JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 18, No. 4, APRIL 1976

Solving the Nonlinear Least Square Problem: Application of a General Method[1,2]

J. T. BETTS[3]

Communicated by R. Howard

Abstract. An algorithm for solving the general nonlinear least-square problem is developed. An estimate for the Hessian matrix is constructed as the sum of two matrices. The first matrix is the usual first-order estimate used by the Gauss method, while the second matrix is generated recursively using a rank-one formula. Test results indicate that the method is superior to the standard Gauss method and compares favorably with other methods, especially for problems with nonzero residuals at the solution.

Key Words. Nonlinear least-square problems, parameter optimization, penalty function methods, mathematical programming.

1. Introduction

In an earlier work (Ref. 1), we developed a general method for solving constrained parameter optimization problems using a penalty-function treatment of the constraints. The nonlinear least-square problem represents an important special case of the general algorithm in which the total objective function consists of just the penalty term.

This paper presents a specialization of the general algorithm of Ref. 1 to the nonlinear least-square problem. The Hessian matrix consists of the usual

[1] This work was supported by the US Air Force under Contract No. F04701-73-C-0074.

[2] The author expresses his appreciation to Dr. H. E. Pickett and Dr. J. L. Searcy for their continuing support in the theoretical and practical development of the algorithm. The recursive method for generating the estimate of the Hessian matrix was developed jointly with Drs. Pickett and Searcy and is included here with their permission. The author would also like to acknowledge the contribution made by the stimulating environment of an optimal control seminar held at The Aerospace Corporation since 1970. Principle members of the seminar have been H. E. Pickett, J. L. Searcy, R. W. Reid, and the author.

[3] Member of the Technical Staff, Performance Analysis Department, Guidance and Control Division, The Aerospace Corporation, El Segundo, California.

first-order approximation used in the Gauss method augmented by a matrix generated recursively. Use of the *residual* matrix, which is generated using a rank-one recursion formula, overcomes the usual difficulties with the Gauss method when the sum of squares of the constraints is nonzero at the solution. A number of test problems drawn from the literature illustrate the effectiveness of the algorithm.

2. Preliminary Developments

The problem of interest in this paper is to find the n-vector x that minimizes the scalar function

$$J(x) = \sum_{i=1}^{m} c_i^2(x) = c^T(x)c(x), \tag{1}$$

where $c(x)$ is an m-vector. A performance index of this form can be obtained from the general algorithm, by setting $r = 1$ and $f(x) = 0$ in Eqs. (4)–(6) of Ref. 1. In many applications, the functions $c_i(x)$ represent residuals between an approximating function and experimentally obtained data.

Explicit expressions for the first and second derivatives can be obtained from Eqs. (15)–(18) of Ref. 1. In particular, we obtain the gradient vector

$$\nabla J = g = 2 \sum_{i=1}^{m} c_i(x)\nabla c_i(x). \tag{2}$$

The Hessian matrix is given by

$$\nabla^2 J = 2 \sum_{i=1}^{m} c_i(x)\nabla^2 c_i(x) + 2 \sum_{i=1}^{m} \nabla c_i(x)\nabla c_i^T(x). \tag{3}$$

It is assumed that the functions $c_i(x)$ are continuously differentiable to second order; consequently, the Hessian matrices are all symmetric. Let us define the following symmetric $n \times n$ matrices:

$$H = \nabla^2 J, \tag{4}$$

$$U = 2 \sum_{i=1}^{m} c_i(x)\nabla^2 c_i(x), \tag{5}$$

$$V = 2 \sum_{i=1}^{m} \nabla c_i(x)\nabla c_i^T(x). \tag{6}$$

From (3), then, the Hessian matrix of J is

$$H = U + V. \tag{7}$$

3. Determination of Search Step

As in the general method, the kth iteration of the algorithm begins at the point x^k and proceeds in a search direction defined by the vector s^k, to the point x^{k+1} where

$$x^{k+1} = x^k - \rho^k s^k. \tag{8}$$

The vector s^k is defined to be the vector that minimizes

$$\|g^{k+1}\| = \|g^k - H^k s^k\|. \tag{9}$$

The minimum-norm solution of this linear least-square problem is used when H^k is singular. The numerical procedure used for solving the linear least-square problem is presented in Ref. 2. So that each search direction will be downhill, the value of s^k obtained from the least-square process is multiplied by the appropriate sign. The scalar ρ^k is chosen such that

$$J(x^{k+1}) < J(x^k),$$

using a cubic search technique with the initial estimate $\rho^k = 1$. Details of this development are found in Section 7 of Ref. 1.

