IMA Journal of Numerical Analysis (1988) **8**, 141–148

Two-Point Step Size Gradient Methods

JONATHAN BARZILAI
School of Business Administration, Dalhousie University, Halifax, N.S., Canada

AND

JONATHAN M. BORWEIN
Department of Mathematics, Statistics and Computing Science, Dalhousie University, Halifax, N.S., Canada

[Received 24 September 1986 and in revised form 14 April 1987]

We derive two-point step sizes for the steepest-descent method by approximating the secant equation. At the cost of storage of an extra iterate and gradient, these algorithms achieve better performance and cheaper computation than the classical steepest-descent method. We indicate a convergence analysis of the method in the two-dimensional quadratic case. The behaviour is highly remarkable and the analysis entirely nonstandard.

1. Introduction

THE classical steepest-descent method (see Cauchy [4]) for the unconstrained minimization of $f: \mathbb{R}^n \to \mathbb{R}$ is of the form

$$x_{k+1} = x_k + \alpha_k d_k. \tag{1}$$

The search direction $d_k \in \mathbb{R}^n$ is chosen as the negative gradient of f at x_k:

$$d_k = -\nabla f(x_k), \tag{2}$$

and the step size α_k is given by

$$\alpha_k = \arg\min_{\alpha} f(x_k + \alpha d_k). \tag{3}$$

We will denote $g_k = g(x_k) = \nabla f(x_k)$.

Despite its simplicity and the optimal property (3), the steepest-descent method performs poorly. It converges linearly and is badly affected by ill-conditioning (see Akaike [1]). Nevertheless, the understanding of its behaviour is fundamental to the theory and design of optimization algorithms (see Luenberger [7]).

In this paper, we propose two new step sizes for use in conjunction with the negative gradient direction (2). These step sizes require less computational effort than (3), and the resulting algorithms seem to be less sensitive to ill-conditioning. More important however, from a theoretical point of view, is the marked difference between the behaviour of these algorithms and steepest descent. We derive these step sizes in Section 2. A numerical example typical of the behaviour

of these algorithms is given in Section 3, and their rate of convergence in the two-dimensional case is analysed in Section 4. The analysis is of theoretical interest, because the unusual behaviour of the new algorithms requires the use of a nonstandard approach. Finally, in Section 5 we comment on implications of our results.

2. Derivation of step sizes

In this paper we study the iteration $x_{k+1} = x_k - S_k g_k$, where S_k has the form $S_k = \alpha_k I$, and α_k minimizes $\|\Delta x - \alpha \Delta g\|^2$, with $\Delta x = x_k - x_{k-1}$ and $\Delta g = g_k - g_{k-1}$. The motivation for this choice is that it provides a two-point approximation to the secant equation underlying quasi-Newton methods (see Dennis & Moré [5] for a general discussion of quasi-Newton methods and Barzilai & Ben-Tal [3] for a comparison of one-point vs. two-point algorithms). This yields the iteration

$$x_{k+1} = x_k - \alpha_k g_k, \tag{4}$$

where α_k is given by

$$\alpha_k = \langle \Delta x, \Delta g \rangle / \langle \Delta g, \Delta g \rangle, \tag{5}$$

and $\langle a, b \rangle$ denotes the scalar product of the vectors a and b.

By symmetry, we may minimize $\|\alpha \Delta x - \Delta g\|^2$ with respect to α. The corresponding step size turns out to be

$$\alpha_k = \langle \Delta x, \Delta x \rangle / \langle \Delta x, \Delta g \rangle. \tag{6}$$

Note that, in the one-dimensional case, the algorithm defined by (4) together with either (5) or (6) is the secant method. Note also that these algorithms are applicable to the solution of the equation $g(x) = 0$, where g: $\mathbb{R}^n \to \mathbb{R}^n$, with a Jacobian that is not necessarily symmetric.

