JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 4, No. 3, 1969

# Study on a Supermemory Gradient Method for the Minimization of Functions[1]

E. E CRAGG[2] AND A. V. LEVY[2]

Communicated by A. Miele

**Abstract.** A new accelerated gradient method for finding the minimum of a function $f(x)$ whose variables are unconstrained is investigated. The new algorithm can be stated as follows:

$$\tilde{x} = x + \delta x, \qquad \delta x = -\alpha g(x) + \sum_{i=1}^{k} \beta_i \, \delta x_i$$

where $x$ is an $n$-vector, $g(x)$ is the gradient of the function $f(x)$, $\delta x$ is the change in the position vector for the iteration under consideration, and $\delta x_i$ is the change in the position vector for the $i$th previous iteration. The quantities $\alpha$ and $\beta_i$ are scalars chosen at each step so as to yield the greatest decrease in the function; the scalar $k$ denotes the number of past iterations remembered.

For $k = 1$, the algorithm reduces to the memory gradient method of Ref. 2; it contains at most two undetermined multipliers to be optimized by a two-dimensional search. For $k = n - 1$, the algorithm contains at most $n$ undetermined multipliers to be optimized by an $n$-dimensional search.

Two nonquadratic test problems are considered. For both problems, the memory gradient method and the supermemory gradient method are compared with the Fletcher–Reeves method and the Fletcher–Powell–Davidon method. A comparison with quasilinearization is also presented.

---

[1] Paper received January 10, 1969. This research, supported by the Office of Scientific Research, Office of Aerospace Research, United States Air Force, Grant No. AF-AFOSR-828-67, is a condensation of the investigation described in Ref. 1. The authors are indebted to Professor Angelo Miele for stimulating discussions.

[2] Graduate Student in Aero-Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

## 1. Introduction

In Ref. 2, the memory gradient algorithm for finding the minimum of a function $f(x)$ whose variables are unconstrained was presented. Compared with the ordinary gradient method, this algorithm has the advantage of high speed since it produces quadratic convergence. In this paper, a generalization of the memory gradient algorithm, called the supermemory gradient algorithm, is investigated. While the memory gradient algorithm is based on remembering one previous iteration, the supermemory gradient algorithm is based on remembering $k$ previous iterations. The added complication is the need for a $(k + 1)$-dimensional search at each iteration as opposed to the two-dimensional search required by the memory gradient algorithm.

## 2. Definitions

The following definitions are used throughout the paper:

(a) The symbol $x$ denotes the position vector whose scalar components are $x^1, x^2, \ldots, x^n$.

(b) The symbol $f(x)$ denotes a scalar function of the vector $x$. It is assumed that the function $f$ is continuous and has continuous first and second derivatives.

(c) The symbol $g(x)$ denotes the column vector whose components are the first partial derivatives of $f$ with respect to the scalar variables $x^1, x^2, \ldots, x^n$. This is the gradient of the function $f$.

(d) The symbol $H(x)$ denotes the square matrix whose components are the second partial derivatives of the function $f$ with respect to the scalar variables $x^1, x^2, \ldots, x^n$.

(e) The symbol $x$ denotes the nominal point. The symbol $\tilde{x}$ denotes the point following $x$.

(*f*) The symbol $\delta x$ denotes the displacement leading from a point to the next point.

(*g*) The symbol $\delta x_i$ denotes the $i$th displacement preceding $\delta x$. Therefore, $\delta x_1$ is the displacement immediately preceding $\delta x$, $\delta x_2$ is the displacement preceding $\delta x_1$, and so on.

(*h*) The superscript $T$ denotes the transpose of a matrix.

## 3. Statement of the Problem

The purpose of this paper is to find the minimum of a function

$$f = f(x) \tag{1}$$

whose variables are unconstrained. The basic idea is to construct corrections $\delta x$ leading from a nominal point $x$ to a varied point $\tilde{x}$ such that

$$f(\tilde{x}) < f(x) \tag{2}$$

Therefore, by an iterative procedure (that is, through successive decreases in the value of the function), it is hoped that the minimum of $f$ is approached to any desired degree of accuracy.

