ANL-AMD-TM-324

ARGONNE NATIONAL LABORATORY

TESTING UNCONSTRAINED OPTIMIZATION SOFTWARE

by

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

U of C-AUA-USDOE

APPLIED MATHEMATICS DIVISION

ANL-AMD-TM-324

ARGONNE NATIONAL LABORATORY
Argonne, Illinois 60439

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.

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

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

TABLE OF CONTENTS

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

TESTING UNCONSTRAINED OPTIMIZATION SOFTWARE

by

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

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. Hillstrom [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 $f_i: R^n \to R$ for $i = 1,\dots,n$, solve

$$f_i(x) = 0, \quad 1 \le i \le n, \quad x \in R^n .$$

II. Nonlinear least squares. Given $f_i: R^n \to R$ for $i = 1,\dots,m$ with $m \ge n$, solve

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

III. Unconstrained minimization. Given $f: R^n \to R$, 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

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

and

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

returns in FJAC the Jacobian matrix

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

(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,C,NPROB)
```

returns in G the gradient vector

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

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 $f_1,\ldots,f_m$, we can obtain an unconstrained minimization problem by setting

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

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

$$f_i(x) = 0 \ , \qquad 1 \le i \le n \ , \tag{3.2}$$

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

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

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

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

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 $(5,4)$

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

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) <u>Brown badly scaled function [unpublished]</u>

a) $n = 2, \quad m = 3$

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) <u>Beale function [2]</u>

a) $n = 2, \quad m = 3$

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) <u>Jennrich and Sampson function [16]</u>

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

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

c) $x_0 = (0.3,0.4)$

d) $f = 124.362...$ at $x_1 = x_2 = 0.2578...$ for $m = 10$

7) Helical valley function [11]

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

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) $n = 3, \quad m = 15$

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

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

9) <u>Gaussian function [unpublished]</u>

a) $n = 3, \quad m = 15$

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) <u>Meyer function [18]</u>

a) $n = 3, \quad m = 16$

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

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

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

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

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

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

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

d) $f = 0$ at the origin

14) <u>Wood function [9]</u>

a) $n = 4, \quad m = 6$

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) <u>Kowalik and Osborne function [17]</u>

a) $n = 4, \quad m = 11$

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

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

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) $f = 85822.2\ldots$ 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) $f = 5.46489\ldots 10^{-5}$

18) Biggs EXP6 function [3]

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

b) $f_i(x) = x_3\exp[-t_ix_1] - x_4\exp[-t_ix_2] + x_6\exp[-t_ix_5] - y_i$

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

19) Osborne 2 function [21]

a) $n = 11, \quad m = 65$

b) $f_i(x) = y_i - (x_1\exp[-t_ix_5] + x_2\exp[-(t_i-x_9)^2x_6]$

$+ x_3\exp[-(t_i-x_{10})^2x_7] + x_4\exp[-(t_i-x_{11})^2x_8])$

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) $f = 4.01377\ldots\ 10^{-2}$

20) Watson function [17]

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

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

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

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

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

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

$f = 4.72238\ldots\ 10^{-10}$ 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 $(1,\ldots,1)$

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) <u>Penalty function I [14]</u>

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

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

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

where $a = 10^{-5}$

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

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

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

24) <u>Penalty function II [14]</u>

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

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

$f = 2.93660\ldots\ 10^{-4}$ 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 $\zeta_j = 1-(j/n)$

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

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) $$f_i(x) = x_i + h\left[(1-t_i)\sum_{j=1}^{i} t_j(x_j+t_j+1)^3 + t_i\sum_{j=i+1}^{n}(1-t_j)(x_j+t_j+1)^3\right]\Big/2$$

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) $x_0 = (-1,\ldots,-1)$

d) $f = 0$

31) Broyden banded function [8]

a) n variable, $m = n$

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

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

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

c) $x_0 = (-1,\ldots,-1)$

d) $f = 0$

32) Linear function - full rank [unpublished]

a) n variable, $m \geq n$

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

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

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

d) $f = m-n$ at $(-1,\ldots,-1)$

33) Linear function - rank 1 [unpublished]

a) n variable, $m \geq n$

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

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

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) $f_1(x) = -1, \quad f_m(x) = -1$

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

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

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

35) Chebyquad function [12]

a) n variable, $m \geq n$

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

where T_i is the ith Chebyshev polynomial shifted to the interval [0,1] and hence,

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

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

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

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

$f = 6.50395\ldots 10^{-3}$ 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

		x_s		$10x_s$		$100x_s$	
PROBLEM	SCALING	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 CALLS TO 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 TO 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

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

where α is a positive scalar and Σ 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

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

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

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

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

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

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

$$(5.2) \qquad \hat{F}(x) = \Sigma_1 F(\Sigma_2 x)$$

where both Σ_1 and Σ_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

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

(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 TO 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 TO 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 CALLS TO NEQ1

NPROB	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 CALLS TO NEQ2

NPROB	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 Applics*, *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. Hillstrom, K. E., A Simulation Test Approach to the Evaluation of Nonlinear 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

```
      SUBROUTINE INITPT(N,X,NPROB,FACTOR)
      INTEGER N,NPROB
      DOUBLE PRECISION FACTOR
      DOUBLE PRECISION X(N)
C     **********
C
C     SUBROUTINE INITPT
C
C     THIS SUBROUTINE SPECIFIES THE STANDARD STARTING POINTS FOR
C     THE FUNCTIONS DEFINED BY SUBROUTINES COMFCN AND VECFCN. THE
C     SUBROUTINE RETURNS IN X A MULTIPLE (FACTOR) OF THE STANDARD
C     STARTING POINT. FOR THE SIXTH FUNCTION THE STANDARD STARTING
C     POINT IS ZERO, SO IN THIS CASE, IF FACTOR IS NOT UNITY, THEN
C     THE SUBROUTINE RETURNS THE VECTOR  X(J) = FACTOR, J=1,...,N.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE INITPT(N,X,NPROB,FACTOR)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER VARIABLE.
C
C       X IS A LINEAR ARRAY OF LENGTH N. ON OUTPUT X CONTAINS THE
C         STANDARD STARTING POINT FOR PROBLEM NPROB MULTIPLIED BY
C         FACTOR.
C
C       NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 14.
C
C       FACTOR SPECIFIES THE MULTIPLE OF THE STANDARD STARTING
C         POINT. IF FACTOR IS UNITY, NO MULTIPLICATION IS PERFORMED.
C
C     MINPACK. VERSION OF SEPTEMBER 1977.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      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
C
C     SELECTION OF INITIAL POINT.
C
      GO TO (100,200,300,400,500,600,700,800,900,1000,
     1       1100,1200,1300,1400),NPROB
C
C     ROSENBROCK FUNCTION.
C
  100 CONTINUE
      X(1) = -C1
      X(2) = ONE
      GO TO 1500
C
C     POWELL SINGULAR FUNCTION.
C
  200 CONTINUE
      X(1) = THREE
      X(2) = -ONE
      X(3) = ZERO
      X(4) = ONE
      GO TO 1500
C
C     POWELL BADLY SCALED FUNCTION.
C
  300 CONTINUE
      X(1) = ZERO
      X(2) = ONE
      GO TO 1500
C
C     WOOD FUNCTION.
C
  400 CONTINUE
      X(1) = -THREE
      X(2) = -ONE
      X(3) = -THREE
      X(4) = -ONE
      GO TO 1500
C
C     HELICAL VALLEY FUNCTION.
C
  500 CONTINUE
      X(1) = -ONE
      X(2) = ZERO
      X(3) = ZERO
      GO TO 1500
C
C     WATSON FUNCTION.
C
  600 CONTINUE
      DO 610 J = 1, N
         X(J) = ZERO
  610    CONTINUE
      GO TO 1500
C
C     CHEBYQUAD FUNCTION.
C
  700 CONTINUE
      H = ONE/DFLOAT(N+1)
      DO 710 J = 1, N
         X(J) = DFLOAT(J)*H
  710    CONTINUE
      GO TO 1500
C
C     BROWN ALMOST-LINEAR FUNCTION.
C
  800 CONTINUE
      DO 810 J = 1, N
         X(J) = HALF
  810    CONTINUE
      GO TO 1500
C
C     DISCRETE BOUNDARY VALUE AND INTEGRAL EQUATION FUNCTIONS.
C
  900 CONTINUE
 1000 CONTINUE
      H = ONE/DFLOAT(N+1)
      DO 1010 J = 1, N
         TJ = DFLOAT(J)*H
         X(J) = TJ*(TJ - ONE)
 1010    CONTINUE
      GO TO 1500
C
C     TRIGONOMETRIC FUNCTION.
C
 1100 CONTINUE
      H = ONE/DFLOAT(N)
      DO 1110 J = 1, N
         X(J) = H
 1110    CONTINUE
      GO TO 1500
C
C     VARIABLY DIMENSIONED FUNCTION.
C
 1200 CONTINUE
      H = ONE/DFLOAT(N)
      DO 1210 J = 1, N
         X(J) = ONE - DFLOAT(J)*H
 1210    CONTINUE
      GO TO 1500
C
C     BROYDEN TRIDIAGONAL AND BANDED FUNCTIONS.
C
 1300 CONTINUE
 1400 CONTINUE
      DO 1410 J = 1, N
         X(J) = -ONE
 1410    CONTINUE
C
C     COMPUTE MULTIPLE OF INITIAL POINT.
C
 1500 CONTINUE
      IF (FACTOR .EQ. ONE) GO TO 1540
      IF (NPROB .EQ. 6) GO TO 1520
      DO 1510 J = 1, N
         X(J) = FACTOR*X(J)
 1510    CONTINUE
      GO TO 1540
 1520 CONTINUE
      DO 1530 J = 1, N
         X(J) = FACTOR
 1530    CONTINUE
 1540 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE INITPT.
C
      END
      SUBROUTINE VECFCN(N,X,FVEC,NPROB)
      INTEGER N,NPROB
      DOUBLE PRECISION X(N),FVEC(N)
C     **********
C
C     SUBROUTINE VECFCN
C
C     THIS SUBROUTINE DEFINES FOURTEEN TEST FUNCTIONS. THE FIRST
C     FIVE TEST FUNCTIONS ARE OF DIMENSIONS 2,4,2,4,3, RESPECTIVELY,
C     WHILE THE REMAINING TEST FUNCTIONS ARE OF VARIABLE DIMENSION
C     N FOR ANY N GREATER THAN OR EQUAL TO 1 (PROBLEM 6 IS AN
C     EXCEPTION TO THIS, SINCE IT DOES NOT ALLOW N = 1).
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE VECFCN(N,X,FVEC,NPROB)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER VARIABLE.
C
C       X IS A LINEAR ARRAY OF LENGTH N.
C
C       FVEC IS A LINEAR ARRAY OF LENGTH N. ON OUTPUT FVEC
C         CONTAINS THE NPROB FUNCTION VECTOR EVALUATED AT X.
C
C       NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 14.
C
C     SUBPROGRAMS REQUIRED
C
C       FORTRAN-SUPPLIED ... DATAN,DCOS,DEXP,DSIGN,DSIN,DSQRT,
C                            MAX0,MIN0
C
C     MINPACK. VERSION OF DECEMBER 1977.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      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                 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
C
C     PROBLEM SELECTOR.
C
      GO TO (100,200,300,400,500,600,700,800,900,1000,
     1       1100,1200,1300,1400), NPROB
C
C     ROSENBROCK FUNCTION.
C
  100 CONTINUE
      FVEC(1) = ONE - X(1)
      FVEC(2) = TEN*(X(2) - X(1)**2)
      GO TO 1500
C
C     POWELL SINGULAR FUNCTION.
C
  200 CONTINUE
      FVEC(1) = X(1) + TEN*X(2)
      FVEC(2) = DSQRT(FIVE)*(X(3) - X(4))
      FVEC(3) = (X(2) - TWO*X(3))**2
      FVEC(4) = DSQRT(TEN)*(X(1) - X(4))**2
      GO TO 1500
C
C     POWELL BADLY SCALED FUNCTION.
C
  300 CONTINUE
      FVEC(1) = C1*X(1)*X(2) - ONE
      FVEC(2) = DEXP(-X(1)) + DEXP(-X(2)) - C2
      GO TO 1500
C
C     WOOD FUNCTION.
C
  400 CONTINUE
      TEMP1 = X(2) - X(1)**2
      TEMP2 = X(4) - X(3)**2
      FVEC(1) = -C3*X(1)*TEMP1 - (ONE - X(1))
      FVEC(2) = C3*TEMP1 + C4*(X(2) - ONE) + C5*(X(4) - ONE)
      FVEC(3) = -C6*X(3)*TEMP2 - (ONE - X(3))
      FVEC(4) = C6*TEMP2 + C4*(X(4) - ONE) + C5*(X(2) - ONE)
      GO TO 1500
C
C     HELICAL VALLEY FUNCTION.
C
  500 CONTINUE
      TPI = EIGHT*DATAN(ONE)
      TEMP1 = DSIGN(C7,X(2))
      IF (X(1) .GT. ZERO) TEMP1 = DATAN(X(2)/X(1))/TPI
      IF (X(1) .LT. ZERO) TEMP1 = DATAN(X(2)/X(1))/TPI + C8
      TEMP2 = DSQRT(X(1)**2+X(2)**2)
      FVEC(1) = TEN*(X(3) - TEN*TEMP1)
      FVEC(2) = TEN*(TEMP2 - ONE)
      FVEC(3) = X(3)
      GO TO 1500
C
C     WATSON FUNCTION.
C
  600 CONTINUE
      DO 610 K = 1, N
         FVEC(K) = ZERO
  610    CONTINUE
      DO 650 I = 1, 29
         TI = DFLOAT(I)/C9
         SUM1 = ZERO
         TEMP = ONE
         DO 620 J = 2, N
            SUM1 = SUM1 + DFLOAT(J-1)*TEMP*X(J)
            TEMP = TI*TEMP
  620       CONTINUE
         SUM2 = ZERO
         TEMP = ONE
         DO 630 J = 1, N
```

