Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: https://www.tandfonline.com/loi/goms20

Analysis of the gradient method with an Armijo–Wolfe line search on a class of non-smooth convex functions

Azam Asl & Michael L. Overton

To cite this article: Azam Asl & Michael L. Overton (2020) Analysis of the gradient method with an Armijo–Wolfe line search on a class of non-smooth convex functions, Optimization Methods and Software, 35:2, 223-242, DOI: 10.1080/10556788.2019.1673388

To link to this article: https://doi.org/10.1080/10556788.2019.1673388

Published online: 09 Oct 2019.

OPTIMIZATION METHODS & SOFTWARE
2020, VOL. 35, NO. 2, 223–242
https://doi.org/10.1080/10556788.2019.1673388

Analysis of the gradient method with an Armijo–Wolfe line search on a class of non-smooth convex functions

Azam Asl and Michael L. Overton

Courant Institute of Mathematical Sciences, New York University, New York, NY, USA

ABSTRACT
It has long been known that the gradient (steepest descent) method may fail on non-smooth problems, but the examples that have appeared in the literature are either devised specifically to defeat a gradient or subgradient method with an exact line search or are unstable with respect to perturbation of the initial point. We give an analysis of the gradient method with steplengths satisfying the Armijo and Wolfe inexact line search conditions on the non-smooth convex function $f(x) = a|x^{(1)}| + \sum_{i=2}^{n} x^{(i)}$. We show that if a is sufficiently large, satisfying a condition that depends only on the Armijo parameter, then, when the method is initiated at any point $x_0 \in \mathbb{R}^n$ with $x_0^{(1)} \neq 0$, the iterates converge to a point $\bar{x}$ with $\bar{x}^{(1)} = 0$, although f is unbounded below. We also give conditions under which the iterates $f(x_k) \to -\infty$, using a specific Armijo–Wolfe bracketing line search. Our experimental results demonstrate that our analysis is reasonably tight.

ARTICLE HISTORY
Received 3 June 2019
Accepted 24 September 2019

KEYWORDS
Steepest descent method; convex optimization; non-smooth optimization

2010 MATHEMATICS SUBJECT CLASSIFICATIONS
49M30; 65K05; 90C25

1. Introduction

First-order methods have experienced a widespread revival in recent years, as the number of variables n in many applied optimization problems has grown far too large to apply methods that require more than $O(n)$ operations per iteration. Yet many widely used methods, including limited-memory quasi-Newton and conjugate gradient methods, remain poorly understood on non-smooth problems, and even the simplest such method, the gradient method, is non-trivial to analyse in this setting. Our interest is in methods with inexact line searches, since exact line searches are clearly out of the question when the number of variables is large. We discuss methods with prescribed step sizes in Section 5.

The gradient method dates back to Cauchy [6]. Armijo [1] was the first to establish convergence to stationary points of smooth functions using an inexact line search with a simple 'sufficient decrease' condition. Wolfe [26], discussing line search methods for more general classes of methods, introduced a 'directional derivative increase' condition among several others. The Armijo condition ensures that the line search step is not too large while the Wolfe condition ensures that it is not too small. Powell [22] seems to have been the

CONTACT Michael L. Overton overton@cs.nyu.edu
Dedicated to Ya-Xiang Yuan on the occasion of his 60th birthday

first to point out that combining the two conditions leads to a convenient bracketing line search, noting also in another paper [23] that use of the Wolfe condition ensures that, for quasi-Newton methods, the updated Hessian approximation is positive definite. Hiriart-Urruty and Lemaréchal [12, Vol. 1, Ch. 11.3] give an excellent discussion of all these issues, although they reference neither [1] nor [22,23]. They also comment (p. 402) on a surprising error in [6].

Suppose that f, the function to be minimized, is a non-smooth convex function. An example of Wolfe [27] shows that the ordinary gradient method with an exact line search may converge to a non-optimal point, without encountering any points where f is non-smooth except in the limit. This example is stable under perturbation of the starting point, but it does not apply when the line search is inexact. Another example given in [12, vol. 1, p. 363] applies to a subgradient method in which the search direction is defined by the steepest descent direction, i.e. the negative of the element of the subdifferential with smallest norm, again showing that use of an exact line search results in convergence to a non-optimal point. This example is also stable under perturbation of the initial point, and, unlike Wolfe's example, it also applies when an inexact line search is used, but it is more complicated than is needed for the results we give below because it was specifically designed to defeat the steepest-descent subgradient method with an exact line search. Another example of convergence to a non-optimal point of a convex max function using a specific subgradient method with an exact line search goes back to [8]; see [9, p. 385]. More generally, in a 'black-box' subgradient method, the search direction is the negative of any subgradient returned by an 'oracle', which may not be a descent direction if the function is not differentiable at the point, although this is unlikely if the current point was not generated by an exact line search since convex functions are differentiable almost everywhere. The key advantage of the subgradient method is that, as long as f is convex and bounded below, convergence to its minimal value can be guaranteed even if f is non-smooth by predefining a sequence of steplengths to be used, but the disadvantage is that convergence is usually slow. Nesterov [19] improved the complexity of such methods using a smoothing technique, but to apply this one needs some knowledge of the structure of the objective function.

The counterexamples mentioned above motivated the introduction of bundle methods by Lemaréchal [15] and Wolfe [27] for non-smooth convex functions and, for non-smooth, non-convex problems, the bundle methods of Kiwiel [13] and the gradient sampling algorithms of Burke *et al.* [4] and Kiwiel [14]. These algorithms all have fairly strong convergence properties, to a non-smooth (Clarke) stationary value when these exist in the non-convex case (for gradient sampling, with probability one), but when the number of variables is large the cost per iteration is much higher than the cost of a gradient step. See the recent survey paper [5] for more details. The 'full' BFGS method is a very effective alternative choice for non-smooth optimization [16], and its $O(n^2)$ cost per iteration (for the matrix-vector products that it requires) is generally much less than the cost of the bundle or gradient sampling methods, but its convergence results for non-smooth functions are limited to very special cases. The limited memory variant of BFGS [17] costs only $O(n)$ operations per iteration, like the gradient method, but its behaviour on non-smooth problems is less predictable.

In this paper we analyse the ordinary gradient method with an inexact line search applied to a simple non-smooth convex function. We require points accepted by the line search to satisfy both Armijo and Wolfe conditions for two reasons. The first is that our longer-term goal is to carry out a related analysis for the limited memory BFGS method for which the Wolfe condition is essential. The second is that although the Armijo condition is enough to prove convergence of the gradient method on smooth functions, the inclusion of the Wolfe condition is potentially useful in the non-smooth case, where the norm of the gradient gives no useful information such as an estimate of the distance to a minimizer. For example, consider the absolute value function in one variable initialized at x_0 with x_0 large. A unit step gradient method with only an Armijo condition will require $O(x_0)$ iterations just to change the sign of x, while an Armijo–Wolfe line search with extrapolation defined by doubling requires only one line search with $O(\log_2(x_0))$ extrapolations to change the sign of x. Obviously, the so-called strong Wolfe condition recommended in many books for smooth optimization, which requires a reduction in the absolute value of the directional derivative, is a disastrous choice when f is non-smooth. We mention here that in a recent paper on the analysis of the gradient method with fixed step sizes [25], Taylor *et al.* remark that 'we believe it would be interesting to analyse [gradient] algorithms involving line-search, such as backtracking or Armijo–Wolfe procedures'.

We focus on the non-smooth convex function mapping from $\mathbb{R}^n$ to $\mathbb{R}$ defined by

$$f(x) = a|x^{(1)}| + \sum_{i=2}^{n} x^{(i)}. \tag{1}$$

We show that if a satisfies a lower bound that depends only on the Armijo parameter, then the iterates generated by the gradient method with steps satisfying Armijo and Wolfe conditions converge to a point $\bar{x}$ with $\bar{x}^{(1)} = 0$, regardless of the starting point, although f is unbounded below. The function f defined in (1) was also used by Lewis and Overton [16, p. 136] with $n = 2$ and $a = 2$ to illustrate failure of the gradient method with a specific line search, but the observations made there are not stable with respect to small changes in the initial point.

