JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 85, No. 3, pp. 677–704, JUNE 1995

Efficient Line Search Algorithm for Unconstrained Optimization[1]

F. A. POTRA[2] AND Y. SHI[3]

Abstract. A new line search algorithm for smooth unconstrained optimization is presented that requires only one gradient evaluation with an inaccurate line search and at most two gradient evaluations with an accurate line search. It terminates in finitely many operations and shares the same theoretical properties as the standard line search rules like the Armijo–Goldstein–Wolfe–Powell rules. This algorithm is especially appropriate for the situation when gradient evaluations are very expensive relative to function evaluations.

Key Words. Unconstrained optimization, line search algorithms.

1. Introduction

A general descent method for solving the unconstrained optimization problem

$$\min f(x), \qquad x \in R^n, \tag{1}$$

where f is twice continuously differentiable and bounded below, can be described as follows.

Algorithm A1.

Step 1. Choose a starting point $x_1 \in R^n$.

Step 2. For $k = 1, 2, \ldots$, execute the computations below.

Step 2a. If $\nabla f(x_k) = 0$, then stop.

[1]The authors would like to thank Margaret Wright and Jorge Moré for valuable comments on earlier versions of this paper.

[2]Professor, Department of Mathematics, University of Iowa, Iowa City, Iowa.

[3]Assistant Professor, Department of Mathematics and Computer Science, Bloomsburg University, Bloomsburg, Pennsylvania.

Step 2b. Determine a search direction s_k such that

$$\nabla f(x_k)^t s_k < 0.$$

Step 2c. Determine a steplength $\alpha_k > 0$.

Step 2d. Set $x_{k+1} = x_k + \alpha_k s_k$.

Here, as throughout this paper, v^t denotes the transpose of the column vector v. Different ways of selecting s_k in Step 2b yield different methods as described, for example, in Ref. 1 or Ref. 2.

The steplength α_k in Step 2c is determined by either an exact line search or an inexact line search. We concentrate on the inexact line search. This topic has been studied and discussed by many authors such as Goldstein (Refs. 3–4), Armijo (Ref. 5), Wolfe (Ref. 6), Powell (Ref. 7), Dennis and Schnabel (Ref. 1), Luenberger (Ref. 8), Fletcher (Ref. 2), Boggs and Schnabel (Ref. 9), Gill *et al.* (Ref. 10), and so on. In general, most line search procedure may generate an enclosing interval $[a, b]$ such that an acceptable steplength α_k lies in $[a, b]$. Then, by generating and testing a new trial point in the current enclosing interval, either an acceptable steplength is obtained or the current enclosing interval is shrunk. There are various rules for accepting a steplength. If we use $f(\alpha)$ to denote $f(x_k + \alpha s_k)$, then we have

$$f'(\alpha) = \nabla f(x_k + \alpha s_k)^t s_k.$$

One of the most popular rules for accepting a steplength α_k, based on the work of Armijo (Ref. 5), Goldstein (Ref. 4), Wolfe (Ref. 6), and Powell (Ref. 7), is given by

$$f(\alpha) \leq f(0) + \alpha\rho f'(0), \tag{2}$$

$$f'(\alpha) \geq \sigma f'(0), \tag{3}$$

where $\rho \in (0, 1/2)$ and $\sigma \in (\rho, 1)$ are two fixed parameters. The latter inequality is sometimes replaced (cf. Ref. 2) with the more stringent one

$$|f'(\alpha)| \leq \sigma |f'(0)|, \qquad \sigma \in (\rho, 1). \tag{4}$$

Both (2), (3) and (2), (4) have good convergence results and are widely used in practice. The convergence is based on the fact that the above rules ensure a sufficient decrease of f for the steplength α_k in the sense that

$$f_k - f_{k+1} \geq \Phi(-g_k^t s_k / \|s_k\|), \qquad \forall k,$$

where $\Phi(t)$ is a forcing function; i.e. for any sequence $\{t_k\} \subset [0, \infty)$, the condition $\lim_{k\to\infty} \Phi(t_k) = 0$ implies that $\lim_{k\to\infty} t_k = 0$. The following lemmas, taken from Ref. 11 and Ref. 10, are used in the proofs of Theorem 1.1 as well as Theorem 3.4.

Lemma 1.1. Let $\sigma \in [0, 1)$, and let $\hat{\alpha}_k$ be the smallest positive number satisfying

$$f'(\hat{\alpha}_k) = \sigma f'(0).$$

Then, there is a forcing function $\Phi(t)$ such that

$$\hat{\alpha}_k \|s_k\| \geq \Phi(-g_k^t s_k / \|s_k\|), \qquad \forall k.$$

Lemma 1.2. Let $\rho \in (0, 1/2)$, and let $\Phi(t)$ be a forcing function. If α_k satisfies (2) and

$$\alpha_k \|s_k\| \geq \Phi(-g_k^t(s_k / \|s_k\|), \qquad \forall k,$$

then f incurs a sufficient decrease for the steplength α_k.

The following theorem shows that the steplength α_k satisfying either (2), (3) or (2), (4) yields a sufficient decrease in f.

Theorem 1.1. Consider a general descent method given by Algorithm A1. For each k, assume that α_k satisfies the line search rules (2), (3) or (2), (4). Then, f incurs a sufficient decrease for the steplength α_k.

Proof. Let $\hat{\alpha}_k$ be as defined in Lemma 1.1. Then α_k, satisfying either (2), (3) or (2), (4), should satisfy $\alpha_k \geq \hat{\alpha}_k$. Hence,

$$\alpha_k \|s_k\| \geq \hat{\alpha}_k \|s_k\|.$$

The result is then implied by Lemma 1.1 and Lemma 1.2. □

One disadvantage of the two sets of rules (2), (3) and (2), (4) is that they require extra gradient evaluations in the line search, especially when the line search is relatively accurate [for example, when $(\sigma, \rho) = (0.1, 0.05)$]. In order to reduce the number of extra gradient evaluations, one has to use a very loose line search by taking (σ, ρ) to be for example $(0.9, 0.001)$, which according to the folklore seems to give best overall results. However, this may cause a significant increase in the number of iterations. In the case of one-dimensional minimization, Brent (Ref. 12) gave an algorithm for enclosing a local minimum of f in an interval $[a, b]$ without evaluating derivatives. Safeguarded quadratic interpolation is used in his algorithm to generate new trial points, and an approximation of a local minimum is accepted when the diameter d of the current enclosing interval satisfies

$$d \leq 2(\epsilon |x| + t), \tag{5}$$

where ϵ is the square root of the relative machine precision and t is an user-supplied positive number. Although the idea of safeguarded quadratic interpolation has been widely used for generating new trial points in the general line search procedure, Brent's criterion (5) is not widely used, since on the one hand many function evaluations may be required for (5) to be satisfied and on the other hand (5) does not guarantee a sufficient decrease of f in general. However, the idea of terminating the search when the enclosing interval is sufficiently small is desirable and will be employed in our main algorithm.

In 1982, Gill *et al.* (Ref. 10) used the divided difference to approximate the first derivative and proved that the steplength α_k yields a sufficient decrease in f if α_k satisfies (2) as well as

$$f(\alpha_k) \geq f(a_k) + (\alpha_k - a_k)\sigma f'(0), \tag{6}$$

for some a_k such that $0 \leq a_k < \alpha_k$. We note that only the derivative $f'(0)$ is used in the formulation of rules (2) and (6). However (2) and (6) may not be satisfied even if the current enclosing interval is very small. In the present paper, we construct a line search criterion that is satisfied whenever the current enclosing interval is small enough, and at the same time guarantees a sufficient decrease of f. Our line search algorithm requires no extra gradient evaluation if $\sigma > 0.5$ and at most one gradient evaluation in addition to $g_k = \nabla f(x_k)$ otherwise. It produces an acceptable α_k within finitely many operations. The steplength α_k satisfies (2) for all k, is close to satisfying (3) in the sense described at the end of next section, and yields a sufficient decrease in f. The new algorithm preserves the good convergence results of (2), (3) and is practically comparable with either (2), (3) or (2), (4). By employing a relatively accurate line search with our new algorithm, we may reduce both the number of iterations as well as the number of gradient evaluations.