4. Construction of the Hessian Matrix

The method for determining the search vector s, described in the preceding section, requires an estimate of the Hessian matrix H. Section 9 of Ref. 1 presents a method of recursively estimating the matrix U, which we summarize for this specific application. Let us define

$$p^k = x^{k+1} - x^k, \tag{10}$$

$$q^k = g^{k+1} - g^k - V^k p^k. \tag{11}$$

If we insist that $q^k = U^k p^k$, then application of a rank-one recursive formula yields

$$U^{k+1} = U^k + (q^k - U^k p^k)(q^k - U^k p^k)^T / (p^k)^T (q^k - U^k p^k). \tag{12}$$

This rank-one formula has been reported by a number of authors, including Broyden (Ref. 3) and Murtagh and Sargent (Ref. 4).

Now, notice that, once an initial estimate for U has been defined, Eq. (12) can be used to provide an estimate of U at every iteration. The only information necessary is the gradient information at successive points and the points themselves. The matrix V can be computed at any point directly from the definition (6). If the residuals are linear, then clearly $\nabla^2 c_i(x) = 0$, in which case $U=0$ from Eq. (5). Furthermore, as the solution is approached, $c_i(x) \to 0$; and again, from Eq. (5), $U \to 0$. In the classical Gauss method, the contribution of U is neglected, and $H=V$. Indeed, the Gauss approximation is quite good if the residuals are linear or nearly zero. The contribution of U becomes significant for nonlinear residuals and points where the $c_i(x)$ are large, which is the situation in many practical problems. In fact, the methods of Levenberg (Ref. 5) and Marquardt (Ref. 6) both represent U by the diagonal matrix λI. Lacking better information, a reasonable estimate to begin the process is $U^0 = 0$.

5. Modified Version of the Basic Algorithm

A detailed description of the basic optimization algorithm can be found in Section 10 of Ref. 1. Essentially, it consists of a sequence of steps of the form (8). The search direction s^k is determined by a linear least-square procedure and involves an estimate of the Hessian matrix H^k. The Hessian estimate, in turn, consists of the Gauss term V^k given by Eq. (6) and the residual term U^k, which is generated recursively by use of Eqs. (10)–(12). If we define the *resolution* by

$$\sigma = \|p^k\| \cdot (\|x^k\| + 1)^{-1}, \tag{13}$$

then the algorithm is assumed to have converged when both

$$\|g^k\| < \delta_1 \qquad \text{and } \sigma < \delta_2.$$

The principal feature of the basic algorithm is the addition of U to the estimate of the Hessian matrix H. It is expected that the contribution of U will significantly affect the search direction when x^k is in the neighborhood of the solution, especially if the sum of squares of the residuals does not approach zero. Therefore, there is justification for using the U contribution in some neighborhood of the answer. Also, since a rank-one recursive algorithm is used to generate U, in general the complete estimate is not available until n iterations have been made.

To make these concepts more precise, let us consider modifying the basic algorithm as follows:

(a) if $\sigma \le \epsilon$ and $k > n$, set $H^k = U^k + V^k$, and compute s^k from (9);

(b) if $\sigma > \epsilon$ or $k \le n$, define s^k as the vector which minimizes

$$\|c(x^k) - G(x^k)s^k\|, \tag{14}$$

where σ is given by (13) and the Jacobian matrix is defined as

$$G(x) = [\nabla c_1(x), \ldots, \nabla c_m(x)]. \tag{15}$$

In essence, this modification defines a neighborhood of the solution as any point satisfying the resolution test $\sigma \leq \epsilon$. Within this neighborhood, the contribution of U^k affects the determination of s^k as long as n iterations have been made. Otherwise, the search direction s^k is computed directly from (14), which is the Gauss direction. This is equivalent to setting $H^k = V^k$ in (9), but is more desirable from a numerical standpoint.

6. Performance of the Algorithm

The algorithm described has been implemented in a computer program (Ref. 7) and has been tested on 11 test problems of varying complexity. The problems are arranged approximately in the order of difficulty and summarized in the Appendix. Table 1 presents the results of the basic algorithm on these test problems. Column one gives the problem number, and column two shows the number of function and gradient evaluations required to solve the problem. Analytic derivatives were used throughout.

