HESFCN — A FORTRAN Package of Hessian Subroutines for Testing Nonlinear Optimization Software

Victoria Z. Averbukh (averbukh@acf9.nyu.edu)
Samuel Figueroa (figueroa@cs.nyu.edu)
Tamar Schlick (schlick@acfclu.nyu.edu)*

Courant Institute of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012

*Generous support for this work was provided by the National Science Foundation through the REU (Research Experience for Undergraduates) Program. Additional support was provided by the NYU Presidential Scholar Program (V.A.), the New York State Science & Technology Foundation, the AAUW Educational Foundation, the San Diego Supercomputer Center, and the Academic Computing Facility at Courant Institute, NYU.

Abstract

We report the development of Hessian FORTRAN routines for testing unconstrained nonlinear optimization. An existing package, "Algorithm 566" (J. Moré, B. G. Garbow, and K. Hillstrom, *ACM Trans. Math. Softw.* **7**, 14–41 and 136–140, 1981) provides function and gradient subroutines of 18 test functions for multivariate minimization. Our supplementary Hessian segments will enable users to test optimization software that requires second derivative information. Eigenvalue analysis throughout the minimization will also be possible in the goal of better understanding minimization progress by different algorithms and the relation of progress to eigenvalue distribution and condition number.

1 Introduction

A robust and easy-to-use package of FORTRAN subroutines for testing unconstrained optimization software was prepared about a decade ago [14,15]. The package includes testing software for three problem areas: systems of nonlinear functions, nonlinear least squares, and unconstrained minimization. For unconstrained minimization, 18 test problems are provided with the following associated subroutines:

- `INITPT(N,X,NPROB,FACTOR)`,
- `OBJFCN(N,X,F,NPROB)`,
- `GRDFCN(N,X,G,NPROB)`.

The user specifies an integer `NPROB` to select the test problem ($1 \leq$ `NPROB` ≤ 18); a dimension `N`, which must be in a feasible range for the function `NPROB`; and a value for `FACTOR`, specifying how the starting vector `X(N)` should be chosen as a multiple of a "standard" initial point associated with each function. The subroutine `INITPT` generates the starting point corresponding to `NPROB` and `FACTOR`, and subroutines `OBJFCN` and `GRDFCN` return, respectively, the values of the function `F`, and the gradient vector, `G(N)`, at `X(N)`. Each of the 18 objective functions for unconstrained optimization is defined through subfunctions $f_1, f_2, \ldots, f_m$:

$$F(X) = \sum_{i=1}^{m} f_i^2(X)$$

where $X = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$.

The names of all objective functions, corresponding values of m and n, and original sources for all functions are summarized in Table I. The test functions were mainly collected from papers reporting new optimization approaches and illustrating the methods on specially-designed functions [1–6, 8, 9, 11, 12, 19, 24]. Thus, some of these functions involve characteristics that complicate minimization progress, such as steep valleys near the minimum

(see Figures 1–4 for examples) and magnitude differences among the independent variables. Although "real life" optimization problems are unlikely to possess these characteristics to such a degree, these model functions are useful for observing trends in performance among different minimization algorithms as a function of problem structure.

In this paper, we report the development of FORTRAN Hessian routines for these 18 test problems, package HESFCN. We use a compatible program format with Algorithm 566 [14,15] in which

- `HESFCN(N,X,HESD,HESL,NPROB)`

returns the diagonal, `HESD`, and lower-diagonal, `HESL` (stored by rows), of the Hessian arrays corresponding to `NPROB` and `X(N)`. These Hessian segments enable optimization-software testers to implement minimization methods that require second-derivative information. Hessian information will also allow users to perform eigenvalue analysis throughout the minimization process in the goal of better understanding progress and its relation to eigenvalue distribution and condition number.

Indeed, new developments in hardware and software are now opening new possibilities for large-scale optimization applications. Greater computing powers and storage are expanding our range of applicable methods for solving important physical problems (e.g., in meteorology, fluid dynamics, computational chemistry) and our ability to exploit function structure (e.g., natural separability among the components) and analytic derivative information. For example, new variants of Newton methods for large-scale problems include truncated Newton [7, 16, 17, 20–22, 25] and limited-memory quasi-Newton methods [10,13,17,18,25]. Truncated Newton methods maintain the rapid convergence of Newton methods while reducing required second-derivative information. They can be particularly powerful when Hessian information is exploited for preconditioning [16,21]. Limited-memory quasi-Newton methods maintain the superlinear convergence of quasi-Newton methods while relaxing the typical storage requirements for the full Hessian (or inverse Hessian) through a flexible and cyclic strategy for

storing and updating vectors. Since limited-memory quasi-Newton and truncated Newton approaches are likely to have quite different performance characteristics, HESFCN can be used to relate problem structure with performance.

In Section 2 we list each function definition, associated standard starting point, gradient and Hessian expressions, and minima. The appendix contains the FORTRAN program segments.

2 Test Functions

The test functions are defined by the following format:

(a) Dimension

(b) Function definition

(c) Standard starting point

(d) Gradient expression

(e) Hessian expression

(f) Minima

We mention special function characteristics when relevant.

2.1 Helical valley function

This function contains a steep-sided helical valley in the direction of x_3.

(a) $m = n = 3$

(b) $f_1(X) = 10(x_3 - 10\theta(x_1, x_2))$

$f_2(X) = 10\sqrt{x_1^2 + x_2^2} - 10$

$f_3(X) = x_3$

where

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

(c) $X_0 = (-1, 0, 0)^T$

(d) $$\frac{\partial F(X)}{\partial x_1} = 200 \left(x_1 - \frac{x_1}{\sqrt{x_1^2 + x_2^2}} + \left(\frac{x_2}{2\pi}\right) \frac{f_1(X)}{x_1^2 + x_2^2} \right)$$

$$\frac{\partial F(X)}{\partial x_2} = 200 \left(x_2 - \frac{x_2}{\sqrt{x_1^2 + x_2^2}} - \left(\frac{x_1}{2\pi}\right) \frac{f_1(X)}{x_1^2 + x_2^2} \right)$$

$$\frac{\partial F(X)}{\partial x_3} = 20 f_1(X) + 2x_3$$

(e) $$\frac{\partial^2 F(X)}{\partial x_1^2} = 200 \left(1 - \frac{x_2^2}{(x_1^2 + x_2^2)^{3/2}} - \frac{1}{2\pi} \left(\frac{2x_1 x_2 f_1(X)}{(x_1^2 + x_2^2)^2} - \frac{100}{2\pi} \frac{x_2^2}{(x_1^2 + x_2^2)^2} \right) \right)$$

$$\frac{\partial^2 F(X)}{\partial x_2^2} = 200 \left(1 - \frac{x_1^2}{(x_1^2 + x_2^2)^{3/2}} + \frac{1}{2\pi} \left(\frac{2x_1 x_2 f_1(X)}{(x_1^2 + x_2^2)^2} - \frac{100}{2\pi} \frac{x_1^2}{(x_1^2 + x_2^2)^2} \right) \right)$$

$$\frac{\partial^2 F(X)}{\partial x_3^2} = 202$$

$$\frac{\partial^2 F(X)}{\partial x_2 \partial x_1} = 200 \left(\frac{x_1 x_2}{(x_1^2 + x_2^2)^{3/2}} + \frac{1}{2\pi} \left(\frac{(x_1^2 - x_2^2) f_1(X)}{(x_1^2 + x_2^2)^2} - \frac{100}{2\pi} \frac{x_1 x_2}{(x_1^2 + x_2^2)^2} \right) \right)$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_1} = 200 \left(\frac{10}{2\pi}\right) \frac{x_2}{x_1^2 + x_2^2}$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_2} = -200 \left(\frac{10}{2\pi}\right) \frac{x_1}{x_1^2 + x_2^2}$$

(f) $F(X) = 0$ at $X = (1, 0, 0)^T$

2.2 Biggs EXP6 function

This function, described by Biggs [2], involves m exponential subfunctions that all contain curving steep-sided valleys. The subfunctions depend on two to six variables, and problem difficulty increases with dimension [2]. Here we use a version of the function with six variables.

