NW

stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE NW 16/75 MARCH

J.C.P. BUS

ON THE CONVERGENCE OF A CLASS OF VARIABLE METRIC ALGORITHMS

NW

5752 832

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS (MOS) subject classification scheme (1970): 65K05, 90C30.

On the convergence of a class of variable metric algorithms

by

J.C.P. Bus.

ABSTRACT

A class of variable metric algorithms is presented for finding the unconstrained minimum of a differentiable function of several variables. These algorithms make use of a relaxed strategy for the line search. Linear convergence of these algorithms is proved without imposing any essential conditions on the updating matrix, provided the function is convex. Furthermore, sufficient conditions on the updating matrix to obtain superlinear convergence are given.

KEY WORDS & PHRASES: *Unconstrained minimization, variable metric methods.*

CONTENTS

1. INTRODUCTION

In this report some results are given about the convergence of variable metric algorithms for finding the unconstrained minimum of a differentiable function of several variables. Let F be a twice differentiable function

(1.1) $\qquad F\colon S \subset \mathbb{R}^n \to \mathbb{R}$,

where S is some convex region in $\mathbb{R}^n$ and F is bounded below on S.
The variable metric algorithm, introduced by DAVIDON [6] and reformulated by FLETCHER & POWELL [10] consists basically of three steps.
Given a point x and a positive definite symmetric matrix H, then a new iterate x^*, and a new positive definite symmetric n-th order matrix H^* is calculated by

1. Calculate a direction of search

(1.2) $\qquad d = -Hg$,

where $g = g(x)$ is the gradient of F at x;

2. calculate some approximation α^* of α_m, where α_m is defined by

(1.3) $\qquad F(x+\alpha_m d) = \min\limits_{\alpha>0} (F(x+\alpha d))$;

set $x^* = x + \alpha d$;

3. calculate a new positive definite symmetric matrix

(1.4) $\qquad H^* = H + U(H,x^*,x,g^*,g)$,

where $g^* = g(x^*)$ and U is some symmetric matrix, which is called the updating matrix. The rank of U is usually one or two.

From the definition of d in (1.2), it is clear that

(1.5) $$\frac{d}{d\alpha} F(x+\alpha d) = d^T g = - g^T Hg < 0.$$

Hence, the function is decreasing in the direction d.

The line search, i.e. the choice of α^*, varies from one algorithm to another. In some algorithms α^* is simply chosen equal to 1 (see for example POWELL [20]), while in other algorithms α^* is calculated with cubic or quadratic interpolation in order to approximate α_m in some sense (see for example FLETCHER & POWELL [10] or FLETCHER [9]).
Using a computer for calculating α_m, it is obviously not possible to obtain a value which exactly equals α_m. However, many results about the behaviour of variable metric algorithms are based on the fact that $\alpha^* = \alpha_m$ (see for example POWELL [21], DIXON [8]). So, these results have only theoretical value and are not applicable to the algorithms published. As far as we know, only LENARD [14] gave conditions for convergence when the line search is not exact. The goal of this report is to choose a strategy for the line search which allows us to define a class of algorithms, in which the updating formula is not yet specified, and for which convergence with a rate that is at least linear, may be proved, provided the function is convex.

In literature, several strategies are proposed to obtain α with as few extra function evaluations as possible, without disturbing the fine behaviour of the algorithm. One of these is given by GOLDSTEIN & PRICE [11]. Their strategy was used in a modified Newton algorithm. However, as is shown by FLETCHER [9], it may also be used in variable metric algorithms. The idea is based on Taylor series expansion of F at x.

Define

(1.6) $$h(\alpha) = F(x) - F(x+\alpha d).$$

Then, choose

$$\alpha^* = 1 \qquad \text{if } h(1) \geq \nu,$$

otherwise choose α^* such that

(1.7) $$\nu \leq h(\alpha^*) \leq 1 - \nu.$$

Here ν is chosen to be a value between 0 and 0.5. WOLFE [25] showed that the right hand inequality of (1.7) is equivalent to

(1.8) $$\frac{d^T g^*}{d^T g} < 1 - \nu$$

while the left hand inequality of (1.7) can be replaced by the condition that the function

(1.9) $$f(\alpha) = F(x+\alpha d)$$

is monotonously decreasing on the interval $(0,\alpha_m)$.

In section 3 we propose a class of algorithms which make use of (1.8) for the line search. In these algorithms we did not specify the updating matrix. For constructing this class and for proving convergence for convex functions, we use some results given by LENARD [13], which are summed up in section 2.

In section 4 we give conditions that should be imposed on the updating matrix in order to obtain superlinear convergence.

Furthermore, in section 5 we consider some particular updating formulas and give some results for the specific members of the given class of algorithms that use these formulas.

In section 6 and 7, we give some numerical results and conclusions. An ALGOL 60 procedure implementing an algorithm which uses the updating formula given by BROYDEN [3], FLETCHER [9] and SHANNO [23], is given in appendix.

2. PRELIMINARY RESULTS

Let F be a given, twice differentiable function

(2.1) $$F: S \subset \mathbb{R}^n \to \mathbb{R},$$

where S is some convex region in $\mathbb{R}^n$ and let F be bounded below on S. We assume in this section that the second derivative matrix G(x) of F satisfies

$$(2.2) \qquad 0 < m \;||u||^2 \le u^T G(x)\, u \le M \;||u||^2,$$

for all $x \in S$ and $u \ne 0 \in \mathbb{R}^n$, where m and M are two positive constants and $||\cdot||$ denotes the euclidean norm. In fact, we demand that F is strictly convex. The following lemma is easily proved (POWELL [21], LENARD [13]).

LEMMA 2.1. *If, for any two points* x *and* x', $\delta = x' - x$ *and* $\gamma = g(x') - g(x)$, *where* $g(x) = \nabla F(x)$ *then*

$$(2.3) \qquad m\,||\delta|| \le ||\gamma|| \le M\,||\delta||, \qquad \delta^T\gamma \ge mM^{-1}\,||\delta||\;||\gamma||\,.$$

Now we give some results due to LENARD [13].

LEMMA 2.2. *Let* $x^* = x + \alpha^* d$, *where* $g^T(x)d < 0$ *and* $g^T(x^*)d = \theta g^T(x)d$, *for some* θ, $-1 \le \theta \le 1$. *Let*

$$\cos\psi = g^T(x)d/(||g(x)||\;||d||).$$

Then

$$(2.4) \qquad \tfrac{1}{2}M^{-1}(1-\theta^2)\;||g(x)||^2\;\cos^2\psi \le F(x) - F(x^*) \le \tfrac{1}{2}m^{-1}(1-\theta^2)\,||g(x)||^2\;\cos^2\psi.$$

PROOF. See LENARD [13]. □

As a consequence of lemma 2.2 we have:

LEMMA 2.3. *Let* $\bar{x}$ *be the position of the minimum of* F(x), *then*

$$(2.5) \qquad \tfrac{1}{2}M^{-1}\;||g(x)||^2 \le F(x) - F(\bar{x}) \le \tfrac{1}{2}m^{-1}\;||g(x)||^2\,.$$

PROOF. See LENARD [13]. $\square$

Using lemmas 2.2 and 2.3 we obtain the following theorem.

THEOREM 2.4. *Let an iterative method for calculating the minimum of* F *generate a sequence of points* $\{x_k\}_{k=0}^{\infty}$.

