NONSMOOTH OPTIMIZATION VIA BFGS

ADRIAN S. LEWIS* AND MICHAEL L. OVERTON†

Abstract. We investigate the BFGS algorithm with an inexact line search when applied to nonsmooth functions, not necessarily convex. We define a suitable line search and show that it generates a sequence of nested intervals containing points satisfying the Armijo and weak Wolfe conditions, assuming only absolute continuity. We also prove that the line search terminates for all semi-algebraic functions. The analysis of the convergence of BFGS using this line search seems very challenging; our theoretical results are limited to the univariate case. However, we systematically investigate the numerical behavior of BFGS with the inexact line search on various classes of examples. The method consistently converges to local minimizers on all but the most difficult class of examples, and even in that case, the method converges to points that are apparently Clarke stationary. Furthermore, the convergence rate is observed to be linear with respect to the number of function evaluations, with a rate of convergence that varies in an unexpectedly consistent way with the problem parameters. When the problem is sufficiently difficult, convergence may not be observed, but this seems to be due to rounding error caused by ill-conditioning. We try to give insight into *why* BFGS works as well as it does, and we conclude with a bold conjecture.

Key words. BFGS, quasi-Newton, nonsmooth, nonconvex, line search, R-linear convergence

AMS subject classifications. 90C30, 65K05

1. Introduction.

This paper is concerned with the behavior of the BFGS (Broyden-Fletcher-Goldfarb-Shanno) variable metric (quasi-Newton) method with an *inexact* line search applied to nonsmooth functions, convex or nonconvex. A companion paper [LO08] analyzes BFGS with an *exact* line search on some specific nonsmooth examples.

It was shown by Powell [Pow76] that, if $f : \mathbf{R}^n \to \mathbf{R}$ is convex and twice continuously differentiable, and the sublevel set $\{x : f(x) \leq f(x_0)\}$ is bounded (x_0 being the starting point), then the sequence of function values generated by the BFGS method with a weak Wolfe inexact line search converges to or terminates at the minimal value of f. This result does not follow directly from the standard Zoutendijk theorem as one needs to know that the eigenvalues of the inverse Hessian approximation H_k do not grow too large or too small. The result has never been extended to the nonconvex case. Indeed, very little is known in theory about the convergence of the standard BFGS algorithm when f is a nonconvex smooth function, although it is widely accepted that the method works well in practice [LF01].

There has been even less study of the behavior of BFGS on nonsmooth functions. While any locally Lipschitz nonsmooth function f can be viewed as a limit of increasingly ill-conditioned differentiable functions (see [RW98, Thm 9.67] for one theoretical approach, via "mollifiers"), such a view has no obvious consequence for the algorithm's asymptotic convergence behavior when f is not differentiable at its minimizer. Yet, when applied to a wide variety of nonsmooth, locally Lipschitz functions, not necessarily convex, BFGS is very effective, automatically using the gradient difference information to update an inverse Hessian approximation H_k that typically becomes extremely ill-conditioned. A key point is that, since locally Lipschitz functions are

*School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14853, U.S.A. aslewis@orie.cornell.edu. Research supported in part by National Science Foundation Grant DMS-0806057.

† Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, U.S.A. overton@cs.nyu.edu. Research supported in part by National Science Foundation Grant DMS-0714321.

differentiable almost everywhere, as long as the method is initialized randomly and an inexact line search is used, it is very unlikely that an iterate will be generated at which f is not differentiable. Under the assumption that such a point is never encountered, the method is well defined, and linear convergence of the function values to a locally optimal value is typical (not superlinear, as in the smooth case).

Although there has been little study of this phenomenon in the literature, the frequent success of variable metric methods on nonsmooth functions was observed by Lemaréchal more than 25 years ago. His comments in [Lem82] include:

> We have also exhibited the fact that it can be good practice to use a quasi-Newton method in nonsmooth optimization [as] convergence is rather rapid, and often a reasonably good approximation of the optimum is found; this, in our opinion, is essentially due to the fact that inaccurate line-searches are made. Of course, there is no theoretical possibility to prove convergence to the right point (in fact counterexamples exist), neither are there any means to assess the results.
>
> ...this raises the question: is there a well-defined frontier between quadratic and piecewise linear, or more generally, between smooth and nonsmooth functions?

Lemaréchal's observation was noted in several papers of Lukšan and Vlček [LV99, LV01, VL01]. They wrote in [VL01]: "standard variable metric methods are relatively robust and efficient even in the nonsmooth case.... On the other hand, no global convergence has been proved for standard variable metric methods applied to nonsmooth problems, and possible failures or inaccurate results can sometimes appear in practical computations." Motivated by the latter observation, combined with the attraction of the low overhead of variable metric methods, Lukšan and Vlček proposed new methods intended to combine the global convergence properties of bundle methods [HUL93, Kiw85] with the efficiency of variable metric methods. Other papers that combine ideas from bundle and variable metric methods include [BGLS95, LS94, MSQ98, RF00].

Our interest is in the standard BFGS method [NW06, Chap. 6] applied directly to nonsmooth functions without any modifications, except that care must be taken in the line search. Although we are motivated by the potential use of BFGS as a practical method, primarily in the nonconvex case, this paper is focused on understanding its behavior on relatively simple examples, many of them convex. Our hope is that even partial success in this direction will lead the way toward a more complete picture of how BFGS behaves in general.

In Section 2 we give a detailed treatment of the line search. Standard line searches for smooth optimization impose an Armijo condition on reduction of the function value and a Wolfe condition controlling the change in the directional derivative. For the latter condition, often a "strong" version is imposed, requiring a reduction in the absolute value of the directional derivative, as opposed to a "weak" version that requires only an algebraic increase. The weak version is all that is required to ensure positive definiteness of the updated inverse Hessian approximation; nonetheless, it is popular both in textbooks and software to require the strong condition, despite the substantial increase in implementation difficulty, perhaps because this is the traditional route to proving convergence results for nonlinear conjugate gradient methods on smooth functions. For nonsmooth optimization, it is clear that enforcing the strong Wolfe condition is not possible in general, and it is essential to base the line search on the simpler weak Wolfe condition. The line search we describe is close to earlier methods in the literature, but our analysis differs. We prove that the line search generates a sequence of nested intervals containing points that satisfy the Armijo and Wolfe conditions, assuming only that the function is absolutely continuous along the line. We also prove that the line search terminates under slightly stronger assumptions, in particular covering all semi-algebraic functions (not necessarily locally Lipschitz).