To first-order terms, the values of the function at the varied point and the nominal point are related by

$$f(\tilde{x}) \cong f(x) + \delta f(x) \tag{3}$$

where the first variation $\delta f(x)$ is given by

$$\delta f(x) = g^T(x)\, \delta x \tag{4}$$

with

$$\delta x = \tilde{x} - x \tag{5}$$

Also to first-order terms, the greatest decrease in the value of the function is achieved if the first variation (4) is minimized. Here, we limit our analysis to those variations $\delta x$ which satisfy the constraint

$$K = \left(\delta x - \sum_{i=1}^{k} \beta_i\, \delta x_i\right)^T \left(\delta x - \sum_{i=1}^{k} \beta_i\, \delta x_i\right) \tag{6}$$

where $\delta x_i$ denotes the $i$th displacement preceding $\delta x$ and where $K$, $k$, $\beta_i$ are prescribed. Since this constraint involves not only the correction $\delta x$ of the iteration under consideration but also the corrections $\delta x_i$ of several previous iterations, the resulting algorithm is called *supermemory gradient algorithm.* It reduces to the memory gradient algorithm of Ref. 2 for $k = 1$.

## 4. Derivation of the Algorithm

Standard methods of the theory of maxima and minima show that the fundamental function of this problem is the scalar function

$$F = g^T(x)\, \delta x + (1/2\alpha) \left(\delta x - \sum_{i=1}^{k} \beta_i\, \delta x_i\right)^T \left(\delta x - \sum_{i=1}^{k} \beta_i\, \delta x_i\right) \tag{7}$$

where $1/2\alpha$ is a constant Lagrange multiplier. The optimum system of variations must be such that

$$G(\delta x) = 0 \tag{8}$$

where $G$ is the gradient of the function $F$ with respect to the scalar variables $\delta x^1, \delta x^2, \ldots, \delta x^n$. In the light of (7), the explicit form of (8) is

$$\delta x = -\alpha g(x) + \sum_{i=1}^{k} \beta_i \, \delta x_i \tag{9}$$

Upon substituting (9) into (6), we see that

$$K = \alpha^2 g^T(x) \, g(x) \tag{10}$$

Therefore, a one-to-one correspondence exists between the value of the constant $K$ and the value of $\alpha$. This being the case, one can bypass prescribing $K$ and reason directly on $\alpha$, as in the considerations which follow.

## 5. Properties of the Algorithm

In order to study the properties of the algorithm (9), it is convenient to introduce the following definitions:

$$\gamma_0 = \alpha, \qquad u_0 = -g(x) \tag{11}$$

and

$$\gamma_i = \beta_i, \qquad u_i = \delta x_i, \qquad i = 1, \ldots, k \tag{12}$$

If Eqs. (9), (11)–(12) are combined, the displacement vector becomes

$$\delta x = \sum_{i=0}^{k} \gamma_i u_i \tag{13}$$

Hence, the position vector at the end of any iteration becomes

$$\tilde{x} = x + \sum_{i=0}^{k} \gamma_i u_i \tag{14}$$

For each point $x$, Eq. (14) defines a $(k+1)$-parameter family of points $\tilde{x}$, for which the function $f$ takes the form

$$f(\tilde{x}) = f\left(x + \sum_{i=0}^{k} \gamma_i u_i\right) = F(\gamma_0, \gamma_1, \ldots, \gamma_k) \tag{15}$$

The greatest decrease in the function $F$ occurs if the parameters $\gamma_i$ satisfy the following necessary conditions:

$$F_{\gamma_i} = 0, \qquad i = 0, 1,..., k \tag{16}$$

where the subscripts $\gamma_i$, $i = 0, 1,..., k$, denote partial derivatives. We note that

$$F_{\gamma_i} = g^T(\tilde{x})\, u_i, \qquad i = 0, 1,..., k \tag{17}$$

Therefore, Eqs. (16) become

$$g^T(\tilde{x})\, u_i = 0, \qquad i = 0, 1,..., k \tag{18}$$

and show that the gradient $g(\tilde{x})$ is orthogonal to each of the vectors $u_0, u_1, ..., u_k$. Because of (11)–(12), we conclude that $g(\tilde{x})$ is orthogonal to $g(x)$ as well as every previous displacement $\delta x_1, \delta x_2, ..., \delta x_k$.