The paper is organized as follows. In Section 2 we establish the main theoretical results, without assuming the use of any specific line search beyond satisfaction of the Armijo and Wolfe conditions. In Section 3, we extend these results assuming the use of a bracketing line search that is a specific instance of the ones outlined by [12,22]. In Section 4, we give experimental results, showing that our theoretical results are reasonably tight. We discuss connections with the convergence theory for subgradient methods in Section 5. We make some concluding remarks in Section 6.

2. Convergence results independent of a specific line search

First let f denote any locally Lipschitz function mapping $\mathbb{R}^n$ to $\mathbb{R}$, and let $x_k \in \mathbb{R}^n$, $k = 0, 1, \dots,$ denote the kth iterate of an optimization algorithm where f is differentiable at x_k with gradient $\nabla f(x_k)$. Let $d_k \in \mathbb{R}^n$ denote a descent direction at the kth iteration, i.e. satisfying $\nabla f(x_k)^T d_k < 0$, and assume that f is bounded below on the line $\{x_k + td_k : t \geq 0\}$. Let c_1 and c_2, respectively, the Armijo and Wolfe parameters, satisfy $0 < c_1 < c_2 < 1$. We say that the step t satisfies the Armijo condition at iteration k if

$$A(t): \quad f(x_k + td_k) \leq f(x_k) + c_1 t \nabla f(x_k)^T d_k \tag{2}$$

and that it satisfies the Wolfe condition if[1]

$$W(t): \quad f \text{ is differentiable at } x_k + td_k \quad \text{with} \quad \nabla f(x_k + td_k)^T d_k \geq c_2 \nabla f(x_k)^T d_k. \quad (3)$$

The condition $0 < c_1 < c_2 < 1$ ensures that points t satisfying $A(t)$ and $W(t)$ exist, as is well known in the convex case and the smooth case; for more general f, see [16]. The results of this section are independent of any choice of line search to generate such points. Note that as long as f is differentiable at the initial iterate, defining subsequent iterates by $x_{k+1} = x_k + t_k d_k$, where $W(t)$ holds for $t = t_k$, ensures that f is differentiable at all x_k.

We now restrict our attention to f defined by (1), with

$$d_k = -\nabla f(x_k) = -\begin{bmatrix} \operatorname{sgn}(x_k^{(1)})a \\ \mathbb{1} \end{bmatrix}, \quad (4)$$

where $\mathbb{1} \in \mathbb{R}^{n-1}$ denotes the vector of all ones. We have

$$f(x_k + td_k) = a\left|x_k^{(1)} - \operatorname{sgn}(x_k^{(1)})at\right| + \sum_{i=2}^{n} x_k^{(i)} - (n-1)t.$$

We assume that $a \geq \sqrt{n-1}$, so that f is bounded below along the negative gradient direction as $t \to \infty$. Hence, $x_{k+1} = x_k + t_k d_k$ satisfies

$$x_{k+1}^{(1)} = x_k^{(1)} - \operatorname{sgn}(x_k^{(1)})at_k \quad \text{and} \quad x_{k+1}^{(i)} = x_k^{(i)} - t_k \quad \text{for } i = 2, \ldots, n. \quad (5)$$

We have

$$\nabla f(x_k)^T d_k = -(a^2 + n - 1) \quad (6)$$

and

$$\nabla f(x_k + t_k d_k)^T d_k = -(a^2 \operatorname{sgn}(x_{k+1}^{(1)}) \operatorname{sgn}(x_k^{(1)}) + n - 1). \quad (7)$$

For clarity we summarize the underlying assumptions that apply to all the results in this section.

Assumption 2.1: Let f be defined by (1) with $a \geq \sqrt{n-1}$ and define $x_{k+1} = x_k + t_k d_k$, with $d_k = -\nabla f(x_k)$, for some step t_k, $k = 1, 2, 3, \ldots$, where x_0 is arbitrary provided that $x_0^{(1)} \neq 0$.

Lemma 2.1: *The Armijo condition $A(t_k)$ (i.e. (2) with $t = t_k$), is equivalent to*

$$c_1 t_k (a^2 + n - 1) \leq f(x_k) - f(x_{k+1}) \quad (8)$$

and the Wolfe condition $W(t_k)$ (i.e. (3) with $t = t_k$) is equivalent to each of the following three conditions:

$$\operatorname{sgn}(x_{k+1}^{(1)}) = -\operatorname{sgn}(x_k^{(1)}), \quad (9)$$

$$t_k > \frac{|x_k^{(1)}|}{a} \quad (10)$$

and

$$at_k = |x_{k+1}^{(1)} - x_k^{(1)}| = |x_k^{(1)}| + |x_{k+1}^{(1)}|. \tag{11}$$

Proof: These all follow easily from (5), (6) and (7), using $c_2 < 1$ and $a \geq \sqrt{n-1}$. ■

Thus, t_k satisfies the Wolfe condition if and only if the iterates x_k oscillate back and forth across the $x^{(1)} = 0$ axis.[2]

Theorem 2.2: *Suppose t_k satisfies $A(t_k)$ and $W(t_k)$ for $k = 1, 2, 3, \ldots, N$ and define $S_N = \sum_{k=0}^{N-1} t_k$. Then*

$$c_1(a^2 + n - 1)S_N \leq f(x_0) - f(x_N) \leq (n-1)S_N + a|x_0^{(1)}|, \tag{12}$$

so that S_N is bounded above as $N \to \infty$ if and only if $f(x_N)$ is bounded below. Furthermore, $f(x_N)$ is bounded below if and only if x_N converges to a point $\bar{x}$ with $\bar{x}^{(1)} = 0$.

Proof: Summing up (8) from $k = 0$ to $k = N-1$ we have

$$c_1(a^2 + n - 1)S_N \leq f(x_0) - f(x_N). \tag{13}$$

Using (5) we have

$$x_0^{(i)} - x_N^{(i)} = \sum_{k=0}^{N-1}(x_k^{(i)} - x_{k+1}^{(i)}) = S_N \quad \text{for } i = 2, \ldots, n,$$

so

$$f(x_0) - f(x_N) = a|x_0^{(1)}| - a|x_N^{(1)}| + (n-1)S_N,$$

using (1). Combining this with (13) and dropping the term $a|x_N^{(1)}|$ we obtain (12), so S_N is bounded above if and only if $f(x_N)$ is bounded below. Now suppose that $f(x_N)$ is bounded below and hence S_N is bounded above, implying that $t_N \to 0$, and therefore, from (11), that $x_N^{(1)} \to 0$. Since $f(x_N) = a|x_N^{(1)}| + \sum_{i=2}^{n-1} x_N^{(i)}$ is bounded below as $N \to \infty$, and since, from (5), for $i = 2, \ldots, n$, each $x_N^{(i)}$ is decreasing as N increases, we must have that each $x_N^{(i)}$ converges to a limit $\bar{x}^{(i)}$. On the other hand, if x_N converges to a point $(0, \bar{x}^{(2)}, \ldots, \bar{x}^{(n)})$ then $f(x_N)$ is bounded below by $\sum_{i=2}^{n-1} \bar{x}^{(i)}$. ■

Note that, as f is unbounded below, convergence of x_N to a point $(0, \bar{x}^{(2)}, \ldots, \bar{x}^{(n)})$ should be interpreted as failure of the method.

We next observe that, because of the bounds (12), it is not possible that $S_N \to \infty$ if

$$a > \sqrt{(n-1)(1/c_1 - 1)}$$

(in addition to $a \geq \sqrt{n-1}$ as required by Assumption 2.1).

It will be convenient to define

$$\tau = c_1 + \frac{(n-1)(c_1 - 1)}{a^2}. \tag{14}$$

Since $c_1 \in (0, 1)$ and $a \geq \sqrt{n-1}$, we have $-1 < -1 + 2c_1 < \tau < c_1 < 1$, with $\tau > 0$ equivalent to $c_1(a^2 + n - 1) > n - 1$.

