MATHEMATICS OF COMPUTATION, VOLUME 26, NUMBER 117, JANUARY, 1972

A Rank Two Algorithm for Unconstrained Minimization

By Ronald Bass

Abstract. A stable second-order unconstrained minimization algorithm with quadratic termination is given. The algorithm does not require any one-dimensional minimizations. Computational results presented indicate that the performance of this algorithm compares favorably with other well-known unconstrained minimization algorithms.

Introduction. Algorithms for unconstrained minimization have enjoyed a great deal of attention in recent years. The fundamental philosophy behind most of these algorithms is the exploitation of the locally quadratic nature of a well-behaved function at an unconstrained local minimum. The Davidon-Fletcher-Powell method [1] was highly successful in this regard and guarantees finite convergence when the function is quadratic, monotone decrease of function value when it is not. However, a potentially time consuming one-dimensional minimization is required in each iteration. Recently developed rank one [2] and rank two [3] methods eliminate the need for this minimization. The rank one method, however, sacrifices finite convergence for stability, and the rank two method eliminates the one-dimensional minimization by trading finite convergence for monotone convergence. The amount of computation in each iteration is reduced, but the number of iterations required may be increased.

In this paper, we present a rank two method which has the combined virtues of finite convergence for quadratic functions and stability for any function, and does not require a one-dimensional search at each iteration. This combination of desirable properties is achieved by making full use of the flexibility of a rank two algorithm.

The algorithm given here is cyclic, i.e., it repeats itself every N iterations (when minimizing a function of N variables) unlike the Davidon-Fletcher-Powell algorithm, which is the same in every iteration. Kelley and Myers [8] presented a cyclic method which is a special case of the algorithm given here.

Statement of the Problem. We are interested in minimizing a scalar function $f(x)$, x an N-vector. Let x^* be the value of x that minimizes f. Assume that f is locally quadratic about x^*, i.e., for x near x^*,

$$f(x) = \tfrac{1}{2}x'Gx + b'x + e, \tag{1}$$

where G is a positive definite symmetric matrix (p.d.s.m.). A necessary and sufficient condition that x^* minimize this quadratic is that

Received December 21, 1970, revised July 27, 1971.

AMS 1969 *subject classifications.* Primary 9058.

Key words and phrases. Optimization, unconstrained optimization, unconstrained minimization, conjugate gradient, rank two, nonlinear equation, quasi-newton, ascent, steepest ascent.

$$g(x^*) \triangleq \text{grad } f(x^*) = 0 = Gx^* + b$$

or

(2) $$x^* = -G^{-1}b.$$

Given any x,

(3) $$x^* - x = -G^{-1}b - x = -G^{-1}(b + Gx) = -G^{-1}g(x).$$

Thus, if f is locally quadratic about x^* and x is near x^*,

$$s = -G^{-1}g(x), \qquad x^* = x + s,$$

i.e., s is a good direction in which to search for a minimum of f. Further, if x is far from x^*, and G is the matrix of second derivatives of f at x, s is the best direction, based on a local quadratic approximation, in which to search for a decrease in f.

It is therefore desirable to have an efficient method for obtaining a good estimate H of the inverse second derivative matrix G^{-1} at x.

N-Term Rank One Decompositions of p.d.s.m. Let A be an $N \times N$ p.d.s.m. Let $d_1, \cdots, d_N$ be A-orthogonal, i.e.,

(4) $$\begin{aligned} d_i' A d_j &= 0 \quad \text{if } i \neq j, \\ &= r_i \quad \text{if } i = j, r_i > 0; \end{aligned}$$

then it is readily verified that

(5a) $$A^{-1} = \sum_{i=1}^{N} \frac{d_i d_i'}{d_i' A d_i}.$$

Conversely, if for some (linearly independent) set of vectors, $d_1, \cdots, d_N$.

(5b) $$A^{-1} = \sum_{i=1}^{N} \frac{1}{r_i} d_i d_i',$$

then, those vectors are A-orthogonal and satisfy Eq. (4). Thus, all N-term symmetric rank one decompositions of a p.d.s.m. A^{-1} are exactly those decompositions given by all sets of A^{-1}-orthogonal vectors.

Constructing a Set of G-Orthogonal Vectors From a Set of Linearly Independent Vectors. In the following two lemmas, algorithms are given for constructing A^{-1}-orthogonal and A-orthogonal vectors for p.d.s.m. A from a set of linearly independent vectors.

LEMMA 1. *Let A be an $N \times N$ p.d.s.m. Let $d_1, \cdots, d_N$ be a set of nonzero vectors. Let $A_0 = 0$, and, for $k = 1, \cdots, N$, let*

$$\begin{aligned} c_k &= d_k - A_{k-1} A d_k, \\ A_k &= A_{k-1} + c_k c_k' / c_k' A c_k \quad \text{if } c_k' A c_k \neq 0, \\ &= A_{k-1} \quad \text{if } c_k' A c_k = 0. \end{aligned}$$

Then:

(1) *There exist $\{a_{ij}\}$, $\{b_{ij}\}$ such that*

(a) $$c_j = d_j + \sum_{i=1}^{j-1} a_{ij} d_i,$$

(b) $$d_j = \sum_{i=1}^{j} b_{ij} c_i.$$

(2) *If and only if* $d_1, \cdots, d_n$, $n \leqq N$, *are linearly independent, then* $c_1, \cdots, c_n$ *are* A*-orthogonal, and therefore if* $d_1, \cdots, d_N$ *are linearly independent,* $A_N = A^{-1}$.