If Eq. (13) is premultiplied by $g^T(\tilde{x})$, the following result is obtained:

$$g^T(\tilde{x})\, \delta x = \sum_{i=0}^{k} \gamma_i g^T(\tilde{x})\, u_i \tag{19}$$

which, in the light of (18), implies that

$$g^T(\tilde{x})\, \delta x = 0 \tag{20}$$

Therefore, $g(\tilde{x})$ is also orthogonal to the correction $\delta x$.

The first variation of the function $f(x)$ is given by

$$\delta f(x) = g^T(x)\, \delta x \tag{21}$$

which, because of (13), can be rewritten as

$$\delta f(x) = \sum_{i=0}^{k} \gamma_i g^T(x)\, u_i \tag{22}$$

and, in the light of (11), is equivalent to

$$\delta f(x) = -\alpha g^T(x)\, g(x) + \sum_{i=1}^{k} \gamma_i g^T(x)\, u_i \tag{23}$$

For the previous iteration, Eqs. (18) become

$$g^T(x)\, u_i = 0, \qquad i = 1, 2,..., k \tag{24}$$

Therefore, the first variation (23) becomes

$$\delta f(x) = -\alpha g^T(x)\, g(x) \tag{25}$$

**5.1. Summary of the Algorithm.** For any iteration except the first $k$ iterations, the complete algorithm can be summarized as follows: (a) for a given nominal point $x$, the gradient $g(x)$ is known; the vectors $\delta x_i$, $i = 1, 2,..., k$, are known from previous iterations; therefore, the vectors $u_i$, $i = 0, 1,..., k$, are known; (b) the optimum values of the multipliers $\gamma_i$, $i = 0, 1,..., k$, must be determined by solving Eqs. (16), as in Section 6; (c) once the multipliers are known, the correction $\delta x$ to the position vector $x$ is determined using Eq. (13); and (d) the new position vector $\tilde{x}$ is computed through Eq. (5).

**5.2. Starting the Algorithm.** Of course, operations (a) through (d) imply that the vectors $\delta x_i$, $i = 1, 2,..., k$, are known from previous iterations. Since this is not the case for the first $k$ iterations, some assumption concerning $\delta x_i$ or $\gamma_i$ must be made in order to start the algorithm. If $N$ denotes the iteration number, the simplest assumption is represented by

$$\begin{aligned} &\gamma_1 = 0, \quad \gamma_2 = 0,..., \gamma_k = 0 \quad \text{for} \quad N = 1 \\ &\qquad\quad \gamma_2 = 0,..., \gamma_k = 0 \quad \text{for} \quad N = 2 \\ &\qquad\quad \cdots\cdots\cdots\cdots\cdots\cdots \\ &\qquad\qquad\quad \gamma_k = 0 \quad \text{for} \quad N = k \end{aligned} \tag{26}$$

For $N = 1$, the system (16) must be replaced by (26-1) and

$$F_{\gamma_0} = 0 \tag{27}$$

which is equivalent to stating that the first iteration is a gradient iteration. Analogously, for $N = 2$, the system (16) must be replaced by (26-2) and

$$F_{\gamma_0} = 0, \qquad F_{\gamma_1} = 0 \tag{28}$$

which is equivalent to stating that the second iteration is a memory gradient iteration. Finally, for $N = k$, the system (16) must be replaced by (26-k) and

$$F_{\gamma_0} = 0, \quad F_{\gamma_1} = 0, \quad ..., \quad F_{\gamma_{k-1}} = 0 \tag{29}$$

which is equivalent to stating that the $k$th iteration is a supermemory gradient iteration of order $k - 1$.