Denote

$$(2.6)\qquad \theta_k = \delta_k^T g_{k+1} \;/\; \delta_k^T g_k,$$

where

$$(2.7)\qquad \delta_k = x_{k+1} - x_k, \quad g_k = g(x_k) = \nabla F(x_k).$$

Assume that

$$(2.8)\qquad F(x_{k+1}) \le F(x_k), \qquad k = 0,1,2,\ldots,$$

$$(2.9)\qquad 1 - (\theta_k)^2 \ge c, \qquad k = 0,1,2,\ldots,$$

$$(2.10)\qquad -\, g_k^T \delta_k \ge r \;\|g_k\|\;\|\delta_k\|, \qquad k = 0,1,2,\ldots,$$

for two constants c *and* r *with* $0 < c,\ r < 1$. *Then* $\{x_k\}_{k=0}^{\infty}$ *converges to a minimum of* F *at a rate that is at least linear.*

PROOF. (See also LENARD [13]) Using inequalities (2.4) and (2.5) and denoting

$$(2.11)\qquad \cos\psi_k = g_k^T \delta_k \;/\; (\|g_k\|\;\|\delta_k\|), \qquad k = 0,1,2,\ldots,$$

we obtain

$$(2.12)\qquad M^{-1}m(1-\theta_k^2)\cos^2\psi_k \le \frac{F(x_k)-F(x_{k+1})}{F(x_k)-F(\bar{x})} \le m^{-1}M(1-\theta_k^2)\cos^2\psi_k\,.$$

Hence, using (2.8) up to (2.10) leads to

$$(2.13) \qquad 0 \le \frac{F(x_{k+1})-F(\bar{x})}{F(x_k)-F(\bar{x})} \le 1 - M^{-1} m\, c\, r^2 < 1,$$

which proves the theorem. □

3. A CLASS OF VARIABLE METRIC ALGORITHMS

Let

$$(3.1) \qquad U = U(A,u,v)$$

be a symmetric matrix for any given matrix A and vectors u and v. Then, we define a variable metric algorithm A(U), depending on U as follows.

<u>Algorithm A(U)</u>

A0. (Initialisation)

Let x_0 be an initial guess for the position of the minimum of F, let H_0 be a symmetric initial approximation to the inverse hessian (matrix of second derivatives) of F at x_0 and let r and c be given constants such that $0 < r,\ c < 1$.

Then, for $k = 0,1,2,\ldots$ we compute x_{k+1} and H_{k+1} as follows:

A1. (calculation of search direction)

set $p_k = - H_k g_k$, if $\|H_k\|$ is bounded,

$p_k = - g_k$, otherwise;

if $- g_k^T p_k \ge r \|g_k\| \|p_k\|$, then set $d_k = p_k$,

if $- g_k^T p_k \le - r \|g_k\| \|p_k\|$, then set $d_k = - p_k$,

otherwise, compute $\lambda_k > 0$ such that

$$(3.2) \qquad g_k^T(\lambda_k I+H_k)g_k = r \|g_k\| \|\lambda_k g_k + H_k g_k\|$$

and set $d_k = - (\lambda_k I+H_k)g_k$;

A2. (line search)

calculate $\alpha_k > 0$ such that

$$F(x_k+\alpha_k d_k) \le F(x_k) \tag{3.3}$$

and

$$\left(\frac{d_k^T g(x_k+\alpha_k d_k)}{d_k^T g_k}\right)^2 \le 1 - c; \tag{3.4}$$

A3. (calculating new approximation).

set $x_{k+1} = x_k + \alpha_k d_k$;

A4. (updating metric)

$$H_{k+1} = H_k + U(H_k,\delta_k,\gamma_k),$$

where $\delta_k = \alpha_k d_k$ and $\gamma_k = g(x_{k+1}) - g_k$.

It is easily shown that $\lambda_k > 0$ and $\alpha_k > 0$ always exist such that (3.2) up to (3.4) are satisfied; (3.2) is based on an idea, first given by LEVENBERG [16] and used by MARQUARDT [17]. Choosing the direction of search according to A1 ensures us of having a direction in which the function is sufficiently decreasing. If λ_k is increasing, then d_k tends to the steepest descent direction $(-g_k)$.

The following theorem is an immediate consequence of theorem 2.4 and the construction of algorithm A(U).

THEOREM 3.1. *Let* F *be given by* (2.1) *and let its second derivative satisfy* (2.2). *Let* $x_0 \in S$, H_0 *be a given symmetric matrix and* c *and* r *constants such that* $0 < c, r < 1$. *Then, for any symmetric* U(A,u,v), *the sequence of points* $\{x_k\}_{k=0}^{\infty}$, *generated by* A(U), *converges to the position of a minimum of* F *at a rate that is at least linear.*

In most variable metric algorithms known, the initial matrix H_0 and the updating formula U are chosen, such that H_k remains positive definite. It seems more likely to do so, since $H(x) = G^{-1}(x)$ is positive definite at the position of the minimum and our goal is to let H_k be as good an approximation to $H(x_k)$ as possible. Restricting ourselves to such updating formulas we may simplify algorithm A(U) by replacing A1 by:

B1. (simplified calculation of search direction).

set $p_k = -H_k g_k$, if $\|H_k\|$ is bounded

$p_k = -g_k$, otherwise.

If $-g_k^T p_k \geq r\|g_k\| \|p_k\|$, then set $d_k = p_k$, otherwise, compute $\lambda_k > 0$ such that (3.2) is satisfied and set

$$d_k = -(\lambda_k I + H_k)\, g_k.$$

In the sequel, the algorithm obtained in this way is called B(U).

The advantage of algorithms A(U) or B(U) is the separation of the different problems arising in variable metric algorithms. On one hand we specify the choice of the direction of search and the line search in such a way that convergence is assured. On the other hand we are completely free in choosing the updating formula U in order to try to obtain superlinear convergence.

4. CONDITIONS FOR SUPERLINEAR CONVERGENCE OF A(U)

In this section we derive conditions for U, such that algorithm A(U) converges superlinearly. Before stating the final theorem, we give a lemma. The proof of this lemma, as well as the proof of the theorem, is based on the proof of a similar theorem given by GOLDSTEIN & PRICE [11]. Their theorem, however, was given for a Newton algorithm with a strategy for the line search based on (1.6) and (1.7).

LEMMA 4.1. *Let* F *be given by* (2.1), *let its second derivative* G(x) *satisfy* (2.2) *and let, moreover,* G(x) *satisfy a Lipschitz condition:*

(4.1) $$\| G(x) - G(x') \| \leq L \| x-x' \| ,$$

for all $x, x' \in S$ *and a certain constant* L. *Let the sequence of points* $\{x_k\}_{k=0}^{\infty}$ *be generated by algorithm* $A(U)$, *where* $r \leq m/(3M)$ *and* $c \leq 0.5$. *Denote, for arbitrary* N

(4.2) $$T(N) = \{u \in \mathbb{R}^n \mid u = \sum_{k=N}^{\infty} \mu_k g_k, \text{ for certain } u_k\}$$

and assume that for all $\varepsilon > 0$ *there exists an* N *such that*

(4.3) $$\| (H_k - H(x_k))u \| \leq \varepsilon \| u \| ,$$

for all $k \geq N$ *and* $u \in T(N)$. *Then, an* N_0 *exists such that* $\delta_k = - H_k g_k$ *for all* $k > N_0$.

PROOF. First we prove that an integer N_1 exists, such that

(4.4) $$0 < \frac{1}{2M} \| u \|^2 \leq u^T H_k u \leq \frac{3}{2m} \| u \|^2 ,$$

for all $k > N_1$ and $u \in T(N_1)$, $u \neq 0$.