```
               SUM2 = SUM2 + TEMP*X(J)
               TEMP = TI*TEMP
  630          CONTINUE
            TEMP1 = SUM1 - SUM2**2 - ONE
            TEMP2 = TWO*TI*SUM2
            TEMP = ONE/TI
            DO 640 K = 1, N
               FVEC(K) = FVEC(K) + TEMP*(DFLOAT(K-1) - TEMP2)*TEMP1
               TEMP = TI*TEMP
  640          CONTINUE
  650       CONTINUE
         TEMP = X(2) - X(1)**2 - ONE
         FVEC(1) = FVEC(1) + X(1)*(ONE - TWO*TEMP)
         FVEC(2) = FVEC(2) + TEMP
         GO TO 1500
C
C        CHEBYQUAD FUNCTION.
C
  700 CONTINUE
         DO 710 K = 1, N
            FVEC(K) = ZERO
  710       CONTINUE
         DO 730 J = 1, N
            TEMP1 = ONE
            TEMP2 = TWO*X(J) - ONE
            TEMP = TWO*TEMP2
            DO 720 I = 1, N
               FVEC(I) = FVEC(I) + TEMP2
               TI = TEMP*TEMP2 - TEMP1
               TEMP1 = TEMP2
               TEMP2 = TI
  720          CONTINUE
  730       CONTINUE
         TK = ONE/DFLOAT(N)
         IEV = -1
         DO 740 K = 1, N
            FVEC(K) = TK*FVEC(K)
            IF (IEV .GT. 0) FVEC(K) = FVEC(K) + ONE/(DFLOAT(K)**2 - ONE)
            IEV = -IEV
  740       CONTINUE
         GO TO 1500
C
C        BROWN ALMOST-LINEAR FUNCTION.
C
  800 CONTINUE
         SUM = -DFLOAT(N+1)
         PROD = ONE
         DO 810 J = 1, N
            SUM = SUM + X(J)
            PROD = X(J)*PROD
  810       CONTINUE
         DO 820 K = 1, N
            FVEC(K) = X(K) + SUM
  820       CONTINUE
         FVEC(N) = PROD - ONE
         GO TO 1500
C
C        DISCRETE BOUNDARY VALUE FUNCTION.
C
```

```
  900 CONTINUE
      H = ONE/DFLOAT(N+1)
      DO 910 K = 1, N
         TEMP = (X(K) + DFLOAT(K)*H + ONE)**3
         TEMP1 = ZERO
         IF (K .NE. 1) TEMP1 = X(K-1)
         TEMP2 = ZERO
         IF (K .NE. N) TEMP2 = X(K+1)
         FVEC(K) = TWO*X(K) - TEMP1 - TEMP2 + TEMP*H**2/TWO
  910    CONTINUE
      GO TO 1500
C
C     DISCRETE INTEGRAL EQUATION FUNCTION.
C
 1000 CONTINUE
      H = ONE/DFLOAT(N+1)
      DO 1040 K = 1, N
         TK = DFLOAT(K)*H
         SUM1 = ZERO
         DO 1010 J = 1, K
            TJ = DFLOAT(J)*H
            TEMP = (X(J) + TJ + ONE)**3
            SUM1 = SUM1 + TJ*TEMP
 1010       CONTINUE
         SUM2 = ZERO
         KP1 = K + 1
         IF (N .LT. KP1) GO TO 1030
         DO 1020 J = KP1, N
            TJ = DFLOAT(J)*H
            TEMP = (X(J) + TJ + ONE)**3
            SUM2 = SUM2 + (ONE - TJ)*TEMP
 1020       CONTINUE
 1030    CONTINUE
         FVEC(K) = X(K) + H*((ONE - TK)*SUM1 + TK*SUM2)/TWO
 1040    CONTINUE
      GO TO 1500
C
C     TRIGONOMETRIC FUNCTION.
C
 1100 CONTINUE
      SUM = ZERO
      DO 1110 J = 1, N
         FVEC(J) = DCOS(X(J))
         SUM = SUM + FVEC(J)
 1110    CONTINUE
      DO 1120 K = 1, N
         FVEC(K) = DFLOAT(N+K) - DSIN(X(K)) - SUM - DFLOAT(K)*FVEC(K)
 1120    CONTINUE
      GO TO 1500
C
C     VARIABLY DIMENSIONED FUNCTION.
C
 1200 CONTINUE
      SUM = ZERO
      DO 1210 J = 1, N
         SUM = SUM + DFLOAT(J)*(X(J) - ONE)
 1210    CONTINUE
      TEMP = SUM*(ONE + TWO*SUM**2)
      DO 1220 K = 1, N
```

```
            FVEC(K) = X(K) - ONE + DFLOAT(K)*TEMP
 1220       CONTINUE
         GO TO 1500
C
C        BROYDEN TRIDIAGONAL FUNCTION.
C
 1300 CONTINUE
         DO 1310 K = 1, N
            TEMP = (THREE - TWO*X(K))*X(K)
            TEMP1 = ZERO
            IF (K .NE. 1) TEMP1 = X(K-1)
            TEMP2 = ZERO
            IF (K .NE. N) TEMP2 = X(K+1)
            FVEC(K) = TEMP - TEMP1 - TWO*TEMP2 + ONE
 1310       CONTINUE
         GO TO 1500
C
C        BROYDEN BANDED FUNCTION.
C
 1400 CONTINUE
         ML = 5
         MU = 1
         DO 1420 K = 1, N
            K1 = MAX0(1,K-ML)
            K2 = MIN0(K+MU,N)
            TEMP = ZERO
            DO 1410 J = K1, K2
               IF (J .EQ. K) GO TO 1410
               TEMP = TEMP + X(J)*(ONE + X(J))
 1410          CONTINUE
            FVEC(K) = X(K)*(TWO + FIVE*X(K)**2) + ONE - TEMP
 1420       CONTINUE
 1500 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE VECFCN.
C
      END
      SUBROUTINE COMFCN(N,K,X,FCNK,NPROB)
      INTEGER N,K,NPROB
      DOUBLE PRECISION FCNK
      DOUBLE PRECISION X(N)
C     **********
C
C     SUBROUTINE COMFCN
C
C     THIS SUBROUTINE DEFINES FOURTEEN TEST FUNCTIONS. THE FIRST
C     FIVE TEST FUNCTIONS ARE OF DIMENSIONS 2,4,2,4,3, RESPECTIVELY,
C     WHILE THE REMAINING TEST FUNCTIONS ARE OF VARIABLE DIMENSION
C     N FOR ANY N GREATER THAN OR EQUAL TO 1 (PROBLEM 6 IS AN
C     EXCEPTION TO THIS, SINCE IT DOES NOT ALLOW N = 1).
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE COMFCN(N,K,X,FCNK,NPROB)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER VARIABLE.
C
C       K IS A POSITIVE INTEGER VARIABLE NOT GREATER THAN N.
C
C       X IS A LINEAR ARRAY OF LENGTH N.
C
C       FCNK IS A REAL VARIABLE WHICH ON OUTPUT CONTAINS THE VALUE OF
C         THE K-TH COMPONENT OF THE NPROB FUNCTION EVALUATED AT X.
C
C       NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 14.
C
C     SUBPROGRAMS REQUIRED
C
C       FORTRAN-SUPPLIED ... DATAN,DCOS,DEXP,DSIGN,DSIN,DSQRT,
C                            MAX0,MIN0,MOD
C
C     MINPACK. VERSION OF SEPTEMBER 1977.
C     BURTON S. GARBOW,KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      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
C
C     PROBLEM SELECTOR.
C
      GO TO (100,200,300,400,500,600,700,800,900,1000,
     1       1100,1200,1300,1400),NPROB
C
C     ROSENBROCK FUNCTION.
C
```

```
  100 CONTINUE
      IF (K .EQ. 1) FCNK = ONE - X(1)
      IF (K .EQ. 2) FCNK = TEN*(X(2) - X(1)**2)
      GO TO 1500
C
C     POWELL SINGULAR FUNCTION.
C
  200 CONTINUE
      IF (K .EQ. 1) FCNK = X(1) + TEN*X(2)
      IF (K .EQ. 2) FCNK = DSQRT(FIVE)*(X(3) - X(4))
      IF (K .EQ. 3) FCNK = (X(2) - TWO*X(3))**2
      IF (K .EQ. 4) FCNK = DSQRT(TEN)*(X(1) - X(4))**2
      GO TO 1500
C
C     POWELL BADLY SCALED FUNCTION.
C
  300 CONTINUE
      IF (K .EQ. 1) FCNK = C1*X(1)*X(2) - ONE
      IF (K .EQ. 2) FCNK = DEXP(-X(1)) + DEXP(-X(2)) - C2
      GO TO 1500
C
C     WOOD FUNCTION.
C
  400 CONTINUE
      TEMP1 = X(2) - X(1)**2
      TEMP2 = X(4) - X(3)**2
      IF (K .EQ. 1) FCNK = -C3*X(1)*TEMP1 - (ONE - X(1))
      IF (K .EQ. 2) FCNK = C3*TEMP1 + C4*(X(2) - ONE) + C5*(X(4) - ONE)
      IF (K .EQ. 3) FCNK = -C6*X(3)*TEMP2 - (ONE - X(3))
      IF (K .EQ. 4) FCNK = C6*TEMP2 + C4*(X(4) - ONE) + C5*(X(2) - ONE)
      GO TO 1500
C
C     HELICAL VALLEY FUNCTION.
C
  500 CONTINUE
      IF (K .NE. 1) GO TO 510
      TPI = EIGHT*DATAN(ONE)
      TEMP1 = DSIGN(C7,X(2))
      IF (X(1) .GT. ZERO) TEMP1 = DATAN(X(2)/X(1))/TPI
      IF (X(1) .LT. ZERO) TEMP1 = DATAN(X(2)/X(1))/TPI + C8
      FCNK = TEN*(X(3) - TEN*TEMP1)
  510 CONTINUE
      IF (K .EQ. 2) FCNK = TEN*(DSQRT(X(1)**2+X(2)**2) - ONE)
      IF (K .EQ. 3) FCNK = X(3)
      GO TO 1500
C
C     WATSON FUNCTION.
C
  600 CONTINUE
      FCNK = ZERO
      DO 630 I = 1, 29
         TI = DFLOAT(I)/C9
         SUM1 = ZERO
         TEMP = ONE
         DO 610 J = 2, N
            SUM1 = SUM1 + DFLOAT(J-1)*TEMP*X(J)
            TEMP = TI*TEMP
  610       CONTINUE
         SUM2 = ZERO
```