**5.3. Quadratic Function.** For a quadratic function, it can be shown that the supermemory gradient algorithm reduces to the Fletcher–Reeves algorithm [see Eqs. (48)–(50)], regardless of the value of $k$.

## 6. Search Technique

The next step is to solve Eqs. (16) for the optimum values of the parameters $\gamma_i$, that is, those values which yield the minimum of the function (15). Clearly, the supermemory gradient method requires a $(k+1)$-dimensional search, while the ordinary gradient method ($k=0$) requires a one-dimensional search and the memory gradient method ($k=1$) requires a two-dimensional search. While several kinds of multidimensional search can be imagined, the one described here is based on quasilinearization with built-in safeguards to ensure that the function decreases at every step of the iterative search.

Let the corrections to $\gamma_i$ from arbitrary nominal values $\gamma_i^*$ be denoted by

$$\delta\gamma_i = \gamma_i - \gamma_i^*, \qquad i = 0, 1, \ldots, k \tag{30}$$

If quasilinearization is applied to Eqs. (16), one obtains the linear algebraic equations

$$\begin{aligned}
&F_{\gamma_0\gamma_0}\Delta\gamma_0 + F_{\gamma_0\gamma_1}\Delta\gamma_1 + \cdots + F_{\gamma_0\gamma_k}\Delta\gamma_k + F_{\gamma_0} = 0\\
&F_{\gamma_1\gamma_0}\Delta\gamma_0 + F_{\gamma_1\gamma_1}\Delta\gamma_1 + \cdots + F_{\gamma_1\gamma_k}\Delta\gamma_k + F_{\gamma_1} = 0\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&F_{\gamma_k\gamma_0}\Delta\gamma_0 + F_{\gamma_k\gamma_1}\Delta\gamma_1 + \cdots + F_{\gamma_k\gamma_k}\Delta\gamma_k + F_{\gamma_k} = 0
\end{aligned} \tag{31}$$

where all partial derivatives are evaluated at $\gamma_i^*$. Next, we imbed Eqs. (31) in the more general equations

$$\begin{aligned}
&F_{\gamma_0\gamma_0}\delta\gamma_0 + F_{\gamma_0\gamma_1}\delta\gamma_1 + \cdots + F_{\gamma_0\gamma_k}\delta\gamma_k + \rho\mu F_{\gamma_0} = 0\\
&F_{\gamma_1\gamma_0}\delta\gamma_0 + F_{\gamma_1\gamma_1}\delta\gamma_1 + \cdots + F_{\gamma_1\gamma_k}\delta\gamma_k + \rho\mu F_{\gamma_1} = 0\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&F_{\gamma_k\gamma_0}\delta\gamma_0 + F_{\gamma_k\gamma_1}\delta\gamma_1 + \cdots + F_{\gamma_k\gamma_k}\delta\gamma_k + \rho\mu F_{\gamma_k} = 0
\end{aligned} \tag{32}$$

where $\mu$ denotes a scaling factor and $\rho$ a direction factor such that

$$0 \leqslant \mu \leqslant 1, \qquad \rho = \pm 1 \tag{33}$$

Equations (32) admit the solutions

$$\delta\gamma_i = \rho\mu\, \Delta\gamma_i\,, \qquad i = 0, 1,..., k \tag{34}$$

where $\Delta\gamma_i$, $i = 0, 1,..., k$, are the solutions of Eqs. (31). The direction factor $\rho$ is determined in such a way that the first variation

$$\delta F = \sum_{i=0}^{k} F_{\gamma_i}\, \delta\gamma_i \tag{35}$$

is negative. From (34)–(35), we obtain

$$\delta F = \mu\rho \sum_{i=0}^{k} F_{\gamma_i}\, \Delta\gamma_i \tag{36}$$

Therefore, (36) is negative if the direction factor $\rho$ is chosen as follows:

$$\rho = -\text{sign}\left(\sum_{i=0}^{k} F_{\gamma_i}\, \Delta\gamma_i\right) \tag{37}$$