Choose

(4.5) $$\varepsilon = 1/(2M).$$

Then, an N_1 exists such that (4.3) is satisfied for all $k > N_1$ and $u \in T(N_1)$.
Writing

(4.6) $$u^T H_k u = u^T (H_k - H(x_k)) u + u^T H(x_k) u$$

and using

(4.7) $$|u^T (H_k - H(x_k)) u| \leq \varepsilon \| u \|^2 ,$$

for all $k > N_1$ and $u \in T(N_1)$, we obtain with (2.2):

(4.8) $$(M^{-1}-\varepsilon)\,\|u\|^2 \le u^T H_k u \le (m^{-1}+\varepsilon)\,\|u\|^2$$

Hence, with the special choice of ε, we obtain immediately the required result (4.4). Analogously, we can prove

(4.9) $$\|H_k u\| \le \frac{3}{2m}\,\|u\|\,,$$

for all $u \in T(N_1)$ and $k > N_1$. Now, substituting $p_k = -H_k g_k$ and using (4.8) and (4.9) we obtain

(4.10) $$\frac{-g_k^T p_k}{\|g_k\|\,\|p_k\|} = \frac{g_k^T H_k g_k}{\|g_k\|\,\|H_k g_k\|} \ge \frac{m}{3M} \ge r, \qquad \text{for all } k > N_1.$$

Since $\|H_k\|$ is bounded for all $k > N_1$, we may therefore choose $d_k = p_k$ in step A1 of algorithm A(U).

For proving the existence of an integer N_2, such that $\alpha_k = 1$ satisfies (3.3) and (3.4) for all $k > N_2$, we choose $\varepsilon = 1/\sqrt{2}$.

Using theorem 3.1 we know that $\{x_i\}_{i=0}^{\infty}$ converges to $\bar{x}$ with $F(\bar{x})$ is minimal. Hence, with Taylor's theorem,

(4.11) $$\|g_k - g(\bar{x})\| = \|g_k\| \le M\|x_k - \bar{x}\|\,.$$

Now, choose N' such that

(4.12) $$\|x_k - \bar{x}\| \le 2m^3\varepsilon/(27LM^2)$$

and

(4.13) $$\|(H_k - H(x_k))u\| \le m\varepsilon\|u\|/(6M^2),$$

for all $k > N'$ and $u \in T(N')$. Using Taylor's theorem again, we obtain

(4.14) $$g(x_k+d_k) = g_k + G(\eta_k)d_k,$$

where $\eta_k = x_k + \theta d_k$, $0 \le \theta \le 1$. Hence

$$(4.15)\qquad \frac{d_k^T g(x_k+d_k)}{d_k^T g_k} = \frac{d_k^T g_k + d_k^T G(\eta_k) d_k}{d_k^T g_k} .$$

Since $d_k = - H_k g_k$ we have

$$(4.16)\qquad |d_k^T g_k| = |g_k^T H_k g_k| \ge \frac{1}{2M} \| g_k \|^2$$

and

$$(4.17)\qquad |d_k^T g_k + p_k^T G(\eta_k) p_k| \le |d_k^T (G(\eta_k) - G(x_k)) d_k| + |g_k^T H_k G(x_k)(H(x_k) - H_k) g_k|$$

Using (4.1), (4.11), (4.12) and the fact that $d_k = - H_k g_k$ we have, for $k > N'$

$$(4.18)\qquad |d_k^T (G(\eta_k) - G(x_k)) d_k| \le L \| \eta_k - x_k \| \, \| d_k \| \le \frac{\varepsilon}{4M} \| g_k \|^2 .$$

For the second term in the right hand side of (4.17) we obtain with (2.2), (4.4) and (4.13)

$$(4.19)\qquad |g_k^T H_k G(x_k)(H(x_k) - H_k) g_k| \le \frac{\varepsilon}{4M} \| g_k \|^2 , \qquad \text{for } k > \max(N_1, N').$$

Substituting (4.16), (4.18) and (4.19) in (4.15) we obtain

$$(4.20)\qquad \frac{|d_k^T g(x_k+p_k)|}{|d_k^T g_k|} \le \varepsilon, \qquad \text{for all } k > \max(N_1, N').$$

Hence, with the choice of ε and c we have proved that (3.4) is satisfied for $\alpha_k = 1$ and for all $k > N_2 = \max(N_1, N')$

Finally, we have to prove that an N_3 exists, such that

$$(4.21)\qquad F(x_k+d_k) \le F(x_k), \qquad \text{for all } k > N_3 .$$

Therefore, denote

$$(4.22)\qquad h(x,\alpha) = -(F(x) - F(x+\alpha d))/(\alpha g^T d).$$

With Taylor's theorem we may write

$$(4.23)\qquad h(x_k,\alpha) = 1 + \frac{\alpha d_k^T G(\eta_k) d_k}{2 g_k^T d_k}\ .$$

So, using $G_k d_k = -g_k$, for $k > N_1$, we obtain

$$(4.24)\qquad h(x_k,\alpha) = 1 - \frac{\alpha}{2} + \alpha \left(\frac{d_k^T (G(\eta_k) - G_k) d_k}{2 g_k^T d_k} \right)\ .$$

With (4.18) this leads to

$$\tfrac{1}{2}(1-\varepsilon) \le h(x_k, 1) \le \tfrac{1}{2}(1+\varepsilon),$$

for all $k > N_3 \ge N_1$ and arbitrary $\varepsilon > 0$. Hence, by the definition of h, we obtain for $k > N_3$

$$(4.25)\qquad F(x_k) - F(x_k+d_k) \ge -\tfrac{1}{2} g_k^T d_k (1-\varepsilon) \ge \tfrac{1}{2} r(1-\varepsilon) \| g_k \| \, \| d_k \|\ .$$

Choosing $\varepsilon < 1$ proves (4.21). By combining (4.10), (4.20) and (4.21) and by choosing $N_0 = \max(N_2, N_3)$ we have proved the lemma. $\square$

Using this lemma we are able to prove the following theorem about the superlinear convergence of algorithm A(U).

THEOREM 4.2. *Let* F *be given by* (2.1) *and let its second derivative* G(x) *satisfy* (2.2) *and* (4.1). *Let, moreover,* r, c *and the updating formula* U *satisfy the conditions of lemma* 4.1. *Then, the sequence of points, generated by* A(U), *converges superlinearly to a point at which* F(x) *has a minimum.*

PROOF. Suppose $\lim_{k\to\infty} x_k = \bar{x}$. Then, using Taylor's theorem

$$\| x_{k+1} - \bar{x} \| = \| \delta_k + x_k - \bar{x} \| = \| (H(\eta_k) - H_k) g_k \|$$

for $\eta_k = x_k + \theta(\bar{x}-x_k)$, $0 \le \theta \le 1$.
Hence

$$\| x_{k+1} - \bar{x}\| \le \| (H(\eta_k)-H(x_k))g_k\| + \| (H(x_k)-H_k)g_k\| .$$

With

$$\| H(\eta_k)-H(x_k)\| \le \| H(\eta_k)\| \, \| G(x_k)-G(\eta_k)\| \, \| H(x_k)\|$$

we obtain

$$\| x_{k+1} - \bar{x}\| \le (\frac{L}{m^2} \| \eta_k - x_k\| + \varepsilon)\| g_k\| .$$

Using Taylor's formula again gives

$$(4.26) \qquad \| x_{k+1} - \bar{x}\| \le M(\frac{L}{m^2} \| \bar{x} - x_k\| + \varepsilon)\| x_k - \bar{x}\| ,$$

for arbitrary $\varepsilon > 0$ and $k > N = N(\varepsilon)$.
This completes the proof. □

It is obvious from (4.26), that the asymptotic order of convergence of algorithm A(U) depends on

$$S_k = \sup_{u \in T(k)} \| (H_k - H(x_k))u\| .$$

If $S_k = O(\| u\|^p)$ for some $p > 1$, then the order of convergence of algorithm A(U) equals min(2,p).