The success of BFGS and most other variable metric methods in the smooth case is in large part because inexact line searches quickly find an acceptable step: eventually the method always accepts the unit initial step. The behavior of BFGS with an inexact line search on *nonsmooth* functions is complex: it is far from clear whether the unit step will be well scaled. As an initial analysis of this crucial but difficult question, we carefully consider the univariate case, which is surprisingly nontrivial and offers substantial insight. In Section 3 we prove that, for $f(x) = |x|$, the method converges with R-linear rate 1/2 with respect to the number of function evaluations. In Section 4, we argue that the rate for the function $\max\{x, -ux\}$ is, for large u, approximately $2^{-1/\log_2 u}$.

In Section 5 we systematically investigate the numerical behavior of BFGS with the inexact line search on various classes of examples. We find that the method consistently converges to local minimizers on all but the most difficult class of examples, and even in that case, the method converges to points that are apparently Clarke stationary. Furthermore, the convergence rate is observed to be linear with respect to the number of function evaluations, with a rate of convergence that varies in an unexpectedly consistent way with the dimension and parameters defining the problem in each class. When the problem is constructed to be sufficiently difficult, convergence may not be observed, but this seems to be due to rounding error caused by ill-conditioning, not a failure of the method to converge in exact arithmetic.

In Section 6 we briefly consider alternative methods that are applicable to nonsmooth, nonconvex problems. We also mention our publicly available MATLAB code HANSO, addressing the issues of stopping criteria and how to assess the quality of the result.

We conclude in Section 7 with a bold conjecture.

An intuitive, although far from complete, argument for the success of BFGS on nonsmooth problems goes as follows. Because the gradient differences may be enormous compared to the difference of the points where they are computed, the inverse Hessian approximation typically becomes very ill-conditioned in the nonsmooth case. Tiny eigenvalues of H_k correspond to directions along which, according to the quadratic model constructed by BFGS, the function has a huge second derivative. In fact, of course, f is not differentiable at the local optimizer being approximated, but can be arbitrarily well approximated by a function with a sufficiently ill-conditioned Hessian. As is familiar from interior point methods for constrained optimization, it is this ill-conditioning of H_k that apparently enables the method to work so well. Remarkably, it often happens that the condition number of H_k approaches the inverse of the machine precision before rounding errors cause a breakdown in the method, usually failure to obtain a reduction of f in the inexact line search. The spectral decomposition of the final H_k typically reveals two subspaces along which the behavior of f is very different: the eigenvalues that are *not* relatively tiny are associated with eigenvectors that identify directions from the final iterate along which f varies smoothly, while the tiny eigenvalues are associated with eigenvectors along which f varies nonsmoothly. More specifically, when applied to partly smooth functions

[Lew03], it seems typical that BFGS *automatically* identifies the U and V-spaces associated with f at the approximate minimizer. Furthermore, even when H_k is very ill-conditioned, the unit step is typically relatively well scaled, and this property does not deteriorate as the iteration count k increases (although, as noted in Section 5.4, it does seem to deteriorate as the dimension n increases). Mysteries that remain include the mechanism that prevents the method from stagnating, the reason for the relative well-scaledness of the unit step in the line search, and the condition measure of f that determines the amazingly consistent linear rates of convergence that we observe.

A natural question is whether these observations extend to the well known limited memory variant of BFGS. Our experience with this is minimal and we do not address the issue here. However, we note that comments in the literature observing that limited memory BFGS sometimes works well in practice on nonsmooth problems have appeared occasionally: see [Lee98, ZZA$^+$00]. Negative comments have also appeared [Haa04, p.83],[YVGS08], leading the authors to propose modifications to the method. Although we do not consider limited memory variants in this paper, in our opinion a key change should be made to the widely used codes L-BFGS and L-BFGS-B [ZBN97] so that they are more generally applicable to nonsmooth problems: include the weak Wolfe line search defined in Section 2 as an optional alternative to the strong Wolfe line search that is currently implemented.

2. The Line Search. We consider here an inexact line search for nonsmooth optimization very close to one suggested by Lemaréchal [Lem81], and similar to analogous methods of Wolfe [Wol75] (for the convex case) and Mifflin [Mif77]. This line search imposes an Armijo condition on reduction of the function value and a weak Wolfe condition imposing an algebraic increase in the directional derivative along the line. Our algorithm differs from theirs in one small respect: it only accepts points at which the function is differentiable along the search direction. Avoiding BFGS iterates at which the function f is nondifferentiable has some merits: in theory we thereby rule out examples of failure due to null steps along the lines discussed in [LO08, Section 3], and in practice, as implied by Lemaréchal's quote in the previous section, the fact that an inexact line search generally avoids such points is crucial for the success of BFGS in practice. For structured functions and initial conditions, the line search might indeed encounter nondifferentiable points. Our experiments, on the other hand, use random starting points and Hessian approximations, avoiding nondifferentiable points almost surely. In any case, in the interests of broad practicality, our implementation does not check for differentiability.

Let $\bar{x}$ be an iterate of an optimization algorithm and $\bar{p}$ be a direction of search. Then the line search objective h is given by

$$h(t) = f(\bar{x} + t\bar{p}) - f(\bar{x}).$$

We seek a method for selecting a step under the following assumption.

Assumption 2.1. The function $h : \mathbf{R}_+ \to \mathbf{R}$ is absolutely continuous on every bounded interval, and bounded below. Furthermore, it satisfies

$$h(0) = 0 \quad \text{and} \quad s = \limsup_{t \downarrow 0} \frac{h(t)}{t} < 0.$$

Absolutely continuous functions may be characterized as indefinite integrals of integrable functions [Roy63]. They are differentiable almost everywhere, and satisfy the fundamental theorem of calculus. Lipschitz functions are absolutely continuous, as are semi-algebraic functions.[1] Hence if the function f is locally Lipschitz or semi-algebraic, the line search objective h satisfies the absolute continuity assumption.

