company, Ref. 1335CN04 (January 1969).
11. Fletcher, R. and Powell, M. J. D., A Rapidly Convergent Descent Method for Minimization, *The Computer Journal*, 6 (1963), 163-168.
12. Fletcher, R., Function Minimization Without Evaluating Derivatives - A Review, *The Computer Journal*, 8 (1965), 33-41.
13. Freudenstein, F. and Roth, B., Numerical Solutions of Systems of Nonlinear Equations, *J. ACM*, 10 (1963), 550-556.

14. Gill, P. E., Murray, W. and Pitfield, R. A., The Implementation of Two Revised Quasi-Newton Algorithms for Unconstrained Optimization, National Physical Laboratory Report NAC 11 (April 1972), 82-83.
15. Hillstom, K. E., A Simulation Test Approach to the Evaluation of Non-linear Optimization Algorithms, *ACM Transactions on Mathematical Software*, 3 (1977), 305-315.
16. Jennrich, R. I. and Sampson, P. F., Application of Stepwise Regression to Nonlinear Estimation, *Technometrics*, 10 (1968), 63-72.
17. Kowalik, J. S. and Osborne, M. R., *Methods for Unconstrained Optimization Problems*, American Elsevier (1968).
18. Meyer, R. R., Theoretical and Computational Aspects of Nonlinear Regression, *Nonlinear Programming*, J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds., Academic Press (1970), 465-486.
19. Moré, J. J. and Cosnard, M. Y., On the Numerical Solution of Nonlinear Equations, Argonne National Laboratory, Applied Mathematics Division, TM-286 (1976).
20. Moré, J. J., The Levenberg-Marquardt Algorithm: Implementation and Theory, *Numerical Analysis*, G. A. Watson, ed., *Lecture Notes in Mathematics* 630, Springer-Verlag (1977), 105-116.
21. Osborne, M. R., Some Aspects of Nonlinear Least Squares Calculations, *Numerical Methods for Nonlinear Optimization*, F. A. Lootsma, ed., Academic Press (1972), 171-189.
22. Powell, M. J. D., A Hybrid Method for Nonlinear Equations, *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, ed., Gordon and Breach (1970), 87-114.
23. Powell, M. J. D., An Iterative Method for Finding Stationary Values of a Function of Several Variables, *The Computer Journal*, 5 (1962), 147-151.
24. Rosenbrock, H. H., An Automatic Method for Finding the Greatest or Least Value of a Function, *The Computer Journal*, 3 (1960), 175-184.
25. Spedicato, E., Computational Experience with Quasi-Newton Algorithms for Minimization Problems of Moderately Large Size (unpublished).

A P P E N D I X 1

Basic Subroutines

Basic Subroutines

```

SUBROUTINE INITPT(N,X,NPROB,FACTOR)
INTEGER N,NPROB
DOUBLE PRECISION FACTOR
DOUBLE PRECISION X(N)
*****
SUBROUTINE INITPT
THIS SUBROUTINE SPECIFIES THE STANDARD STARTING POINTS FOR
THE FUNCTIONS DEFINED BY SUBROUTINES COMFCN AND VECFCN. THE
SUBROUTINE RETURNS IN X A MULTIPLE (FACTOR) OF THE STANDARD
STARTING POINT. FOR THE SIXTH FUNCTION THE STANDARD STARTING
POINT IS ZERO, SC IN THIS CASE, IF FACTOR IS NOT UNITY, THEN
THE SUBROUTINE RETURNS THE VECTOR X(J) = FACTOR, J=1,...,N.
THE SUBROUTINE STATEMENT IS
SUBROUTINE INITPT(N,X,NPROB,FACTOR)
WHERE
N IS A POSITIVE INTEGER VARIABLE.
X IS A LINEAR ARRAY OF LENGTH N. ON OUTPUT X CONTAINS THE
STANDARD STARTING POINT FOR PROBLEM NPROB MULTIPLIED BY
FACTOR.
NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 14.
FACTOR SPECIFIES THE MULTIPLE OF THE STANDARD STARTING
POINT. IF FACTOR IS UNITY, NO MULTIPLICATION IS PERFORMED.
MINPACK. VERSION OF SEPTEMBER 1977.
BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
*****
INTEGER IVAR,J
DOUBLE PRECISION C1,H,HALF,ONE,THREE,TJ,ZERO
DOUBLE PRECISION DFLOAT
DATA ZERO,HALF,ONE,THREE,C1 /0.D0,5.D-1,1.D0,3.D0,1.2D0/
DFLOAT(IVAR) = IVAR
SELECTION OF INITIAL POINT.
GO TO (100,200,300,400,500,600,700,800,900,1000,
1 1100,1200,1300,1400),NPROB
ROSENBROCK FUNCTION.
100 CONTINUE
X(1) = -C1
X(2) = ONE
GO TO 1500
POWELL SINGULAR FUNCTION.
200 CONTINUE
X(1) = THREE

```

```

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590

```

X(2) = -ONE	00000600
X(3) = ZERC	00000610
X(4) = ONE	00000620
GO TO 1500	00000630
C	00000640
C PCWELL BADLY SCALED FUNCTION.	00000650
C	00000660
300 CONTINUE	00000670
X(1) = ZERO	00000680
X(2) = ONE	00000690
GO TO 1500	00000700
C	00000710
C WOOD FUNCTION.	00000720
C	00000730
400 CONTINUE	00000740
X(1) = -THREE	00000750
X(2) = -ONE	00000760
X(3) = -THREE	00000770
X(4) = -ONE	00000780
GO TO 1500	00000790
C	00000800
C HELICAL VALLEY FUNCTION.	00000810
C	00000820
500 CONTINUE	00000830
X(1) = -ONE	00000840
X(2) = ZERO	00000850
X(3) = ZERO	00000860
GO TO 1500	00000870
C	00000880
C WATSON FUNCTION.	00000890
C	00000900
600 CONTINUE	00000910
DO 610 J = 1, N	00000920
X(J) = ZERO	00000930
610 CONTINUE	00000940
GO TO 1500	00000950
C	00000960
C CHEBYQUAD FUNCTION.	00000970
C	00000980
700 CONTINUE	00000990
H = ONE/DFLOAT(N+1)	00001000
DO 710 J = 1, N	00001010
X(J) = DFLOAT(J)*H	00001020
710 CONTINUE	00001030
GO TO 1500	00001040
C	00001050
C BROWN ALMOST-LINEAR FUNCTION.	00001060
C	00001070
800 CONTINUE	00001080
DO 810 J = 1, N	00001090
X(J) = HALF	00001100
810 CONTINUE	00001110
GO TO 1500	00001120
C	00001130
C DISCRETE BOUNDARY VALUE AND INTEGRAL EQUATION FUNCTIONS.	00001140
C	00001150
900 CONTINUE	00001160
1000 CONTINUE	00001170
H = ONE/DFLOAT(N+1)	00001180

DO 1010 J = 1, N	00001190
TJ = DFLOAT(J) * H	00001200
X(J) = TJ * (TJ - ONE)	00001210
1010 CONTINUE	00001220
GO TO 1500	00001230
C	00001240
C TRIGONOMETRIC FUNCTION.	00001250
C	00001260
1100 CONTINUE	00001270
H = ONE/DFLOAT(N)	00001280
DO 1110 J = 1, N	00001290
X(J) = H	00001300
1110 CONTINUE	00001310
GO TO 1500	00001320
C	00001330
C VARIABLY DIMENSIONED FUNCTION.	00001340
C	00001350
1200 CONTINUE	00001360
H = ONE/DFLOAT(N)	00001370
DO 1210 J = 1, N	00001380
X(J) = ONE - DFLOAT(J) * H	00001390
1210 CONTINUE	00001400
GO TO 1500	00001410
C	00001420
C BROYDEN TRIDIAGONAL AND BANDED FUNCTIONS.	00001430
C	00001440
1300 CONTINUE	00001450
1400 CONTINUE	00001460
DO 1410 J = 1, N	00001470
X(J) = -ONE	00001480
1410 CONTINUE	00001490
C	00001500
C COMPUTE MULTIPLE OF INITIAL POINT.	00001510
C	00001520
1500 CONTINUE	00001530
IF (FACTOR .EQ. ONE) GO TO 1540	00001540
IF (NPROB .EQ. 6) GO TO 1520	00001550
DO 1510 J = 1, N	00001560
X(J) = FACTOR * X(J)	00001570
1510 CONTINUE	00001580
GO TO 1540	00001590
1520 CONTINUE	00001600
DO 1530 J = 1, N	00001610
X(J) = FACTOR	00001620
1530 CONTINUE	00001630
1540 CONTINUE	00001640
RETURN	00001650
C	00001660
C LAST CARD OF SUBROUTINE INITPT.	00001670
C	00001680
END	00001690


```

SUBROUTINE VECFCN(N,X,FVEC,NPROB)
INTEGER N,NPROB
DOUBLE PRECISION X(N),FVEC(N)
*****

```

```

SUBROUTINE VECFCN

```

```

THIS SUBROUTINE DEFINES FOURTEEN TEST FUNCTIONS. THE FIRST
FIVE TEST FUNCTIONS ARE OF DIMENSIONS 2,4,2,4,3, RESPECTIVELY,
WHILE THE REMAINING TEST FUNCTIONS ARE OF VARIABLE DIMENSION
N FOR ANY N GREATER THAN OR EQUAL TO 1 (PROBLEM 6 IS AN
EXCEPTION TO THIS, SINCE IT DOES NOT ALLOW N = 1).

```

```

THE SUBROUTINE STATEMENT IS

```

```

SUBROUTINE VECFCN(N,X,FVEC,NPROB)

```

```

WHERE

```

```

N IS A POSITIVE INTEGER VARIABLE.

```

```

X IS A LINEAR ARRAY OF LENGTH N.

```

```

FVEC IS A LINEAR ARRAY OF LENGTH N. ON OUTPUT FVEC
CONTAINS THE NPROB FUNCTION VECTOR EVALUATED AT X.

```

```

NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 14.

```

```

SUBPROGRAMS REQUIRED

```

```

FORTRAN-SUPPLIED ... DATAN,DCOS,DEXP,DSIGN,DSIN,DSQRT,
MAX0,MIN0

```

```

MINPACK. VERSION OF DECEMBER 1977.

```

```

BURTON S. GARROW, KENNETH E. HILLSPROM, JORGE J. MORE

```

```

*****

```

```

INTEGER I,IEV,IVAR,J,K,K1,K2,KP1,ML,MU

```

```

DOUBLE PRECISION C1,C2,C3,C4,C5,C6,C7,C8,C9,EIGHT,FIVE,H,

```

```

1 CNE,PROD,SUM,SUM1,SUM2,TEMP,TEMP1,TEMP2,TEN,THREE,

```

```

2 TI,TJ,TK,TPI,TWO,ZERO

```

```

DOUBLE PRECISION DFLOAT

```

```

DATA ZERO,ONE,TWO,THREE,FIVE,EIGHT,TEN

```

```

1 /0.D0,1.D0,2.D0,3.D0,5.D0,8.D0,1.D1/

```

```

DATA C1,C2,C3,C4,C5,C6,C7,C8,C9

```

```

1 /1.D4,1.0001D0,2.D2,2.02D1,1.98D1,1.8D2,2.5D-1,5.D-1,2.9D1/

```

```

DFLOAT(IVAR) = IVAR

```

```

PROBLEM SELECTOR.

```

```

GO TO (100,200,300,400,500,600,700,800,900,1000,

```

```

1 1100,1200,1300,1400),NPROB

```

```

ROSENBROCK FUNCTION.

```

```

100 CONTINUE

```

```

FVEC(1) = ONE - X(1)

```

```

FVEC(2) = TEN*(X(2) - X(1)**2)

```

```

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590

```

	GO TO 1500	00000600
C		00000610
C	POWELL SINGULAR FUNCTION.	00000620
C		00000630
200	CONTINUE	00000640
	FVEC(1) = X(1) + TEN*X(2)	00000650
	FVEC(2) = DSQRT(FIVE)*(X(3) - X(4))	00000660
	FVEC(3) = (X(2) - TWO*X(3))**2	00000670
	FVEC(4) = DSQRT(TEN)*(X(1) - X(4))**2	00000680
	GO TO 1500	00000690
C		00000700
C	POWELL BADLY SCALED FUNCTION.	00000710
C		00000720
300	CONTINUE	00000730
	FVEC(1) = C1*X(1)*X(2) - ONE	00000740
	FVEC(2) = DEXP(-X(1)) + DEXP(-X(2)) - C2	00000750
	GO TO 1500	00000760
C		00000770
C	WOOD FUNCTION.	00000780
C		00000790
400	CONTINUE	00000800
	TEMP1 = X(2) - X(1)**2	00000810
	TEMP2 = X(4) - X(3)**2	00000820
	FVEC(1) = -C3*X(1)*TEMP1 - (ONE - X(1))	00000830
	FVEC(2) = C3*TEMP1 + C4*(X(2) - ONE) + C5*(X(4) - ONE)	00000840
	FVEC(3) = -C6*X(3)*TEMP2 - (ONE - X(3))	00000850
	FVEC(4) = C6*TEMP2 + C4*(X(4) - ONE) + C5*(X(2) - ONE)	00000860
	GO TO 1500	00000870
C		00000880
C	HELICAL VALLEY FUNCTION.	00000890
C		00000900
500	CONTINUE	00000910
	TPI = EIGHT*DATAN(ONE)	00000920
	TEMP1 = DSIGN(C7,X(2))	00000930
	IF (X(1) .GT. ZERO) TEMP1 = DATAN(X(2)/X(1))/TPI	00000940
	IF (X(1) .LT. ZERO) TEMP1 = DATAN(X(2)/X(1))/TPI + C8	00000950
	TEMP2 = DSQRT(X(1)**2+X(2)**2)	00000960
	FVEC(1) = TEN*(X(3) - TEN*TEMP1)	00000970
	FVEC(2) = TEN*(TEMP2 - ONE)	00000980
	FVEC(3) = X(3)	00000990
	GO TO 1500	00001000
C		00001010
C	WATSON FUNCTION.	00001020
C		00001030
600	CONTINUE	00001040
	DO 610 K = 1, N	00001050
	FVEC(K) = ZERO	00001060
610	CONTINUE	00001070
	DO 650 I = 1, 29	00001080
	TI = DFLOAT(I)/C9	00001090
	SUM1 = ZERO	00001100
	TEMP = ONE	00001110
	DO 620 J = 2, N	00001120
	SUM1 = SUM1 + DFLOAT(J-1)*TEMP*X(J)	00001130
	TEMP = TI*TEMP	00001140
620	CONTINUE	00001150
	SUM2 = ZERO	00001160
	TEMP = ONE	00001170
	DO 630 J = 1, N	00001180

	SUM2 = SUM2 + TEMP*X(J)	00001190
	TEMP = TI*TEMP	00001200
630	CONTINUE	00001210
	TEMP1 = SUM1 - SUM2**2 - ONE	00001220
	TEMP2 = TWO*TI*SUM2	00001230
	TEMP = ONE/TI	00001240
	DO 640 K = 1, N	00001250
	FVEC(K) = FVEC(K) + TEMP*(DFLOAT(K-1) - TEMP2)*TEMP1	00001260
	TEMP = TI*TEMP	00001270
640	CONTINUE	00001280
650	CONTINUE	00001290
	TEMP = X(2) - X(1)**2 - ONE	00001300
	FVEC(1) = FVEC(1) + X(1)*(ONE - TWO*TEMP)	00001310
	FVEC(2) = FVEC(2) + TEMP	00001320
	GO TO 1500	00001330
C		00001340
C	CHEBYQUAD FUNCTION.	00001350
C		00001360
700	CONTINUE	00001370
	DO 710 K = 1, N	00001380
	FVEC(K) = ZERO	00001390
710	CONTINUE	00001400
	DO 730 J = 1, N	00001410
	TEMP1 = ONE	00001420
	TEMP2 = TWO*X(J) - ONE	00001430
	TEMP = TWO*TEMP2	00001440
	DO 720 I = 1, N	00001450
	FVEC(I) = FVEC(I) + TEMP2	00001460
	TI = TEMP*TEMP2 - TEMP1	00001470
	TEMP1 = TEMP2	00001480
	TEMP2 = TI	00001490
720	CONTINUE	00001500
730	CONTINUE	00001510
	TK = ONE/DFLOAT(N)	00001520
	IEV = -1	00001530
	DO 740 K = 1, N	00001540
	FVEC(K) = TK*FVEC(K)	00001550
	IF (IEV .GT. 0) FVEC(K) = FVEC(K) + ONE/(DFLOAT(K)**2 - ONE)	00001560
	IEV = -IEV	00001570
740	CONTINUE	00001580
	GO TO 1500	00001590
C		00001600
C	BROWN ALMOST-LINEAR FUNCTION.	00001610
C		00001620
800	CONTINUE	00001630
	SUM = -DFLOAT(N+1)	00001640
	PROD = ONE	00001650
	DO 810 J = 1, N	00001660
	SUM = SUM + X(J)	00001670
	PROD = X(J)*PROD	00001680
810	CONTINUE	00001690
	DO 820 K = 1, N	00001700
	FVEC(K) = X(K) + SUM	00001710
820	CONTINUE	00001720
	FVEC(N) = PROD - ONE	00001730
	GO TO 1500	00001740
C		00001750
C	DISCRETE BOUNDARY VALUE FUNCTION.	00001760
C		00001770