The convergence parameters used were $\delta_1 = \delta_2 = 10^{-5}$. This ensures approximately five significant figures in the solution values. It is important to note that the definition of convergence can also greatly influence the effectiveness of one algorithm when compared with another. The third

Table 1. Results of the basic algorithm.

Problem	Required function evaluations	$J(x^*)$	x^*
8.1	20	0.408928×10^{-20}	(1.00000, 1.00000, 1.00000)
8.2	36	0.232164×10^{-19}	(1.00000, 1.00000)
8.3	16	0.185902×10^{-13}	$(-3.14145 \times 10^{-4}, 3.14145 \times 10^{-5}$ $-1.33049 \times 10^{-4}, -1.33049 \times 10^{-4})$
8.4	23	0.788258×10^{-17}	(3.00000, 0.500000)
8.5	2	0.144400×10^{-19}	$(2.05539 \times 10^{-11}, -3.05297 \times 10^{-11})$
8.6	9	0.489842×10^{2}	$(1.14127 \times 10^{1}, -8.96805 \times 10^{-1})$
8.7	16	0.821487×10^{-2}	$(8.24105 \times 10^{-2}, 1.13303, 2.34369)$
8.8	18	0.124362×10^{3}	$(2.57825 \times 10^{-1}, 2.57825 \times 10^{-1})$
8.9	15	0.307505×10^{-3}	$(1.92806 \times 10^{-1}, 1.91282 \times 10^{-1}$ $1.23056 \times 10^{-1}, 1.36062 \times 10^{-1})$
8.10	44	0.546489×10^{-4}	$(3.75410 \times 10^{-1}, 1.93584, -1.46468,$ $1.28675 \times 10^{-2}, 2.21227 \times 10^{-2})$
8.11	29	0.401377×10^{-1}	(1.30997, 0.431554, 0.633661, 0.599430, 0.754183, 0.904286, 1.36581, 4.82369, 2.39868, 4.56887, 5.67534)

column in Table 1 presents the minimum value of the sum of squares $J(x^*)$, and the fourth column gives the actual solution point x^*.

Table 2 compares the performance of the modified algorithm to the basic algorithm, the standard Gauss algorithm, and the best reported results. The first column defines the problem number, and the second column presents the best results obtained using the modified algorithm. Four different values of ϵ were tested, and this column shows the number of function and gradient evaluations required to solve the problem, with the corresponding value of ϵ in parentheses. Column three presents the results of the basic algorithm, which uses the contribution of U^k to generate all of the search directions. The fourth column gives the results of the standard Gauss method, which does not use the U^k matrix. Column five tabulates the best reported results for the given problem. Unfortunately, because of the diversity of the various methods, it is difficult to establish any uniform criterion by which to compare the methods. Consequently, each result quoted is followed by a symbol indicating the meaning of the quantity in the source paper.

It is the author's opinion that the process of numerical differentiation is a distinct numerical problem that should be considered independent of the optimization process. No attempt has been made to present an *equivalent*

Table 2. Comparison of results.

	Algorithm			
Problem	Modified	Basic	Gauss	Best reported
8.1	7 (0.01)	20	7	28(f)
8.2	27 (0.1)	36	27	17(f)
8.3	7 (0.1)	16	16	66(f)
8.4	10 (0.01)	23	10	20(g)
8.5	2 (0.1)	2	2	(n)
8.6	9 (0.1)	9	158(d)	121(f)
8.7	7 (0.1)	16	7	36(f)
8.8	19 (0.1)	18	176(d)	100(i)
8.9	12 (0.1)	15	25	60(g)
8.10	10 (0.01)	44	10	34(g)
8.11	16 (0.1)	29	16	14(g)

(d) Algorithm failed to converge after 100 iterations.
(f) Function evaluations or equivalent function evaluations.
(g) Function and gradient evaluations.
(i) Iterations.
(n) No results reported.

number of function evaluations, because such a quantity is highly dependent upon the numerical differentiation procedure. Such quantities as perturbation sizes and error tolerance can greatly influence the accuracy of numerical derivatives and, consequently, obscure the overall behavior of the optimization process. Let it suffice to say that the algorithm presented requires gradient information. If this information must be obtained numerically one would expect a twofold degradation: (i) because each gradient evaluation will require one or more additional function evaluations and (ii) because inaccurate gradient information will necessitate more optimization iterations.