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

(b) $f_i(X) = x_3 \exp(-t_i x_1) - x_4 \exp(-t_i x_2) + x_6 \exp(-t_i x_5) - \exp(-t_i) + 5\exp(-i)$
$\qquad - 3\exp(-4t_i), \qquad i = 1, \ldots, m$

where $t_i = i/10$

(c) $X_0 = (1, 2, 1, 1, 1, 1)^T$

(d) $$\frac{\partial F(X)}{\partial x_1} = -2 \sum_{i=1}^{m} t_i x_3 \exp(-t_i x_1) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_2} = 2 \sum_{i=1}^{m} t_i x_4 \exp(-t_i x_2) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_3} = 2 \sum_{i=1}^{m} \exp(-t_i x_1) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_4} = -2 \sum_{i=1}^{m} \exp(-t_i x_2) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_5} = -2 \sum_{i=1}^{m} t_i x_6 \exp(-t_i x_5) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_6} = 2 \sum_{i=1}^{m} \exp(-t_i x_5) f_i(X)$$

(e) $$\frac{\partial^2 F(X)}{\partial x_1^2} = 2 \sum_{i=1}^{m} t_i^2 x_3 \exp(-t_i x_1) f_i(X) + t_i^2 x_3^2 \exp(-2t_i x_1)$$

$$\frac{\partial^2 F(X)}{\partial x_2^2} = -2 \sum_{i=1}^{m} t_i^2 x_4 \exp(-t_i x_2) f_i(X) - t_i^2 x_4^2 \exp(-2t_i x_2)$$

$$\frac{\partial^2 F(X)}{\partial x_3^2} = 2\sum_{i=1}^{m} \exp(-2t_i x_1)$$

$$\frac{\partial^2 F(X)}{\partial x_4^2} = 2\sum_{i=1}^{m} \exp(-2t_i x_2)$$

$$\frac{\partial^2 F(X)}{\partial x_5^2} = 2\sum_{i=1}^{m} t_i^2 x_6 \exp(-t_i x_5) f_i(X) + t_i^2 x_6^2 \exp(-2t_i x_5)$$

$$\frac{\partial^2 F(X)}{\partial x_6^2} = 2\sum_{i=1}^{m} \exp(-2t_i x_5)$$

$$\frac{\partial^2 F(X)}{\partial x_2 \partial x_1} = -2\sum_{i=1}^{m} t_i^2 x_3 \exp(-t_i x_1) x_4 \exp(-t_i x_2)$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_1} = -2\sum_{i=1}^{m} t_i \exp(-t_i x_1) f_i(X) - t_i x_3 \exp(-2t_i x_1)$$

$$\frac{\partial^2 F(X)}{\partial x_4 \partial x_1} = 2\sum_{i=1}^{m} t_i x_3 \exp(-t_i x_1) \exp(-t_i x_2)$$

$$\frac{\partial^2 F(X)}{\partial x_5 \partial x_1} = 2\sum_{i=1}^{m} t_i^2 x_3 \exp(-t_i x_1) x_6 \exp(-t_i x_5)$$

$$\frac{\partial^2 F(X)}{\partial x_6 \partial x_1} = -2\sum_{i=1}^{m} t_i x_3 \exp(-t_i x_1) \exp(-t_i x_5)$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_2} = 2\sum_{i=1}^{m} t_i x_4 \exp(-t_i x_2) \exp(-t_i x_1)$$

$$\frac{\partial^2 F(X)}{\partial x_4 \partial x_2} = 2\sum_{i=1}^{m} t_i \exp(-t_i x_2)\big(f_i(X) - x_4 \exp(-t_i x_2)\big)$$

$$\frac{\partial^2 F(X)}{\partial x_5 \partial x_2} = -2\sum_{i=1}^{m} t_i x_4 \exp(-t_i x_2) x_6 \exp(-t_i x_5)$$

$$\frac{\partial^2 F(X)}{\partial x_6 \partial x_2} = 2\sum_{i=1}^{m} t_i x_4 \exp(-t_i x_2) \exp(-t_i x_5)$$

$$\frac{\partial^2 F(X)}{\partial x_4 \partial x_3} = -2 \sum_{i=1}^{m} \exp(-t_i x_2) \exp(-t_i x_1)$$

$$\frac{\partial^2 F(X)}{\partial x_5 \partial x_3} = -2 \sum_{i=1}^{m} t_i x_6 \exp(-t_i x_5) \exp(-t_i x_1)$$

$$\frac{\partial^2 F(X)}{\partial x_6 \partial x_3} = 2 \sum_{i=1}^{m} \exp(-t_i x_1) \exp(-t_i x_5)$$

$$\frac{\partial^2 F(X)}{\partial x_5 \partial x_4} = 2 \sum_{i=1}^{m} t_i x_6 \exp(-t_i x_5) \exp(-t_i x_2)$$

$$\frac{\partial^2 F(X)}{\partial x_6 \partial x_4} = -2 \sum_{i=1}^{m} \exp(-t_i x_2) \exp(-t_i x_5)$$

$$\frac{\partial^2 F(X)}{\partial x_6 \partial x_5} = -2 \sum_{i=1}^{m} t_i \exp(-t_i x_5)(f_i(X) + x_6 \exp(-t_i x_5))$$

(f) $F(X) \approx 5.7 \times 10^{-3}$ for $m = 13$

$F(X) = 0$ at $X = (1, 10, 1, 5, 4, 3)^T$ and $X = (4, 10, 3, 5, 1, 1)^T$ for $m = 6$

(These minima are taken from [14].)

2.3 Gaussian function

(a) $m = 15$, $n = 3$

(b) $f_i(X) = x_1 \exp\left(\frac{-x_2(t_i - x_3)^2}{2}\right) - y_i$, $i = 1, \ldots, 15$

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) $X_0 = (0.4, 1, 0)^T$

(d) $$\frac{\partial F(X)}{\partial x_1} = 2\sum_{i=1}^{15} \frac{\partial f_i(X)}{\partial x_1} f_i(X)$$

$$\frac{\partial F(X)}{\partial x_2} = -2\sum_{i=1}^{15} \frac{x_1(t_i - x_3)^2}{2} \frac{\partial f_i(X)}{\partial x_1} f_i(X)$$

$$\frac{\partial F(X)}{\partial x_3} = 2\sum_{i=1}^{15} x_1 x_2 (t_i - x_3) \frac{\partial f_i(X)}{\partial x_1} f_i(X)$$

(e) $$\frac{\partial^2 F(X)}{\partial x_1^2} = 2\sum_{i=1}^{15} \exp\left(-x_2(t_i - x_3)^2\right)$$

$$\frac{\partial^2 F(X)}{\partial x_2^2} = 2\sum_{i=1}^{15} \frac{x_1(t_i - x_3)^4}{4} \left(\frac{\partial f_i(X)}{\partial x_1} f_i(X) + x_1 \exp\left(-x_2(t_i - x_3)^2\right) \right)$$

$$\frac{\partial^2 F(X)}{\partial x_3^2} = -2\sum_{i=1}^{15} x_1 x_2 \frac{\partial f_i(X)}{\partial x_1} (f_i(X) - x_2(t_i - x_3)^2 (2f_i(X) + y_i))$$

$$\frac{\partial^2 F(X)}{\partial x_2 \partial x_1} = -2\sum_{i=1}^{15} \frac{(t_i - x_3)^2}{2} \frac{\partial f_i(X)}{\partial x_1} f_i(X) + \frac{x_1(t_i - x_3)^2}{2} \exp\left(-x_2(t_i - x_3)^2\right)$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_1} = 2\sum_{i=1}^{15} x_2(t_i - x_3) \frac{\partial f_i(X)}{\partial x_1} f_i(X) + x_1 x_2 (t_i - x_3) \exp\left(-x_2(t_i - x_3)^2\right)$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_2} = 2\sum_{i=1}^{15} x_1(t_i - x_3) \frac{\partial f_i(X)}{\partial x_1} (f_i(X) + x_2(t_i - x_3)^2 f_i(X))$$

(f) $F(X) \approx 1.12793 \times 10^{-8}$ at $X \approx (0.39896, 1.0002, -10^{-21})^T$

2.4 Powell badly scaled function

(a) $m = n = 2$

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

$f_2(X) = \exp(-x_1) + \exp(-x_2) - 1.0001$

(c) $X_0 = (0,1)^T$

(d) $\dfrac{\partial F(X)}{\partial x_1} = 2x_2(10^4)f_1(X) - 2\exp(-x_1)f_2(X)$

$\dfrac{\partial F(X)}{\partial x_2} = 2x_1(10^4)f_1(X) - 2\exp(-x_2)f_2(X)$

(e) $\dfrac{\partial^2 F(X)}{\partial x_1^2} = 2\big(10^8 x_2^2 + \exp(-x_1)f_2(X) + \exp(-2x_1)\big)$

$\dfrac{\partial^2 F(X)}{\partial x_2^2} = 2\big(10^8 x_1^2 + \exp(-x_2)f_2(X) + \exp(-2x_2)\big)$

$\dfrac{\partial^2 F(X)}{\partial x_2 \partial x_1} = 2\big(10^4 f_1(X) + 10^8 x_1 x_2 + \exp(-x_1)\exp(-x_2)\big)$

(f) $F(X) = 0$ at $X \approx (1.098 \times 10^{-5}, 9.106)^T$

2.5 Box three-dimensional function

This problem was derived from the two-dimensional problem

$$f(x_1, x_2) = \sum_{i=1}^{10} \Big(\big(\exp(-ix_1/10) - \exp(-ix_2/10)\big) - \big(\exp(-i/10) - \exp(-i)\big) \Big)^2,$$

possessing an assymetric curved valley.