900	CONTINUE	00001780
	H = ONE/DFLOAT(N+1)	00001790
	DO 910 K = 1, N	00001800
	TEMP = (X(K) + DFLOAT(K)*H + ONE)**3	00001810
	TEMP1 = ZERO	00001820
	IF (K.NE. 1) TEMP1 = X(K-1)	00001830
	TEMP2 = ZERO	00001840
	IF (K.NE. N) TEMP2 = X(K+1)	00001850
	FVEC(K) = TWO*X(K) - TEMP1 - TEMP2 + TEMP*H**2/TWO	00001860
910	CONTINUE	00001870
	GO TO 1500	00001880
C		00001890
C	DISCRETE INTEGRAL EQUATION FUNCTION.	00001900
C		00001910
1000	CONTINUE	00001920
	H = ONE/DFLOAT(N+1)	00001930
	DO 1040 K = 1, N	00001940
	TK = DFLOAT(K)*H	00001950
	SUM1 = ZERO	00001960
	DO 1010 J = 1, K	00001970
	TJ = DFLOAT(J)*H	00001980
	TEMP = (X(J) + TJ + ONE)**3	00001990
	SUM1 = SUM1 + TJ*TEMP	00002000
1010	CONTINUE	00002010
	SUM2 = ZERO	00002020
	KP1 = K + 1	00002030
	IF (N.LT. KP1) GO TO 1030	00002040
	DO 1020 J = KP1, N	00002050
	TJ = DFLOAT(J)*H	00002060
	TEMP = (X(J) + TJ + ONE)**3	00002070
	SUM2 = SUM2 + (ONE - TJ)*TEMP	00002080
1020	CONTINUE	00002090
1030	CONTINUE	00002100
	FVEC(K) = X(K) + H*((ONE - TK)*SUM1 + TK*SUM2)/TWO	00002110
1040	CONTINUE	00002120
	GO TO 1500	00002130
C		00002140
C	TRIGONOMETRIC FUNCTION.	00002150
C		00002160
1100	CONTINUE	00002170
	SUM = ZERO	00002180
	DO 1110 J = 1, N	00002190
	FVEC(J) = DCOS(X(J))	00002200
	SUM = SUM + FVEC(J)	00002210
1110	CONTINUE	00002220
	DO 1120 K = 1, N	00002230
	FVEC(K) = DFLOAT(N+K) - DSIN(X(K)) - SUM - DFLOAT(K)*FVEC(K)	00002240
1120	CONTINUE	00002250
	GO TO 1500	00002260
C		00002270
C	VARIABLY DIMENSIONED FUNCTION.	00002280
C		00002290
1200	CONTINUE	00002300
	SUM = ZERO	00002310
	DO 1210 J = 1, N	00002320
	SUM = SUM + DFLOAT(J)*(X(J) - ONE)	00002330
1210	CONTINUE	00002340
	TEMP = SUM*(ONE + TWO*SUM**2)	00002350
	DO 1220 K = 1, N	00002360

	FVEC(K) = X(K) - ONE + DFPLOAT(K)*TEMP	00002370
1220	CONTINUE	00002380
	GO TO 1500	00002390
C		00002400
C	BROYDEN TRIDIAGONAL FUNCTION.	00002410
C		00002420
1300	CONTINUE	00002430
	DO 1310 K = 1, N	00002440
	TEMP = (THREE - TWO*X(K))*X(K)	00002450
	TEMP1 = ZERO	00002460
	IF (K.NE. 1) TEMP1 = X(K-1)	00002470
	TEMP2 = ZERO	00002480
	IF (K.NE. N) TEMP2 = X(K+1)	00002490
	FVEC(K) = TEMP - TEMP1 - TWO*TEMP2 + ONE	00002500
1310	CONTINUE	00002510
	GO TO 1500	00002520
C		00002530
C	BROYDEN BANDED FUNCTION.	00002540
C		00002550
1400	CONTINUE	00002560
	NL = 5	00002570
	MU = 1	00002580
	DO 1420 K = 1, N	00002590
	K1 = MAX0(1,K-NL)	00002600
	K2 = MIN0(K+MU,N)	00002610
	TEMP = ZERO	00002620
	DO 1410 J = K1, K2	00002630
	IF (J.EQ. K) GO TO 1410	00002640
	TEMP = TEMP + X(J)*(ONE + X(J))	00002650
1410	CONTINUE	00002660
	FVEC(K) = X(K)*(TWO + FIVE*X(K)**2) + ONE - TEMP	00002670
1420	CONTINUE	00002680
1500	CONTINUE	00002690
	RETURN	00002700
C		00002710
C	LAST CARD OF SUBROUTINE VECFCN.	00002720
C		00002730
	END	00002740

```

SUBROUTINE COMFCN(N,K,X,FCNK,NPROB)
INTEGER N,K,NPROB
DOUBLE PRECISION FCNK
DOUBLE PRECISION X(N)
*****

SUBROUTINE COMFCN

THIS SUBROUTINE DEFINES FOURTEEN TEST FUNCTIONS. THE FIRST
FIVE TEST FUNCTIONS ARE OF DIMENSIONS 2,4,2,4,3, RESPECTIVELY,
WHILE THE REMAINING TEST FUNCTIONS ARE OF VARIABLE DIMENSION
N FOR ANY N GREATER THAN OR EQUAL TO 1 (PROBLEM 6 IS AN
EXCEPTION TO THIS, SINCE IT DOES NOT ALLOW N = 1).

THE SUBROUTINE STATEMENT IS

SUBROUTINE COMFCN(N,K,X,FCNK,NPROB)

WHERE

N IS A POSITIVE INTEGER VARIABLE.

K IS A POSITIVE INTEGER VARIABLE NOT GREATER THAN N.

X IS A LINEAR ARRAY OF LENGTH N.

FCNK IS A REAL VARIABLE WHICH ON OUTPUT CONTAINS THE VALUE OF
THE K-TH COMPONENT OF THE NPROB FUNCTION EVALUATED AT X.

NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 14.

SUBPROGRAMS REQUIRED

FORTRAN-SUPPLIED ... DATAN,DCOS,DEXP,DSIGN,DSIN,DSQRT,
MAXO,MINO,MOD

MINPACK. VERSION OF SEPTEMBER 1977.
BURTON S. GARBOW,KENNETH E. HILLSTROM, JORGE J. MORE

*****
INTEGER I,IVAR,J,K1,K2,KP1,ML,MU
DOUBLE PRECISION C1,C2,C3,C4,C5,C6,C7,C8,C9,EIGHT,FIVE,H,
1 ONE,PROD,SUM,SUM1,SUM2,TEMP,TEMP1,TEMP2,TEN,THREE,
2 TI,TJ,TK,TPI,TWO,ZERO
DOUBLE PRECISION DFLOAT
DATA ZERO,ONE,TWO,THREE,FIVE,EIGHT,TEN
1 /0.D0,1.D0,2.D0,3.D0,5.D0,8.D0,1.D1/
DATA C1,C2,C3,C4,C5,C6,C7,C8,C9
1 /1.D4,1.0001D0,2.D2,2.02D1,1.98D1,1.8D2,2.5D-1,5.D-1,2.9D1/
DFLOAT(IVAR) = IVAR

PROBLEM SELECTOR.

GO TO (100,200,300,400,500,600,700,800,900,1000,
1 1100,1200,1300,1400),NPROB

ROSENBROCK FUNCTION.

```

```

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590

```

100	CONTINUE	00000600
	IF (K .EQ. 1) FCNK = ONE - X(1)	00000610
	IF (K .EQ. 2) FCNK = TEN*(X(2) - X(1)**2)	00000620
	GO TO 1500	00000630
C		00000640
C	POWELL SINGULAR FUNCTION.	00000650
C		00000660
200	CONTINUE	00000670
	IF (K .EQ. 1) FCNK = X(1) + TEN*X(2)	00000680
	IF (K .EQ. 2) FCNK = DSQRT(FIVE)*(X(3) - X(4))	00000690
	IF (K .EQ. 3) FCNK = (X(2) - TWO*X(3))**2	00000700
	IF (K .EQ. 4) FCNK = DSQRT(TEN)*(X(1) - X(4))**2	00000710
	GO TO 1500	00000720
C		00000730
C	POWELL BADLY SCALED FUNCTION.	00000740
C		00000750
300	CONTINUE	00000760
	IF (K .EQ. 1) FCNK = C1*X(1)*X(2) - ONE	00000770
	IF (K .EQ. 2) FCNK = DEXP(-X(1)) + DEXP(-X(2)) - C2	00000780
	GO TO 1500	00000790
C		00000800
C	WOOD FUNCTION.	00000810
C		00000820
400	CONTINUE	00000830
	TEMP1 = X(2) - X(1)**2	00000840
	TEMP2 = X(4) - X(3)**2	00000850
	IF (K .EQ. 1) FCNK = -C3*X(1)*TEMP1 - (ONE - X(1))	00000860
	IF (K .EQ. 2) FCNK = C3*TEMP1 + C4*(X(2) - ONE) + C5*(X(4) - ONE)	00000870
	IF (K .EQ. 3) FCNK = -C6*X(3)*TEMP2 - (ONE - X(3))	00000880
	IF (K .EQ. 4) FCNK = C6*TEMP2 + C4*(X(4) - ONE) + C5*(X(2) - ONE)	00000890
	GO TO 1500	00000900
C		00000910
C	HELICAL VALLEY FUNCTION.	00000920
C		00000930
500	CONTINUE	00000940
	IF (K .NE. 1) GO TO 510	00000950
	TPI = EIGHT*DATAN(ONE)	00000960
	TEMP1 = DSIGN(C7,X(2))	00000970
	IF (X(1) .GT. ZERO) TEMP1 = DATAN(X(2)/X(1))/TPI	00000980
	IF (X(1) .LT. ZERO) TEMP1 = DATAN(X(2)/X(1))/TPI + C8	00000990
	FCNK = TEN*(X(3) - TEN*TEMP1)	00001000
510	CONTINUE	00001010
	IF (K .EQ. 2) FCNK = TEN*(DSQRT(X(1)**2+X(2)**2) - ONE)	00001020
	IF (K .EQ. 3) FCNK = X(3)	00001030
	GO TO 1500	00001040
C		00001050
C	WATSON FUNCTION.	00001060
C		00001070
600	CONTINUE	00001080
	FCNK = ZERO	00001090
	DO 630 I = 1, 29	00001100
	TI = DFLOAT(I)/C9	00001110
	SUM1 = ZERO	00001120
	TEMP = ONE	00001130
	DO 610 J = 2, N	00001140
	SUM1 = SUM1 + DFLOAT(J-1)*TEMP*X(J)	00001150
	TEMP = TI*TEMP	00001160
610	CONTINUE	00001170
	SUM2 = ZERO	00001180

	TEMP = ONE	00001190
	DC 620 J = 1, N	00001200
	SUM2 = SUM2 + TEMP*X(J)	00001210
	TEMP = TI*TEMP	00001220
620	CONTINUE	00001230
	TEMP1 = SUM1 - SUM2**2 - ONE	00001240
	TEMP2 = TWO*TI*SUM2	00001250
	FCNK = FCNK + TI**(K-2)*(DFLOAT(K-1) - TEMP2)*TEMP1	00001260
630	CONTINUE	00001270
	TEMP = X(2) - X(1)**2 - ONE	00001280
	IF (K .EQ. 1) FCNK = FCNK + X(1)*(ONE - TWO*TEMP)	00001290
	IF (K .EQ. 2) FCNK = FCNK + TEMP	00001300
	GO TO 1500	00001310
C		00001320
C	CHEBYQUAD FUNCTION.	00001330
C		00001340
700	CONTINUE	00001350
	SUM = ZERO	00001360
	DO 730 J = 1, N	00001370
	TEMP1 = ONE	00001380
	TEMP2 = TWO*X(J) - ONE	00001390
	TEMP = TWO*TEMP2	00001400
	IF (K .LT. 2) GO TO 720	00001410
	DO 710 I = 2, K	00001420
	TI = TEMP*TEMP2 - TEMP1	00001430
	TEMP1 = TEMP2	00001440
	TEMP2 = TI	00001450
710	CONTINUE	00001460
720	CONTINUE	00001470
	SUM = SUM + TEMP2	00001480
730	CONTINUE	00001490
	FCNK = SUM/DFLOAT(N)	00001500
	IF (MOD(K,2) .EQ. 0) FCNK = FCNK + ONE/(DFLOAT(K)**2 - ONE)	00001510
	GO TO 1500	00001520
C		00001530
C	BROWN ALMOST-LINEAR FUNCTION.	00001540
C		00001550
800	CONTINUE	00001560
	IF (K .EQ. N) GO TO 820	00001570
	SUM = -DFLOAT(N+1)	00001580
	DO 810 J = 1, N	00001590
	SUM = SUM + X(J)	00001600
810	CONTINUE	00001610
	FCNK = X(K) + SUM	00001620
	GO TO 840	00001630
820	CONTINUE	00001640
	PROD = ONE	00001650
	DO 830 J = 1, N	00001660
	PROD = X(J)*PROD	00001670
830	CONTINUE	00001680
	FCNK = PROD - ONE	00001690
840	CONTINUE	00001700
	GO TO 1500	00001710
C		00001720
C	DISCRETE BOUNDARY VALUE FUNCTION.	00001730
C		00001740
900	CONTINUE	00001750
	H = ONE/DFLOAT(N+1)	00001760
	TEMP = (X(K) + DFLOAT(K)*H + ONE)**3	00001770

	TEMP1 = ZERO	00001780
	IF (K .NE. 1) TEMP1 = X(K-1)	00001790
	TEMP2 = ZERO	00001800
	IF (K .NE. N) TEMP2 = X(K+1)	00001810
	PCNK = TWO*X(K) - TEMP1 - TEMP2 + TEMP*H**2/TWO	00001820
	GO TO 1500	00001830
C		00001840
C	DISCRETE INTEGRAL EQUATION FUNCTION.	00001850
C		00001860
1000	CONTINUE	00001870
	H = ONE/DFLOAT(N+1)	00001880
	TK = DFLOAT(K)*H	00001890
	SUM1 = ZERO	00001900
	DO 1010 J = 1, K	00001910
	TJ = DFLOAT(J)*H	00001920
	TEMP = (X(J) + TJ + ONE)**3	00001930
	SUM1 = SUM1 + TJ*TEMP	00001940
1010	CONTINUE	00001950
	SUM2 = ZERO	00001960
	KP1 = K + 1	00001970
	IF (N .LT. KP1) GO TO 1030	00001980
	DO 1020 J = KP1, N	00001990
	TJ = DFLOAT(J)*H	00002000
	TEMP = (X(J) + TJ + ONE)**3	00002010
	SUM2 = SUM2 + (ONE - TJ)*TEMP	00002020
1020	CONTINUE	00002030
1030	CONTINUE	00002040
	PCNK = X(K) + H*((ONE - TK)*SUM1 + TK*SUM2)/TWO	00002050
	GO TO 1500	00002060
C		00002070
C	TRIGONOMETRIC FUNCTION.	00002080
C		00002090
1100	CONTINUE	00002100
	SUM = ZERO	00002110
	DO 1110 J = 1, N	00002120
	SUM = SUM + DCOS(X(J))	00002130
1110	CONTINUE	00002140
	PCNK = DFLOAT(N+K) - DSIN(X(K)) - SUM - DFLOAT(K)*DCOS(X(K))	00002150
	GO TO 1500	00002160
C		00002170
C	VARIABLY DIMENSIONED FUNCTION.	00002180
C		00002190
1200	CONTINUE	00002200
	SUM = ZERO	00002210
	DO 1210 J = 1, N	00002220
	SUM = SUM + DFLOAT(J)*(X(J) - ONE)	00002230
1210	CONTINUE	00002240
	TEMP = SUM*(ONE + TWO*SUM**2)	00002250
	PCNK = X(K) - ONE + DFLOAT(K)*TEMP	00002260
	GO TO 1500	00002270
C		00002280
C	BROYDEN TRIDIAGONAL FUNCTION.	00002290
C		00002300
1300	CONTINUE	00002310
	TEMP = (THREE - TWO*X(K))*X(K)	00002320
	TEMP1 = ZERO	00002330
	IF (K .NE. 1) TEMP1 = X(K-1)	00002340
	TEMP2 = ZERO	00002350
	IF (K .NE. N) TEMP2 = X(K+1)	00002360

