UNCLASSIFIED

AD NUMBER:	AD0856000

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to US Government Agencies and their Contractors; Export Control; 1 Jan 1969. Other requests shall be referred to Air Force Office of Scientific Research, Arlington, VA 22209.

AUTHORITY

Per AFOSR ltr dtd 12 Nov 1971

UNCLASSIFIED

THIS PAGE IS UNCLASSIFIED

AERO-ASTRONAUTICS REPORT NO. 56

GRADIENT METHODS IN MATHEMATICAL PROGRAMMING
PART 2 - MEMORY GRADIENT METHOD

by

A. MIELE AND J.W. CANTRELL

AF-OSR/SRM(A)
arlington, Va. 22209

RICE UNIVERSITY

1969

AUG 6 1969

Gradient Methods in Mathematical Programming
Part 2 - Memory Gradient Method[1]

by

A. MIELE[2] AND J.W. CANTRELL[3]

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

$$\tilde{x} = x + \delta x , \qquad \delta x = - \alpha g(x) + \beta \delta \hat{x}$$

where δx is the change in the position vector x, $g(x)$ is the gradient of the function f(x), and α and β are scalars chosen at each step so as to yield the greatest decrease in the function. The symbol $\delta \hat{x}$ denotes the change in the position vector for the iteration preceeding that under consideration.

It is shown that, for a quadratic function, the present algorithm reduces to the Fletcher-Reeves algorithm; thus, quadratic convergence is assured. However, for a nonquadratic function, initial convergence of the present method is much faster than that of the Fletcher-Reeves method because of the extra degree of freedom available. For a test problem, the number of iterations was about 40-50% that of the Fletcher-Reeves method and the computing time about 60-75% that of the Fletcher-Reeves method, using comparable search techniques.

---

1.   Introduction

In Ref. 1, we considered the problem of minimizing a scalar function $f(x)$ of an n-vector x. We reviewed the ordinary gradient algorithm and the Fletcher-Reeves algorithm of Ref. 2. Compared with the ordinary gradient method, the Fletcher-Reeves algorithm has the advantage of high speed since it produces quadratic convergence. Compared with other conjugate gradient methods (such as the Davidon variable-metric algorithm), it has the advantage of simplicity of concept and small storage requirement while yielding comparable computing time.

In this paper, a generalization of the Fletcher-Reeves algorithm is investigated. This generalization retains the property of quadratic convergence, simplicity of concept, and small storage requirement, while yielding shorter computing time. The only added complication is the need for a two-dimensional search at each iteration as opposed to the one-dimensional search required by the Fletcher-Reeves algorithm.

2.   Definitions

The following definitions are used throughout the paper:

(a) The symbol x denotes the position vector

$$x = \begin{bmatrix} x^1 \\ x^2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x^n \end{bmatrix} \tag{1}$$

whose scalar components are $x^1, x^2, \ldots, x^n$.

(b) The symbol f denotes a scalar function of the vector x, that is,

$$f = f(x) \tag{2}$$

(c) The symbol g denotes the column vector

$$g(x) = \begin{bmatrix} \partial f / \partial x^1 \\ \partial f / \partial x^2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \partial f / \partial x^n \end{bmatrix} \tag{3}$$

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 denotes the square matrix

$$H(x) = \begin{bmatrix} \partial^2 f/\partial x^1 \partial x^1 & \partial^2 f/\partial x^1 \partial x^2 & \ldots\ldots\ldots & \partial^2 f/\partial x^1 \partial x^n \\ \partial^2 f/\partial x^2 \partial x^1 & \partial^2 f/\partial x^2 \partial x^2 & \ldots\ldots\ldots & \partial^2 f/\partial x^2 \partial x^n \\ \ldots & \ldots & \ldots & \ldots \\ \partial^2 f/\partial x^n \partial x^1 & \partial^2 f/\partial x^n \partial x^2 & \ldots\ldots\ldots & \partial^2 f/\partial x^n \partial x^n \end{bmatrix} \qquad (4)$$

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. The symbol $\hat{x}$ denotes the point preceeding x. The symbol $\hat{\hat{x}}$ denotes the point preceeding $\hat{x}$ (see Fig. 1).

(f) The symbol $\delta(\ldots)$ denotes the displacement leading from a point to the next point. Therefore, the following relations hold (see Fig. 1):

$$\tilde{x} = x + \delta x$$

$$x = \hat{x} + \delta\hat{x} \qquad (5)$$

$$\hat{x} = \hat{\hat{x}} + \delta\hat{\hat{x}}$$

(g) 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

$$f = f(x) \tag{6}$$

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

$$f(\tilde{x}) < f(x) \tag{7}$$

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

$$f(\tilde{x}) \cong f(x) + \delta f \tag{8}$$

where the first variation δf is given by

$$\delta f = g^T(x)\delta x \tag{9}$$

with

$$\delta x = \tilde{x} - x \tag{10}$$

Also to first-order terms, the greatest decrease in the value of the function is achieved

if the first variation (9) is minimized. Here, we limit our analysis to those variations

δx which satisfy the constraint

$$K = (\delta x - \beta \delta \hat{x})^T (\delta x - \beta \delta \hat{x}) \tag{11}$$

where $\delta \hat{x} = x - \hat{x}$ and where K and β are prescribed. Since this constraint involves not

only the correction δx of the iteration under consideration but also the correction $\delta \hat{x}$

of the previous iteration, the resulting algorithm is called memory gradient algorithm.