(3) *If* $d_1, \cdots, d_{j-1}$ *are linearly independent and* d_j *is linearly dependent on* $d_1, \cdots, d_{j-1}$, *then* $c_j = 0$.

(4) $A_k A A_k = A_k$.

Proof. (1a) We have, by definition, $c_1 = d_1$. Assume

$$c_j = \sum_{i=1}^{j} a_{ij} d_i, \qquad j = 1, \cdots, k-1, a_{jj} = 1,$$

where some of the c_j may be zero.

Then, by definition of c_k and A_k,

$$c_k = d_k - A_{k-1} A d_k = d_k - \sum_{j=1;c_j \neq 0}^{k-1} c_j \frac{c_j' A d_k}{c_j' A c_j}$$
$$= d_k - \sum_{j=1;c_j \neq 0}^{k-1} \sum_{i=1}^{j} a_{ij}\, d_i \frac{c_j' A d_k}{c_j' A c_j}.$$

(b) follows immediately from (a).

(2) Assume $d_1, \cdots, d_n$ are linearly independent. We will show that $c_1, \cdots, c_n$ are A-orthogonal. From (1a) of Lemma 1, $c_1, \cdots, c_n$ are nonzero, and

$$c_1' A c_2 = d_1' A I - \frac{d_1 d_1' A}{d_1' A d_1} d_2 = 0.$$

Let the inductive hypothesis be that c_i, $i \leqq k < n$ are A-orthogonal. Then, for $i \leqq k$,

$$c_i' A c_{k+1} = c_i' A(I - A_k A)\, d_{k+1}$$
$$= \left(c_i' A - c_i' A \sum_{j=1}^{k} \frac{c_j c_j' A}{c_j' A c_j}\right) d_{k+1} = 0$$

and $c_1, \cdots, c_{k+1}$ are A-orthogonal.

Conversely, suppose $d_1, \cdots, d_n$ are linearly dependent. We will show that $c_1, \cdots, c_n$ are not A-orthogonal. From (1a) of Lemma 1, $c_1, \cdots, c_n$ are also linearly dependent, and for some $j \leqq n$,

$$c_j = \sum_{k \neq j} a_k c_k, \qquad a_l \neq 0 \text{ for some } l \neq j.$$

Then, if $c_1, \cdots, c_n$ are A-orthogonal,

$$0 = c_j' A c_l = a_l c_l' A c_l \neq 0, \quad \text{a contradiction.}$$

(3) Let $d_j = \sum_{i=1}^{j-1} e_i\, d_i$.

From (1a) of Lemma 1, we have

$$c_j = d_j + \sum_{i=1}^{j-1} a_{ij} d_i = \sum_{i=1}^{j-1} (a_{ij} + e_i) \sum_{l=1}^{i} b_{li} c_l = \sum_{i=1}^{j-1} h_i c_i$$

and from (1b) of Lemma 1,

$$d_j = \sum_{i=1}^{j} b_{ij}c_i = b_{jj} \sum_{i=1}^{j-1} h_ic_i + \sum_{i=1}^{j-1} b_{ij}c_i = \sum_{i=1}^{j-1} m_ic_i.$$

Then, by (2) of Lemma 1,

$$c_j = d_j - A_{j-1}Ad_j = \sum_{i=1}^{j-1} m_ic_i - \sum_{i=1}^{j-1} \frac{c_ic_i'}{c_i'Ac_i} A \sum_{i=1}^{j-1} m_ic_i = 0.$$

(4) Since, by (2) of Lemma 1, $c_1, \cdots, c_k$ are A-orthogonal or zero,

$$A_kAA_k = \sum_{i=1;c_i\neq 0}^{k} \sum_{j=1;c_j\neq 0}^{k} \frac{c_ic_i'}{c_i'Ac_i} A \frac{c_jc_j'}{c_j'Ac_j} = A_k.$$

LEMMA 2. *Let A be an $N \times N$ p.d.s.m. Let $d_1, \cdots, d_N$ be a set of nonzero vectors. Let $A_1 = A$ and, for $k = 1, \cdots, N$, let*

$$\begin{aligned} c_k &= A_kd_k, \\ A_{k+1} &= A_k - c_kc_k'/c_k'd_k \qquad \text{if } c_k'd_k \neq 0, \\ &= A_k \qquad\qquad\qquad\quad \text{if } c_k'd_k = 0. \end{aligned}$$

Then:

(1) *Let $z'Ad_i = 0$, $i = 1, \cdots, k-1$. Then $Az = A_kz$.*

(2) *A_k is positive semidefinite and $A_kd_j = 0$, $j < k$.*

(3) *Let j be the number of linearly independent vectors in $d_1, \cdots, d_{k-1}$. Then* $\text{rank}(A_k) = N - j$.