5. SOME PARTICULAR UPDATING FORMULAS

We consider in this section the following updating formulas:

$$(5.1) \qquad U^D(H,\delta,\gamma) = \frac{\delta\delta^T}{\delta^T\gamma} - \frac{H\gamma\gamma^T H}{\gamma^T H\gamma} ,$$

which is, originally due to DAVIDON [6];

(5.2) $$U^F(H,\delta,\gamma) = \left(1 + \frac{\gamma^T H\gamma}{\delta^T\gamma}\right)\frac{\delta\delta^T}{\delta^T\gamma} - \frac{H\gamma\delta^T + \delta\gamma^T H}{\delta^T\gamma},$$

which is due to FLETCHER [9], BROYDEN [3], and SHANNO [23];

(5.3) $$U^C(H,\delta,\gamma) = \theta U^D(H,\delta,\gamma) + (1-\theta)U^F(H,\delta,\gamma),$$

where $\theta = \theta(H,\delta,\gamma)$ is some parameter such that $0 \leq \theta_k \leq 1$ (see FLETCHER [9]).

Before proving some properties of these updating formulas we give two lemmas which appear to by useful.

LEMMA 5.1. *Let* A *be a symmetric matrix with eigenvalues* $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. *Let* A^* *be obtained from* A *by adding a symmetric perturbation matrix of rank* 1 *to it*

(5.4) $$A^* = A + \alpha vv^T,$$

for some vector v *and some scalar* $\alpha \neq 0$.
Let the eigenvalues of A^* *be denoted by* $\lambda_1^* \geq \lambda_2^* \geq \ldots \geq \lambda_n^*$. Then,

(5.5) $$\alpha > 0 \Rightarrow \lambda_1^* \geq \lambda_1 \geq \lambda_2^* \geq \ldots \geq \lambda_n^* \geq \lambda_n;$$

(5.6) $$\alpha < 0 \Rightarrow \lambda_1 \geq \lambda_1^* \geq \lambda_2 \geq \ldots \geq \lambda_n \geq \lambda_n^*.$$

PROOF. See WILKINSON [24], section 44-47. □

LEMMA 5.2. *Let* A *and* A^* *be given as in lemma* 5.1. *Let* x_i *denote the eigenvector of* A *corresponding to eigenvalue* λ_i, i = 1,...,n. *Then, the following implications hold for* q, p = 1,...,n.

(5.7) $v^T x_p = 0 \Rightarrow \lambda_p$ *is an eigenvalue of* A^* *and* x_p *is the corresponding eigenvector.*

(5.8) $$\lambda_p = \lambda_{p+1} = \dots = \lambda_q \Rightarrow \lambda^*_{p+1} = \dots = \lambda^*_q = \lambda_q, \quad \text{if } \alpha > 0$$
$$\lambda_p = \lambda^*_p = \lambda^*_{p+1} = \dots = \lambda^*_{q-1}, \quad \text{if } \alpha < 0.$$

(5.9) $$\lambda^*_q = \lambda_p \ (q=p-1,p,p+1) \Rightarrow v^T x_p = 0 \text{ or } \lambda_q = \lambda_p \text{ and } q \neq p.$$

PROOF. Suppose $A = X\Lambda X^T$, where $\Lambda = \mathrm{diag}(\lambda_1,\lambda_2,\dots,\lambda_n)$ and X is the orthogonal matrix of eigenvectors $x_1,\dots,x_n$. Then, with the notation $u = X^T v$, we have

(5.10) $$A^* = X(\Lambda+\alpha uu^T)X^T.$$

Hence, the eigenvalues of A^* are those of $\Lambda + \alpha uu^T$. Some elementary algebra shows that these eigenvalues are equal to the roots of the equation

(5.11) $$K(\mu) = \prod_{i=1}^{n} (\lambda_i-\mu) + \alpha \sum_{j=1}^{n} u_j^2 \prod_{\substack{i=1\\ i\neq j}}^{n} (\lambda_i-\mu) = 0,$$

where $u = (u_1,u_2,\dots,u_n)^T = (v^T x_1, v^T x_2,\dots,v^T x_n)^T$. Now, suppose $u_p = 0$. Then

$$K(\mu) = (\lambda_p-\mu)\left[\prod_{\substack{i=1\\ i\neq p}}^{n} (\lambda_i-\mu) + \alpha \sum_{\substack{j=1\\ j\neq p}}^{n} u_j^2 \prod_{\substack{i=1\\ i\neq j,p}}^{n} (\lambda_i-\mu)\right].$$

Hence, $K(\lambda_p) = 0$ and λ_p is an eigenvalue of A^*. Furthermore, since

(5.12) $$A^* x_p = (A+\alpha vv^T)x_p = \lambda_p x_p + \alpha u_p v = \lambda_p x_p,$$

we proved implication (5.7).

In order to prove (5.8), assume that $\lambda_p = \lambda_{p+1} = \dots = \lambda_q$. Then,

$$K(\mu) = (\lambda_p-\mu)^{q-p}\left[\prod_{i\in I} (\lambda_i-\mu) + \alpha \sum_{j=1}^{n} u_j^2 \prod_{i\in I_j} (\lambda_i-\mu)\right],$$

where $I = \{i \mid 1 \le i < p,\ q \le i \le n\}$ and $I_j = I\backslash\{j\}$. Therefore, using lemma 5.1, we have proved implication (5.8).

Finally, suppose $\lambda_q^* = \lambda_p$, $q = p - 1$, p or $p + 1$. Then,

$$K(\lambda_q^*) = \alpha u_p^2 \prod_{\substack{i=1 \\ i \neq p}}^{n} (\lambda_i - \lambda_q^*) = 0.$$

Hence, $u_p = v^T x_p = 0$ or $\lambda_i = \lambda_q^*$ for some $i \neq p$. Using lemma 5.1, simple checking yields $\lambda_q = \lambda_p$. This completes the proof. □

LEMMA 5.3. *If* H_0 *is positive definite, then*

(5.13) $$H_k^c = H_0 + \sum_{j=1}^{k-1} U^c(H_j, \delta_j, \gamma_j),$$

where U^c *is defined by* (5.3), *is positive definite for all* k *if* $\delta_j^T \gamma_j > 0$ *for all* $j \leq k$.