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

$$F = g^T(x)\delta x + (1/2\alpha)(\delta x - \beta\delta\hat{x})^T(\delta x - \beta\delta\hat{x}) \qquad (12)$$

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

$$G(\delta x) = 0 \qquad (13)$$

where G is the gradient of the function F with respect to the scalar variables δx^1, δx^2, ..., δx^n. In the light of (12), the explicit form of (13) is the following:

$$\delta x = -\alpha g(x) + \beta\delta\hat{x}' \qquad (14)$$

Upon substituting (14) into (11), we see that

$$K = \alpha^2 g^T(x)g(x) \qquad (15)$$

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

Remark 4.1. We note that, for

$$\beta = 0 \qquad (16)$$

Eq. (14) reduces to

$$\delta x = -\alpha g(x) \tag{17}$$

which is the algorithm employed in the ordinary gradient method. In this method, the search direction is the direction $-g(x)$ and the stepsize α is determined so that the function $f(x)$ is minimized along the direction $-g(x)$.[4]

We also note that, for

$$\beta = \frac{\alpha g^T(x)g(x)}{\hat{\alpha} g^T(\hat{x})g(\hat{x})} \tag{18}$$

Eq. (14) reduces to

$$\delta x = -\alpha p(x) \tag{19}$$

where

$$p(x) = g(x) + \frac{g^T(x)g(x)}{g^T(\hat{x})g(\hat{x})} p(\hat{x}) \tag{20}$$

which is the algorithm employed in the Fletcher-Reeves method (Ref. 2). In this method, the search direction is the direction $-p(x)$ and the stepsize α is determined so that the function $f(x)$ is minimized along the direction $-p(x)$.

---

[4] To change the stepsize α is the same as to explore the effect of different values of the constant K on the algorithm.

Remark 4.2. Generally speaking, the convergence properties of the Fletcher-Reeves method are better than those of the ordinary gradient method. In particular, for a quadratic function of n scalar variables, (19) converges in no more than n steps to the exact solution, while (17) does not converge in a predetermined number of steps. Since the algorithm (19) was derived in Refs. 2 and 3 from the consideration of quadratic functions, it is not immediately clear that its properties should be excellent for nonquadratic functions as well. For these functions, it is felt that the addition of one extra degree of freedom to the system of corrections δx should substantially improve convergence, at least in the initial stage of the descent process. Therefore, in the subsequent sections of this paper, we employ the algorithm (14) with α and β optimized independently.

5. Properties of the Algorithm

If Eqs. (10) and (14) are combined, the position vector at the end of any iteration becomes

$$\tilde{x} = x - \alpha g(x) + \beta \delta\hat{x} \tag{21}$$

For each point x, Eq. (21) defines a two-parameter family of points $\tilde{x}$ for which the function f takes the form

$$f(\tilde{x}) = f(x - \alpha g(x) + \beta \delta\hat{x}) = F(\alpha, \beta) \tag{22}$$

The greatest decrease in the function $F(\alpha, \beta)$ occurs if the parameters α and β satisfy the following necessary conditions:

$$F_\alpha = 0 , \quad F_\beta = 0 \tag{23}$$

where the subscripts denote partial derivatives. On account of (22), the following relations hold:

$$F_\alpha = - g^T(\tilde{x})g(x) , \quad F_\beta = g^T(\tilde{x})\delta\hat{x} \tag{24}$$

Therefore, Eqs. (23) become

$$g^T(\tilde{x})g(x) = 0 , \quad g^T(\tilde{x})\delta\hat{x} = 0 \tag{25}$$

and show that the gradient $g(\tilde{x})$ is orthogonal to the gradient $g(x)$ and the correction $\delta\hat{x}$.

If Eq. (14) is premultiplied by $g^T(\tilde{x})$, the following result is obtained:

$$g^T(\tilde{x})\delta x = -\alpha g^T(\tilde{x})g(x) + \beta g^T(\tilde{x})\delta\hat{x} \tag{26}$$

which, in the light of (25), implies that

$$g^T(\tilde{x})\delta x = 0 \tag{27}$$

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

If Eq. (14) is premultiplied by $g^T(x)$, the following result is obtained:

$$g^T(x)\delta x = -\alpha g^T(x)g(x) + \beta g^T(x)\delta\hat{x} \tag{28}$$

Because of (9) and (27) applied to the previous iteration, Eq. (28) becomes

$$\delta f = -\alpha g^T(x)g(x) \tag{29}$$

Since $g^T(x)g(x) > 0$, Eq. (29) shows that, if the optimum values of α and β are employed, the first variation is negative for $\alpha > 0$. Equations (25), (27), (29) summarize the general properties of the algorithm. They are valid regardless of the function $f(x)$, as long as it is continuous and has continuous first derivatives.

For any iteration except the first, the complete algorithm can be summarized as follows: (a) for a given nominal point x, the gradient $g(x)$ is known, and the vector $\delta\hat{x}$ is known from the previous iteration; (b) the optimum values of the multipliers α and β must be determined by solving Eqs. (23), as in Section 6; (c) the correction δx to the position vector x is determined using Eq. (14); and (d) the new position vector $\tilde{x}$ is computed through Eq. (10).

Of course, operations (a) through (d) imply that $\delta\hat{x}$ is known from the previous iteration. Since this is not the case for the first iteration, an arbitrary assumption concerning $\delta\hat{x}$ must be made in order to start the algorithm; the simplest assumption is $\delta\hat{x} = 0$, equivalent to stating that the first step is a gradient step.

In the Fletcher-Reeves algorithm, restarting the process every n or n + 1 iterations has proved helpful (Ref. 2). Since the memory gradient algorithm is a modification of the Fletcher-Reeves algorithm, it is likely that restarting every n or n + 1 iterations may prove helpful in this algorithm as well.