Further, if and only if $d_1, \cdots, d_N$ are linearly independent,

(4) *$A_{N+1} = 0$, i.e. $A = \sum_{i=1}^{N} c_ic_i'/c_i'd_i$.*

(5) *$c_1, \cdots, c_N$ are A^{-1}-orthogonal.*

Proof. (1) Choose $z \neq 0$ such that $z'Ad_1 = 0$. Then,

$$A_2z = Az - \frac{Ad_1d_1'Az}{d_1'Ad_1} = Az.$$

Let the inductive hypothesis be that $A_kz = Az$ for all z such that $z'Ad_j = 0, j = 1, \cdots, k-1$. Then, certainly, $A_kz = Az$ for all z such that $z'Ad_j = 0$, $j = 1, \cdots, k$, and, for all such z,

$$\begin{aligned} A_{k+1}z &= A_kz - \frac{A_kd_kd_k'A_k}{d_k'A_kd_k} z = A_kz = Az, \qquad \text{if } c_k'd_k \neq 0, \\ &= A_kz = Az, \qquad \text{if } c_k'd_k = 0. \end{aligned}$$

(2) By definition, A_1 is positive definite. For $d_1 \neq 0$,

$$A_2d_1 = \left(A - \frac{Ad_1d_1'A}{d_1'Ad_1}\right) d_1 = 0.$$

Let the inductive hypothesis be that A_k is positive semidefinite and $A_kd_j = 0$, $j < k$. We will show that A_{k+1} is positive semidefinite and $A_{k+1}d_j = 0$, $j < k+1$.

Clearly, any x may be written

$$x = z + \sum_{i=1}^{k} a_id_i, \qquad z'Ad_i = 0, \qquad i \leqq k.$$

Then, $z'A = z'A_k$ by (1) of Lemma 2, so that $z'A_k d_i = 0$, $i \leqq k$, and since by hypothesis $A_k d_j = 0$, $j < k$, it is easily seen that

$$x'A_{k+1}x = z'A_k z \geqq 0.$$

Thus, A_{k+1} is positive semidefinite. By definition, $A_{k+1}d_j = 0$, $j < k$. If d_k is linearly dependent on $d_1, \cdots, d_{k-1}$, then $A_k d_k = 0$ and $A_{k+1} = A_k$. Otherwise, $d_k' A_k d_k > 0$, since A_k is positive semidefinite and, by definition, rank(A_k) is at least $N - (k - 1)$. Then,

$$A_{k+1}d_k = A_k d_k - A_k d_k \frac{d_k' A_k d_k}{d_k' A_k d_k} = 0.$$

This completes the induction.

(3) By definition, rank(A_k) $\geqq N - j$, and by (2) of Lemma 2, rank(A_k) $\leqq N - j$.

(4) By (3) of Lemma 2, rank $A_{N+1} = 0$ if and only if $d_1, \cdots, d_N$ are linearly independent.

(5) We need only note that $c_k' d_k = d_k' A_k d_k > 0$; then, by (4) of Lemma 2 and Eq. (5b), $c_k' d_k = c_k' A^{-1} c_k$ and $c_1, \cdots, c_N$ are A^{-1}-orthogonal.

Let f be given by Eq. (1). Now, suppose we have an algorithm for minimizing f such that in the kth iteration we take a step $d_k = x_{k+1} - x_k$, resulting in a gradient change $y_k = g_{k+1} - g_k$.

We show in Lemma 3 how to construct N G-orthogonal vectors from any N steps (such that $d_1, \cdots, d_N$ are linearly independent) without using G or G^{-1} explicitly.

LEMMA 3. *Let $d_1, \cdots, d_N$ be linearly independent. Let $H_0 = 0$. Let $H_k = H_{k-1} + s_k s_k' / s_k' y_k$, where $s_k = d_k - H_{k-1} y_k$. Then, $H_N = G^{-1}$.*

Proof. We will show that

$$s_k' G s_k = d_k' y_k - y_k' H_{k-1} y_k = s_k' y_k.$$

Then, it follows from Lemma 1 and the fact that

$$s_k = d_k - H_{k-1} y_k = d_k - H_{k-1} G d_k$$

that $s_1, \cdots, s_N$ are G-orthogonal and $G^{-1} = H_N$. Clearly,

$$s_1 = d_1 \quad \text{and} \quad s_1' y_1 = d_1' y_1 = d_1' G d_1 = s_1' G s_1.$$

Then, by Lemma 1, $H_1 G H_1 = H_1$. Let the inductive hypothesis be that

$$H_{k-1} G H_{k-1} = H_{k-1}.$$

Then, since $G d_k = y_k$, we have

$$\begin{aligned} s_k' G s_k &= (d_k - H_{k-1} y_k)' G (d_k - H_{k-1} y_k) \\ &= d_k' G d_k - y_k' H_{k-1} G d_k - d_k' G H_{k-1} y_k + y_k' H_{k-1} G H_{k-1} y_k \\ &= d_k' y_k - y_k' H_{k-1} y_k = s_k' y_k, \end{aligned}$$

and, by (4) of Lemma 1,

$$H_k G H_k = H_k.$$

Note that the formula for updating H_k in Lemma 3 is the same formula used in the rank one algorithm (Eq. (6)). However, in Lemma 3, $H_0 = 0$, whereas in the rank one algorithm H_0 is arbitrary.

It should be noted that Lemmas 1 and 2 are simply applications of the Gram-Schmidt orthogonalization procedure with respect to the inner product defined by A or A^{-1}. The matrices A_k in these lemmas are A-orthogonal and A^{-1}-orthogonal projection operators. Lemma 3 is an application of Lemma 1.

Desirable Properties for H_k. Suppose f is given by Eq. (1), and let $d_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$, $G^{-1} = H$. Then,

$$Hy_k = G^{-1}(Gx_{k+1} + b - Gx_k - b) = d_k.$$

We can now give two definitions of a "good" estimate H_k of G^{-1}.