To summarize, our algorithm employs an acceptance criterion that is similar to (2)–(3), so that all the convergence properties will be preserved, does not require extra gradient values, induces a sufficient decrease in f, and terminates the line search when the enclosing interval is sufficiently small. The basic ideas can be described as follows. We note that, with the rules (2), (3), only the values $f(0)$ and $f'(0)$ are needed to reject a steplength α if (2) is not satisfied. If (2) is satisfied, then one has to compute

$$f'(\alpha) = g(x_k + \alpha s_k)^t s_k.$$

Then, α is accepted if (3) is satisfied and $g(x_k + \alpha s_k)$ will be used to calculate $f'(0)$ at the next step, or α is rejected if (3) is not satisfied and the value of $g(x_k + \alpha s_k)$ is no longer of use in the algorithm. The main point of our algorithm is making good use of all gradient and function evaluations once they are computed. To this effect, at each step we consider a point $a \geq 0$ that satisfies (2) (initially a is set to be zero), and we accept the steplength

$\alpha > a$ if

$$f(\alpha) \le f(a) + (\alpha - a)\rho f'(0), \tag{7}$$

$$f(\alpha) \ge f(a) + (\alpha - a)\sigma f'(0). \tag{8}$$

Note that, for $a = 0$, the above conditions reduce to the Goldstein conditions. Also, if (7) is replaced by (2) [i.e., $a = 0$ in (7)], then the above conditions reduce to the rules of Gill *et al.* (Ref. 10). If either (7) or (8) is not satisfied, then we use the computed value of $f(\alpha)$ to determine new values for the point a and the steplength α. At the same time, unlike in the procedures of Goldstein or Gill *et al.* we also use the currently computed value of $f(\alpha)$ to determine a new criterion for acceptability. The new criterion requires the satisfaction of either an inequality similar to (3), but where $f'(\alpha)$ is approximated by the divided difference

$$f[a, \alpha] = (f(\alpha) - f(a))/(\alpha - a),$$

or another inequality that uses an approximation of the second-order derivative to guarantee that the current enclosing interval is small enough so that f will incur a sufficient decrease and that α is very close to satisfying (3). While the idea of approximating the second-order derivative has been widely used with quadratic interpolation for generating new trial points in the line search, to our knowledge this is the first time that this approximation is used for establishing an acceptance criterion which has all the above-mentioned advantages. The algorithm is rather complicated to describe because it takes into account all possible cases. However, the extra work needed for coding it seems to pay off, because it makes use of all information available at a given time. In the next section, after giving a complete description of the algorithm, we will add more comments on its geometrical interpretation.

2. Algorithm and Basic Properties

Throughout this paper, we assume that f in (1) is twice continuously differentiable and bounded below, and consider the general Algorithm A1. Our line search algorithm includes six user-given parameters ρ, σ, J, τ_1, τ_2, τ_3, such that

$$\rho \in (0, 1/2), \qquad \sigma \in (\rho, 1), \qquad J \in [2, 9],$$

$$0 < \tau_1 < \tau_2 \le 1/2, \qquad \tau_3 > 2.$$

Algorithm A2. Given $f(0) = f(x_k)$, and $f'(0) = g_k^t s_k < 0$.

Step 1. Check $\alpha = 1$.

Step 1a. If $f(1) > f(0) + \rho f'(0)$, then set $a = 0$, $b = 1$, go to Step 3; else, go to Step 1b.

Step 1b. If $\sigma > 1/2$, go to Step 1c; else, go to Step 1d.

Step 1c. If $f(1) \geq f(0) + \sigma f'(0)$, then set $\alpha_k = 1$, terminate; else, go to Step 2.

Step 1d. Evaluate $f'(1) = g(x_k + s_k)^t s_k$. If $f'(1) \geq \sigma f'(0)$, then set $\alpha_k = 1$, terminate; else, go to Step 2.

Step 2. At this step, note that we have $f(1) \leq f(0) + \rho f'(0)$ and either $f(1) < f(0) + \sigma f'(0)$, or $f'(1) < \sigma f'(0)$. Set $a_1 = 1$, $b_1 = J$; for $n = 1, 2, \ldots$, do the computations below.

Step 2a. If $f(b_n) > f(a_n) + (b_n - a_n)\rho f'(0)$, then set $a = a_n$, $b = b_n$, go to Step 3; else, go to Step 2b.

Step 2b. If $f(b_n) \geq f(a_n) + (b_n - a_n)\sigma f'(0)$, then set $\alpha_k = b_n$, terminate; else, go to Step 2c.

Step 2c. Set $a_{n+1} = b_n$, $b_{n+1} = Jb_n$, go to Step 2a.

Step 3. At this step, note that we have an interval $[a, b]$ with $a \geq 0$ such that $f(a) \leq f(0) + a\rho f'(0)$ (see Lemma 2.1) as well as $f(b) > f(a) + (b - a)\rho f'(0)$. Set $a_1 = a$, $b_1 = b$; for $n = 1, 2, \ldots$, do the computations below.

Step 3a. Take c_n in $[a_n + \tau_1(b_n - a_n), a_n + \tau_2(b_n - a_n)]$.

Step 3b. If

$$f(c_n) \leq f(a_n) + (c_n - a_n)\rho f'(0),$$

$$f(c_n) \geq f(a_n) + (c_n - a_n)\sigma f'(0),$$

then set $\alpha_k = c_n$, terminate; else, go to Step 3c.

Step 3c. Set

$$\begin{aligned}\Delta_n &= |f[a_n, c_n, b_n]| \\ &= |[(f(b_n) - f(c_n))/(b_n - c_n) \\ &\qquad - (f(c_n) - f(a_n))/(c_n - a_n)]/(b_n - a_n)|.\end{aligned}$$

Step 3d. If $f(c_n) \leq f(a_n) + (c_n - a_n)\rho f'(0)$ and if $(\rho - \sigma)f'(0) \geq \tau_3 (b_n - a_n)\Delta_n$, then $\alpha_k = c_n$ and terminate; else, set $a_{n+1} = c_n$, $b_{n+1} = b_n$, go to Step 3a; else, go to Step 3e.

Step 3e. If $(\rho - \sigma)f'(0) \geq \tau_3(b_n - a_n)\Delta_n$ and $a_n > 0$, then set $\alpha_k = a_n$ and terminate; else, set $a_{n+1} = a_n$, $b_{n+1} = c_n$, go to Step 3a.

We note that the steplength α accepted by the above algorithm must satisfy (2). In fact, if $a \geq 0$ satisfies (2) and if $\alpha > a$, then

$$f(\alpha) \leq f(a) + (\alpha - a)\rho f'(0)$$

implies

$$f(\alpha) \leq f(0) + \alpha\rho f'(0).$$

Therefore, if $a \geq 0$ satisfies (2), then we accept a steplength $\alpha > a$ provided that (7) and (8) are satisfied. Steps 1c, 2b, 3b of Algorithm A2 terminate with this condition. In order to reduce the number of function evaluations that are necessary for obtaining a steplength satisfying (7) and (8), we also accept a steplength that lies together with the current steplength value in a sufficiently small interval $[a, b]$, as in Steps 3d and 3e. Our criterion for sufficiently small is based on condition (3) and on approximating $f''(\alpha)/2$ by

$$f[a, \alpha, b] = [(f(b) - f(\alpha))/(b - \alpha) - (f(\alpha) - f(a))/(\alpha - a)]/(b - a),$$

as described at the end of this section. In this case, if $f(\alpha) \leq f(a) + (\alpha - a)\rho f'(0)$ then we accept α as α_k as in Step 3d; otherwise, we accept a as α_k as in Step 3e. Step 1 enforces that $\alpha = 1$ is always tried first, and Steps 1b–1d guarantee that $\alpha_k = 1$ will be asymptotically acceptable, as stated in Lemma 2.2. Hence Step 1 of Algorithm A2 is not theoretically important and can be modified in practice. Theorem 3.2 in the next section tells that asymptotically our α_k satisfies (3), as desired. Theorem 3.4 shows that f will always incur a sufficient decrease with our steplength α_k.

The following lemmas describe some basic properties of Algorithm A2.

Lemma 2.1.

(i) In Steps 2–2c, $a_n \geq 1$ and $f(a_n) \leq f(0) + a_n\rho f'(0)$.

(ii) At Step 3, $a \geq 0$ and $f(a) \leq f(0) + a\rho f'(0)$.

(iii) In Steps 3a–3e, $a_n \geq 0, f(a_n) \leq f(0) + a_n\rho f'(0)$, and $f(b_n) > f(a_n) + (b_n - a_n)\rho f'(0)$.

(iv) The steplength α_k obtained from Algorithm A2 always satisfies (2).

The proof of Lemma 2.1 is straightforward and therefore is omitted.

