JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 3, No. 6, 1969

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

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 investigated. 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 preceding that under consideration.

For a nonquadratic function, initial convergence of the present method is 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. 2, the Fletcher–Reeves conjugate gradient algorithm for finding the minimum of a function $f(x)$ whose variables are unconstrained was

[1] Paper received November 22, 1968. 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 condensed version of the investigation described in Ref. 1. Portions of this paper were presented by the senior author at the International Symposium on Optimization Methods, Nice, France, 1969.

[2] Professor of Astronautics and Director of the Aero-Astronautics Group, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

[3] Graduate Student in Aero-Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas. Presently, Aerodynamics Engineer, LTV Aerospace Corporation, Dallas, Texas.

809/3/6-6

investigated. Compared with the ordinary gradient method, this 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 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 n-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 $n \times n$ symmetric 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. The symbol $\hat{x}$ denotes the point preceding x.

(f) The symbol $\delta(\ldots)$ denotes the displacement leading from a point to the next point. Specifically, $\tilde{x} = x + \delta x$ and $x = \hat{x} + \delta\hat{x}$.

(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

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

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

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

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 δx which satisfy the constraint

$$K = (\delta x - \beta \delta \hat{x})^T (\delta x - \beta \delta \hat{x}) \tag{6}$$

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

$$F = g^T(x)\, \delta x + (1/2\alpha)(\delta x - \beta \delta \hat{x})^T (\delta x - \beta \delta \hat{x}) \tag{7}$$

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 the following:

$$\delta x = -\alpha g(x) + \beta \delta \hat{x} \tag{9}$$

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 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

$$\beta = 0 \tag{11}$$

Eq. (9) reduces to

$$\delta x = -\alpha g(x) \tag{12}$$

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

$$\beta = \frac{\alpha g^T(x)\, g(x)}{\hat{\alpha} g^T(\hat{x})\, g(\hat{x})} \tag{13}$$

Eq. (9) reduces to

$$\delta x = -\alpha p(x) \tag{14}$$

where

$$p(x) = g(x) + \frac{g^T(x)\, g(x)}{g^T(\hat{x})\, g(\hat{x})}\, p(\hat{x}) \tag{15}$$

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 $-p(x)$.

Remark 4.2. Generally speaking, the convergence properties of the Fletcher–Reeves method are better than those of the ordinary gradient

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

method. In particular, for a quadratic function of n scalar variables, (14) converges in no more than n steps to the exact solution, while (12) does not converge in a predetermined number of steps. Since the algorithm (14) was derived in Refs. 2 and 3 from the consideration of quadratic functions, it is not immediately clear that its properties should always be excellent for nonquadratic functions. For these functions, it is possible that the addition of one extra degree of freedom to the system of corrections δx should improve convergence, at least in the initial stage of the descent process. Therefore, in the subsequent sections of this paper, we employ the algorithm (9) with α and β optimized independently.

5. Properties of the Algorithm

If Eqs. (5) and (9) are combined, the position vector at the end of any iteration becomes

$$\tilde{x} = x - \alpha g(x) + \beta \delta \hat{x} \tag{16}$$

For each point x, Eq. (16) defines a two-parameter family of points $\tilde{x}$ for which the function f takes the form

$$f(\tilde{x}) = f(x - \alpha g(x) + \beta \delta \hat{x}) = F(\alpha, \beta) \tag{17}$$

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

$$F_\alpha = 0, \qquad F_\beta = 0 \tag{18}$$

where the subscripts denote partial derivatives. On account of (17), Eqs. (18) become

$$g^T(\tilde{x})\, g(x) = 0, \qquad g^T(\tilde{x})\, \delta \hat{x} = 0 \tag{19}$$

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

If Eq. (9) is premultiplied by $g^T(\tilde{x})$, the following result is obtained:

$$g^T(\tilde{x})\, \delta x = -\alpha g^T(\tilde{x})\, g(x) + \beta g^T(\tilde{x})\, \delta \hat{x} \tag{20}$$

which, in the light of (19), implies that

$$g^T(\tilde{x})\, \delta x = 0 \tag{21}$$

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

If Eq. (9) is premultiplied by $g^T(x)$, the following result is obtained:

$$g^T(x)\,\delta x = -\alpha g^T(x)\,g(x) + \beta g^T(x)\,\delta\hat{x} \tag{22}$$

Because of (4) and (21) applied to the previous iteration, Eq. (22) becomes

$$\delta f(x) = -\alpha g^T(x)\,g(x) \tag{23}$$

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. (18), as in Section 6; (c) the correction δx to the position vector x is determined using Eq. (9); and (d) the new position vector $\tilde{x}$ is computed through Eq. (5).