Corollary 2.3: *Suppose $A(t_k)$ and $W(t_k)$ hold for all k. If $\tau > 0$ then $f(x_k)$ is bounded below as $k \to \infty$.*

Proof: This is now immediate from (12) and the definition of τ. ■

So, the larger a is, the smaller the Armijo parameter c_1 must be in order to have $\tau \leq 0$ and therefore the possibility that $f(x_k) \to -\infty$.

At this point it is natural to ask whether $\tau \leq 0$ implies that $f(x_k) \to -\infty$. We will see in the next section (in Corollary 3.4, for $\tau = 0$) that the answer is no. However, we can show that there is a specific choice of t_k satisfying $A(t_k)$ and $W(t_k)$ for which $\tau \leq 0$ implies $f(x_k) \to -\infty$. We start with a lemma.

Lemma 2.4: *Suppose $W(t_k)$ holds. Then $A(t_k)$ holds if and only if*

$$(1+\tau)\frac{at_k}{2} \leq |x_k^{(1)}|. \tag{15}$$

Proof: Suppose $x_k^{(1)} > 0$. Since $W(t_k)$ holds, using (9), we can rewrite the Armijo condition (8) as

$$\begin{aligned} c_1 t_k(a^2+n-1) &\leq f(x_k) - f(x_{k+1}) \\ &= \left(ax_k^{(1)} + \sum_{i=2}^{n} x_k^{(i)}\right) - \left(-a(x_k^{(1)} - at_k) + \sum_{i=2}^{n} x_k^{(i)} - (n-1)t_k\right) \\ &\Leftrightarrow t_k\Big(c_1(a^2+n-1) + a^2 - (n-1)\Big) \leq 2ax_k^{(1)} \\ &\Leftrightarrow t_k a^2(\tau+1) \leq 2ax_k^{(1)}, \end{aligned}$$

giving (15). A similar argument applies when $x_k^{(1)} < 0$. ■

Theorem 2.5: *Let*

$$t_k = \frac{2|x_k^{(1)}|}{(\tau+1)a}. \tag{16}$$

Then

(1) *$A(t_k)$ and $W(t_k)$ both hold.*
(2) *if $\tau \leq 0$, then $f(x_k)$ is unbounded below as $k \to \infty$.*

Proof: The first statement follows immediately from (10) (since $|\tau| < 1$) and Lemma 2.4. Furthermore, (11) allows us to write (15) equivalently as

$$(1+\tau)|x_{k+1}^{(1)}| \leq (1-\tau)|x_k^{(1)}|. \tag{17}$$

Since t_k is the maximum steplength satisfying (15), it follows that (17) holds with equality, so $|x_{k+1}^{(1)}| = C|x_k^{(1)}|$, where

$$C = \frac{1-\tau}{1+\tau},$$

and hence

$$|x_{k+1}^{(1)}| = C^{k+1}|x_0^{(1)}|.$$

Then, we can rewrite (16) as

$$t_k = \frac{2C^k|x_0^{(1)}|}{a(\tau+1)}.$$

When $-1 < \tau \leq 0$, we have $C \geq 1$, so $S_N = \sum_{k=0}^{N-1} t_k \to \infty$ as $N \to \infty$ and hence, by Theorem 2.2, $f(x_N) \to -\infty$. ■

3. Additional results depending on a specific choice of Armijo–Wolfe line search

In this section we continue to assume that f and d_k are defined by (1) and (4), respectively, with $a \geq \sqrt{n-1}$, and that $A(t)$ and $W(t)$ are defined as earlier. However, unlike in the previous section, we now assume that t_k is generated by a specific line search, namely the one given in Algorithm 1, which is taken from [16, p. 147] and is a specific realization of the line searches described implicitly in [22] and explicitly in [12]. Since the line search function $s \mapsto f(x_k + sd_k)$ is locally Lipschitz and bounded below, it follows, as shown in [16], that at any stage during the execution of Algorithm 1, the interval $[\alpha, \beta]$ must always contain a set of points t with non-zero measure satisfying $A(t)$ and $W(t)$, and furthermore, the line search must terminate at such a point. This defines the point t_k. A crucial aspect of Algorithm 1 is that, in the 'while' loop, the Armijo condition is tested first and the Wolfe condition is then tested only if the Armijo condition holds.

We already know from Theorem 2.2 and Corollary 2.3 that, for *any* set of Armijo–Wolfe points, if $\tau > 0$, then $f(x_N)$ is bounded below. In this section we analyse the case $\tau \leq 0$, assuming that the steps t_k are generated by the Armijo–Wolfe bracketing line search. It simplifies the discussion to make a probabilistic analysis, assuming that $x_0 = (x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(n)})$ is generated randomly, say from the normal distribution. Clearly, all intermediate values t generated by Algorithm 1 are rational, and with probability one all corresponding points $x = (x_0^{(1)} - \operatorname{sgn}(x_0^{(1)})at, x_0^{(2)} - t, \ldots, x_0^{(n)} - t)$ where the Armijo and Wolfe conditions are tested during the first line search are irrational (this is obvious if a is rational but it also holds if a is irrational assuming that x_0 is generated independently of a). It follows that, with probability one, f is differentiable at these points, which include the next iterate $x_1 = (x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(n)})$. It is clear that, by induction, the points $x_k = (x_k^{(1)}, x_k^{(2)}, \ldots, x_k^{(n)})$ are irrational with probability one for all k, and in particular, $x_k^{(1)}$ is non-zero for all k and hence f is differentiable at all points x_k.

Let us summarize the underlying assumptions for all the results in this section.

Assumption 3.1: Let f be defined by (1), with $a \geq \sqrt{n-1}$, and define $x_{k+1} = x_k + t_k d_k$, with $d_k = -\nabla f(x_k)$, and with t_k defined by Algorithm 1, $k = 1, 2, 3, \ldots$, where $x_k =$

Algorithm 1 (Armijo–Wolfe Bracketing Line Search)

```
$\alpha \leftarrow 0$
$\beta \leftarrow +\infty$
$t \leftarrow 1$
while true do
    if $A(t)$ fails (see (2)) then
        $\beta \leftarrow t$
    else if $W(t)$ fails (see (3)) then
        $\alpha \leftarrow t$
    else
        stop and return $t$
    end if
    if $\beta < +\infty$ then
        $t \leftarrow (\alpha + \beta)/2$
    else
        $t \leftarrow 2\alpha$
    end if
end while
```

$(x_k^{(1)}, x_k^{(2)}, \dots, x_k^{(n)})$, and $x_0 = (x_0^{(1)}, x_0^{(2)}, \dots, x_0^{(n)})$ is randomly generated from the normal distribution. All statements in this section are understood to hold with probability one.

Lemma 3.1: *Suppose $\tau \le 0$ and suppose $|x_k^{(1)}| > a$. Define*

$$r_k = \left\lceil \log_2 \frac{|x_k^{(1)}|}{a} \right\rceil \quad \text{so that} \quad a2^{r_k - 1} < |x_k^{(1)}| < a2^{r_k}. \tag{18}$$

Then, $t_k = 2^{r_k}$.

Proof: Since $|x_k^{(1)}| > a$ any steplength $t \le |x_k^{(1)}|/a$ satisfies $A(t)$ but fails $W(t)$. Starting with $t = 1$, the 'while' loop in Algorithm 1 will carry out r_k doublings of t until $t > |x_k^{(1)}|/a$, i.e. $W(t)$ holds. Hence, in the beginning of stage $r_k + 1$, we have $\alpha = 2^{r_k - 1}$ (a lower bound on t_k), $t = 2^{r_k}$ and $\beta = +\infty$. At this point, t satisfies $W(t)$ and since $\tau \le 0$, it also satisfies (15), i.e. $A(t)$. So $t_k = 2^{r_k}$. ■

Theorem 3.2: *Suppose $\tau \le 0$ and $|x_0^{(1)}| > a$. Then after $j \le r_0$ iterations we have $|x_j^{(1)}| < a$, where r_0 is defined by (18), and furthermore, for all subsequent iterations, the condition $|x_k^{(1)}| < a$ continues to hold.*

Proof: For any k with $|x_k^{(1)}| > a$ we know from the previous lemma that $t_k = 2^{r_k}$ with $r_k > 0$. From (11) and (18) we get

$$|x_{k+1}^{(1)}| = at_k - |x_k^{(1)}| < a2^{r_k} - a2^{r_k - 1} = a2^{r_k - 1}. \tag{19}$$