First, we could require that $H_{k+1} - H_k$ be of rank one and that

$$H_{k+1}y_k = d_k,$$

in which case

$$H_{k+1} = H_k + \frac{(d_k - H_k y_k)(d_k - H_k y_k)'}{(d_k - H_k y_k)' y_k} \tag{6}$$

is the only possible such formula for updating H_k [2]. Alternately, we can require that

$$H_k = \sum_{i=1}^{k} \frac{c_i c_i'}{c_i' G c_i} + \sum_{i=k+1}^{N} z_i z_i',$$

where $(c_1, \cdots, c_k, z_{k+1}, \cdots, z_N)$ is a set of linearly independent vectors and $c_1, \cdots, c_k$ are G-orthogonal vectors constructed from the first k steps as described in Lemma 3. This criterion motivates the algorithm given below.

We have shown that a good search direction, $-H_k g_k$, can be obtained if H_k is a "good" estimate of the inverse second derivative matrix of f (we will use the second definition given above, i.e., Lemma 3). Thus, H_k should be updated so that if f is quadratic, $H_N = G^{-1}$ (quadratic termination property). This property ensures rapid convergence after reaching a point x near x^* if f is quadratic locally about x^*. Further, it is desirable that H_k be positive definite. Then, no matter how poor an estimate is H_k of the inverse second derivative matrix, the "best" search direction, $-H_k g_k$, is a locally downhill direction, i.e., there is some positive t such that $f(x_k + t(-H_k g_k)) < f(x_k)$ (stability property). In the algorithm given below, H_k has both of these properties.

Algorithm for Unconstrained Minimization. We now give an outline of an unconstrained minimization algorithm and show in detail how H_k is updated. Choice of step direction and step size are discussed later.

Step (0) x_0 = initial guess of x^*,
H_0 = initial guess of inverse second derivative matrix at x_0,
$B_0 = H_0$,
$A_0 = 0$,
$g_0 = \text{grad } f(x_0)$,
$k = 1$.

Step (1) (choose a step d_k),
$x_k = x_{k-1} + d_k$,
(compute $f(x_k)$, g_k and test for convergence),

$y_k = g_k - g_{k-1}$,
$s_k = d_k - A_{k-1}y_k$,
$a_k = s_k'y_k$,
if $a_k \leqq 0$ go to Step (3) (see discussion below).

Step (2) $A_k = A_{k-1} + s_ks_k'/a_k$
$B_k = B_{k-1} - B_{k-1}s_ks_k'B_{k-1}/s_k'B_{k-1}s_k$,
$H_k = A_k + B_k$,
increase k by 1,
if $k \leqq N$ go to Step (1),
if $k > N$ go to Step (3).

Step (3) set $H_0 = B_0 = H_k$,
$A_0 = 0$,
$x_0 = x_k$,
$k = 1$,
go to Step (1).

Properties of the Algorithm. We now exhibit some important properties of the above algorithm.

Let C_1 be the condition: $d_1, \cdots, d_N$ are linearly independent.

Property (1) Assume that $a_1, \cdots, a_N$ are positive. Then, $s_1, \cdots, s_N$ are linearly independent if and only if C_1 holds.

Proof. It is sufficient to show that s_k is a linear combination of $d_1, \cdots, d_k$, where the coefficient of d_k is not zero. Clearly, $s_1 = d_1$. Let the inductive hypothesis be that

$$s_j = \sum_{i=1}^{j} a_{ij}d_i, \qquad j = 1, \cdots, k-1.$$

Then

$$s_k = d_k - A_{k-1}y_k = d_k - \sum_{j=1}^{k-1} s_j \frac{s_j'y_k}{a_j}$$
$$= d_k - \sum_{j=1}^{k-1} \sum_{i=1}^{j} a_{ij}\, d_i \frac{s_j'y_k}{a_j}.$$

Property (2) $B_N = 0$ if and only if C_1 holds (by (4) of Lemma 2 and Property (1)).

Property (3) Let f be quadratic, f given by Eq. (1). Then

(a) $a_j > 0$ if and only if C_1 holds.

Proof. (a) If f is quadratic, then, from the proof of Lemma 3, $a_j = s_j'Gs_j$, and a_j is zero if and only if s_j is zero, and is positive otherwise. But by (1) and (3) of Lemma 1 and by Property (1), s_j is zero if and only if C_1 does not hold. (b) $H_N = G^{-1}$ if and only if C_1 holds (by Lemma 3 and Property (2)).

Property (4) Let $a_k > 0$ for all k. Then, H_k is positive definite for all k if and only if C_1 holds.

Proof. Assume C_1 holds. Let $w_1, \cdots, w_k$ satisfy

$$w_i = \sum_{j=1}^{k} c_{ji}s_j, \qquad w_i's_j = 1, \qquad i = j,$$
$$= 0, \qquad i \neq j.$$

Then, any vector x can be written as $x = v + y$, where

$$y = \sum_{i=1}^{k} b_i w_i, \qquad v's_i = 0, \qquad i = 1, \cdots, k.$$

From Lemma 2, $B_k s_i = 0$, $i = 1, \cdots, k$, and B_k is of rank $N - k$ and is positive semidefinite. If y and v are not zero, then

$$y'B_k y = 0, \qquad v'B_k v > 0,$$

$$y'A_k y > 0, \qquad v'A_k v = 0.$$

For $x \neq 0$, y and v are not both zero, and

$$x'H_k x = y'A_k y + v'B_k v > 0.$$

Conversely, if C_1 does not hold, then, by Property (1), for some k, s_k is a linear combination of $s_1, \cdots, s_{k-1}$, and from (2) of Lemma 2, $B_{k-1}s_k = 0$ and B_k is undefined.