6.   Search Technique

The next step is to solve Eqs. (23) for the optimum values of the paramters α and β, that is, those values which yield the minimum of $F(\alpha, \beta)$. Clearly, the memory gradient method requires a two-dimensional search, while the ordinary gradient method and the Fletcher-Reeves method require a one-dimensional search. While several kinds of two-dimensional 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 α and β from arbitrary nominal values α_o and β_o be denoted by

$$\delta\alpha = \alpha - \alpha_o , \qquad \delta\beta = \beta - \beta_o \tag{30}$$

If quasilinearization is applied to Eqs. (23), one obtains the linear algebraic equations

$$F_{\alpha\alpha}\delta\alpha + F_{\alpha\beta}\delta\beta + F_{\alpha} = 0$$

$$F_{\beta\alpha}\delta\alpha + F_{\beta\beta}\delta\beta + F_{\beta} = 0 \tag{31}$$

where $F_{\alpha}, F_{\beta}, F_{\alpha\alpha}, F_{\alpha\beta}, F_{\beta\beta}$ are computed at α_o, β_o. Next, we imbed Eqs. (31) in the more general equations

$$F_{\alpha\alpha}\delta\alpha + F_{\alpha\beta}\delta\beta + \mu\rho F_{\alpha} = 0$$

$$F_{\beta\alpha}\delta\alpha + F_{\beta\beta}\delta\beta + \mu\rho F_{\beta} = 0 \tag{32}$$

where u denotes a scaling factor and ρ a direction factor such that

$$0 \leq u \leq 1 , \qquad \rho = \pm 1 \tag{33}$$

Equations (32) admit the solutions

$$\delta\alpha = - u\rho(D_1/D_3) , \qquad \delta\beta = - u\rho(D_2/D_3) \tag{34}$$

where

$$D_1 = F_\alpha F_{\beta\beta} - F_\beta F_{\alpha\beta}$$

$$D_2 = F_\beta F_{\alpha\alpha} - F_\alpha F_{\alpha\beta} \tag{35}$$

$$D_3 = F_{\alpha\alpha} F_{\beta\beta} - F_{\alpha\beta}^2$$

The direction factor ρ is determined in such a way that the first variation

$$\delta F = F_\alpha \delta\alpha + F_\beta \delta\beta \tag{36}$$

is negative. From (34) and (36), we obtain

$$\delta F = - u\rho(D_4/D_3) \tag{37}$$

where

$$D_4 = F_\alpha^2 F_{\beta\beta} - 2F_\alpha F_\beta F_{\alpha\beta} + F_\beta^2 F_{\alpha\alpha} \tag{38}$$

Therefore, (37) is negative if the direction factor ρ is chosen as follows:

$$\rho = \text{sign} \, (D_4/D_3) \tag{39}$$

Because of this choice, the corrections (34) to α and β become

$$\delta\alpha = - u(D_1/D_3) \, \text{sign} \, (D_4/D_3) \, , \quad \delta\beta = - u(D_2/D_3) \, \text{sign} \, (D_4/D_3) \tag{40}$$

Note that the partial derivatives appearing in D_1, D_2, D_3, D_4 are computed at α_o, β_o. Therefore, they are given by

$$F_\alpha = - g^T(\tilde{x}_o)g(x) \, , \quad F_\beta = g^T(\tilde{x}_o)\delta\hat{x} \tag{41}$$

and

$$F_{\alpha\alpha} = g^T(x)H(\tilde{x}_o)g(x), \quad F_{\alpha\beta} = - g^T(x)H(\tilde{x}_o)\delta\hat{x}, \quad F_{\beta\beta} = \delta\hat{x}^T H(\tilde{x}_o)\delta\hat{x} \tag{42}$$

where $\tilde{x}_o$ denotes the vector

$$\tilde{x}_o = x - \alpha_o g(x) + \beta_o \delta\hat{x} \tag{43}$$

If the matrix $H(x)$ is not explicitly available, the second derivatives cannot be computed with Eqs. (42). In this case, one can use the following difference scheme:

$$F_{\alpha\alpha} = (1/2\epsilon_1)[g^T(\tilde{x}_3) - g^T(\tilde{x}_1)]g(x)$$

$$F_{\alpha\beta} = (1/2\epsilon_1)[g^T(\tilde{x}_1) - g^T(\tilde{x}_3)]\delta\hat{x}$$

$$= (1/2\epsilon_2)[g^T(\tilde{x}_4) - g^T(\tilde{x}_2)]g(x) \tag{44}$$

$$F_{\beta\beta} = (1/2\epsilon_2)[g^T(\tilde{x}_2) - g^T(\tilde{x}_4)]\delta\hat{x}$$

where

$$\tilde{x}_1 = \tilde{x}_0 - \epsilon_1 g(x) , \quad \tilde{x}_2 = \tilde{x}_0 + \epsilon_2 \delta\hat{x}$$

$$\tilde{x}_3 = \tilde{x}_0 + \epsilon_1 g(x) , \quad \tilde{x}_4 = \tilde{x}_0 - \epsilon_2 \delta\hat{x} \qquad (45)$$

In practice, one may choose

$$\epsilon_1 = \epsilon / |g(x)| , \quad \epsilon_2 = \epsilon / |\delta\hat{x}| \qquad (46)$$

where ϵ is a small number.

To perform the search, nominal values must be given to α_0, β_0. Then, one sets $u = 1$, computes $\delta\alpha, \delta\beta$ from Eqs. (40), and α, β from Eqs. (30). If $F(\alpha, \beta) < F(\alpha_0, \beta_0)$, the scaling factor $u = 1$ is acceptable. If $F(\alpha, \beta) > F(\alpha_0, \beta_0)$, the previous value of u must be replaced by some smaller value in the range $0 \le \mu \le 1$ until the condition $F(\alpha, \beta) < F(\alpha_0, \beta_0)$ is met (this can be obtained through bisection, that is, by successively dividing the value of u by 2). At this point, the search step is completed. The values obtained for α, β become the nominal values α_0, β_0 for the next search step, and the procedure is repeated until a desired degree of accuracy on α, β is obtained. In the absence of better information, the first step in the search procedure can be made with $\alpha_0 = \beta_0 = 0$.