```
            TEMP = ONE
            DO 620 J = 1, N
               SUM2 = SUM2 + TEMP*X(J)
               TEMP = TI*TEMP
  620          CONTINUE
            TEMP1 = SUM1 - SUM2**2 - ONE
            TEMP2 = TWO*TI*SUM2
            FCNK = FCNK + TI**(K-2)*(DFLOAT(K-1) - TEMP2)*TEMP1
  630       CONTINUE
         TEMP = X(2) - X(1)**2 - ONE
         IF (K .EQ. 1) FCNK = FCNK + X(1)*(ONE - TWO*TEMP)
         IF (K .EQ. 2) FCNK = FCNK + TEMP
         GO TO 1500
C
C        CHEBYQUAD FUNCTION.
C
  700 CONTINUE
         SUM = ZERO
         DO 730 J = 1, N
            TEMP1 = ONE
            TEMP2 = TWO*X(J) - ONE
            TEMP = TWO*TEMP2
            IF (K .LT. 2) GO TO 720
            DO 710 I = 2, K
               TI = TEMP*TEMP2 - TEMP1
               TEMP1 = TEMP2
               TEMP2 = TI
  710          CONTINUE
  720       CONTINUE
            SUM = SUM + TEMP2
  730       CONTINUE
         FCNK = SUM/DFLOAT(N)
         IF (MOD(K,2) .EQ. 0) FCNK = FCNK + ONE/(DFLOAT(K)**2 - ONE)
         GO TO 1500
C
C        BROWN ALMOST-LINEAR FUNCTION.
C
  800 CONTINUE
         IF (K .EQ. N) GO TO 820
         SUM = -DFLOAT(N+1)
         DO 810 J = 1, N
            SUM = SUM + X(J)
  810       CONTINUE
         FCNK = X(K) + SUM
         GO TO 840
  820 CONTINUE
         PROD = ONE
         DO 830 J = 1, N
            PROD = X(J)*PROD
  830       CONTINUE
         FCNK = PROD - ONE
  840 CONTINUE
         GO TO 1500
C
C        DISCRETE BOUNDARY VALUE FUNCTION.
C
  900 CONTINUE
         H = ONE/DFLOAT(N+1)
         TEMP = (X(K) + DFLOAT(K)*H + ONE)**3
```

```
      TEMP1 = ZERO
      IF (K .NE. 1) TEMP1 = X(K-1)
      TEMP2 = ZERO
      IF (K .NE. N) TEMP2 = X(K+1)
      FCNK = TWO*X(K) - TEMP1 - TEMP2 + TEMP*H**2/TWO
      GO TO 1500
C
C     DISCRETE INTEGRAL EQUATION FUNCTION.
C
 1000 CONTINUE
      H = ONE/DFLOAT(N+1)
      TK = DFLOAT(K)*H
      SUM1 = ZERO
      DO 1010 J = 1, K
         TJ = DFLOAT(J)*H
         TEMP = (X(J) + TJ + ONE)**3
         SUM1 = SUM1 + TJ*TEMP
 1010    CONTINUE
      SUM2 = ZERO
      KP1 = K + 1
      IF (N .LT. KP1) GO TO 1030
      DO 1020 J = KP1, N
         TJ = DFLOAT(J)*H
         TEMP = (X(J) + TJ + ONE)**3
         SUM2 = SUM2 + (ONE - TJ)*TEMP
 1020    CONTINUE
 1030 CONTINUE
      FCNK = X(K) + H*((ONE - TK)*SUM1 + TK*SUM2)/TWO
      GO TO 1500
C
C     TRIGONOMETRIC FUNCTION.
C
 1100 CONTINUE
      SUM = ZERO
      DO 1110 J = 1, N
         SUM = SUM + DCOS(X(J))
 1110    CONTINUE
      FCNK = DFLOAT(N+K) - DSIN(X(K)) - SUM - DFLOAT(K)*DCOS(X(K))
      GO TO 1500
C
C     VARIABLY DIMENSIONED FUNCTION.
C
 1200 CONTINUE
      SUM = ZERO
      DO 1210 J = 1, N
         SUM = SUM + DFLOAT(J)*(X(J) - ONE)
 1210    CONTINUE
      TEMP = SUM*(ONE + TWO*SUM**2)
      FCNK = X(K) - ONE + DFLOAT(K)*TEMP
      GO TO 1500
C
C     BROYDEN TRIDIAGONAL FUNCTION.
C
 1300 CONTINUE
      TEMP = (THREE - TWO*X(K))*X(K)
      TEMP1 = ZERO
      IF (K .NE. 1) TEMP1 = X(K-1)
      TEMP2 = ZERO
      IF (K .NE. N) TEMP2 = X(K+1)
      FCNK = TEMP - TEMP1 - TWO*TEMP2 + ONE
      GO TO 1500
C
C     BROYDEN BANDED FUNCTION.
C
 1400 CONTINUE
      ML = 5
      MU = 1
      K1 = MAX0(1,K-ML)
      K2 = MIN0(K+MU,N)
      TEMP = ZERO
      DO 1410 J = K1, K2
         IF (J .EQ. K) GO TO 1410
         TEMP = TEMP + X(J)*(ONE + X(J))
 1410    CONTINUE
      FCNK = X(K)*(TWO + FIVE*X(K)**2) + ONE - TEMP
 1500 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE COMFCN.
C
      END
      SUBROUTINE INITPT(N,X,NPROB,FACTOR)
      INTEGER N,NPROB
      DOUBLE PRECISION FACTOR
      DOUBLE PRECISION X(N)
C     **********
C
C     SUBROUTINE INITPT
C
C     THIS SUBROUTINE SPECIFIES THE STANDARD STARTING POINTS FOR THE
C     FUNCTIONS DEFINED BY SUBROUTINE SSQFCN. THE SUBROUTINE RETURNS
C     IN X A MULTIPLE (FACTOR) OF THE STANDARD STARTING POINT. FOR
C     THE 11TH FUNCTION THE STANDARD STARTING POINT IS ZERO, SO IN
C     THIS CASE, IF FACTOR IS NOT UNITY, THEN THE SUBROUTINE RETURNS
C     THE VECTOR  X(J) = FACTOR, J=1,...,N.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE INITPT(N,X,NPROB,FACTOR)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER VARIABLE.
C
C       X IS A LINEAR ARRAY OF LENGTH N. ON OUTPUT X CONTAINS THE
C         STANDARD STARTING POINT FOR PROBLEM NPROB MULTIPLIED BY
C         FACTOR.
C
C       NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
C
C       FACTOR SPECIFIES THE MULTIPLE OF THE STANDARD STARTING
C         POINT. IF FACTOR IS UNITY, NO MULTIPLICATION IS PERFORMED.
C
C     MINPACK. VERSION OF OCTOBER 1977.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      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.D0,5.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
C
C     SELECTION OF INITIAL POINT.
C
      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,
     1       1200,1300,1400,1500,1600,1700,1800), NPROB
C
C     LINEAR FUNCTION - FULL RANK OR RANK 1.
C
  100 CONTINUE
  200 CONTINUE
  300 CONTINUE
```

```
      DO 310 J = 1, N
         X(J) = ONE
  310    CONTINUE
      GO TO 1900
C
C     ROSENBROCK FUNCTION.
C
  400 CONTINUE
      X(1) = -C1
      X(2) = ONE
      GO TO 1900
C
C     HELICAL VALLEY FUNCTION.
C
  500 CONTINUE
      X(1) = -ONE
      X(2) = ZERO
      X(3) = ZERO
      GO TO 1900
C
C     POWELL SINGULAR FUNCTION.
C
  600 CONTINUE
      X(1) = THREE
      X(2) = -ONE
      X(3) = ZERO
      X(4) = ONE
      GO TO 1900
C
C     FREUDENSTEIN AND ROTH FUNCTION.
C
  700 CONTINUE
      X(1) = HALF
      X(2) = -TWO
      GO TO 1900
C
C     BARD FUNCTION.
C
  800 CONTINUE
      X(1) = ONE
      X(2) = ONE
      X(3) = ONE
      GO TO 1900
C
C     KOWALIK AND OSBORNE FUNCTION.
C
  900 CONTINUE
      X(1) = C2
      X(2) = C3
      X(3) = C4
      X(4) = C3
      GO TO 1900
C
C     MEYER FUNCTION.
C
 1000 CONTINUE
      X(1) = C5
      X(2) = C6
      X(3) = C7
      GO TO 1900
C
C     WATSON FUNCTION.
C
 1100 CONTINUE
      DO 1110 J = 1, N
         X(J) = ZERO
 1110    CONTINUE
      GO TO 1900
C
C     BOX 3-DIMENSIONAL FUNCTION.
C
 1200 CONTINUE
      X(1) = ZERO
      X(2) = TEN
      X(3) = TWENTY
      GO TO 1900
C
C     JENNRICH AND SAMPSON FUNCTION.
C
 1300 CONTINUE
      X(1) = C8
      X(2) = C9
      GO TO 1900
C
C     BROWN AND DENNIS FUNCTION.
C
 1400 CONTINUE
      X(1) = TWNTF
      X(2) = FIVE
      X(3) = -FIVE
      X(4) = -ONE
      GO TO 1900
C
C     CHEBYQUAD FUNCTION.
C
 1500 CONTINUE
      H = ONE/DFLOAT(N+1)
      DO 1510 J = 1, N
         X(J) = DFLOAT(J)*H
 1510    CONTINUE
      GO TO 1900
C
C     BROWN ALMOST-LINEAR FUNCTION.
C
 1600 CONTINUE
      DO 1610 J = 1, N
         X(J) = HALF
 1610    CONTINUE
      GO TO 1900
C
C     OSBORNE 1 FUNCTION.
C
 1700 CONTINUE
      X(1) = HALF
      X(2) = C10
      X(3) = -ONE
      X(4) = C11
      X(5) = C5
      GO TO 1900
C
C     OSBORNE 2 FUNCTION.
C
 1800 CONTINUE
      X(1) = C12
      X(2) = C13
      X(3) = C13
      X(4) = C14
      X(5) = C15
      X(6) = THREE
      X(7) = FIVE
      X(8) = SEVEN
      X(9) = TWO
      X(10) = C16
      X(11) = C17
C
C     COMPUTE MULTIPLE OF INITIAL POINT.
C
 1900 CONTINUE
      IF (FACTOR .EQ. ONE) GO TO 1940
      IF (NPROB .EQ. 11) GO TO 1920
      DO 1910 J = 1, N
         X(J) = FACTOR*X(J)
 1910    CONTINUE
      GO TO 1940
 1920 CONTINUE
      DO 1930 J = 1, N
         X(J) = FACTOR
 1930    CONTINUE
 1940 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE INITPT.
C
      END
      SUBROUTINE SSQFCN(M,N,X,FVEC,NPROB)
      INTEGER M,N,NPROB
      DOUBLE PRECISION X(N),FVEC(M)
C     **********
C
C     SUBROUTINE SSQFCN
C
C     THIS SUBROUTINE DEFINES THE FUNCTIONS OF EIGHTEEN NONLINEAR
C     LEAST SQUARES PROBLEMS. THE ALLOWABLE VALUES OF (M,N) FOR
C     FUNCTIONS 1,2 AND 3 ARE VARIABLE BUT WITH M .GE. N.
C     FOR FUNCTIONS 4,5,6,7,8,9 AND 10 THE VALUES OF (M,N) ARE
C     (2,2),(3,3),(4,4),(2,2),(15,3),(11,4) AND (16,3), RESPECTIVELY.
C     FUNCTION 11 (WATSON) HAS M = 31 WITH N USUALLY 6 OR 9.
C     HOWEVER, ANY N, N = 2,...,31, IS PERMITTED.
C     FUNCTIONS 12,13 AND 14 HAVE N = 3,2 AND 4, RESPECTIVELY, BUT
C     ALLOW ANY M .GE. N, WITH THE USUAL CHOICES BEING 10,10 AND 20.
C     FUNCTION 15 (CHEBYQUAD) ALLOWS M AND N VARIABLE WITH M .GE. N.
C     FUNCTION 16 (BROWN) ALLOWS N VARIABLE WITH M = N.
C     FOR FUNCTIONS 17 AND 18, THE VALUES OF (M,N) ARE
C     (33,5) AND (65,11), RESPECTIVELY.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE SSQFCN(M,N,X,FVEC,NPROB)
C
C     WHERE
C
C       M AND N ARE POSITIVE INTEGER VARIABLES. N MUST NOT EXCEED M.
C
C       X IS A LINEAR ARRAY OF LENGTH N.
C
C       FVEC IS A LINEAR ARRAY OF LENGTH M. ON OUTPUT FVEC
C         CONTAINS THE NPROB FUNCTION EVALUATED AT X.
C
C       NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
C
C     SUBPROGRAMS REQUIRED
C
C       FORTRAN-SUPPLIED ... DATAN,DCOS,DEXP,DSIN,DSQRT,DSIGN
C
C     MINPACK. VERSION OF OCTOBER 1977.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,IEV,IVAR,J,NM1
      DOUBLE PRECISION C13,C14,C29,C45,DIV,DX,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,
     3     3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.10D0,4.39D0/
      DATA Y2(1),Y2(2),Y2(3),Y2(4),Y2(5),Y2(6),
     1     Y2(7),Y2(8),Y2(9),Y2(10),Y2(11)
     2    /1.957D-1,1.947D-1,1.735D-1,1.600D-1,8.44D-2,6.27D-2,
     3     4.56D-2,3.42D-2,3.23D-2,2.35D-2,2.46D-2/
      DATA Y3(1),Y3(2),Y3(3),Y3(4),Y3(5),Y3(6),Y3(7),Y3(8),Y3(9),
     1     Y3(10),Y3(11),Y3(12),Y3(13),Y3(14),Y3(15),Y3(16)
     2    /3.478D4,2.861D4,2.365D4,1.963D4,1.637D4,1.372D4,1.154D4,
     3     9.744D3,8.261D3,7.03D3,6.005D3,5.147D3,4.427D3,3.82D3,
     4     3.307D3,2.872D3/
      DATA Y4(1),Y4(2),Y4(3),Y4(4),Y4(5),Y4(6),Y4(7),Y4(8),Y4(9),
     1     Y4(10),Y4(11),Y4(12),Y4(13),Y4(14),Y4(15),Y4(16),Y4(17),
     2     Y4(18),Y4(19),Y4(20),Y4(21),Y4(22),Y4(23),Y4(24),Y4(25),
     3     Y4(26),Y4(27),Y4(28),Y4(29),Y4(30),Y4(31),Y4(32),Y4(33)
     4    /8.44D-1,9.08D-1,9.32D-1,9.36D-1,9.25D-1,9.08D-1,8.81D-1,
     5     8.50D-1,8.18D-1,7.84D-1,7.51D-1,7.18D-1,6.85D-1,6.58D-1,
     6     6.28D-1,6.03D-1,5.80D-1,5.58D-1,5.38D-1,5.22D-1,
     7     5.06D-1,4.90D-1,4.78D-1,4.67D-1,4.57D-1,4.48D-1,4.38D-1,
     8     4.31D-1,4.24D-1,4.20D-1,4.14D-1,4.11D-1,4.06D-1/
      DATA Y5(1),Y5(2),Y5(3),Y5(4),Y5(5),Y5(6),Y5(7),Y5(8),Y5(9),
     1     Y5(10),Y5(11),Y5(12),Y5(13),Y5(14),Y5(15),Y5(16),Y5(17),
     2     Y5(18),Y5(19),Y5(20),Y5(21),Y5(22),Y5(23),Y5(24),Y5(25),
     3     Y5(26),Y5(27),Y5(28),Y5(29),Y5(30),Y5(31),Y5(32),Y5(33),
     4     Y5(34),Y5(35),Y5(36),Y5(37),Y5(38),Y5(39),Y5(40),Y5(41),
     5     Y5(42),Y5(43),Y5(44),Y5(45),Y5(46),Y5(47),Y5(48),Y5(49),
     6     Y5(50),Y5(51),Y5(52),Y5(53),Y5(54),Y5(55),Y5(56),Y5(57),
     7     Y5(58),Y5(59),Y5(60),Y5(61),Y5(62),Y5(63),Y5(64),Y5(65)
     8    /1.366D0,1.191D0,1.112D0,1.013D0,9.91D-1,8.85D-1,
     9     8.31D-1,8.47D-1,7.86D-1,7.25D-1,7.46D-1,6.79D-1,6.08D-1,
     A     6.55D-1,6.16D-1,6.06D-1,6.02D-1,6.26D-1,6.51D-1,7.24D-1,
     B     6.49D-1,6.49D-1,6.94D-1,6.44D-1,6.24D-1,6.61D-1,6.12D-1,
     C     5.58D-1,5.33D-1,4.95D-1,5.00D-1,4.23D-1,3.95D-1,3.75D-1,
     D     3.72D-1,3.91D-1,3.96D-1,4.05D-1,4.28D-1,4.29D-1,5.23D-1,
     E     5.62D-1,6.07D-1,6.53D-1,6.72D-1,7.08D-1,6.33D-1,6.68D-1,
     F     6.45D-1,6.32D-1,5.91D-1,5.59D-1,5.97D-1,6.25D-1,7.39D-1,
     G     7.10D-1,7.29D-1,7.20D-1,6.36D-1,5.81D-1,4.28D-1,2.92D-1,
     H     1.62D-1,9.8D-2,5.4D-2/
      DFLOAT(IVAR) = IVAR
C
C     FUNCTION ROUTINE SELECTOR.
C
      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,
     1       1200,1300,1400,1500,1600,1700,1800),NPROB
C
C     LINEAR FUNCTION - FULL RANK.
C
  100 CONTINUE
      SUM = ZERO
      DO 110 J = 1, N
         SUM = SUM + X(J)
  110    CONTINUE
      TEMP = TWO*SUM/DFLOAT(M) + ONE
      DO 120 I = 1, M
         FVEC(I) = -TEMP
         IF (I .LE. N) FVEC(I) = FVEC(I) + X(I)
  120    CONTINUE
      GO TO 1900
C
```