(a) $n = 3$, $m \geq 3 = n$

(b) $f_i(X) = \exp(-t_i x_1) - \exp(-t_i x_2) - x_3\big(\exp(-t_i) - \exp(-i)\big), \qquad i = 1, \ldots, m$

where $t_i = i/10$

(c) $X_0 = (0, 10, 20)^T$

(d) $\dfrac{\partial F(X)}{\partial x_1} = -2\sum_{i=1}^{m} t_i \exp(-t_i x_1) f_i(X)$

$\dfrac{\partial F(X)}{\partial x_2} = 2\sum_{i=1}^{m} t_i \exp(-t_i x_2) f_i(X)$

$$\frac{\partial F(X)}{\partial x_3} = -2\sum_{i=1}^{m}\big(\exp(-t_i) - \exp(-i)\big) f_i(X)$$

(e) $$\frac{\partial^2 F(X)}{\partial x_1^2} = 2\sum_{i=1}^{m} t_i^2 \exp(-t_i x_1) f_i(X) + t_i^2 \exp(-2t_i x_1)$$

$$\frac{\partial^2 F(X)}{\partial x_2^2} = -2\sum_{i=1}^{m} t_i^2 \exp(-t_i x_2) f_i(X) - t_i^2 \exp(-2t_i x_2)$$

$$\frac{\partial^2 F(X)}{\partial x_3^2} = 2\sum_{i=1}^{m}\big(\exp(-t_i) - \exp(-i)\big)^2$$

$$\frac{\partial^2 F(X)}{\partial x_2 \partial x_1} = -2\sum_{i=1}^{m} t_i^2 \exp\big(-t_i(x_1 + x_2)\big)$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_1} = 2\sum_{i=1}^{m} t_i \exp(-t_i x_1)\big(\exp(-t_i) - \exp(-i)\big)$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_2} = -2\sum_{i=1}^{m} t_i \exp(-t_i x_2)\big(\exp(-t_i) - \exp(-i)\big)$$

(f) $F(X) = 0$ at $X = (1, 10, 1)^T$, $X = (10, 1, -1)^T$, and $X = (a, a, 0)^T$, where $a \in \mathbb{R}$

2.6 Variably dimensioned function

(a) $m = n + 2$, n variable

(b) $f_i(X) = x_i - 1, \qquad i = 1, \ldots, n$

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

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

(c) $X_0 = (1 - 1/n, 1 - 2/n, \ldots, 0)^T$

(d) $$\frac{\partial F(X)}{\partial x_j} = 2\big(f_j(X) + j f_{n+1}(X) + 2j f_{n+1}(X) f_{n+2}(X)\big), \qquad j = 1, \ldots, n$$

(e) $\dfrac{\partial^2 F(X)}{\partial x_j \partial x_k} = 2(\delta_{jk} + jk + 6jkf_{n+2}(X)), \qquad j,k = 1,\ldots,n$

(f) $F(X) = 0$ at $X = (1,\ldots,1)^T$

2.7 Watson function

(a) $2 \le n \le 31 = m$

(b) $f_i(X) = \sum_{j=2}^{n}(j-1)t_i^{j-2}x_j - \left(\sum_{j=1}^{n} t_i^{j-1}x_j\right)^2 - 1, \qquad i = 1,\ldots,29$

$f_{30}(X) = x_1$

$f_{31}(X) = x_2 - x_1^2 - 1$

where $t_i = i/29$

(c) $X_0 = (0,\ldots,0)^T$

(d) $\dfrac{\partial F(X)}{\partial x_j} = 2\sum_{i=1}^{29}\left((j-1)t_i^{j-2} - 2t_i^{j-1}\sum_{k=1}^{n} t_i^{k-1}x_k\right)f_i(X)$
$+ \delta_{1j}(2f_{30}(X) - 4x_1 f_{31}(X)) + 2\delta_{2j}f_{31}(X), \qquad j = 1,\ldots,n$

(e) $\dfrac{\partial^2 F(X)}{\partial x_j \partial x_k} = 2\sum_{i=1}^{29}\left[\left((k-1)t_i^{k-2} - 2t_i^{k-1}\sum_{l=1}^{n} t_i^{l-1}x_l\right)\left((j-1)t_i^{j-2} - 2t_i^{j-1}\sum_{l=1}^{n} t_i^{l-1}x_l\right)\right.$
$\left. - 2t_i^{j+k-2}f_i(X)\right] + \delta_{1j}\delta_{1k}(12x_1^2 - 4x_2 + 6) + 2\delta_{2j}\delta_{2k} - 4(\delta_{1j}\delta_{2k} + \delta_{2j}\delta_{1k})x_1,$
$j,k = 1,\ldots,n$

(f) $F(X) \approx 2.28767 \times 10^{-3}$ for $n = 6$
$F(X) \approx 1.39976 \times 10^{-6}$ for $n = 9$
$F(X) \approx 4.72238 \times 10^{-10}$ for $n = 12$

2.8 Penalty function I

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

(b) $f_i(X) = \sqrt{10^{-5}}(x_i - 1), \qquad i = 1, \ldots, n$

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

(c) $X_0 = (1, 2, \ldots, n)^T$

(d) $\dfrac{\partial F(X)}{\partial x_j} = 2\sqrt{10^{-5}} f_j(X) + 4x_j f_{n+1}(X), \qquad j = 1, \ldots, n$

(e) $\dfrac{\partial^2 F(X)}{\partial x_j \partial x_k} = 2\big(10^{-5}\delta_{jk} + 2(\delta_{jk} f_{n+1}(X) + 2x_j x_k)\big), \qquad j, k = 1, \ldots, n$

(f) $F(X) \approx 2.24997 \times 10^{-5}$ for $n = 4$

$F(X) \approx 7.08765 \times 10^{-5}$ for $n = 10$

2.9 Penalty function II

(a) $m = 2n$, n variable

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

$$f_i(X) = \sqrt{10^{-5}}\Big(\exp(x_i/10) + \exp(x_{i-1}/10) - \exp(i/10) - \exp\big((i-1)/10\big)\Big),$$
$$i = 2, \ldots, n$$

$$f_i(X) = \sqrt{10^{-5}}\big(\exp(x_{i-n+1}/10) - \exp(-1/10)\big), \qquad i = n+1, \ldots, 2n-1$$

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

(c) $X_0 = (\frac{1}{2}, \ldots, \frac{1}{2})^T$

(d) $$\frac{\partial F(X)}{\partial x_j} = \frac{\sqrt{10^{-5}}}{5} \exp\left(\frac{x_j}{10}\right)\Big((1 - \delta_{1j}) f_j(X) + (1 - \delta_{jn}) f_{j+1}(X) + (1 - \delta_{1j}) f_{j+n-1}(X)\Big)$$
$$+ 2\delta_{1j} f_1(X) + 4(n - j + 1)x_j f_{2n}(X), \qquad j = 1, \ldots, n$$

(e) $$\frac{\partial^2 F(X)}{\partial x_j \partial x_k} = \frac{\sqrt{10^{-5}}}{50}\exp\left(\frac{x_j}{10}\right)\left(\sqrt{10^{-5}}(\delta_{j,k+1}+\delta_{jk}+2\delta_{jk}(1-\delta_{1j})+\delta_{j+1,k})\exp\left(\frac{x_k}{10}\right)\right.$$
$$\left. + \delta_{jk}((1-\delta_{1j})f_j(X)+f_{j+1}(X)+(1-\delta_{1j})f_{j+n-1}(X))\right) + 2\delta_{1j}\delta_{1k}$$
$$+ 4(n-j+1)(2(n-k+1)x_j x_k + \delta_{jk} f_{2n}(X)), \qquad j,k = 1,\ldots,n-1$$

$$\frac{\partial^2 F(X)}{\partial x_j \partial x_n} = \frac{\sqrt{10^{-5}}}{50}\exp\left(\frac{x_j}{10}\right)\left(\sqrt{10^{-5}}(\delta_{j,n-1}+2\delta_{jn})\exp\left(\frac{x_n}{10}\right) + \delta_{jn}(f_n(X)+f_{2n-1}(X))\right)$$
$$+ 8(n-j+1)x_j x_n + 4\delta_{jn} f_{2n}(X), \qquad j = 1,\ldots,n$$