Remark 6.1. Whenever the independent multiplier algorithm is started or restarted, the assumption $\delta\hat{x} = 0$ must be made, with the implication that $F_\beta = F_{\alpha\beta} = F_{\beta\beta} = 0$. Note that Eqs. (40) yield undetermined corrections and must be replaced by

$$\delta\alpha = - u(F_\alpha / F_{\alpha\alpha}) \operatorname{sign}(F_{\alpha\alpha}) \qquad (47)$$

Also note that, for the starting or restarting step, the multiplier β remains undetermined and has ro effect on the algorithm, since $\beta\delta\hat{x} = 0$.

7. Relation of the Memory Gradient Method and the Fletcher-Reeves Method

Consider a quadratic function f(x), that is, a function of the type

$$f(x) = a + b^T x + \frac{1}{2} x^T C x \tag{48}$$

where a is a scalar, b is an n-vector, and C an nx n symmetric matrix. For this function, the gradient g(x) is the linear function

$$g(x) = b + Cx \tag{49}$$

and the matrix of the second derivatives is the constant matrix

$$H(x) = C \tag{50}$$

At the point $\tilde{x}$, Eq. (49) becomes

$$g(\tilde{x}) = b + C\tilde{x} = b + C(x + \delta x) = g(x) + C\delta x \tag{51}$$

with the implication that

$$g^T(\tilde{x}) = g^T(x) + \delta x^T C \tag{52}$$

Therefore, the conditions (25) optimizing α and β become

$$g^T(x)g(x) + \delta x^T C g(x) = 0$$

$$g^T(x)\delta \hat{x} + \delta x^T C \delta \hat{x} = 0 \tag{53}$$

After observing that

$$\delta x^T = -\alpha g^T(x) + \beta \delta \hat{x}^T \tag{54}$$

and accounting for (27), we see that Eqs. (53) reduce to

$$g^T(x)g(x) - \alpha g^T(x)Cg(x) + \beta g^T(x)C\delta\hat{x} = 0 \tag{55}$$

$$-\alpha g^T(x)C\delta\hat{x} + \beta \delta\hat{x}^T C\delta\hat{x} = 0$$

and admit the solutions

$$\alpha = \lambda \delta\hat{x}^T C\delta\hat{x}, \quad \beta = \lambda g^T(x)C\delta\hat{x} \tag{56}$$

where λ is a scalar given by

$$\lambda = \frac{g^T(x)g(x)}{[g^T(x)Cg(x)][\delta\hat{x}^T C\delta\hat{x}] - [g^T(x)C\delta\hat{x}]^2} \tag{57}$$

From (56), we deduce the following relation between the optimum parameters α and β:

$$\beta = \alpha \frac{g^T(x)C\delta\hat{x}}{\delta\hat{x}^T C\delta\hat{x}} \tag{58}$$

If Eq. (51) is rewritten for the previous iteration, one obtains the relation

$$g(x) = g(\hat{x}) + C\delta\hat{x} \tag{59}$$

which, upon premultiplication by $g^T(x)$, leads to

$$g^T(x)g(x) = g^T(x)g(\hat{x}) + g^T(x)C\delta\hat{x} \tag{60}$$

Therefore, on account of (25-1), one concludes that

$$g^T(x)C\delta\hat{x} = g^T(x)g(x) \tag{61}$$

Next, if Eq. (59) is premultiplied by $\delta\hat{x}^T$, one obtains

$$\delta\hat{x}^T g(x) = \delta\hat{x}^T g(\hat{x}) + \delta\hat{x}^T C\delta\hat{x} \tag{62}$$

On account of Eq. (27), Eq. (62) reduces to

$$\delta\hat{x}^T C\delta\hat{x} = -\delta\hat{x}^T g(\hat{x}) \tag{63}$$

Owing to the fact that

$$\delta\hat{x} = -\hat{\alpha}g(\hat{x}) + \hat{\beta}\delta\hat{\hat{x}} \tag{64}$$

and that

$$\delta\hat{x}^T = -\hat{\alpha}g^T(\hat{x}) + \hat{\beta}\delta\hat{\hat{x}}^T \tag{65}$$

Eq. (63) becomes

$$\delta\hat{x}^T C\delta\hat{x} = \hat{\alpha}g^T(\hat{x})g(\hat{x}) - \hat{\beta}\delta\hat{\hat{x}}^T g(\hat{x}) \tag{66}$$

We note that, because of (27),

$$\delta\hat{\hat{x}}^T g(\hat{x}) = 0 \tag{67}$$

and, therefore,

$$\delta \hat{x}^T C \delta \hat{x} = \hat{\alpha} g^T(\hat{x}) g(\hat{x}) \tag{68}$$

From (58), (61), (68), we conclude that

$$\beta = \frac{\alpha g^T(x) g(x)}{\hat{\alpha} g^T(\hat{x}) g(\hat{x})} \tag{69}$$

which is identical with (18). Thus, for a quadratic function, the memory gradient algorithm yields values of α and β identical with those of the Fletcher-Reeves algorithm.