**6.1. Evaluation of the Derivatives.** The partial derivatives appearing in Eqs. (31) are given by

$$F_{\gamma_i} = g^T(\tilde{x})\, u_i\,, \qquad i = 0, 1,..., k \tag{38}$$

and

$$F_{\gamma_i\gamma_j} = u_i{}^T H(\tilde{x})\, u_j\,, \qquad i, j = 0, 1,..., k \tag{39}$$

If the matrix $H(\tilde{x})$ is not explicitly available, the second derivatives cannot be computed with Eqs. (39). In this case, one can use the following second-order difference scheme:

$$F_{\gamma_i\gamma_j} = \frac{1}{2\theta_i}\left[g^T(\tilde{x} + \theta_i u_i) - g^T(\tilde{x} - \theta_i u_i)\right] u_j\,, \qquad i, j = 0, 1,..., k \tag{40}$$

where

$$\theta_i = \epsilon/|\, u_i\,|, \qquad i = 0, 1,..., k \tag{41}$$

and where $\epsilon$ is a small number.

**6.2. Implementation of the Search.** To perform the search, nominal values $\gamma_i{}^*$ must be given to the multipliers. Then, one computes $\Delta\gamma_i$

with Eqs. (31) and $\rho$ with Eq. (37). Next, one sets $\mu = 1$ and computes $\delta\gamma_i$ with Eqs. (34) and $\gamma_i$ with Eqs. (30). If the following inequality is satisfied:

$$F(\gamma_0, \gamma_1, ..., \gamma_k) \leqslant F(\gamma_0^*, \gamma_1^*, ..., \gamma_k^*) \tag{42}$$

the scaling factor $\mu = 1$ is acceptable. Otherwise, the previous value of $\mu$ must be replaced by some smaller value in the range $0 \leqslant \mu \leqslant 1$ until Ineq. (42) is met; this can be obtained through bisection, that is, by successively dividing the value of $\mu$ by 2. At this point, the search step is completed. The values obtained for $\gamma_i$ become the nominal values $\gamma_i^*$ for the next search step, and the procedure is repeated until a desired degree of accuracy on $\gamma_i$ is obtained. In the absence of better information, the first step in the search procedure can be made with

$$\gamma_i^* = 0, \qquad i = 0, 1, ..., k \tag{43}$$

**6.3. Stopping Condition.** The search can be stopped when a predetermined condition is satisfied. The following are examples of stopping conditions:

$$\Psi(\gamma_0, \gamma_1, ..., \gamma_k) \leqslant \epsilon_1 \tag{44}$$

and/or

$$\Psi(\gamma_0, \gamma_1, ..., \gamma_k) \leqslant \epsilon_2 \Psi(0, 0, ..., 0) \tag{45}$$

where

$$\Psi = \sum_{i=0}^{k} F_{\gamma_i}^2 \tag{46}$$

An alternate stopping condition is that employed in Ref. 2, namely,

$$|\delta\gamma_i| \leqslant \epsilon_3 |\gamma_i|, \qquad i = 0, 1, ..., k \tag{47}$$

In the above expressions, $\epsilon_1$, $\epsilon_2$, $\epsilon_3$ denote small numbers.

## 7. Comparison Algorithms

In Section 8, several numerical examples are developed. In these examples, the memory gradient algorithm and the supermemory gradient algorithm are compared with the Fletcher–Reeves algorithm (Ref. 3), the Davidon algorithm (Ref. 4), and the quasilinearization algorithm. These algorithms are now reviewed:

**7.1. Fletcher–Reeves Algorithm.** This algorithm is represented by

$$\delta x = -\alpha p(x) \tag{48}$$

where

$$p(x) = g(x) + \gamma p(\hat{x}) \tag{49}$$

with

$$\gamma = \frac{g^T(x)\, g(x)}{g^T(\hat{x})\, g(\hat{x})} \tag{50}$$

In the above equations, $x$ denotes the nominal point and $\hat{x}$ the point preceding the nominal point. The scale factor $\alpha$ is determined by a one-dimensional search so as to minimize the function

$$f(\tilde{x}) = f\big(x - \alpha p(x)\big) = F(\alpha) \tag{51}$$

The Fletcher–Reeves algorithm is started or restarted by assuming $\gamma = 0$, which is equivalent to a pure gradient step.