PROOF. First we prove the statement for $\theta \equiv 1$ in (5.3), by showing that, if H_k^D is positive definite, then H_{k+1}^D is positive definite. To simplify the notation we will omit the indices k and the superscript D, and mark with an asterisk those quantities which should have subscript k+1.
Denote

$$\bar{H} = H - \frac{H\gamma\gamma^T H}{\gamma^T H \gamma}.$$

Then, by lemma 5.1 and the positive definiteness of H, the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ of H and $\bar{\lambda}_1 \geq \bar{\lambda}_2 \geq \ldots \geq \bar{\lambda}_n$ of $\bar{H}$ satisfy

(5.14) $$\lambda_1 \geq \bar{\lambda}_1 \geq \lambda_2 \geq \ldots \geq \lambda_{n-1} \geq \bar{\lambda}_{n-1} \geq \lambda_n > \bar{\lambda}_n = 0,$$

where the last equality holds since $\bar{H}\gamma = 0$. Hence γ is an eigenvector of $\bar{H}$ with eigenvalue 0.
With (5.1) we obtain

$$H^* = \bar{H} + \frac{\delta\delta^T}{\delta^T \gamma}.$$

Therefore, denoting the eigenvalues of H^* by $\lambda_1^* \geq \lambda_2^* \geq \ldots \geq \lambda_n^*$, we know by lemma 5.1 and $\delta^T\gamma > 0$ that

$$\lambda_1^* \geq \bar{\lambda}_1 \geq \lambda_2^* \geq \ldots \geq \lambda_n^* \geq \bar{\lambda}_n = 0.$$

Since $\lambda_n \neq 0$, we know by (5.14) that $\bar{\lambda}_{n-1} \neq \bar{\lambda}_n$, so that using lemma 5.2 and $\delta^T\gamma > 0$, we see that $\lambda_n^* > \bar{\lambda}_n = 0$. Hence H^* is positive definite, which proves that H_k^D is positive definite for all k. As simple checking may show, we have the relation

$$(5.15) \qquad [H + U^F(H,\delta,\gamma)]^{-1} = H^{-1} + U^D(H,\gamma,\delta).$$

Hence, with the same arguments as above we can prove that $(H_k^F)^{-1}$ and, consequently, H_k^F is positive definite. Therefore, using (5.3) the lemma is proved. □

It is obvious that algorithm $B(U^c)$ converges at least linearly for any quadratic function with a positive definite hessian matrix (see section 3 for the definition of algorithm B and (5.3) for the definition of U^c). In order to prove superlinear convergence in this case we need the following theorem.

<u>THEOREM 5.4</u>. *Let* F *be a quadratic function with positive definite hessian* G *and let* H_0 *be any positive definite symmetric matrix. Let the sequence of matrices* $\{H_k\}_{k=0}^{\infty}$ *be generated by* $B(U^c)$, *where* $U^c = U^c(H,\delta,\gamma)$ *is defined by* (5.3). *Then we have*

$$(5.16) \qquad \lim_{k\to\infty} \| (H_k - H)u \| \,/\, \| u \| = 0,$$

for all $u \in T(N)$. *Here* $H = G^{-1}$ *and* T(N) *is defined by* (4.2).

<u>PROOF</u>. Define $K_k = G^{\frac{1}{2}} H_k G^{\frac{1}{2}}$ and $z_k = G^{\frac{1}{2}} \delta_k$. Then, using $\gamma_k = G\delta_k$ for quadratic functions, we have for $\theta = 1$ in (5.3):

$$(5.17) \qquad K_{k+1} = K_k + \frac{z_k z_k^T}{z_k^T z_k} - \frac{K_k z_k z_k^T K_k}{z_k^T K_k z_k} \quad .$$

Consider $Z(N) = \{u \in \mathbb{R}^n \mid u = \sum_{k=N}^{\infty} \mu_k z_k$, for certain $\mu_k\}$. Then, since $Z(N_1) \subset Z(N_2)$ if $N_1 \geq N_2$, and $\mathbb{R}^n$ is finite-dimensional, there exists an N_0 such that $Z(k) = Z(N_0)$ for all $k \geq N_0$. Suppose P is a projector on $Z(N_0)$. Then $Pz_k = z_k$ and denoting $L_k = PK_kP$ we have from (5.17).

$$(5.18) \qquad L_{k+1} = L_k + \frac{z_k z_k^T}{z_k^T z_k} - \frac{L_k z_k z_k^T L_k}{z_k^T L_k z_k} .$$

By the definition of K_k and L_k and using (2.3) and lemma 5.3 we know that L_k is positive semi-definite for all k.

We restrict ourselves to the nonzero eigenvalues $\lambda_k^{(1)} \geq \lambda_k^{(2)} \geq \ldots \geq \lambda_k^{(r)}$ of L_k whose corresponding eigenvectors $x_k^{(1)},\ldots,x_k^{(r)}$ are in $Z(N_0)$, where r equals the dimension of $Z(N_0)$.
Let

$$\bar{L}_k = L_k - \frac{L_k z_k z_k^T L_k}{z_k^T L_k z_k}$$

have eigenvalues $\bar{\lambda}_k^{(1)} \geq \bar{\lambda}_k^{(2)} \geq \ldots \geq \bar{\lambda}_k^{(n)}$, then lemma 5.1 shows that

$$\lambda_k^{(1)} \geq \bar{\lambda}_k^{(1)} \geq \lambda_k^{(2)} \geq \ldots \geq \lambda_k^{(r)} \geq \bar{\lambda}_k^{(r)} = 0$$

and since $L_k z_k = 0$ we see that the eigenvalues of L_{k+1} are equal to $\lambda_k^{(1)},\ldots,\lambda_k^{(r-1)}$ and 1. Since $Z(k)$ is r-dimensional for all $k > N_0$ we know that $N_1 \geq N_0 + r$ exists, such that $z_{N_0}, z_{N_0+1},\ldots,z_{N_1}$ span the whole space $Z(N_0)$. Hence, using (5.7) we may conclude that an index $j\,(N_0 \leq j \leq N_1)$ exists, such that $z_j^T x_j^{(1)} \neq 0$. Now suppose

$$\lambda_{N_0}^{(1)} = \lambda_{N_0}^{(2)} = \ldots = \lambda_{N_0}^{(q)} > 1.$$

Then with lemma 5.2, we see that $\lambda_{N_0+j}^{(q-1)} \neq \lambda_{N_0+j}^{(q)}$. Repeated use of this argument leads to $\lambda_m^{(1)} \neq \lambda_m^{(2)}$, for some $m > N_0 + j$. Since $z_j^T x_j^{(2)} \neq 0$, for some $j \geq m$, we obtain with (5.10), that a number N_2 exists such that

$$\lambda_{N_2}^{(1)} < \lambda_{N_0}^{(1)}.$$

Therefore, using the fact, that all L_k have an eigenvalue equal to 1, we have shown that $\lambda_k^{(1)}$ converges to 1 for k tending to infinity. Analogously, we can prove that $\lambda_k^{(r)}$ converges to 1, since $\lambda_{k+1}^{(r)} = \bar{\lambda}_k^{(r-1)} \geq \lambda_k^{(r)}$. Therefore we proved that L_k converges to a matrix with all eigenvalues, corresponding to eigenvector in $Z(N_0)$, equal to 1. Hence

$$\| (L_k - I)u \| \,/\, \| u \| \to 0,$$

for all $u \in Z(N_0)$ and k tending to infinity. Since P is a projector on $Z(N_0)$ we have

$$\| (K_k - I)u \| \,/\, \| u \| \to 0,$$

for all $u \in Z(N_0)$ and therefore

(5.19) $$\| G^{\frac{1}{2}} (H_k - H) G^{\frac{1}{2}} u \| \,/\, \| u \| \to 0.$$