(f) $F(X) \approx 9.37629 \times 10^{-6}$ for $n = 4$

$F(X) \approx 2.93660 \times 10^{-4}$ for $n = 10$

2.10 Brown badly scaled function

(a) $m = 3$, $n = 2$

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

$f_2(X) = x_2 - 2 \times 10^{-6}$

$f_3(X) = x_1 x_2 - 2$

(c) $X_0 = (1,1)^T$

(d) $$\frac{\partial F(X)}{\partial x_1} = 2(f_1(X) + x_2 f_3(X))$$

$$\frac{\partial F(X)}{\partial x_2} = 2(f_2(X) + x_1 f_3(X))$$

(e) $$\frac{\partial^2 F(X)}{\partial x_1^2} = 2(1 + x_2^2)$$

$$\frac{\partial^2 F(X)}{\partial x_2^2} = 2(1 + x_1^2)$$

$$\frac{\partial^2 F(X)}{\partial x_2 \partial x_1} = 4(x_1 x_2 - 1)$$

(f) $F(X) = 0$ at $X = (10^6, 2 \times 10^{-6})^T$

2.11 Brown and Dennis function

(a) $m \geq 4 = n$

(b) $f_i(X) = \left(x_1 + t_i x_2 - \exp(t_i)\right)^2 + \left(x_3 + x_4 \sin(t_i) - \cos(t_i)\right)^2, \qquad i = 1, \ldots, m$

where $t_i = i/5$

(c) $X_0 = (25, 5, -5, -1)^T$

(d) $$\frac{\partial F(X)}{\partial x_1} = 4 \sum_{i=1}^{m} \left(x_1 + t_i x_2 - \exp(t_i)\right) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_2} = 4 \sum_{i=1}^{m} t_i \left(x_1 + t_i x_2 - \exp(t_i)\right) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_3} = 4 \sum_{i=1}^{m} \left(x_3 + x_4 \sin(t_i) - \cos(t_i)\right) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_4} = 4 \sum_{i=1}^{m} \left(x_3 + x_4 \sin(t_i) - \cos(t_i)\right) \sin(t_i) f_i(X)$$

(e) $$\frac{\partial^2 F(X)}{\partial x_1^2} = 4 \sum_{i=1}^{m} f_i(X) + 2\left(x_1 + t_i x_2 - \exp(t_i)\right)^2$$

$$\frac{\partial^2 F(X)}{\partial x_2^2} = 4 \sum_{i=1}^{m} t_i^2 f_i(X) + 2t_i^2 \left(x_1 + t_i x_2 - \exp(t_i)\right)^2$$

$$\frac{\partial^2 F(X)}{\partial x_3^2} = 4 \sum_{i=1}^{m} f_i(X) + 2\left(x_3 + x_4 \sin(t_i) - \cos(t_i)\right)^2$$

$$\frac{\partial^2 F(X)}{\partial x_4^2} = 4 \sum_{i=1}^{m} \sin^2(t_i) f_i(X) + 2 \sin^2(t_i) \left(x_3 + x_4 \sin(t_i) - \cos(t_i)\right)^2$$

$$\frac{\partial^2 F(X)}{\partial x_2 \partial x_1} = 4 \sum_{i=1}^{m} t_i f_i(X) + 2t_i \left(x_1 + t_i x_2 - \exp(t_i)\right)^2$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_1} = 8 \sum_{i=1}^{m} \left(x_1 + t_i x_2 - \exp(t_i)\right) \left(x_3 + x_4 \sin(t_i) - \cos(t_i)\right)$$

$$\frac{\partial^2 F(X)}{\partial x_4 \partial x_1} = 8\sum_{i=1}^{m}(x_1 + t_i x_2 - \exp(t_i))(x_3 + x_4 \sin(t_i) - \cos(t_i))\sin(t_i)$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_2} = 8\sum_{i=1}^{m} t_i(x_1 + t_i x_2 - \exp(t_i))(x_3 + x_4 \sin(t_i) - \cos(t_i))$$

$$\frac{\partial^2 F(X)}{\partial x_4 \partial x_2} = 8\sum_{i=1}^{m} t_i(x_1 + t_i x_2 - \exp(t_i))(x_3 + x_4 \sin(t_i) - \cos(t_i))\sin(t_i)$$

$$\frac{\partial^2 F(X)}{\partial x_4 \partial x_3} = 4\sum_{i=1}^{m} \sin(t_i) f_i(X) + 2\sin(t_i)(x_3 + x_4 \sin(t_i) - \cos(t_i))^2$$

(f) $F(X) \approx 85822.2$ for $m = 20$

2.12 Gulf research and development function

(Reference [23] and some others call this function the Weibull function.)

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

(b) $f_i(X) = \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right) - t_i, \qquad i = 1, \ldots, m$

where $t_i = i/100$ and $y_i = 25 + (-50 \ln t_i)^{2/3} - x_2$

(c) $X_0 = (5, 2.5, 0.15)^T$

(d) $$\frac{\partial F(X)}{\partial x_1} = 2\sum_{i=1}^{m} \frac{|y_i|^{x_3}}{x_1^2} \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_2} = 2\sum_{i=1}^{m} \frac{x_3 |y_i|^{x_3}}{x_1 y_i} \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right) f_i(X)$$

$$\frac{\partial F(X)}{\partial x_3} = 2\sum_{i=1}^{m} -\frac{|y_i|^{x_3}}{x_1} \ln|y_i| \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right) f_i(X)$$

(e) $$\frac{\partial^2 F(X)}{\partial x_1^2} = 2\sum_{i=1}^{m} \frac{|y_i|^{x_3}}{x_1^2} \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right)\left(\frac{\partial f_i(X)}{\partial x_1} + \frac{|y_i|^{x_3} - 2x_1}{x_1^2} f_i(X)\right)$$

$$\frac{\partial^2 F(X)}{\partial x_2^2} = 2\sum_{i=1}^{m} \frac{x_3 |y_i|^{x_3-2}}{x_1} \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right) \left(\frac{|y_i|^2}{y_i}\frac{\partial f_i(X)}{\partial x_2} + \left(1 - x_3 + \frac{x_3|y_i|^{x_3}}{x_1}\right) f_i(X)\right)$$

$$\frac{\partial^2 F(X)}{\partial x_3^2} = 2\sum_{i=1}^{m} -\frac{|y_i|^{x_3}}{x_1} \ln|y_i| \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right) \left(\frac{\partial f_i(X)}{\partial x_3} - \frac{|y_i|^{x_3} - x_1}{x_1} \ln|y_i| f_i(X)\right)$$

$$\frac{\partial^2 F(X)}{\partial x_2 \partial x_1} = 2\sum_{i=1}^{m} \frac{x_3 |y_i|^{x_3}}{x_1 y_i} \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right) \left(\frac{\partial f_i(X)}{\partial x_1} + \frac{|y_i|^{x_3} - x_1}{x_1^2} f_i(X)\right)$$

$$\frac{\partial^2 F(X)}{\partial x_3 x_1} = 2\sum_{i=1}^{m} -\frac{|y_i|^{x_3}}{x_1} \ln|y_i| \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right) \left(\frac{\partial f_i(X)}{\partial x_1} + \frac{|y_i|^{x_3} - x_1}{x_1^2} f_i(X)\right)$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_2} = 2\sum_{i=1}^{m} \frac{|y_i|^{x_3}}{x_1 y_i} \exp\left(-\frac{|y_i|^{x_3}}{x_1}\right) \left(x_3 \frac{\partial f_i(X)}{\partial x_3} + \left(1 - \frac{|y_i|^{x_3} - x_1}{x_1} x_3 \ln|y_i|\right) f_i(X)\right)$$

(f) $F(X) = 0$ at $X = (50, 25, 1.5)^T$

2.13 Trigonometric function

This function introduced by Spedicato is based on the formula of a trigonometric function with random coefficients [24].

(a) $m = n$, n variable

(b) $f_i(X) = n + i(1 - \cos x_i) - \sin x_i - \sum_{j=1}^{n} \cos x_j, \qquad i = 1, \ldots, m$

(c) $X_0 = (1/n, \ldots, 1/n)^T$

(d) $\frac{\partial F(X)}{\partial x_j} = 2\left((j \sin x_j - \cos x_j) f_j(X) + \sin x_j \sum_{i=1}^{m} f_i(X)\right), \qquad j = 1, \ldots, n$

(e) $\frac{\partial^2 F(X)}{\partial x_j x_k} = 2((n + j + k) \sin x_j \sin x_k - \sin x_j \cos x_k - \sin x_k \cos x_j)$
$+ 2\delta_{jk}\left((j \sin x_j - \cos x_j)^2 + (j \cos x_j + \sin x_j) f_j(X) + \cos x_j \sum_{i=1}^{m} f_i(X)\right),$
$j, k = 1, \ldots, n$

(f) $F(X) = 0$ at $X = (0, \ldots, 0)^T$
other minima also exist

2.14 Extended Rosenbrock function

This function has a well-known banana-shaped valley near the minumum in two dimensions.