The first five problems all have zero residuals at the solution and, consequently, one would expect the Gauss method to perform well.

Problem 8.1 was presented by Box (Ref. 8), and the best results of the eight methods tested required 28 function evaluations to reduce the sum of squares to 10^{-5}. The modified algorithm required 7 function evaluations, although the convergence tolerance was much tighter [$J(x^*) = 0.5 \times 10^{-14}$].

Problem 8.2 was first presented by Rosenbrock (Ref. 9) and has been widely used as a test problem. Jones (Ref. 10) reports solving this problem in 17 function evaluations, which is significantly better than the current algorithm. The current algorithm, however, appears to be considerably better than the results obtained by use of the well-known Marquardt algorithm, which required 92 evaluations, and the Powell method, which required 143 evaluations.

Problem 8.3 was originally presented by Powell (Ref. 11) and is significant because the Hessian matrix is singular at the solution. This problem is especially sensitive to variations in the convergence criteria. The best results for this problem, also obtained by Jones, required 66 function evaluations, and the Marquardt algorithm required 98 function evaluations. The current algorithm solved the problem in seven function and gradient evaluations.

Problem 8.4 was originally presented in Beale (Ref. 12) and studied by Kowalik and Osborn (Ref. 13), who reported that Davidon's method required 20 function and gradient evaluations to solve the problem. The ten function and gradient evaluations required by the current algorithm is considerably better than this result.

Problem 8.5 was presented by Branin (Ref. 14) and is troublesome for Branin's method, because the Jacobian is singular along an ellipse. No difficulty was encountered with the current algorithm.

Problem 8.6 was originally presented by Freudenstein and Roth (Ref. 15) and studied by Fletcher (Ref. 16). The Jacobian is singular along a line, and the sum of squares is significantly different than zero at the local minimum. Fletcher's method required 121 function evaluations for a solution. The current algorithm required only 9 function and gradient evaluations. Notice that the Gauss method failed on this problem.

Problem 8.7 is given in Bard (Ref. 17) where a number of algorithms were tested. Of all thirteen algorithms tested, the best required 36 function evaluations, as compared to the 7 function and gradient evaluations required by the current algorithm.

Problem 8.8 was given by Jennrich and Sampson (Ref. 18) and has a Jacobian that is not of full rank at the minimum and is ill-conditioned in the neighborhood of the minimum. The Gauss method used by Jennrich and Sampson required 100 iterations to reach a solution, and the Gauss method tested by the author failed to converge. The poor performance of the Gauss method is principally due to the fact that the residual is nonzero at the solution. The new algorithm solved this problem in 19 function and gradient evaluations.

Kowalik and Osborn (Ref. 13) presented Problem 8.9 and reported that the Davidon method solved the problem after 60 function and gradient calculations. The new algorithm solved this typical curve-fit problem in 12 function and gradient evaluations, which is a significant improvement.

Problem 8.10 is given by Osborn (Ref. 19) as an example of an exponential fitting problem. Because of the presence of the exponential terms, the objective function is not well approximated by a quadratic function. The sum of squares is small, although nonzero at the solution; consequently, the Gauss method should work reasonably well. In fact, there is no difference between the Gauss results presented in Table 2 and the modified version of the new algorithm. The ten function and gradient evaluations compare favorably with the results obtained by Osborn.

Problem 8.11, suggested by R. Howard, was also presented by Osborn. With $m = 65$ and $n = 11$, it is considerably larger than the other problems. Although the modified algorithm required two more function and gradient evaluations than required by Osborn, a comparison of the results indicates that a tighter convergence tolerance was used. In fact, because of the use of the U-matrix one would expect the modified algorithm to converge quadratically, whereas comparable behavior should not be exhibited by the Gauss method for this problem.

On the basis of the test results presented, it appears that the modified algorithm with $\epsilon = 10^{-1}$ is generally the most effective. This algorithm is as good or better than the Gauss method on all 11 problems. For the problems with zero residuals, i.e., the first five, the modified algorithm is generally competitive with the Gauss method. For the problems with nonzero residuals, the new algorithm is clearly superior to the Gauss method. Although it has not been proven, the numerical results tend to indicate that the modified algorithm is quadratically convergent, presumably because of the contribution of the U-matrix. In contrast, the Gauss method for nonlinear problems with nonzero residuals exhibits linear convergence. It also appears that the modified algorithm is superior to the basic algorithm.

