Testing Unconstrained Optimization Software

JORGE J. MORÉ, BURTON S. GARBOW, and KENNETH E. HILLSTROM
Argonne National Laboratory

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.

Key Words and Phrases: performance testing, systems of nonlinear equations, nonlinear least squares, unconstrained minimization, optimization software
CR Categories. 4.6, 5.15, 5.41
The Algorithm: FORTRAN Subroutines for Testing Unconstrained Optimization Software. *ACM Trans. Math. Software 7,* 1 (March 1981), 136–140.

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 that succeed from

This work was performed under the auspices of the U.S. Department of Energy.
Authors' address: Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439.

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 nonstandard 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 that other testing procedures have ignored.

2. THE BASIC SUBROUTINES

Testing of optimization software requires a basic set of subroutines that 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 $i = 1, \ldots, n$, solve

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

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

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

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

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

The subroutines that 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

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

and

VECJAC(N, X, FJAC, LDFJAC, NPROB)

returns in FJAC the Jacobian matrix

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

(The parameter LDFJAC is the leading dimension of the array FJAC as defined in the main program.) In order to prevent gross inefficiencies with solvers that only require one component at a time,

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

returns in FCNK the kth 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

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

For unconstrained minimization

OBJFCN(N, X, F, NPROB)

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

GRDFCN(N, X, G, 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

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

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.

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 minimizers. In a few cases a 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 $(1.098\cdots 10^{-5}, 9.106\ldots)$

(4) *Brown badly scaled function* [unpublished]
(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) *Beale function* [2]
(a) $n = 2, \quad m = 3$
(b) $f_i(x) = y_i - x_1(1 - x_2^i)$,
where $y_1 = 1.5, \quad y_2 = 2.25, \quad y_3 = 2.625$
(c) $x_0 = (1, 1)$
(d) $f = 0$ at $(3, 0.5)$

(6) *Jennrich and Sampson function* [16]
(a) $n = 2, \quad m \geq n$
(b) $f_i(x) = 2 + 2i - (\exp[ix_1] + \exp[ix_2])$
(c) $x_0 = (0.3, 0.4)$
(d) $f = 124.362\ldots$ at $x_1 = x_2 = 0.2578\ldots$ for $m = 10$

(7) *Helical valley function* [12]
(a) $n = 3, \quad m = 3$
(b) $f_1(x) = 10[x_3 - 10\theta(x_1, x_2)]$
$f_2(x) = 10[x_1^2 + x_2^2)^{1/2} - 1]$
$f_3(x) = x_3$

where

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

(c) $x_0 = (-1, 0, 0)$
(d) $f = 0$ at $(1, 0, 0)$

(8) *Bard function* [1]
(a) $n = 3, \quad m = 15$
(b) $f_i(x) = y_i - \left(x_1 + \dfrac{u_i}{v_i x_2 + w_i x_3}\right)$

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

i	y	i	y	i	y
1	0.14	5	0.29	11	0.73
2	0.18	6	0.32	12	0.96
3	0.22	7	0 35	13	1.34
4	0.25	8	0.39	14	2.10
		9	0.37	15	4.39
		10	0.58		

(c) $x_0 = (1, 1, 1)$
(d) $f = 8.21487\cdots 10^{-3}$
$f = 17.4286\cdots$ at $(0.8406\ldots, -\infty, -\infty)$

(9) *Gaussian function* [unpublished]
(a) $n = 3, \quad m = 15$
(b) $f_i(x) = x_1 \exp\left[\dfrac{-x_2(t_i - x_3)^2}{2}\right] - y_i$

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) $f = 1.12793\cdots 10^{-8}$

(10) *Meyer function* [18]

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

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

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) $f = 87.9458\ldots$

(11) *Gulf research and development function* [10]

(a) $n = 3, \quad n \le m \le 100$

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

where $t_i = i/100$

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

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

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

(12) *Box three-dimensional function* [4]

(a) $n = 3, \quad m \ge n$ variable

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

where $t_i = (0.1)i$

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

(d) $f = 0$ at $(1, 10, 1), (10, 1, -1)$

and wherever $(x_1 = x_2$ and $x_3 = 0)$

(13) *Powell singular function* [23]

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

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

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

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

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

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

(d) $f = 0$ at the origin

(14) *Wood function* [9]