```
C     LINEAR FUNCTION - RANK 1.
C
  200 CONTINUE
      SUM = ZERO
      DO 210 J = 1, N
         SUM = SUM + DFLOAT(J)*X(J)
  210    CONTINUE
      DO 220 I = 1, M
         FVEC(I) = DFLOAT(I)*SUM - ONE
  220    CONTINUE
      GO TO 1900
C
C     LINEAR FUNCTION - RANK 1 WITH ZERO COLUMNS AND ROWS.
C
  300 CONTINUE
      SUM = ZERO
      NM1 = N - 1
      IF (NM1 .LT. 2) GO TO 320
      DO 310 J = 2, NM1
         SUM = SUM + DFLOAT(J)*X(J)
  310    CONTINUE
  320 CONTINUE
      DO 330 I = 1, M
         FVEC(I) = DFLOAT(I-1)*SUM - ONE
  330    CONTINUE
      FVEC(M) = -ONE
      GO TO 1900
C
C     ROSENBROCK FUNCTION.
C
  400 CONTINUE
      FVEC(1) = TEN*(X(2) - X(1)**2)
      FVEC(2) = ONE - X(1)
      GO TO 1900
C
C     HELICAL VALLEY FUNCTION.
C
  500 CONTINUE
      TPI = EIGHT*DATAN(ONE)
      TMP1 = DSIGN(ZP25,X(2))
      IF (X(1) .GT. ZERO) TMP1 = DATAN(X(2)/X(1))/TPI
      IF (X(1) .LT. ZERO) TMP1 = DATAN(X(2)/X(1))/TPI + ZP5
      TMP2 = DSQRT(X(1)**2+X(2)**2)
      FVEC(1) = TEN*(X(3) - TEN*TMP1)
      FVEC(2) = TEN*(TMP2 - ONE)
      FVEC(3) = X(3)
      GO TO 1900
C
C     POWELL SINGULAR FUNCTION.
C
  600 CONTINUE
      FVEC(1) = X(1) + TEN*X(2)
      FVEC(2) = DSQRT(FIVE)*(X(3) - X(4))
      FVEC(3) = (X(2) - TWO*X(3))**2
      FVEC(4) = DSQRT(TEN)*(X(1) - X(4))**2
      GO TO 1900
C
C     FREUDENSTEIN AND ROTH FUNCTION.
C
```

```
  700 CONTINUE
      FVEC(1) = -C13 + X(1) + ((FIVE - X(2))*X(2) - TWO)*X(2)
      FVEC(2) = -C29 + X(1) + ((ONE + X(2))*X(2) - C14)*X(2)
      GO TO 1900
C
C     BARD FUNCTION.
C
  800 CONTINUE
      DO 810 I = 1, 15
         TMP1 = DFLOAT(I)
         TMP2 = DFLOAT(16-I)
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         FVEC(I) = Y1(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
  810    CONTINUE
      GO TO 1900
C
C     KOWALIK AND OSBORNE FUNCTION.
C
  900 CONTINUE
      DO 910 I = 1, 11
         TMP1 = V(I)*(V(I) + X(2))
         TMP2 = V(I)*(V(I) + X(3)) + X(4)
         FVEC(I) = Y2(I) - X(1)*TMP1/TMP2
  910    CONTINUE
      GO TO 1900
C
C     MEYER FUNCTION.
C
 1000 CONTINUE
      DO 1010 I = 1, 16
         TEMP = FIVE*DFLOAT(I) + C45 + X(3)
         TMP1 = X(2)/TEMP
         TMP2 = DEXP(TMP1)
         FVEC(I) = X(1)*TMP2 - Y3(I)
 1010    CONTINUE
      GO TO 1900
C
C     WATSON FUNCTION.
C
 1100 CONTINUE
      DO 1130 I = 1, 29
         DIV = DFLOAT(I)/C29
         S1 = ZERO
         DX = ONE
         DO 1110 J = 2, N
            S1 = S1 + DFLOAT(J-1)*DX*X(J)
            DX = DIV*DX
 1110       CONTINUE
         S2 = ZERO
         DX = ONE
         DO 1120 J = 1, N
            S2 = S2 + DX*X(J)
            DX = DIV*DX
 1120       CONTINUE
         FVEC(I) = S1 - S2**2 - ONE
 1130    CONTINUE
      FVEC(30) = X(1)
      FVEC(31) = X(2) - X(1)**2 - ONE
      GO TO 1900
C
C     BOX 3-DIMENSIONAL FUNCTION.
C
 1200 CONTINUE
      DO 1210 I = 1, M
         TEMP = DFLOAT(I)
         TMP1 = TEMP/TEN
         FVEC(I) = DEXP(-TMP1*X(1)) - DEXP(-TMP1*X(2))
     1             + (DEXP(-TEMP) - DEXP(-TMP1))*X(3)
 1210    CONTINUE
      GO TO 1900
C
C     JENNRICH AND SAMPSON FUNCTION.
C
 1300 CONTINUE
      DO 1310 I = 1, M
         TEMP = DFLOAT(I)
         FVEC(I) = TWO + TWO*TEMP - DEXP(TEMP*X(1)) - DEXP(TEMP*X(2))
 1310    CONTINUE
      GO TO 1900
C
C     BROWN AND DENNIS FUNCTION.
C
 1400 CONTINUE
      DO 1410 I = 1, M
         TEMP = DFLOAT(I)/FIVE
         TMP1 = X(1) + TEMP*X(2) - DEXP(TEMP)
         TMP2 = X(3) + DSIN(TEMP)*X(4) - DCOS(TEMP)
         FVEC(I) = TMP1**2 + TMP2**2
 1410    CONTINUE
      GO TO 1900
C
C     CHEBYQUAD FUNCTION.
C
 1500 CONTINUE
      DO 1510 I = 1, M
         FVEC(I) = ZERO
 1510    CONTINUE
      DO 1530 J = 1, N
         TMP1 = ONE
         TMP2 = TWO*X(J) - ONE
         TEMP = TWO*TMP2
         DO 1520 I = 1, M
            FVEC(I) = FVEC(I) + TMP2
            TI = TEMP*TMP2 - TMP1
            TMP1 = TMP2
            TMP2 = TI
 1520       CONTINUE
 1530    CONTINUE
      DX = ONE/DFLOAT(N)
      IEV = -1
      DO 1540 I = 1, M
         FVEC(I) = DX*FVEC(I)
         IF (IEV .GT. 0) FVEC(I) = FVEC(I) + ONE/(DFLOAT(I)**2 - ONE)
         IEV = -IEV
 1540    CONTINUE
      GO TO 1900
C
```