Basic Subroutines

	FCNK = TEMP - TEMP1 - TWO*TEMP2 + ONE	00002370
	GO TO 1500	00002380
C		00002390
C	BROYDEN BANDED FUNCTION.	00002400
C		00002410
	1400 CONTINUE	00002420
	ML = 5	00002430
	MU = 1	00002440
	K1 = MAX0(1,K-ML)	00002450
	K2 = MIN0(K+MU,N)	00002460
	TEMP = ZERO	00002470
	DO 1410 J = K1, K2	00002480
	IF (J .EQ. K) GO TO 1410	00002490
	TEMP = TEMP + X(J)*(ONE + X(J))	00002500
	1410 CONTINUE	00002510
	FCNK = X(K)*(TWO + FIVE*X(K)**2) + ONE - TEMP	00002520
	1500 CONTINUE	00002530
	RETURN	00002540
C		00002550
C	LAST CARD OF SUBROUTINE COMFCN.	00002560
C		00002570
	END	00002580

```

SUBROUTINE INITPT(N,X,NPROB,FACTOR)
INTEGER N,NPROB
DOUBLE PRECISION FACTOR
DOUBLE PRECISION X(N)
*****
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590

SUBROUTINE INITPT
THIS SUBROUTINE SPECIFIES THE STANDARD STARTING POINTS FOR THE
FUNCTIONS DEFINED BY SUBROUTINE SSQFCN. THE SUBROUTINE RETURNS
IN X A MULTIPLE (FACTOR) OF THE STANDARD STARTING POINT. FOR
THE 11TH FUNCTION THE STANDARD STARTING POINT IS ZERO, SO IN
THIS CASE, IF FACTOR IS NOT UNITY, THEN THE SUBROUTINE RETURNS
THE VECTOR X(J) = FACTOR, J=1,...,N.

THE SUBROUTINE STATEMENT IS

SUBROUTINE INITPT(N,X,NPROB,FACTOR)
WHERE
N IS A POSITIVE INTEGER VARIABLE.
X IS A LINEAR ARRAY OF LENGTH N. ON OUTPUT X CONTAINS THE
STANDARD STARTING POINT FOR PROBLEM NPROB MULTIPLIED BY
FACTOR.
NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
FACTOR SPECIFIES THE MULTIPLE OF THE STANDARD STARTING
POINT. IF FACTOR IS UNITY, NO MULTIPLICATION IS PERFORMED.
MINPACK. VERSION OF OCTOBER 1977.
BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
*****
INTEGER IVAR,J
DOUBLE PRECISION C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,
1 C11,C12,C13,C14,C15,C16,C17,FIVE,H,HALF,
2 ONE,SEVEN,TEN,THREE,TWENTY,TWNTF,TWO,ZERO
DOUBLE PRECISION DFLOAT
DATA ZERO,HALF,ONE,TWO,THREE,FIVE,SEVEN,TEN,TWENTY,TWNTF
1 /0.5D0,5.D-1,1.D0,2.D0,3.D0,5.D0,7.D0,1.D1,2.D1,2.5D1/
DATA C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16,C17
1 /1.2D0,2.5D-1,3.9D-1,4.15D-1,2.D-2,4.D3,2.5D2,3.D-1,4.D-1,
2 1.5D0,1.D-2,1.3D0,6.5D-1,7.D-1,6.D-1,4.5D0,5.5D0/
DFLOAT(IVAR) = IVAR
SELECTION OF INITIAL POINT.
GC TO (100,200,300,400,500,600,700,800,900,1000,1100,
1 1200,1300,1400,1500,1600,1700,1800),NPROB
LINEAR FUNCTION - FULL RANK OR RANK 1.
100 CONTINUE
200 CONTINUE
300 CONTINUE

```

DO 310 J = 1, N	00000600
X(J) = ONE	00000610
310 CONTINUE	00000620
GO TO 1900	00000630
C	00000640
C ROSEN BROCK FUNCTION.	00000650
C	00000660
400 CONTINUE	00000670
X(1) = -C1	00000680
X(2) = ONE	00000690
GO TO 1900	00000700
C	00000710
C HELICAL VALLEY FUNCTION.	00000720
C	00000730
500 CONTINUE	00000740
X(1) = -ONE	00000750
X(2) = ZERO	00000760
X(3) = ZERO	00000770
GO TO 1900	00000780
C	00000790
C POWELL SINGULAR FUNCTION.	00000800
C	00000810
600 CONTINUE	00000820
X(1) = THREE	00000830
X(2) = -ONE	00000840
X(3) = ZERO	00000850
X(4) = ONE	00000860
GO TO 1900	00000870
C	00000880
C FREUDENSTEIN AND ROTH FUNCTION.	00000890
C	00000900
700 CONTINUE	00000910
X(1) = HALF	00000920
X(2) = -TWO	00000930
GO TO 1900	00000940
C	00000950
C BARD FUNCTION.	00000960
C	00000970
800 CONTINUE	00000980
X(1) = ONE	00000990
X(2) = ONE	00001000
X(3) = ONE	00001010
GO TO 1900	00001020
C	00001030
C KOWALIK AND OSPORNE FUNCTION.	00001040
C	00001050
900 CONTINUE	00001060
X(1) = C2	00001070
X(2) = C3	00001080
X(3) = C4	00001090
X(4) = C3	00001100
GO TO 1900	00001110
C	00001120
C MEYER FUNCTION.	00001130
C	00001140
1000 CONTINUE	00001150
X(1) = C5	00001160
X(2) = C6	00001170
X(3) = C7	00001180

C	GC TO 1900	00001190
C	WATSON FUNCTION.	00001200
C		00001210
	1100 CONTINUE	00001220
	DO 1110 J = 1, N	00001230
	X(J) = ZERO	00001240
	1110 CONTINUE	00001250
	GO TO 1900	00001260
C		00001270
C	BOX 3-DIMENSIONAL FUNCTION.	00001280
C		00001290
	1200 CONTINUE	00001300
	X(1) = ZERO	00001310
	X(2) = TEN	00001320
	X(3) = TWENTY	00001330
	GO TO 1900	00001340
C		00001350
C	JENNRICH AND SAMPSON FUNCTION.	00001360
C		00001370
	1300 CONTINUE	00001380
	X(1) = C8	00001390
	X(2) = C9	00001400
	GO TO 1900	00001410
C		00001420
C	BROWN AND DENNIS FUNCTION.	00001430
C		00001440
	1400 CONTINUE	00001450
	X(1) = TWNTF	00001460
	X(2) = FIVE	00001470
	X(3) = -FIVE	00001480
	X(4) = -ONE	00001490
	GO TO 1900	00001500
C		00001510
C	CHEBYQUAD FUNCTION.	00001520
C		00001530
	1500 CONTINUE	00001540
	H = ONE/DFLOAT(N+1)	00001550
	DO 1510 J = 1, N	00001560
	X(J) = DFLOAT(J)*H	00001570
	1510 CONTINUE	00001580
	GO TO 1900	00001590
C		00001600
C	BROWN ALMOST-LINEAR FUNCTION.	00001610
C		00001620
	1600 CONTINUE	00001630
	DO 1610 J = 1, N	00001640
	X(J) = HALF	00001650
	1610 CONTINUE	00001660
	GO TO 1900	00001670
C		00001680
C	OSBOENE 1 FUNCTION.	00001690
C		00001700
	1700 CONTINUE	00001710
	X(1) = HALF	00001720
	X(2) = C10	00001730
	X(3) = -ONE	00001740
	X(4) = C11	00001750
	X(5) = C5	00001760
		00001770

Basic Subroutines

	GO TO 1900	00001780
C		00001790
C	OSBORNE 2 FUNCTION.	00001800
C		00001810
	1800 CONTINUE	00001820
	X(1) = C12	00001830
	X(2) = C13	00001840
	X(3) = C13	00001850
	X(4) = C14	00001860
	X(5) = C15	00001870
	X(6) = THREE	00001880
	X(7) = FIVE	00001890
	X(8) = SEVEN	00001900
	X(9) = TWO	00001910
	X(10) = C16	00001920
	X(11) = C17	00001930
C		00001940
C	COMPUTE MULTIPLE OF INITIAL POINT.	00001950
C		00001960
	1900 CONTINUE	00001970
	IF (FACTOR .EQ. ONE) GO TO 1940	00001980
	IF (NPROB .EQ. 11) GO TO 1920	00001990
	DO 1910 J = 1, N	00002000
	X(J) = FACTOR*X(J)	00002010
	1910 CONTINUE	00002020
	GO TO 1940	00002030
	1920 CONTINUE	00002040
	DO 1930 J = 1, N	00002050
	X(J) = FACTOR	00002060
	1930 CONTINUE	00002070
	1940 CONTINUE	00002080
	RETURN	00002090
C		00002100
C	LAST CARD OF SUBROUTINE INITPT.	00002110
C		00002120
	END	00002130

```

SUBROUTINE SSQFCN(M,N,X,FVEC,NPROB)
INTEGER M,N,NPROB
DOUBLE PRECISION X(N),FVEC(M)
*****
SUBROUTINE SSQFCN
THIS SUBROUTINE DEFINES THE FUNCTIONS OF EIGHTEEN NONLINEAR
LEAST SQUARES PROBLEMS. THE ALLOWABLE VALUES OF (M,N) FOR
FUNCTIONS 1,2 AND 3 ARE VARIABLE BUT WITH M .GE. N.
FOR FUNCTIONS 4,5,6,7,8,9 AND 10 THE VALUES OF (M,N) ARE
(2,2), (3,3), (4,4), (2,2), (15,3), (11,4) AND (16,3), RESPECTIVELY.
FUNCTION 11 (WATSON) HAS M = 31 WITH N USUALLY 6 OR 9.
HOWEVER, ANY N, N = 2,...,31, IS PERMITTED.
FUNCTIONS 12,13 AND 14 HAVE N = 3,2 AND 4, RESPECTIVELY, BUT
ALLOW ANY M .GE. N, WITH THE USUAL CHOICES BEING 10,10 AND 20.
FUNCTION 15 (CHEBYQUAD) ALLOWS M AND N VARIABLE WITH M .GE. N.
FUNCTION 16 (BROWN) ALLOWS N VARIABLE WITH M = N.
FOR FUNCTIONS 17 AND 18, THE VALUES OF (M,N) ARE
(33,5) AND (65,11), RESPECTIVELY.
THE SUBROUTINE STATEMENT IS
SUBROUTINE SSQFCN(M,N,X,FVEC,NPROB)
WHERE
M AND N ARE POSITIVE INTEGER VARIABLES. N MUST NOT EXCEED M.
X IS A LINEAR ARRAY OF LENGTH N.
FVEC IS A LINEAR ARRAY OF LENGTH M. ON OUTPUT FVEC
CONTAINS THE NPROB FUNCTION EVALUATED AT X.
NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
SUBPROGRAMS REQUIRED
FORTRAN-SUPPLIED ... DATAN,DCOS,DEXP,DSIN,DSQRT,DSIGN
MINPACK. VERSION OF OCTOBER 1977.
BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
*****
INTEGER I,IEV,IVAR,J,NM1
DOUBLE PRECISION C13,C14,C29,C45,DIV,DT,EIGHT,FIVE,ONE,
1 PROD,SUM,S1,S2,TEMP,TEN,TI,TMP1,TMP2,TMP3,TMP4,
2 TPI,TWO,ZERO,ZP25,ZP5
DOUBLE PRECISION V(11),Y1(15),Y2(11),Y3(16),Y4(33),Y5(65)
DOUBLE PRECISION DFLOAT
DATA ZERO,ZP25,ZP5,ONE,TWO,FIVE,EIGHT,TEN,C13,C14,C29,C45
1 /0.D0,2.5D-1,5.D-1,1.D0,2.D0,5.D0,8.D0,1.D1,
2 1.3D1,1.4D1,2.9D1,4.5D1/
DATA V(1),V(2),V(3),V(4),V(5),V(6),V(7),V(8),V(9),V(10),V(11)
1 /4.0D0,2.0D0,1.0D0,5.D-1,2.5D-1,1.67D-1,1.25D-1,1.D-1,
2 8.33D-2,7.14D-2,6.25D-2/
DATA Y1(1),Y1(2),Y1(3),Y1(4),Y1(5),Y1(6),Y1(7),Y1(8),
1 Y1(9),Y1(10),Y1(11),Y1(12),Y1(13),Y1(14),Y1(15)

```

```

2      /1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,      00000600
3      3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.10D0,4.39D0/      00000610
DATA Y2(1),Y2(2),Y2(3),Y2(4),Y2(5),Y2(6),      00000620
1      Y2(7),Y2(8),Y2(9),Y2(10),Y2(11)      00000630
2      /1.957D-1,1.947D-1,1.735D-1,1.600D-1,8.44D-2,6.27D-2,      00000640
3      4.56D-2,3.42D-2,3.23D-2,2.35D-2,2.46D-2/      00000650
DATA Y3(1),Y3(2),Y3(3),Y3(4),Y3(5),Y3(6),Y3(7),Y3(8),Y3(9),      00000660
1      Y3(10),Y3(11),Y3(12),Y3(13),Y3(14),Y3(15),Y3(16)      00000670
2      /3.478D4,2.861D4,2.365D4,1.963D4,1.637D4,1.372D4,1.154D4,      00000680
3      9.744D3,8.261D3,7.03D3,6.005D3,5.147D3,4.427D3,3.82D3,      00000690
4      3.307D3,2.872D3/      00000700
DATA Y4(1),Y4(2),Y4(3),Y4(4),Y4(5),Y4(6),Y4(7),Y4(8),Y4(9),      00000710
1      Y4(10),Y4(11),Y4(12),Y4(13),Y4(14),Y4(15),Y4(16),Y4(17),      00000720
2      Y4(18),Y4(19),Y4(20),Y4(21),Y4(22),Y4(23),Y4(24),Y4(25),      00000730
3      Y4(26),Y4(27),Y4(28),Y4(29),Y4(30),Y4(31),Y4(32),Y4(33)      00000740
4      /8.44D-1,9.08D-1,9.32D-1,9.36D-1,9.25D-1,9.08D-1,8.81D-1,      00000750
5      8.50D-1,8.18D-1,7.84D-1,7.51D-1,7.18D-1,6.85D-1,6.58D-1,      00000760
6      6.28D-1,6.03D-1,5.80D-1,5.58D-1,5.38D-1,5.22D-1,      00000770
7      5.06D-1,4.90D-1,4.78D-1,4.67D-1,4.57D-1,4.48D-1,4.38D-1,      00000780
8      4.31D-1,4.24D-1,4.20D-1,4.14D-1,4.11D-1,4.06D-1/      00000790
DATA Y5(1),Y5(2),Y5(3),Y5(4),Y5(5),Y5(6),Y5(7),Y5(8),Y5(9),      00000800
1      Y5(10),Y5(11),Y5(12),Y5(13),Y5(14),Y5(15),Y5(16),Y5(17),      00000810
2      Y5(18),Y5(19),Y5(20),Y5(21),Y5(22),Y5(23),Y5(24),Y5(25),      00000820
3      Y5(26),Y5(27),Y5(28),Y5(29),Y5(30),Y5(31),Y5(32),Y5(33),      00000830
4      Y5(34),Y5(35),Y5(36),Y5(37),Y5(38),Y5(39),Y5(40),Y5(41),      00000840
5      Y5(42),Y5(43),Y5(44),Y5(45),Y5(46),Y5(47),Y5(48),Y5(49),      00000850
6      Y5(50),Y5(51),Y5(52),Y5(53),Y5(54),Y5(55),Y5(56),Y5(57),      00000860
7      Y5(58),Y5(59),Y5(60),Y5(61),Y5(62),Y5(63),Y5(64),Y5(65)      00000870
8      /1.366D0,1.191D0,1.112D0,1.013D0,9.91D-1,8.85D-1,      00000880
9      8.31D-1,8.47D-1,7.86D-1,7.25D-1,7.46D-1,6.79D-1,6.08D-1,      00000890
A      6.55D-1,6.16D-1,6.06D-1,6.02D-1,6.26D-1,6.51D-1,7.24D-1,      00000900
B      6.49D-1,6.49D-1,6.94D-1,6.44D-1,5.24D-1,6.61D-1,6.12D-1,      00000910
C      5.58D-1,5.33D-1,4.95D-1,5.00D-1,4.23D-1,3.95D-1,3.75D-1,      00000920
D      3.72D-1,3.91D-1,3.96D-1,4.05D-1,4.28D-1,4.29D-1,5.23D-1,      00000930
E      5.62D-1,6.07D-1,6.53D-1,6.72D-1,7.08D-1,6.33D-1,6.68D-1,      00000940
F      6.45D-1,6.32D-1,5.91D-1,5.59D-1,5.97D-1,6.25D-1,7.39D-1,      00000950
G      7.10D-1,7.29D-1,7.20D-1,6.36D-1,5.81D-1,4.28D-1,2.92D-1,      00000960
H      1.62D-1,9.8D-2,5.4D-2/      00000970
DFLOAT(IVAR) = IVAR      00000980
C      FUNCTION ROUTINE SELECTOR.      00000990
C      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,      00001000
C      1200,1300,1400,1500,1600,1700,1800),NPROB      00001010
C      LINEAR FUNCTION - FULL RANK.      00001020
C      100 CONTINUE      00001030
C      SUM = ZERO      00001040
C      DO 110 J = 1, N      00001050
C      SUM = SUM + X(J)      00001060
C      110 CONTINUE      00001070
C      TEMP = TWO*SUM/DFLOAT(N) + ONE      00001080
C      DO 120 I = 1, N      00001090
C      FVEC(I) = -TEMP      00001100
C      IF (I .LE. N) FVEC(I) = FVEC(I) + X(I)      00001110
C      120 CONTINUE      00001120
C      GO TO 1900      00001130
C      00001140
C      00001150
C      00001160
C      00001170
C      00001180