3. Numerical example

To compare the behaviour of the classical steepest-descent method with the two methods obtained by using the step sizes (5) and (6), we minimize the function $f(x) = \frac{1}{2}\langle x, Ax \rangle - \langle b, x \rangle$, where

$$A = \begin{bmatrix} 20 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad b = (1, 1, 1, 1).$$

Note that, even when f is a quadratic, i.e. $\nabla^2 f = A$ is constant, the two-point step sizes (5) and (6) are cheaper to compute than the steepest-descent step size

$$\alpha_k = \langle g_k, g_k \rangle / \langle g_k, A g_k \rangle. \tag{7}$$

Errors for the steepest-descent and the two-point algorithms are listed in Tables 1–3. the tables list the (Euclidean) norm of the gradient g_k, the step size

TABLE 1
Errors for the steepest-descent algorithm

k	$\|g_k\|$	α_k	G_k
1	2·000000000 E 000	1·212121212 E −001	8·549127640 E −001
2	1·849229855 E 000	7·963901542 E −002	5·189012029 E −001
3	1·332088978 E 000	1·127377746 E −001	1·059794639 E −000
4	1·371336685 E 000	8·020781752 E −002	5·407043781 E −001
5	1·008379568 E 000	1·133463025 E −001	1·084311605 E 000
6	1·050028508 E 000	8·049623326 E −002	5·499767190 E −001
7	7·787055015 E −001	1·138600987 E −001	1·103991388 E 000
8	8·181936146 E −001	8·072316633 E −002	5·573162281 E −001
9	6·108111137 E −001	1·142513869 E −001	1·119140047 E 000
10	6·461735081 E −001	8·089015460 E −002	5·627445328 E −001
11	4.847354600 E −001	1·145320525 E −001	1·130094914 E 000
12	5·153025020 E −001	8·100693746 E −002	5·665549695 E −001
⋮	⋮	⋮	⋮
171	2·984608494 E −008	1·150846953 E −001	1·151888863 E 000
172	3·203263464 E −008	8·122983744 E −002	5·738608549 E −001
173	2·426587526 E −008	1·150846953 E −001	1·151888860 E 000
174	2·604361399 E −008	8·122983752 E −002	5·738608549 E −001
175	1·972897626 E −008	1·150846952 E −001	1·151888861 E 000
176	2·117433792 E −008	8·122983720 E −002	5·738608570 E −001
177	1·604032417 E −008	1·150846936 E −001	1·151888838 E 000
178	1·721545185 E −008	8·122983804 E −002	5·738608682 E −001
179	1·304132540 E −008	1·150846924 E −001	1·151888821 E 000
180	1·399674380 E −008	8·122983788 E −002	5·738608743 E −001
181	1·060303808 E −008	1·150846926 E −001	1·151888816 E 000
182	1·137982548 E −008	8·122983846 E −002	5·738608822 E −001
183	8·620628156 E −009	1·150846933 E −001	1·151888825 E 000

α_k, and the quantity $G_k = \|g_{k+1}\|^2/\|g_k\|^2$ used to analyse the convergence rate in Section 4. All algorithms are started at the origin, i.e. $x_1 = (0, 0, 0, 0)$ for the one-point step size (7), and $x_1 = x_2 = (0, 0, 0, 0)$ for the two-point step sizes (5) and (6) (with step size $\alpha_2 = 0/0 = 1$.) The algorithms were coded in APL* PLUS/PC on an IBM Personal Computer with an 8087 coprocessor. We stopped when $\|g_k(x)\| < 10^{-8}$.

This example is typical of many we ran: the behaviour of the two-point algorithms is very similar, and both are significantly faster than the classical one-point algotithm. The analysis of the next section suggests that this indeed is the case in general.