Property (5) H_k satisfies the second criterion for a good estimate of the inverse second derivative matrix, since A_k is in fact constructed as indicated in Lemma 3. As with the other properties, this condition is contingent upon C_1 being satisfied.

Choice of Step Direction. The natural choice of step direction, as discussed in the introduction, is

$$d_k/||d_k|| = -H_k g_k/||H_k g_k||,$$

where $||x|| = (x'x)^{1/2}$. However, it is clear from Properties (1)–(5) that if the algorithm is to be well behaved, C_1 must be satisfied. Suppose C_1 is violated by $-H_k g_k$. Then, a satisfactory direction is

$$\frac{d_k}{||d_k||} = -\frac{H_k g_k}{||H_k g_k||}(1 - \alpha_k^2)^{1/2} \pm \alpha_k e_k,$$

where e_k is any unit vector perpendicular to $d_1, \cdots, d_{k-1}$ and α_k is some prespecified constant, $0 < \alpha_k < 1$ so that $d_k/||d_k||$ makes an angle of $\tan^{-1}(1 - \alpha_k^2)^{1/2}/\alpha_k$ with the manifold generated by $d_1, \cdots, d_{k-1}$.

The choice of sign is such that $-g_k'd_k$ is maximized, i.e., the modified d_k is in the most downhill direction. In practice, an e_k component would be added if $-H_k g_k$ almost violated C_1, and we would then choose

$$\frac{d_k}{||d_k||} = -\frac{H_k g_k - e_k'H_k g_k e_k}{||H_k g_k - e_k'H_k g_k e_k||}(1 - \alpha_k^2)^{1/2} \pm \alpha_k e_k,$$

where the sign is again chosen to maximize $-g_k'd_k$ (see Appendix I for an efficient way to compute e_k and check for linear independence).

The choice of $-H_k g_k$ as a search direction is predicated upon the assumption that H_k is in some sense a fair approximation of the inverse second derivative matrix. This assumption is not necessarily valid for the first few iterations, and a more rapid initial reduction of the cost function might be obtained by taking these steps in the $-g_k$ direction.

Choice of Step Size. In practical application of the Davidon-Fletcher-Powell algorithm, it is not possible to find the exact minimum along the search direction, and any method which is used to get an accurate approximation to that minimum, such as a Fibonacci search, is very time consuming. Consequently, some method which gives an approximation to the minimum is usually used, and it is hoped that the inaccuracy will not materially upset the performance of the algorithm. In the algorithm given here, any step size which decreases f is acceptable, and will yield convergence of H_k to G^{-1} in N steps if f is quadratic. However, use of a cubic interpolation scheme to achieve approximately the minimum along the search direction, such as that given by Fletcher and Powell will guarantee that if $H_k = G^{-1}$, the minimum will be achieved on the kth step, and will also ensure that g_{k+1} is approximately perpendicular to d_k, which in practice is often sufficient to guarantee that C_1 is satisfied.

A somewhat more satisfactory method is the following: take a step $d_k = -H_k g_k$; if $f(x_k + d_k) < f(x_k)$, let $x_{k+1} = x_k + d_k$; otherwise, try $x_{k+1} = x_k + d_k/h^m$, $m = 1, 2, \cdots$, until a decrease in f is obtained, where, for example, $h = 10$ may be used so that m will remain small. This procedure guarantees that if $H_k = G^{-1}$, the minimum will be achieved on the kth step, and keeps the number of function evaluation per iteration small by using a relatively large h. Of course, if enough function values have been computed ($m = 2$), a cubic interpolation can be used rather than simply using a larger m. This combined approach seems to be quite effective in practice (see test problems).

Case when $a_k \leqq 0$. Let $G(x_k)$ denote the second derivative matrix at x_k. We have shown (Property (3)) that if f is quadratic, $a_k > 0$. If $a_k \leqq 0$, then, either H_k is not a good estimate of $G^{-1}(x_k)$ or the step size is so large that a quadratic approximation to f with metric $G(x_k)$ is not a good representation of f on the set $\{x_k + \beta(x_{k+1} - x_k), 0 \leqq \beta \leqq 1\}$. In either case, the curvature information in H_k is no longer very accurate. Thus, it is desirable to deemphasize the information in H_k by treating H_k and x_k as an initial guess and starting again, rather than assuming that H_k is composed of G-orthogonal vectors constructed from the last k steps. Therefore, if $a_k \leqq 0$ (or $a_k \leqq \epsilon$, $\epsilon > 0$), we go to Step (3). Then, for the kth step

$$s_k = s_1 = d_k,$$
$$a_k = a_1 = d_k' y_k = d_k' G d_k \quad \text{if } f \text{ is quadratic.}$$

If a_k is still negative, then f is probably not well approximated by a quadratic on $\{x_k + \beta d_k, 0 \leqq \beta \leqq 1\}$; again we let $x_{k+1} = x_k + d_k$ and return to Step (3) (without updating A_0 or B_0).

The effect of this procedure is to remove old information from A_k while retaining that information in B_k. The search direction, $-H_k g_k$, is affected by the old information, but s_k depends only on new information in A_k.