```
C     BROWN ALMOST-LINEAR FUNCTION.
C
 1600 CONTINUE
      SUM = -DFLOAT(N+1)
      PROD = ONE
      DO 1610 J = 1, N
         SUM = SUM + X(J)
         PROD = X(J)*PROD
 1610    CONTINUE
      DO 1620 I = 1, N
         FVEC(I) = X(I) + SUM
 1620    CONTINUE
      FVEC(N) = PROD - ONE
      GO TO 1900
C
C     OSBORNE 1 FUNCTION.
C
 1700 CONTINUE
      DO 1710 I = 1, 33
         TEMP = TEN*DFLOAT(I-1)
         TMP1 = DEXP(-X(4)*TEMP)
         TMP2 = DEXP(-X(5)*TEMP)
         FVEC(I) = Y4(I) - (X(1) + X(2)*TMP1 + X(3)*TMP2)
 1710    CONTINUE
      GO TO 1900
C
C     OSBORNE 2 FUNCTION.
C
 1800 CONTINUE
      DO 1810 I = 1, 65
         TEMP = DFLOAT(I-1)/TEN
         TMP1 = DEXP(-X(5)*TEMP)
         TMP2 = DEXP(-X(6)*(TEMP - X(9))**2)
         TMP3 = DEXP(-X(7)*(TEMP - X(10))**2)
         TMP4 = DEXP(-X(8)*(TEMP - X(11))**2)
         FVEC(I) = Y5(I) - (X(1)*TMP1 + X(2)*TMP2
     1                      + X(3)*TMP3 + X(4)*TMP4)
 1810    CONTINUE
 1900 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE SSQFCN.
C
      END
      SUBROUTINE SSQJAC(M,N,X,FJAC,LDFJAC,NPROB)
      INTEGER M,N,LDFJAC,NPROB
      DOUBLE PRECISION X(N),FJAC(LDFJAC,N)
C     **********
C
C     SUBROUTINE SSQJAC
C
C     THIS SUBROUTINE DEFINES THE JACOBIAN MATRICES OF EIGHTEEN
C     NONLINEAR LEAST SQUARES PROBLEMS. THE PROBLEM DIMENSIONS ARE
C     AS DESCRIBED IN THE PROLOGUE COMMENTS OF SSQFCN.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE SSQJAC(M,N,X,FJAC,LDFJAC,NPROB)
C
C     WHERE
C
C       M AND N ARE POSITIVE INTEGER VARIABLES. N MUST NOT EXCEED M.
C
C       X IS A LINEAR ARRAY OF LENGTH N.
C
C       FJAC IS AN M BY N ARRAY. ON OUTPUT FJAC CONTAINS THE
C         JACOBIAN MATRIX OF THE NPROB FUNCTION EVALUATED AT X.
C
C       LDFJAC IS A POSITIVE INTEGER VARIABLE NOT LESS THAN M
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY FJAC.
C
C       NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
C
C     SUBPROGRAMS REQUIRED
C
C       FORTRAN-SUPPLIED ... DATAN,DCOS,DEXP,DSIN,DSQRT
C
C     MINPACK. VERSION OF OCTOBER 1977.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      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.D0,1.D0,2.D0,3.D0,4.D0,5.D0,8.D0,1.D1,
     2      1.4D1,2.D1,2.9D1,4.5D1,1.D2/
      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/
      DFLOAT(IVAR) = IVAR
C
C     JACOBIAN ROUTINE SELECTOR.
C
      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,
     1       1200,1300,1400,1500,1600,1700,1800), NPROB
C
C     LINEAR FUNCTION - FULL RANK.
C
```

```
  100 CONTINUE
      TEMP = TWO/DFLOAT(M)
      DO 120 J = 1, N
         DO 110 I = 1, M
            FJAC(I,J) = -TEMP
  110       CONTINUE
         FJAC(J,J) = FJAC(J,J) + ONE
  120    CONTINUE
      GO TO 1900
C
C     LINEAR FUNCTION - RANK 1.
C
  200 CONTINUE
      DO 220 J = 1, N
         DO 210 I = 1, M
            FJAC(I,J) = DFLOAT(I)*DFLOAT(J)
  210       CONTINUE
  220    CONTINUE
      GO TO 1900
C
C     LINEAR FUNCTION - RANK 1 WITH ZERO COLUMNS AND ROWS.
C
  300 CONTINUE
      DO 320 J = 1, N
         DO 310 I = 1, M
            FJAC(I,J) = ZERO
  310       CONTINUE
  320    CONTINUE
      NM1 = N - 1
      MM1 = M - 1
      IF (NM1 .LT. 2) GO TO 350
      DO 340 J = 2, NM1
         DO 330 I = 2, MM1
            FJAC(I,J) = DFLOAT(I-1)*DFLOAT(J)
  330       CONTINUE
  340    CONTINUE
  350 CONTINUE
      GO TO 1900
C
C     ROSENBROCK FUNCTION.
C
  400 CONTINUE
      FJAC(1,1) = -C20*X(1)
      FJAC(1,2) = TEN
      FJAC(2,1) = -ONE
      FJAC(2,2) = ZERO
      GO TO 1900
C
C     HELICAL VALLEY FUNCTION.
C
  500 CONTINUE
      TPI = EIGHT*DATAN(ONE)
      TEMP = X(1)**2 + X(2)**2
      TMP1 = TPI*TEMP
      TMP2 = DSQRT(TEMP)
      FJAC(1,1) = C100*X(2)/TMP1
      FJAC(1,2) = -C100*X(1)/TMP1
      FJAC(1,3) = TEN
      FJAC(2,1) = TEN*X(1)/TMP2
      FJAC(2,2) = TEN*X(2)/TMP2
      FJAC(2,3) = ZERO
      FJAC(3,1) = ZERO
      FJAC(3,2) = ZERO
      FJAC(3,3) = ONE
      GO TO 1900
C
C     POWELL SINGULAR FUNCTION.
C
  600 CONTINUE
      DO 620 J = 1, 4
         DO 610 I = 1, 4
            FJAC(I,J) = ZERO
  610       CONTINUE
  620    CONTINUE
      FJAC(1,1) = ONE
      FJAC(1,2) = TEN
      FJAC(2,3) = DSQRT(FIVE)
      FJAC(2,4) = -FJAC(2,3)
      FJAC(3,2) = TWO*(X(2) - TWO*X(3))
      FJAC(3,3) = -TWO*FJAC(3,2)
      FJAC(4,1) = TWO*DSQRT(TEN)*(X(1) - X(4))
      FJAC(4,4) = -FJAC(4,1)
      GO TO 1900
C
C     FREUDENSTEIN AND ROTH FUNCTION.
C
  700 CONTINUE
      FJAC(1,1) = ONE
      FJAC(1,2) = X(2)*(TEN - THREE*X(2)) - TWO
      FJAC(2,1) = ONE
      FJAC(2,2) = X(2)*(TWO + THREE*X(2)) - C14
      GO TO 1900
C
C     BARD FUNCTION.
C
  800 CONTINUE
      DO 810 I = 1, 15
         TMP1 = DFLOAT(I)
         TMP2 = DFLOAT(16-I)
         TMP3 = TMP1
         IF (I .GT. 8) TMP3 = TMP2
         TMP4 = (X(2)*TMP2 + X(3)*TMP3)**2
         FJAC(I,1) = -ONE
         FJAC(I,2) = TMP1*TMP2/TMP4
         FJAC(I,3) = TMP1*TMP3/TMP4
  810    CONTINUE
      GO TO 1900
C
C     KOWALIK AND OSBORNE FUNCTION.
C
  900 CONTINUE
      DO 910 I = 1, 11
         TMP1 = V(I)*(V(I) + X(2))
         TMP2 = V(I)*(V(I) + X(3)) + X(4)
         FJAC(I,1) = -TMP1/TMP2
         FJAC(I,2) = -V(I)*X(1)/TMP2
         FJAC(I,3) = FJAC(I,1)*FJAC(I,2)
         FJAC(I,4) = FJAC(I,3)/V(I)
  910    CONTINUE
      GO TO 1900
C
C     MEYER FUNCTION.
C
 1000 CONTINUE
      DO 1010 I = 1, 16
         TEMP = FIVE*DFLOAT(I) + C45 + X(3)
         TMP1 = X(2)/TEMP
         TMP2 = DEXP(TMP1)
         FJAC(I,1) = TMP2
         FJAC(I,2) = X(1)*TMP2/TEMP
         FJAC(I,3) = -TMP1*FJAC(I,2)
 1010    CONTINUE
      GO TO 1900
C
C     WATSON FUNCTION.
C
 1100 CONTINUE
      DO 1130 I = 1, 29
         DIV = DFLOAT(I)/C29
         S2 = ZERO
         DX = ONE
         DO 1110 J = 1, N
            S2 = S2 + DX*X(J)
            DX = DIV*DX
 1110       CONTINUE
         TEMP = TWO*DIV*S2
         DX = ONE/DIV
         DO 1120 J = 1, N
            FJAC(I,J) = DX*(DFLOAT(J-1) - TEMP)
            DX = DIV*DX
 1120       CONTINUE
 1130    CONTINUE
      DO 1150 J = 1, N
         DO 1140 I = 30, 31
            FJAC(I,J) = ZERO
 1140       CONTINUE
 1150    CONTINUE
      FJAC(30,1) = ONE
      FJAC(31,1) = -TWO*X(1)
      FJAC(31,2) = ONE
      GO TO 1900
C
C     BOX 3-DIMENSIONAL FUNCTION.
C
 1200 CONTINUE
      DO 1210 I = 1, M
         TEMP = DFLOAT(I)
         TMP1 = TEMP/TEN
         FJAC(I,1) = -TMP1*DEXP(-TMP1*X(1))
         FJAC(I,2) = TMP1*DEXP(-TMP1*X(2))
         FJAC(I,3) = DEXP(-TEMP) - DEXP(-TMP1)
 1210    CONTINUE
      GO TO 1900
C
C     JENNRICH AND SAMPSON FUNCTION.
C
 1300 CONTINUE
```

```
      DO 1310 I = 1, M
         TEMP = DFLOAT(I)
         FJAC(I,1) = -TEMP*DEXP(TEMP*X(1))
         FJAC(I,2) = -TEMP*DEXP(TEMP*X(2))
 1310    CONTINUE
      GO TO 1900
C
C     BROWN AND DENNIS FUNCTION.
C
 1400 CONTINUE
      DO 1410 I = 1, M
         TEMP = DFLOAT(I)/FIVE
         TI = DSIN(TEMP)
         TMP1 = X(1) + TEMP*X(2) - DEXP(TEMP)
         TMP2 = X(3) + TI*X(4) - DCOS(TEMP)
         FJAC(I,1) = TWO*TMP1
         FJAC(I,2) = TEMP*FJAC(I,1)
         FJAC(I,3) = TWO*TMP2
         FJAC(I,4) = TI*FJAC(I,3)
 1410    CONTINUE
      GO TO 1900
C
C     CHEBYQUAD FUNCTION.
C
 1500 CONTINUE
      DX = ONE/DFLOAT(N)
      DO 1520 J = 1, N
         TMP1 = ONE
         TMP2 = TWO*X(J) - ONE
         TEMP = TWO*TMP2
         TMP3 = ZERO
         TMP4 = TWO
         DO 1510 I = 1, M
            FJAC(I,J) = DX*TMP4
            TI = FOUR*TMP2 + TEMP*TMP4 - TMP3
            TMP3 = TMP4
            TMP4 = TI
            TI = TEMP*TMP2 - TMP1
            TMP1 = TMP2
            TMP2 = TI
 1510       CONTINUE
 1520    CONTINUE
      GO TO 1900
C
C     BROWN ALMOST-LINEAR FUNCTION.
C
 1600 CONTINUE
      PROD = ONE
      DO 1620 J = 1, N
         PROD = X(J)*PROD
         DO 1610 I = 1, N
            FJAC(I,J) = ONE
 1610       CONTINUE
         FJAC(J,J) = TWO
 1620    CONTINUE
      DO 1650 J = 1, N
         TEMP = X(J)
         IF (TEMP .NE. ZERO) GO TO 1640
         TEMP = ONE
```