4. Analysis of convergence rate

In two dimensions, we can analyse explicitly the positive definite quadratic case, and we now do so. To study the quadratic case, with

$$f(x) = \tfrac{1}{2}x^{\mathsf{T}}Qx - b^{\mathsf{T}}x + c,$$

TABLE 2
Errors for the two-point algorithm with step (5)

k	$\|g_k\|$	α_k	G_k
2	2·000000000 E 000	1·000000000 E 000	1·107500000 E 002
3	2·104755618 E 001	6·534653465 E −002	1·004316268 E −001
4	6·670173211 E 000	5·266747102 E −002	6·475154033 E −002
5	1·697313884 E 000	5·342022916 E −002	3·317050549 E −001
6	9·775482639 E −001	9·626310100 E −002	3·303201353 E −001
7	5·618310441 E −001	1·154512281 E −001	5·917097096 E −001
8	4·321754377 E −001	4.347699330 E −001	2·296744619 E −001
9	2·071173278 E −001	3·807080298 E −001	4·037369346 E 001
10	1·316029653 E 000	5·041534722 E −002	3·506376805 E −004
11	2·464307889 E −002	5·001051865 E −002	3·351910989 E −001
12	1·426728071 E −002	7·389820331 E −002	4·498428297 E −001
13	9·569111327 E −003	1·209389230 E −001	5·409783352 E −001
14	7·038199027 E −003	2·482426364 E −001	2·635500606 E −001
15	3·613209171 E −003	5.541214228 E −001	5·473142473 E −001
16	2·673077071 E −003	2·883826457 E −001	3·505893771 E 000
17	5·005078082 E −003	1·001678954 E −001	2·763391604 E −003
18	2·631068008 E −004	9·956182394 E −002	9·221708673 E −001
19	2·526607454 E −004	5·096258259 E −002	9·796871398 E −002
20	7·908269786 E −005	5·061897578 E −002	8·050877670 E −001
21	7·095828174 E −005	3·874271444 E −001	5·071951483 E −002
22	1·598051002 E −005	4·999713616 E −001	4·607139320 E −002
23	3·430096264 E −006	4·757638507 E −001	7·245794986 E 001
24	2·919774622 E −005	5·001113713 E −002	4·340580021 E −005
25	1·923637403 E −007	5·000216906 E −002	2·496926497 E −001
26	9·612272894 E −008	9·977221465 E −002	5·278130229 E −006
27	2·208341036 E −010	1·000000022 E −001	1·085445105 E −002

we first note that, using (5),

$$x_{k+1} = x_k - \sigma_k(Qx_k - b), \qquad Qx_k - b = g_k,$$

$$\sigma_k = \frac{\langle g_k - g_{k-1}, x_k - x_{k-1}\rangle}{\langle g_k - g_{k-1}, g_k - g_{k-1}\rangle} = \frac{\langle g_{k-1}, Qg_{k-1}\rangle}{\langle Qg_{k-1}, Qg_{k-1}\rangle}, \qquad Qx_{k+1} = Qx_k - \sigma_k Qg_k.$$

Thus $g_{k+1} = g_k - \sigma_k Qg_k$.

We can assume that Q is of the form $Q = \begin{bmatrix} 1 & 0 \\ 0 & \lambda \end{bmatrix}$ with $\lambda \geq 1$, since, if $y_k = Ug_k$ with U orthogonal, then

$$\langle y_{k+1}, y_{k+1}\rangle / \langle y_k, y_k\rangle = \langle g_{k+1}, g_{k+1}\rangle / \langle g_k, g_k\rangle.$$

To simplify the notation in the analysis that follows, we parametrize $g_k = \alpha_k(\lambda^{m_k}, \pm 1)$ with $\alpha_k > 0$. (The analysis shows that the first term is positive.) Using this parametrization, we get

$$g_{k+1} = \alpha_k\big(\lambda^{m_k}(1 - \sigma_k), \pm(1 - \lambda\sigma_k)\big), \qquad \sigma_k = (\lambda^{2m_{k-1}} + \lambda)/(\lambda^{2m_{k-1}} + \lambda^2).$$

Hence

$$g_{k+1} = \frac{\alpha_k}{\lambda^{2m_{k-1}} + \lambda^2}\big((\lambda^2 - \lambda)\lambda^{m_k}, \pm\lambda^{2m_{k-1}}(1 - \lambda)\big)$$

$$= (\lambda - 1)\frac{\alpha_k \lambda^{2m_{k-1}}}{\lambda^{2m_{k-1}} + \lambda^2}(\lambda^{1+m_k-2m_{k-1}}, \mp 1).$$