Given constants $c_1 < c_2$ in the interval $(0,1)$, we seek a *weak Wolfe step*, which we define to be a number $t > 0$ satisfying the Armijo and Wolfe conditions

$$S(t): \qquad h(t) < c_1 st \tag{2.2}$$
$$C(t): \qquad h \text{ is differentiable at } t \text{ with } h'(t) > c_2 s. \tag{2.3}$$

LEMMA 2.4. *If the condition S holds at the number $\alpha > 0$ but fails at the number $\beta > \alpha$, then the set of weak Wolfe steps in the interval $[\alpha, \beta]$ has nonzero measure.*

Proof Define a number

$$t^* \;=\; \inf\{t \in [\alpha,\beta] : h' \leq c_2 s \text{ a.e. on } [\alpha, t]\}.$$

Then $h' \leq c_2 s$ a.e. on the interval $[\alpha, t^*]$, so

$$h(t^*) - h(\alpha) = \int_{\alpha}^{t^*} h' \leq c_2 s(t^* - \alpha) \leq c_1 s(t^* - \alpha).$$

Since the condition $S(\alpha)$ holds,

$$h(t^*) - c_1 st^* \leq h(\alpha) - c_1 s\alpha < 0,$$

so the condition $S(t^*)$ holds. Since the condition $S(\beta)$ fails, $t^* \neq \beta$, so in fact $t^* < \beta$. By the definition of t^*, for all small $\delta > 0$, condition C must hold on a subset of the interval $[t^*, t^* + \delta]$ of positive measure. But by continuity, the condition S holds throughout this interval for small δ. $\square$

THEOREM 2.5 (existence of step). *Under Assumption 2.1, the set of weak Wolfe steps has nonzero measure.*

Proof The "lim sup" assumption ensures that there exists $\alpha > 0$ satisfying

$$\frac{h(\alpha)}{\alpha} < c_1 s$$

so condition $S(\alpha)$ holds. On the other hand, condition $S(\beta)$ must fail for all large $\beta > 0$ because the function h is bounded below. Now apply the lemma. $\square$

In fact, for the purposes of the above result, the "lim sup" in Assumption 2.1 could be replaced by "lim inf".

[1] The graph of a semi-algebraic function is a finite union of sets, each defined by a finite list of polynomial inequalities.

2.1. Definition of the inexact line search.

We next describe the method.

ALGORITHM 2.6 (line search).

```
$\alpha \leftarrow 0$
$\beta \leftarrow +\infty$
$t \leftarrow 1$
**repeat**
        **if** $S(t)$ fails, $\beta \leftarrow t$
        **else if** $C(t)$ fails, $\alpha \leftarrow t$
        **else STOP**
        **if** $\beta < +\infty$, $t \leftarrow (\alpha + \beta)/2$
        **else** $t \leftarrow 2\alpha$
**end(repeat)**
```

Each execution of the repeat loop involves trying one new choice of the step t. We call such an execution a *trial*.

THEOREM 2.7 (convergence). *Whenever the above line search iteration terminates, the final trial step t is a weak Wolfe step. In particular, it terminates under the assumption*

$$\lim_{t \uparrow \bar{t}} h'(t) \text{ exists in } [-\infty, +\infty] \text{ for all } \bar{t} > 0. \tag{2.8}$$

If, on the other hand, the iteration does not terminate, then it eventually generates a nested sequence of finite intervals $[\alpha, \beta]$, halving in length at each iteration, and each containing a set of nonzero measure of weak Wolfe steps. These intervals converge to a step $t_0 > 0$ such that

$$h(t_0) = c_1 s t_0 \quad \text{and} \quad \limsup_{t \uparrow t_0} h'(t) \geq c_2 s. \tag{2.9}$$

Proof It is clear that if the line search terminates at t, both conditions $S(t)$ and $C(t)$ hold. Suppose the iteration does not terminate. Eventually, the upper bound β becomes finite, since otherwise condition $S(2^k)$ must hold for all $k = 1, 2, \ldots$, contradicting the boundedness assumption. Furthermore, from the update for β, once β is finite, condition $S(\beta)$ always fails.

Next, notice that eventually the lower bound $\alpha > 0$. Otherwise, α is always zero, and after the upper bound β becomes finite the trial step t keeps halving and the condition $S(t)$ keeps failing, contradicting the "lim sup" condition in Assumption 2.1. Notice also that after any update to the lower bound α, the condition $S(\alpha)$ must hold.

Let us denote by $[\alpha_k, \beta_k]$ the sequence of intervals generated by the iteration. Once $\alpha_k > 0$ and $\beta_k < \infty$, the intervals are positive, finite, and halve in length at each iteration, and the sequences (α_k) and (β_k) are monotonic increasing and decreasing respectively. Hence there must exist a point $t_0 > 0$ such that $\alpha_k \uparrow t_0$ and $\beta_k \downarrow t_0$. Furthermore, we know that the condition $S(\alpha_k)$ holds and the condition $S(\beta_k)$ fails.

We deduce several consequences. First, by the continuity of the function h at the point t_0, we must have $h(t_0) = c_1 s t_0$, so the condition $S(t_0)$ fails. On the other hand, the condition $S(\alpha_k)$ holds, so $\alpha_k < t_0$ for all k. Now, Lemma 2.4 shows the existence of a weak Wolfe step $t_k \in [\alpha_k, t_0]$. In particular, we know $h'(t_k) > c_2 s$, so property (2.9) follows.