Test Problems. The Davidon-Fletcher-Powell algorithm is apparently the most successful unconstrained minimization algorithm to date, so the examples presented here are taken from Fletcher and Powell's paper [1] and compared with their results. The currently accepted basis for comparison of unconstrained minimization algorithms is the number of objective function evaluations required for convergence, since in most practical problems these consume the bulk of the computing time. Calculation of the gradient is counted as N function evaluations (N the number of variables), since evaluation of each of the N components of the gradient analytically is roughly equivalent to evaluation of the objective function, and evaluation of the gradient by perturbation requires at least N function evaluations at perturbed values of the variables.

Since Fletcher and Powell's algorithm, as applied to their test problems [1], requires at least two function and gradient evaluations per iteration (one for the main algorithm and one for the cubic interpolation), we shall ascribe to their examples a minimum of $2N + 2$ function evaluations per iteration.

The calculations for the algorithm given here were carried out on a Univac 1108 computer in single-precision arithmetic (8 significant figures). In all examples, the step direction and step size schemes given above were used, with $h = 10$ and $\alpha_k = .1$ and were found to give good results (convergence rates were not very sensitive to changes in these parameters). The first test problem is (Table 1) the parabolic valley

TABLE 1
Parabolic Valley

	Our Method		Fletcher and Powell	
Iteration	f	*Number of function evaluations*	f	*Minimum number of function evaluations*
0	24.2	1	24.2	1
3	4.18	12	3.69	18
6	3.67	21	1.605	36
9	3.62	30	.745	54
12	3.37	40	.196	72
15	2.14	50	.012	90
18	1.78	60	1×10^{-8}	108
25	.631	82		
30	.344	100		
35	.260	116		
40	.116	133		
45	.066	151		
50	.025	171		
55	.015	191		
58	.0012	203		
60	2.7×10^{-4}	210		
62	3.5×10^{-5}	217		
64	1.4×10^{-7}	224		
65	4.4×10^{-8}	228		
66	4.6×10^{-12}	231		

originally given by Rosenbrock [5],

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

with starting point $(-1.2, 1)$ and a zero at $(1, 1)$. The second problem (Table 2) is a steep-sided helical valley,

$$f(x_1, x_2, x_3) = 100[(x_3 - 10A)^2 + (R - 1)^2] + x_3^2,$$

where

$$2\pi A = \arc\tan (x_2/x_1) \qquad \text{if } x_1 > 0,$$
$$= \arc\tan (x_2/x_1) + \pi \quad \text{if } x_1 < 0,$$

and

$$R = (x_1^2 + x_2^2)^{1/2}.$$

The distance x_3 along the axis of the helix is restricted so that $-2.5 < x_3 < 7.5$. The starting point is $(-1, 0, 0)$ and the function has a zero minimum at $(1, 0, 0)$.

TABLE 2
Helical Valley

	Our Method		Fletcher and Powell	
Iteration	*f*	*Number of function evaluations*	*f*	*Minimum number of function evaluations*
0	2500	1	2500	1
1	2139	4	520	8
2	13.34	8	110	16
3	6.99	13	74.1	24
4	6.84	17	24.2	32
5	.46	21	10.9	40
6	.14	25	9.8	48
7	.065	29	6.3	56
8	.049	33	6.09	64
9	.047	37	1.89	72
10	.047	41	1.75	80
11	.040	46	.76	88
12	.0089	50	.38	96
13	.0063	54	.14	104
14	.0045	59	.058	112
15	5.6×10^{-4}	63	.018	120
16	1.4×10^{-4}	67	8×10^{-4}	128
17	1.1×10^{-4}	72	3×10^{-6}	136
18	5.0×10^{-6}	76	7×10^{-8}	144
19	3.6×10^{-6}	81		
20	2.5×10^{-6}	86		
21	3.7×10^{-9}	90		

Finally, we give some examples (Table 3) of solutions of functions of many variables. The function used is

$$f(x) = \sum_{i=1}^{N} x_i^2 + \left(\sum_{i=1}^{N} i^{1/2} x_i\right)^2 + \left(\sum_{i=1}^{N} i^{1/2} x_i\right)^4$$

which has a zero minimum at $x_i = 0$. Starting values were $x_i = .1$.

TABLE 3
Function of Many Variables

N = 10			N = 20		
Iteration	*Number of function evaluations*	*Function value*	*Iteration*	*Number of function evaluations*	*Function value*
0	1	30.6	0	1	1484
3	37	.154	3	69	39.1
6	70	.0035	6	132	1.86
9	103	.098	10	176	.47
12	125	.00019	20	386	.010
15	158	6×10^{-7}	30	596	.0017
18	192	6×10^{-8}	40	806	.0016
20	205	4×10^{-10}	50	1017	6.6×10^{-4}
			60	1227	3.8×10^{-4}
			70	1437	2.2×10^{-4}
			80	1649	1.7×10^{-4}
			90	1859	7.2×10^{-5}
			100	2069	6.2×10^{-5}
			110	2279	4.9×10^{-5}
			120	2490	1.3×10^{-5}
			125	2600	1.4×10^{-6}
			126	2621	3.1×10^{-7}
			127	2642	8.7×10^{-10}

Conclusions. The above test problems indicate that our results compare favorably with those given by Fletcher and Powell in [1], and that our algorithm is applicable to problems of moderate size. The advantages of this algorithm are that stable finite convergence is obtained in the case of a quadratic objective function without the need for a line search, and there is almost complete freedom in the choice of step direction and step size (the choices used here are certainly not definitive). The main disadvantage of this algorithm seems to be that the need to separately store the A_k and B_k matrices and to check for linear independence (see Appendix I) requires storage of two more $N \times N$ symmetric matrices than in Fletcher and Powell's algorithm.