TABLE 3
Errors for the two-point algorithm with step (6)

k	$\|g_k\|$	α_k	G_k
2	2·000000000 E 000	1·000000000 E 000	1·107500000 E 002
3	2·104756518 E 001	1·212121212 E −001	1·662516816 E 000
4	2·713844044 E 001	5·515438247 E −002	1·217824662 E −002
5	2·994865127 E 000	5·015928785 E −002	6·130648239 E −002
6	7·415329742 E −001	5·473128024 E −002	5·981954047 E −001
7	5·735245384 E −001	2·149779845 E −001	4·380743717 E −001
8	7·395997585 E −001	3·439341351 E −001	2·102868447 E 000
9	5·504678760 E −001	2·109907996 E −001	1·212563677 E 000
10	6·061557888 E −001	1·024061516 E −001	1·412557431 E −002
11	7·204225765 E −002	9·992090956 E −002	8·226279372 E −001
12	6·534149118 E −002	7·792830276 E −002	4·291597069 E −001
13	4·280539524 E −002	6·786426828 E −002	4·284160804 E −001
14	2·801786298 E −002	8·882203072 E −002	6·650185209 E −001
15	2·284800386 E −002	2·101416069 E −001	1·714600888 E 000
16	2·991780903 E −002	2·221805587 E −001	9·909179687 E 000
17	9·417777814 E −002	5·895504423 E −002	3·599990937 E −002
18	1·786895454 E −002	5·023775240 E −002	9·198078913 E −002
19	5·419356357 E −003	5·569706724 E −002	7·894511705 E −001
20	4·815155825 E −003	4·990214595 E −001	2·927974897 E −004
21	8·239370279 E −005	4·999838767 E −001	7·993467323 E 001
22	7·366507283 E −004	5·059567565 E −002	1·419457683 E −004
23	8·776530117 E −006	5·000000143 E −002	2·463096211 E −005
24	4·355755920 E −008	5·000246318 E −002	2·499932052 E −001
25	2·177848363 E −008	1·000025482 E −001	6·604282526 E −005
26	1·769866299 E −010	1·000082555 E −001	6·399668062 E −001

If we start with

$$g_0 = (\lambda, 1), \qquad g_1 = \lambda \frac{\lambda - 1}{\lambda + 1}(\lambda^{-1}, -1)$$

(for comparison with steepest descent), this leads to

$$g_k = \alpha_k(\lambda^{m_k}, (-1)^k),$$

$$m_0 = 1, \qquad m_1 = -1, \qquad m_{k+1} = 1 + m_k - 2m_{k-1},$$

$$\alpha_0 = 1, \qquad \alpha_1 = \lambda \frac{\lambda - 1}{\lambda + 1}, \qquad \alpha_{k+1} = (\lambda - 1)\frac{\lambda^{2m_{k-1}}}{\lambda^{2m_{k-1}} + \lambda^2}\alpha_k.$$

Note that the general solution of the recurrence relation for m_k is given by (see [6])

$$m_k = \tfrac{1}{2} + 2^{\frac{1}{2}k}\theta \cos(\phi + k \arctan \sqrt{7}) \quad (k = 0, 1, 2, \ldots)$$

for some constant scalars θ and ϕ. When k is large, at least one of any four consecutive values of m_k must be large and negative. Hence the expression for α_{k+1} implies $|\alpha_{k+1}| \ll |\alpha_k|$ at least once every four iterations, while $|\alpha_{k+1}| < (\lambda - 1)|\alpha_k|$ on every iteration. To analyse the behaviour of the error terms, we

define $G_k = \|g_{k+1}\|^2/\|g_k\|^2$ and $\Delta_k = m_k - m_{k+1}$. Note that, in the notation above,

$$\Delta_k = 2^{\frac{1}{2}k+\frac{1}{2}}\theta \cos[\phi + (k-1)\arctan\sqrt{7}] \quad (k = 0, 1, 2, \ldots).$$