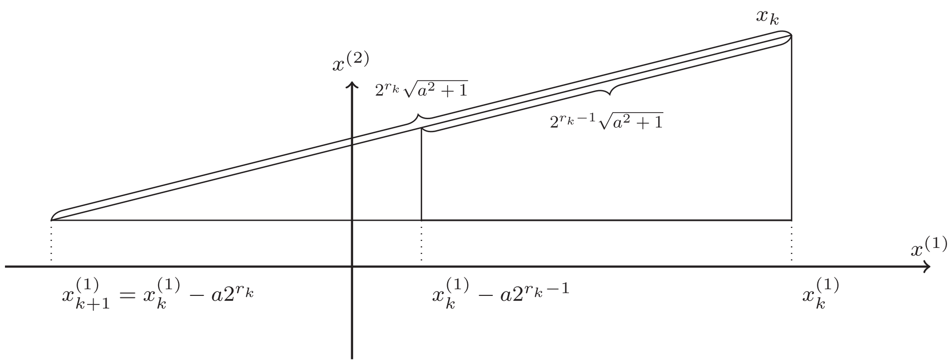


Figure 1. Doubling t in order to satisfy $W(t)$.

See Figure 1 for an illustration with $n = 2$, with $x_k^{(1)} > 0$, so $-a2^{r_k-1} < x_{k+1}^{(1)} < 0$. Hence, either $|x_{k+1}^{(1)}| < a$, or $a < |x_{k+1}^{(1)}| < a2^{r_k-1}$, in which case from (18) and (19) we have

$$r_{k+1} \leq r_k - 1.$$

So, beginning with $k = 0$, r_k is decremented by at least one at every iteration until $|x_k^{(1)}| < a$. Finally, once $|x_k^{(1)}| < a$ holds, it follows that the initial step $t = 1$ satisfies the Wolfe condition $W(t)$, and hence, if $A(t)$ also holds, t_k is set to one, while if not, the upper bound β is set to one so $t_k < 1$. Hence, the next value $x_{k+1}^{(1)} = x_k^{(1)} - \operatorname{sgn}(x_k^{(1)})at_k$ also satisfies $|x_{k+1}^{(1)}| < a$. ■

Theorem 3.2 shows that for any $\tau \leq 0$ and sufficiently large k using Algorithm 1 we always have $|x_k^{(1)}| < a$. In the reminder of this section we provide further details on the step t_k generated when $|x_k^{(1)}| < a$. In this case, the initial step $t = 1$ satisfies $W(t)$ but not necessarily $A(t)$. So Algorithm 1 will repeatedly halve t, until it satisfies $A(t)$. See Figure 2 for an illustration.

Suppose for the time being that $\tau = 0$ and define p_k by

$$p_k = \left\lceil \log_2 \frac{a}{|x_k^{(1)}|} \right\rceil \quad \text{so that} \quad \frac{a}{2^{p_k}} < |x_k^{(1)}| < \frac{a}{2^{p_k-1}}. \tag{20}$$

For example, in Figure 2, $p_k = 2$. So, $a/4 < |x_k^{(1)}| < a/2$. Hence $t = 1/2$ satisfies $W(t)$. In fact it also satisfies $A(t)$, because for $\tau = 0$, we have

$$\frac{(1+\tau)at}{2} = \frac{a}{4} < |x_k^{(1)}|,$$

which is exactly the Armijo condition (15). So, Algorithm 1 returns $t_k = 1/2$.

On the other hand if we had $\tau \leq -1/2$, $t = 1$ would have satisfied the Armijo condition (15) since

$$\frac{(1+\tau)a}{2} \leq \frac{a}{4} < |x_k^{(1)}|.$$

By taking τ into the formulation we are able to compute the exact value of t_k in the following theorem.

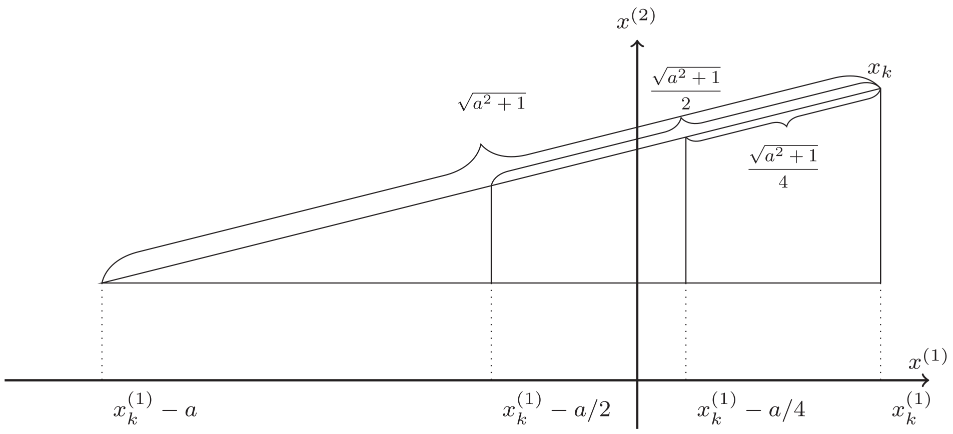


Figure 2. Halving t in order to satisfy $A(t)$.

Theorem 3.3: *Suppose $\tau \leq 0$ and $|x_k^{(1)}| < a$. Then $t_k = \min(1, 1/2^{q_k-1})$, where*

$$q_k = \left\lceil \log_2 \frac{(1+\tau)a}{|x_k^{(1)}|} \right\rceil,$$

so

$$\frac{(1+\tau)a}{2^{q_k}} < |x_k^{(1)}| < \frac{(1+\tau)a}{2^{q_k-1}}. \tag{21}$$

Note that, unlike r_k and p_k, the quantity q_k could be zero or negative.

Proof: If $|x_k^{(1)}| > (1+\tau)a/2$, then $t = 1$ satisfies the Armijo condition (15) as well as the Wolfe condition, so t_k is set to 1. Otherwise, $q_k > 1$, so $1/2^{q_k-1} < 1$ and Algorithm 1 repeatedly halves t until $A(t)$ holds. We now show that the first t that satisfies $A(t)$ is such that $|x_k^{(1)}| < at$, i.e. it satisfies $W(t)$ as well. Since $\tau \leq 0$, the second inequality in (21) proves that steplength $t = 1/2^{q_k-1}$ satisfies $W(t)$. Moreover, the first inequality is the Armijo condition (15) with the same steplength. Furthermore, the second inequality in (21) also shows that $t' = 2t = 1/2^{q_k-2}$ is too large to satisfy the Armijo condition (15). Hence $t = 1/2^{q_k-1}$ is the first steplength satisfying both $A(t)$ and $W(t)$. So, Algorithm 1 returns $t_k = 1/2^{q_k-1}$. ∎

Note that if $\tau = 0$, p_k and q_k coincide, with $p_k \geq 1$ since $|x_k^{(1)}| < a$, and hence $t_k = 1/2^{p_k-1} \leq 1$. Furthermore, $p_k = 1$ and hence $t_k = 1$ when $a/2 < |x_k^{(1)}| < a$.