Now suppose assumption (2.8) holds, and yet, by way of contradiction, that the iteration does not terminate but instead generates an infinite sequence of intervals $[\alpha_k, \beta_k]$ as above, shrinking to a point $t_0 > 0$. Every α_k is a trial step at some iteration $j \le k$, so the condition $C(\alpha_k)$ fails. By our assumption, the function h is differentiable on some nonempty open interval (t', t_0), and hence in particular at α_k for all large k, and so must satisfy $h'(\alpha_k) \le c_2 s$. We deduce

$$\lim_{t \uparrow t_0} h'(t) \le c_2 s < c_1 s. \tag{2.10}$$

On the other hand, h is continuous, so by the Mean Value Theorem there exists a point γ_k in the interval (α_k, t_0) satisfying

$$h'(\gamma_k) = \frac{h(t_0) - h(\alpha_k)}{t_0 - \alpha_k} \ge \frac{c_1 s t_0 - c_1 s \alpha_k}{t_0 - \alpha_k} = c_1 s.$$

Since γ_k converges to t_0 from the left, this contradicts inequality (2.10). □

The above convergence result is not restricted to Lipschitz functions. In particular, assumption (2.8) holds for any semi-algebraic function h. By contrast with our result, [Lem81] considers locally Lipschitz functions and assumes a "semismoothness" property. As we now sketch, a very similar argument to the proof above covers that case too.

Suppose the function h *weakly lower semismooth* at every point $\bar{t} > 0$: in other words, it is locally Lipschitz around $\bar{t}$ and satisfies

$$\liminf_{t \downarrow 0} \frac{h(\bar{t} + sd) - h(\bar{t})}{s} \;\ge\; \limsup_{k} g_k d$$

for $d = \pm 1$ and any sequence of Clarke subgradients g_k of h at $\bar{t} + s_k d$ where $s_k \downarrow 0$. In the language of [**?**], this is equivalent to the function $-h$ being "weakly upper semismooth". Suppose in addition that h is differentiable at every trial step. We then claim that the line search terminates.

To see this, assume as in the proof that the iteration does not terminate, so we obtain a sequence of points $\alpha_k > 0$ increasing to a point $t_0 > 0$ such that $h(t_0) = c_1 s t_0$, the condition $S(\alpha_k)$ holds, the condition $C(\alpha_k)$ fails, and h is differentiable at α_k, for each $k = 1, 2, 3, \ldots$. We deduce the inequalities

$$\begin{aligned}
\liminf_{s \downarrow 0} \frac{h(t_0 - s) - h(t_0)}{s} &\le \liminf_{k} \frac{h(\alpha_k) - h(t_0)}{t_0 - \alpha_k} \\
&\le \liminf_{k} \frac{c_1 s \alpha_k - c_1 s t_0}{t_0 - \alpha_k} \\
&= -c_1 s \\
&< -c_2 s \\
&\le \limsup_{k} h'(\alpha_k)(-1),
\end{aligned}$$

which contradicts the definition of weak lower semismoothness.

2.2. The line search on a convex function.

We now consider the complexity of the line search applied to a convex function h.

Unlike the method of [Lem81], due to our different approach to nondifferentiable points, our line search method may fail to terminate on some pathological functions, even assuming convexity. For example, consider the function $h \colon \mathbf{R}_+ \to \mathbf{R}$ defined by $h(t) = t^2 - t$ for any number t of the form

$$t_k = \sum_{j=0}^{k} (-2)^{-j},$$

and for t equal to 0 or $\frac{2}{3}$ or larger than 1. On the closed intervals between neighboring points of this form, define h by linear interpolation. Then h is convex (although not semi-algebraic), and has a piecewise linear graph with corners $(t_k, h(t_k))$ accumulating at the point $(\frac{2}{3}, -\frac{2}{9})$. If $c_1 = \frac{2}{3}$, then the points satisfying the Armijo condition $S(\cdot)$ constitute the interval $(0, \frac{2}{3})$. For any $c_2 \in (c_1, 1)$, the sequence of trial points is then the sequence of partial sums $t_0, t_1, t_2, \ldots$ above. The condition $S(t_k)$ fails for even integers k and holds for odd k, and condition $C(t_k)$ always fails due to nondifferentiability. Hence the line search does not terminate.

However, in the convex case we can bound the number of function trials that are needed to generate a point inside an interval in which almost every point satisfies the Armijo and Wolfe conditions.

PROPOSITION 2.11 (complexity of line search). *Consider a convex function h satisfying Assumption 2.1. Then the set of weak Wolfe steps is an open interval $I \subset \mathbf{R}_+$, with any points where h is nondifferentiable removed. Suppose I has left-hand endpoint $b > 0$ and length $a \in (0, +\infty]$. Define*

$$d \;=\; \max\{1 + \lfloor \log_2 b \rfloor, 0\}.$$

Then after a number of trials between

$$d+1 \quad \textit{and} \quad d + 1 + \max\Big\{d + \Big\lfloor \log_2 \frac{1}{a} \Big\rfloor, 0\Big\}$$

(interpreted in the natural way when $a = +\infty$), the line search tries a step in I.

Proof By convexity, it is easy to see that the interval of interest is given by

$$b = \inf\{t > 0 : C(t) \text{ holds}\}$$
$$b + a = \sup\{t > 0 : S(t) \text{ holds}\}.$$

The line search behaves rather simply. It doubles the initial step until the trial satisfies $t > b$. Assuming this step does not lie in the interval I, the condition $S(t)$ must fail, so the interval $[\alpha, \beta]$ used by the line search to bracket a weak Wolfe step is $[0, t]$. After this "doubling" phase, the method moves to a "bisection" phase, repeatedly trying a point t equal to the midpoint of the current bracketing interval. As long as this point lies outside I, the trial t replaces either the left or right endpoint of the bracket, depending on whether $t \le b$ or $t \ge b + a$.