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

(b) $f_1(x) = 10(x_2 - x_1^2)$
$f_2(x) = 1 - x_1$
$f_3(x) = (90)^{1/2}(x_4 - x_3^2)$
$f_4(x) = 1 - x_3$
$f_5(x) = (10)^{1/2}(x_2 + x_4 - 2)$
$f_6(x) = (10)^{-1/2}(x_2 - x_4)$
(c) $x_0 = (-3, -1, -3, -1)$
(d) $f = 0$ at $(1, 1, 1, 1)$

(15) *Kowalik and Osborne function* [17]
(a) $n = 4, \quad m = 11$

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

where

i	y_i	u_i	i	y_i	u_i
1	0.1957	4.0000	7	0.0456	0.1250
2	0.1947	2.0000	8	0.0342	0.1000
3	0.1735	1.0000	9	0.0323	0.0833
4	0 1600	0.5000	10	0 0235	0.0714
5	0.0844	0.2500	11	0.0246	0.0625
6	0 0627	0.1670			

(c) $x_0 = (0.25, 0.39, 0.415, 0.39)$
(d) $f = 3.07505 \cdots 10^{-4}$
$f = 1.02734 \cdots 10^{-3}$ at $(+\infty, -14.07 \ldots, -\infty, -\infty)$

(16) *Brown and Dennis function* [6]
(a) $n = 4, \quad m \geq n$ 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) $n = 5, \quad m = 33$
(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	i	y_i	i	y_i
1	0.844	10	0 784	19	0.538	28	0.431
2	0.908	11	0.751	20	0.522	29	0.424
3	0.932	12	0.718	21	0.506	30	0.420
4	0.936	13	0.685	22	0.490	31	0.414
5	0.925	14	0.658	23	0.478	32	0.411
6	0.908	15	0.628	24	0 467	33	0.406
7	0.881	16	0.603	25	0.457		
8	0 850	17	0.580	26	0.448		
9	0.818	18	0.558	27	0.438		

(c) $x_0 = (0.5, 1.5, -1, 0.01, 0.02)$
(d) $f = 5.46489 \cdots 10^{-5}$

(18) *Biggs EXP6 function* [3]
(a) $n = 6, \quad m \geq n$ variable
(b) $f_i(x) = x_3 \exp[-t_i x_1] - x_4 \exp[-t_i x_2] + x_6 \exp[-t_i x_5] - y_i$
where $t_i = (0.1)i$
and $y_i = \exp[-t_i] - 5 \exp[-10t_i] + 3 \exp[-4t_i]$
(c) $x_0 = (1, 2, 1, 1, 1, 1)$
(d) $f = 5.65565 \cdots 10^{-3}$ if $m = 13$
$f = 0$ at $(1, 10, 1, 5, 4, 3)$

(19) *Osborne 2 function* [21]
(a) $n = 11, \quad m = 65$
(b) $f_i(x) = y_i - (x_1 \exp[-t_i x_5] + x_2 \exp[-(t_i - x_9)^2 x_6] + x_3 \exp[-(t_i - x_{10})^2 x_7] + x_4 \exp[-(t_i - x_{11})^2 x_8])$
where $t_i = (i - 1)/10$ and

i	y_i	i	y_i	i	y_i
1	1.366	23	0.694	45	0.672
2	1.191	24	0.644	46	0.708
3	1.112	25	0.624	47	0.633
4	1.013	26	0.661	48	0.668
5	0.991	27	0.612	49	0.645
6	0.885	28	0.558	50	0.632
7	0.831	29	0.533	51	0.591
8	0.847	30	0.495	52	0.559
9	0.786	31	0 500	53	0.597
10	0.725	32	0.423	54	0.625
11	0.746	33	0 395	55	0.739
12	0.679	34	0.375	56	0.710
13	0.608	35	0.372	57	0.729
14	0.655	36	0.391	58	0.720
15	0.616	37	0 396	59	0.636
16	0.606	38	0.405	60	0.581
17	0.602	39	0 428	61	0.428
18	0.626	40	0.429	62	0.292
19	0.651	41	0.523	63	0.162
20	0.724	42	0.562	64	0.098
21	0.649	43	0.607	65	0.054
22	0.649	44	0.653		

(c) $x_0 = (1.3, 0.65, 0.65, 0.7, 0.6, 3, 5, 7, 2, 4.5, 5.5)$
(d) $f = 4.01377 \cdots 10^{-2}$