8.   Numerical Example

In order to compare the present method with the Fletcher-Reeves method and the ordinary gradient method, a numerical example was carried out. The function to be minimized is the following (Ref. 4):

$$f(x) = 100(z - y^2)^2 + (1 - y)^2 + 90(w - u^2)^2 + (1 - u)^2$$
$$+ 10.1[(z - 1)^2 + (w - 1)^2] + 19.8(z - 1)(w - 1) \tag{70}$$

and exhibits a relative minimum as well as an absolute minimum at the point

$$y = 1, \quad z = 1, \quad u = 1, \quad w = 1 \tag{71}$$

where $f = 0$. This function is a particular case of (6), if one sets

$$x = \begin{bmatrix} x^1 \\ x^2 \\ x^3 \\ x^4 \end{bmatrix} = \begin{bmatrix} y \\ z \\ u \\ w \end{bmatrix} \tag{72}$$

The nominal point chosen for starting the descent process is the point

$$y = -3, \quad z = -1, \quad u = -3, \quad w = -1 \tag{73}$$

where $f = 19,192$. The following convergence criterion was adopted for stopping the descent process:

$$f(x) \leq 10^{-13} \tag{74}$$

and is approximately equivalent to an accuracy of 10^{-7} in each of the coordinates involved.

Concerning a particular iteration, the search for the optimum values of the multipliers

α and β was terminated when the changes of α and β predicted through Eqs. (40) satisfied

the inequalities

$$|\delta\alpha/\alpha_0| \leq 10^{-6}, \quad |\delta\beta/\beta_0| \leq 10^{-6} \tag{75}$$

The first derivatives of the function F with respect to the multipliers α and β were

computed analytically through Eqs. (41). The second derivatives were computed

numerically through Eqs. (44).

Computations were performed in double precision arithmetic using the Rice

University Burroughs B-5500 computer. The memory gradient method, the Fletcher-Reeves

method, and the ordinary gradient method were investigated. For the first two methods,

three variations were considered: (a) no restart, (b) restart every n iterations, and

(c) restart every n + 1 iterations, were n = 4. If ΔN denotes the number of iterations

performed before restarting, case (a) is characterized by $\Delta N = \infty$, case (b) by $\Delta N = 4$,

and case (c) by $\Delta N = 5$. The dependence of the function f on N (the number of iterations)

and t (the computing time)[5] is shown in Figs. 2-7. Also, the number of iterations

necessary to satisfy the inequality $f \leq 10^{-13}$ and the associated computing time are

shown in Tables 1 and 2.

---

[5] The computing time t is defined as the actual time employed doing numerical calculations
in binary form; therefore, t does not include the binary-to-decimal conversion time and
the printout time.

Table 1. Number of iterations to $f(x) \leq 10^{-13}$

	$\Delta N = \infty$	$\Delta N = 4$	$\Delta N = 5$
Memory gradient method	N = 34	N = 17	N = 15
Fletcher-Reeves method		N = 39	N = 29

Table 2. Computing time to $f(x) \leq 10^{-13}$

	$\Delta N = \infty$	$\Delta N = 4$	$\Delta N = 5$
Memory gradient method	t = 17.8 sec	t = 9.2 sec	t = 8.8 sec
Fletcher-Reeves method		t = 14.8 sec	t = 11.9 sec

It should be noted that the ordinary gradient method did not converge to the degree of accuracy $f \leq 10^{-13}$ even after 100 iterations. On the other hand, the memory gradient method converged in every case, while the Fletcher-Reeves method converged only for $\Delta N = 4$ and $\Delta N = 5$. For those cases where the methods converged, it was found that (a) the number of iterations of the memory gradient method is 40-50% of that of the Fletcher-Reeves method and (b) the computing time of the memory gradient method is 60-75% of that of the Fletcher-Reeves method. Clearly, the present algorithm requires more computing time per iteration, owing to the fact that a two-dimensional search

rather than a one-dimensional search, is necessary. Despite this, the overall computing

time is less than in the Fletcher-Reeves method owing to the considerable decrease in

the number of iterations.

Remark 8.1. In order to compute the second derivatives through Eqs. (44), a

specification of the value of ϵ is necessary. If ϵ is too large, the second derivatives

are not accurate because the first derivatives do not behave linearly. If ϵ is too small,

the second derivatives are not accurate because the differences in the first derivatives

cannot be computed with sufficient precision. Therefore, ϵ must be in a proper range.

For the test problem under consideration, convergence to $f \leq 10^{-13}$ using the memory

gradient algorithm was achieved for ϵ in the range $10^{-20} \leq \epsilon \leq 10^{-2}$. In this range, the

number of iterations n and the computing time t for convergence were found to be insensitive

to changes in ϵ. At any rate, the numerical data presented here refer to $\epsilon = 10^{-8}$.

Remark 8.2. The computations presented here were done in double precision

arithmetic. Experiments were made by using various combinations of single precision

and double precision. In one of the experiments, the position vector x and the gradient

g(x) were calculated in double precision while everything else (in particular, the optimum

values of α and β) were calculated in single precision. With this mixed approach, the

independent multiplier method converged in the same number of iterations as with the

double precision approach; however, the computing time decreased. With the same mixed

approach, the Fletcher-Reeves method converged in a greater number of iterations and

greater computing time than with the double precision approach. It appears that the

independent multiplier algorithm has greater flexibility than the Fletcher-Reeves algorithm,

since it is less sensitive to round-off errors.

9.   Discussion and Conclusions

In this paper, a new accelerated gradient method for finding the minimum of a function f(x) is presented, f being a scalar and x an n-vector. The new algorithm, called the memory gradient method, can be stated as follows:

$$\tilde{x} = x + \delta x , \quad \delta x = - \alpha g(x) + \beta \delta \hat{x} \tag{76}$$

where α and β are scalars, chosen at each step so as to yield the greatest decrease in the function f. The above algorithm is a generalization of the Fletcher-Reeves algorithm and its principal objective is to reduce the computing time required for convergence. The following comments are pertinent:

(a) With the memory gradient algorithm, both α and β are optimized. With the Fletcher-Reeves algorithm, only α is optimized since the ratio β/α is kept at a preselected value determined from quadratic considerations.