```


C	LINEAR FUNCTION - RANK 1.	00001190
C		00001200
200	CONTINUE	00001210
	SUM = ZERO	00001220
	DO 210 J = 1, N	00001230
	SUM = SUM + DFLOAT(J)*X(J)	00001240
210	CONTINUE	00001250
	DO 220 I = 1, M	00001260
	FVEC(I) = DFLOAT(I)*SUM - ONE	00001270
220	CONTINUE	00001280
	GO TO 1900	00001290
C		00001300
C	LINEAR FUNCTION - RANK 1 WITH ZERO COLUMNS AND ROWS.	00001310
C		00001320
300	CONTINUE	00001330
	SUM = ZERO	00001340
	NM1 = N - 1	00001350
	IF (NM1 .LT. 2) GO TO 320	00001360
	DO 310 J = 2, NM1	00001370
	SUM = SUM + DFLOAT(J)*X(J)	00001380
310	CONTINUE	00001390
320	CONTINUE	00001400
	DO 330 I = 1, M	00001410
	FVEC(I) = DFLOAT(I-1)*SUM - ONE	00001420
330	CONTINUE	00001430
	FVEC(M) = -ONE	00001440
	GO TO 1900	00001450
C		00001460
C	ROSENBROCK FUNCTION.	00001470
C		00001480
400	CONTINUE	00001490
	FVEC(1) = TEN*(X(2) - X(1)**2)	00001500
	FVEC(2) = ONE - X(1)	00001510
	GO TO 1900	00001520
C		00001530
C	HELICAL VALLEY FUNCTION.	00001540
C		00001550
500	CONTINUE	00001560
	TPI = EIGHT*DATAN(ONE)	00001570
	TMP1 = DSIGN(ZP25,X(2))	00001580
	IF (X(1) .GT. ZERO) TMP1 = DATAN(X(2)/X(1))/TPI	00001590
	IF (X(1) .LT. ZERO) TMP1 = DATAN(X(2)/X(1))/TPI + ZP5	00001600
	TMP2 = DSQRT(X(1)**2+X(2)**2)	00001610
	FVEC(1) = TEN*(X(3) - TEN*TMP1)	00001620
	FVEC(2) = TEN*(TMP2 - ONE)	00001630
	FVEC(3) = X(3)	00001640
	GO TO 1900	00001650
C		00001660
C	POWELL SINGULAR FUNCTION.	00001670
C		00001680
600	CONTINUE	00001690
	FVEC(1) = X(1) + TEN*X(2)	00001700
	FVEC(2) = DSQRT(FIVE)*(X(3) - X(4))	00001710
	FVEC(3) = (X(2) - TWO*X(3))**2	00001720
	FVEC(4) = DSQRT(TEN)*(X(1) - X(4))**2	00001730
	GO TO 1900	00001740
C		00001750
C	FREUDENSTEIN AND ROTH FUNCTION.	00001760
C		00001770

Basic Subroutines

700	CONTINUE	00001780
	FVEC(1) = -C13 + X(1) + ((FIVE - X(2))*X(2) - TWO)*X(2)	00001790
	FVEC(2) = -C29 + X(1) + ((ONE + X(2))*X(2) - C14)*X(2)	00001800
	GO TO 1900	00001810
C		00001820
C	BARD FUNCTION.	00001830
C		00001840
800	CONTINUE	00001850
	DO 810 I = 1, 15	00001860
	TMP1 = DFLOAT(I)	00001870
	TMP2 = DFLOAT(16-I)	00001880
	TMP3 = TMP1	00001890
	IF (I .GT. 8) TMP3 = TMP2	00001900
	FVEC(I) = Y1(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))	00001910
810	CONTINUE	00001920
	GO TO 1900	00001930
C		00001940
C	KOWALIK AND OSBOFNE FUNCTION.	00001950
C		00001960
900	CONTINUE	00001970
	DO 910 I = 1, 11	00001980
	TMP1 = V(I)*(V(I) + X(2))	00001990
	TMP2 = V(I)*(V(I) + X(3)) + X(4)	00002000
	FVEC(I) = Y2(I) - X(1)*TMP1/TMP2	00002010
910	CONTINUE	00002020
	GO TO 1900	00002030
C		00002040
C	MEYER FUNCTION.	00002050
C		00002060
1000	CONTINUE	00002070
	DO 1010 I = 1, 15	00002080
	TEMP = FIVE*DFLOAT(I) + C45 + X(3)	00002090
	TMP1 = X(2)/TEMP	00002100
	TMP2 = DEXP(TMP1)	00002110
	FVEC(I) = X(1)*TMP2 - Y3(I)	00002120
1010	CONTINUE	00002130
	GO TO 1900	00002140
C		00002150
C	WATSON FUNCTION.	00002160
C		00002170
1100	CONTINUE	00002180
	DO 1130 I = 1, 29	00002190
	DIV = DFLOAT(I)/C29	00002200
	S1 = ZERO	00002210
	DX = ONE	00002220
	DO 1110 J = 2, N	00002230
	S1 = S1 + DFLOAT(J-1)*DX*X(J)	00002240
	DX = DIV*DX	00002250
1110	CONTINUE	00002260
	S2 = ZERO	00002270
	DX = ONE	00002280
	DO 1120 J = 1, N	00002290
	S2 = S2 + DX*X(J)	00002300
	DX = DIV*DX	00002310
1120	CONTINUE	00002320
	FVEC(I) = S1 - S2**2 - ONE	00002330
1130	CONTINUE	00002340
	FVEC(30) = X(1)	00002350
	FVEC(31) = X(2) - X(1)**2 - ONE	00002360

GO TO 1900	00002370
C	00002380
C BOX 3-DIMENSIONAL FUNCTION.	00002390
C	00002400
1200 CONTINUE	00002410
DO 1210 I = 1, M	00002420
TEMP = DFLOAT(I)	00002430
TMP1 = TEMP/TEN	00002440
FVEC(I) = DEXP(-TMP1*X(1)) - DEXP(-TMP1*X(2))	00002450
1 + (DEXP(-TEMP) - DEXP(-TMP1))*X(3)	00002460
1210 CONTINUE	00002470
GO TO 1900	00002480
C	00002490
C JENNRICH AND SAMPSON FUNCTION.	00002500
C	00002510
1300 CONTINUE	00002520
DO 1310 I = 1, M	00002530
TEMP = DFLOAT(I)	00002540
FVEC(I) = TWO + TWO*TEMP - DEXP(TEMP*X(1)) - DEXP(TEMP*X(2))	00002550
1310 CONTINUE	00002560
GO TO 1900	00002570
C	00002580
C BROWN AND DENNIS FUNCTION.	00002590
C	00002600
1400 CONTINUE	00002610
DO 1410 I = 1, M	00002620
TEMP = DFLOAT(I)/FIVE	00002630
TMP1 = X(1) + TEMP*X(2) - DEXP(TEMP)	00002640
TMP2 = X(3) + DSIN(TEMP)*X(4) - DCOS(TEMP)	00002650
FVEC(I) = TMP1**2 + TMP2**2	00002660
1410 CONTINUE	00002670
GO TO 1900	00002680
C	00002690
C CHEBYQUAD FUNCTION.	00002700
C	00002710
1500 CONTINUE	00002720
DO 1510 I = 1, M	00002730
FVEC(I) = ZERO	00002740
1510 CONTINUE	00002750
DO 1530 J = 1, N	00002760
TMP1 = ONE	00002770
TMP2 = TWO*X(J) - ONE	00002780
TEMP = TWO*TMP2	00002790
DO 1520 I = 1, M	00002800
FVEC(I) = FVEC(I) + TMP2	00002810
TI = TEMP*TMP2 - TMP1	00002820
TMP1 = TMP2	00002830
TMP2 = TI	00002840
1520 CONTINUE	00002850
1530 CONTINUE	00002860
DX = ONE/DFLOAT(N)	00002870
IEV = -1	00002880
DO 1540 I = 1, M	00002890
FVEC(I) = DX*FVEC(I)	00002900
IF (IEV .GT. 0) FVEC(I) = FVEC(I) + ONE/(DFLOAT(I)**2 - ONE)	00002910
IEV = -IEV	00002920
1540 CONTINUE	00002930
GO TO 1900	00002940
C	00002950

Basic Subroutines

C	BROWN ALMOST-LINEAR FUNCTION.	00002960
C		00002970
1600	CONTINUE	00002980
	SUM = -DFLOAT(N+1)	00002990
	PROD = ONE	00003000
	DO 1610 J = 1, N	00003010
	SUM = SUM + X(J)	00003020
	PROD = X(J)*PROD	00003030
1610	CONTINUE	00003040
	DO 1620 I = 1, N	00003050
	FVEC(I) = X(I) + SUM	00003060
1620	CONTINUE	00003070
	FVEC(N) = PROD - ONE	00003080
	GO TO 1900	00003090
C		00003100
C	OSBORNE 1 FUNCTION.	00003110
C		00003120
1700	CONTINUE	00003130
	DO 1710 I = 1, 33	00003140
	TEMP = TEN*DFLOAT(I-1)	00003150
	TMP1 = DEXP(-X(4)*TEMP)	00003160
	TMP2 = DEXP(-X(5)*TEMP)	00003170
	FVEC(I) = Y4(I) - (X(1) + X(2)*TMP1 + X(3)*TMP2)	00003180
1710	CONTINUE	00003190
	GO TO 1900	00003200
C		00003210
C	OSBORNE 2 FUNCTION.	00003220
C		00003230
1800	CONTINUE	00003240
	DO 1810 I = 1, 65	00003250
	TEMP = DFLOAT(I-1)/TEN	00003260
	TMP1 = DEXP(-X(5)*TEMP)	00003270
	TMP2 = DEXP(-X(6)*(TEMP - X(9))**2)	00003280
	TMP3 = DEXP(-X(7)*(TEMP - X(10))**2)	00003290
	TMP4 = DEXP(-X(8)*(TEMP - X(11))**2)	00003300
	FVEC(I) = Y5(I) - (X(1)*TMP1 + X(2)*TMP2	00003310
	+ X(3)*TMP3 + X(4)*TMP4)	00003320
1810	CONTINUE	00003330
1900	CONTINUE	00003340
	RETURN	00003350
C		00003360
C	LAST CARD OF SUBROUTINE SSQFCN.	00003370
C		00003380
	END	00003390

```

SUBROUTINE SSQJAC(M,N,X,FJAC,LDFJAC,NPROB)
INTEGER M,N,LDFJAC,NPROB
DOUBLE PRECISION X(N),FJAC(LDFJAC,N)
*****
SUBROUTINE SSQJAC
THIS SUBROUTINE DEFINES THE JACOBIAN MATRICES OF EIGHTEEN
NONLINEAR LEAST SQUARES PROBLEMS. THE PROBLEM DIMENSIONS ARE
AS DESCRIBED IN THE PROLOGUE COMMENTS OF SSQPCN.
THE SUBROUTINE STATEMENT IS
    SUBROUTINE SSQJAC(M,N,X,FJAC,LDFJAC,NPROB)
WHERE
    M AND N ARE POSITIVE INTEGER VARIABLES. N MUST NOT EXCEED M.
    X IS A LINEAR ARRAY OF LENGTH N.
    FJAC IS AN M BY N ARRAY. ON OUTPUT FJAC CONTAINS THE
        JACOBIAN MATRIX OF THE NPROB FUNCTION EVALUATED AT X.
    LDFJAC IS A POSITIVE INTEGER VARIABLE NOT LESS THAN M
        WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
    NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
        NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
SUBPROGRAMS REQUIRED
    FORTRAN-SUPPLIED ... DATAN,DCOS,DEXP,DSIN,DSQRT
MINPACK. VERSION OF OCTOBER 1977.
BURTON S. GARROW, KENNETH E. HILLSTROM, JORGE J. MCRE
*****
INTEGER I,IVAR,J,K,MM1,NM1
DOUBLE PRECISION C14,C20,C29,C45,C100,DIV,DX,EIGHT,FIVE,FOUR,
1     ONE,PROD,S2,TEMP,TEN,THREE,TI,TMP1,TMP2,TMP3,TMP4,
2     TPI,TWO,ZERO
DOUBLE PRECISION V(11)
DOUBLE PRECISION DFLOAT
DATA ZERO,ONE,TWO,THREE,FOUR,FIVE,EIGHT,TEN,C14,C20,C29,C45,C100
1     /0.00,1.00,2.00,3.00,4.00,5.00,8.00,1.01,
2     1.401,2.01,2.901,4.501,1.02/
DATA V(1),V(2),V(3),V(4),V(5),V(6),V(7),V(8),V(9),V(10),V(11)
1     /4.000,2.000,1.000,5.0-1,2.50-1,1.670-1,1.250-1,1.0-1,
2     8.330-2,7.140-2,6.250-2/
DFLOAT(IVAR) = IVAR
JACOBIAN ROUTINE SELECTOR.
GO TO (100,200,300,400,500,600,700,800,900,1000,1100,
1     1200,1300,1400,1500,1600,1700,1800),NPROB
LINEAR FUNCTION - FULL RANK.