Now

$$\begin{aligned} G_k &= \left((\lambda-1)\frac{\lambda^{2m_{k-1}}}{\lambda^{2m_{k-1}}+\lambda^2}\right)^2 \frac{\lambda^{2m_{k+1}}+1}{\lambda^{2m_k}+1} \\ &= (\lambda-1)^2 \frac{\lambda^{2\Delta_k}}{(\lambda^{\Delta_k}+\lambda)^2} \frac{\lambda^{2m_{k+1}}+1}{\lambda^{2m_k}+1} \\ &= \frac{(\lambda-1)^2}{(\lambda^{\Delta_k}+\lambda)^2} \frac{\lambda^{2m_k}+\lambda^{2\Delta_k}}{\lambda^{2m_k}+1}. \end{aligned}$$

Using Baker's work (see [2]) on the transcendence of logarithms, we can prove that $|m_k|$ tends to infinity. We take this result as given, although the analysis could be largely carried out without it. This implies that $|\Delta_k|$ also tends to infinity, since $\Delta_k = 2m_{k-1} - 1$. With this in mind, we now break up the analysis to three cases, as follows.

CASE (i) $\Delta_k \leqslant 0$.

$$\begin{aligned} G_k &= \left(\frac{\lambda-1}{\lambda}\right)^2 \frac{\lambda^{2m_k}+\lambda^{2\Delta_k}}{(1+\lambda^{\Delta_k-1})^2(\lambda^{2m_k}+1)} \\ &< \left(\frac{\lambda-1}{\lambda}\right)^2 \frac{\lambda^{2m_k}+\lambda^{2\Delta_k}}{\lambda^{2m_k}+1} \leqslant \left(\frac{\lambda-1}{\lambda}\right)^2. \end{aligned}$$

CASE (ii) $\Delta_k > 0$ and $m_k \leqslant 0$.

$$G_k = (\lambda-1)^2 \frac{\lambda^{2\Delta_k}+\lambda^{2m_k}}{(\lambda+\lambda^{\Delta_k})^2(1+\lambda^{2m_k})} \to (\lambda-1)^2 \qquad \text{as } |m_k| \to \infty \text{ and } |\Delta_k| \to \infty.$$

CASE (iii) $\Delta_k > 0$ and $m_k > 0$.

$$G_k \leqslant (\lambda-1)^2(\lambda^{2m_{k+1}}+1)/\lambda^{2m_k} \leqslant (\lambda-1)^2(\lambda^{-2\Delta_k}+\lambda^{-2m_k}) \to 0.$$

We see that, in case (ii), G_k may increae (cf. the examples above). This will not happen if $\lambda < 2$, but in general we do not have monotonicity. Since the sign of m_k behaves like that of $\cos(k \arctan\sqrt{7})$, we may expect increases every 5–6 steps. In fact, Δ_k is also governed by a Fibonacci-like recurrence relation, and can be solved analytically, yielding

$$m_k = \tfrac{1}{2} + 2^{\frac{1}{2}k+\frac{1}{2}} \cos(k \arctan\sqrt{7}), \qquad \Delta_k = 2^{\frac{1}{2}k+1} \cos(k \arctan\sqrt{7}).$$

From this one can establish that, asymptotically,

$$\|g_k\|^2 \leqslant [(\lambda-1)^2/\lambda]^k \lambda^{-2^{\frac{1}{2}k-1}} \leqslant C\lambda^{-2^{\frac{1}{2}k}}$$

so that the asymptotic convergence rate of $\|g_k\|$ is $\sqrt{2}$.