(b) Because of (a), the memory gradient algorithm requires a two-dimensional search, while the Fletcher-Reeves algorithm requires a one-dimensional search. The two-dimensional search can be performed by using quasilinearization with built-in safeguards to insure the stability of the descent process.

(c) For a quadratic function, the memory gradient algorithm yields values of α and β identical with those of the Fletcher-Reeves algorithm. Therefore, in this case, the number of iterations required for convergence is identical for the two methods. However, owing to the two-dimensional search, the computing time for the memory gradient method is larger than that for the Fletcher-Reeves method.

(d) For a nonquadratic function, the memory gradient algorithm exhibits faster initial convergence than the Fletcher-Reeves algorithm. This is due to the extra degree of freedom available.

(e) The memory gradient algorithm has definite advantages with respect to the Fletcher-Reeves algorithm in compensating for round-off errors. Again, this is due to the extra degree of freedom available.

(f) A test case was considered, that of a quartic involving four variables. The initial coordinates were such that $f = 19,192$. After the first iteration, both methods reduced the function to $f = 134.4$. After four iterations, the memory gradient method reduced the function to $f = 0.0045$ while the Fletcher-Reeves method reduced the function to $f = 31.5$.

(g) For the above test case, both algorithms were continued until convergence was achieved $(f \leq 10^{-13})$. The number of iterations with the memory gradient method was about 40-50% that of the Fletcher-Reeves method; the computing time with the memory gradient method was about 60-75% that of the Fletcher-Reeves method.

References

1. MIELE, A., and CANTRELL, J.W., Gradient Methods in Mathematical Programming,
    Part 1, Review of Previous Methods, Rice University, Aero-Astronautics Report
    No. 55, 1969.

2. FLETCHER, R., and REEVES, C.M., Function Minimization by Conjugate Gradients,
    Computer Journal, Vol. 7, No. 2, 1964.

3. BECKMAN, F.S., The Solution of Linear Equations by the Conjugate Gradient
    Method, Mathematical Methods for Digital Computers, Edited by A. Ralston
    and H.S. Wilf, John Wiley and Sons, New York, New York, 1960.

4. PEARSON, J.D., On Variable Metric Methods of Minimization, Research Analysis
    Corporation, McLean Virginia.

List of Captions

Fig. 1   Nomenclature.