Lemma 2.2. Let x_* be a local minimizer of f such that $G_* = G(x_*)$ is positive definite. Assume that the search directions in Algorithm A1 are such that the sequence $\{x_k\}$ generated by Algorithm A1 with $\alpha_k = 1$ for all

k converges superlinearly to x_* provided x_1 is sufficiently close to x_*. Then, the steplength $\alpha_k = 1$ will become acceptable for Algorithm A2 for all k sufficiently large.

The above lemma applies clearly for the cases where the search directions are provided by the Newton method or BFGS method. The proof of Lemma 2.2 follows immediately by using Lemma 2.5.3 of Ref. 2 and Steps 1c–1d of Algorithm A2. This also explains why, in Algorithm A2, we check $\alpha = 1$ in different ways with respect to different values of σ. Also notice that, with Algorithm A2, the only case where we may have an extra gradient evaluation (in Step 1d) is $\sigma \leq 1/2$ and $f'(1) < \sigma f'(0)$. Practically, if $\sigma \leq 1/2$, we can use Step 1c when k is small, and then switch to Step 1d when k is large enough. This gives practical advantages while the theory still holds.

Lemma 2.3. Only finitely many operations will be spent in Step 2 of Algorithm A2.

Proof. Otherwise, we have a sequence $\{b_n\}$ such that $b_1 = J$, $b_{n+1} = Jb_n$, $n = 1, 2, \ldots$, and

$$f(b_{n+1}) < f(b_n) + (b_{n+1} - b_n)\sigma f'(0),$$

for all n. By induction, we get

$$f(b_{n+1}) < f(0) + b_{n+1}\rho f'(0), \qquad \forall n.$$

This contradicts the fact that $f(x)$ is bounded below. □

Lemma 2.4. At Step 3 of Algorithm A2, the interval $[a, b]$ includes an $\bar{\alpha}$ which satisfies (2)–(3).

Proof. We suppose that $f'(a) < \sigma f'(0)$, since otherwise $\bar{\alpha} = a$. Then for some $\epsilon \in (0, b - a)$,

$$\begin{aligned} f(\alpha) &< f(a) + (\alpha - a)\sigma f'(0) \\ &< f(a) + (\alpha - a)\rho f'(0), \qquad \forall \alpha \in (a, a + \epsilon). \end{aligned} \tag{9}$$

Let

$$\mu = \min\{\alpha \in [a + \epsilon, b]; f(\alpha) \geq f(a) + (\alpha - a)\rho f'(0)\}.$$

Then, there is an $\bar{\alpha} \in (a, \mu)$ such that

$$f'(\bar{\alpha}) = (f(\mu) - f(a))/(\mu - a) \geq \rho f'(0) > \sigma f'(0).$$

From the definition of μ and (9), we deduce that

$$f(\bar{\alpha}) < f(0) + \bar{\alpha}\rho f'(0).$$

Hence, $\bar{\alpha} \in (a, \mu) \subseteq [a, b]$ and (2)–(3) are satisfied for $\bar{\alpha}$. □

Lemma 2.5. Only finitely many operations will be spent in Step 3 of Algorithm A2.

Proof. Otherwise, we have a sequence of intervals $\{[a_n, b_n]\}_{n=1}^{\infty}$ such that

$$b_n - a_n \leq (1-\tau_1)^{n-1}(b_1 - a_1) \to 0, \qquad \text{when } n \to \infty,$$

$$f(b_n) > f(a_n) + (b_n - a_n)\rho f'(0),$$

$$0 \leq a = a_1 \leq \cdots \leq a_n \leq \cdots \leq b_n \leq \cdots \leq b_1 = b.$$

If $a_n = 0$, for all n, then $b_n \to 0$ and

$$f(b_n) > f(0) + b_n \rho f'(0).$$

This implies that there is a sequence $\{\xi_n\}$ such that $\xi_n \to 0$ and $f'(\xi_n) > \rho f'(0)$, which is a contradiction. Hence, there must be an N such that

$$a_n > 0, \qquad \text{for all } n > N.$$

But this implies that, when $n > N$,

$$\begin{aligned} 0 &< (\rho - \sigma) f'(0) \\ &< \tau_3 (b_n - a_n)\Delta_n \\ &\leq (1/2)\tau_3 (b_n - a_n)\Phi \to 0, \qquad n \to \infty, \end{aligned}$$

where $\Phi = \max_{a \leq \alpha \leq b} |f''(\alpha)|$, also a contradiction. □

Let us note that the condition

$$(\rho - \sigma) f'(0) \geq \tau_3 (b_n - a_n)\Delta_n \tag{10}$$

is an approximation of (3). In fact, if α_k is accepted under (10) at either Step 3d or 3e with $f'(\alpha_k) < \sigma f'(0)$, then the inequality

$$f(b_n) > f(\alpha_k) + (b_n - \alpha_k)\rho f'(0)$$

will imply that there is a $t_n \in (\alpha_k, b_n) \subseteq [a_n, b_n]$ such that

$$f'(t_n) > \rho f'(0) > \sigma f'(0) > f'(\alpha_k).$$

Hence, there is a $v_n \in (\alpha_k, t_n) \subseteq [a_n, b_n]$ such that

$$(\rho - \sigma) f'(0) < f''(v_n)(t_n - \alpha_k) \leq f''(v_n)(b_n - a_n). \tag{11}$$

Now if $(b_n - a_n)\|s_k\|_2$ is very small, which asymptotically always happens, then $f''(v_n)$ is very close to $2\Delta_n$. Since $\tau_3 > 2$, a contradiction would occur between (10) and (11). Therefore, if α_k is accepted under (10), then it is very likely that α_k satisfies (3). Actually in the next section, we show that under certain conditions the steplength α_k obtained from Algorithm A2 will satisfy (3) asymptotically. It will also be shown in the next section that, with our α_k, whether it satisfies (3) or not, a sufficient decrease in f is always produced.

3. Convergence Theorems and Asymptotic Properties

In this section, we prove some convergence theorems and asymptotic properties of Algorithm A2. Throughout this section, we are considering Algorithm A1 where the steplength α_k in Step 2c is determined by Algorithm A2. Before stating our results, it is convenient to introduce some notations. We write $g(x)$ for $\nabla f(x)$, g_k for $g(x_k)$, and $G(x)$ for the Hessian $\nabla^2 f(x)$. We denote by D the level set

$$D = \{x; f(x) \leq f(x_1)\}, \tag{12}$$

where x_1 is the starting point in Algorithm A1. Some of the results are proved under the assumption that g is Lipschitz continuous on D; i.e.,

$$\|g(x) - g(y)\|_2 \leq \lambda \|x - y\|_2, \qquad \forall x, y \in D, \tag{13}$$

for some $\lambda > 0$. Note that, if D is bounded, then (13) is clearly satisfied with

$$\lambda = \max_{x \in D} \|G(x)\|_2.$$

However, we do not assume the boundedness of D. Throughout this paper, we will consider sequences generated by the general Algorithm A1. This algorithm terminates if $g_k = 0$. In our analysis, we will be interested only in the case $g_k \neq 0$ for all k, and this will be implicitly assumed in what follows. We denote

$$\cos\theta_k = -g_k^t s_k / (\|g_k\|_2 \|s_k\|_2).$$

Lemma 3.1. Let $\{x_n\}$ be generated by Algorithm A1 where α_k is given by Algorithm A2. Then for all k, there is an $\bar{\alpha}_k$ satisfying (2), (3) and

$$0 < \bar{\alpha}_k \leq M\alpha_k, \tag{14}$$

where

$$M=\max\{J, 1/\tau_1\}>1. \tag{15}$$

Moreover, if (13) is satisfied, then

$$\alpha_k \geq g_k^t s_k(\sigma-1)/(M\lambda\|s_k\|_2^2). \tag{16}$$

Proof. For each k, let

$$\bar{\alpha}_k=\min\{\alpha>0; f'(\alpha)\geq\sigma f'(0)\}. \tag{17}$$

Due to the continuous differentiability and the boundedness assumptions on f, $\bar{\alpha}_k$ in (17) exists and

$$f'(\bar{\alpha}_k)=\sigma f'(0). \tag{18}$$

Furthermore, it is clear that, for any $\alpha \in [0, \bar{\alpha}_k]$,

$$f(\alpha)\leq f(0)+\alpha\rho f'(0), \tag{19}$$

because otherwise the mean-value theorem would imply that there is a number $t\in(0, \alpha)\subseteq(0, \bar{\alpha}_k)$ such that

$$f'(t)=(f(\alpha)-f(0))/\alpha>\rho f'(0)>\sigma f'(0), \tag{20}$$

which contradicts the definition of $\bar{\alpha}_k$. We note that (18) and (19) show that $\bar{\alpha}_k$ satisfies (2)–(3). If (13) is satisfied, then

$$\begin{aligned}
0&<(\sigma-1)f'(0)\\
&=f'(\bar{\alpha}_k)-f'(0)\\
&=|f'(\bar{\alpha}_k)-f'(0)|\\
&=|(g(x_k+\bar{\alpha}_k s_k)-g_k)^t s_k|\\
&\leq\|g(x_k+\bar{\alpha}_k s_k)-g_k\|_2\|s_k\|_2\\
&\leq\lambda\bar{\alpha}_k\|s_k\|_2^2,
\end{aligned}$$

which shows that

$$\bar{\alpha}_k\geq g_k^t s_k(\sigma-1)/(\lambda\|s_k\|_2^2). \tag{21}$$

Now, we consider three possibilities:

(i) if α_k is accepted at either Step 1c or 1d or 2b or 3b, then there is a number $t_k\in(0, \alpha_k]$ such that $f'(t_k)\geq\sigma f'(0)$. Hence,

$$\alpha_k\geq t_k\geq\bar{\alpha}_k,$$

which implies (14);

(ii) if α_k is accepted at Step 3d, then there is an interval $[a_n, b_n]$ such that $a_n \geq 0, f(b_n) > f(a_n) + (b_n - a_n)\rho f'(0)$, and

$$\alpha_k \in [a_n + \tau_1(b_n - a_n), a_n + \tau_2(b_n - a_n)].$$

This implies that $\alpha_k \geq \tau_1 b_n$. As in (i), we see that $b_n \geq \bar{\alpha}_k$, and hence (14) is true;

(iii) if α_k is accepted at Step 3e, since $\alpha_k > 0$, then α_k is either equal to c_i for some $i < n$, or equal to a_1 with $a_1 \geq 1$. In the first case, the proof in (ii) applies, while in the second case we only need to notice that $\alpha_k = a_1 \geq 1, b_1 = Ja_1 = J\alpha_k$, and

$$f(b_1) > f(a_1) + (b_1 - a_1)\rho f'(0),$$

which implies that $b_1 \geq \bar{\alpha}_k$. Therefore, (14) holds.

Finally, if (13) is satisfied, then by combining (14) and (21), we obtain (16). $\square$

In the next theorem, we show that our line search algorithm enjoys the same theoretical properties as the standard line search procedures; see Ref. 2, pp. 30–32 or Ref. 1, p. 121.

Theorem 3.1. Consider a general descent method given by Algorithm A1. For each k, assume that α_k is obtained from Algorithm A2.

(i) If $g(x)$ is uniformly continuous on D and if $\cos \theta_k \geq \epsilon$ for some $\epsilon > 0$ and all k, then $\lim_{k\to\infty} g_k = 0$.
(ii) If (13) is satisfied, then $\lim_{k\to\infty} g_k^t s_k / \|s_k\|_2 = 0$.
(iii) If $\cos \theta_k \geq \epsilon$ for some $\epsilon > 0$ and all k, then $\liminf_{k\to\infty} \|g_k\|_2 = 0$.

Proof. We first note that the boundedness of f implies that

$$f(x_k) - f(x_{k+1}) \to 0.$$

(i) Lemma 2.1 and the assumptions in (i) imply that

$$f(x_k) - f(x_{k+1}) \geq \rho\epsilon \|g_k\|_2 \|x_{k+1} - x_k\|_2. \tag{22}$$

Take $\bar{\alpha}_k$ as obtained in Lemma 3.1; then, $f'(\bar{\alpha}_k) \geq \sigma f'(0)$ implies that

$$\begin{aligned}-g_k^t s_k &\leq [(g(x_k + \bar{\alpha}_k s_k) - g_k)^t s_k]/(1-\sigma)\\ &\leq [\|g(x_k + \bar{\alpha}_k s_k) - g_k\|_2 \|s_k\|_2]/(1-\sigma).\end{aligned} \tag{23}$$

If the conclusion is not true, then there would be an $\eta > 0$ and a subsequence $\{g_{k_i}\}$ such that $\|g_{k_i}\|_2 \geq \eta$ for all k_i. This and (22) imply that

$$\|x_{k_i+1} - x_{k_i}\|_2 \to 0.$$

Hence,

$$\begin{aligned}\|x_{k_i} + \bar{\alpha}_{k_i} s_{k_i} - x_{k_i}\|_2 &= \bar{\alpha}_{k_i}\|s_{k_i}\|_2\\ &\leq M\alpha_{k_i}\|s_{k_i}\|_2\\ &= M\,\|x_{k_i+1} - x_{k_i}\|_2 \to 0.\end{aligned}$$

Since both x_{k_i} and $x_{k_i} + \bar{\alpha}_{k_i} s_{k_i}$ are in the level set D, we have

$$\|g(x_{k_i} + \bar{\alpha}_{k_i} s_{k_i}) - g_{k_i}\|_2 \to 0.$$

Then, (23) implies the contradictory relation

$$\begin{aligned}\epsilon &\leq \cos\theta_{k_i}\\ &\leq \|g(x_{k_i} + \bar{\alpha}_{k_i} s_{k_i}) - g_{k_i}\|_2 / ((1-\sigma)\|g_{k_i}\|_2)\\ &\leq \|g(x_{k_i} + \bar{\alpha}_{k_i} s_{k_i}) - g_{k_i}\|_2 / ((1-\sigma)\eta) \to 0.\end{aligned}$$

(ii) (23) and (13) imply that

$$\begin{aligned}\cos\theta_k\|g_k\|_2 &\leq (\lambda\bar{\alpha}_k\|s_k\|_2)/(1-\sigma)\\ &\leq (\lambda M\alpha_k\|s_k\|_2)/(1-\sigma).\end{aligned}$$

This, plus Lemma 2.1, gives

$$\begin{aligned}f(x_k) - f(x_{k+1}) &\geq \rho\alpha_k \cos\theta_k\|g_k\|_2\|s_k\|_2\\ &\geq [\rho(1-\sigma)/(\lambda M)]\cos^2\theta_k\|g_k\|_2^2\\ &= [\rho(1-\sigma)/(\lambda M)]((g_k^t s_k)/\|s_k\|_2)^2.\end{aligned} \tag{24}$$

Hence, the conclusion is true.

(iii) Otherwise, for some $\eta > 0$, $\|g_k\|_2 \geq \eta$ for all k. This and Lemma 2.1 imply that

$$f(x_k) - f(x_{k+1}) \geq \rho\epsilon\eta\|x_{k+1} - x_k\|_2,$$

which furthermore implies that

$$\sum_{k=1}^{\infty} \|x_{k+1} - x_k\|_2 < \infty.$$

Hence, $\{x_k\}$ is a Cauchy sequence in R^n, and therefore $x_k \to x_*$ for some x_*. From this, we see that there exists a neighborhood $B(x_*, r)$ of x_* such that

$$\{x_k + t(x_{k+1} - x_k); 0 \le t \le M\} \subseteq B(x_*, r),$$

for all k. Since f is twice continuously differentiable, g is Lipschitz continuous on the bounded set $B(x_*, r)$ with some Lipschitz coefficient $\lambda > 0$. Take $\bar{\alpha}_k$ as in Lemma 3.1 (21) still holds because $\bar{\alpha}_k \in (0, M\alpha_k]$ and

$$\begin{aligned}\{x_k + \alpha s_k; 0 \le \alpha \le M\alpha_k\} &= \{x_k + t(x_{k+1} - x_k); 0 \le t \le M\} \\ &\subseteq B(x_*, r).\end{aligned} \tag{25}$$

Therefore, (16) is also true for all k. Lemma 2.1 then indicates that

$$\begin{aligned}f(x_k) - f(x_{k+1}) &\ge [\rho(1-\sigma)/(M\lambda)] \cos^2 \theta_k \|g_k\|_2^2 \\ &\ge [\rho(1-\sigma)/(M\lambda)]\epsilon^2 \|g_k\|_2^2, \qquad \forall k.\end{aligned} \tag{26}$$

(26) implies that $\lim_{k\to\infty} \|g_k\|_2 = 0$, a contradiction. □

Note that point (iii) of Theorem 3.1 does not require any extra properties of g, while at point (i) we assume that g is uniformly continuous. In the hypothesis of the following theorem, we have the stronger assumption that G is Lipschitz continuous.