```

100	CONTINUE	00000600
	TEMP = TWO/DFLOAT(M)	00000610
	DO 120 J = 1, N	00000620
	DO 110 I = 1, M	00000630
	FJAC(I,J) = -TEMP	00000640
110	CONTINUE	00000650
	FJAC(J,J) = FJAC(J,J) + ONE	00000660
120	CONTINUE	00000670
	GO TO 1900	00000680
C		00000690
C	LINEAR FUNCTION - RANK 1.	00000700
C		00000710
200	CONTINUE	00000720
	DO 220 J = 1, N	00000730
	DO 210 I = 1, M	00000740
	FJAC(I,J) = DFLOAT(I)*DFLOAT(J)	00000750
210	CONTINUE	00000760
220	CONTINUE	00000770
	GO TO 1900	00000780
C		00000790
C	LINEAR FUNCTION - RANK 1 WITH ZERO COLUMNS AND ROWS.	00000800
C		00000810
300	CONTINUE	00000820
	DO 320 J = 1, N	00000830
	DO 310 I = 1, M	00000840
	FJAC(I,J) = ZERO	00000850
310	CONTINUE	00000860
320	CONTINUE	00000870
	NM1 = N - 1	00000880
	MM1 = M - 1	00000890
	IF (NM1 .LT. 2) GO TO 350	00000900
	DO 340 J = 2, NM1	00000910
	DO 330 I = 2, MM1	00000920
	FJAC(I,J) = DFLOAT(I-1)*DFLOAT(J)	00000930
330	CONTINUE	00000940
340	CONTINUE	00000950
350	CONTINUE	00000960
	GO TO 1900	00000970
C		00000980
C	ROSENBROCK FUNCTION.	00000990
C		00001000
400	CONTINUE	00001010
	FJAC(1,1) = -C20*X(1)	00001020
	FJAC(1,2) = TEN	00001030
	FJAC(2,1) = -ONE	00001040
	FJAC(2,2) = ZERO	00001050
	GO TO 1900	00001060
C		00001070
C	HELICAL VALLEY FUNCTION.	00001080
C		00001090
500	CONTINUE	00001100
	TPI = EIGHT*DATAN(ONE)	00001110
	TEMP = X(1)**2 + X(2)**2	00001120
	TMP1 = TPI*TEMP	00001130
	TMP2 = DSQRT(TEMP)	00001140
	FJAC(1,1) = C100*X(2)/TMP1	00001150
	FJAC(1,2) = -C100*X(1)/TMP1	00001160
	FJAC(1,3) = TEN	00001170
	FJAC(2,1) = TEN*X(1)/TMP2	00001180

	FJAC(2,2) = TEN*X(2)/TMP2	00001190
	FJAC(2,3) = ZERO	00001200
	FJAC(3,1) = ZERO	00001210
	FJAC(3,2) = ZERO	00001220
	FJAC(3,3) = ONE	00001230
	GO TO 1900	00001240
C		00001250
C	POWELL SINGULAR FUNCTION.	00001260
C		00001270
600	CONTINUE	00001280
	DO 620 J = 1, 4	00001290
	DO 610 I = 1, 4	00001300
	FJAC(I,J) = ZERO	00001310
610	CONTINUE	00001320
620	CONTINUE	00001330
	FJAC(1,1) = ONE	00001340
	FJAC(1,2) = TEN	00001350
	FJAC(2,3) = DSQRT(FIVE)	00001360
	FJAC(2,4) = -FJAC(2,3)	00001370
	FJAC(3,2) = TWO*(X(2) - TWO*X(3))	00001380
	FJAC(3,3) = -TWO*FJAC(3,2)	00001390
	FJAC(4,1) = TWO*DSQRT(TEN)*(X(1) - X(4))	00001400
	FJAC(4,4) = -FJAC(4,1)	00001410
	GO TO 1900	00001420
C		00001430
C	FREUDENSTEIN AND ROTH FUNCTION.	00001440
C		00001450
700	CONTINUE	00001460
	FJAC(1,1) = ONE	00001470
	FJAC(1,2) = X(2)*(TEN - THREE*X(2)) - TWO	00001480
	FJAC(2,1) = ONE	00001490
	FJAC(2,2) = X(2)*(TWO + THREE*X(2)) - C14	00001500
	GO TO 1900	00001510
C		00001520
C	BARD FUNCTION.	00001530
C		00001540
800	CONTINUE	00001550
	DO 810 I = 1, 15	00001560
	TMP1 = DFLOAT(I)	00001570
	TMP2 = DFLOAT(16-I)	00001580
	TMP3 = TMP1	00001590
	IF (I .GT. 8) TMP3 = TMP2	00001600
	TMP4 = (X(2)*TMP2 + X(3)*TMP3)**2	00001610
	FJAC(I,1) = -ONE	00001620
	FJAC(I,2) = TMP1*TMP2/TMP4	00001630
	FJAC(I,3) = TMP1*TMP3/TMP4	00001640
810	CONTINUE	00001650
	GO TO 1900	00001660
C		00001670
C	KOWALIK AND OSBORNE FUNCTION.	00001680
C		00001690
900	CONTINUE	00001700
	DO 910 I = 1, 11	00001710
	TMP1 = V(I)*(V(I) + X(2))	00001720
	TMP2 = V(I)*(V(I) + X(3)) + X(4)	00001730
	FJAC(I,1) = -TMP1/TMP2	00001740
	FJAC(I,2) = -V(I)*X(1)/TMP2	00001750
	FJAC(I,3) = FJAC(I,1)*FJAC(I,2)	00001760
	FJAC(I,4) = FJAC(I,3)/V(I)	00001770

910	CONTINUE	00001780
	GO TO 1900	00001790
C		00001800
C	MEYER FUNCTION.	00001810
C		00001820
1000	CONTINUE	00001830
	DO 1010 I = 1, 16	00001840
	TEMP = FIVE*DFLOAT(I) + C45 + X(3)	00001850
	TMP1 = X(2)/TEMP	00001860
	TMP2 = DEXP(TMP1)	00001870
	FJAC(I,1) = TMP2	00001880
	FJAC(I,2) = X(1)*TMP2/TEMP	00001890
	FJAC(I,3) = -TMP1*FJAC(I,2)	00001900
1010	CONTINUE	00001910
	GO TO 1900	00001920
C		00001930
C	WATSON FUNCTION.	00001940
C		00001950
1100	CONTINUE	00001960
	DO 1130 I = 1, 29	00001970
	DIV = DFLOAT(I)/C29	00001980
	S2 = ZERO	00001990
	DX = ONE	00002000
	DO 1110 J = 1, N	00002010
	S2 = S2 + DX*X(J)	00002020
	DX = DIV*DX	00002030
1110	CONTINUE	00002040
	TEMP = TWO*DIV*S2	00002050
	DX = ONE/DIV	00002060
	DO 1120 J = 1, N	00002070
	FJAC(I,J) = DX*(DFLOAT(J-1) - TEMP)	00002080
	DX = DIV*DX	00002090
1120	CONTINUE	00002100
1130	CONTINUE	00002110
	DO 1150 J = 1, N	00002120
	DO 1140 I = 30, 31	00002130
	FJAC(I,J) = ZERO	00002140
1140	CONTINUE	00002150
1150	CONTINUE	00002160
	FJAC(30,1) = ONE	00002170
	FJAC(31,1) = -TWO*X(1)	00002180
	FJAC(31,2) = ONE	00002190
	GO TO 1900	00002200
C		00002210
C	BOX 3-DIMENSIONAL FUNCTION.	00002220
C		00002230
1200	CONTINUE	00002240
	DO 1210 I = 1, M	00002250
	TEMP = DFLOAT(I)	00002260
	TMP1 = TEMP/TEN	00002270
	FJAC(I,1) = -TMP1*DEXP(-TMP1*X(1))	00002280
	FJAC(I,2) = TMP1*DEXP(-TMP1*X(2))	00002290
	FJAC(I,3) = DEXP(-TEMP) - DEXP(-TMP1)	00002300
1210	CONTINUE	00002310
	GO TO 1900	00002320
C		00002330
C	JENNRICH AND SAMPSON FUNCTION.	00002340
C		00002350
1300	CONTINUE	00002360

DO 1310 I = 1, M	00002370
TEMP = DFLOAT(I)	00002380
FJAC(I,1) = -TEMP*DEXP(TEMP*X(1))	00002390
FJAC(I,2) = -TEMP*DEXP(TEMP*X(2))	00002400
1310 CONTINUE	00002410
GO TO 1900	00002420
C	00002430
C BROWN AND DENNIS FUNCTION.	00002440
C	00002450
1400 CONTINUE	00002460
DO 1410 I = 1, M	00002470
TEMP = DFLOAT(I)/FIVE	00002480
TI = DSIN(TEMP)	00002490
TMP1 = X(1) + TEMP*X(2) - DEXP(TEMP)	00002500
TMP2 = X(3) + TI*X(4) - DCOS(TEMP)	00002510
FJAC(I,1) = TWO*TMP1	00002520
FJAC(I,2) = TEMP*FJAC(I,1)	00002530
FJAC(I,3) = TWO*TMP2	00002540
FJAC(I,4) = TI*FJAC(I,3)	00002550
1410 CONTINUE	00002560
GO TO 1900	00002570
C	00002580
C CHEBYQUAD FUNCTION.	00002590
C	00002600
1500 CONTINUE	00002610
DX = ONE/DFLOAT(N)	00002620
DO 1520 J = 1, N	00002630
TMP1 = ONE	00002640
TMP2 = TWO*X(J) - ONE	00002650
TEMP = TWO*TMP2	00002660
TMP3 = ZERO	00002670
TMP4 = TWO	00002680
DO 1510 I = 1, M	00002690
FJAC(I,J) = DX*TMP4	00002700
TI = FOUR*TMP2 + TEMP*TMP4 - TMP3	00002710
TMP3 = TMP4	00002720
TMP4 = TI	00002730
TI = TEMP*TMP2 - TMP1	00002740
TMP1 = TMP2	00002750
TMP2 = TI	00002760
1510 CONTINUE	00002770
1520 CONTINUE	00002780
GO TO 1900	00002790
C	00002800
C BROWN ALMOST-LINEAR FUNCTION.	00002810
C	00002820
1600 CONTINUE	00002830
PROD = ONE	00002840
DO 1620 J = 1, N	00002850
PROD = X(J)*PROD	00002860
DO 1610 I = 1, N	00002870
FJAC(I,J) = ONE	00002880
1610 CONTINUE	00002890
FJAC(J,J) = TWO	00002900
1620 CONTINUE	00002910
DO 1650 J = 1, N	00002920
TEMP = X(J)	00002930
IF (TEMP.NE. ZERO) GO TO 1640	00002940
TEMP = ONE	00002950

	PROD = ONE	00002960
	DO 1630 K = 1, N	00002970
	IF (K .NE. J) PROD = X(K)*PROD	00002980
1630	CONTINUE	00002990
1640	CONTINUE	00003000
	FJAC(N,J) = PROD/TEMP	00003010
1650	CONTINUE	00003020
	GO TO 1900	00003030
C		00003040
C	OSBORNE 1 FUNCTION.	00003050
C		00003060
1700	CONTINUE	00003070
	DO 1710 I = 1, 33	00003080
	TEMP = TEN*DFLOAT(I-1)	00003090
	TMP1 = DEXP(-X(4)*TEMP)	00003100
	TMP2 = DEXP(-X(5)*TEMP)	00003110
	FJAC(I,1) = -ONE	00003120
	FJAC(I,2) = -TMP1	00003130
	FJAC(I,3) = -TMP2	00003140
	FJAC(I,4) = TEMP*X(2)*TMP1	00003150
	FJAC(I,5) = TEMP*X(3)*TMP2	00003160
1710	CONTINUE	00003170
	GO TO 1900	00003180
C		00003190
C	OSBORNE 2 FUNCTION.	00003200
C		00003210
1800	CONTINUE	00003220
	DO 1810 I = 1, 65	00003230
	TEMP = DFLOAT(I-1)/TEN	00003240
	TMP1 = DEXP(-X(5)*TEMP)	00003250
	TMP2 = DEXP(-X(6)*(TEMP - X(9))**2)	00003260
	TMP3 = DEXP(-X(7)*(TEMP - X(10))**2)	00003270
	TMP4 = DEXP(-X(8)*(TEMP - X(11))**2)	00003280
	FJAC(I,1) = -TMP1	00003290
	FJAC(I,2) = -TMP2	00003300
	FJAC(I,3) = -TMP3	00003310
	FJAC(I,4) = -TMP4	00003320
	FJAC(I,5) = TEMP*X(1)*TMP1	00003330
	FJAC(I,6) = X(2)*(TEMP - X(9))**2*TMP2	00003340
	FJAC(I,7) = X(3)*(TEMP - X(10))**2*TMP3	00003350
	FJAC(I,8) = X(4)*(TEMP - X(11))**2*TMP4	00003360
	FJAC(I,9) = -TWO*X(2)*X(6)*(TEMP - X(9))*TMP2	00003370
	FJAC(I,10) = -TWO*X(3)*X(7)*(TEMP - X(10))*TMP3	00003380
	FJAC(I,11) = -TWO*X(4)*X(8)*(TEMP - X(11))*TMP4	00003390
1810	CONTINUE	00003400
1900	CONTINUE	00003410
	RETURN	00003420
C		00003430
C	LAST CARD OF SUBROUTINE SSQJAC.	00003440
C		00003450
	END	00003460

```

SUBROUTINE INITPT(N,X,NPROB,FACTOR)
INTEGER N,NPROB
DOUBLE PRECISION FACTOR
DOUBLE PRECISION X(N)
*****
SUBROUTINE INITPT

THIS SUBROUTINE SPECIFIES THE STANDARD STARTING POINTS FOR THE
FUNCTIONS DEFINED BY SUBROUTINE OBJPCN. THE SUBROUTINE RETURNS
IN X A MULTIPLE (FACTOR) OF THE STANDARD STARTING POINT. FOR
THE SEVENTH FUNCTION THE STANDARD STARTING POINT IS ZERO, SO IN
THIS CASE, IF FACTOR IS NOT UNITY, THEN THE SUBROUTINE RETURNS
THE VECTOR  $X(J) = \text{FACTOR}, J=1, \dots, N$ .

THE SUBROUTINE STATEMENT IS

SUBROUTINE INITPT(N,X,NPROB,FACTOR)

WHERE

N IS A POSITIVE INTEGER VARIABLE.

X IS A LINEAR ARRAY OF LENGTH N. ON OUTPUT X CONTAINS THE
STANDARD STARTING POINT FOR PROBLEM NPROB MULTIPLIED BY
FACTOR.

NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.

FACTOR SPECIFIES THE MULTIPLE OF THE STANDARD STARTING
POINT. IF FACTOR IS UNITY, NO MULTIPLICATION IS PERFORMED.

MINPACK. VERSION OF JANUARY 1978.
BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE

*****
INTEGER IVAR,J
DOUBLE PRECISION C1,C2,C3,C4,FIVE,H,HALF,
1 ONE,TEN,THREE,TWENTY,TWNTF,TWO,ZERO
DOUBLE PRECISION DFLOAT
DATA ZERO,HALF,ONE,TWO,THREE,FIVE,TEN,TWENTY,TWNTF
1 /0.D0,0.5D0,1.D0,2.D0,3.D0,5.D0,1.D1,2.D1,2.5D1/
DATA C1,C2,C3,C4 /4.D-1,2.5D0,1.5D-1,1.2D0/
DFLOAT(IVAR) = IVAR

SELECTION OF INITIAL POINT.

GO TO (100,200,300,400,500,600,700,800,900,1000,1100,
1 1200,1300,1400,1500,1600,1700,1800),NPROB

HELICAL VALLEY FUNCTION.

100 CONTINUE
X(1) = -ONE
X(2) = ZERO
X(3) = ZERO
GO TO 1900

```

C	BIGGS EXP6 FUNCTION.	00000600
C		00000610
200	CONTINUE	00000620
	X(1) = ONE	00000630
	X(2) = TWO	00000640
	X(3) = ONE	00000650
	X(4) = ONE	00000660
	X(5) = ONE	00000670
	X(6) = ONE	00000680
	GO TO 1900	00000690
C		00000700
C	GAUSSIAN FUNCTION.	00000710
C		00000720
300	CONTINUE	00000730
	X(1) = C1	00000740
	X(2) = ONE	00000750
	X(3) = ZERO	00000760
	GO TO 1900	00000770
C		00000780
C	POWELL BADLY SCALED FUNCTION.	00000790
C		00000800
400	CONTINUE	00000810
	X(1) = ZERO	00000820
	X(2) = ONE	00000830
	GO TO 1900	00000840
C		00000850
C	BOX 3-DIMENSIONAL FUNCTION.	00000860
C		00000870
500	CONTINUE	00000880
	X(1) = ZERO	00000890
	X(2) = TEN	00000900
	X(3) = TWENTY	00000910
	GO TO 1900	00000920
C		00000930
C	VARIABLY DIMENSIONED FUNCTION.	00000940
C		00000950
600	CONTINUE	00000960
	H = ONE/DFLOAT(N)	00000970
	DO 610 J = 1, N	00000980
	X(J) = ONE - DFLOAT(J) * H	00000990
610	CONTINUE	00001000
	GO TO 1900	00001010
C		00001020
C	WATSON FUNCTION.	00001030
C		00001040
700	CONTINUE	00001050
	DO 710 J = 1, N	00001060
	X(J) = ZERO	00001070
710	CONTINUE	00001080
	GO TO 1900	00001090
C		00001100
C	PENALTY FUNCTION I.	00001110
C		00001120
800	CONTINUE	00001130
	DO 810 J = 1, N	00001140
	X(J) = DFLOAT(J)	00001150
810	CONTINUE	00001160
	GO TO 1900	00001170
C		00001180