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, some 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. (18) for the optimum values of the parameters 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 (Ref. 1).

Let $\delta\alpha = \alpha - \alpha_0$ and $\delta\beta = \beta - \beta_0$ denote the corrections to α and β starting from arbitrary nominal values α_0 and β_0. Then, it is shown in Ref. 1 that the following corrections must be employed:

$$\delta\alpha = -\mu(D_1/D_3)\,\mathrm{sign}(D_4/D_3), \qquad \delta\beta = -\mu(D_2/D_3)\,\mathrm{sign}(D_4/D_3) \tag{24}$$

where[5]

$$\begin{aligned} D_1 &= F_\alpha F_{\beta\beta} - F_\beta F_{\alpha\beta} \\ D_2 &= F_\beta F_{\alpha\alpha} - F_\alpha F_{\alpha\beta} \\ D_3 &= F_{\alpha\alpha} F_{\beta\beta} - F_{\alpha\beta}^2 \\ D_4 &= F_\alpha{}^2 F_{\beta\beta} - 2F_\alpha F_\beta F_{\alpha\beta} + F_\beta{}^2 F_{\alpha\alpha} \end{aligned} \tag{25}$$

and where F_α, F_β, $F_{\alpha\alpha}$, $F_{\alpha\beta}$, $F_{\beta\beta}$ are computed at α_0, β_0, that is, at the point $\tilde{x}_0$ defined by

$$\tilde{x}_0 = x - \alpha_0 g(x) + \beta_0 \delta\hat{x} \tag{26}$$

The symbol μ, $0 \leqslant \mu \leqslant 1$, denotes a scaling factor for the variations $\delta\alpha$ and $\delta\beta$.

The partial derivatives appearing in Eqs. (25) can be computed from the expressions

$$F_\alpha = -g^T(\tilde{x}_0)\, g(x), \qquad F_\beta = g^T(\tilde{x}_0)\, \delta\hat{x} \tag{27}$$

and

$$F_{\alpha\alpha} = g^T(x)\, H(\tilde{x}_0)\, g(x), \qquad F_{\alpha\beta} = -g^T(x)\, H(\tilde{x}_0)\, \delta\hat{x}, \qquad F_{\beta\beta} = \delta\hat{x}^T H(\tilde{x}_0)\, \delta\hat{x} \tag{28}$$

If the matrix $H(x)$ is not explicitly available, the second derivatives cannot be computed with Eqs. (28). In this case, one can use the difference schemes

$$\begin{aligned} F_{\alpha\alpha} &= \{g[\tilde{x}_0 + \epsilon_1 g(x)] - g[\tilde{x}_0 - \epsilon_1 g(x)]\}^T g(x)/2\epsilon_1 \\ F_{\beta\beta} &= \{g[\tilde{x}_0 + \epsilon_2\, \delta\hat{x}] - g[\tilde{x}_0 - \epsilon_2\, \delta\hat{x}]\}^T \delta\hat{x}/2\epsilon_2 \end{aligned} \tag{29}$$

and

$$\begin{aligned} F_{\alpha\beta} &= \{g[\tilde{x}_0 - \epsilon_1 g(x)] - g[\tilde{x}_0 + \epsilon_1 g(x)]\}^T \delta\hat{x}/2\epsilon_1 \\ &= \{g[\tilde{x}_0 - \epsilon_2\, \delta\hat{x}] - g[\tilde{x}_0 + \epsilon_2\, \delta\hat{x}]\}^T g(x)/2\epsilon_2 \end{aligned} \tag{30}$$

In practice, one may choose

$$\epsilon_1 = \epsilon/|\, g(x)|, \qquad \epsilon_2 = \epsilon/|\, \delta\hat{x}\,| \tag{31}$$

where ϵ is a small number.

---

[5] The term $\text{sign}(D_4/D_3)$ in Eqs. (24) forces the first variation of the function $F(\alpha, \beta)$ to be negative at every step of the search. Therefore, if μ is sufficiently small, the function $F(\alpha, \beta)$ decreases at every step of the search.