Theorem 3.2. Suppose that $\{s_k\}$ determined by Algorithm A1 is such that $x_k \to x_*$, where x_* is a local minimizer of f and $x_{k+1} = x_k + \alpha_k s_k$, with α_k determined by Algorithm A2, that $G_* = G(x_*)$ is positive definite, and that there is an $r_1 > 0$ and a $\Lambda > 0$ such that

$$\|G(x) - G(y)\|_2 \le \Lambda \|x - y\|_2, \qquad \forall x, y \in B(x_*, r_1). \tag{27}$$

Then, either $x_k = x_*$ for some k, or there is a k_0 such that α_k satisfies (3) for all $k \ge k_0$.

Proof. Let us assume that $x_k \ne x_*$ for all k. The assumption of the theorem implies that there is an $r_2 \in (0, r_1)$ such that:

(i) f is strictly convex on $B(x_*, r_2)$;
(ii) there is an $m > 0$ such that

$$z^t G(x) z \ge m\|z\|_2^2, \qquad \forall x \in B(x_*, r_2), z \in R^n; \tag{28}$$

(iii) $\|G(x) - G(y)\|_2 \le \Lambda \|x - y\|_2, \forall x, y \in B(x_*, r_2)$.

Now, take

$$r=\min\{(1-2/\tau_3)/(2\Lambda M/m), r_2\}>0,$$

where $M=\max\{J, 1/\tau_1\}$. Since $x_k \rightarrow x_*$, there is a k_0 such that, for all $k \geq k_0$,

$$\begin{aligned}\{x_k+\alpha s_k; 0\leq \alpha \leq M\alpha_k\} &= \{x_k+t(x_{k+1}-x_k); 0\leq t\leq M\}\\ &\subseteq B(x_*, r).\end{aligned}$$

For $k\geq k_0$, we consider two possibilities.

(I) If α_k is accepted at either Step 1c or 1d or 2b or 3b, then either $f'(\alpha_k)\geq \sigma f'(0)$, or there is an $a_n \in [0, \alpha_k)$ such that

$$f(\alpha_k)\geq f(a_n)+(\alpha_k-a_n)\sigma f'(0). \tag{29}$$

Since $f(x)$ is strictly convex on $B(x_*, r_2)$, the conclusion is true.

(II) If α_k is accepted at Step 3d or 3e and if $f'(\alpha_k)<\sigma f'(0)$, then it is easy to see that, in Step 3d or 3e, we have $0\leq a_n\leq \alpha_k<b_n\leq M\alpha_k$, as well as

$$f(b_n)>f(\alpha_k)+(b_n-\alpha_k)\rho f'(0). \tag{30}$$

Since $\{x_k+\alpha s_k; 0\leq \alpha\leq M\alpha_k\}\subseteq B(x_*, r)$ and $f(x)$ is strictly convex on $B(x_*, r)$, we have

$$f'(b_n)\geq (f(b_n)-f(\alpha_k))/(b_n-\alpha_k)>\rho f'(0).$$

Hence, there are numbers $s\in[\alpha_k, b_n]\subseteq[a_n, b_n]$ and $t\in[a_n, b_n]$ such that

$$\begin{aligned}f''(s)(b_n-\alpha_k) &= f'(b_n)-f'(\alpha_k)\\ &> (\rho-\sigma)f'(0)\\ &\geq \tau_3\Delta_n(b_n-a_n)\\ &\geq \tau_3\Delta_n(b_n-\alpha_k)\\ &= (\tau_3/2)|f''(t)|(b_n-\alpha_k).\end{aligned} \tag{31}$$

Since

$$f''(t)=s_k^t G(x_k+ts_k)s_k\geq m\|s_k\|_2^2>0,$$

(31) implies that

$$f''(s)>(\tau_3/2)f''(t)>0.$$

This furthermore implies that

$$1>(\tau_3/2)[1-(f''(s)-f''(t))/f''(s)]. \tag{32}$$

On the other hand,

$$
\begin{aligned}
(f''(s)-f''(t))/f''(s) &\le (\|G(x_k+ss_k) \\
&\qquad - G(x_k+ts_k)\|_2\|s_k\|_2^2)/(m\|s_k\|_2^2) \\
&\le (\Lambda/m)|s-t|\|s_k\|_2 \\
&\le (\Lambda/m)(b_n-a_n)\|s_k\|_2 \\
&\le (\Lambda/m)M\alpha_k\|s_k\|_2 \\
&= (\Lambda/m)M\|x_{k+1}-x_k\|_2 \\
&\le (2\Lambda/m)Mr \\
&\le 1-2/\tau_3. \qquad (33)
\end{aligned}
$$

Combining (32) and (33), we arrive at a contradiction. □

The above theorem proves an asymptotic property of Algorithm A2. In what follows, we show that our algorithm is globally convergent under the same assumptions as those considered in Ref. 13.

Lemma 3.2. Assume that the level set D is convex and that f is uniformly convex on D; i.e., there are two numbers $\mu \ge \nu > 0$, such that

$$\nu\|z\|_2^2 \le z^tG(x)z \le \mu\|z\|_2^2, \qquad \forall x\in D, \forall z\in R^n. \qquad (34)$$

Consider Algorithm A1, where α_k is obtained from Algorithm A2. Then, there is a number $\sigma_1 \in (\rho, 1)$ such that $f'(\alpha_k) \ge \sigma_1 f'(0)$ for all k.

Proof. The uniform convexity (34) implies that the Lipschitz condition (13) is satisfied with $\lambda=\mu$. Then from (34) and Lemma 3.1, and by using the fact that both x_k and $x_k+\alpha_k s_k$ are in the convex set D, we obtain

$$
\begin{aligned}
f'(\alpha_k)-f'(0) &= f''(t)\alpha_k \\
&= s_k^tG(x_k+ts_k)s_k\cdot\alpha_k \\
&\ge \nu\|s_k\|_2^2[(\sigma-1)/(M\mu\|s_k\|_2^2)]g_k^ts_k \\
&= [\nu(\sigma-1)/(M\mu)]f'(0),
\end{aligned}
$$

for some $t\in(0,\alpha_k)$. The lemma follows by taking

$$\sigma_1 = 1-\nu(1-\sigma)/(M\mu). \qquad □$$

Using the above result and the techniques developed in Ref. 13, we can easily prove the following theorem.

Theorem 3.3. Assume that the level set D is convex and that f is uniformly convex on D. Consider Algorithm A1, where we choose $s_k = -B_k^{-1} g_k$, with B_k given by a member of the restricted Broyden family with $\phi \in [0, 1)$; (see Ref. 13), and where α_k is obtained from Algorithm A2. Then starting with x_1 and any symmetric positive-definite matrix B_1, we have $x_k \to x_*$, where x_* is the unique minimizer of f.

Finally, the following theorem shows that our line search algorithm guarantees a sufficient decrease in f.

Theorem 3.4. Let $\{x_k\}$ be generated by Algorithm A1, where α_k is given by Algorithm A2. Then, f incurs a sufficient decrease with the steplength α_k for all k.

Proof. From Lemma 3.1,

$$\alpha_k \geq (1/M)\bar{\alpha}_k,$$

for all k where $\bar{\alpha}_k$ satisfies (2)–(3) and

$$M = \max\{J, 1/\tau_1\} > 1.$$

It is clear that, if $\Phi(t)$ is a forcing function, then so is $(1/M)\Phi(t)$. Lemma 1.1, Lemma 1.2, Theorem 1.1, and Lemma 2.1 then imply our conclusion. □

4. Preliminary Numerical Experiments

The conclusions of the previous sections show that the new line search Algorithm A2 preserves all the convergence properties of (2), (3), yet the computational work of extra gradient evaluations is saved. In this section, we present some preliminary numerical experiments comparing Algorithm A2 with Algorithm A3 [for (2), (4)] and Algorithm A4 [for (2), (3)] of Appendix A. Algorithm A3 is the same as the one presented in Ref. 2 and Algorithm A4 is a slight modification of A3 for the case in which we want to use rules (2), (3). Our implementations are straightforward. All the parameters are chosen in such a way that they would be consistent with all three algorithms. The descent direction in Step 2b of Algorithm A1 was computed as $s_k = -H_k g_k$, where the approximation H_k to the inverse of the Hessian is updated by the BFGS formula as described in Ref. 2. H_1 is chosen to be the identity matrix. The test problems are taken from Ref. 14 and are

Table 1. Test problems.