(a) $m = n$ variable but even

(b) $f_{2i-1}(X) = 10(x_{2i} - x_{2i-1}^2), \qquad i = 1, \ldots, m/2$

$f_{2i}(X) = 1 - x_{2i-1}, \qquad i = 1, \ldots, m/2$

(c) $X_0 = (-1.2, 1, \ldots, -1.2, 1)^T$

(d) $\dfrac{\partial F(X)}{\partial x_{2j-1}} = -2\left(x_{2j-1}\dfrac{\partial F(X)}{\partial x_{2j}} + f_{2j}(X)\right), \qquad j = 1, \ldots, n/2$

$\dfrac{\partial F(X)}{\partial x_{2j}} = 20 f_{2j-1}(X), \qquad j = 1, \ldots, n/2$

(e) $\dfrac{\partial^2 F(X)}{\partial x_j \partial x_{2k-1}} = -2\delta_{j,2k-1}(20 f_{2j-1}(X) - 400x_{2j-1}^2 - 1) - 400\delta_{j,2k}x_{2k-1},$
$j = 1, \ldots, n,\ k = 1, \ldots, n/2$

$\dfrac{\partial^2 F(X)}{\partial x_j \partial x_{2k}} = 200\delta_{j,2k} - 400\delta_{j,2k-1}x_{2k-1}, \qquad j = 1, \ldots, n,\ k = 1, \ldots, n/2$

(f) $F(X) = 0$ at $X = (1, \ldots, 1)^T$

2.15 Extended Powell singular function

(a) $m = n$ multiple of 4

(b) $f_{4i-3}(X) = x_{4i-3} + 10x_{4i-2}, \qquad i = 1, \ldots, m/4$

$f_{4i-2}(X) = \sqrt{5}(x_{4i-1} - x_{4i}), \qquad i = 1, \ldots, m/4$

$f_{4i-1}(X) = (x_{4i-2} - 2x_{4i-1})^2, \qquad i = 1, \ldots, m/4$

$f_{4i}(X) = \sqrt{10}(x_{4i-3} - x_{4i})^2, \qquad i = 1, \ldots, m/4$

(c) $X_0 = (3, -1, 0, 1, \ldots, 3, -1, 0, 1)^T$

(d) $$\frac{\partial F(X)}{\partial x_{4j-3}} = 2f_{4j-3}(X) + 4\sqrt{10}(x_{4j-3} - x_{4j})f_{4j}(X), \qquad j = 1, \ldots, n/4$$

$$\frac{\partial F(X)}{\partial x_{4j-2}} = 20f_{4j-3}(X) + 4(x_{4j-2} - 2x_{4j-1})f_{4j-1}(X), \qquad j = 1, \ldots, n/4$$

$$\frac{\partial F(X)}{\partial x_{4j-1}} = 2\sqrt{5}f_{4j-2}(X) - 8(x_{4j-2} - 2x_{4j-1})f_{4j-1}(X), \qquad j = 1, \ldots, n/4$$

$$\frac{\partial F(X)}{\partial x_{4j}} = -2\sqrt{5}f_{4j-2}(X) - 4\sqrt{10}(x_{4j-3} - x_{4j})f_{4j}(X), \qquad j = 1, \ldots, n/4$$

(e) $$\frac{\partial^2 F(X)}{\partial x_j \partial x_{4k-3}} = \delta_{j,4k-3}(2 + 12\sqrt{10}f_{4k}(X)) + 20\delta_{j,4k-2} - 12\delta_{j,4k}\sqrt{10}f_{4k}(X),$$
$$j = 1, \ldots, n,\ k = 1, \ldots, n/4$$

$$\frac{\partial^2 F(X)}{\partial x_j \partial x_{4k-2}} = \delta_{j,4k-2}(200 + 12f_{4k-1}(X)) + 20\delta_{j,4k-3} - 24\delta_{j,4k-1}f_{4k-1}(X),$$
$$j = 1, \ldots, n,\ k = 1, \ldots, n/4$$

$$\frac{\partial^2 F(X)}{\partial x_j \partial x_{4k-1}} = \delta_{j,4k-1}(10 + 48f_{4k-1}(X)) - 24\delta_{j,4k-2}f_{4k-1}(X) - 10\delta_{j,4k},$$
$$j = 1, \ldots, n,\ k = 1, \ldots, n/4$$

$$\frac{\partial^2 F(X)}{\partial x_j \partial x_{4k}} = \delta_{j,4k}(10 + 12\sqrt{10}f_{4k}(X)) - 12\delta_{j,4k-3}\sqrt{10}f_{4k}(X) - 10\delta_{j,4k-1},$$
$$j = 1, \ldots, n,\ k = 1, \ldots, n/4$$

(f) $F(X) = 0$ at $X = (0, \ldots, 0)^T$

2.16 Beale function

This function contains a narrow curving valley [1].

(a) $m = 3$, $n = 2$

(b) $f_i(X) = y_i - x_1(1 - x_2^i), \qquad i = 1, \ldots, m$

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

(c) $X_0 = (1,1)^T$

(d) $\dfrac{\partial F(X)}{\partial x_1} = -2\left((1-x_2)f_1(X) + (1-x_2^2)f_2(X) + (1-x_2^3)f_3(X)\right)$

$\dfrac{\partial F(X)}{\partial x_2} = 2x_1(f_1(X) + 2x_2 f_2(X) + 3x_2^2 f_3(X))$

(e) $\dfrac{\partial^2 F(X)}{\partial x_1^2} = 2((1-x_2)^2 + (1-x_2^2)^2 + (1-x_2^3)^2)$

$\dfrac{\partial^2 F(X)}{\partial x_2^2} = 2x_1\left(x_1 + 2(f_2(X) + 2x_2^2 x_1) + 3\big(2x_2(f_3(X) + 3x_1 x_2^4)\big)\right)$

$\dfrac{\partial^2 F(X)}{\partial x_2 \partial x_1} = 2\left(f_1(X) - x_1(1-x_2) + 2x_2\big(f_2(X) - (1-x_2^2)\big) + 3x_2^2(f_3(X) - (1-x_2^3))\right)$

(f) $F(X) = 0$ at $X = (3, 0.5)^T$

2.17 Wood function

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

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

$f_2(X) = 1 - x_1$

$f_3(X) = \sqrt{90}(x_4 - x_3^2)$

$f_4(X) = 1 - x_3$

$f_5(X) = \sqrt{10}(x_2 + x_4 - 2)$

$f_6(X) = (x_2 - x_4)/\sqrt{10}$

(c) $X_0 = (-3, -1, -3, -1)^T$

(d) $\dfrac{\partial F(X)}{\partial x_1} = -40x_1 f_1(X) - 2f_2(X)$

$\dfrac{\partial F(X)}{\partial x_2} = 2(10f_1(X) + \sqrt{10}f_5(X) + f_6(X)/\sqrt{10})$

$$\frac{\partial F(X)}{\partial x_3} = -4\sqrt{90}x_3 f_3(X) - 2f_4(X)$$

$$\frac{\partial F(X)}{\partial x_4} = 2(\sqrt{90}f_3(X) + \sqrt{10}f_5(X) - f_6(X)/\sqrt{10}\,)$$

(e) $$\frac{\partial^2 F(X)}{\partial x_1^2} = -2(20f_1(X) - 400x_1^2 - 1)$$

$$\frac{\partial^2 F(X)}{\partial x_2^2} = 220.2$$

$$\frac{\partial^2 F(X)}{\partial x_3^2} = -360(x_4 - 3x_3^2) + 2$$

$$\frac{\partial^2 F(X)}{\partial x_4^2} = 200.2$$

$$\frac{\partial^2 F(X)}{\partial x_2 \partial x_1} = -400x_1$$

$$\frac{\partial^2 F(X)}{\partial x_4 \partial x_2} = 19.8$$

$$\frac{\partial^2 F(X)}{\partial x_4 \partial x_3} = -360x_3$$

$$\frac{\partial^2 F(X)}{\partial x_3 \partial x_1} = \frac{\partial^2 F(X)}{\partial x_3 \partial x_2} = \frac{\partial^2 F(X)}{\partial x_4 \partial x_1} = 0$$

(f) $F(X) = 0$ at $X = (1, 1, 1, 1)^T$

2.18 Chebyquad function

(a) $m \geq n$ variable

(b) $$f_i(X) = \frac{1}{n}\left(\sum_{j=1}^{n} T_i(x_j)\right) - \int_0^1 T_i(y)dy, \qquad i = 1, \ldots, m$$

where $T_0(y) = 1$, $T_1(y) = 2y - 1$, and in general

$$T_i(y) = 2T_1(y)T_{i-1}(y) - T_{i-2}(y), \qquad i \geq 2$$

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

$$\int_0^1 T_i(y)dy = \begin{cases} 0 & \text{for } i \text{ odd} \\ 1/(1-i^2) & \text{for } i \text{ even} \end{cases}$$

(c) $X_0 = \big(1/(n+1), 2/(n+1), \ldots, n/(n+1)\big)^T$

(d) $\dfrac{\partial F(X)}{\partial x_j} = \dfrac{2}{n}\displaystyle\sum_{i=1}^{m} f_i(X)\frac{\partial T_i(x_j)}{\partial x_j}, \qquad j = 1, \ldots, n$

(e) $\dfrac{\partial^2 F(X)}{\partial x_j \partial x_k} = \dfrac{2}{n}\displaystyle\sum_{i=1}^{m}\left(\delta_{jk} f_i(X)\frac{\partial^2 T_i(x_j)}{\partial x_j^2} + \frac{1}{n}\frac{\partial T_i(x_j)}{\partial x_j}\frac{\partial T_i(x_k)}{\partial x_k}\right), \qquad j, k = 1, \ldots, n$

(f) $F(X) = 0$ for $m = n$, $1 \le n \le 7$, and $n = 9$

$F(X) \approx 0.00351687$ for $m = n = 8$

$F(X) \approx 0.00650395$ for $m = n = 10$

References

[1] E. M. L. Beale, "On an iterative method for finding a local minimum of a function of more than one variable," Tech. Rep. 25, Statistical Techniques Research Group, Princeton University (1958).