7. Summary

This paper describes a new algorithm for solving the general nonlinear least square problem. The Hessian matrix of the objective function is represented as the sum of two matrices U and V. The V matrix is the first-order estimate originally suggested by Gauss, and the U matrix is generated recursively using a rank-one formula. The algorithm is effective on general problems that are characterized by nonzero residuals, primarily because the contribution of U is included in the total Hessian estimate. Although we do not prove quadratic convergence, numerical experience tends to confirm this assertion.

8. Appendix: Test Problems

The following problems are arranged approximately in order of increasing difficulty.

Problem 8.1. Minimize

$$J=\sum_{i=1}^{10}[(\exp(-x_1t_i)-\exp(-x_2t_i))-x_3(\exp(-t_i)-\exp(-10t_i))]^2,$$

where

$$t_i=0.1i.$$

Minimum value: $J^*=0$ at $x^*=(1, 10, 1)$.
Starting point: $x^0=(0, 10, 20)$.

Problem 8.2. Minimize

$$J=100(x_2-x_1^2)^2+(1-x_1)^2.$$

Minimum value: $J^*=0$ at $x^*=(1, 1)$.
Starting point: $x^0=(-1.2, 1.0)$.

Problem 8.3. Minimize

$$J=(x_1+10x_2)^2+5(x_3-x_4)^2+(x_2-2x_3)^4+10(x_1-x_4)^4.$$

Minimum value: $J^*=0$ at $x^*=(0, 0, 0, 0)$.
Starting point: $x^0=(3, -1, 0, 1)$.

Problem 8.4. Minimize

$$J=\sum_{i=1}^{3}[c_i-x_1(1-x_2^i)]^2,$$

where

$$c_1=1.5, \qquad c_2=2.25, \qquad c_3=2.625.$$

Minimum value: $J^*=0$, at $x^*=(3, 0.5)$.
Starting point: $x^0=(0.1, 0.1)$.

Problem 8.5. Minimize

$$J=[4(x_1+x_2)]^2+[4(x_1+x_2)+(x_1-x_2)((x_1-2)^2+x_2^2-1)]^2.$$

Minimum value: $J^*=0$ at $x^*=(0, 0)$.
Starting point: $x^0=(2, 0)$.

Problem 8.6. Minimize

$$J=c_1^2(x)+c_2^2(x),$$

where

$$c_1(x)=-13+x_1-2x_2+5x_2^2-x_2^3, \qquad c_2(x)=-29+x_1-14x_2+x_2^2+x_2^3.$$

Local minimum value: $J^*=48.98425$ at $x^*=(11.4128, -0.89681)$.
Global minimum value: $J^*=0$ at $x^*=(5, 4)$.
Starting point: $x^0=(15, -2)$.

Problems 8.7. Minimize

$$J=\sum_{i=1}^{15}\{y_i-[x_1+u_i/(x_2v_i+x_3w_i)]\}^2,$$

where the vectors y, u, v, w are given in Table 3.
Minimum value: $J^*=8.214877\times10^{-3}$ at $x^*=(0.08241, 1.1330, 2.3437)$.
Starting point: $x^0=(1, 1, 1)$.

Problem 8.8. Minimize

$$J=\sum_{i=1}^{10}[y_i-(\exp(ix_1)+\exp(ix_2))]^2,$$

where

$$y_i=2+2i.$$

Minimum value: $J^*=124.36$ at $x^*=(0.25783, 0.25783)$.
Starting point: $x^0=(0.3, 0.4)$.

Table 3. Data for Problem 8.7.

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

Problem 8.9. Minimize

$$J=\sum_{i=1}^{11}[y_i-x_1(u_i^2+x_2u_i)/(u_i^2+x_3u_i+x_4)]^2,$$

where the vectors y and u are given in Table 4.
Minimum value: $J^*=3.070561\times10^{-4}$ at $x^*=(0.19281, 0.19128, 0.12306, 0.13606)$.
Starting point: $x^0=(0.25, 0.39, 0.415, 0.39)$.

Problem 8.10. Minimize

$$J=\sum_{i=1}^{33}[y_i-(x_1+x_2\exp(-x_4t_i)+x_3\exp(-x_5t_i)]^2,$$

where $t_i=10(i-1)$ and the vector y is given in Table 5.
Minimum value: $J^*=0.546\times10^{-4}$ at $x^*=(0.3754, 1.9358, -1.4647, 0.01287, 0.02212)$.
Starting point: $x^0=(0.5, 1.5, -1, 0.01, 0.02)$.