```
            PROD = ONE
            DO 1630 K = 1, N
               IF (K .NE. J) PROD = X(K)*PROD
 1630          CONTINUE
 1640       CONTINUE
            FJAC(N,J) = PROD/TEMP
 1650       CONTINUE
      GO TO 1900
C
C     OSBORNE 1 FUNCTION.
C
 1700 CONTINUE
      DO 1710 I = 1, 33
         TEMP = TEN*DFLOAT(I-1)
         TMP1 = DEXP(-X(4)*TEMP)
         TMP2 = DEXP(-X(5)*TEMP)
         FJAC(I,1) = -ONE
         FJAC(I,2) = -TMP1
         FJAC(I,3) = -TMP2
         FJAC(I,4) = TEMP*X(2)*TMP1
         FJAC(I,5) = TEMP*X(3)*TMP2
 1710    CONTINUE
      GO TO 1900
C
C     OSBORNE 2 FUNCTION.
C
 1800 CONTINUE
      DO 1810 I = 1, 65
         TEMP = DFLOAT(I-1)/TEN
         TMP1 = DEXP(-X(5)*TEMP)
         TMP2 = DEXP(-X(6)*(TEMP - X(9))**2)
         TMP3 = DEXP(-X(7)*(TEMP - X(10))**2)
         TMP4 = DEXP(-X(8)*(TEMP - X(11))**2)
         FJAC(I,1) = -TMP1
         FJAC(I,2) = -TMP2
         FJAC(I,3) = -TMP3
         FJAC(I,4) = -TMP4
         FJAC(I,5) = TEMP*X(1)*TMP1
         FJAC(I,6) = X(2)*(TEMP - X(9))**2*TMP2
         FJAC(I,7) = X(3)*(TEMP - X(10))**2*TMP3
         FJAC(I,8) = X(4)*(TEMP - X(11))**2*TMP4
         FJAC(I,9) = -TWO*X(2)*X(6)*(TEMP - X(9))*TMP2
         FJAC(I,10) = -TWO*X(3)*X(7)*(TEMP - X(10))*TMP3
         FJAC(I,11) = -TWO*X(4)*X(8)*(TEMP - X(11))*TMP4
 1810    CONTINUE
 1900 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE SSQJAC.
C
      END
      SUBROUTINE INITPT(N,X,NPROB,FACTOR)
      INTEGER N,NPROB
      DOUBLE PRECISION FACTOR
      DOUBLE PRECISION X(N)
C     **********
C
C     SUBROUTINE INITPT
C
C     THIS SUBROUTINE SPECIFIES THE STANDARD STARTING POINTS FOR THE
C     FUNCTIONS DEFINED BY SUBROUTINE OBJFCN. THE SUBROUTINE RETURNS
C     IN X A MULTIPLE (FACTOR) OF THE STANDARD STARTING POINT. FOR
C     THE SEVENTH FUNCTION THE STANDARD STARTING POINT IS ZERO, SO IN
C     THIS CASE, IF FACTOR IS NOT UNITY, THEN THE SUBROUTINE RETURNS
C     THE VECTOR  X(J) = FACTOR, J=1,...,N.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE INITPT(N,X,NPROB,FACTOR)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER VARIABLE.
C
C       X IS A LINEAR ARRAY OF LENGTH N. ON OUTPUT X CONTAINS THE
C         STANDARD STARTING POINT FOR PROBLEM NPROB MULTIPLIED BY
C         FACTOR.
C
C       NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
C
C       FACTOR SPECIFIES THE MULTIPLE OF THE STANDARD STARTING
C         POINT. IF FACTOR IS UNITY, NO MULTIPLICATION IS PERFORMED.
C
C     MINPACK. VERSION OF JANUARY 1978.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      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
C
C     SELECTION OF INITIAL POINT.
C
      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,
     1       1200,1300,1400,1500,1600,1700,1800),NPROB
C
C     HELICAL VALLEY FUNCTION.
C
  100 CONTINUE
      X(1) = -ONE
      X(2) = ZERO
      X(3) = ZERO
      GO TO 1900
C
```

```
C     BIGGS EXP6 FUNCTION.
C
  200 CONTINUE
      X(1) = ONE
      X(2) = TWO
      X(3) = ONE
      X(4) = ONE
      X(5) = ONE
      X(6) = ONE
      GO TO 1900
C
C     GAUSSIAN FUNCTION.
C
  300 CONTINUE
      X(1) = C1
      X(2) = ONE
      X(3) = ZERO
      GO TO 1900
C
C     POWELL BADLY SCALED FUNCTION.
C
  400 CONTINUE
      X(1) = ZERO
      X(2) = ONE
      GO TO 1900
C
C     BOX 3-DIMENSIONAL FUNCTION.
C
  500 CONTINUE
      X(1) = ZERO
      X(2) = TEN
      X(3) = TWENTY
      GO TO 1900
C
C     VARIABLY DIMENSIONED FUNCTION.
C
  600 CONTINUE
      H = ONE/DFLOAT(N)
      DO 610 J = 1, N
         X(J) = ONE - DFLOAT(J)*H
  610    CONTINUE
      GO TO 1900
C
C     WATSON FUNCTION.
C
  700 CONTINUE
      DO 710 J = 1, N
         X(J) = ZERO
  710    CONTINUE
      GO TO 1900
C
C     PENALTY FUNCTION I.
C
  800 CONTINUE
      DO 810 J = 1, N
         X(J) = DFLOAT(J)
  810    CONTINUE
      GO TO 1900
C
```

```
C     PENALTY FUNCTION II.
C
  900 CONTINUE
      DO 910 J = 1, N
         X(J) = HALF
  910    CONTINUE
      GO TO 1900
C
C     BROWN BADLY SCALED FUNCTION.
C
 1000 CONTINUE
      X(1) = ONE
      X(2) = ONE
      GO TO 1900
C
C     BROWN AND DENNIS FUNCTION.
C
 1100 CONTINUE
      X(1) = TWNTF
      X(2) = FIVE
      X(3) = -FIVE
      X(4) = -ONE
      GO TO 1900
C
C     GULF RESEARCH AND DEVELOPMENT FUNCTION.
C
 1200 CONTINUE
      X(1) = FIVE
      X(2) = C2
      X(3) = C3
      GO TO 1900
C
C     TRIGONOMETRIC FUNCTION.
C
 1300 CONTINUE
      H = ONE/DFLOAT(N)
      DO 1310 J = 1, N
         X(J) = H
 1310    CONTINUE
      GO TO 1900
C
C     EXTENDED ROSENBROCK FUNCTION.
C
 1400 CONTINUE
      DO 1410 J = 1, N, 2
         X(J) = -C4
         X(J+1) = ONE
 1410    CONTINUE
      GO TO 1900
C
C     EXTENDED POWELL SINGULAR FUNCTION.
C
 1500 CONTINUE
      DO 1510 J = 1, N, 4
         X(J) = THREE
         X(J+1) = -ONE
         X(J+2) = ZERO
         X(J+3) = ONE
 1510    CONTINUE
```