C	PENALTY FUNCTION II.	00001190
C		00001200
900	CONTINUE	00001210
	DO 910 J = 1, N	00001220
	X(J) = HALF	00001230
910	CONTINUE	00001240
	GO TO 1900	00001250
C		00001260
C	BROWN BADLY SCALED FUNCTION.	00001270
C		00001280
1000	CONTINUE	00001290
	X(1) = ONE	00001300
	X(2) = ONE	00001310
	GO TO 1900	00001320
C		00001330
C	BROWN AND DENNIS FUNCTION.	00001340
C		00001350
1100	CONTINUE	00001360
	X(1) = TWENTY	00001370
	X(2) = FIVE	00001380
	X(3) = -FIVE	00001390
	X(4) = -ONE	00001400
	GO TO 1900	00001410
C		00001420
C	GULF RESEARCH AND DEVELOPMENT FUNCTION.	00001430
C		00001440
1200	CONTINUE	00001450
	X(1) = FIVE	00001460
	X(2) = C2	00001470
	X(3) = C3	00001480
	GO TO 1900	00001490
C		00001500
C	TRIGONOMETRIC FUNCTION.	00001510
C		00001520
1300	CONTINUE	00001530
	H = ONE/DFLOAT(N)	00001540
	DO 1310 J = 1, N	00001550
	X(J) = H	00001560
1310	CONTINUE	00001570
	GO TO 1900	00001580
C		00001590
C	EXTENDED ROSENBERG FUNCTION.	00001600
C		00001610
1400	CONTINUE	00001620
	DO 1410 J = 1, N, 2	00001630
	X(J) = -C4	00001640
	X(J+1) = ONE	00001650
1410	CONTINUE	00001660
	GO TO 1900	00001670
C		00001680
C	EXTENDED POWELL SINGULAR FUNCTION.	00001690
C		00001700
1500	CONTINUE	00001710
	DO 1510 J = 1, N, 4	00001720
	X(J) = THREE	00001730
	X(J+1) = -ONE	00001740
	X(J+2) = ZERO	00001750
	X(J+3) = ONE	00001760
1510	CONTINUE	00001770

	GO TO 1900	00001780
C		00001790
C	BEALE FUNCTION.	00001800
C		00001810
	1600 CONTINUE	00001820
	X(1) = ONE	00001830
	X(2) = ONE	00001840
	GO TO 1900	00001850
C		00001860
C	WCOD FUNCTION.	00001870
C		00001880
	1700 CONTINUE	00001890
	X(1) = -THREE	00001900
	X(2) = -ONE	00001910
	X(3) = -THREE	00001920
	X(4) = -ONE	00001930
	GO TO 1900	00001940
C		00001950
C	CHEBYQUAD FUNCTION.	00001960
C		00001970
	1800 CONTINUE	00001980
	H = ONE/DFLOAT(N+1)	00001990
	DO 1810 J = 1, N	00002000
	X(J) = DFLOAT(J)*H	00002010
	1810 CONTINUE	00002020
C		00002030
C	COMPUTE MULTIPLE OF INITIAL POINT.	00002040
C		00002050
	1900 CONTINUE	00002060
	IF (FACTOR .EQ. ONE) GO TO 1940	00002070
	IF (NPROB .EQ. 7) GO TO 1920	00002080
	DO 1910 J = 1, N	00002090
	X(J) = FACTOR*X(J)	00002100
	1910 CONTINUE	00002110
	GO TO 1940	00002120
	1920 CONTINUE	00002130
	DO 1930 J = 1, N	00002140
	X(J) = FACTOR	00002150
	1930 CONTINUE	00002160
	1940 CONTINUE	00002170
	RETURN	00002180
C		00002190
C	LAST CARD OF SUBROUTINE INITPT.	00002200
C		00002210
	END	00002220

```

SUBROUTINE OBJFCN(N,X,F,NPROB)
INTEGER N,NPROB
DOUBLE PRECISION F
DOUBLE PRECISION X(N)
*****
SUBROUTINE OBJFCN
THIS SUBROUTINE DEFINES THE OBJECTIVE FUNCTIONS OF EIGHTEEN
NONLINEAR UNCONSTRAINED MINIMIZATION PROBLEMS. THE VALUES
OF N FOR FUNCTIONS 1,2,3,4,5,10,11,12,16 AND 17 ARE
3,6,3,2,3,2,4,3,2 AND 4, RESPECTIVELY.
FOR FUNCTION 7, N MAY BE 2 OR GREATER BUT IS USUALLY 6 OR 9.
FOR FUNCTIONS 6,8,9,13,14,15 AND 18 N MAY BE VARIABLE,
HOWEVER IT MUST BE EVEN FOR FUNCTION 14, A MULTIPLE OF 4 FOR
FUNCTION 15, AND NOT GREATER THAN 50 FOR FUNCTION 18.
THE SUBROUTINE STATEMENT IS
SUBROUTINE OBJFCN(N,X,F,NPROB)
WHERE
N IS A POSITIVE INTEGER VARIABLE.
X IS A LINEAR ARRAY OF LENGTH N.
F IS A REAL VARIABLE WHICH ON OUTPUT CONTAINS THE VALUE OF
THE NPROB OBJECTIVE FUNCTION EVALUATED AT X.
NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
SUBPROGRAMS REQUIRED
FORTRAN-SUPPLIED ... DABS,DATAN,DCOS,DEXP,DLOG,DSIGN,DSIN,
DSQRT
MINPACK. VERSION OF JANUARY 1978.
BURTON S. GARROW, KENNETH E. HILLSTROM, JORGE J. MORE
*****
INTEGER I,IEV,IVAR,J
DOUBLE PRECISION AP,ARG,AP,C2PDM6,CP0001,CP1,CP2,CP25,CP5,
1 C1P5,C2P25,C2P625,C3P5,C25,C29,C90,C100,C10000,C1PD6,
2 D1,D2,EIGHT,FIFTY,FIVE,FOUR,ONE,R,S1,S2,S3,
3 T,T1,T2,T3,TEN,TH,THREE,TPI,TWO,ZERO
DOUBLE PRECISION FVEC(50),Y(15)
DOUBLE PRECISION DFLOAT
DATA ZERO,ONE,TWO,THREE,FOUR,FIVE,EIGHT,TEN,FIFTY
1 /0.D0,1.D0,2.D0,3.D0,4.D0,5.D0,8.D0,1.D1,5.D1/
DATA C2PDM6,CP0001,CP1,CP2,CP25,CP5,C1P5,C2P25,
1 C2P625,C3P5,C25,C29,C90,C100,C10000,C1PD6
2 /2.D-6,1.D-4,1.D-1,2.D-1,2.5D-1,5.D-1,1.5D0,2.25D0,
3 2.625D0,3.5D0,2.5D1,2.9D1,9.D1,1.D2,1.D4,1.D6/
DATA AP,BP /1.D-5,1.D0/
DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7),
1 Y(8),Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15)
2 /9.D-4,4.4D-3,1.75D-2,5.4D-2,1.295D-1,2.42D-1,3.521D-1,

```

```

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590

```

Basic Subroutines

3	3.989D-1,3.521D-1,2.42D-1,1.295D-1,5.4D-2,1.75D-2,	00000600
4	4.4D-3,9.D-4/	00000610
	DFLOAT(IVAR) = IVAR	00000620
C		00000630
C	FUNCTION ROUTINE SELECTOR.	00000640
C		00000650
	GO TO (100,200,300,400,500,600,700,800,900,1000,1100,	00000660
1	1200,1300,1400,1500,1600,1700,1800),NPROB	00000670
C		00000680
C	HELICAL VALLEY FUNCTION.	00000690
C		00000700
100	CONTINUE	00000710
	TPI = EIGHT*DATAN(ONE)	00000720
	TH = DSIGN(CP25,X(2))	00000730
	IF (X(1) .GT. ZERO) TH = DATAN(X(2)/X(1))/TPI	00000740
	IF (X(1) .LT. ZERO) TH = DATAN(X(2)/X(1))/TPI + CP5	00000750
	ARG = X(1)**2 + X(2)**2	00000760
	R = DSQRT(ARG)	00000770
	T = X(3) - TEN*TH	00000780
	F = C100*(T**2 + (R - ONE)**2) + X(3)**2	00000790
	GC TO 1900	00000800
C		00000810
C	BIGGS EXP6 FUNCTION.	00000820
C		00000830
200	CONTINUE	00000840
	F = ZERO	00000850
	DO 210 I = 1, 13	00000860
	D1 = DFLOAT(I)/TEN	00000870
	D2 = DEXP(-D1) - FIVE*DEXP(-TEN*D1) + THREE*DEXP(-FOUR*D1)	00000880
	S1 = DEXP(-D1*X(1))	00000890
	S2 = DEXP(-D1*X(2))	00000900
	S3 = DEXP(-D1*X(5))	00000910
	T = X(3)*S1 - X(4)*S2 + X(6)*S3 - D2	00000920
	F = F + T**2	00000930
210	CONTINUE	00000940
	GC TO 1900	00000950
C		00000960
C	GAUSSIAN FUNCTION.	00000970
C		00000980
300	CONTINUE	00000990
	F = ZERO	00001000
	DO 310 I = 1, 15	00001010
	D1 = CP5*DFLOAT(I-1)	00001020
	D2 = C3P5 - D1 - X(3)	00001030
	ARG = -CP5*X(2)*D2**2	00001040
	R = DEXP(ARG)	00001050
	T = X(1)*R - Y(I)	00001060
	F = F + T**2	00001070
310	CONTINUE	00001080
	GC TO 1900	00001090
C		00001100
C	PCWELL BADIY SCALED FUNCTION.	00001110
C		00001120
400	CONTINUE	00001130
	T1 = C10000*X(1)*X(2) - ONE	00001140
	S1 = DEXP(-X(1))	00001150
	S2 = DEXP(-X(2))	00001160
	T2 = S1 + S2 - ONE - CP0001	00001170
	F = T1**2 + T2**2	00001180

C	GO TO 1900	03001190
C	BOX 3-DIMENSIONAL FUNCTION.	00001200
C		00001210
		00001220
500	CONTINUE	03001230
	F = ZERO	00001240
	DO 510 I = 1, 10	00001250
	D1 = DFLOAT(I)	00001260
	D2 = D1/TEN	03001270
	S1 = DEXP(-D2*X(1))	00001280
	S2 = DEXP(-D2*X(2))	00001290
	S3 = DEXP(-D2) - DEXP(-D1)	00001300
	T = S1 - S2 - S3*X(3)	00001310
	F = F + T**2	00001320
510	CONTINUE	00001330
	GO TO 1900	03001340
C		00001350
C	VARIABLY DIMENSIONED FUNCTION.	03001360
C		00001370
600	CONTINUE	00001380
	T1 = ZERO	00001390
	T2 = ZERO	03001400
	DO 610 J = 1, N	00001410
	T1 = T1 + DFLOAT(J)*(X(J) - ONE)	00001420
	T2 = T2 + (X(J) - ONE)**2	00001430
610	CONTINUE	00001440
	F = T2 + T1**2*(ONE + T1**2)	00001450
	GO TO 1900	03001460
C		00001470
C	WATSON FUNCTION.	00001480
C		00001490
700	CONTINUE	00001500
	F = ZERO	00001510
	DO 730 I = 1, 29	00001520
	D1 = DFLOAT(I)/C29	00001530
	S1 = ZERO	00001540
	D2 = ONE	00001550
	DO 710 J = 2, N	00001560
	S1 = S1 + DFLOAT(J-1)*D2*X(J)	00001570
	D2 = D1*D2	00001580
710	CONTINUE	03001590
	S2 = ZERO	00001600
	D2 = ONE	03001610
	DO 720 J = 1, N	00001620
	S2 = S2 + D2*X(J)	00001630
	D2 = D1*D2	00001640
720	CONTINUE	03001650
	T = S1 - S2**2 - ONE	00001660
	F = F + T**2	00001670
730	CONTINUE	03001680
	T1 = X(2) - X(1)**2 - ONE	00001690
	F = F + X(1)**2 + T1**2	00001700
	GO TO 1900	00001710
C		00001720
C	PENALTY FUNCTION I.	00001730
C		00001740
800	CONTINUE	00001750
	T1 = -CP25	00001760
	T2 = ZERO	00001770

DO 810 J = 1, N	00001780
T1 = T1 + X(J)**2	00001790
T2 = T2 + (X(J) - ONE)**2	00001800
810 CONTINUE	00001810
F = AF*T2 + BP*T1**2	00001820
GO TO 1900	00001830
C	00001840
C PENALTY FUNCTION II.	00001850
C	00001860
900 CONTINUE	00001870
T1 = -ONE	00001880
T2 = ZERO	00001890
T3 = ZERO	00001900
D1 = DEXP(CP1)	00001910
D2 = ONE	00001920
DO 920 J = 1, N	00001930
T1 = T1 + DFLOAT(N-J+1)*X(J)**2	00001940
S1 = DEXP(X(J)/TEN)	00001950
IF (J .EQ. 1) GO TO 910	00001960
S3 = S1 + S2 - D2*(D1 + ONE)	00001970
T2 = T2 + S3**2	00001980
T3 = T3 + (S1 - ONE/D1)**2	00001990
910 CONTINUE	00002000
S2 = S1	00002010
D2 = D1*D2	00002020
920 CONTINUE	00002030
F = AF*(T2 + T3) + BP*(T1**2 + (X(1) - CP2)**2)	00002040
GO TO 1900	00002050
C	00002060
C BROWN BADLY SCALED FUNCTION.	00002070
C	00002080
1000 CONTINUE	00002090
T1 = X(1) - C1PD6	00002100
T2 = X(2) - C2PD36	00002110
T3 = X(1)*X(2) - TWO	00002120
F = T1**2 + T2**2 + T3**2	00002130
GO TO 1900	00002140
C	00002150
C BROWN AND DENNIS FUNCTION.	00002160
C	00002170
1100 CONTINUE	00002180
F = ZERO	00002190
DO 1110 I = 1, 20	00002200
D1 = DFLOAT(I)/FIVE	00002210
D2 = DSIN(D1)	00002220
T1 = X(1) + D1*X(2) - DEXP(D1)	00002230
T2 = X(3) + D2*X(4) - DCOS(D1)	00002240
T = T1**2 + T2**2	00002250
F = F + T**2	00002260
1110 CONTINUE	00002270
GO TO 1900	00002280
C	00002290
C GULF RESEARCH AND DEVELOPMENT FUNCTION.	00002300
C	00002310
1200 CONTINUE	00002320
F = ZERO	00002330
D1 = TWO/THREE	00002340
DO 1210 I = 1, 99	00002350
ARG = DFLOAT(I)/C100	00002360