(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 $t_i = i/29, \quad 1 \le i \le 29$
$f_{30}(x) = x_1, \qquad f_{31}(x) = x_2 - x_1^2 - 1$
(c) $x_0 = (0, \ldots, 0)$
$f = 2.28767 \cdots 10^{-3}$ if $n = 6$
(d) $f = 1.39976 \cdots 10^{-6}$ if $n = 9$
$f = 4.72238 \cdots 10^{-10}$ if $n = 12$

(21) *Extended Rosenbrock function* [25]
(a) n variable but even, $m = n$
(b) $f_{2i-1}(x) = 10(x_{2i} - x^2_{i-1})$
$f_{2i}(x) = 1 - x_{2i-1}$
(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) $f_{4i-3}(x) = x_{4i-3} + 10x_{4i-2}$
$f_{4i-2}(x) = 5^{1/2}(x_{4i-1} - x_{4i})$
$f_{4i-1}(x) = (x_{4i-2} - 2x_{4i-1})^2$
$f_{4i}(x) = 10^{1/2}(x_{4i-3} - x_{4i})^2$
(c) $x_0 = (\xi_j)$
where $\xi_{4j-3} = 3, \quad \xi_{4j-2} = -1, \quad \xi_{4j-1} = 0, \quad \xi_{4j} = 1$
(d) $f = 0$ at the origin

(23) *Penalty function I* [14]
(a) n variable, $m = n + 1$
(b) $f_i(x) = a^{1/2}(x_i - 1), \quad 1 \le i \le n$

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

where $a = 10^{-5}$
(c) $x_0 = (\xi_j)$ where $\xi_j = j$
(d) $f = 2.24997 \cdots 10^{-5}$ if $n = 4$
$f = 7.08765 \cdots 10^{-5}$ if $n = 10$

(24) *Penalty function II* [14]
(a) n variable, $m = 2n$
(b) $f_1(x) = x_1 - 0.2$

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

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

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

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) $f = 9.37629 \cdots 10^{-6}$ if $n = 4$
$f = 2.93660 \cdots 10^{-4}$ if $n = 10$

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

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

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

(c) $x_0 = (\xi_j)$ where $\xi_j = 1 - (j/n)$
(d) $f = 0$ at $(1, \ldots, 1)$

(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) $x_0 = (1/n, \ldots, 1/n)$
(d) $f = 0$

(27) *Brown almost-linear function* [5]
(a) n variable, $m = n$

(b) $f_i(x) = x_i + \sum_{j=1}^{n} x_j - (n + 1), \quad 1 \le i < n$

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

(c) $x_0 = (\frac{1}{2}, \ldots, \frac{1}{2})$
(d) $f = 0$ at $(\alpha, \ldots, \alpha, \alpha^{1-n})$
where α satisfies $n\alpha^n - (n + 1)\alpha^{n-1} + 1 = 0$; in particular, $\alpha = 1$
$f = 1$ at $(0, \ldots, 0, n + 1)$

(28) *Discrete boundary value function* [20]
(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 equation function* [20]
(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 $J_i = \{j : j \neq i, \max(1, i - m_l) \leq j \leq \min(n, i + m_u)\}$
and $m_l = 5$, $m_u = 1$
(c) $x_0 = (-1, \ldots, -1)$
(d) $f = 0$

(32) *Linear function—full rank* [unpublished]
(a) n variable, $m \geq n$
(b) $f_i(x) = x_i - \frac{2}{m}\left(\sum_{j=1}^{n} x_j\right) - 1, \qquad 1 \leq i \leq n$

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

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

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

(c) $x_0 = (1, \ldots, 1)$
(d) $f = \frac{m^2 + 3m - 6}{2(2m-3)}$ at any point where $\sum_{j=2}^{m-1} jx_j = \frac{3}{2m-3}$

(35) *Chebyquad function* [11]

(a) n variable, $m \geq n$

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

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

$\int_0^1 T_i(x)\, dx = 0$ for i odd,

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

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

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

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

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

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 three-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 three-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 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.

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

Table I

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

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 subfunctions, 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 I was presented in [19]. In this table "FC" means failure to converge within 1000 function evaluations.

Table II. 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

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, 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 that indicates the status of the computation, and in Tables II and III we have used the parameter INFO to report this information. If the subroutine claims success, then INFO is set to 1; otherwise it is set to 0.

Table III. 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

We first ran these algorithms with the standard starting points; the results are shown in Tables II and III. 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 (fewer than 50).

The conclusion from Tables II and III 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 IV and V, 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 toward 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 $n = 1$, and 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 IV and V 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.