To perform the search, nominal values must be given to α_0, β_0. Then, one sets $\mu = 1$, computes $\delta\alpha, \delta\beta$ from Eqs. (24), and α, β from $\alpha = \alpha_0 + \delta\alpha$, $\beta = \beta_0 + \delta\beta$. If $F(\alpha, \beta) < F(\alpha_0, \beta_0)$, the scaling factor $\mu = 1$ is acceptable. If $F(\alpha, \beta) > F(\alpha_0, \beta_0)$, the previous value of μ must be replaced by some smaller value in the range $0 \leqslant \mu \leqslant 1$ until the condition $F(\alpha, \beta) < F(\alpha_0, \beta_0)$ is met (this can be obtained by successively dividing the value of μ 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 memory gradient 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. (24) yield undetermined corrections and must be replaced by

$$\delta\alpha = -\mu(F_\alpha/F_{\alpha\alpha})\,\mathrm{sign}(F_{\alpha\alpha}) \tag{32}$$

Also note that, for the starting or restarting step, the multiplier β remains undetermined and has no effect on the algorithm, since $\beta\delta\hat{x} = 0$.

7. Numerical Example

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

$$\begin{aligned} f(x) = {} & 100(z - y^2)^2 + (1 - y)^2 + 90(w - u^2)^2 + (1 - u)^2 \\ & + 10.1[(z - 1)^2 + (w - 1)^2] + 19.8(z - 1)(w - 1) \end{aligned} \tag{33}$$

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

$$y = 1, \qquad z = 1, \qquad u = 1, \qquad w = 1 \tag{34}$$

where $f = 0$. This function is a particular case of (1), if one sets

$$x = \begin{bmatrix} x^1 \\ x^2 \\ x^3 \\ x^4 \end{bmatrix} = \begin{bmatrix} y \\ z \\ u \\ w \end{bmatrix} \tag{35}$$

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

$$y = -3, \quad z = -1, \quad u = -3, \quad w = -1 \tag{36}$$

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

$$f(x) \leqslant 10^{-13} \tag{37}$$

This 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. (24) satisfied the inequalities

$$|\delta\alpha| \leqslant 10^{-6}|\alpha_0|, \qquad |\delta\beta| \leqslant 10^{-6}|\beta_0| \tag{38}$$

The first derivatives of the function F with respect to the multipliers α and β were computed analytically through Eqs. (27). The second derivatives were computed numerically through Eqs. (29)–(31).

Computations were performed in double precision FORTRAN IV 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, where $n = 4$. 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 $\Delta N = 5$. In this connection, the number of iterations N necessary to satisfy the inequality $f \leqslant 10^{-13}$ and the associated computing time T are given in Tables 1 and 2. (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.)

It should be noted that the ordinary gradient method did not converge to the degree of accuracy $f \leqslant 10^{-13}$ even after 100 iterations. On the other hand, the memory gradient method converged in every case, while the

Table 1. Total number of iterations

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

Table 2. Computing time

Method	$\Delta N = \infty$	$\Delta N = 4$	$\Delta N = 5$
Memory gradient	$T = 17.8$ sec	$T = 9.2$ sec	$T = 8.8$ sec
Fletcher–Reeves		$T = 14.8$ sec	$T = 11.9$ sec

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 7.1. In order to compute the second derivatives through Eqs. (29)–(31), 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 $f \leqslant 10^{-13}$ using the memory gradient algorithm was achieved for ϵ in the range $10^{-20} \leqslant \epsilon \leqslant 10^{-2}$. 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 7.2. The computations presented here were done in double precision FORTRAN IV. 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 memory gradient 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.

8. Discussion and Conclusions

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

$$\tilde{x} = x + \delta x, \qquad \delta x = -\alpha g(x) + \beta \delta \hat{x} \tag{39}$$

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 (Refs. 5–6). Therefore, in this case, the number of iterations required for convergence is identical for the two methods.

(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) 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 $f = 134.2$. After four iterations, the memory gradient method reduced the function to $f = 0.0044$ while the Fletcher–Reeves method reduced the function to $f = 31.5$.

(f) For the above test case, both algorithms were continued until convergence was achieved ($f \leqslant 10^{-13}$). 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 2, Memory Gradient Method*, Rice University, Aero-Astronautics Report No. 56, 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, 1960.
4. PEARSON, J. D., *On Variable Metric Methods of Minimization*, Research Analysis Corporation, Technical Paper No. RAC-TP-302, 1968.
5. CANTRELL, J. W., *Method of Independent Multipliers for Minimizing Unconstrained Functions*, Rice University, M. S. Thesis, 1969.
6. CANTRELL, J. W., *On the Relation between the Memory Gradient Method and the Fletcher–Reeves Method*, Journal of Optimization Theory and Applications, Vol. 4, No. 1, 1969.