```
      GO TO 1900
C
C     BEALE FUNCTION.
C
 1600 CONTINUE
      X(1) = ONE
      X(2) = ONE
      GO TO 1900
C
C     WOOD FUNCTION.
C
 1700 CONTINUE
      X(1) = -THREE
      X(2) = -ONE
      X(3) = -THREE
      X(4) = -ONE
      GO TO 1900
C
C     CHEBYQUAD FUNCTION.
C
 1800 CONTINUE
      H = ONE/DFLOAT(N+1)
      DO 1810 J = 1, N
         X(J) = DFLOAT(J)*H
 1810    CONTINUE
C
C     COMPUTE MULTIPLE OF INITIAL POINT.
C
 1900 CONTINUE
      IF (FACTOR .EQ. ONE) GO TO 1940
      IF (NPROB .EQ. 7) GO TO 1920
      DO 1910 J = 1, N
         X(J) = FACTOR*X(J)
 1910    CONTINUE
      GO TO 1940
 1920 CONTINUE
      DO 1930 J = 1, N
         X(J) = FACTOR
 1930    CONTINUE
 1940 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE INITPT.
C
      END
      SUBROUTINE OBJFCN(N,X,F,NPROB)
      INTEGER N,NPROB
      DOUBLE PRECISION F
      DOUBLE PRECISION X(N)
C     **********
C
C     SUBROUTINE OBJFCN
C
C     THIS SUBROUTINE DEFINES THE OBJECTIVE FUNCTIONS OF EIGHTEEN
C     NONLINEAR UNCONSTRAINED MINIMIZATION PROBLEMS. THE VALUES
C     OF N FOR FUNCTIONS 1,2,3,4,5,10,11,12,16 AND 17 ARE
C     3,6,3,2,3,2,4,3,2 AND 4, RESPECTIVELY.
C     FOR FUNCTION 7, N MAY BE 2 OR GREATER BUT IS USUALLY 6 OR 9.
C     FOR FUNCTIONS 6,8,9,13,14,15 AND 18 N MAY BE VARIABLE,
C     HOWEVER IT MUST BE EVEN FOR FUNCTION 14, A MULTIPLE OF 4 FOR
C     FUNCTION 15, AND NOT GREATER THAN 50 FOR FUNCTION 18.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE OBJFCN(N,X,F,NPROB)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER VARIABLE.
C
C       X IS A LINEAR ARRAY OF LENGTH N.
C
C       F IS A REAL VARIABLE WHICH ON OUTPUT CONTAINS THE VALUE OF
C         THE NPROB OBJECTIVE FUNCTION EVALUATED AT X.
C
C       NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
C
C     SUBPROGRAMS REQUIRED
C
C       FORTRAN-SUPPLIED ... DABS,DATAN,DCOS,DEXP,DLOG,DSIGN,DSIN,
C                            DSQRT
C
C     MINPACK. VERSION OF JANUARY 1978.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,IEV,IVAR,J
      DOUBLE PRECISION AP,ARG,BP,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,
     3      3.989D-1,3.521D-1,2.42D-1,1.295D-1,5.4D-2,1.75D-2,
     4      4.4D-3,9.D-4/
      DFLOAT(IVAR) = IVAR
C
C     FUNCTION ROUTINE SELECTOR.
C
      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,
     1        1200,1300,1400,1500,1600,1700,1800),NPROB
C
C     HELICAL VALLEY FUNCTION.
C
  100 CONTINUE
      TPI = EIGHT*DATAN(ONE)
      TH = DSIGN(CP25,X(2))
      IF (X(1) .GT. ZERO) TH = DATAN(X(2)/X(1))/TPI
      IF (X(1) .LT. ZERO) TH = DATAN(X(2)/X(1))/TPI + CP5
      ARG = X(1)**2 + X(2)**2
      R = DSQRT(ARG)
      T = X(3) - TEN*TH
      F = C100*(T**2 + (R - ONE)**2) + X(3)**2
      GO TO 1900
C
C     BIGGS EXP6 FUNCTION.
C
  200 CONTINUE
      F = ZERO
      DO 210 I = 1, 13
         D1 = DFLOAT(I)/TEN
         D2 = DEXP(-D1) - FIVE*DEXP(-TEN*D1) + THREE*DEXP(-FOUR*D1)
         S1 = DEXP(-D1*X(1))
         S2 = DEXP(-D1*X(2))
         S3 = DEXP(-D1*X(5))
         T = X(3)*S1 - X(4)*S2 + X(6)*S3 - D2
         F = F + T**2
  210    CONTINUE
      GO TO 1900
C
C     GAUSSIAN FUNCTION.
C
  300 CONTINUE
      F = ZERO
      DO 310 I = 1, 15
         D1 = CP5*DFLOAT(I-1)
         D2 = C3P5 - D1 - X(3)
         ARG = -CP5*X(2)*D2**2
         R = DEXP(ARG)
         T = X(1)*R - Y(I)
         F = F + T**2
  310    CONTINUE
      GO TO 1900
C
C     POWELL BADLY SCALED FUNCTION.
C
  400 CONTINUE
      T1 = C10000*X(1)*X(2) - ONE
      S1 = DEXP(-X(1))
      S2 = DEXP(-X(2))
      T2 = S1 + S2 - ONE - CP0001
      F = T1**2 + T2**2
      GO TO 1900
C
C     BOX 3-DIMENSIONAL FUNCTION.
C
  500 CONTINUE
      F = ZERO
      DO 510 I = 1, 10
         D1 = DFLOAT(I)
         D2 = D1/TEN
         S1 = DEXP(-D2*X(1))
         S2 = DEXP(-D2*X(2))
         S3 = DEXP(-D2) - DEXP(-D1)
         T = S1 - S2 - S3*X(3)
         F = F + T**2
  510    CONTINUE
      GO TO 1900
C
C     VARIABLY DIMENSIONED FUNCTION.
C
  600 CONTINUE
      T1 = ZERO
      T2 = ZERO
      DO 610 J = 1, N
         T1 = T1 + DFLOAT(J)*(X(J) - ONE)
         T2 = T2 + (X(J) - ONE)**2
  610    CONTINUE
      F = T2 + T1**2*(ONE + T1**2)
      GO TO 1900
C
C     WATSON FUNCTION.
C
  700 CONTINUE
      F = ZERO
      DO 730 I = 1, 29
         D1 = DFLOAT(I)/C29
         S1 = ZERO
         D2 = ONE
         DO 710 J = 2, N
            S1 = S1 + DFLOAT(J-1)*D2*X(J)
            D2 = D1*D2
  710       CONTINUE
         S2 = ZERO
         D2 = ONE
         DO 720 J = 1, N
            S2 = S2 + D2*X(J)
            D2 = D1*D2
  720       CONTINUE
         T = S1 - S2**2 - ONE
         F = F + T**2
  730    CONTINUE
      T1 = X(2) - X(1)**2 - ONE
      F = F + X(1)**2 + T1**2
      GO TO 1900
C
C     PENALTY FUNCTION I.
C
  800 CONTINUE
      T1 = -CP25
      T2 = ZERO
      DO 810 J = 1, N
         T1 = T1 + X(J)**2
         T2 = T2 + (X(J) - ONE)**2
  810    CONTINUE
      F = AP*T2 + BP*T1**2
      GO TO 1900
C
C     PENALTY FUNCTION II.
C
  900 CONTINUE
      T1 = -ONE
      T2 = ZERO
      T3 = ZERO
      D1 = DEXP(CP1)
      D2 = ONE
      DO 920 J = 1, N
         T1 = T1 + DFLOAT(N-J+1)*X(J)**2
         S1 = DEXP(X(J)/TEN)
         IF (J .EQ. 1) GO TO 910
         S3 = S1 + S2 - D2*(D1 + ONE)
         T2 = T2 + S3**2
         T3 = T3 + (S1 - ONE/D1)**2
  910    CONTINUE
         S2 = S1
         D2 = D1*D2
  920    CONTINUE
      F = AP*(T2 + T3) + BP*(T1**2 + (X(1) - CP2)**2)
      GO TO 1900
C
C     BROWN BADLY SCALED FUNCTION.
C
 1000 CONTINUE
      T1 = X(1) - C1PD6
      T2 = X(2) - C2PD36
      T3 = X(1)*X(2) - TWO
      F = T1**2 + T2**2 + T3**2
      GO TO 1900
C
C     BROWN AND DENNIS FUNCTION.
C
 1100 CONTINUE
      F = 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
         F = F + T**2
 1110    CONTINUE
      GO TO 1900
C
C     GULF RESEARCH AND DEVELOPMENT FUNCTION.
C
 1200 CONTINUE
      F = ZERO
      D1 = TWO/THREE
      DO 1210 I = 1, 99
         ARG = DFLOAT(I)/C100
         R = DABS((-FIFTY*DLOG(ARG))**D1 + C25 - X(2))
         T1 = R**X(3)/X(1)
         T2 = DEXP(-T1)
         T = T2 - ARG
         F = F + T**2
 1210    CONTINUE
      GO TO 1900
C
C     TRIGONOMETRIC FUNCTION.
C
 1300 CONTINUE
      S1 = ZERO
      DO 1310 J = 1, N
         S1 = S1 + DCOS(X(J))
 1310    CONTINUE
      F = ZERO
      DO 1320 J = 1, N
         T = DFLOAT(N+J) - DSIN(X(J)) - S1 - DFLOAT(J)*DCOS(X(J))
         F = F + T**2
 1320    CONTINUE
      GO TO 1900
C
C     EXTENDED ROSENBROCK FUNCTION.
C
 1400 CONTINUE
      F = ZERO
      DO 1410 J = 1, N, 2
         T1 = ONE - X(J)
         T2 = TEN*(X(J+1) - X(J)**2)
         F = F + T1**2 + T2**2
 1410    CONTINUE
      GO TO 1900
C
C     EXTENDED POWELL FUNCTION.
C
 1500 CONTINUE
      F = ZERO
      DO 1510 J = 1, N, 4
         T = X(J) + TEN*X(J+1)
         T1 = X(J+2) - X(J+3)
         S1 = FIVE*T1
         T2 = X(J+1) - TWO*X(J+2)
         S2 = T2**3
         T3 = X(J) - X(J+3)
         S3 = TEN*T3**3
         F = F + T**2 + S1*T1 + S2*T2 + S3*T3
 1510    CONTINUE
      GO TO 1900
C
C     BEALE FUNCTION.
C
 1600 CONTINUE
      S1 = ONE - X(2)
      T1 = C1P5 - X(1)*S1
      S2 = ONE - X(2)**2
      T2 = C2P25 - X(1)*S2
      S3 = ONE - X(2)**3
      T3 = C2P625 - X(1)*S3
      F = T1**2 + T2**2 + T3**2
      GO TO 1900
C
C     WOOD FUNCTION.
C
 1700 CONTINUE
      S1 = X(2) - X(1)**2
      S2 = ONE - X(1)
      S3 = X(2) - ONE
      T1 = X(4) - X(3)**2
      T2 = ONE - X(3)
      T3 = X(4) - ONE
      F = C100*S1**2 + S2**2 + C90*T1**2 + T2**2 +
     1    TEN*(S3 + T3)**2 + (S3 - T3)**2/TEN
      GO TO 1900
C
C     CHEBYQUAD FUNCTION.
C
 1800 CONTINUE
      DO 1810 I = 1, N
         FVEC(I) = ZERO
 1810    CONTINUE
      DO 1830 J = 1, N
         T1 = ONE
         T2 = TWO*X(J) - ONE
         T = TWO*T2
         DO 1820 I = 1, N
            FVEC(I) = FVEC(I) + T2
            TH = T*T2 - T1
            T1 = T2
            T2 = TH
 1820       CONTINUE
 1830    CONTINUE
      F = ZERO
      D1 = ONE/DFLOAT(N)
      IEV = -1
      DO 1840 I = 1, N
         T = D1*FVEC(I)
         IF (IEV .GT. 0) T = T + ONE/(DFLOAT(I)**2 - ONE)
         F = F + T**2
         IEV = -IEV
 1840    CONTINUE
 1900 CONTINUE
      RETURN
C
C     LAST CARD OF SUBROUTINE OBJFCN.
C
      END
      SUBROUTINE GRDFCN(N,X,G,NPROB)
      INTEGER N,NPROB
      DOUBLE PRECISION X(N),G(N)
C     **********
C
C     SUBROUTINE GRDFCN
C
C     THIS SUBROUTINE DEFINES THE GRADIENT VECTORS OF EIGHTEEN
C     NONLINEAR UNCONSTRAINED MINIMIZATION PROBLEMS. THE PROBLEM
C     DIMENSIONS ARE AS DESCRIBED IN THE PROLOGUE COMMENTS OF OBJFCN.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE GRDFCN(N,X,G,NPROB)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER VARIABLE.
C
C       X IS A LINEAR ARRAY OF LENGTH N.
C
C       G IS A LINEAR ARRAY OF LENGTH N WHICH ON OUTPUT CONTAINS
C         THE COMPONENTS OF THE GRADIENT VECTOR OF THE NPROB
C         OBJECTIVE FUNCTION EVALUATED AT X.
C
C       NPROB IS A POSITIVE INTEGER VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
C
C     SUBPROGRAMS REQUIRED
C
C       FORTRAN-SUPPLIED ... DABS,DATAN,DCOS,DEXP,DLOG,DSIGN,DSIN,
C                            DSQRT
C
C     MINPACK. VERSION OF JANUARY 1978.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,IEV,IVAR,J
      DOUBLE PRECISION AP,ARG,BP,C2PDM6,CP0001,CP1,CP2,CP25,CP5,C1P5,
     1                 C2P25,C2P625,C3P5,C19P8,C20P2,C25,C29,C100,C180,C200,
     2                 C10000,C1PD6,D1,D2,EIGHT,FIFTY,FIVE,FOUR,ONE,R,S1,S2,S3,
     3                 T,T1,T2,T3,TEN,TH,THREE,TPI,TWENTY,TWO,ZERO
      DOUBLE PRECISION FVEC(50),Y(15)
      DOUBLE PRECISION DFLOAT
      DATA ZERO,ONE,TWO,THREE,FOUR,FIVE,EIGHT,TEN,TWENTY,FIFTY
     1     /0.D0,1.D0,2.D0,3.D0,4.D0,5.D0,8.D0,1.D1,2.D1,5.D1/
      DATA C2PDM6,CP0001,CP1,CP2,CP25,CP5,C1P5,C2P25,C2P625,
     1     C3P5,C19P8,C20P2,C25,C29,C100,C180,C200,C10000,C1PD6
     2     /2.D-6,1.D-4,1.D-1,2.D-1,2.5D-1,5.D-1,1.5D0,2.25D0,2.625D0,
     3      3.5D0,1.98D1,2.02D1,2.5D1,2.9D1,1.D2,1.8D2,2.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,
     3      3.989D-1,3.521D-1,2.42D-1,1.295D-1,5.4D-2,1.75D-2,
     4      4.4D-3,9.D-4/
      DFLOAT(IVAR) = IVAR
C
C     GRADIENT ROUTINE SELECTOR.
```

```
C
      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,
     1       1200,1300,1400,1500,1600,1700,1800),NPROB
C
C     HELICAL VALLEY FUNCTION.
C
  100 CONTINUE
      TPI = EIGHT*DATAN(ONE)
      TH = DSIGN(CP25,X(2))
      IF (X(1) .GT. ZERO) TH = DATAN(X(2)/X(1))/TPI
      IF (X(1) .LT. ZERO) TH = DATAN(X(2)/X(1))/TPI + CP5
      ARG = X(1)**2 + X(2)**2
      R = DSQRT(ARG)
      T = X(3) - TEN*TH
      S1 = TEN*T/(TPI*ARG)
      G(1) = C200*(X(1) - X(1)/R + X(2)*S1)
      G(2) = C200*(X(2) - X(2)/R - X(1)*S1)
      G(3) = TWO*(C100*T + X(3))
      GO TO 1900
C
C     BIGGS EXP6 FUNCTION.
C
  200 CONTINUE
      DO 210 J = 1, N
         G(J) = ZERO
  210    CONTINUE
      DO 220 I = 1, 13
         D1 = DFLOAT(I)/TEN
         D2 = DEXP(-D1) - FIVE*DEXP(-TEN*D1) + THREE*DEXP(-FOUR*D1)
         S1 = DEXP(-D1*X(1))
         S2 = DEXP(-D1*X(2))
         S3 = DEXP(-D1*X(5))
         T = X(3)*S1 - X(4)*S2 + X(6)*S3 - D2
         TH = D1*T
         G(1) = G(1) - S1*TH
         G(2) = G(2) + S2*TH
         G(3) = G(3) + S1*T
         G(4) = G(4) - S2*T
         G(5) = G(5) - S3*TH
         G(6) = G(6) + S3*T
  220    CONTINUE
      G(1) = TWO*X(3)*G(1)
      G(2) = TWO*X(4)*G(2)
      G(3) = TWO*G(3)
      G(4) = TWO*G(4)
      G(5) = TWO*X(6)*G(5)
      G(6) = TWO*G(6)
      GO TO 1900
C
C     GAUSSIAN FUNCTION.
C
  300 CONTINUE
      G(1) = ZERO
      G(2) = ZERO
      G(3) = ZERO
      DO 310 I = 1, 15
         D1 = CP5*DFLOAT(I-1)
         D2 = C3P5 - D1 - X(3)
         ARG = -CP5*X(2)*D2**2
```

```
            R = DEXP(ARG)
            T = X(1)*R - Y(I)
            S1 = R*T
            S2 = D2*S1
            G(1) = G(1) + S1
            G(2) = G(2) - D2*S2
            G(3) = G(3) + S2
  310       CONTINUE
        G(1) = TWO*G(1)
        G(2) = X(1)*G(2)
        G(3) = TWO*X(1)*X(2)*G(3)
        GO TO 1900
C
C       POWELL BADLY SCALED FUNCTION.
C
  400 CONTINUE
        T1 = C10000*X(1)*X(2) - ONE
        S1 = DEXP(-X(1))
        S2 = DEXP(-X(2))
        T2 = S1 + S2 - ONE - CP0001
        G(1) = TWO*(C10000*X(2)*T1 - S1*T2)
        G(2) = TWO*(C10000*X(1)*T1 - S2*T2)
        GO TO 1900
C
C       BOX 3-DIMENSIONAL FUNCTION.
C
  500 CONTINUE
        G(1) = ZERO
        G(2) = ZERO
        G(3) = ZERO
        DO 510 I = 1, 10
            D1 = DFLOAT(I)
            D2 = D1/TEN
            S1 = DEXP(-D2*X(1))
            S2 = DEXP(-D2*X(2))
            S3 = DEXP(-D2) - DEXP(-D1)
            T = S1 - S2 - S3*X(3)
            TH = D2*T
            G(1) = G(1) - S1*TH
            G(2) = G(2) + S2*TH
            G(3) = G(3) - S3*T
  510       CONTINUE
        G(1) = TWO*G(1)
        G(2) = TWO*G(2)
        G(3) = TWO*G(3)
        GO TO 1900
C
C       VARIABLY DIMENSIONED FUNCTION.
C
  600 CONTINUE
        T1 = ZERO
        DO 610 J = 1, N
            T1 = T1 + DFLOAT(J)*(X(J) - ONE)
  610       CONTINUE
        T = T1*(ONE + TWO*T1**2)
        DO 620 J = 1, N
            G(J) = TWO*(X(J) - ONE + DFLOAT(J)*T)
  620       CONTINUE
        GO TO 1900
```