Since G is positive definite we can show

(5.20) $$u \in Z(N_0) \Rightarrow G^{\frac{1}{2}} u \in T(N_0).$$

This is easily seen using $G^{\frac{1}{2}} z_k = \gamma_k$ and

$$T(N_0) = \{ u \in \mathbb{R}^n \,|\, u = \sum_{k=N_0}^{\infty} \mu_k \gamma_k, \text{ for certain } \mu_k \}$$

which holds because of

$$u = \sum_{k=N_0}^{\infty} \mu_k g_k \Rightarrow u = - \sum_{k=N_0}^{\infty} \left(\sum_{i=M}^{k} \mu_i \right) \gamma_k$$

and

$$u = \sum_{k=N_0}^{\infty} \mu_k \gamma_k \Rightarrow u = \sum_{k=N_0}^{\infty} \nu_k \gamma_k, \text{ with } \nu_{N_0} = - \mu_{N_0}$$
$$\nu_k = \mu_k - \mu_{k+1}.$$

Using (5.19) en (5.20), the theorem is proved for $U^C = U^D$ ($\theta=1$ in (5.3)). However, with $\gamma_k = G\delta_k$ and (5.15) we can use the same arguments for proving that

$$\lim_{k\to\infty} \| (G_k^F - G)u \| \,/\, \| u \| = 0,$$

for all $u \in \{u \in \mathbb{R}^n \mid u = \sum_{k=N_0}^{\infty} \nu_k \delta_k$, for certain N_0 and $\nu_k\}$. Therefore the theorem is also proved for $U^{C^0} = U^F$ ($\theta=0$ in (5.3)) and, in fact, for all θ, $0 \le \theta \le 1$. □

As an immediate consequence of theorem 4.2 and 5.4 we have the following extension of a theorem given by FLETCHER [9].

THEOREM 5.5. *Let* F *be a quadratic function with positive definite hessian* G *and let* H_0 *be any positive definite symmetric matrix. Then, the sequence of points* $\{x_k\}_{k=0}^{\infty}$ *generated by algorithm* $B(U^C)$, *where* U^C *is defined by* (5.3), *converges superlinearly to a minimum of* F.

In our opinion, theorem 5.5 is an indication for the usefulness of theorem 4.2 as a tool for proving superlinear convergence of algorithm A(U) for various updating formulas U and for more general (convex) functions.

6. NUMERICAL COMPARISONS

In order to obtain some insight in the practical usefulness of algorithm A(U), we have implemented algorithm $A(U^D)$, where U^D is given by (5.1), and algorithm $A(U^F)$, where U^F is given by (5.2).

These two algorithms are compared with an implementation of an algorithm given by FLETCHER [9], which is called algorithm F in this section. A detailed description of this implementation, together with an ALGOL 60 procedure, is given in BUS [4].

The functions, used for comparison are known from literature.

1. A function given by ROSENBROCK [22].

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

The initial guess is chosen to be $(-1.2,1)^T$.

2. A function given by LEON [15].

$$F(x) = 100(x_2-x_1^3)^2 + (1-x_1)^2.$$

The initial guess is $(-1.2,-1)^T$.

3. A function given by BEALE [1].

$$F(x) = \sum_{k=1}^{3} (c_k-x_1(1-x_2^k)),$$

where $c_1 = 1.5$, $c_2 = 2.25$ and $c_3 = 2.625$.
The initial guess is $(0.1,0.1)^T$.

4. A function given by FLETCHER & POWELL [10].

$$F(x) = 100((x_3-10\theta)^2 + (r-1)^2) + x_3$$

where

$$r = (x_1^2+x_2^2)^{\frac{1}{2}}$$

and

$$2\pi\theta = \begin{cases} \arctan(x_2/x_1) & \text{if } x_1 > 0 \\ \pi + \arctan(x_2/x_1) & \text{if } x_1 < 0. \end{cases}$$

The initial guess is $(-1,0,0)^T$.

5. A function given by COLVILLE [5], also known as Wood's function.

$$F(x) = 100(x_2-x_1^2)^2 + (1-x_1)^2 + 90(x_4-x_3)^2 + (1-x_3)^2 +$$
$$+ 10.1[(x_2-1)^2 + (x_4-1)^2] + 19.8(x_2-1)(x_4-1).$$

The initial guess is $(-3,-1,-3,-1)^T$.

6. A function given by POWELL [18].

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

The initial guess is $(3,-1,0,1)^T$.

7. Another function given by POWELL [19].

$$F(x) = 3 - (1+(x_1-x_2)^2)^{-1} - \sin(0.5\pi x_2 x_3) - \exp(-((x_1+x_3)/\ x_2-2)^2).$$

The initial guess is $(0,1,2)^T$.

8. A function given by BOX [2].

$$F(x) = \sum_{i=1}^{10} (\exp(-ix_1/10) - \exp(-ix_2/10) - x_3(\exp(-i/10) - \\ - \exp(-i)))^2.$$

The initial guess is $(0,20,1)^T$.

In all tests H_0 is chosen equal to the identity matrix, $c = 0.0001$ and $r = 0.01$ or 0.1. The testing has been done on a Cyber 73 computer with a machine precision of 48 bits. The results are listed in table 6.1, where n_f denotes the number of function evaluations and n_i the number of iteration steps needed to obtain the position of the minimum within a relative and absolute precision of 10^{-5}. In this table N means that 151 function evaluations were not sufficient to obtain the required result, but the algorithm did converge. D means that no convergence or convergence to a non-minimizing stationary point occurred.

table 6.1.

function	Alg. $A(U^F)$				Alg. $A(U^D)$				ALG. F	
	r = 0.1		r = 0,01		r = 0.1		r = 0.01			
	n_i	n_f	n_i	n_f	n_i	n_f	n_i	n_f	n_i	n_f
1	28	37	32	42	72	113	34	40	40	46
2	-	N	55	58*	-	N	-	N	70	78*
3	27	31	27	31	32	36*	-	N	28	32
4	-	N	40	57	56	106	50	61	46	62
5	12	14	12	14	13	15	13	15	12	16
6	11	21	11	21	12	22	12	22	12	14
7	82	134	73	97	-	N	-	D	70	83
8	76	150	21	30	-	N	-	D	30	35

* precision not reached.

Table 6.1 indicates that algorithm $A(U^F)$ with r = 0.01 is at least as efficient as algorithm F. Furthermore, the choice r = 0.1 appears to be bad for ill-conditioned problems, i.e. problems for which m/M (see (2.2)) is very small relative to 1. This is affirmed by the theory, since in lemma 4.1 r is related to the quantity m/M. Finally, using the updating formula U^D seems to be a bad choice for ill-conditioned problems. As is mentioned earlier in various papers (e.g. FLETCHER [9]), the tendency to singularity of the matrices H_k^D (k=0,1,2,...) is greater than of the matrices H_k^F.

7. DISCUSSION

In this report, we gave a class of variable metric algorithms without specifying the updating formula. It is proved that these algorithms are convergent (at least linearly) for convex functions. Furthermore, conditions on the updating formula are given to obtain superlinear convergence. In our opinion, the separation of the problem of the line search on one hand and the choice of the updating formula on the other hand, provides a good starting point for examinating the various updating formulas. It is clear that the choice of the updating formula is only a tool for increasing the order of convergence, since chosing $H_k = I$ will give also a convergent algorithm. Although LENARD [14] gave conditions for superlinear convergence of a Davidon-Fletcher-Powell-algorithm with a relaxed strategy for the line-search, these conditions are not very transparent and difficult to implement in an algorithm. Moreover, she considered only DAVIDON's [6] updating formula (cf. (5.1)), which is not as good as the formula given by FLETCHER [9], BROYDEN [3] and SHANNO [23] (cf. (5.2)), as is shown by the results in section 6. We hope that the results given in this report will contribute to a more general convergence theory for variable metric algorithms in optimization.