Problem	Function name	Dimension	Initial point
1	Helical valley function	3	$(-1, 0, 0)$
2	Biggs exp6 function	6	$(1, 2, 1, 1, 1, 1)$
3	Gaussian function	3	$(0.4, 1, 0)$
4	Powell badly scaled function	2	$(0, 1)$
5	Box 3-dimensional function	3	$(0, 10, 20)$
6	Variably dimensioned function	n	$(1-i/n, i=1, \dots, n)$
7	Watson function	n	$(0, \dots, 0)$
8	Penalty function 1	n	$(1, 2, \dots, n)$
9	Penalty function 2	n	$(0.5, \dots, 0.5)$
10	Brown badly scaled function	2	$(1, 1)$
11	Brown and Dennis function	4	$(25, 5, -5, -1)$
12	Gulf research and development function	3	$(5, 2.5, 0.15)$
13	Trigonometric function	n	$(1/n, 1/n, \dots, 1/n)$
14	Extended Rosebrock function	n	$(-1.2, 1, \dots, -1.2, 1)$
15	Extended Powell singular function	n	$(3, -1, 0, 1, \dots, 3, -1, 0, 1)$
16	Beale function	2	$(1, 1)$
17	Wood function	4	$(-3, -1, -3, -1)$
18	Chebyquad function	n	$(i/(n+1), i=1, \dots, n)$

Only function names are listed; for the explicit expressions of the function, see Ref. 14; n is a user-given integer.

listed in Table 1. The machine used was an Encore-Multimax with double precision. The termination criterion is

$$f(x_k) - f(x_{k+1}) \leq \epsilon,$$

where ϵ is a given positive number (cf. Ref. 2). The parameters in Algorithm A2 are chosen as

$$J=2, \qquad \tau_1=0.1, \qquad \tau_2=0.5, \qquad \tau_3=2.5.$$

The parameters in Algorithms A3 and A4 are chosen as

$$\lambda=9, \qquad \tau_1=0.1, \qquad \tau_2=0.5.$$

This corresponds to the values proposed in Ref. 2, pp. 34–36. Without loss of generality, the user-supplied lower bound $\bar{f}$ is set as $\bar{f}=-$maxreal, where maxreal is the maximal real number representable on the machine. In Algorithms A3 and A4, if

$$\mu > 2\bar{\alpha}_i - \bar{\alpha}_{i-1} \quad \text{and} \quad \min(\mu, \bar{\alpha}_i + \lambda(\bar{\alpha}_i - \bar{\alpha}_{i-1})) = \bar{\alpha}_i + \lambda(\bar{\alpha}_i - \bar{\alpha}_{i-1}),$$

then

$$\bar{\alpha}_{i+1} \in [2\bar{\alpha}_i - \bar{\alpha}_{i-1}, \bar{\alpha}_i + \lambda(\bar{\alpha}_i - \bar{\alpha}_{i-1})]$$

is chosen as

$$\bar{\alpha}_2 = 2 = 2\bar{\alpha}_1 - \bar{\alpha}_0 = 2\bar{\alpha}_1, \qquad \text{if } i=1, \tag{35a}$$

$$\bar{\alpha}_{i+1} = \bar{\alpha}_i + 2(\bar{\alpha}_i - \bar{\alpha}_{i-1}) = 2\bar{\alpha}_i, \qquad \text{if } i \geq 2. \tag{35b}$$

This is consistent with $J=2$ in Algorithm A2. In choosing c_n at Step 3a of Algorithm A2, we noticed that a_n is the current best estimate of α_k; hence, we took

$c_1 = a_1 + 0.3(b_1 - a_1)$;
for $n = 2, 3, \ldots$, if $\{a_n, b_n\} = \{a_{n-1}, c_{n-1}\}$, then $t_n = b_{n-1}$, else $t_n = a_{n-1}$.

Let $P(a_n, b_n, t_n)(x)$ be the quadratic polynomial interpolating $f(\alpha)$ at a_n, b_n, t_n. If $P(a_n, b_n, t_n)(x)$ has a minimum x_n and if

$$x_n \in [a_n + 0.1(b_n - a_n), a_n + 0.5(b_n - a_n)],$$

then $c_n = x_n$; otherwise,

$$c_n = a_n + 0.3(b_n - a_n).$$

For Algorithms A3 and A4, we chose

$$\bar{\alpha}_j \in [[a_j + 0.1(b_j - a_j), b_j - 0.5(b_j - a_j)]]$$

in a similar manner, using the fact that a_j is the current best estimate of α_k.

As we mentioned in Section 2, when using Algorithm A2 with $\sigma \leq 1/2$, we simply check Step 1c when k is small, and then switch to Step 1d when k is large enough in some sense. In our experiments, with $\sigma = 0.1$, we switched from Step 1c to 1d when

$$f(x_k) - f(x_{k+1}) \leq 1.05\epsilon.$$

With Algorithm A2, we do not guarantee that

$$(g_{k+1} - g_k)^t (x_{k+1} - x_k) > 0, \qquad \forall k. \tag{36}$$

However, in the BFGS formula (Ref. 2), (36) is needed to preserve the positive definiteness of the matrix H_k. Therefore in our implementation, we used the following modification after calling Algorithm A2:

denote $\gamma_k = g_{k+1} - g_k$, $\delta_k = x_{k+1} - x_k$;
if $\gamma_k^t \delta_k > 0$, no modification;
if $\gamma_k^t \delta_k \leq 0$, then set $\bar{\delta}_k = \delta_k + c_k \gamma_k$, where

$c_k = [(\sigma g_k - g_{k+1})^t \delta_k] / (\gamma_k^t \gamma_k)$;

use $\bar{\delta}_k$ in place of δ_k in updating H_k.

With the above modification,

$$\gamma_k^t \bar{\delta}_k > 0, \qquad \text{for all } k.$$

H_{k+1} (see Ref. 2) will be positive definite and $H_{k+1}\gamma_k = \bar{\delta}_k$. Furthermore, according to Theorem 3.2, asymptotically we will have $f'(\alpha_k) \geq \sigma f'(0)$, which implies that $\gamma_k^t \delta_k > 0$. Hence, the superlinear local convergence of BFGS will be preserved.

We also want to mention that, although the above modification was implemented, in our experiments the case $\gamma_k^t \delta_k \leq 0$ never actually occurred, supporting our arguments at the end of Section 2.

In our experiments, we tested all problems listed in Table 1 with $n = 20, 40$, $\epsilon = 10^{-8}$, 10^{-12}, $(\sigma, \rho) = (0.9, 0.001)$ for inaccurate line search, and $(\sigma, \rho) = (0.1, 0.05)$ or relatively accurate line search. The results are listed in Tables 3–6 of Appendix B; they are summarized in Table 2. In Tables 3–6, the first column indicates the number of the test problem; for example, 6 (20) means Problem 6 with dimension $n = 20$. The second, third, and fourth columns show the number of iterations, number of function evaluations, and number of gradient evaluations, respectively. The abbreviations 'nit', 'nfe', 'nge' stand for number of iterations, number of function evaluations, and number of gradient evaluations, respectively. We set the maximum iterations number to be 500 in our experiments, and — means that the problem is not solved within 500 iterations. For the convenience of comparison, we list in Table 2 the total number of iterations (NIT), total number of function evaluations (NFE), and total number of gradient evaluations (NGE) in solving all the problems. The tables show that Algorithm A2 is comparable with A3 and A4 in our experiments for inexact line search.

When we use a line search algorithm with either rules (2), (3) or rules (2), (4), we can reduce the number of iterations by making the line search relatively more accurate. In fact, when $(\sigma, \rho) = (0.1, 0.05)$ nit is much less than that for $(\sigma, \rho) = (0.9, 0.001)$. However, nfe and nge increase tremendously. Since generally the function and gradient evaluations are more expensive, many authors suggested that in practice accurate line searches should be avoided. Actually, choices such as $(\sigma, \rho) = (0.9, 0.001)$ are very often used in practice. By contrast, our experiments show that, with our line search algorithm, a relatively more accurate line search may significantly reduce the number of iterations as well as the number of gradient evaluations. However, the number of function evaluations will increase. Therefore, this algorithm is especially appropriate for the situation when gradient evaluations are very expensive relative to function evaluations.

Table 2. Total number of iterations, function evaluations, and gradient evaluations in solving all the problems.