**7.2. Fletcher–Powell–Davidon Algorithm.** This algorithm is represented by

$$\delta x = -\alpha p(x) \tag{52}$$

where

$$p(x) = A(x)\, g(x) \tag{53}$$

with

$$A(x) = A(\hat{x}) + \frac{\delta\hat{x}\, \delta\hat{x}^T}{\delta\hat{x}^T\, \Delta\hat{g}(x)} - \frac{A(\hat{x})\, \Delta\hat{g}(x)\, \Delta\hat{g}^T(x)\, A(\hat{x})}{\Delta\hat{g}^T(x)\, A(\hat{x})\, \Delta\hat{g}(x)} \tag{54}$$

and

$$\Delta\hat{g}(x) = g(x) - g(\hat{x}) \tag{55}$$

In the above equations, $A(x)$ denotes a positive-definite, symmetric matrix. The scale factor $\alpha$ is determined by a one-dimensional search so as to minimize the function

$$f(\tilde{x}) = f\big(x - \alpha p(x)\big) = F(\alpha) \tag{56}$$

The algorithm is started by assuming $A(x) = I$, where $I$ is the $n \times n$ identity matrix; again, this is equivalent to a pure gradient step.

**7.3. Uncorrected Quasilinearization Algorithm.** The uncorrected quasilinearization algorithm is represented by

$$\delta x = -H^{-1}(x)\, g(x) \tag{57}$$

where $H^{-1}(x)$ denotes the inverse of the second derivative matrix $H(x)$.

**7.4. Corrected Quasilinearization Algorithm.** Since (57) is not endowed with guaranteed descent properties, one can modify it as follows:

$$\delta x = -\mu\rho H^{-1}(x)\, g(x) \tag{58}$$

where $\mu$ denotes a scaling factor and $\rho$ a direction factor such that

$$0 \leqslant \mu \leqslant 1, \qquad \rho = \pm 1 \tag{59}$$

The direction factor $\rho$ is chosen so that the first variation of $f(x)$ is negative; this is precisely the case for

$$\rho = \operatorname{sign}\left(g^T(x)\, H^{-1}(x)\, g(x)\right) \tag{60}$$

The scale factor $\mu$ is determined so that the function

$$f(\tilde{x}) = f\left(x - \mu\rho H^{-1}(x)\, g(x)\right) = F(\mu) \tag{61}$$

satisfies the inequality

$$F(\mu) < F(0) \tag{62}$$

This can be obtained through a bisection process starting from $\mu = 1$.

## 8. Numerical Experiments

In order to compare the memory and supermemory gradient methods with the Fletcher–Reeves method, the Fletcher–Powell–Davidon method, and the quasilinearization method, two test functions have been considered. For identification purposes, they are called the Wood function (Ref. 5) and the Miele function. These functions are described below using scalar terminology, for simplicity.

*Wood Function.* This is a function of four variables defined by

$$\begin{aligned} f = {} & 100(x^2 - y)^2 + (x - 1)^2 + (z - 1)^2 + 90(z^2 - t)^2 \\ & + 10.1[(y - 1)^2 + (t - 1)^2] + 19.8(y - 1)(t - 1) \end{aligned} \tag{63}$$

It has the absolute minimum $f = 0$ at the point

$$x = 1, \quad y = 1, \quad z = 1, \quad t = 1 \tag{64}$$

and the nonoptimum stationary value $f = 7.876$ at the point

$$x = -0.9679, \quad y = 0.9471, \quad z = -0.9695, \quad t = 0.9512 \tag{65}$$

*Miele Function.* This is a highly nonlinear function of four variables defined by

$$f = (e^x - y)^4 + 100(y - z)^6 + \tan^4(z - t) + x^8 \tag{66}$$

It has the absolute minimum $f = 0$ at the points

$$x = 0, \quad y = 1, \quad z = 1, \quad t = 1 \pm n\pi \tag{67}$$

where $n$ is an integer.