ACKNOWLEDGEMENTS

The author is grateful to prof. T.J. DEKKER for his valuable suggestions concerning some eigenvalue problems; to dr. P.J. VAN DER HOUWEN and drs. J. KOK for their careful reading of the manuscript, and to mr. Th. Gunsing, mrs. R. Riechelmann and mr. D. Zwarst for typing and printing printing this report.

REFERENCES

[1] BEALE, E.M.L., *On an iterative method for finding a local minimum of a function of more than one variable*, Techn. Report No. 25, Statistical Techniques Research Group, Princeton Univ. (1958).

[2] BOX, M.J., *A comparison of several current optimization methods and the use of transformations in constrained problems*, Comp. J. 9 (1966) p. 67-77.

[3] BROYDEN, C.G., *The convergence of a class of double-rank minimization algorithms, Part* I *and* II, J. Inst. Maths. Applics. 6 (1970) p. 76-90, 222-231.

[4] BUS, J.C.P., *Minimization of functions of several variables* (Dutch), Mathematical Centre, report NR 29/72 (1972) Amsterdam.

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

[6] DAVIDON, W.C., *Variable metric method for minimization,* Argonne Nat. Lab. Report ANL - 5990 (1959).

[7] DAVIDON, W.C., *Variance algorithm for minimization,* Comp. J. 10 (1968) p. 406-410.

[8] DIXON, L.C.W., *Quasi-Newton algorithms generate identical points,* Math. Progr. 2 (1972) p. 383-387.

[9] FLETCHER, R., *A new approach to variable metric algorithms,* Comp. J. 13 (1970) p. 317-322.

[10] FLETCHER, R. & POWELL, M.J.D., *A rapidly convergent descent method for minimization,* Comp. J. 6 (1963) p. 163-168.

[11] GOLDSTEIN, A.A. & PRICE, J.F., *An effective algorithm for minimization,* Numer. Math. 10 (1967) p. 184-189.

[12] HEMKER, P.W., (ed.), *NUMAL, a library of numerical procedures in* ALGOL 60, 8 vols., Mathematical Centre, Amsterdam (1974).

[13] LENARD, M.L., *Practical convergence conditions for unconstrained optimization,* Math. Progr. 4 (1973) p. 309-323.

[14] LENARD, M.L., *Practical convergence conditions for the Davidon-Fletcher-Powell method,* Univ. of Wisconsin, MRC Tech. Summ. Rep. 1356 (1974).

[15] LEON, A., *A comparison of eight known optimizing procedures,* In: Lavi, A. and Vogl, T.P. (eds.), Recent advances in optimization techniques, Wiley (1966).

[16] LEVENBERG, K., *A method for the solution of certain non-linear problems in least squares,* Appl. Math. 2 (1944) p. 164-168.

[17] MARQUARDT, D.W., *An algorithm for least-squares estimation of non-linear parameters,* J. SIAM, 11 (1963) p. 431-441.

[18] POWELL, M.J.D., *An iterative method for finding stationary values of a function of several variables*, Comp. J. 5 (1962) p. 147-151.

[19] POWELL, M.J.D., *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, Comp. J. 7 (1964) p. 155-162.

[20] POWELL, M.J.D., In: Abadie, J. (ed.), *Integer and non-linear programming*, North-Holland (1970).

[21] POWELL, M.J.D., *On the convergence of the variable metric algorithms.* J. Inst. Maths. Applics, 7 (1971) p. 21-36.

[22] ROSENBROCK, H.H., *An automatic method for finding the greatest or least value of a function*, Comp. J. 3 (1960) p. 175-184.

[23] SHANNO, D.F., *Conditioning of quasi-Newton methods for function minimization*, Math. Comput. 24 (1970) p. 647-656.

[24] WILKINSON, J.H., *The algebraic eigenvalue problem*, Clarendon Press (1965).

[25] WOLFE, P., *Convergence conditions for ascent methods*, SIAM Rev. 11 (1969) p. 226-235.

APPENDIX

In this appendix we give the text of an ALGOL 60 procedure implementing algorithm $A(U^F)$. A description of the meaning of the formal parameters is also given. The procedures which are given as "code"-declarations are described in HEMKER [12].

calling sequence:

the heading of this procedure is:

<u>procedure</u> minimize(n, x, g, h, funct, in, out);

<u>value</u> n; <u>integer</u> n;

<u>array</u> x, g, h, in, out; <u>real</u> <u>procedure</u> funct;

the meaning of the formal parameters is:

n:	<arithmetic expression>;
	the number of variables of the function to be minimized;
x:	<array identifier>;
	<u>array</u> x[1 : n];
	the independent variables;
	entry: an approximation of the position of a minimum;
	exit: the calculated position of a minimum;
g:	<array identifier>;
	<u>array</u> g[1 : n];
	exit: the gradient of the function at the calculated position of the minimum;
h:	<array identifier>;
	a one - dimensional array h[1 : n × (n + 1) ÷ 2];
	the uppertriangle of an approximation of the inverse hessian is stored columnwise in h; i.e. the (i,j)-th element is given in h[j × (j + 1) ÷ 2 + i];
	if in[9] > 0 initializing of h will be done automatically

and the initial approximation of the inverse hessian will equal the unit-matrix multiplied with the value of in[6]; if in[9] < 0, then no initializing of h will be done and the user should give in h an approximation of the inverse hessian at the starting point;
the uppertriangle of an approximation of the inverse hessian at the calculated position of the minimum is delivered in h;

funct: <procedure identifier>;
the heading of this procedure should be:
<u>real</u> <u>procedure</u> funct(n, x, g); <u>value</u> n;
<u>integer</u> n; <u>array</u> x, g;

funct:= the value of the function evaluated at the point
as given in x[1:n];

the meaning of the formal parameters is:
n: <arithmetic expression>;
the number of variables;
x: <array identifier>; <u>array</u> x[1:n];
entry: the value of the variables for which the
function has to be evaluated;
g: <array identifier>; <u>array</u> g[1:n];
exit: the gradient of the function;

in: <array identifier>;
<u>array</u> in[0:10];

entry:

in[0]: the machine precision; for the cyber 73 a suitable value is $_{10}-14$;

in[1], in[2]: the relative and absolute tolerance for the improvement of the variables (relative to the current estimates of the variables);

in[3], in[4]: the relative and absolute tolerance for the difference between the penultimate and the ultimate function value;

the process is terminated if the improvement of the variables is less than norm(x) × in[1] + in[2], and the improvement of the function value is less than abs(f) × in[3] + in[4]; here norm(.) denotes the euclidean norm;

in[5]: the maximum number of function evaluations allowed; since the process is terminated at the end of an iteration step, it may happen that the actual number of function evaluations , given in out[4], exceeds the value of out[5] at the end of the process;

in[6]: the maximum steplength allowed;

in[7]: a value that is used for calculating the direction of search, see section 3; usually, a suitable value is 0.01;

in[8]: a value that is used for calculating the steplength , see section 3; usually, a suitable value is $_{10}-4$;

in[9]: a value for controlling the initialisation of h, see above; when no information about the inverse hessian at the starting point is known, then the user is advised to set in[9]:= 1;

in[10]: a lowerbound for the function value;

out: <array identifier>;

array out[1:6];

exit:

out[1]: this value gives information about the termination of the process;

out[1] = 0: normal termination;

out[1] = 1: the process is terminated at the end of a step in which the number of function evaluations exceeded the value of in[5];

out[1] = 2: this is only possible when input is wrong; for instance, in[0] = 0 or in[9] $\leq$ 0 and h is not initialized well;

out[1] = 3: the procedure cannot improve the function value, while the steplength in the last step was not small enough; this may happen if programming of the gradient is wrong, if the precision asked for is too high, or if the function is very flat in a neighbourhood of the position of the minimum (the problem is ill-conditioned);

out[2]: the calculated minimum value of the function;

out[3]: the value of the function at the initial guess;

out[4]: the number of calls of funct necessary to obtain the calculated result;

out[5]: the total number of iteration steps performed;

out[6]: the euclidean norm of the stepvector in the last iteration step.