R = DABS((-FIFTY*DLOG(ARG))**D1 + C25 - X(2))	00002370
T1 = R**X(3)/X(1)	00002380
T2 = DEXP(-T1)	00002390
T = T2 - ARG	00002400
F = F + T**2	00002410
1210 CONTINUE	00002420
GO TO 1900	00002430
C	00002440
C TRIGONOMETRIC FUNCTION.	00002450
C	00002460
1300 CONTINUE	00002470
S1 = ZERO	00002480
DO 1310 J = 1, N	00002490
S1 = S1 + DCOS(X(J))	00002500
1310 CONTINUE	00002510
F = ZERO	00002520
DO 1320 J = 1, N	00002530
T = DFLCAT(N+J) - DSIN(X(J)) - S1 - DFLOAT(J)*DCOS(X(J))	00002540
F = F + T**2	00002550
1320 CONTINUE	00002560
GO TO 1900	00002570
C	00002580
C EXTENDED ROSENBROCK FUNCTION.	00002590
C	00002600
1400 CONTINUE	00002610
F = ZERO	00002620
DO 1410 J = 1, N, 2	00002630
T1 = ONE - X(J)	00002640
T2 = TEN*(X(J+1) - X(J)**2)	00002650
F = F + T1**2 + T2**2	00002660
1410 CONTINUE	00002670
GO TO 1900	00002680
C	00002690
C EXTENDED POWELL FUNCTION.	00002700
C	00002710
1500 CONTINUE	00002720
F = ZERO	00002730
DO 1510 J = 1, N, 4	00002740
T = X(J) + TEN*X(J+1)	00002750
T1 = X(J+2) - X(J+3)	00002760
S1 = FIVE*T1	00002770
T2 = X(J+1) - TWO*X(J+2)	00002780
S2 = T2**3	00002790
T3 = X(J) - X(J+3)	00002800
S3 = TEN*T3**3	00002810
F = F + T**2 + S1*T1 + S2*T2 + S3*T3	00002820
1510 CONTINUE	00002830
GO TO 1900	00002840
C	00002850
C BEALE FUNCTION.	00002860
C	00002870
1600 CONTINUE	00002880
S1 = ONE - X(2)	00002890
T1 = C1P5 - X(1)*S1	00002900
S2 = ONE - X(2)**2	00002910
T2 = C2P25 - X(1)*S2	00002920
S3 = ONE - X(2)**3	00002930
T3 = C2P625 - X(1)*S3	00002940
F = T1**2 + T2**2 + T3**2	00002950

Basic Subroutines

	GO TO 1900	00002960
C		00002970
C	WOOD FUNCTION.	00002980
C		00002990
1700	CONTINUE	00003000
	S1 = X(2) - X(1)**2	00003010
	S2 = ONE - X(1)	00003020
	S3 = X(2) - ONE	00003030
	T1 = X(4) - X(3)**2	00003040
	T2 = ONE - X(3)	00003050
	T3 = X(4) - ONE	00003060
	F = C100*S1**2 + S2**2 + C90*T1**2 + T2**2 +	00003070
1	TEN*(S3 + T3)**2 + (S3 - T3)**2/TEN	00003080
	GO TO 1900	00003090
C		00003100
C	CHEBYQUAD FUNCTION.	00003110
C		00003120
1800	CONTINUE	00003130
	DO 1810 I = 1, N	00003140
	FVEC(I) = ZERO	00003150
1810	CONTINUE	00003160
	DO 1830 J = 1, N	00003170
	T1 = ONE	00003180
	T2 = TWO*X(J) - ONE	00003190
	T = TWO*T2	00003200
	DO 1820 I = 1, N	00003210
	FVEC(I) = FVEC(I) + T2	00003220
	TH = T*T2 - T1	00003230
	T1 = T2	00003240
	T2 = TH	00003250
1820	CONTINUE	00003260
1830	CONTINUE	00003270
	F = ZERO	00003280
	D1 = ONE/DFLOAT(N)	00003290
	IEV = -1	00003300
	DO 1840 I = 1, N	00003310
	T = D1*FVEC(I)	00003320
	IF (IEV .GT. 0) T = T + ONE/(DFLOAT(I)**2 - ONE)	00003330
	F = F + T**2	00003340
	IEV = -IEV	00003350
1840	CONTINUE	00003360
1900	CONTINUE	00003370
	RETURN	00003380
C		00003390
C	LAST CARD OF SUBROUTINE OBJFCN.	00003400
C		00003410
	END	00003420

```

SUBROUTINE GRDFCN(N,X,G,NPROB) 00000010
INTEGER N,NPROB 00000020
DOUBLE PRECISION X(N),G(N) 00000030
***** 00000040
00000050
SUBROUTINE GRDFCN 00000060
00000070
THIS SUBROUTINE DEFINES THE GRADIENT VECTORS OF EIGHTEEN 00000080
NONLINEAR UNCONSTRAINED MINIMIZATION PROBLEMS. THE PROBLEM 00000090
DIMENSIONS ARE AS DESCRIBED IN THE PROLOGUE COMMENTS OF OBJFCN. 00000100
00000110
THE SUBROUTINE STATEMENT IS 00000120
00000130
SUBROUTINE GRDFCN(N,X,G,NPROB) 00000140
00000150
WHERE 00000160
00000170
N IS A POSITIVE INTEGER VARIABLE. 00000180
00000190
X IS A LINEAR ARRAY OF LENGTH N. 00000200
00000210
G IS A LINEAR ARRAY OF LENGTH N WHICH ON OUTPUT CONTAINS 00000220
THE COMPONENTS OF THE GRADIENT VECTOR OF THE NPROB 00000230
OBJECTIVE FUNCTION EVALUATED AT X. 00000240
00000250
NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE 00000260
NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18. 00000270
00000280
SUBPROGRAMS REQUIRED 00000290
00000300
FORTRAN-SUPPLIED ... DABS,DATAN,DCOS,DEXP,DLOG,DSIGN,DSIN, 00000310
DSQRT 00000320
00000330
MINPACK. VERSION OF JANUARY 1978. 00000340
BURTON S. GARROW, KENNETH E. HILLSTROM, JORGE J. MORE 00000350
00000360
***** 00000370
00000380
INTEGER I,IEV,IVAR,J 00000390
DOUBLE PRECISION AP,ARG,BP,C2PDM6,CP0001,CP1,CP2,CP25,CP5,C1P5, 00000400
1 C2P25,C2P625,C3P5,C1P8,C20P2,C25,C29,C100,C180,C200, 00000410
2 C10000,C1PD6,D1,D2,EIGHT,FIFTY,FIVE,FOUR,ONE,R,S1,S2,S3, 00000420
3 T,T1,T2,T3,TEN,TH,THREE,TPI,TWENTY,TWO,ZERO 00000430
DOUBLE PRECISION FVEC(50),Y(15) 00000440
DOUBLE PRECISION DFLOAT 00000450
DATA ZERO,ONE,TWO,THREE,FOUR,FIVE,EIGHT,TEN,TWENTY,FIFTY 00000460
1 /0.D0,1.D0,2.D0,3.D0,4.D0,5.D0,8.D0,1.D1,2.D1,5.D1/ 00000470
DATA C2PDM6,CP0001,CP1,CP2,CP25,CP5,C1P5,C2P25,C2P625, 00000480
1 C3P5,C1P8,C20P2,C25,C29,C100,C180,C200,C10000,C1PD6 00000490
2 /2.D-6,1.D-4,1.D-1,2.D-1,2.5D-1,5.D-1,1.5D0,2.25D0,2.625D0, 00000500
3 3.5D0,1.98D1,2.02D1,2.5D1,2.9D1,1.D2,1.8D2,2.D2,1.D4,1.D6/ 00000510
DATA AP,BP /1.D-5,1.D0/ 00000520
DATA Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),Y(7), 00000530
1 Y(8),Y(9),Y(10),Y(11),Y(12),Y(13),Y(14),Y(15) 00000540
2 /9.D-4,4.4D-3,1.75D-2,5.4D-2,1.295D-1,2.42D-1,3.521D-1, 00000550
3 3.989D-1,3.521D-1,2.42D-1,1.295D-1,5.4D-2,1.75D-2, 00000560
4 4.4D-3,9.D-4/ 00000570
DFLOAT(IVAR) = IVAR 00000580
00000590
GRADIENT ROUTINE SELECTOR.

```

C	GO TO (100,200,300,400,500,600,700,800,900,1000,1100,	00000600
	1 1200,1300,1400,1500,1600,1700,1800),NPROB	00000610
C		00000620
C	HELICAL VALLEY FUNCTION.	00000630
C		00000640
	100 CONTINUE	00000650
	TPI = EIGHT*DATAN(ONE)	00000660
	TH = DSIGN(CP25,X(2))	00000670
	IF (X(1) .GT. ZERO) TH = DATAN(X(2)/X(1))/TPI	00000680
	IF (X(1) .LT. ZERO) TH = DATAN(X(2)/X(1))/TPI + CP5	00000690
	ARG = X(1)**2 + X(2)**2	00000700
	R = DSQRT(ARG)	00000710
	T = X(3) - TEN*TH	00000720
	S1 = TEN*T/(TPI*ARG)	00000730
	G(1) = C200*(X(1) - X(1)/R + X(2)*S1)	00000740
	G(2) = C200*(X(2) - X(2)/R - X(1)*S1)	00000750
	G(3) = TWO*(C100*T + X(3))	00000760
	GO TO 1900	00000770
C		00000780
C	BIGGS EXP6 FUNCTION.	00000790
C		00000800
	200 CONTINUE	00000810
	DO 210 J = 1, N	00000820
	G(J) = ZERO	00000830
	210 CONTINUE	00000840
	DO 220 I = 1, 13	00000850
	D1 = DFLOAT(I)/TEN	00000860
	D2 = DEXP(-D1) - FIVE*DEXP(-TEN*D1) + THREE*DEXP(-FOUR*D1)	00000870
	S1 = DEXP(-D1*X(1))	00000880
	S2 = DEXP(-D1*X(2))	00000890
	S3 = DEXP(-D1*X(5))	00000900
	T = X(3)*S1 - X(4)*S2 + X(6)*S3 - D2	00000910
	TH = D1*T	00000920
	G(1) = G(1) - S1*TH	00000930
	G(2) = G(2) + S2*TH	00000940
	G(3) = G(3) + S1*T	00000950
	G(4) = G(4) - S2*T	00000960
	G(5) = G(5) - S3*TH	00000970
	G(6) = G(6) + S3*T	00000980
	220 CONTINUE	00000990
	G(1) = TWO*X(3)*G(1)	00001000
	G(2) = TWO*X(4)*G(2)	00001010
	G(3) = TWO*G(3)	00001020
	G(4) = TWO*G(4)	00001030
	G(5) = TWO*X(6)*G(5)	00001040
	G(6) = TWO*G(6)	00001050
	GO TO 1900	00001060
C		00001070
C	GAUSSIAN FUNCTION.	00001080
C		00001090
	300 CONTINUE	00001100
	G(1) = ZERO	00001110
	G(2) = ZERO	00001120
	G(3) = ZERO	00001130
	DO 310 I = 1, 15	00001140
	D1 = CP5*DFLOAT(I-1)	00001150
	D2 = C3P5 - D1 - X(3)	00001160
	ARG = -CP5*X(2)*D2**2	00001170
		00001180

	R = DEXP(ARG)	00001190
	T = X(1)*R - Y(I)	00001200
	S1 = R*T	00001210
	S2 = D2*S1	00001220
	G(1) = G(1) + S1	00001230
	G(2) = G(2) - D2*S2	00001240
	G(3) = G(3) + S2	00001250
310	CCONTINUE	00001260
	G(1) = TWO*G(1)	00001270
	G(2) = X(1)*G(2)	00001280
	G(3) = TWO*X(1)*X(2)*G(3)	00001290
	GO TO 1900	00001300
C		00001310
C	PCWELL BADLY SCALED FUNCTION.	00001320
C		00001330
400	CONTINUE	00001340
	T1 = C10000*X(1)*X(2) - ONE	00001350
	S1 = DEXP(-X(1))	00001360
	S2 = DEXP(-X(2))	00001370
	T2 = S1 + S2 - ONE - CP0001	00001380
	G(1) = TWO*(C10000*X(2)*T1 - S1*T2)	00001390
	G(2) = TWO*(C10000*X(1)*T1 - S2*T2)	00001400
	GO TO 1900	00001410
C		00001420
C	BOX 3-DIMENSIONAL FUNCTION.	00001430
C		00001440
500	CONTINUE	00001450
	G(1) = ZERO	00001460
	G(2) = ZERO	00001470
	G(3) = ZERO	00001480
	DO 510 I = 1, 10	00001490
	D1 = DFLOAT(I)	00001500
	D2 = D1/TEN	00001510
	S1 = DEXP(-D2*X(1))	00001520
	S2 = DEXP(-D2*X(2))	00001530
	S3 = DEXP(-D2) - DEXP(-D1)	00001540
	T = S1 - S2 - S3*X(3)	00001550
	TH = D2*T	00001560
	G(1) = G(1) - S1*TH	00001570
	G(2) = G(2) + S2*TH	00001580
	G(3) = G(3) - S3*TH	00001590
510	CONTINUE	00001600
	G(1) = TWO*G(1)	00001610
	G(2) = TWO*G(2)	00001620
	G(3) = TWO*G(3)	00001630
	GO TO 1900	00001640
C		00001650
C	VARIABLELY DIMENSIONED FUNCTION.	00001660
C		00001670
600	CONTINUE	00001680
	T1 = ZERO	00001690
	DO 610 J = 1, N	00001700
	T1 = T1 + DFLOAT(J)*(X(J) - ONE)	00001710
610	CONTINUE	00001720
	T = T1*(ONE + TWO*T1**2)	00001730
	DO 620 J = 1, N	00001740
	G(J) = TWO*(X(J) - ONE + DFLOAT(J)*T)	00001750
620	CONTINUE	00001760
	GO TO 1900	00001770

C		00001780
C	WATSON FUNCTION.	00001790
C		00001800
700	CONTINUE	00001810
	DO 710 J = 1, N	00001820
	G(J) = ZERO	00001830
710	CONTINUE	00001840
	DO 750 I = 1, 29	00001850
	D1 = DFLOAT(I)/C29	00001860
	S1 = ZERO	00001870
	D2 = ONE	00001880
	DO 720 J = 2, N	00001890
	S1 = S1 + DFLOAT(J-1)*D2*X(J)	00001900
	D2 = D1*D2	00001910
720	CONTINUE	00001920
	S2 = ZERO	00001930
	D2 = ONE	00001940
	DO 730 J = 1, N	00001950
	S2 = S2 + D2*X(J)	00001960
	D2 = D1*D2	00001970
730	CONTINUE	00001980
	T = S1 - S2**2 - ONE	00001990
	S3 = TWO*D1*S2	00002000
	D2 = TWO/D1	00002010
	DO 740 J = 1, N	00002020
	G(J) = G(J) + D2*(DFLOAT(J-1) - S3)*T	00002030
	D2 = D1*D2	00002040
740	CONTINUE	00002050
750	CONTINUE	00002060
	T1 = X(2) - X(1)**2 - ONE	00002070
	G(1) = G(1) + X(1)*(TWO - FOUR*T1)	00002080
	G(2) = G(2) + TWO*T1	00002090
	GO TO 1900	00002100
C		00002110
C	PENALTY FUNCTION I.	00002120
C		00002130
800	CONTINUE	00002140
	T1 = -CP25	00002150
	DO 810 J = 1, N	00002160
	T1 = T1 + X(J)**2	00002170
810	CONTINUE	00002180
	D1 = TWO*AP	00002190
	TH = FOUR*BP*T1	00002200
	DO 820 J = 1, N	00002210
	G(J) = D1*(X(J) - ONE) + X(J)*TH	00002220
820	CONTINUE	00002230
	GO TO 1900	00002240
C		00002250
C	PENALTY FUNCTION II.	00002260
C		00002270
900	CONTINUE	00002280
	T1 = -ONE	00002290
	DO 910 J = 1, N	00002300
	T1 = T1 + DFLOAT(N-J+1)*X(J)**2	00002310
910	CONTINUE	00002320
	D1 = DEXP(CP1)	00002330
	D2 = ONE	00002340
	TH = FOUR*BP*T1	00002350
	DO 930 J = 1, N	00002360


```

      G(J) = DFLOAT(N-J+1)*X(J)*TH
      S1 = DEXP(X(J)/TEN)
      IF (J.EQ. 1) GO TO 920
      S3 = S1 + S2 - D2*(D1 + ONE)
      G(J) = G(J) + AP*S1*(S3 + S1 - ONE/D1)/FIVE
      G(J-1) = G(J-1) + AP*S2*S3/FIVE
920   CONTINUE
      S2 = S1
      D2 = D1*D2
930   CONTINUE
      G(1) = G(1) + TWO*BP*(X(1) - CP2)
      GO TO 1900

C
C   BROWN BADLY SCALED FUNCTION.
C
1000  CONTINUE
      T1 = X(1) - C1PD6
      T2 = X(2) - C2PDM6
      T3 = X(1)*X(2) - TWO
      G(1) = TWO*(T1 + X(2)*T3)
      G(2) = TWO*(T2 + X(1)*T3)
      GO TO 1900

C
C   BROWN AND DENNIS FUNCTION.
C
1100  CONTINUE
      G(1) = ZERO
      G(2) = ZERO
      G(3) = ZERO
      G(4) = ZERO
      DO 1110 I = 1, 20
        D1 = DFLOAT(I)/FIVE
        D2 = DSIN(D1)
        T1 = X(1) + D1*X(2) - DEXP(D1)
        T2 = X(3) + D2*X(4) - DCOS(D1)
        T = T1**2 + T2**2
        S1 = T1*T
        S2 = T2*T
        G(1) = G(1) + S1
        G(2) = G(2) + D1*S1
        G(3) = G(3) + S2
        G(4) = G(4) + D2*S2
1110  CONTINUE
      G(1) = FOUR*G(1)
      G(2) = FOUR*G(2)
      G(3) = FOUR*G(3)
      G(4) = FOUR*G(4)
      GO TO 1900

C
C   GULF RESEARCH AND DEVELOPMENT FUNCTION.
C
1200  CONTINUE
      G(1) = ZERO
      G(2) = ZERO
      G(3) = ZERO
      D1 = TWO/THREE
      DO 1210 I = 1, 99
        ARG = DFLOAT(I)/C100
        R = DABS((-FIFTY*DLOG(ARG))**D1 + C25 - X(2))