**8.1. Experimental Conditions.** The numerical calculations were carried out in double-precision arithmetic on the Rice University Burroughs B-5500 computer. The algorithms were programmed in Extended ALGOL. Both the number of iterations $N$ and the computing time $T$ required for convergence were recorded.

*Initial Points.* The following initial points were considered:

$$\text{Wood function:} \quad x = -3, \quad y = -1, \quad z = -3, \quad t = -1 \tag{68}$$

$$\text{Miele function:} \quad x = 1, \quad y = 2, \quad z = 2, \quad t = 2 \tag{69}$$

*Convergence Criterion.* Since the Wood and Miele functions have the minimum value $f = 0$, the algorithms were considered to have converged to the desired solution when the inequality

$$f \leqslant 10^{-13} \tag{70}$$

was satisfied.

*Multidimensional Search.* The memory and supermemory gradient methods require the multidimensional search described in Section 6. The following stopping conditions were employed in conjunction:

$$\Psi(\gamma_0, \gamma_1, \ldots, \gamma_k) \leqslant 10^{-10} \tag{71}$$

and

$$\Psi(\gamma_0, \gamma_1, \ldots, \gamma_k) \leqslant \Psi(0, 0, \ldots, 0) \times 10^{-4} \tag{72}$$

where $\Psi$ is defined by Eq. (46).

*One-Dimensional Search.* The Fletcher–Reeves and Davidon methods require a one-dimensional search. Here, we use quasilinearization with built-in safeguards to ensure the decrease of the function at every step of the iterative search (Ref. 6). If one defines the function

$$f(\tilde{x}) = f(x - \alpha p) = F(\alpha) \tag{73}$$

the stopping conditions analogous to (71)–(72) are the following:

$$F_\alpha^2(\alpha) \leqslant 10^{-10} \tag{74}$$

and

$$F_\alpha^2(\alpha) \leqslant F_\alpha^2(0) \times 10^{-4} \tag{75}$$

**8.2. Experimental Results.** The results of the calculations are presented in Tables 1 and 2, where $N$ denotes the number of iterations and $T$

Table 1. Wood function

| Method | $N$ | $T$ (sec) | Remarks |
|---|---|---|---|
| Ordinary gradient | – | – | Does not converge in 1000 iterations |
| Fletcher–Reeves | 29 | 2.4 | $\Delta N = 5$ |
| Fletcher–Powell–Davidon | 39 | 4.5 | |
| Memory gradient | 18 | 2.0 | $k = 1$, $\Delta N = 5$ |
| Supermemory gradient | 4 | 1.4 | $k = 3$ |
| Quasilinearization, uncorrected | – | – | Converges to the nonoptimum stationary point |
| Quasilinearization, corrected | 39 | 1.5 | |

Table 2. Miele function

| Method | $N$ | $T$ (sec) | Remarks |
|---|---|---|---|
| Ordinary gradient | – | – | Does not converge in 1000 iterations |
| Fletcher–Reeves | 68 | 14.0 | $\Delta N = 5$ |
| Fletcher–Powell–Davidon | 30 | 6.7 | |
| Memory gradient | 32 | 12.8 | $k = 1$, $\Delta N = 5$ |
| Supermemory gradient | 7 | 5.0 | $k = 3$ |
| Quasilinearization, uncorrected | 25 | 1.6 | |
| Quasilinearization, corrected | 25 | 1.6 | |

the time required to obtain convergence. For the memory and supermemory gradient methods, $k$ denotes the number of past iterations remembered. For the Fletcher–Reeves and memory-gradient methods, $\Delta N$ denotes the number of iterations between two successive restarts.