Fig. 2   The function f(N) for $\Delta N = \infty$.
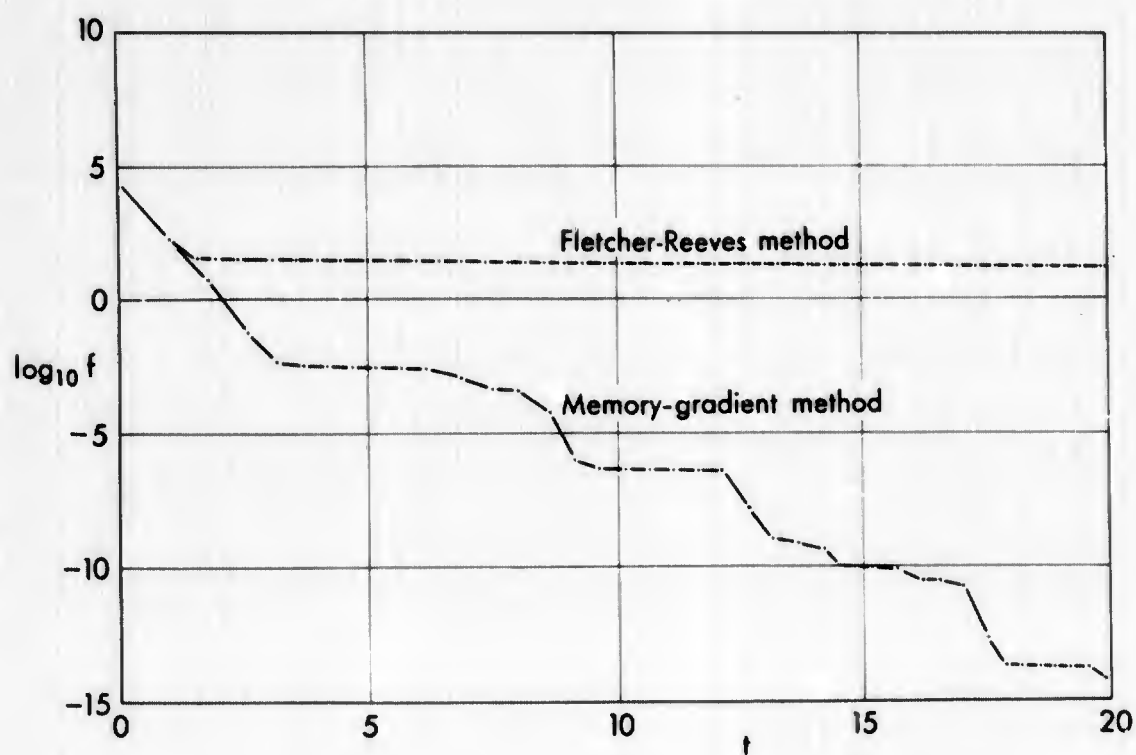
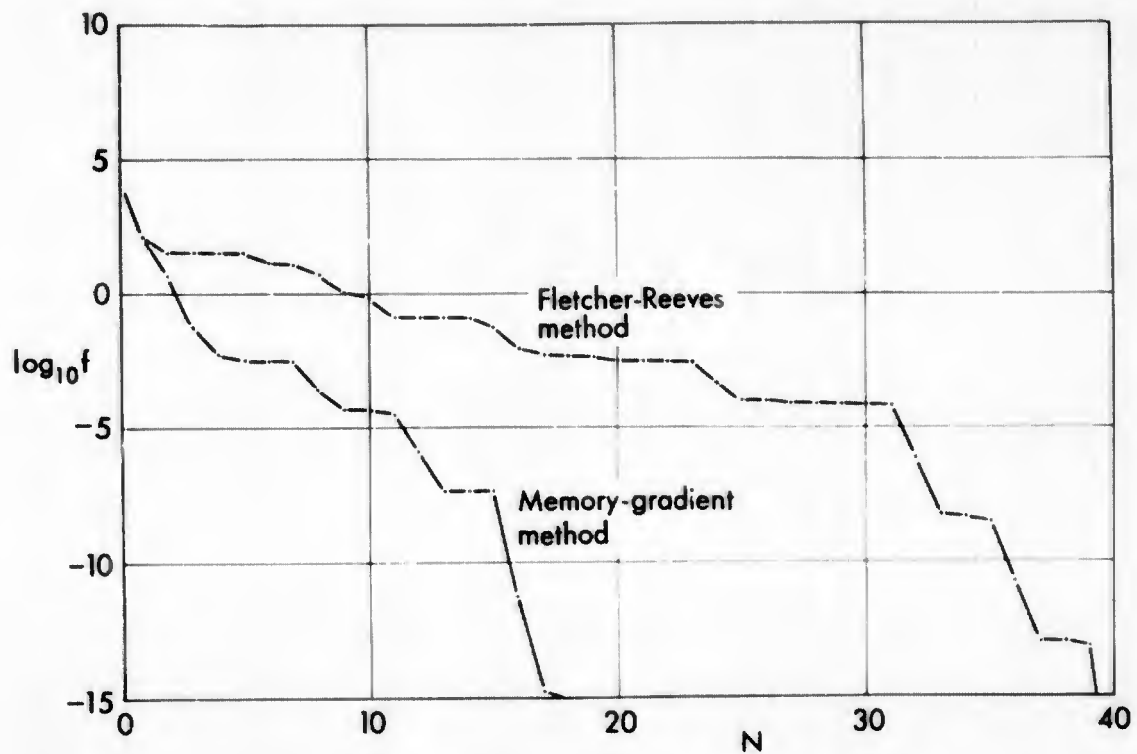Fig. 3   The function f(t) for $\Delta N = \infty$.
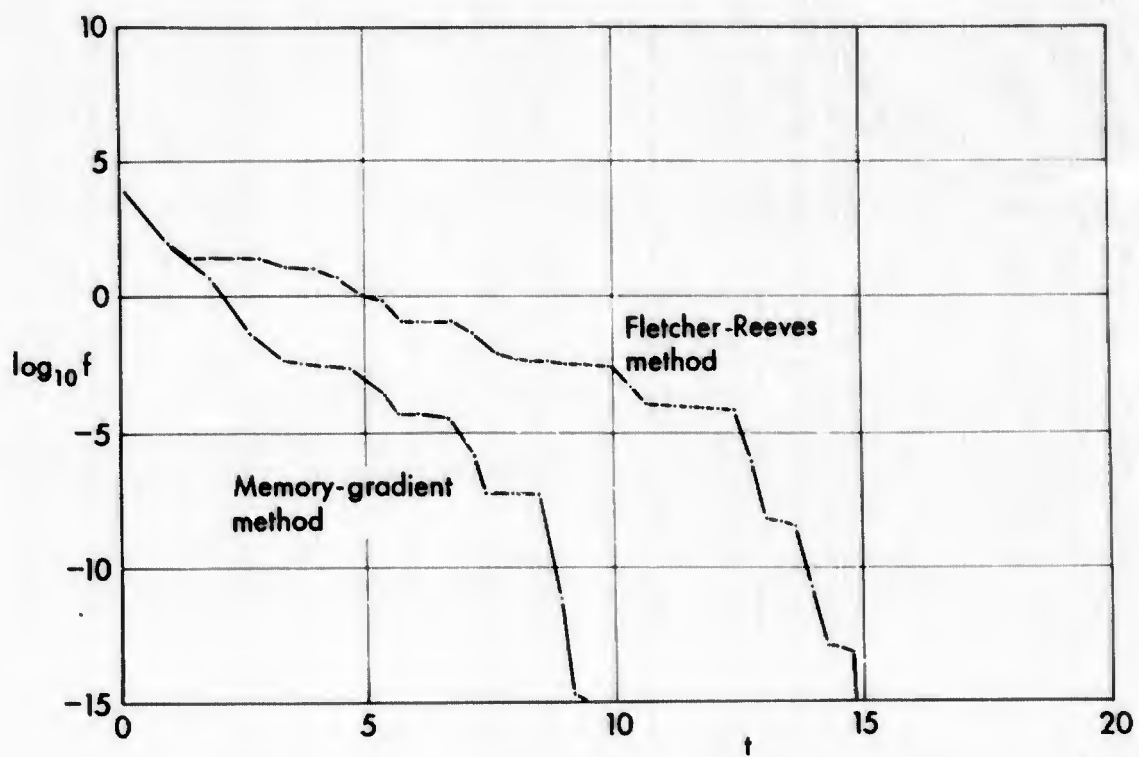
Fig. 4   The function f(N) for $\Delta N = 4$.
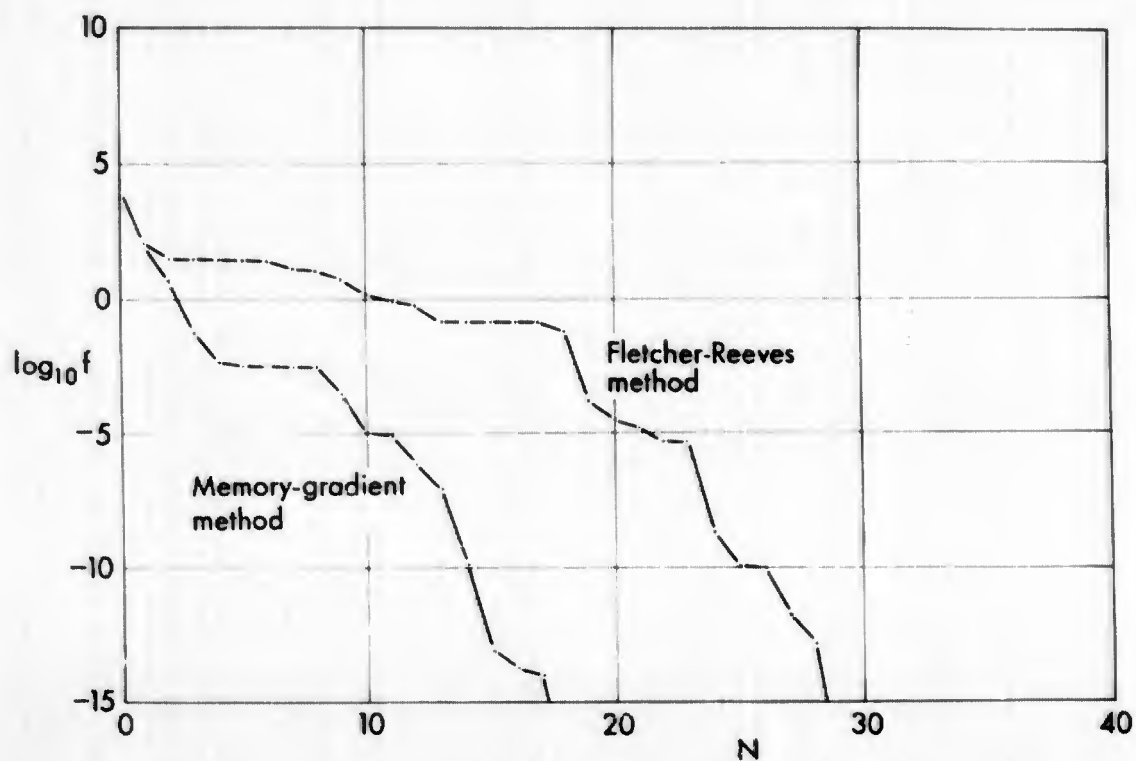