Corollary 3.4: *Suppose $\tau = 0$. Then x_k converges to a limit $\bar{x}$ with $\bar{x}^{(1)} = 0$.*

Proof: Assume that k is sufficiently large so that $|x_k^{(1)}| < a$. From (20) we have $a/2^{p_k} < |x_k^{(1)}|$. Using Theorem 3.3 we have $t_k = 1/2^{p_k-1}$ and therefore

$$|x_{k+1}^{(1)}| = at_k - |x_k^{(1)}| < \frac{a}{2^{p_k-1}} - \frac{a}{2^{p_k}} = \frac{a}{2^{p_k}}$$

(see Figure 2 for an illustration). So $p_{k+1} \geq p_k + 1$. Using Theorem 3.3 again we conclude $t_{k+1} \leq 1/2^{p_k}$ and so $t_{k+1} \leq t_k/2$. The same argument holds for all subsequent iterates so $S_N = \sum_{k=0}^{N-1} t_k$ is bounded above as $N \to \infty$. The result therefore follows from Theorem 2.2. ■

Corollary 3.5: *If $\tau \leq -0.5$ then eventually $t_k = 1$ at every iteration, and $f(x_k) \to -\infty$.*

Proof: As we showed in Theorem 3.2, for sufficiently large k, $|x_k^{(1)}| < a$ and therefore $t = 1$ always satisfies the Wolfe condition, so $t_k \leq 1$. If $|x_k^{(1)}| > (1+\tau)a/2$, then $t = 1$ also satisfies the Armijo condition (15), so $t_k = 1$. If $|x_{k+1}^{(1)}| > (1+\tau)a/2$ as well, then $t_{k+1} = 1$ and hence $x_{k+2}^{(1)} = x_k^{(1)}$. It follows that $t_j = 1$ for all $j > k+1$. Hence, by Theorem 2.2, $f(x_k) \to -\infty$. Otherwise, suppose $|x_k^{(1)}| < (1+\tau)a/2$ (in case $|x_k^{(1)}| > (1+\tau)a/2$ and $|x_{k+1}^{(1)}| < (1+\tau)a/2$ just shift the index by one so that we have $|x_{k-1}^{(1)}| > (1+\tau)a/2$ and $|x_k^{(1)}| < (1+\tau)a/2$).

Since $|x_k^{(1)}| < (1+\tau)a/2$, from the definition of q_k in (21) we conclude that $2 \leq q_k$, i.e. $1/2^{q_k-1} \leq 1/2$, so from Theorem 3.3 we have $t_k = 1/2^{q_k-1} \leq 1/2$. Since $|x_k^{(1)}| < (1+\tau)a/2^{q_k-1}$ and $1+\tau \leq 1/2$ we have

$$|x_k^{(1)}| < \frac{a}{2^{q_k}}. \tag{22}$$

So by (11)

$$|x_{k+1}^{(1)}| = at_k - |x_k^{(1)}| \geq \frac{a}{2^{q_k-1}} - \frac{a}{2^{q_k}} = \frac{a}{2^{q_k}} > \frac{(1+\tau)a}{2^{q_k-1}} \tag{23}$$

and using (21) again we conclude $q_{k+1} \leq q_k - 1$. So,

$$t_{k+1} = \min\left(1, \frac{1}{2^{q_{k+1}-1}}\right) \geq \min\left(1, \frac{1}{2^{q_k-2}}\right) = \frac{1}{2^{q_k-2}} = 2t_k$$

and therefore, applying this repeatedly, after a finite number of iterations, say at iteration $\bar{k}$, we must have $t_{\bar{k}} = 1$ for the first time. Furthermore, from (22) and (23) we have $|x_k^{(1)}| < |x_{k+1}^{(1)}|$, and applying this repeatedly as well we have $|x_{\bar{k}}^{(1)}| < |x_{\bar{k}+1}^{(1)}|$. From the Armijo condition (15) at iteration $\bar{k}$ we have $(1+\tau)a/2 \leq |x_{\bar{k}}^{(1)}|$ and therefore

$$\frac{(1+\tau)a}{2} < |x_{\bar{k}+1}^{(1)}|.$$

Hence, $t = 1$ also satisfies the Armijo condition (15) at iteration $\bar{k}+1$. With $t_{\bar{k}} = 1$ and $t_{\bar{k}+1} = 1$, we conclude $x_{\bar{k}+2}^{(1)} = x_{\bar{k}}^{(1)}$. It follows that $t_j = 1$ for all $j > \bar{k}+1$. Hence $f(x_k) \to -\infty$ by Theorem 2.2. ■

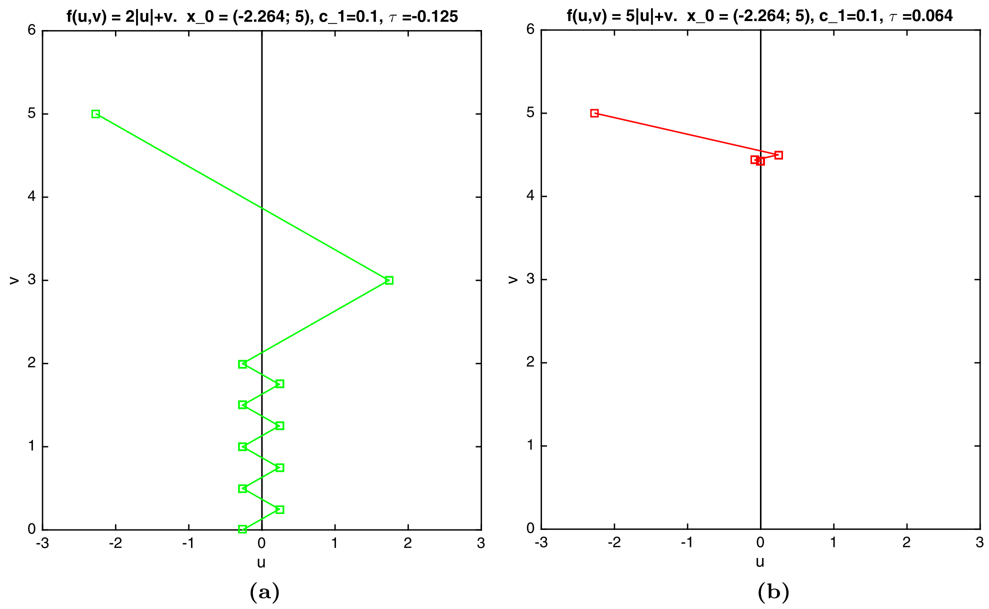


Figure 3. Minimizing f with $n = 2$, $u = x^{(1)}$, $v = x^{(2)}$ and $c_1 = 0.1$. *Left*, with $a = 2$, so $\tau < 0$ and $f(u_k, v_k) \to -\infty$ (success). *Right*, with $a = 5$, so $\tau > 0$ and $(u_k, v_k) \to (0, \bar{v})$ (failure).

4. Experimental results

In this section we again continue to assume that f and d_k are defined by (1) and (4), respectively. For simplicity we also assume that $n = 2$, writing $u = x^{(1)}$ and $v = x^{(2)}$ for convenience. Our experiments confirm the theoretical results presented in the previous sections and provide some additional insight. We know from Theorem 2.2 that when the gradient algorithm fails, i.e. x_k converges to a point $(0, \bar{v})$, the step t_k converges to zero. However, an implementation of Algorithm 1 in floating point arithmetic must terminate the 'while' loop after it executes a maximum number of times. We used the MATLAB implementation in hanso,[3] which limits the number of bisections in the 'while' loop to 30.

Figure 3 shows two examples of minimizing f with $a = 2$ and $a = 5$, with $c_1 = 0.1$ in both cases, and hence with $\tau < 0$ and $\tau > 0$, respectively. Starting from the same randomly generated point, we have $f(x_k) \to -\infty$ (success) when $\tau < 0$ and $x_k \to (0, \bar{v})$ (failure) when $\tau > 0$.

For various choices of a and c_1 we generated 5000 starting points $x_0 = (u_0, v_0)$, each drawn from the normal distribution with mean 0 and variance 1, and measured how frequently 'failure' took place, meaning that the line search failed to find an Armijo–Wolfe point within 30 bisections. If failure did not take place within 50 iterations, i.e. with $k \leq 50$, we terminated the gradient method declaring success. Figure 4 shows the failure rates when (top) c_1 is fixed to 0.05 and a is varied and (bottom) when $a = \sqrt{2}$ and c_1 is varied. Both cases confirm that when $\tau > 0$ the method always fails, as predicted by Corollary 2.3, while when $\tau \leq -0.5$, failure does not occur, as shown in Corollary 3.5.