data and results:

usually the precision of the calculated position x of the minimum will be at least equal to norm(x) × in[1] + in[2]; however, we can not guarantee such a result; the solution will possibly not satisfy this condition if the hessian matrix is singular at the position of the minimum; the user can discover such a situation by examining the approximation to the inverse hessian at the position of the minimum which is given in h; when the norm of this matrix is very large relative to 1 then it is very likely that the hessian matrix is (almost) singular at the solution, and that the precision is not reached.

source text:

```
procedure minimize(n, x, g, h, funct, in, out);
value n; integer n;
```

```
array x, g, h, in, out;
real procedure funct;
begin integer it, fcntmax, fcnt, err;
    real f, f0, macheps, rtol, atol, rtolf,atolf, r, c, h0,
    alfa, nrmdelta, fmin, smx;
    array delta, g0[1:n];

    real procedure vecvec(l, u, shift, a, b); code 34010;
    real procedure symmatvec(l, u, s, a, b); code 34018;
    procedure inivec(l, u, a, x); code 31010;
    procedure inisymd(l, u, s, a, x); code 31013;
    procedure elmvec(l, u, shift, a, b, x); code 34020;
    procedure mulvec(l, u, shift, a, b, x); code 31020;
    procedure dupvec(l, u, shift, a, b); code 31030;
    boolean procedure zeroin(x, y, fx, tolx); code 34150;
    real procedure mininder(x, y, fx, dfx, tolx); code 34435;

    real procedure eval(n, x, g); value n; integer n;
    array x, g;
    begin fcnt:= fcnt + 1; if fcnt ≥ fcntmax then err:= 1;
        eval:= funct(n, x, g)
    end eval;

    procedure update(h, n, delta, gamma); value n;
    integer n; array h, delta, gamma;
    begin integer i; real dg; array hg[1:n];
    procedure fleupd(h, n, v, w, c1, c2); code 34213;
    for i:= 1 step 1 until n do
    hg[i]:= symmatvec(1, n, i, h, gamma);
    dg:= 1 / vecvec(1, n, 0, delta, gamma);
    fleupd(h, n, delta, hg, dg,
    (1 + vecvec(1, n, 0, gamma, hg) × dg) × dg)
end update;

procedure length(x, alfa, delta, nrmdelta, f, g);
real alfa, nrmdelta, f; array x, delta, g;
begin real dg, dg0, f0, lb, t, aid; array x1[1:n];

    real procedure linfu(par); value par; real par;
    if par = 0 then linfu:= f0 else
    begin dupvec(1, n, 0, x1, x);
        elmvec(1, n, 0, x1, delta, par);
        linfu:= f:= eval(n, x1, g)
    end linfu;

    real procedure dlinfu(par); value par; real par;
    if par = 0 then dlinfu:= dg0 else
    dlinfu:= dg:= vecvec(1, n, 0, delta, g);

    real procedure tol;
    tol:=(if (dg / dg0) ↑ 2 ≤ c ∧ f < f0 then aid
    else sqrt(vecvec(1, n, 0, x1, x1)) × rtol + atol);
    dg0:= vecvec(1, n, 0, delta, g); f0:= f;
    if it > n ∨ h0 < 0 then alfa:= 1 else
    begin alfa:= (fmin - f) × 2 / dg0;
        t:= (sqrt(vecvec(1, n, 0, x, x)) × rtol + atol) /
        nrmdelta; if alfa < t then alfa:= t
    end; lb:= 0;
    aid:= smx / nrmdelta; if alfa > aid then alfa:= aid;
    f:= mininder(alfa, lb, linfu(alfa), dlinfu(alfa), tol);
    if alfa = 0 then
    begin err:= 3; nrmdelta:= 0 end
    else if alfa ≠ 1 then
    begin mulvec(1, n, 0, delta, delta, alfa);
        nrmdelta:= nrmdelta × alfa
    end; dupvec(1, n, 0, x, x1)
end length;

boolean procedure test(er, a, nd, ed, ng, eg);
value er, a, nd, ed, ng, eg; integer er;
real a, nd, ed, ng, eg;
test:= er ≠ 0 ∨ (a = 1 ∧ nd ≤ ed ∧ ng ≤ eg);

boolean procedure direction(delta, nd, g, h);
real nd; array delta, g, h;
begin integer i; real ghg, nrmg2, aid, y, nrmg, par;
    boolean d;
    nrmg2:= vecvec(1, n, 0, g, g); nrmg:= sqrt(nrmg2);
    for i:= 1 step 1 until n do
    delta[i]:= -symmatvec(1, n, i, h, g);
    nd:= sqrt(vecvec(1, n, 0, delta, delta));
    ghg:= - vecvec( 1, n, 0, g, delta);
    aid:= nd × nrmg × r; if ghg ≥ aid then
    d:= true else if ghg ≤ -aid then
    begin mulvec(1, n, 0, delta, delta, -1); d:= true end
    else
    begin real procedure f(par); value par; real par;
        begin array v[1:n];
            dupvec(1, n, 0, v, delta);
            elmvec(1, n, 0, v, g, - par);
            f:= nrmg2 × par + ghg - sqrt(vecvec(1, n, 0, v, v))
            × nrmg × r
        end f;
        y:= 0;
        for i:= 1 step 1 until n × (n + 1)  ÷ 2 do
        begin aid:= abs(h[i]); if aid > y then y:= aid
        end; y:= y × n × 2; par:= 0;
        if ¬ zeroin(par, y, f(par), abs(par) × macheps +
        macheps) then d:= false else
       begin d:= true; elmvec(1, n, 0, delta, g, -par);
            nd:= sqrt(vecvec(1, n, 0, delta, delta))
        end
    end; direction:= d
end direction;
```

```
        macheps:= in[0] × 2; rtol:= in[1]; atol:= in[2]; rtolf:= in[3];
        atolf:= in[4]; fcntmax:= in[5]; smx:= in[6]; r:= in[7];
        c:= 1 - in[8]; h0:= in[9]; fmin:= in[10]; it:= err:= fcnt:= 0;
        out[3]:= f:= eval(n, x, g); if h0 > 0 then
        begin inivec(1, n × (n + 1) ÷ 2, h, 0);
            inisymd(1, n, 0, h, h0)
        end initialisation;

    iteration: it:= it + 1;
        dupvec(1, n, 0, g0, g); f0:= f;
        if ¬ direction(delta, nrmdelta, g, h) then err:= 2
        else length(x, alfa, delta, nrmdelta, f, g);
        if test(err, alfa, nrmdelta, sqrt(vecvec(1, n, 0, x, x)) ×
        rtol + atol, f0 - f, abs(f) × rtolf + atolf) then goto end;
        mulvec(1, n, 0, g0, g0, -1); elmvec(1, n, 0, g0, g, 1);
        update(h, n, delta, g0);
        goto iteration;
    end: out[1]:= err; out[2]:= f; out[4]:= fcnt; out[5]:= it;
        out[6]:= nrmdelta
    end minimize;
```