This analysis rested critically on beginning with a steepest-descent step. Indeed, as Ya-xiang Yuan has pointed out to us, $m_k = \frac{1}{2}$ for all k satisfies the relation

$m_{k+1} = 1 + m_k - 2m_{k-1}$ and leads to the gradient sequences

$$g_k = \left(\frac{\lambda - 1}{\lambda + 1}\right)^k \begin{bmatrix} \lambda^{\frac{1}{2}} \\ (-1)^k \end{bmatrix}$$

and

$$g_k = \left(\frac{\lambda - 1}{\lambda + 1}\right)^k \begin{bmatrix} 1 \\ (-1)^k \end{bmatrix}$$

for (5) and (6) respectively. These show that an only linear convergence rate may be obtained if m_k does not tend to infinity. However, these solutions are numerically unstable, and one might argue that they will not show up in practice.

5. Concluding remarks

The classical steepest-descent method was proposed by Cauchy [4] in 1847, and its first partial analysis is due to Forsythe & Motzkin [8] in 1951. But the full analysis of this algorithm by Akaike (in 1959) is quite nonstandard. In view of the highly remarkable behaviour of the new algorithms and the fact that they are not monotone, it is probably not surprising that our analysis, too, is entirely nonstandard.

A better understanding of the behaviour of the new algorithms is important for a better understanding of the general theory of optimization algorithms, and may lead to a better design of new algorithms. The analysis of the new algorithms (and the example of Section 3) clearly suggests that the latter constitute a substantial improvement over the classical steepest-descent method. In general, it seems that the behaviour of these algorithms depends on the clustering of the eigenvalues, not just on the condition number. In the two-dimensional case we have shown that, surprisingly, as λ increases (ill conditioning), the convergence rate improves. It follows that the performance of the steepest-descent method cannot be attributed solely to the choice of the search direction. Indeed, Akaike's analysis of the behaviour of the steepest-descent method (see Akaike [1]) depends heavily on the step size being give by (4). Clearly, the behaviour of any algorithm depends on the choice of a step size no less than it depends on the choice of a search direction.

Finally, note that, since the two-point algorithms are not descent algorithms, they have an advantage in that the restriction to descent algorithms often results in small step sizes for ill-conditioned problems. This may seem, however, undesirable since it is difficult to control nonmonotone algorithms. It is possible to overcome this difficulty without losing the benefit of the more general nondecreasing two-point algorithms, by using the following simple scheme. If for some iteration the norm of the gradient is not decreased, then compute the point obtained by using steepest descent as well as the point obtained by performing N iterations of the two-point algorithm for some fixed number N, starting in both cases from the last point computed. Of the two points obtained this way, choose the one with the lower value for the gradient norm as the next point. Our experience shows (compare Section 3 above) that the number of steps in which

this extra computation is needed is small, and that a small value of N (say 2 or 3) is a reasonable value to use.

Acknowledgements

We would like to thank M. J. D. Powell and Ya-xiang Yuan for their helpful comments on this paper. This research was supported in part by NSERC Canada grants, Numbers A-8802 and A-5116.

References

[1] Akaike, H. 1959 On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Ann. Inst. Statist. Math. Tokyo* **11,** 1–17.
[2] Baker, A. 1975 *Transcendental Number Theory*. Cambridge University Press.
[3] Barzilai, J., & Ben-Tal, A. 1982 Nonpolynomial and inverse interpolation for line search: synthesis and convergence rates. *SIAM J. Numer. Ana.* **19,** 1263–1277.
[4] Cauchy, A. 1847 Méthode générale pour la résolution des systèms d'équations simultanées. *Comp. Rend. Sci. Paris* **25,** 536–538.
[5] Dennis, J. E., & Moré, J. 1977 Quasi-Newton methods, motivation and theory. *SIAM Rev.* **19,** 46–89.
[6] Hardy, G. H., & Wright, E. M. 1971 *An Introduction to the Theory of Numbers*. Oxford: Oxford University Press.
[7] Luenberger, D. G. 1973 *Introduction to Linear and Nonlinear Programming*. Reading, Mass: Addison-Wesley.
[8] Forsythe, G. E., & Motzkin, T. S. 1951 Asymptotic properties of the optimum gradient method. *Bull. Am. Math. Soc.* **57,** 183.