[2] M. C. Biggs, "Minimization algorithms making use of non-quadratic properties of the objective function," *J. Inst. Maths. Applics.* **8**, 315–327 (1971).

[3] M. J. Box, "A comparison of several current optimization methods, and the use of transformations in constrained problems," *Comput. J.* **9**, 67–77 (1966).

[4] K. M. Brown and J. E. Dennis, "New computational algorithms for minimizing a sum of squares of nonlinear functions," Report 71-6, Yale University, Department of Computer Science (1971).

[5] A. R. Colville, "A comparative study of nonlinear programming codes," Report 320-2949, IBM New York Scientific Center (1968).

[6] R. A. Cox, "Comparison of the performance of seven optimization algorithms on twelve unconstrained minimization problems," Ref. 1335CNO4, Gulf Research and Development Company, Pittsburg (1969).

[7] R. S. Dembo and T. Steihaug, "Truncated Newton algorithms for large-scale unconstrained optimization," *Math. Prog.* **26**, 190–212 (1983).

[8] R. Fletcher, "Function minimization without evaluating derivatives—a review," *Comput. J.* **8**, 33–41 (1965).

[9] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *Comput. J.* **6**, 163–168 (1963).

[10] J. C. Gilbert and C. Lemaréchal, "Some numerical experiments with variable-storage quasi-Newton algorithms," *Math. Prog.* **45**, 407–435 (1989).

[11] P. E. Gill, W. Murray, and R. A. Pitfield, "The implementation of two revised quasi-Newton algorithms for unconstrained optimization," Report NAC 11, National Phys. Lab. (1972).

[12] J. S. Kowalik and M. R. Osborne, *Methods for Unconstrained Optimization Problems*, New York, Elsevier North-Holland (1968).

[13] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large-scale optimization," *Math. Prog.* **45**, 503–528 (1989).

[14] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Testing unconstrained optimization software," *ACM Trans. Math. Soft.* **7**, 17–41 (1981).

[15] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Algorithm 566: FORTRAN subroutines for testing unconstrained optimization software," *ACM Trans. Math. Soft.* **7**, 136–140 (1981).

[16] S. G. Nash, "Solving nonlinear programming problems using truncated Newton techniques," in *Numerical Optimization 1984*, P. T. Boggs, R. H. Byrd, and R. B. Schnabel, eds., Philadelphia, 119–136, SIAM (1985).

[17] S. G. Nash and J. Nocedal, "A numerical study of the limited-memory BFGS method and the truncated-Newton method for large-scale optimization," *SIAM J. Opt.* **1**, 358–372 (1991).

[18] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Math. Comput.* **35**, 773–782 (1980).

[19] M. J. D. Powell, "A FORTRAN subroutine for solving systems of nonlinear algebraic equations," in *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, ed., New York, 115–161, Gordon & Breach (1970).

[20] T. Schlick and A. Fogelson, "TNPACK — a large scale truncated Newton package for large-scale minimization: I. algorithm and usage," *ACM Trans. Math. Soft.* **18**, 41–66 (1992).

[21] T. Schlick and A. Fogelson, "TNPACK — a large scale truncated Newton package for large-scale minimization: II. implementation examples," *ACM Trans. Math. Soft.* **18**, 67–111 (1992).

[22] T. Schlick and M. L. Overton, "A powerful truncated Newton method for potential energy functions," *J. Comp. Chem.* **8**, 1025–1039 (1987).

[23] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Maths. Comp.* **24**, 647–656 (1970).

[24] E. Spedicato, "Computational experience with quasi-Newton algorithms for minimization problems of moderately large size," in *Towards Global Optimization 2*, L. C. W. Dixon and G. P. Szegö, eds., Amsterdam, 209–219, North-Holland (1978).

[25] X. Zou, I. M. Navon, M. Berger, P. K. H. Phua, T. Schlick, and F. X. LeDimet, "Numerical experience with limited-memory quasi-newton and truncated newton methods," *SIAM J. Opt.* to appear.

Table I — Test Functions for Unconstrained Minimization

Function Name	Number of Functions m	Number of Variables n	Source	Hessian Structure
Powell badly scaled	2	2	[19]	dense
Brown badly scaled	3	2	unpub.	dense
Beale	3	2	[1]	dense
Helical valley	3	3	[9]	dense
Box three-dimensional	$m \geq 3$	3	[3]	dense
Wood	6	4	[5]	sparse
Brown and Dennis	$m \geq 4$	4	[4]	dense
Biggs EXP6	$m \geq 6$	6	[2]	dense
Gaussian	15	3	unpub.	dense
Gulf Research & Development	$3 \leq m \leq 100$	3	[6,23]	dense
Trigonometric	$m = n$	variable	[24]	dense
Extended Rosenbrock	$m = n$	even	[24]	sparse
Watson	31	$2 \leq n \leq 31$	[12]	dense
Extended Powell singular	$m = n$	multiple of 4	[24]	sparse
Variably dimensioned	$n + 2$	variable	unpub.	dense
Penalty I	$n + 1$	variable	[11]	dense
Penalty II	$2n$	variable	[11]	dense
Chebyquad	$m \geq n$	variable	[8]	dense