It is easy to see that the number of doubling steps is d, so the number of trials needed in this phase is $d + 1$. After this phase, the bracketing interval has length 2^d. In fact, if the method continues, the interval I must be contained within the bracket $[2^{d-1}, 2^d]$, and has length a. To find a point in I, the bisection phase repeatedly halves the length of the current bracket. Notice 2^{d-1} is a previous trial point. Hence we need at most

$$h \;=\; \max\Big\{d + \Big\lfloor \log_2 \frac{1}{a} \Big\rfloor, 0\Big\}$$

further trials before trying a point in I. The result follows. □

In the above result, consider the special case where b is large but $a = 1$, so the interval I is $(b, b+1)$. Then the line search will perform a large number,

$$d = 1 + \lfloor \log_2 b \rfloor,$$

doubling steps, and then performs between zero and d additional bisection steps. The point b lies in the interval $[2^{d-1}, 2^d]$. If b lies in the open unit interval $2^d - (0,1)$, no further trials will be needed. If, on the other hand, b lies in the interval $2^{d-2} + 2^{d-1} - (0,1)$, one further trial will be needed. Similarly, there exist two open unit intervals of possible values of b requiring two further trials, four requiring three, and more generally, 2^{m-1} unit intervals requiring m further trials, for $m = 1, 2, \ldots, d-1$. If the point b was a random variable, uniformly distributed in the interval $[2^{d-1}, 2^d]$, the expected number of trials until we try a point in I is then

$$\begin{aligned}
&2^{1-d} \cdot 0 + 2^{1-d} \cdot 1 + 2^{2-d} \cdot 2 + 2^{3-d} \cdot 3 + \cdots + 2^{-1} \cdot (d-1) \\
&= 2^{1-d}(1 + 2 \cdot 2 + 2^2 \cdot 3 + \cdots + 2^{d-2} \cdot (d-1)) \\
&= d - 2 + 2^{1-d}.
\end{aligned}$$

Thus the expected number of trials in the bisection phase is roughly $\log_2 b$, so the expected total number of trials is about $2 \log_2 b$.

3. Analysis for the Absolute Value.

We now consider the behavior of the line search given in Algorithm 2.6 when applied to the problem of minimizing the absolute value function. Given a current nonzero iterate $x_k \in \mathbf{R}$ (where $k = 0, 1, 2, \ldots$ is the iteration counter), and a nonzero initial step $\gamma \in \mathbf{R}$, we therefore apply the line search to the function

$$h(t) = |x_k + t\gamma|.$$

Assumption 2.1 becomes

$$\gamma x_k < 0 \quad \text{and} \quad s = -|\gamma|.$$

Since

$$h'(t) = \begin{cases} -|\gamma| & (t < |x_k|/|\gamma|) \\ |\gamma| & (t > |x_k|/|\gamma|), \end{cases}$$

the value of the Wolfe parameter $c_2 \in (0,1)$ is irrelevant to the algorithm. It simplifies our analysis, and causes no difficulty for the absolute value function, if we set the Armijo parameter $c_1 = 0$. The line search conditions then become

$$\begin{aligned}
S(t) :& |x_k + t\gamma| < |x_k| \\
C(t) :& x_k(x_k + t\gamma) < 0.
\end{aligned}$$

We make one last slight modification to the line search algorithm: we terminate it if we encounter the point zero (or in other words if $t = -x_k/\gamma$). For the remainder of this section, when we refer to the *inexact line search* as applied to the absolute function $x \mapsto |x|$ we mean this version: $0 = c_1 < c_2 < 1$ and terminating if we encounter $x = 0$.

Using this line search method, we now consider the behavior of the BFGS method for minimizing the absolute value. Clearly properties S and C guarantee

$$|x_{k+1}| < |x_k| \quad \text{and} \quad x_k x_{k+1} < 0,$$

providing $x_k \neq 0 \neq x_{k+1}$. The BFGS formula for the inverse Hessian approximation reduces to the secant equation

$$H_{k+1} = \frac{2}{|x_{k+1} - x_k|}$$

and hence the initial step for the next line search is

$$\gamma = -\frac{|x_{k+1} - x_k|}{2} \operatorname{sgn}(x_{k+1}).$$

To summarize, the iterates alternate signs, and the initial step in the line search has size half the distance to the previous iterate.

The following very easy tool is basic in our analysis.

LEMMA 3.1. *Consider any integer a, and any number x in the interval $(2^{-a-1}, 2^{-a})$. Then the unique integer r satisfying both the properties*

(i) $|x - 2^{-r}| < x$

(ii) $x < 2^{-r}$

is $r = a$. More precisely, if $r < a$, then inequality (i) *fails (in fact strictly), and if $r > a$, then inequality* (i) *holds but inequality* (ii) *fails (in fact strictly).*

Using this tool, the first application of the line search is easy to analyze.

PROPOSITION 3.2 (first iteration). *Consider the problem of minimizing the absolute value function. Suppose that the initial point x_0 lies in the interval $(2^{-a-1}, 2^{-a}]$ for some integer a, and that the initial step in the inexact line search is -1. Then the line search terminates after $1 + |a|$ trials with the next iterate $x_1 = x_0 - 2^{-a}$.*

Proof We distinguish two cases. If $a \geq 0$, then the line search tries in turn the points

$$x_0 - 2^{-r} \quad \text{for} \quad r = 0, 1, 2, \ldots.$$

Using Lemma 3.1, as long as $r < a$, the descent condition S fails, so the trial step halves. Eventually $r = a$, and then Lemma 3.1 implies that both conditions S and C hold, so the line search terminates.

On the other hand, if $a < 0$, then the line search tries in turn the points

$$x_0 - 2^{-r} \quad \text{for} \quad r = 0, -1, -2, \ldots.$$

Using Lemma 3.1, as long as $r > a$, the Wolfe condition $C(t)$ fails, so the trial step doubles. Eventually $r = a$, and again the line search terminates. □

An inductive argument now shows how the line search behaves on the subsequent iterations.

PROPOSITION 3.3 (subsequent iterations). *Consider the problem of minimizing the absolute value function using BFGS with the inexact line search. Suppose two successive iterates x_k and x_{k+1} are both nonzero. Then the next iterate is*

$$x_{k+2} = x_{k+1} + \frac{x_k - x_{k+1}}{2^r},$$

where the number $r > 0$ is the minimal strictly positive integer for which the right-hand side has absolute value strictly less than $|x_{k+1}|$. The number of trials required by the line search is exactly r.