```
C
C     WATSON FUNCTION.
C
  700 CONTINUE
      DO 710 J = 1, N
         G(J) = ZERO
  710    CONTINUE
      DO 750 I = 1, 29
         D1 = DFLOAT(I)/C29
         S1 = ZERO
         D2 = ONE
         DO 720 J = 2, N
            S1 = S1 + DFLOAT(J-1)*D2*X(J)
            D2 = D1*D2
  720       CONTINUE
         S2 = ZERO
         D2 = ONE
         DO 730 J = 1, N
            S2 = S2 + D2*X(J)
            D2 = D1*D2
  730       CONTINUE
         T = S1 - S2**2 - ONE
         S3 = TWO*D1*S2
         D2 = TWO/D1
         DO 740 J = 1, N
            G(J) = G(J) + D2*(DFLOAT(J-1) - S3)*T
            D2 = D1*D2
  740       CONTINUE
  750    CONTINUE
      T1 = X(2) - X(1)**2 - ONE
      G(1) = G(1) + X(1)*(TWO - FOUR*T1)
      G(2) = G(2) + TWO*T1
      GO TO 1900
C
C     PENALTY FUNCTION I.
C
  800 CONTINUE
      T1 = -CP25
      DO 810 J = 1, N
         T1 = T1 + X(J)**2
  810    CONTINUE
      D1 =TWO*AP
      TH = FOUR*BP*T1
      DO 820 J = 1, N
         G(J) = D1*(X(J) - ONE) + X(J)*TH
  820    CONTINUE
      GO TO 1900
C
C     PENALTY FUNCTION II.
C
  900 CONTINUE
      T1 = -ONE
      DO 910 J = 1, N
         T1 = T1 + DFLOAT(N-J+1)*X(J)**2
  910    CONTINUE
      D1 = DEXP(CP1)
      D2 = ONE
      TH = FOUR*BP*T1
      DO 930 J = 1, N
```

```
            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))
            T1 = R**X(3)/X(1)
            T2 = DEXP(-T1)
            T = T2 - ARG
            S1 = T1*T2*T
            G(1) = G(1) + S1
            G(2) = G(2) + S1/R
            G(3) = G(3) - S1*DLOG(R)
 1210       CONTINUE
         G(1) = TWO*G(1)/X(1)
         G(2) = TWO*X(3)*G(2)
         G(3) = TWO*G(3)
         GO TO 1900
C
C        TRIGONOMETRIC FUNCTION.
C
 1300 CONTINUE
         S1 = ZERO
         DO 1310 J = 1, N
            G(J) = DCOS(X(J))
            S1 = S1 + G(J)
 1310       CONTINUE
         S2 = ZERO
         DO 1320 J = 1, N
            TH = DSIN(X(J))
            T = DFLOAT(N+J) - TH - S1 - DFLOAT(J)*G(J)
            S2 = S2 + T
            G(J) = (DFLOAT(J)*TH - G(J))*T
 1320       CONTINUE
         DO 1330 J = 1, N
            G(J) = TWO*(G(J) + DSIN(X(J))*S2)
 1330       CONTINUE
         GO TO 1900
C
C        EXTENDED ROSENBROCK FUNCTION.
C
 1400 CONTINUE
         DO 1410 J = 1, N, 2
            T1 = ONE - X(J)
            G(J+1) = C200*(X(J+1) - X(J)**2)
            G(J) = -TWO*(X(J)*G(J+1) + T1)
 1410       CONTINUE
         GO TO 1900
C
C        EXTENDED POWELL FUNCTION.
C
 1500 CONTINUE
         DO 1510 J = 1, N, 4
            T = X(J) + TEN*X(J+1)
            T1 = X(J+2) - X(J+3)
            S1 = FIVE*T1
            T2 = X(J+1) - TWO*X(J+2)
            S2 = FOUR*T2**3
            T3 = X(J) - X(J+3)
            S3 = TWENTY*T3**3
            G(J) = TWO*(T + S3)
            G(J+1) = TWENTY*T + S2
            G(J+2) = TWO*(S1 - S2)
            G(J+3) = -TWO*(S1 + S3)
 1510       CONTINUE
```

```
      GO TO 1900
C
C     BEALE FUNCTION.
C
 1600 CONTINUE
      S1 = ONE - X(2)
      T1 = C1P5 - X(1)*S1
      S2 = ONE - X(2)**2
      T2 = C2P25 - X(1)*S2
      S3 = ONE - X(2)**3
      T3 = C2P625 - X(1)*S3
      G(1) = -TWO*(S1*T1 + S2*T2 + S3*T3)
      G(2) = TWO*X(1)*(T1 + X(2)*(TWO*T2 + THREE*X(2)*T3))
      GO TO 1900
C
C     WOOD FUNCTION.
C
 1700 CONTINUE
      S1 = X(2) - X(1)**2
      S2 = ONE - X(1)
      S3 = X(2) - ONE
      T1 = X(4) - X(3)**2
      T2 = ONE - X(3)
      T3 = X(4) - ONE
      G(1) = -TWO*(C200*X(1)*S1 + S2)
      G(2) = C200*S1 + C20P2*S3 + C19P8*T3
      G(3) = -TWO*(C180*X(3)*T1 + T2)
      G(4) = C180*T1 + C20P2*T3 + C19P8*S3
      GO TO 1900
C
C     CHEBYQUAD FUNCTION.
C
 1800 CONTINUE
      DO 1810 I = 1, N
         FVEC(I) = ZERO
 1810    CONTINUE
      DO 1830 J = 1, N
         T1 = ONE
         T2 = TWO*X(J) - ONE
         T = TWO*T2
         DO 1820 I = 1, N
            FVEC(I) = FVEC(I) + T2
            TH = T*T2 - T1
            T1 = T2
            T2 = TH
 1820       CONTINUE
 1830    CONTINUE
      D1 = ONE/DFLOAT(N)
      IEV = -1
      DO 1840 I = 1, N
         FVEC(I)= D1*FVEC(I)
         IF (IEV .GT. 0) FVEC(I) = FVEC(I) + ONE/(DFLOAT(I)**2 - ONE)
         IEV = -IEV
 1840    CONTINUE
      DO 1860 J = 1, N
         G(J) = ZERO
         T1 = ONE
         T2 = TWO*X(J) - ONE
         T = TWO*T2
```

```
            S1 = ZERO
            S2 = TWO
            DO 1850 I = 1, N
               G(J) = G(J) + FVEC(I)*S2
               TH = FOUR*T2 + T*S2 - S1
               S1 = S2
               S2 = TH
               TH = T*T2 - T1
               T1 = T2
               T2 = TH
 1850          CONTINUE
 1860       CONTINUE
         D2 = TWO*D1
         DO 1870 J = 1, N
            G(J) = D2*G(J)
 1870       CONTINUE
 1900    CONTINUE
         RETURN
C
C        LAST CARD OF SUBROUTINE GRDFCN.
C
         END
```

A P P E N D I X 2

Sample Driver and Interface Function

```
C     **********
C
C     THIS PROGRAM TESTS CODES FOR THE LEAST-SQUARES SOLUTION OF
C     M NONLINEAR EQUATIONS IN N VARIABLES. IT CONSISTS OF A DRIVER
C     AND AN INTERFACE SUBROUTINE FCN. THE DRIVER READS IN DATA,
C     CALLS THE NONLINEAR LEAST-SQUARES SOLVER, AND FINALLY PRINTS
C     OUT INFORMATION ON THE PERFORMANCE OF THE SOLVER. THIS IS
C     ONLY A SAMPLE DRIVER, MANY OTHER DRIVERS ARE POSSIBLE. THE
C     INTERFACE SUBROUTINE FCN IS NECESSARY TO TAKE INTO ACCOUNT THE
C     FORMS OF CALLING SEQUENCES USED BY THE FUNCTION AND JACOBIAN
C     SUBROUTINES IN THE VARIOUS NONLINEAR LEAST-SQUARES SOLVERS.
C
C     SUBPROGRAMS REQUIRED
C
C       USER-SUPPLIED ...... FCN
C
C       MINPACK-SUPPLIED ... ENORM,INITPT,SOLVER,SSQFCN
C
C     MINPACK. VERSION OF OCTOBER 1977.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER I,IC,INFO,K,LDFJAC,LWA,M,N,NFEV,NJEV,
     1        NPROB,NREAD,NTRIES,NWRITE
      INTEGER IWA(40),MA(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
C
C     LOGICAL INPUT UNIT IS ASSUMED TO BE NUMBER 5.
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.
C
      DATA NREAD,NWRITE /5,6/
C
      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
         NJ(IC) = NJEV
         NX(IC) = INFO
         FNM(IC) = FNORM2
         WRITE (NWRITE,3000) FNORM1,FNORM2,NFEV,NJEV,INFO,(X(I),I=1,N)
         FACTOR = TEN*FACTOR
   20    CONTINUE
      GO TO 10
   30 CONTINUE
      WRITE (NWRITE,4000) IC
      WRITE (NWRITE,5000)
      DO 40 I = 1, IC
         WRITE (NWRITE,6000) NP(I),NA(I),MA(I),NF(I),NJ(I),NX(I),FNM(I)
   40    CONTINUE
      STOP
 1000 FORMAT (4I5)
 2000 FORMAT ( //// 5X,8H PROBLEM,I5,5X,11H DIMENSIONS,2I5,5X // )
 3000 FORMAT (5X,33H INITIAL L2 NORM OF THE RESIDUALS,D15.7 //
     1        5X,33H FINAL L2 NORM OF THE RESIDUALS  ,D15.7 //
     2        5X,33H NUMBER OF FUNCTION EVALUATIONS  ,I10 //
     3        5X,33H NUMBER OF JACOBIAN EVALUATIONS  ,I10 //
     4        5X,15H EXIT PARAMETER ,18X,I10 //
     5        5X,27H FINAL APPROXIMATE SOLUTION  // (5X,5D15.7))
 4000 FORMAT (12H1SUMMARY OF ,I3,16H CALLS TO SOLVER/)
 5000 FORMAT (49H NPROB   N    M   NFEV  NJEV  INFO  FINAL L2 NORM/)
 6000 FORMAT (3I5,3I6,2X,D15.7)
C
C     LAST CARD OF DRIVER.
C
      END
      SUBROUTINE FCN(M,N,X,FVEC,FJAC,LDFJAC,IFLAG)
      INTEGER M,N,LDFJAC,IFLAG
      DOUBLE PRECISION X(N),FVEC(M),FJAC(LDFJAC,N)
C     **********
C
C     THE CALLING SEQUFNCE 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     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER NPROB,NFEV,NJEV
      COMMON /REFNUM/ NPROB,NFEV,NJEV
      IF (IFLAG .EQ. 1) CALL SSQFCN(M,N,X,FVEC,NPROB)
      IF (IFLAG .EQ. 2) CALL SSQJAC(M,N,X,FJAC,LDFJAC,NPROB)
      IF (IFLAG .EQ. 1) NFEV = NFEV + 1
      IF (IFLAG .EQ. 2) NJEV = NJEV + 1
      RETURN
C
C     LAST CARD OF INTERFACE SUBROUTINE FCN.
C
      END
```

A P P E N D I X   3

Sample Data

Sample Data for Nonlinear Equations

```
NPROB   N  NTRIES

  1     2    3                                                              00000010
  2     4    3                                                              00000020
  3     2    2                                                              00000030
  4     4    3                                                              00000040
  5     3    3                                                              00000050
  6     6    2                                                              00000060
  6     9    2                                                              00000070
  6    12    2                                                              00000080
  7     5    3                                                              00000090
  7     6    3                                                              00000100
  7     7    3                                                              00000110
  7     8    1                                                              00000120
  7     9    1                                                              00000130
  8    10    3                                                              00000140
  8    30    1                                                              00000150
  9    10    3                                                              00000160
 10     1    3                                                              00000170
 10    10    3                                                              00000180
 11    10    3                                                              00000190
 12    10    3                                                              00000200
 13    10    3                                                              00000210
 14    10    3                                                              00000220
  0     0    0                                                              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
```

```
NPROB   N  NTRIES

   1    3    3                                                               00000010
   2    6    1                                                               00000020
   3    3    1                                                               00000030
   4    2    1                                                               00000040
   5    3    1                                                               00000050
   6   10    3                                                               00000060
   7    9    3                                                               00000070
   7   12    3                                                               00000080
   8   10    3                                                               00000090
   9    1    3                                                               00000100
   9    4    3                                                               00000110
   9   10    3                                                               00000120
  10    2    3                                                               00000130
  11    4    3                                                               00000140
  12    3    2                                                               00000150
  13   10    3                                                               00000160
  14    2    3                                                               00000170
  15    4    3                                                               00000180
  16    2    3                                                               00000190
  17    4    3                                                               00000200
  18    7    1                                                               00000210
  18    8    1                                                               00000220
  18    9    1                                                               00000230
  18   10    1                                                               00000240
   0    0    0                                                               00000250
```