## 9. Discussion and Conclusions

In this paper, a new accelerated gradient method for finding the minimum of a function $f(x)$ whose variables are unconstrained is investigated. The new algorithm can be stated as follows:

$$\tilde{x} = x + \delta x, \qquad \delta x = -\alpha g(x) + \sum_{i=1}^{k} \beta_i \, \delta x_i \tag{76}$$

where $x$ is an $n$-vector, $g(x)$ is the gradient of the function $f(x)$, $\delta x$ is the change in the position vector for the iteration under consideration, and $\delta x_i$ is the change in the position vector for the $i$th previous iteration. The quantities $\alpha$ and $\beta_i$ are scalars chosen at each step so as to yield the greatest decrease in the function; the scalar $k$ denotes the number of past iterations remembered. For a quadratic function, (76) is identical with the Fletcher–Reeves algorithm.

Two nonquadratic test problems are considered. For these problems, the memory gradient method ($k = 1$) and the supermemory gradient method ($1 < k \leqslant n - 1$) are compared with the Fletcher–Reeves method, the Fletcher–Powell–Davidon method, and quasilinearization.

We note that the Fletcher–Reeves method and the Fletcher–Powell–Davidon method require a one-dimensional search, while the memory-gradient method requires a two-dimensional search and the supermemory gradient method requires a $(k + 1)$-dimensional search. Here, all the searches are performed by quasilinearization with built-in safeguards to force the function $f$ to decrease at every step of the iterative search. While quasilinearization is an essential ingredient of the memory and supermemory gradient methods, it is not indispensable for the Fletcher–Reeves and Fletcher–Powell–Davidon methods. For the last two methods, efficient techniques not requiring second derivatives (for instance, cubic interpolation) are available. Thus, the memory and supermemory gradient methods are hybrid methods in that they employ a combination of first-order and second-order considerations. In particular, the supermemory gradient method supplies a step-by-step transition from the ordinary gradient method (needed in order to start the algorithm) to full-quasilinearization (achieved when $k = n - 1$).

For the three test problems considered in Section 8 and under the given experimental conditions, the following conclusions are reached:

(a) For the Miele problem (Table 2), the shortest computing time is obtained by quasilinearization and the next shortest is obtained by the supermemory gradient method.

(b) For the Wood problem (Table 1), the shortest computing time is obtained by the supermemory gradient method and the next shortest is obtained by quasilinearization. The slight advantage exhibited by the supermemory gradient method is due to the existence of a nonoptimal stationary point for the surface under consideration. Uncorrected quasilinearization leads to this nonoptimal stationary point, while corrected quasilinearization avoids this point at the expense of several iterations in its neighborhood.

(c) On the average, the supermemory gradient method exhibits time characteristics which are between corrected quasilinearization and the gradient methods proper, such as the Fletcher–Reeves method and the Fletcher–Powell–Davidon method.

## References

1. CRAGG, E. E, and LEVY, A. V., *Gradient Methods in Mathematical Programming, Part 3, Supermemory Gradient Method*, Rice University, Aero-Astronautics Report No. 58, 1969.
2. MIELE, A., and CANTRELL, J. W., *Study on a Memory Gradient Method for the Minimization of Functions*, Journal of Optimization Theory and Applications, Vol. 3, No. 6, 1969.
3. FLETCHER, R., and REEVES, C. M., *Function Minimization by Conjugate Gradients*, Computer Journal, Vol. 7, No. 2, 1964.
4. MYERS, G. E., *Properties of the Conjugate Gradient and Davidon Methods*, Journal of Optimization Theory and Applications, Vol. 2, No. 4, 1968.
5. PEARSON, J. D., *On Variable Metric Methods of Minimization*, Research Analysis Corporation, Technical Paper No. RAC-TP-302, 1968.
6. MIELE, A., HUANG, H. Y., and CANTRELL, J. W., *Gradient Methods in Mathematical Programming, Part 1, Review of Previous Techniques*, Rice University, Aero-Astronautics Report No. 55, 1969.