Problem 8.11. Minimize

$$J=\sum_{i=1}^{65}c_i^2(x),$$

Table 4. Data for Problem 8.9.

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

Table 5. Data for Problem 8.10.

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

Table 6. Data for Problem 8.11.

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

where $t_i = 0.1(i-1)$, the vector y is given in Table 6, and

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

Minimum value: $J^* = 0.401377 \times 10^{-1}$ at $x^* = (1.30997,\ 0.431554,\ 0.633661,\ 0.599430,\ 0.754183,\ 0.904286,\ 1.36581,\ 4.82369,\ 2.39868,\ 4.56887,\ 5.67534)$.
Starting point: $x^0 = (1.3, 0.65, 0.65, 0.7, 0.6, 3.0, 5.0, 7.0, 2.0, 4.5, 5.5)$.

References

1. BETTS, J. T., *An Improved Penalty Function Method for Solving Constrained Parameter Optimization Problems*, Journal of Optimization Theory and Applications, Vol. 16, Nos. 1/2, 1975.
2. HANSON, R. J., and LAWSON, C. L., *Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems*, Mathematics of Computation, Vol. 23, No. 108, 1969.
3. BROYDEN, C. G., *The Convergence of Single-Rank Quasi-Newton Methods*, Mathematics of Computation, Vol. 24, No. 110, 1970.
4. MURTAGH, B. A., and SARGENT, R. W. H., *A Constrained Minimization Method with Quadratic Convergence*, Optimization, Edited by R. Fletcher, Academic Press, New York, New York, 1969.
5. LEVENBERG, K., *A Method for the Solution of Certain Nonlinear Problems in Least Squares*, Quarterly of Applied Mathematics, Vol. 2, No. 2, 1944.
6. MARQUARDT, D. W., *An Algorithm for Least Squares Estimation of Nonlinear Parameters*, SIAM Journal on Applied Mathematics, Vol. 2, pp. 431–441, 1963.
7. BETTS, J. T., *An Effective Method for Solving Constrained Parameter Optimization Problems*, The Aerospace Corporation, El Segundo, California, Report No. TR-0073(3450-10)-1, 1972.
8. BOX, M. J., *A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems*, Computer Journal, Vol. 9, pp. 67–77, 1966.
9. ROSENBROCK, H. H., *An Automatic Method for Finding the Greatest or Least Value of a Function*, Computer Journal, Vol. 3, pp. 175–184, 1960.
10. JONES, A. P., *SPIRAL—A New Algorithm for Nonlinear Parameter Estimation Using Least Squares*, Computer Journal, Vol. 13, No. 3, 1970.
11. POWELL, M. J. D., *An Iterative Method for Finding Stationary Values of a Function of Several Variables*, Computer Journal, Vol. 5, pp. 147–151, 1962.
12. BEALE, E. M. L., *On An Iterative Method for Finding a Local Minimum of a Function of More than One Variable*, Princeton University, Princeton, New Jersey, Statistical Techniques Research Group, Technical Report No. 25, 1958.
13. KOWALIK, J. S., and OSBORN, M. R., *Methods for Unconstrained Optimization Problems*, American Elsevier Publishing Company, New York, New York, 1968.
14. BRANIN, F. H., JR., *Widely Convergent Method for Finding Multiple Solutions of Simultaneous Nonlinear Equations*, IBM Journal of Research and Development, Vol. 16, pp. 504–522, 1971.
15. FREUDENSTEIN, F., and ROTH, B., *Numerical Solutions of Systems of Nonlinear Equations*, Journal of the ACM, Vol. 10, pp. 550–556, 1963.
16. FLETCHER, R., *Generalized Inverse Methods for the Best Least Squares Solution of Systems of Nonlinear Equations*, Computer Journal, Vol. 10, pp. 392–399, 1968.
17. BARD, Y., *Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems*, SIAM Journal on Numerical Analysis, Vol. 7, No. 1, 1970.

18. JENNRICH, R. I., and SAMPSON, P. F., *Application of Stepwise Regression to Nonlinear Estimation*, Technometrics, Vol. 10, No. 1, 1968.
19. OSBORN, M. R., *Some Aspects of Nonlinear Least Squares Calculations*, Numerical Methods for Nonlinear Optimization, Edited by F. A. Lootsma, Academic Press, New York, New York, 1972.