It should be noted that in the Davidon-Fletcher-Powell type of algorithm, completely analogous A_k and B_k matrices are constructed [6]; however, the step directions are chosen to be G-orthogonal vectors, and the step direction and size is completely determined by this consideration. Such restrictions are avoided in our algorithm by computing G-orthogonal vectors from an almost arbitrary step, rather than requiring the step itself to be G-orthogonal.

Finally, we note that minimization of a positive semidefinite quadratic function is treated in Appendix II. We show that with almost no modification, all the above results hold if A^{-1} and G^{-1} are replaced by A^* and G^* (pseudo-inverse of A and G) and that the above algorithm may be applied with only slight modification, and therefore may be used to minimize functions that have singular second derivative matrices at a minimum.

Appendix I (Gram-Schmidt Orthogonalization Procedure). Let $d_1, \cdots, d_k$ be step directions. Let

$$\begin{aligned} P_0 &= 0, \qquad c_k = d_k - P_{k-1}\, d_k, \\ P_k &= P_{k-1} + c_k c_k'/c_k' c_k \quad \text{if } c_k \neq 0, \\ &= P_{k-1} \qquad\qquad\qquad \text{if } c_k = 0. \end{aligned}$$

Then $P_{k-1}d_k$ is the projection of d_k on the manifold generated by $d_1, \cdots, d_{k-1}$. The angle between this manifold and d_k is $\cos^{-1}(||P_{k-1}d_k||/||d_k||)$. Thus, d_k is "almost" linearly dependent on $d_1, \cdots, d_{k-1}$, if $|1 - ||P_{k-1}d_k||/||d_k|||$ is "almost" zero.

Clearly, the columns of M, $M = I - P_{k-1}$, where I is the identity, are orthogonal to $d_1, \cdots, d_{k-1}$. Let m_j be the jth column of M, j chosen so that $m_j \neq 0$. Let u_j be a vector with the jth component 1 and all other components zero. Let p_j be the jth column of P_{k-1}. Then a unit vector e_k orthogonal to $d_1, \cdots, d_k$ is

$$e_k = \frac{m_j}{||m_j||} = \frac{u_j - P_j}{||u_j - p_j||}.$$

Appendix II (Minimization of a Positive Semidefinite Quadratic Function). Let

$$f(x) = \tfrac{1}{2}x'Ax + b'x + e, \tag{II.1}$$

where A is a positive semidefinite symmetric matrix (p.s.s.m.). The corollary to Lemma 1 given below may be used to modify the algorithm to include the case where A is a p.s.s.m.

Notation.

N_A = null space of A,

P_A = $N_A^{\perp}$ (orthogonal complement of N_A),

R_A = range of A.

Definition. A^* is the pseudo-inverse of A if A^* satisfies

$$\begin{aligned} AA^*x &= x, \qquad x \text{ in } P_A, \\ A^*x &= 0, \qquad x \text{ in } R_A^{\perp}, \end{aligned}$$

(for definition and properties of A^*, see [7, Appendix C]). We note that $(A^*)^* = A$, and $AA^*A = A$. Further, we have from the symmetry of A that $R_A^{\perp} = N_A$ and $R_A = P_A$.

COROLLARY 1 (TO LEMMA 1). *For $k < n$, suppose there is no d_{k+1} such that $Ac_{k+1} \neq 0$. Then for any x, y such that $y = Ax$,*

$$A(A_k y - x) = 0.$$

Proof. First, we prove that Ac_i, $i \leqq k$, span R_A. If this is not true, then, for some $z \neq 0$ in R_A, $z'Ac_i = 0$, $i \leqq k$, and (since $R_A = P_A$) $Az \neq 0$; then, for $d_{k+1} = z$,

$$A_k A d_{k+1} = \sum_{i=1}^{k} \frac{c_i c_i'}{c_i' A c_i} A d_{k+1} = 0$$

and

$$c_{k+1} = d_{k+1} - A_k A d_{k+1} = d_{k+1} = z.$$

Hence, we have found a d_{k+1} such that

$$Ac_{k+1} = Az \neq 0,$$

a contradiction to the main hypothesis. Thus, Ac_i, $i \leqq k$, span R_A. Hence, for any x, $Ax = \sum_{j=1}^{k} a_j Ac_j$ so that for $y = Ax$, we have

$$\begin{aligned} A(A_k y - x) &= AA_k Ax - Ax \\ &= A\left[\sum_{i=1}^{k} \frac{c_i c_i'}{c_i' Ac_i} \sum_{j=1}^{k} a_j Ac_j\right] - \sum_{j=1}^{k} a_j Ac_j = 0. \end{aligned}$$

The algorithm is modified as follows. Suppose f is given by (II.1), A is a p.s.s.m., and there is no step d_k such that in Step (1) of the algorithm, $a_k \neq 0$. Since $a_k = s_k' y_k = s_k' A s_k$, $a_k > 0$ if and only if $As_k \neq 0$; hence, $As_k = 0$, A is of rank $k - 1$, and As_i, $i \leqq k - 1$, span R_A (see above proof). We will show that for any x, f is minimized by

$$z = x - A_{k-1} g(x),$$

where $g(x) = \text{grad} f(x) = Ax + b$. First, note that $\hat{x}$ minimizes f if and only if $g(\hat{x}) = 0$. For any such $\hat{x}$, let

$$\begin{aligned} v &= A_{k-1} g(x) - (x - \hat{x}) = A_{k-1}(g(x) - g(\hat{x})) - (x - \hat{x}) \\ &= A_{k-1} A(x - \hat{x}) - (x - \hat{x}), \end{aligned}$$

where, by Corollary 1, $Av = 0$, since there is no d_k such that $As_k \neq 0$ ($a_k \neq 0$). Then,

$$z = x - A_{k-1} g(x) = \hat{x} - v$$

and

$$g(z) = A(\hat{x} - v) + b = A\hat{x} + b = g(\hat{x}) = 0.$$

Thus, z minimizes f. In particular, $x_{k-1} - A_{k-1} g_{k-1}$ minimizes f.