Proof We can without loss of generality suppose $x_k < 0$. Then, by our earlier observations, we have $0 < x_{k+1} < -x_k$, and the initial trial step in the line search takes us from the point x_{k+1} to the point

$$\frac{x_k + x_{k+1}}{2}.$$

We now claim that the line search tries in turn the points

$$x_{k+1} - \frac{x_{k+1} - x_k}{2^r} \quad \text{for} \quad r = 1, 2, 3, \ldots,$$

until the descent condition S is satisfied, or equivalently, until the displayed quantity has absolute value strictly less than $|x_{k+1}|$: our result then follows.

The claim follows by induction. The choice $r = 1$ gives the first trial point, as we have already argued. As long as the descent condition S fails, the line search halves the trial step, thus incrementing the integer r by one. But as soon as we reach an integer r such that the descent condition holds, then so does the Wolfe condition C, by Lemma 3.1, so the line search terminates. $\square$

Remarkably, minimizing the absolute value function in the above fashion, starting from the initial point x_0, mimics exactly an algorithm for computing the alternating binary expansion of x_0 that is described in Appendix A.

THEOREM 3.4 (BFGS on the absolute value). *Given any positive number x_0, consider its alternating binary expansion*

$$x_0 = \sum_{j=0}^{m} (-1)^j 2^{-a_j},$$

where m is either a nonnegative integer or ∞ and $a_0 < a_1 < a_2 < \cdots$. Then applying BFGS to minimize the absolute value function, using the inexact line search, with initial point x_0 and initial Hessian approximation 1, generates the iterates

$$x_k = \sum_{j=k}^{m} (-1)^j 2^{-a_j} \quad \textit{for all integers } k \le m. \tag{3.5}$$

Calculating the iterate x_1 takes $1 + |a_0|$ trials in the line search. For all $k < m$, given the iterate x_k, calculating the subsequent iterate x_{k+1} takes $a_k - a_{k-1}$ trials. If the alternating binary expansion is finite (that is, $m < \infty$), then BFGS terminates at zero after finitely many line search trials. Otherwise, with respect to the number of trials, the iterates x_k converge to zero with R-linear rate $\frac{1}{2}$.

Proof We prove the most interesting case: $m = \infty$. The argument when x has a finite alternating binary expansion is very similar.

We prove equation (3.5) by induction on k. The equation is trivially true when $k = 0$. By Lemma A.1, we have $2^{-a_0-1} < x < 2^{-a_0}$. The case $k = 1$ then follows by Proposition 3.2, and the number of trials the line search needs to compute x_1 is $1 + |a_0|$, as claimed.

Now suppose equation (3.5) holds for some given k, and also when k is replaced by $k + 1$. By Lemma A.1, we know

$$2^{-a_{k+1}-1} < (-1)^{k+1} x_{k+1} < 2^{-a_{k+1}}.$$

Furthermore, by Proposition 3.3, the next iterate is

$$x_{k+2} = x_{k+1} + \frac{x_k - x_{k+1}}{2^r} = x_{k+1} + (-1)^k 2^{-a_k - r}$$

where the number $r > 0$ is the minimal strictly positive integer for which the right-hand side has absolute value strictly less than $|x_{k+1}|$. The number of trials required by the line search is exactly r. Now Lemma 3.1 shows $r = a_{k+1} - a_k$, and equation (3.5) with k replaced by $k+2$ follows. This completes the induction.

To calculate the R-linear convergence rate, notice that if the total number of trials in all the line searches so far is i, where

$$|a_0| + a_{k-1} - a_0 \ < \ i \ \leq \ |a_0| + a_k - a_0$$

for some index $k = 1, 2, 3, \ldots$, then the current error is

$$|x_k| \ \in \ (2^{-a_k - 1}, 2^{-a_k}),$$

which is bounded above by $2^{|a_0| - a_0 - i}$. Hence, with respect to trials in the line search, the convergence to zero in R-linear with rate at most $\frac{1}{2}$. Furthermore, since for all k the error after $|a_0| + a_k - a_0$ trials is at least $2^{-a_k - 1}$, the rate $\frac{1}{2}$ is exact. $\square$

As an example, consider the initial point

$$x_0 = \frac{4}{7} = 1 - \frac{1}{2} + \frac{1}{8} - \frac{1}{16} + \ldots = \sum_{r=0}^{\infty} (2^{-3r} - 2^{-3r-1})$$

After one trial in the line search, we arrive at the point $x_1 = -3/7$. One more trial takes us to the point $x_2 = 1/14$. The next line search takes two trials before terminating at the point $x_3 = -3/56$. This pattern now repeats: the line search between

$$x_{2j} = \frac{4}{7 \cdot 8^j} \quad \text{and} \quad x_{2j+1} = -\frac{3}{7 \cdot 8^j} \quad \text{for } j = 1, 2, 3, \ldots$$

takes just one trial, but from x_{2j+1} to x_{2j+2} takes two trials. It is easy to confirm that this is exactly the behavior predicted by Theorem 3.4.

Theorem 3.4 shows that, for any initial point $x_0 \in (1/2, 1)$, after a_k trials BFGS guarantees an error less than 2^{-a_k}. Thus, the error is reduced to $\epsilon > 0$ after about $\log_2(1/\epsilon)$ trials. By contrast, it is easy to check that steepest descent on $f(x) = |x|$, starting with $x_0 = 2/3$, needs $k(k+1)/2$ trials to reduce the error to $2^{1-k}/3$: consequently, reducing the error to ϵ requires about $(\log_2(1/\epsilon))^2/2$ trials.

4. A Conjecture for the Tilted Absolute Value.

In the previous section we saw that BFGS, when applied to the absolute value function, either terminates or converges to zero with R-linear rate 1/2. We next consider the more general "tilted" case, where we add a linear function to the absolute value. Numerical experiments suggest the following behavior.