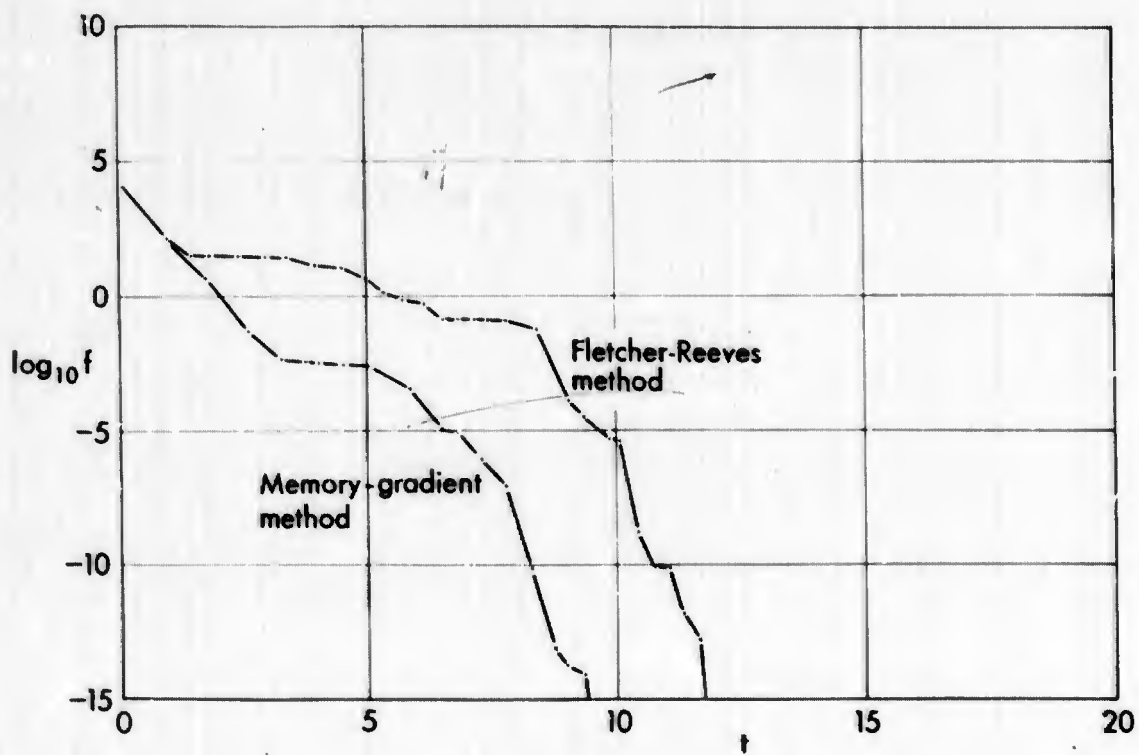Fig. 5   The function f(t) for $\Delta N = 4$.

Fig. 6   The function f(N) for $\Delta N = 5$.

Fig. 7   The function f(t) for $\Delta N = 5$.