As Figure 4 shows, when $\tau < 0$ with $|\tau|$ small, the method may or may not fail, with failure more likely the closer τ is to zero. Further experiments for three specific values

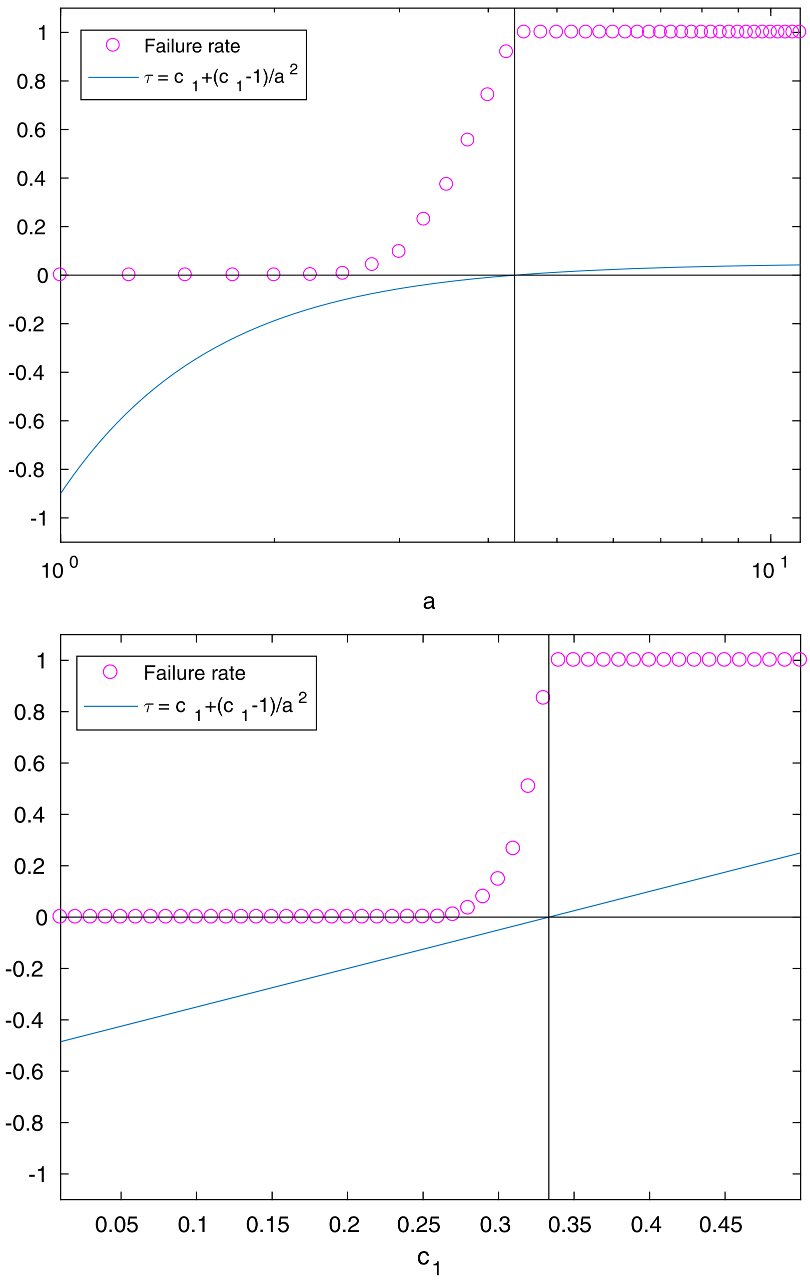


Figure 4. Failure rates (small circles) for f with $n = 2$ when (top) c_1 is fixed to 0.05 and a is varied and (bottom) a is fixed to $\sqrt{2}$ and c_1 is varied. The solid curves show the value of τ. Each experiment was repeated 5000 times.

of τ, namely $-0.1, -0.01$ and -0.001, using a fixed value of $c_1 = 0.05$ and a defined by $a = \sqrt{(1 - c_1)/(c_1 - \tau)}$, confirmed that failure is more likely the closer that τ gets to zero and also showed that the set of initial points from which failure takes place is complex; see Figure 5. The initial points were drawn uniformly from the box $(-100, 100) \times (-100, 100)$.

We know from Corollary 3.4 that, for $\tau = 0$, with probability one $t_k \to 0$, so even if high precision were being used, for sufficiently large k an implementation in floating point must fail. It may well be the case that failures for $\tau < 0$ occur only because of the limited precision being used, and that with sufficiently high precision, these failures would be eliminated. This suggestion is supported by experiments done reducing the maximum number of bisections to 15, for which the number of failures for $\tau < 0$ increased significantly, and increasing it to 50, for which the number of failures decreased significantly.

5. Relationship with convergence results for subgradient methods

Let h be any convex function. The subgradient method [3,24] applied to h is a generalization of the gradient method, where h is not assumed to be differentiable at the iterates $\{x_k\}$ and hence, instead of setting $-d_k = \nabla h(x_k)$, one defines $-d_k$ to be any element of the subdifferential set

$$\partial h(x_k) = \left\{ g : h(x_k + z) \geq h(x_k) + g^T z \,\forall\, z \in \mathbb{R}^n \right\}.$$

The steplength t_k in the subgradient method is not determined dynamically, as in an Armijo–Wolfe line search, but according to a predetermined rule. The advantages of the subgradient method with predetermined steplengths are that it is robust, has low iteration cost, and has a well established convergence theory that does not require h to be differentiable at the iterates $\{x_k\}$, but the disadvantage is that convergence is usually slow. Provided h is differentiable at the iterates, the subgradient method reduces to the gradient method with the same step sizes, but it is not necessarily the case that f decreases at each iterate.

We cannot apply the convergence theory of the subgradient method directly to our function f defined in (1), because f is not bounded below. However, we can argue as follows. Suppose that $\tau > 0$, so that we know (by Corollary 2.3) that for all x_0 with $x_0^{(1)} \neq 0$, the iterates x_k generated by the gradient method with Armijo–Wolfe steplengths applied to f converge to a point $\bar{x}$ with $\bar{x}^{(1)} = 0$. Fix any initial point x_0 with $x_0^{(1)} \neq 0$, and let $M = f(\bar{x})$, where $\bar{x}$ is the resulting limit point (to make this well defined, we can assume that the Armijo–Wolfe bracketing line search of Section 3 is in use). Now define

$$\tilde{f}(x) = \max\left(M - 1, a|x^{(1)}| + \sum_{i=2}^{n} x^{(i)} \right).$$

Clearly, the iterates generated by the gradient method with Armijo–Wolfe steplengths initiated at x_0 are identical for f and $\tilde{f}$, with f (equivalently, $\tilde{f}$) differentiable at all iterates $\{x_k\}$, and with $f(x_k) = \tilde{f}(x_k) \to M$. Furthermore, the theory of subgradient methods applies to $\tilde{f}$. One well-known result states that provided the steplengths $\{t_k\}$ are square-summable (that is, $\sum_{k=0}^{\infty} t_k^2 < \infty$, and hence the steps are 'not too long'), but not summable (that is, $\sum_{k=0}^{\infty} t_k = \infty$, and hence the steps are 'not too short'), then convergence of $\tilde{f}(x_k)$ to the optimal value $M-1$ must take place [18]. Since this does *not* occur, we conclude that the Armijo–Wolfe steplenths $\{t_k\}$ do *not* satisfy these conditions. Indeed, the 'not summable' condition is exactly the condition $S_N \to \infty$, where $S_N = \sum_{k=0}^{N-1} t_k$, and Theorem 2.2 established that the converse, that S_N is bounded above, is equivalent to the function values