Conjecture 4.1. *Given any $u > 0$, consider the BFGS iteration for minimizing the function*

$$x \mapsto \max\{x, -ux\} \quad (x \in \mathbf{R})$$

using the inexact line search, with any initial point and any strictly positive initial Hessian approximation. The method either terminates at zero, or generates a sequence of interates converging to zero, with R-linear rate $r(u)$ (with respect to the number of trials) depending only on the parameter u. Furthermore, this rate satisfies

$$\log_2 r(u) \sim -\frac{1}{\log_2 u} \quad \textit{as } u \to +\infty.$$

We discuss this conjecture in some detail. First notice that, as in the case of the absolute value function, the initial Hessian approximation is relevant only in the first iteration. Indeed, the iterates x_k alternate in sign, because of the Wolfe condition, and to calculate the next iterate x_{k+1} using the line search, we try adding to x_k the initial trial step

$$\begin{cases} \dfrac{x_{k-1} - x_k}{u+1} & \text{if } x_k > 0 \\[2ex] \dfrac{u(x_{k-1} - x_k)}{u+1} & \text{if } x_k < 0. \end{cases}$$

Our conjecture of an R-linear convergence rate independent of the initial point is motivated by the case of the absolute value.

We motivate the conjectured asymptotic behavior of the rate for large values of the parameter u very loosely as follows. Consider a "typical" nonterminating instance of the method. Given an iterate $x_k > 0$ for some large iteration count k, the next iterate x_{k+1} must lie in the interval $(-x_k/u, 0)$. Since u is large, this interval is small relative to its distance from the iterate x_k, so we would expect the line search to require multiple trials before finding x_{k+1}, thereby "randomizing" its position. (Indeed, an argument similar to the discussion after Proposition 2.11 suggests that, independent of the size of the initial trial step, we typically need at least $\log_2 u$ bisection steps in the line search.) It therefore seems reasonable to approximate the next iterate x_{k+1} by a random variable, uniformly distributed on $(-x_k/u, 0)$, an assumption supported by numerical experiments.

As we have observed, the initial trial step at x_{k+1} is

$$\frac{u(x_k - x_{k+1})}{u+1}.$$

Throughout what follows, we loosely approximate expressions involving the large parameter u by the leading term in a power series expansion. In particular, therefore, we approximate the above step simply by x_k. The line search repeatedly bisects this trial step until it finds an acceptable iterate x_{k+2}. Thus we bisect the step m times, where $m = 0, 1, 2, \ldots$ is minimal such that

$$x_{k+2} \approx x_{k+1} + 2^{-m} x_k < -u x_{k+1},$$

or in other words

$$\frac{1}{2^m} < \frac{-x_{k+1}}{x_k/(u+1)} \approx \frac{-x_{k+1}}{x_k/u}.$$

Thus the line search terminates after m bisections (and hence $m+1$ trials), for $m = 1, 2, 3, \ldots$, exactly when the iterate x_{k+1} lies in an interval approximating

$$\frac{x_k}{u}\Big(\frac{-1}{2^{m-1}}, \frac{-1}{2^m}\Big).$$

This event, which we denote E_m, takes place with probability approximately 2^{-m}, due to the uniform distribution of x_{k+1}, and then

$$x_{k+2} \approx \frac{x_k}{2^m}.$$

Again, this approximation is supported by numerical evidence.

Let us now restrict attention to event E_m, for some fixed integer $m = 1, 2, 3, \ldots$. Since

$$x_{k+2} - x_{k+1} \approx \frac{x_k}{2^m},$$

the initial trial step at x_{k+2} is approximately

$$-\frac{x_k}{2^m(1+u)} \approx -\frac{x_{k+2}}{u}.$$

The behavior of the algorithm is scale-invariant, in the following sense. Consider two situations. In the first, the current iterate is $\bar{x}$, and the initial trail step at that iteration is z. In the second, the current iterate is $\gamma\bar{x}$, for some constant $\gamma > 0$, and the initial trail step is γz. Then it is easy to see that the behavior of the algorithm in the second situation is identical to that in the first, except that every trial is scaled by γ.

With this observation, we see that the algorithm, starting at the iterate x_{k+2} and with initial trial step $-x_{k+2}/u$, proceeds exactly as it would starting at the point u (resulting in the acceptable interval $(-1, 0)$), and with initial trial step approximately -1, except that all trials are scaled by the factor x_{k+2}/u. The number of trials necessary will therefore equal the number of trials needed by our standard line search when seeking a point in the interval $(u, u+1)$. The discussion after Proposition 2.11 suggests that the line search therefore makes approximately $2\log_2 u$ trials in this iteration. Notice that this number is independent of m.

To summarize, in the course of a typical pair of iterations of the line search, event E_m occurs with probability 2^{-m} (for $m = 1, 2, 3, \ldots$), and then approximately $m + 2\log_2 u$ trials results in a decrease in the value of the function by an approximate factor 2^{-m}. If the rate of linear convergence per trial, over this pair of iterations, is the random variable $R \in [0, 1)$, then

$$R^{m+2\log_2 u} \approx \frac{1}{2^m}$$

if E_m occurs. Hence the expectation of $\log_2 R$ is given approximately by

$$\mathbf{E}(\log_2 R) \approx \sum_{m=1}^{\infty} \frac{1}{2^m} \frac{-m}{m + 2\log_2 u}.$$

We expect this pattern to repeat in the long term, giving

$$\log_2 r(u) = \mathbf{E}(\log_2 R),$$

Hence, as $u \to \infty$, we deduce

$$(\log_2 u)(\log_2 r(u)) \;\approx\; -\sum_{m=1}^{\infty} \frac{1}{2^m} \frac{m \log_2 u}{m + 2\log_2 u} \;\to\; -\sum_{m=1}^{\infty} \frac{1}{2^m} \frac{m}{2} \;=\; -1,$$

using the fact that each term in the infinite sum is monotonic increasing in u. Hence we estimate