```

```

00002370
00002380
00002390
00002400
00002410
00002420
00002430
00002440
00002450
00002460
00002470
00002480
00002490
00002500
00002510
00002520
00002530
00002540
00002550
00002560
00002570
00002580
00002590
00002600
00002610
00002620
00002630
00002640
00002650
00002660
00002670
00002680
00002690
00002700
00002710
00002720
00002730
00002740
00002750
00002760
00002770
00002780
00002790
00002800
00002810
00002820
00002830
00002840
00002850
00002860
00002870
00002880
00002890
00002900
00002910
00002920
00002930
00002940
00002950

```

	T1 = R**X(3)/X(1)	00002960
	T2 = DEXP(-T1)	00002970
	T = T2 - ARG	00002980
	S1 = T1*T2*T	00002990
	G(1) = G(1) + S1	00003000
	G(2) = G(2) + S1/R	00003010
	G(3) = G(3) - S1*DLOG(R)	00003020
1210	CONTINUE	00003030
	G(1) = TWO*G(1)/X(1)	00003040
	G(2) = TWO*X(3)*G(2)	00003050
	G(3) = TWO*G(3)	00003060
	GO TO 1900	00003070
C		00003080
C	TRIGONOMETRIC FUNCTION.	00003090
C		00003100
1300	CONTINUE	00003110
	S1 = ZERO	00003120
	DO 1310 J = 1, N	00003130
	G(J) = DCOS(X(J))	00003140
	S1 = S1 + G(J)	00003150
1310	CONTINUE	00003160
	S2 = ZERO	00003170
	DO 1320 J = 1, N	00003180
	TH = DSIN(X(J))	00003190
	T = DFLOAT(N+J) - TH - S1 - DFLOAT(J)*G(J)	00003200
	S2 = S2 + T	00003210
	G(J) = (DFLOAT(J)*TH - G(J))*T	00003220
1320	CONTINUE	00003230
	DO 1330 J = 1, N	00003240
	G(J) = TWO*(G(J) + DSIN(X(J))*S2)	00003250
1330	CONTINUE	00003260
	GO TO 1900	00003270
C		00003280
C	EXTENDED ROSENBRACK FUNCTION.	00003290
C		00003300
1400	CONTINUE	00003310
	DO 1410 J = 1, N, 2	00003320
	T1 = ONE - X(J)	00003330
	G(J+1) = C200*(X(J+1) - X(J)**2)	00003340
	G(J) = -TWO*(X(J)*G(J+1) + T1)	00003350
1410	CONTINUE	00003360
	GO TO 1900	00003370
C		00003380
C	EXTENDED POWELL FUNCTION.	00003390
C		00003400
1500	CONTINUE	00003410
	DO 1510 J = 1, N, 4	00003420
	T = X(J) + TEN*X(J+1)	00003430
	T1 = X(J+2) - X(J+3)	00003440
	S1 = FIVE*T1	00003450
	T2 = X(J+1) - TWO*X(J+2)	00003460
	S2 = FOUR*T2**3	00003470
	T3 = X(J) - X(J+3)	00003480
	S3 = TWENTY*T3**3	00003490
	G(J) = TWO*(T + S3)	00003500
	G(J+1) = TWENTY*T + S2	00003510
	G(J+2) = TWO*(S1 - S2)	00003520
	G(J+3) = -TWO*(S1 + S3)	00003530
1510	CONTINUE	00003540

GO TO 1900	00003550
C	00003560
C	00003570
C	00003580
1600 CONTINUE	00003590
S1 = ONE - X(2)	00003600
T1 = C1P5 - X(1)*S1	00003610
S2 = ONE - X(2)**2	00003620
I2 = C2P25 - X(1)*S2	00003630
S3 = ONE - X(2)**3	00003640
T3 = C2P625 - X(1)*S3	00003650
G(1) = -TWO*(S1*T1 + S2*T2 + S3*T3)	00003660
G(2) = TWO*X(1)*(T1 + X(2)*(TWO*T2 + THREE*X(2)*T3))	00003670
GO TO 1900	00003680
C	00003690
C	00003700
C	00003710
1700 CONTINUE	00003720
S1 = X(2) - X(1)**2	00003730
S2 = ONE - X(1)	00003740
S3 = X(2) - ONE	00003750
T1 = X(4) - X(3)**2	00003760
T2 = ONE - X(3)	00003770
T3 = X(4) - ONE	00003780
G(1) = -TWO*(C200*X(1)*S1 + S2)	00003790
G(2) = C200*S1 + C20P2*S3 + C19P8*T3	00003800
G(3) = -TWO*(C180*X(3)*T1 + T2)	00003810
G(4) = C180*T1 + C20P2*T3 + C19P8*S3	00003820
GO TO 1900	00003830
C	00003840
C	00003850
C	00003860
1800 CONTINUE	00003870
DO 1810 I = 1, N	00003880
FVEC(I) = ZERO	00003890
1810 CONTINUE	00003900
DO 1830 J = 1, N	00003910
T1 = ONE	00003920
I2 = TWO*X(J) - ONE	00003930
T = TWO*T2	00003940
DO 1820 I = 1, N	00003950
FVEC(I) = FVEC(I) + T2	00003960
TH = T*T2 - T1	00003970
T1 = I2	00003980
T2 = TH	00003990
1020 CONTINUE	00004000
1830 CONTINUE	00004010
D1 = ONE/DFLOAT(N)	00004020
IEV = -1	00004030
DO 1840 I = 1, N	00004040
FVEC(I) = D1*FVEC(I)	00004050
IF (IEV.GT. 0) FVEC(I) = FVEC(I) + ONE/(DFLOAT(I)**2 - ONE)	00004060
IEV = -IEV	00004070
1840 CONTINUE	00004080
DO 1860 J = 1, N	00004090
G(J) = ZERO	00004100
T1 = ONE	00004110
T2 = TWO*X(J) - ONE	00004120
T = TWO*T2	00004130

S1 = ZERO	00004140
S2 = TWO	00004150
DO 1850 I = 1, N	00004160
G(J) = G(J) + FVEC(I)*S2	00004170
TH = FOUR*T2 + T*S2 - S1	00004180
S1 = S2	00004190
S2 = TH	00004200
TH = I*T2 - T1	00004210
T1 = T2	00004220
T2 = TH	00004230
1850 CONTINUE	00004240
1860 CONTINUE	00004250
D2 = TWO*D1	00004260
DO 1870 J = 1, N	00004270
G(J) = D2*G(J)	00004280
1870 CONTINUE	00004290
1900 CONTINUE	00004300
RETURN	00004310
C	00004320
C LAST CARD OF SUBROUTINE GRDFCN.	00004330
C	00004340
END	00004350

A P P E N D I X 2

Sample Driver and Interface Function

THIS PROGRAM TESTS CODES FOR THE LEAST-SQUARES SOLUTION OF
M NONLINEAR EQUATIONS IN N VARIABLES. IT CONSISTS OF A DRIVER
AND AN INTERFACE SUBROUTINE FCN. THE DRIVER READS IN DATA,
CALLS THE NONLINEAR LEAST-SQUARES SOLVER, AND FINALLY PRINTS
OUT INFORMATION ON THE PERFORMANCE OF THE SOLVER. THIS IS
ONLY A SAMPLE DRIVER, MANY OTHER DRIVERS ARE POSSIBLE. THE
INTERFACE SUBROUTINE FCN IS NECESSARY TO TAKE INTO ACCOUNT THE
FORMS OF CALLING SEQUENCES USED BY THE FUNCTION AND JACOBIAN
SUBROUTINES IN THE VARIOUS NONLINEAR LEAST-SQUARES SOLVERS.

SUBPROGRAMS REQUIRED

USER-SUPPLIED FCN

MINPACK-SUPPLIED ... ENORM,INITPT,SOLVER,SSQFCN

MINPACK. VERSION OF OCTOBER 1977.

BURTON S. GARROW, KENNETH E. HILLSTROM, JORGE J. MORE

INTEGER I,IC,INFO,K,LDFJAC,LWA,M,N,NFEV,NJEV,

1 NPEGE,NREAD,NTRIES,NWRITE

INTEGER IWA(40),JA(60),NA(60),NF(60),NJ(60),NP(60),NX(60)

DOUBLE PRECISION FACTOR,FNORM1,FNORM2,ONE,TEN,TOL

DOUBLE PRECISION FJAC(65,40),FNM(60),FVEC(65),WA(265),X(40)

DOUBLE PRECISION ENORM

EXTERNAL FCN

COMMON /REFNUM/ NPROB,NFEV,NJEV

LOGICAL INPUT UNIT IS ASSUMED TO BE NUMBER 5.

LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.

DATA NREAD,NWRITE /5,6/

DATA ONE,TEN,TOL /1.D0,1.D1,1.D-10/

LDFJAC = 65

LWA = 265

IC = 0

10 CONTINUE

READ (NREAD,1000) NPROB,N,M,NTRIES

IF (NPROB.LE. 0) GO TO 30

FACTOR = ONE

DO 20 K = 1, NTRIES

IC = IC + 1

CALL INITPT(N,X,NPROB,FACTOR)

CALL SSQFCN(M,N,X,FVEC,NPROB)

FNORM1 = ENORM(M,FVEC)

WRITE (NWRITE,2000) NPROB,N,M

NFEV = 0

NJEV = 0

CALL SOLVER(FCN,M,N,X,FVEC,FJAC,LDFJAC,TOL,INFO,IWA,WA,LWA)

CALL SSQFCN(M,N,X,FVEC,NPROB)

FNORM2 = ENORM(M,FVEC)

NP(IC) = NPROB

NA(IC) = N

MA(IC) = M

NF(IC) = NFEV

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590

```

      NJ(IC) = NJEV                                00000600
      NX(IC) = INFO                                00000610
      FNM(IC) = FNORM2                             00000620
      WRITE (NWRITE,3000) FNORM1,FNORM2,NFEV,NJEV,INFO,(X(I),I=1,N) 00000630
      FACTOR = TEN*FACTOR                          00000640
20    CONTINUE                                    00000650
      GO TO 10                                     00000660
30    CONTINUE                                    00000670
      WRITE (NWRITE,4000) IC                       00000680
      WRITE (NWRITE,5000)                          00000690
      DO 40 I = 1, IC                              00000700
          WRITE (NWRITE,6000) NP(I),NA(I),MA(I),NF(I),NJ(I),NX(I),FNM(I) 00000710
40    CONTINUE                                    00000720
      STOP                                         00000730
1000 FORMAT (4I5)                                00000740
2000 FORMAT ( // // 5X,8H PROBLEM,I5,5X,11H DIMENSIONS,2I5,5X // ) 00000750
3000 FORMAT (5X,33H INITIAL L2 NORM OF THE RESIDUALS,D15.7 // ) 00000760
      1      5X,33H FINAL L2 NORM OF THE RESIDUALS ,D15.7 // 00000770
      2      5X,33H NUMBER OF FUNCTION EVALUATIONS ,I10 // 00000780
      3      5X,33H NUMBER OF JACOBIAN EVALUATIONS ,I10 // 00000790
      4      5X,15H EXIT PARAMETER ,18X,I10 // 00000800
      5      5X,27H FINAL APPROXIMATE SOLUTION // (5X,5D15.7) 00000810
4000 FORMAT (12H1SUMMARY OF ,I3,16H CALLS TO SOLVER/) 00000820
5000 FORMAT (49H NPROB N M NFEV NJEV INFO FINAL L2 NORM/) 00000830
6000 FORMAT (3I5,3I6,2X,D15.7)                 00000840
C                                         00000850
C      LAST CARD OF DRIVER.                00000860
C                                         00000870
      END                                         00000880

```

Interface Function

```

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

```

```

C THE CALLING SEQUENCE FOR FCN SHOULD BE IDENTICAL WITH THE
C CALLING SEQUENCE OF THE FUNCTION SUBROUTINE IN THE NONLINEAR
C LEAST-SQUARES SOLVER. FCN SHOULD ONLY CALL THE TESTING
C FUNCTION AND JACOBIAN SUBROUTINES SSQFCN AND SSQJAC WITH
C THE APPROPRIATE VALUE OF PROBLEM NUMBER (NPROB).
C

```

```

C SUBPROGRAMS REQUIRED
C

```

```

C   MINPACK-SUPPLIED ... SSQFCN,SSQJAC
C

```

```

C MINPACK. VERSION OF OCTOBER 1977.
C

```

```

C BURTON S. GARROW, KENNETH E. HILLSTROM, JORGE J. MORE
C

```

```

C *****
C

```

```

C INTEGER NPROB,NFEV,NJEV
C

```

```

C COMMON /REFNUM/ NPROB,NFEV,NJEV
C

```

```

C IF (IFLAG .EQ. 1) CALL SSQFCN(M,N,X,FVEC,NPROB)
C

```

```

C IF (IFLAG .EQ. 2) CALL SSQJAC(M,N,X,FJAC,LDFJAC,NPROB)
C

```

```

C IF (IFLAG .EQ. 1) NFEV = NFEV + 1
C

```

```

C IF (IFLAG .EQ. 2) NJEV = NJEV + 1
C

```

```

C RETURN
C

```

```

C LAST CARD OF INTERFACE SUBROUTINE FCN.
C

```

```

C END

```

```

00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
00001000
00001010
00001020
00001030
00001040
00001050
00001060
00001070
00001080
00001090
00001100
00001110
00001120
00001130
00001140
00001150
00001160
00001170
00001180

```


A P P E N D I X 3

Sample Data

Sample Data for Nonlinear Equations

NPROB	N	NTRIES
-------	---	--------

1	2	3
2	4	3
3	2	2
4	4	3
5	3	3
6	6	2
6	9	2
6	12	2
7	5	3
7	6	3
7	7	3
7	8	1
7	9	1
8	10	3
8	30	1
9	10	3
10	1	3
10	10	3
11	10	3
12	10	3
13	10	3
14	10	3
0	0	0

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230

NPROB	N	M	NTRIES	
1	5	10	1	00000010
1	5	50	1	00000020
2	5	10	1	00000030
2	5	50	1	00000040
3	5	10	1	00000050
3	5	50	1	00000060
4	2	2	3	00000070
5	3	3	3	00000080
6	4	4	3	00000090
7	2	2	3	00000100
8	3	15	3	00000110
9	4	11	3	00000120
10	3	16	3	00000130
11	6	31	3	00000140
11	9	31	3	00000150
11	12	31	3	00000160
12	3	10	1	00000170
13	2	10	1	00000180
14	4	20	3	00000190
15	1	8	3	00000200
15	8	8	1	00000210
15	9	9	1	00000220
15	10	10	1	00000230
16	10	10	3	00000240
16	30	30	1	00000250
16	40	40	1	00000260
17	5	33	1	00000270
18	11	65	1	00000280
0	0	0	0	00000290

Sample Data for Unconstrained Minimization

NPROB	N	NTRIES
1	3	3
2	6	1
3	3	1
4	2	1
5	3	1
6	10	3
7	9	3
7	12	3
8	10	3
9	1	3
9	4	3
9	10	3
10	2	3
11	4	3
12	3	2
13	10	3
14	2	3
15	4	3
16	2	3
17	4	3
18	7	1
18	8	1
18	9	1
18	10	1
0	0	0

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250