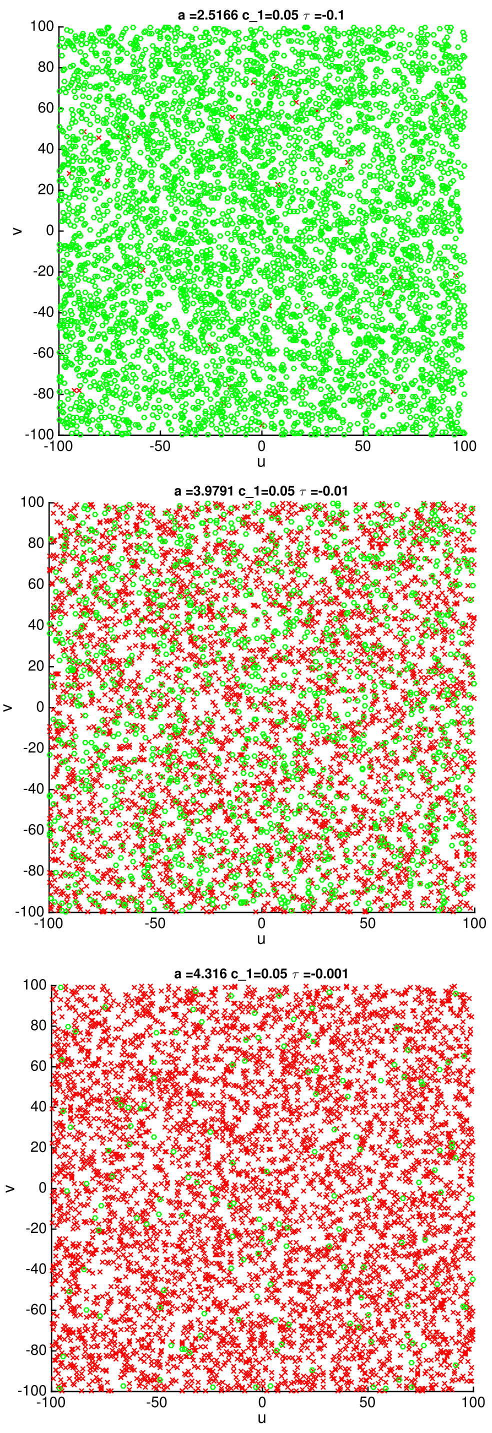


Figure 5. Mixed success and failure when $\tau = -0.1$ (top), $\tau = -0.01$ (middle) and $\tau = -0.001$ (bottom). Each plot shows 5000 points. The green circles show starting points for which the method succeeded, generating $x_k = (u_k, v_k) \in \mathbb{R}^2$ for which $f(x_k)$ is apparently unbounded below, while the red crosses show starting points for which the method failed, generating x_k converging to a point on the v-axis.

$f(x_k)$ being bounded below. This, then, is consistent with the convergence theory for the subgradient method, which says that the steps must not be 'too short'; in the context of an Armijo–Wolfe line search, when c_1 is not sufficiently small, and hence $\tau > 0$, the Armijo condition is too restrictive: it is causing the $\{t_k\}$ to be 'too short' and hence summable.

Of course, in practice, one usually optimizes functions that are bounded below, but one hopes that a method applied to a convex function that is not bounded below will not converge, but will generate points x_k with $f(x_k) \to -\infty$. The main contribution of our paper is to show that, in fact, this does not happen for a simple well known method on a simple convex non-smooth function, *regardless of the starting point,* unless the Armijo parameter is chosen to be sufficiently small — how small, one does not know without advance information on the properties of f.

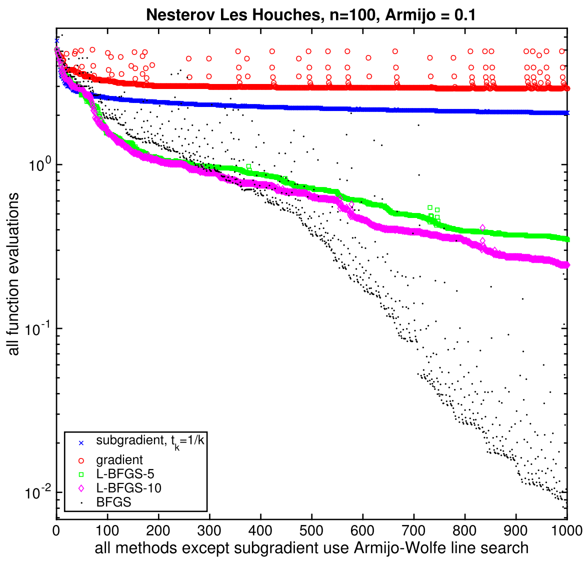
6. Concluding remarks

Should we conclude from the results of this paper that, if the gradient method with an Armijo–Wolfe line search is applied to a non-smooth function, the Armijo parameter c_1 should be chosen to be small? Results for a very ill-conditioned convex non-smooth function $\hat{f}$ devised by Nesterov [20] suggest that the answer is yes. The function is defined by

$$\hat{f}(x) = \max\{|x_1|, |x_i - 2x_{i-1}|, i = 2, \dots, n\}.$$

Let $\hat{x}_1 = 1, \hat{x}_i = 2\hat{x}_{i-1} + 1, i = 2, \dots, n$. Then $\hat{f}(\hat{x}) = 1 = \hat{f}(\mathbb{1})$ although $\|\hat{x}\|_\infty \approx 2^n$ and $\|\mathbb{1}\|_\infty = 1$, so the level sets of $\hat{f}$ are very ill conditioned. The minimizer is $x = 0$ with $\hat{f}(x) = 0$. Figure 6 shows function values computed by applying five different methods to minimize $\hat{f}$ with $n = 100$. The five methods are: the subgradient method with $t_k = 1/k$, a square-summable but not summable sequence that guarantees convergence; the gradient method using the Armijo–Wolfe bracketing line search of Section 3; the limited memory BFGS method [21] with 5 and 10 updates, respectively (using 'scaling'); and the full BFGS method [16,21]; the BFGS variants also use the same Armijo–Wolfe line search.[4] The top and bottom plots in Figure 6 show the results when the Armijo parameter c_1 is set to 0.1 and to 10^{-6}, respectively. The Wolfe parameter was set to 0.5 in both cases. These values were chosen to satisfy the usual requirement that $0 < c_1 < c_2 < 1$, while still ensuring that c_1 is not so tiny that it is effectively zero in floating point arithmetic. All function values generated by the methods are shown, including those evaluated in the line search. The same initial point, generated randomly, was used for all methods; the results using other initial points were similar.

For this particular example, we see that, in terms of reduction of the function value within a given number of evaluations, the gradient method with the Armijo–Wolfe line search when the Armijo parameter is set to 10^{-6} performs better than using the subgradient method's predetermined sequence $t_k = 1/k$, but that this is not the case when the Armijo parameter is set to 0.1. The smaller value allows the gradient method to take steps with $t_k = 1$ early in the iteration, leading to rapid progress, while the larger value forces shorter steps, quickly leading to stagnation. Eventually, even the small Armijo parameter requires many steps in the line search — one can see that on the right side of the lower figure, at least 8 function values per iteration are required. One should not read too much into the results for one example, but the most obvious observation from Figure 6 is that the



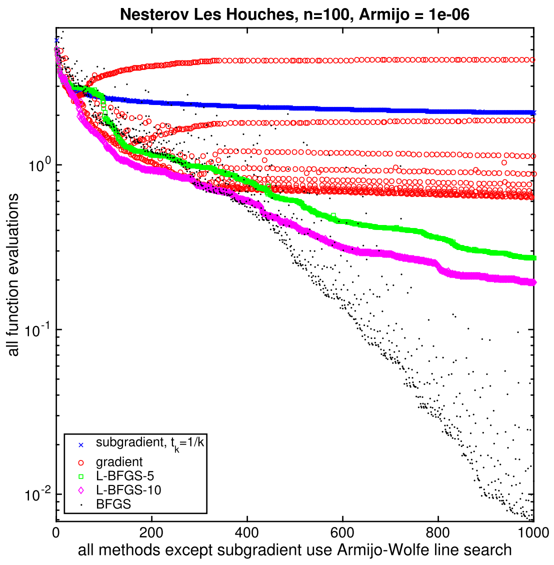


Figure 6. Comparison of five methods for minimizing Nesterov's ill-conditioned convex nonsmooth function $\hat{f}$. The subgradient method (blue crosses) uses $t_k = 1/k$. The gradient, limited-memory BFGS (with 5 and 10 updates, respectively) and full BFGS methods (red circles, green squares, magenta diamonds and black dots) all use the Armijo–Wolfe bracketing line search. All function evaluations are shown. Top: Armijo parameter $c_1 = 0.1$. Bottom: Armijo parameter $c_1 = 10^{-6}$.

full BFGS and limited memory BFGS methods are much more effective than the gradient or subgradient methods. This distinction becomes far more dramatic if we run the methods for more iterations: BFGS is typically able to reduce $\hat{f}$ to about 10^{-12} in about 5000 function evaluations, while the gradient and subgradient methods fail to reduce $\hat{f}$ below 10^{-1} in the same number of function evaluations. The limited memory BFGS methods consistently perform better than the gradient/subgradient methods but worse than full BFGS. The value of the Armijo parameter c_1 has little effect on the BFGS variants.