In order to make use of Corollary 1, we must be able to determine that there is no d_k such that $a_k \neq 0$. This is easily accomplished by trying $N - (k - 1)$ directions independent of $d_1, \cdots, d_{k-1}$, determining these directions by use of the Gram-Schmidt procedure given in Appendix I (by (2) of Lemma 1, we do not have to try $d_1, \cdots, d_{k-1}$ or any combination of these).

It is often desirable to find A, A^*, and the vector of least norm that minimizes f (least squares minimum of f). Once a minimizing solution has been obtained, these are all easily found by straightforward application of the following results. Regarding notation used below, note that in the algorithm, $As_i = y_i$ when f is given by (II.1).

LEMMA 4. (1) *Let* $s_1, \cdots, s_k$ *be an A-orthogonal basis for* P_A. *Then* $\hat{A} = A^*$, *where*

$$\hat{A} = \sum_{i=1}^{k} \frac{s_i s_i'}{s_i' A s_i}.$$

(2) *Let* $z_1, \cdots, z_k$ *be an* A^**-orthogonal basis for* P_A^*. *Then,*

$$A = \sum_{i=1}^{k} \frac{z_i z_i'}{z_i' A^* z_i}.$$

Suppose that in Step (1) *of the algorithm, there is no* d_k *such that* $a_k \neq 0$, *and* f *is given by* (II.1). *Then,*

(3) $A = \sum_{i=1}^{k-1} As_i s_i' A / s_i' As_i$.

(4) *Using* A *from* (3), *an* A*-orthogonal basis* r_i, $i \leqq k-1$, *for* P_A *can be generated from* As_i, $i \leqq k-1$, *by the Gram-Schmidt orthogonalization procedure, viz. for* $i = 1, \cdots, k-1$,

$$M_0 = 0,$$

$$r_i = As_i - M_{i-1} As_i,$$

$$M_i = M_{i-1} + r_i r_i' A / r_i' A r_i,$$

then, by (1) *of Lemma* 4,

$$A^* = \sum_{i=1}^{k-1} r_i r_i' / r_i' A r_i.$$

(5) *Let* $\hat{x}$ *minimize* f. *Then, the least squares minimum of* f *is* x^*,

$$x^* = A^* A \hat{x}.$$

Proof. (1) For x in P_A $(=R_A)$, $x = \sum_{i=1}^{k} a_i As_i$ and clearly $A\hat{A}x = x$. For x in $R_A^\perp$ $(=N_A)$, $x's_i = 0$, $i \leqq k$, and clearly $Ax = 0$.

(2) Follows from (1) of Lemma 4 and the fact that $(A^*)^* = A$.

(3) Since $AA^*A = A$, $s_i' AA^* As_i = s_i' As_j$, i.e., As_i, $i \leqq k-1$, are A^*-orthogonal. Then As_i, $i \leqq k-1$, are a basis for P_A^* since they span P_A^* (see proof of Corollary 1) and (3) follows from (2) of Lemma 4.

(4) Follows from (1) of Lemma 4, as stated.

(5) This is a property of the pseudo-inverse (see [7, Appendix C, 17.10]).

Executive Office of the President
Office of Emergency Preparedness
Systems Evaluation Division
Washington, D. C. 20504

1. R. FLETCHER & M. J. D. POWELL, "A rapidly convergent descent method for minimization," *Comput. J.*, v. 6, 1963, pp. 163–168. MR **27** #2096.

2. B. A. MURTAGH & R. W. H. SARGENT, *A Constrained Minimization Method With Quadratic Convergence*, presented at the I. M. A. Conference on Optimization, held at the University of Keele, 1968.

3. R. FLETCHER, *A New Approach to Variable Metric Algorithms*, U.K.A.E.A. Report HL 69/4734.

4. M. J. D. POWELL, "A survey of numerical methods for unconstrained optimization," *SIAM Rev.*, v. 12, 1970, pp. 79–97. MR **41** #2900.

5. H. H. ROSENBROCK, "An automatic method for finding the greatest or least value of a function," *Comput. J.*, v. 3, 1960/61, pp. 175–184. MR **24** #B2081.

6. H. W. SORENSON, "Comparison of some conjugate direction procedures for function minimization," *J. Franklin Inst.*, v. 288, 1969, pp. 421–441. MR **40** #8233.

7. L. A. ZADEH & C. A. DESOER, *Linear System Theory*, McGraw-Hill, New York, 1963.

8. H. J. KELLEY & G. E. MYERS, *Conjugate Direction Methods for Parameter Optimization*, presented at the 18th Congress of the International Astronautical Federation, Belgrade, Yugoslavia, September 1967.