	Algorithm A2			Algorithm A3			Algorithm A4		
Cases	NIT	NFE	NGE	NIT	NFE	NGE	NIT	NFE	NGE
1	1455	2928	1455	1470	2878	1560	1450	2820	1489
2	1258	8765	1270	1104	4408	2801	1326	3718	2466
3	1 problem unsolved			2 problems unsolved			2 problems unsolved		
4	1 problem unsolved			1 problem unsolved			1 problem unsolved		

Case 1: $\sigma=0.9$, $\rho=0.001$, $\epsilon=10^{-8}$.
Case 2: $\sigma=0.1$, $\rho=0.05$, $\epsilon=10^{-8}$.
Case 3: $\sigma=0.9$, $\rho=0.001$, $\epsilon=10^{-12}$.
Case 4: $\sigma=0.1$, $\rho=0.05$, $\epsilon=10^{-12}$.

5. Appendix A: Line Search Algorithms

In this appendix, we list two line search algorithms with rules (2), (3) and (2), (4), respectively. Algorithm A3, using (2), (4), is the same as the one presented in Ref. 2. We check $\alpha_k=1$ first as we do in Algorithm A2. Algorithm A4 is a modification of Algorithm A3 so that we use (2), (3) instead of (2), (4). Algorithm A3 is associated with a subroutine Section 1 (a_i, b_i) and Algorithm A4 needs to use a subroutine Section 2 (a_i, b_i). There are six user-given parameters ρ, σ, λ, τ_1, τ_2, $\bar{f}$, such that

$$\rho\in(0, 1/2), \qquad \sigma\in(\rho, 1), \qquad \lambda>1, \qquad 0<\tau_1<\tau_2\leq 1/2,$$

and $\bar{f}$ is a user-supplied lower bound of $f(\alpha)$.

Algorithm A3.

```
Step 0.   Set $\bar{\alpha}_0=0$, $\bar{\alpha}_1=1$, $\mu=(\bar{f}-f(0))/(\rho f'(0))$.
Step 1.   For $i=1, 2, \ldots$, execute the computations below.
Step 2.   Evaluate $f(\bar{\alpha}_i)$.
Step 2a.  If $f(\bar{\alpha}_i)\leq\bar{f}$, then set $\alpha_k=\bar{\alpha}_i$, terminate.
Step 2b.  If $f(\bar{\alpha}_i)>f(0)+\bar{\alpha}_i\rho f'(0)$ or $f(\bar{\alpha}_i)\geq f(\bar{\alpha}_{i-1})$, then set
          $a_i=\bar{\alpha}_{i-1}$, $b_i=\bar{\alpha}_i$, execute subroutine Section 1 $(a_i, b_i)$,
          terminate.
Step 3.   Evaluate $f'(\bar{\alpha}_i)$.
Step 3a.  If $|f'(\bar{\alpha}_i)|\leq-\sigma f'(0)$, then set $\alpha_k=\bar{\alpha}_i$, terminate.
```

Step 3b. If $f'(\bar{a}_i) \geq 0$, then set $a_i = \bar{a}_i, b_i = \bar{a_{i-1}}$, execute subroutine Section 1 (a_i, b_i), terminate.

Step 4. If $\mu \leq 2\bar{a}_i - \bar{a}_{i-1}$, then take $\bar{a_{i+1}} = \mu$; else, take $\bar{a_{i+1}} \in [2\bar{a}_i - \bar{a_{i-1}}, \min(\mu, \bar{a}_i + \lambda(\bar{a}_i - \bar{a_{i-1}}))]$.

Subroutine Section 1 (a_i, b_i). Note that now it is not necessary that $a_i < b_i$. For convenience, let us use $[[a, b]]$ to denote an interval with endpoints a and b, and a is not necessarily less than b.

Step 1. For $j = i, i+1, \ldots$, execute the computations below.

Step 2. Take

$$\bar{a}_j \in [[a_j + \tau_1(b_j - a_j), b_j - \tau_2(b_j - a_j)]].$$

Step 3. Evaluate $f(\bar{a}_j)$.

Step 4. If $f(\bar{a}_j) > f(0) + \bar{a}_j \rho f'(0)$ or $f(\bar{a}_j) \geq f(a_j)$, then set $a_{j+1} = a_j$, $b_{j+1} = \bar{a}_j$; else, evaluate $f'(\bar{a}_j)$.

Step 5. If $|f'(\bar{a}_j)| \leq -\sigma f'(0)$, then set $\alpha_k = \bar{a}_j$, terminate; else, set $a_{j+1} = \bar{a}_j$.

Step 6. If $(b_j - a_j) f'(\bar{a}_j) \geq 0$, then $b_{j+1} = a_j$; else, $b_{j+1} = b_j$.

Algorithm A4.

Step 0. Set $\bar{a}_0 = 0$, $\bar{a}_1 = 1$, $\mu = (\bar{f} - f(0))/(\rho f'(0))$.

Step 1. For $i = 1, 2, \ldots$, execute the computations below.

Step 2. Evaluate $f(\bar{a}_i)$.

Step 2a. If $f(\bar{a}_i) \leq \bar{f}$, then set $\alpha_k = \bar{a}_i$, terminate.

Step 2b. If $f(\bar{a}_i) > f(0) + \bar{a}_i \rho f'(0)$, then set $a_i = \bar{a_{i-1}}, b_i = \bar{a}_i$, execute subroutine Section 2 (a_i, b_i), terminate.

Step 3. Evaluate $f'(\bar{a}_i)$.

Step 3a. If $f'(\bar{a}_i) \geq \sigma f'(0)$, then set $\alpha_k = \bar{a}_i$, terminate.

Step 4. If $\mu \leq 2\bar{a}_i - \bar{a}_{i-1}$, then take $\bar{a}_{i+1} = \mu$; else, take $\bar{a_{i+1}} \in [2\bar{a}_i - \bar{a_{i-1}}, \min(\mu, \bar{a}_i + \lambda(\bar{a}_i - \bar{a_{i-1}}))]$.

Subroutine Section 2 (a_i, b_i).

Step 1. For $j = i, i+1, \ldots$, execute the computations below.

Step 2. Take

$$\bar{\alpha}_j \in [[a_j + \tau_1(b_j - a_j), b_j - \tau_2(b_j - a_j)]].$$

Step 3. Evaluate $f(\bar{\alpha}_j)$.

Step 4. If $f(\bar{\alpha}_j) > f(0) + \bar{\alpha}_j \rho f'(0)$, then set $a_{j+1} = a_j$, $b_{j+1} = \bar{\alpha}_j$; else, evaluate $f'(\bar{\alpha}_j)$.

Step 5. If $f'(\bar{\alpha}_j) \geq \sigma f'(0)$, then set $\alpha_k = \bar{\alpha}_j$, terminate; else, set $a_{j+1} = \bar{\alpha}_j$, $b_{j+1} = b_j$.

Note that in subroutine Section 2 (a_i, b_i), by induction it is easy to see that $b_j > a_j$ for all $j = i, i+1, \ldots$. Therefore, $f'(\bar{\alpha}_j) < \sigma f'(0) < 0$ will imply that $(b_j - a_j) f'(\bar{\alpha}_j) < 0$.

6. Appendix B: Results of Numerical Experiments

This appendix presents detailed results of the numerical experiments summarized in Table 2 of Section 4. We mention again that the first column in the following tables (3–6) indicates the number of the test problem, where for example 6 (20) means Problem 6 with dimension $n = 20$. See Table 1 of Section 4 for the list of test problems.

Table 3. Number of iterations, function evaluations, and gradient evaluations when $\sigma=0.9$, $\rho=0.001$, and $\epsilon=10^{-8}$.

	Algorithm A2			Algorithm A3			Algorithm A4		
Problem	nit	nfe	nge	nit	nfe	nge	nit	nfe	nge
1	28	45	28	27	44	28	27	44	28
2	37	44	37	36	44	39	37	43	38
3	3	7	3	3	6	4	3	6	4
4	67	98	67	65	100	74	53	80	55
5	22	37	22	16	26	19	16	26	19
6 (20)	23	47	23	23	47	24	23	47	24
6 (40)	25	57	25	25	56	27	25	56	27
7 (20)	45	73	45	45	73	46	45	73	46
7 (40)	40	74	40	40	74	41	40	74	41
8 (20)	17	34	17	17	33	18	17	33	18
8 (40)	25	54	25	25	54	26	25	54	26
9 (20)	81	225	81	71	207	78	80	216	83
9 (40)	263	545	263	270	509	283	286	522	290
10	10	47	10	12	49	13	12	49	13
11	22	70	22	22	70	23	22	70	23
12	30	42	30	33	45	36	33	45	36
13 (20)	39	44	39	39	44	42	39	44	42
13 (40)	35	37	35	35	38	37	35	37	36
14 (20)	140	250	140	123	241	127	124	235	125
14 (40)	193	420	193	190	421	204	199	427	200
15 (20)	60	103	60	60	102	61	60	102	61
15 (40)	68	136	68	69	140	72	68	136	69
16	13	19	13	13	19	14	13	19	14
17	28	56	28	28	56	29	28	56	29
18 (20)	43	98	43	42	98	46	43	98	44
18 (40)	138	266	138	141	282	149	97	228	98

Table 4. Number of iterations, function evaluations, and gradient evaluations when $\sigma = 0.1$, $\rho = 0.05$, and $\epsilon = 10^{-8}$.