These results are consistent with substantial prior experience with applying the full BFGS method to non-smooth problems, both convex and non-convex [7,10,11,16]. However, although the BFGS method requires far fewer operations per iteration than bundle methods or gradient sampling, it is still not practical when n is large. Hence, the attraction of limited-memory BFGS which, like the gradient and subgradient methods, requires only $O(n)$ operations per iteration. In a companion paper [2], we investigate under what conditions the limited-memory BFGS method applied to the function f studied in this paper generates iterates that converge to a non-optimal point, and, more generally, how reliable a choice it is for non-smooth optimization.

Notes

1. There is a subtle distinction between the Wolfe condition given here and that given in [16], since here the Wolfe condition is understood to fail if the gradient of f does not exist at $x_k + td_k$, while in [16] it is understood to fail if the function of one variable $s \mapsto f(x_k + sd_k)$ is not differentiable at $s = t$. For the example analysed here, these conditions are equivalent.
2. The same oscillatory behaviour occurs if we replace the Wolfe condition by the Goldstein condition $f(x_k + td_k) \geq f(x_k) + c_2 t \nabla f(x_k)^T d_k$.
3. www.cs.nyu.edu/overton/software/hanso/
4. In our implementation, we made no attempt to determine whether $\hat{f}$ is differentiable at a given point or not. This is essentially impossible in floating point arithmetic, but as noted earlier, the gradient is defined at randomly generated points with probability one; there is no reason to suppose that any of the methods tested will generate points where $\hat{f}$ is not differentiable, except in the limit, and hence the 'subgradient' method actually reduces to the gradient method with $t_k = 1/k$. See [16] for further discussion.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

Michael L. Overton was supported in part by National Science Foundation [grant number DMS-1620083].

Notes on contributors

Azam Asl obtained her B.Sc. in Computer Engineering at Sharif University of Technology in Teheran in 2008. She will receive her Ph.D. in Computer Science from New York University in 2020.

Michael L. Overton obtained his Ph.D. in Computer Science from Stanford University in 1979. He is a Silver Professor of Computer Science and Mathematics at the Courant Institute of Mathematical Sciences, New York University.

ORCID

Azam Asl http://orcid.org/0000-0001-5865-9623
Michael L. Overton http://orcid.org/0000-0002-6563-6371

References

[1] L. Armijo, *Minimization of functions having Lipschitz continuous first partial derivatives*, Pacific J. Math. 16 (1966), pp. 1–3. Available at http://projecteuclid.org/euclid.pjm/1102995080. MR 0191071.

[2] A. Asl and M.L. Overton, *Analysis of limited-memory BFGS on a class of nonsmooth convex functions*, IMA J. Numer. Anal. (2019). Available at arXiv:1810.00292.

[3] D. Bertsekas, *Nonlinear Programming*, 2nd ed., Athena Scientific, Nashua, NH, 1999.

[4] J.V. Burke, A.S. Lewis, and M.L. Overton, *A robust gradient sampling algorithm for non-smooth, nonconvex optimization*, SIAM J. Optim. 15 (2005), pp. 751–779. Available at http://dx.doi.org/10.1137/030601296. MR 2142859.

[5] J.V. Burke, F.E. Curtis, A.S. Lewis, M.L. Overton, and L.E.A. Simões, *Gradient sampling methods for nonsmooth optimization*, Submitted to *Special methods for nonsmooth optimization*, A. Bagirov, M. Gaudioso, N. Karmitsa and M. Mäkelä, eds., Springer, 2018. Available at arXiv:1804.11003v1.

[6] A. Cauchy, *Méthode générale pour la résolution des systèmes d'équations simultanées*, Comp. Rend. Sci. Paris. 25 (1847), pp. 135–163.

[7] F.E. Curtis, T. Mitchell, and M.L. Overton, *A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles*, Optim. Methods Softw. 32 (2017), pp. 148–181. Available at http://dx.doi.org/10.1080/10556788.2016.1208749.

[8] V.F. Dem'janov and V.N. Malozemov, *The theory of nonlinear minimax problems*, Uspehi Mat. Nauk 26 (1971), pp. 53–104. MR 0297378.

[9] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., A Wiley-Interscience Publication, John Wiley & Sons, Ltd., Chichester, 1987. MR 955799.

[10] A. Greenbaum, A.S. Lewis, and M.L. Overton, *Variational analysis of the Crouzeix ratio*, Math. Program. 164 (2017), pp. 229–243. Available at http://dx.doi.org/10.1007/s10107-016-1083-6. MR 3661030.

[11] J. Guo and A. Lewis, *Nonsmooth variants of Powell's BFGS convergence theorem*, SIAM J. Optim. 28 (2018), pp. 1301–1311. Available at https://doi.org/10.1137/17M1121883.

[12] J.B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms. I*, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences] Vol. 305, Springer-Verlag, Berlin, 1993. MR 1261420.

[13] K.C. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, Lecture Notes in Mathematics, Vol. 1133, Springer-Verlag, Berlin, 1985, Available at http://dx.doi.org/10.1007/BFb0074500. MR 797754.

[14] K.C. Kiwiel, *C onvergence of the gradient sampling algorithm for nonsmooth nonconvex optimization*, SIAM J. Optim. 18 (2007), pp. 379–388. Available at https://doi.org/10.1137/050639673.

[15] C. Lemaréchal, *An extension of Davidon methods to non differentiable problems*, Math. Programming Stud., 1975, pp. 95–109. MR 0436586.

[16] A.S. Lewis and M.L. Overton, *Nonsmooth optimization via quasi-Newton methods*, Math. Program. 141 (2013), pp. 135–163. Available at http://dx.doi.org/10.1007/s10107-012-0514-2. MR 3097282.

[17] D.C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Program. 45 (1989), pp. 503–528. Available at https://doi.org/10.1007/BF01589116. MR 1038245.

[18] A. Nedić and D.P. Bertsekas, *Incremental subgradient methods for nondifferentiable optimization*, SIAM J. Optim. 12 (2001), pp. 109–138. Available at https://doi.org/10.1137/S1052623499362111. MR 1870588.

[19] Y. Nesterov, *Smooth minimization of non-smooth functions*, Math. Program. 103 (2005), pp. 127–152. Available at http://dx.doi.org/10.1007/s10107-004-0552-5. MR 2166537.

[20] Y. Nesterov, *Private communication* (2016). Les Houches, France.

[21] J. Nocedal and S.J. Wright, *Numerical Optimization*, 2nd ed., Springer, New York, 2006.

[22] M.J.D. Powell, *A view of unconstrained optimization*, in *Optimization in Action (Proc. Conf., Univ. Bristol, Bristol, 1975)*. Academic Press, London, 1976, pp. 117–152.

[23] M.J.D. Powell, *Some global convergence properties of a variable metric algorithm for minimization without exact line searches*, in *Nonlinear Programming*, SIAM-AMS Proc. Vol. IX, Amer. Math. Soc., Providence, 1976, pp. 53–72.

[24] N.Z. Shor, *Minimization Methods for Non-differentiable Functions*, Springer Series in Computational Mathematics, Springer, Berlin, 1985.

[25] A.B. Taylor, J.M. Hendrickx, and F. Glineur, *Exact worst-case performance of first-order methods for composite convex optimization*, SIAM J. Optim. 27 (2017), pp. 1283–1313. Available at https://doi.org/10.1137/16M108104X.

[26] P. Wolfe, *Convergence conditions for ascent methods*, SIAM Rev. 11 (1969), pp. 226–235. Available at http://dx.doi.org/10.1137/1011036. MR 0250453.

[27] P. Wolfe, *A method of conjugate subgradients for minimizing nondifferentiable functions*, Math. Programming Stud., 1975, pp. 145–173. MR 0448896.