Appendix: Source Code

```
      subroutine hesfcn (n, x, hesd, hesl, nprob)
      integer n, nprob
      double precision x(n), hesd(n), hesl(*)
c     **********
c
c     subroutine hesfcn
c
c     This subroutine defines the Hessian matrices 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 hesfcn (n, x, hesd, hesl, nprob)
```

```
c
c     where
c
c       n is a positive integer input variable.
c
c       x is an input array of length n.
c
c       hesd is an output array of length n which contains the
c         diagonal components of the Hessian matrix of the nprob
c         objective function evaluated at x.
c
c       hesl is an output array of length n*(n-1)/2 which contains
c         the lower triangular part of the Hessian matrix of the
c         nprob objective function evaluated at x.
c
c       nprob is a positive integer input variable which defines the
c         number of the problem.  nprob must not exceed 18.
c
c     subprograms called
c
c       fortran-supplied ... abs, atan, cos, exp, log, sign, sin,
c                            sqrt
c
c     Taken from minpack. version of july 1978.
c     burton s. garbow, kenneth e. hillstrom, jorge j. more
c     Modified 1993 by Victoria Z. Averbukh and Samuel A. Figueroa.
c
c     **********
      double precision zero, one, two, three, four, five, six, eight,
     &      nine, ten, fifty, cp0001, cp1, cp2, cp25, cp5, c1p5, c2p25,
     &      c2p625, c3p5, c12, c19p8, c25, c29, c50, c90, c100, c120,
     &      c180, c200, c200p2, c202, c220p2, c360, c400, c1000, c1080,
     &      c1200, c2000, c20000, c2e8, c4e8, ap, bp, pi
      parameter (zero=0.0d0, one=1.0d0, two=2.0d0, three=3.0d0,
     &      four=4.0d0, five=5.0d0, six=6.0d0, eight=8.0d0, nine=9.0d0,
     &      ten=1.0d1, fifty=5.0d1, cp0001=1.0d-4, cp1=1.0d-1,
     &      cp2=2.0d-1, cp25=2.5d-1, cp5=5.0d-1, c1p5=1.5d0,
     &      c2p25=2.25d0, c2p625=2.625d0, c3p5=3.5d0, c12=1.2d1,
     &      c19p8=1.98d1, c25=2.5d1, c29=2.9d1, c50=5.0d1, c90=9.0d1,
     &      c100=1.0d2, c120=1.2d2, c180=1.8d2, c200=2.0d2,
     &      c200p2=2.002d2, c202=2.02d2, c220p2=2.202d2, c360=3.6d2,
```

```
     &      c400=4.0d2, c1000=1.0d3, c1080=1.08d3, c1200=1.2d3,
     &      c2000=2.0d3, c20000=2.0d4, c2e8=2.0d8, c4e8=4.0d8,
     &      ap=1.0d-5, bp=one, pi=3.141592653589793d0)
      integer i, j, k, m, ii, jj, ix, ivar
      double precision arg, d1, d2, d3, logr, p1, p2, piarg, piarg2,
     *      r, r3inv, s1, s2, s3, s1s2, s1s3, s2s3, ss1, ss2,
     *      t, t1, t2, t3, th, tt, tt1, tt2, tth
      double precision fvec(50), gvec(50), y(15)
      logical iev
      double precision dfloat
      ix(ii,jj)=(ii-1)*(ii-2)/2+jj

C The function IX (defined above) gives the location of a Hessian
C element (ii,jj), ii>jj, in a 1-dim. array in which the lower part
C of H is stored row-by-row.

      dfloat(ivar) = ivar
      data y /9.0d-4, 4.4d-3, 1.75d-2, 5.4d-2, 1.295d-1, 2.42d-1,
     &      3.521d-1, 3.989d-1, 3.521d-1, 2.42d-1, 1.295d-1, 5.4d-2,
     &      1.75d-2, 4.4d-3, 9.0d-4/

c
c     Hessian routine selector.
c
      go to (100, 200, 300, 400, 500, 600, 700, 800, 900, 1000,
     *      1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800), nprob
c
c     Helical valley function.
c
  100 continue
      if (x(1) .eq. zero) then
         th = sign(cp25,x(2))
      else
         th = atan(x(2)/x(1)) / (two*pi)
         if (x(1) .lt. zero) th = th + cp5
      end if
      arg = x(1)**2 + x(2)**2
      piarg = pi * arg
      piarg2 = piarg * arg
      r3inv = one / sqrt(arg)**3
      t = x(3) - ten*th
```

```
      s1 = five*t / piarg
      p1 = c2000*x(1)*x(2)*t / piarg2
      p2 = (five/piarg)**2
      hesd(1) = c200 - c200*(r3inv+p2)*x(2)**2 - p1
      hesd(2) = c200 - c200*(r3inv-p2)*x(1)**2 + p1
      hesd(3) = c202
      hesl(1) = c200*x(1)*x(2)*r3inv +
     1      c1000/piarg2 * ( t*(x(1)**2-x(2)**2) - five*x(1)*x(2)/pi )
      hesl(2) =  c1000*x(2) / piarg
      hesl(3) = -c1000*x(1) / piarg
      return
c
c     Biggs EXP6 function.
c
  200 continue
      do 210, i = 1, 6
         hesd(i) = zero
  210 continue
      do 220, i = 1, 15
         hesl(i) = zero
  220 continue
      do 230, i = 1, 13
         d1 = dfloat(i)/ten
         d2 = exp(-d1) - five*exp(-ten*d1) + three*exp(-four*d1)
         s1 = exp(-d1*x(1))
         s2 = exp(-d1*x(2))
         s3 = exp(-d1*x(5))
         t = x(3)*s1 - x(4)*s2 + x(6)*s3 - d2
         d2 = d1**2
         s1s2 = s1 * s2
         s1s3 = s1 * s3
         s2s3 = s2 * s3
         hesd(1) = hesd(1) + d2*s1*(t+x(3)*s1)
         hesd(2) = hesd(2) - d2*s2*(t-x(4)*s2)
         hesd(3) = hesd(3) + s1**2
         hesd(4) = hesd(4) + s2**2
         hesd(5) = hesd(5) + d2*s3*(t+x(6)*s3)
         hesd(6) = hesd(6) + s3**2
         hesl(1) = hesl(1) - d2*s1s2
         hesl(2) = hesl(2) - d1*s1*(t+x(3)*s1)
         hesl(3) = hesl(3) + d1*s1s2
```

```
         hesl(4) = hesl(4) + d1*s1s2
         hesl(5) = hesl(5) + d1*s2*(t-x(4)*s2)
         hesl(6) = hesl(6) - s1s2
         hesl(7) = hesl(7) + d2*s1s3
         hesl(8) = hesl(8) - d2*s2s3
         hesl(9) = hesl(9) - d1*s1s3
         hesl(10) = hesl(10) + d1*s2s3
         hesl(11) = hesl(11) - d1*s1s3
         hesl(12) = hesl(12) + d1*s2s3
         hesl(13) = hesl(13) + s1s3
         hesl(14) = hesl(14) - s2s3
         hesl(15) = hesl(15) - d1*s3*(t+x(6)*s3)
  230 continue
      hesd(1) = x(3)*hesd(1)
      hesd(2) = x(4)*hesd(2)
      hesd(5) = x(6)*hesd(5)
      hesl(1) = x(3)*x(4)*hesl(1)
      hesl(3) = x(4)*hesl(3)
      hesl(4) = x(3)*hesl(4)
      hesl(7) = x(3)*x(6)*hesl(7)
      hesl(8) = x(4)*x(6)*hesl(8)
      hesl(9) = x(6)*hesl(9)
      hesl(10) = x(6)*hesl(10)
      hesl(11) = x(3)*hesl(11)
      hesl(12) = x(4)*hesl(12)
      do 240, i = 1, 6
         hesd(i) = two*hesd(i)
  240 continue
      do 250, i = 1, 15
         hesl(i) = two*hesl(i)
  250 continue
      return
c
c     Gaussian function.
c
  300 continue
      hesd(1) = zero
      hesd(2) = zero
      hesd(3) = zero
      hesl(1) = zero
      hesl(2) = zero
```

```
      hesl(3) = zero
      do 310, i = 1, 15
         d1 = cp5*dfloat(i-1)
         d2 = c3p5 - d1 - x(3)
         arg = cp5*x(2)*d2**2
         r = exp(-arg)
         t = x(1)*r - y(i)
         t1 = two*x(1)*r - y(i)
         hesd(1) = hesd(1) + r**2
         hesd(2) = hesd(2) + r*t1*d2**4
         hesd(3) = hesd(3) + r*(x(2)*t1*d2**2-t)
         hesl(1) = hesl(1) - r*t1*d2**2
         hesl(2) = hesl(2) + d2*r*t1
         hesl(3) = hesl(3) + d2*r*(t+arg*t1)
  310 continue
      hesd(1) = two*hesd(1)
      hesd(2) = cp5*x(1)*hesd(2)
      hesd(3) = two*x(1)*x(2)*hesd(3)
      hesl(2) = two*x(2)*hesl(2)
      hesl(3) = two*x(1)*hesl(3)
      return
c
c     Powell badly scaled function.
c
  400 continue
      s1 = exp(-x(1))
      s2 = exp(-x(2))
      t2 = s1 + s2 - one - cp0001
      hesd(1) = c2e8*x(2)**2 + two*s1*(s1+t2)
      hesd(2) = c2e8*x(1)**2 + two*s2*(s2+t2)
      hesl(1) = c4e8*x(1)*x(2) + two*s1*s2 - c20000
      return
c
c     Box 3-dimensional function.
c
  500 continue
      hesd(1) = zero
      hesd(2) = zero
      hesd(3) = zero
      hesl(1) = zero
      hesl(2) = zero
```

```
      hesl(3) = zero
      do 510, i = 1, 10
         d1 = dfloat(i)
         d2 = d1/ten
         s1 = exp(-d2*x(1))
         s2 = exp(-d2*x(2))
         s3 = exp(-d2) - exp(-d1)
         t = s1 - s2 - s3*x(3)
         th = t*d2**2
         hesd(1) = hesd(1) + th*s1 + (d2*s1)**2
         hesd(2) = hesd(2) - th*s2 + (d2*s2)**2
         hesd(3) = hesd(3) + s3**2
         hesl(1) = hesl(1) - s1*s2*d2**2
         hesl(2) = hesl(2) + d2*s1*s3
         hesl(3) = hesl(3) - d2*s2*s3
  510 continue
      hesd(1) = two*hesd(1)
      hesd(2) = two*hesd(2)
      hesd(3) = two*hesd(3)
      hesl(1) = two*hesl(1)
      hesl(2) = two*hesl(2)
      hesl(3) = two*hesl(3)
      return
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 = one + six*t1**2
      m = 0
      do 630, j = 1, n
         hesd(j) = two + two*t*dfloat(j)**2
         do 620, k = 1, j-1
            m = m + 1
            hesl(m) = two*t*dfloat(j*k)
  620    continue
  630 continue