	Algorithm A2			Algorithm A3			Algorithm A4		
Problem	nit	nfe	nge	nit	nfe	nge	nit	nfe	nge
1	23	151	23	19	65	43	28	62	49
2	25	170	25	25	86	69	31	78	71
3	3	24	3	3	11	9	3	10	8
4	45	306	45	36	154	106	47	120	93
5	14	100	14	13	57	42	21	55	48
6 (20)	9	80	9	7	48	25	7	46	23
6 (40)	11	100	11	9	60	30	12	62	33
7 (20)	29	211	29	30	122	84	36	103	76
7 (40)	29	203	29	29	121	79	40	110	78
8 (20)	70	497	70	73	283	214	106	237	207
8 (40)	90	613	90	8	47	29	12	51	32
9 (20)	51	436	51	38	177	91	58	245	110
9 (40)	180	1170	180	176	722	472	220	619	438
10	10	94	10	10	55	20	10	49	14
11	15	125	15	14	82	32	19	80	33
12	22	160	22	17	74	60	16	39	32
13 (20)	34	207	34	33	89	74	37	69	69
13 (40)	33	183	33	32	102	81	34	69	66
14 (20)	88	623	90	86	336	210	91	266	167
14 (40)	155	1105	155	151	613	352	165	489	285
15 (20)	49	295	52	31	124	78	40	107	68
15 (40)	62	442	62	34	138	75	75	219	147
16	11	65	11	10	35	27	13	27	22
17	24	153	24	52	192	132	30	90	60
18 (20)	42	302	42	38	150	84	41	106	55
18 (40)	144	950	151	130	465	283	134	310	182

Table 5. Number of iterations, function evaluations, and gradient evaluations when $\sigma=0.9$, $\rho=0.001$, and $\epsilon=10^{-12}$.

	Algorithm A2			Algorithm A3			Algorithm A4		
Problem	nit	nfe	nge	nit	nfe	nge	nit	nfe	nge
1	29	46	29	29	46	30	29	46	30
2	39	46	39	38	46	41	39	45	40
3	5	9	5	5	8	6	5	8	6
4	152	205	152	146	211	166	153	207	155
5	23	38	23	17	27	20	17	27	20
6 (20)	25	49	25	25	49	26	25	49	26
6 (40)	26	58	26	26	57	28	26	57	28
7 (20)	60	89	60	60	88	61	60	88	61
7 (40)	68	103	68	68	102	69	68	102	69
8 (20)	141	209	141	130	196	147	142	200	149
8 (40)	262	366	262	144	225	158	256	354	265
9 (20)	437	681	437	416	666	446	—		
9 (40)	321	603	321	331	570	344	329	565	333
10	11	48	11	13	50	14	13	50	14
11	23	73	23	—			24	79	27
12	32	44	32	35	47	38	35	47	38
13 (20)	45	50	45	45	50	48	45	50	48
13 (40)	47	49	47	47	50	49	47	49	48
14 (20)	174	284	174	146	264	150	149	260	150
14 (40)	227	454	227	224	455	238	243	471	244
15 (20)	111	156	111	107	150	109	114	156	115
15 (40)	120	189	120	116	187	119	124	192	125
16	14	20	14	14	20	15	14	20	15
17	31	59	31	31	59	32	31	59	32
18 (20)	54	109	54	54	110	58	54	109	55
18 (40)	—			—			—		

Table 6. Number of iterations, function evaluations, and gradient evaluations when $\sigma = 0.1$, $\rho = 0.05$, and $\epsilon = 10^{-12}$.

	Algorithm A2			Algorithm A3			Algorithm A4		
Problem	nit	nfe	nge	nit	nfe	nge	nit	nfe	nge
1	25	163	25	21	67	45	29	63	50
2	26	176	26	27	88	71	33	80	73
3	5	36	5	5	13	11	5	12	10
4	107	750	107	91	356	261	120	299	247
5	15	105	15	14	58	43	22	56	49
6 (20)	10	86	10	8	49	26	9	48	25
6 (40)	12	106	12	10	61	31	14	64	35
7 (20)	34	249	34	34	142	102	45	130	103
7 (40)	40	283	40	38	166	120	58	162	130
8 (20)	76	530	76	77	297	227	114	250	220
8 (40)	110	741	110	34	137	104	133	339	284
9 (20)	244	1676	257	211	800	576	295	760	584
9 (40)	202	1329	202	199	819	554	271	748	567
10	10	94	10	10	55	20	10	49	14
11	15	125	15	—			20	130	64
12	24	172	24	19	76	62	26	67	60
13 (20)	40	259	40	37	104	86	40	73	73
13 (40)	42	237	42	43	134	109	49	98	95
14 (20)	99	707	99	98	391	258	111	320	221
14 (40)	177	1262	177	169	696	422	190	563	359
15 (20)	67	404	67	39	148	99	70	187	148
15 (40)	93	653	93	55	219	139	120	325	253
16	13	75	13	11	36	28	14	28	23
17	25	159	25	54	194	134	32	92	62
18 (20)	48	344	48	44	171	102	50	126	75
18 (40)	—			362	1199	870	—		

References

1. Dennis, J. E., and Schnabel, R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
2. Fletcher, R., *Practical Methods of Optimization*, John Wiley and Sons, New York, New York, 1987.
3. Goldstein, A. A., *On Steepest Descent*, SIAM Journal on Control, Vol. 3, pp. 147-151, 1965.
4. Goldstein, A. A., *Constructive Real Analysis*, Harpers and Row, New York, New York, 1967.
5. Armijo, L., *Minimization of Functions Having Lipschitz Continuous First Partial Derivatives*, Pacific Journal of Mathematics, Vol. 16, pp. 1-3, 1966.
6. Wolfe, P., *Convergence Conditions for Ascent Methods*, SIAM Review, Vol. 11, pp. 226-235, 1968.
7. Powell, M. J. D., *Some Global Convergence Properties of a Variable-Metric Algorithm for Minimization without Exact Line Searches*, SIAM-AMS Proceedings, SIAM Publications, Philadelphia, Pennsylvania, Vol. 9, pp. 53-72, 1976.
8. Luenberger, D. G., *Linear and Nonlinear Programming*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.
9. Boggs, P. T., and Schnabel, R. B., *Lecture Notes for a Short Course on Numerical Optimization*, Manuscript, Houston, Texas, 1987.
10. Gill, E. P., Murray, W., Saunders, A. M., and Wright, H. M. *A Note on a Sufficient-Decrease Criterion for a Nonderivative Steplength Procedure*, Mathematical Programming, Vol. 23, pp. 349-352, 1982.
11. Ortega, J. M., and Rheinboldt, W. C., *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, London, England, 1970.
12. Brent, R. P., *Algorithms for Minimization without Derivatives*, Prentice-Hall, Englewood Cliffs, New Jersey, 1972.
13. Byrd, R. H., Nocedal, and Yuan, Y. X., *Global Convergence of a Class of Quasi-Newton Methods on Convex Problems*, SIAM Journal on Numerical Analysis, Vol. 24, pp. 1171-1190, 1987.
14. Moré, J. J., Garbow, B. S., and Hillstrom, K. E., *Testing Unconstrained Optimization Software*, ACM Transactions on Mathematical Software, Vol. 7, pp. 17-41, 1981.
15. Prapasri, A., *Application of Conjugate Directions and Quasi-Newton Methods in Parallel Unconstrained Optimization*, PhD Thesis, University of Iowa, 1989.
16. Prapasri, A., and Potra, F. A., *Parallel Line Search Algorithms for Solving Convex Unconstrained Minimization Problems*, Libertas Mathematica, Vol. 8, pp. 31-46, 1988.