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

$$\hat{F}(x) = \alpha F(\Sigma x), \qquad \hat{x}_0 = \Sigma^{-1} x_0 \tag{5.1}$$

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 that satisfy

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

Table IV. 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

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 $\hat{F}$.

Table V. 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

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

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

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 VI and VII. 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 that is not a solution to the system of nonlinear equations.

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

We next ran these algorithms on the scaled problem (5.2) where Σ_1 is the identity matrix and Σ_2 is chosen by (5.3); the results are shown in Tables VIII

Table VI. 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
8	40	182	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 VII. 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

Table VIII. Summary of 22 Calls to NEQ1

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

Table IX. 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

and IX. 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.

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. Tech. Rep. No. 25, Statistical Techniques Research Group, Princeton Univ., Princeton, N.J., 1958.
3. BIGGS, M.C. Minimization algorithms making use of non-quadratic properties of the objective function. *J Inst. Math Appl 8* (1971), 315–327.
4. BOX, M.J. A comparison of several current optimization methods, and the use of transformations in constrained problems. *Comput. J 9* (1966), 67–77.
5. BROWN, K.M. A quadratically convergent Newton-like method based upon Gaussian elimination. *SIAM J. Numer. Anal. 6* (1969), 560–569.
6. BROWN, K.M., AND DENNIS, J.E. New computational algorithms for minimizing a sum of squares of nonlinear functions. Rep. No. 71–6, Yale Univ, Dep. Comput. Science, New Haven, Conn., March 1971.
7. BROYDEN, C.G. A class of methods for solving nonlinear simultaneous equations. *Math Comput 19* (1965), 577–593.
8. BROYDEN, C.G. The convergence of an algorithm for solving sparse nonlinear systems. *Math. Comput. 25* (1971), 285–294.
9. COLVILLE, A.R. A comparative study of nonlinear programming codes. Rep. 320–2949, IBM New York Scientific Center, 1968.
10. COX, R A. Comparison of the performance of seven optimization algorithms on twelve unconstrained optimization problems. Ref. 1335CNO4, Gulf Research and Development Company, Pittsburg, Jan. 1969.
11. FLETCHER, R. Function minimization without evaluating derivatives—A review. *Comput. J. 8* (1965), 33–41.
12. FLETCHER, R., AND POWELL, M.J.D. A rapidly convergent descent method for minimization *Comput. J. 6* (1963), 163–168.
13. FREUDENSTEIN, F., AND ROTH, B. Numerical solutions of systems of nonlinear equations. *J ACM 10*, 4 (Oct. 1963), 550–556.
14. GILL, P E, MURRAY W, AND PITFIELD, R.A. The implementation of two revised quasi-Newton algorithms for unconstrained optimization. Rep. NAC 11, National Phys. Lab., April 1972, pp. 82–83.
15. HILLSTROM, K.E. A simulation test approach to the evaluation of nonlinear optimization algorithms. *ACM Trans. Math. Softw. 3*, 4 (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.* Elsevier North-Holland, New York, 1968
18. MEYER, R.R. Theoretical and computational aspects of nonlinear regression. In *Nonlinear Programming*, J. B. Rosen, O. L. Mangasarian, and K. Ritter (Eds), Academic Press, New York, 1970, pp. 465–486.
19. MORÉ, J J. The Levenberg-Marquardt algorithm. Implementation and theory In *Numerical Analysis*, G. A. Watson (Ed.), *Lecture Notes in Mathematics 630*, Springer-Verlag, New York, 1977, pp. 105–116.
20. MORÉ, J.J., AND COSNARD, M.Y. Numerical solution of nonlinear equations. *ACM Trans. Math. Softw 5*, 1 (March 1979), 64–85.

21. OSBORNE, M.R. Some aspects of nonlinear least squares calculations. In *Numerical Methods for Nonlinear Optimization*, F. A. Lootsma (Ed), Academic Press, New York, 1972, pp 171–189.
22. POWELL, M.J.D. A hybrid method for nonlinear equations. In *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz (Ed), Gordon & Breach, New York, 1970, pp. 87–114
23. POWELL, M.J.D. An iterative method for finding stationary values of a function of several variables. *Comput. J. 5* (1962), 147–151.
24. ROSENBROCK, H.H. An automatic method for finding the greatest or least value of a function. *Comput. J. 3* (1960), 175–184.
25. SPEDICATO, E. Computational experience with quasi-Newton algorithms for minimization problems of moderately large size Rep. CISE-N-175, Segrate (Milano), 1975.

Received September 1978; revised May 1979; accepted July 1979.