      return
```

```
c
c     Watson function.
c
  700 continue
      do 710, j = 1, n
         hesd(j) = zero
  710 continue
      do 720, j = 1, n*(n-1)/2
         hesl(j) = zero
  720 continue
      do 760, i = 1, 29
         d1 = dfloat(i)/c29
         d2 = one
         s1 = zero
         s2 = x(1)
         do 730, j = 2, n
            s1 = s1 + dfloat(j-1)*d2*x(j)
            d2 = d1*d2
            s2 = s2 + d2*x(j)
  730    continue
         t = two * (s1-s2**2-one) * d1**2
         s3 = two*d1*s2
         d2 = one/d1
         m = 0
         do 750, j = 1, n
            t1 = dfloat(j-1) - s3
            hesd(j) = hesd(j) + (t1**2-t)*d2**2
            d3 = one/d1
            do 740, k = 1, j-1
               m = m + 1
               hesl(m) = hesl(m) + (t1*(dfloat(k-1)-s3) - t) * d2*d3
               d3 = d1*d3
  740       continue
            d2 = d1*d2
  750    continue
  760 continue
      t3 = x(2) - x(1)**2 - one
      hesd(1) = hesd(1) + one - two*(t3-two*x(1)**2)
      hesd(2) = hesd(2) + one
      hesl(1) = hesl(1) - two*x(1)
      do 770, j = 1, n
```

```
            hesd(j) = two * hesd(j)
      770 continue
          do 780, j = 1, n*(n-1)/2
            hesl(j) = two * hesl(j)
      780 continue
          return
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
          m = 0
          do 830, j = 1, n
            hesd(j) = d1 + th + eight*x(j)**2
            do 820, k = 1, j-1
              m = m + 1
              hesl(m) = eight*x(j)*x(k)
      820     continue
      830 continue
          return
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 = exp(cp1)
          d2 = one
          th = four*bp*t1
          m = 0
          do 930, j = 1, n
            hesd(j) = eight*bp*(dfloat(n-j+1)*x(j))**2 + dfloat(n-j+1)*th
            s1 = exp(x(j)/ten)
            if (j .gt. 1) then
```

```
            s3 = s1 + s2 - d2*(d1 + one)
            hesd(j) = hesd(j) + ap*s1*(s3 + s1 - one/d1 + two*s1)/c50
            hesd(j-1) = hesd(j-1) + ap*s2*(s2+s3)/c50
            do 920, k = 1, j-1
               m = m + 1
               t1 = exp(dfloat(k)/ten)
               hesl(m) = eight*dfloat(n-j+1)*dfloat(n-k+1)*x(j)*x(k)
  920       continue
            hesl(m) = hesl(m) + ap*s1*s2/c50
         end if
         s2 = s1
         d2 = d1*d2
  930 continue
      hesd(1) = hesd(1) + two*bp
      return
c
c     Brown badly scaled function.
c
 1000 continue
      hesd(1) = two + two*x(2)**2
      hesd(2) = two + two*x(1)**2
      hesl(1) = four*x(1)*x(2) - four
      return
c
c     Brown and Dennis function.
c
 1100 continue
      do 1110, i = 1, 4
         hesd(i) = zero
 1110 continue
      do 1120, i = 1, 6
         hesl(i) = zero
 1120 continue
      do 1130, i = 1, 20
         d1 = dfloat(i)/five
         d2 = sin(d1)
         t1 = x(1) + d1*x(2) - exp(d1)
         t2 = x(3) + d2*x(4) - cos(d1)
         t = eight * t1 * t2
         s1 = c12*t1**2 + four*t2**2
         s2 = c12*t2**2 + four*t1**2
```

```
         hesd(1) = hesd(1) + s1
         hesd(2) = hesd(2) + s1*d1**2
         hesd(3) = hesd(3) + s2
         hesd(4) = hesd(4) + s2*d2**2
         hesl(1) = hesl(1) + s1*d1
         hesl(2) = hesl(2) + t
         hesl(4) = hesl(4) + t*d2
         hesl(3) = hesl(3) + t*d1
         hesl(5) = hesl(5) + t*d1*d2
         hesl(6) = hesl(6) + s2*d2
 1130 continue
      return
c
c     Gulf Research and Development function.
c
 1200 continue
      do 1210, i = 1, 3
         hesd(i) = zero
         hesl(i) = zero
 1210 continue
      d1 = two/three
      do 1220, i = 1, 99
         arg = dfloat(i)/c100
         r = (-fifty*log(arg))**d1+c25-x(2)
         t1 = abs(r)**x(3)/x(1)
         t2 = exp(-t1)
         t3 = t1 * t2 * (t1*t2+(t1-one)*(t2-arg))
         t = t1 * t2 * (t2-arg)
         logr = log(abs(r))
         hesd(1) = hesd(1) + t3 - t
         hesd(2) = hesd(2) + (t+x(3)*t3)/r**2
         hesd(3) = hesd(3) + t3*logr**2
         hesl(1) = hesl(1) + t3/r
         hesl(2) = hesl(2) - t3*logr
         hesl(3) = hesl(3) + (t-x(3)*t3*logr)/r
 1220 continue
      hesd(1) = hesd(1) / x(1)**2
      hesd(2) = hesd(2) * x(3)
      hesl(1) = hesl(1) * x(3)/x(1)
      hesl(2) = hesl(2) / x(1)
      do 1230, i = 1, 3
```

```
         hesd(i) = two * hesd(i)
         hesl(i) = two * hesl(i)
 1230 continue
      return
c
c     Trigonometric function.
c
 1300 continue
      s1 = zero
      do 1310, j = 1, n
         hesd(j) = sin(x(j))
         s1 = s1 + cos(x(j))
 1310 continue
      s2 = zero
      m = 0
      do 1330, j = 1, n
         th = cos(x(j))
         t = dfloat(n+j) - hesd(j) - s1 - dfloat(j)*th
         s2 = s2 + t
         do 1320, k = 1, j-1
            m = m + 1
            hesl(m) = sin(x(k))*(dfloat(n+j+k)*hesd(j)-th) -
     *            hesd(j)*cos(x(k))
            hesl(m) = two*hesl(m)
 1320    continue
         hesd(j) = dfloat(j*(j+2)+n)*hesd(j)**2 +
     *         th*(th-dfloat(2*j+2)*hesd(j)) + t*(dfloat(j)*th+hesd(j))
 1330 continue
      do 1340, j = 1, n
         hesd(j) = two*(hesd(j) + cos(x(j))*s2)
 1340 continue
      return
c
c     Extended Rosenbrock function.
c
 1400 continue
      do 1410, j = 1, n*(n-1)/2
         hesl(j) = zero
 1410 continue
      do 1420, j = 1, n, 2
         hesd(j+1) = c200
```

```
         hesd(j) = c1200*x(j)**2 - c400*x(j+1) + two
         hesl(ix(j+1,j)) = -c400*x(j)
 1420 continue
      return
c
c     Extended Powell function.
c
 1500 continue
      do 1510, j = 1, n*(n-1)/2
         hesl(j) = zero
 1510 continue
      do 1520, j = 1, n, 4
         t2 = x(j+1) - two*x(j+2)
         t3 = x(j) - x(j+3)
         s1 = c12 * t2**2
         s2 = c120 * t3**2
         hesd(j) = two + s2
         hesd(j+1) = c200 + s1
         hesd(j+2) = ten + four*s1
         hesd(j+3) = ten + s2
         hesl(ix(j+1,j)) = two*ten
         hesl(ix(j+2,j)) = zero
         hesl(ix(j+2,j+1)) = -two*s1
         hesl(ix(j+3,j)) = -s2
         hesl(ix(j+3,j+1)) = zero
         hesl(ix(j+3,j+2)) = -ten
 1520 continue
      return
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
      hesd(1) = two * (s1**2 + s2**2 + s3**2)
      hesd(2) = two*x(1) * (x(1) + two*t2 + four*x(1)*x(2)**2 +
     1      six*x(2)*t3 + nine*x(1)*x(2)**4)
```

```
      hesl(1) = two*(t1-x(1)*s1) + four*x(2)*(t2-x(1)*s2) +
     2      six*(t3-x(1)*s3)*x(2)**2
      return
c
c     Wood function.
c
 1700 continue
      hesd(1) = c1200*x(1)**2 - c400*x(2) + two
      hesd(2) = c220p2
      hesd(3) = c1080*x(3)**2 - c360*x(4) + two
      hesd(4) = c200p2
      hesl(1) = -c400*x(1)
      hesl(2) = zero
      hesl(3) = zero
      hesl(4) = zero
      hesl(5) = c19p8
      hesl(6) = -c360*x(3)
      return
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/float(n)
      iev = .false.
      do 1840, i = 1, n
         fvec(i) = d1*fvec(i)
         if (iev) fvec(i) = fvec(i) + one/(dfloat(i)**2 - one)
```

```
        iev = .not. iev
1840 continue
     d2 = two*d1
     m = 0
     do 1880, j = 1, n
        hesd(j) = four*d1
        t1 = one
        t2 = two*x(j) - one
        t = two*t2
        s1 = zero
        s2 = two
        p1 = zero
        p2 = zero
        gvec(1) = s2
        do 1850, i = 2, n
           th = four*t2 + t*s2 - s1
           s1 = s2
           s2 = th
           th = t*t2 - t1
           t1 = t2
           t2 = th
           th = eight*s1 + t*p2 - p1
           p1 = p2
           p2 = th
           gvec(i) = s2
           hesd(j) = hesd(j) + fvec(i)*th + d1*s2**2
1850    continue
        hesd(j) = d2*hesd(j)
        do 1870, k = 1, j-1
           m = m + 1
           hesl(m) = zero
           tt1 = one
           tt2 = two*x(k) - one
           tt = two*tt2
           ss1 = zero
           ss2 = two
           do 1860, i = 1, n
              hesl(m) = hesl(m) + ss2*gvec(i)
              tth = four*tt2 + tt*ss2 - ss1
              ss1 = ss2
              ss2 = tth
```

```
               tth = tt*tt2 - tt1
               tt1 = tt2
               tt2 = tth
 1860       continue
            hesl(m) = d2*d1*hesl(m)
 1870    continue
 1880 continue
      return
c
c     last card of subroutine hesfcn.
c
      end
```