$$\log_2 r(u) \approx -\frac{1}{\log_2 u}$$

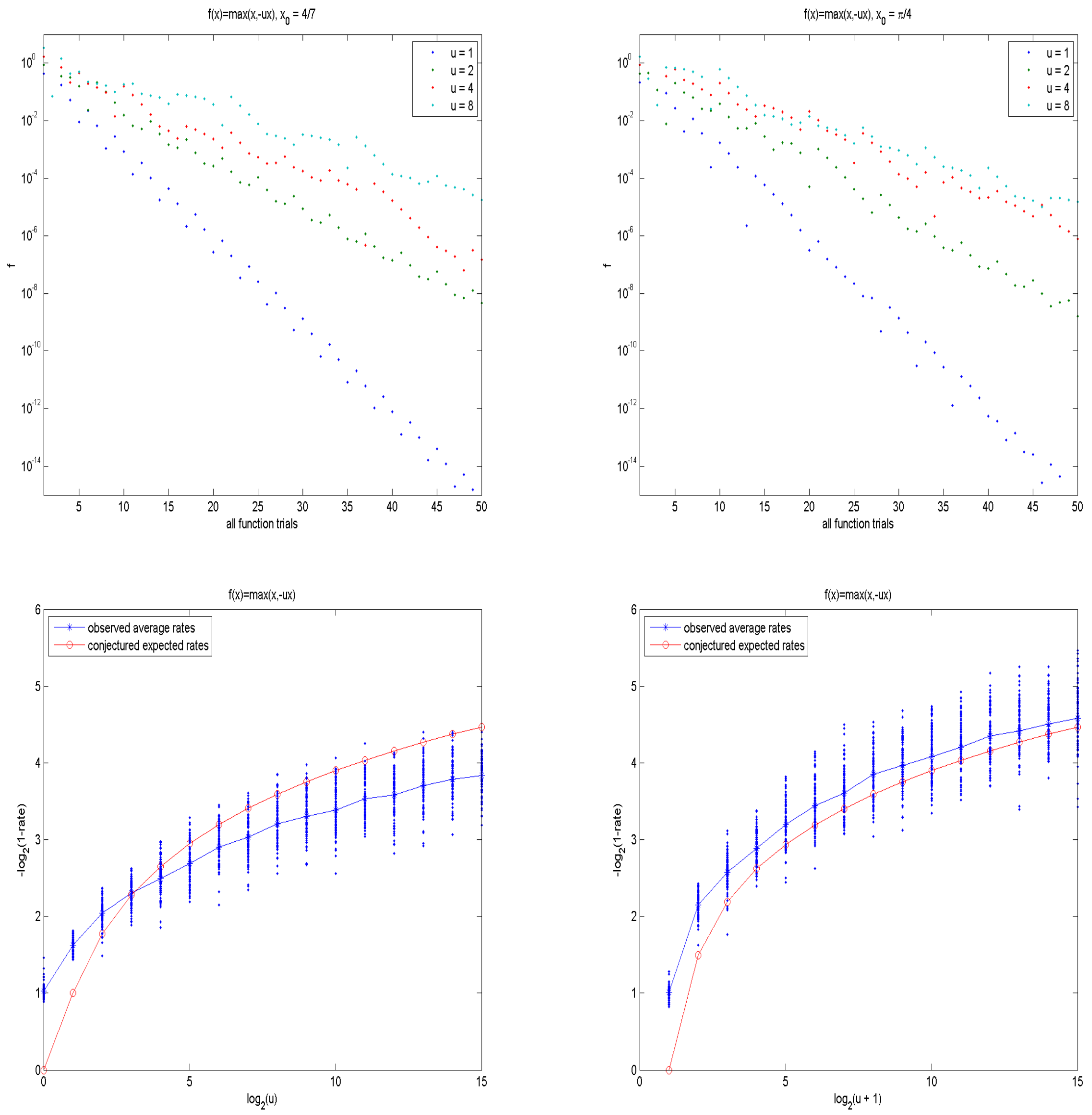

FIG. 4.1. *BFGS with the inexact line search on the tilted absolute value function* $f(x) = \max\{x, -ux\}$. *Top left: function trials when* $x_0 = 4/7$, *for* $u = 1, 2, 4$ *and* 8. *Top right: same when* $x_0 = \pi/4$. *Bottom left: plots* $-\log_2(1 - r)$, *where* r *is the average observed convergence rate with respect to the number of function trials, against* $\log_2(u)$, *where* u *takes values equal to powers of 2. Bottom right: same as a function of* $\log_2(u + 1)$, *where* $u + 1$ *takes values equal to powers of 2.*

for large u.

Figure 4.1 shows numerical results obtained by applying BFGS with the inexact line search to $f(x) = \max\{x, -ux\}$. As in the theoretical analysis, we set the Armijo parameter $c_1 = 0$; the Wolfe parameter c_2 is not relevant for this function. The top left figure shows the function trials using $x_0 = 4/7$ (to machine precision) for $u = 1, 2, 4$ and 8. In the case $u = 1$ we see the periodic behavior described at the end of Section 3. In the top right, we see no such periodic behavior with the choice $x_0 = \pi/4$, but the overall convergence rate remains about the same. The bottom left plot shows $-\log_2(1-r)$ as a function of $u = 2^j$, $j = 0, \ldots, 15$, where r is an observed convergence rate with respect to all function trials. Each dot in the plot represents a convergence rate computed by a least squares fit for a different random starting point. The least squares fits were made to the pairs (ν_k, f_k), where f_k is the function value $f(x_k)$ at the end of the kth line search and ν_k is the cumulative number of function trials up to that point. Thus, each least squares fit takes account of the *number* of function trials in the line searches but not the function trial values. Each least squares fit uses 40% of the iterates, excluding the first half to avoid the transient initial behavior, and excluding the final 10% to avoid contamination from rounding errors. The asterisks show the mean of 100 such observations for each value of u. The circles plot $-\log_2(1-\hat{r})$ where $\hat{r}$ is the conjectured approximate convergence rate $2^{-1/\log(u)}$. The data roughly support the conjecture.

A startling observation results from comparing the plot on the lower left to the one on the lower right, which is generated in the same way except that the experiments were made for $u = 2^j - 1$, $j = 0, \ldots, 15$. The convergence rates are noticeably different! We spent some time searching for an explanation until we realized that this discrepancy reflects the special roles of powers of two in the line search. We might think of the results in the lower left as being better than we could expect because the values of u happen to be powers of two.

5. Experimental Results.

We have found that the BFGS algorithm converges consistently at a linear rate on many different kinds of examples. The ones that we present here are chosen to gradually increase in complexity in order to illustrate a number of interesting points. We first make some comments on practical aspects of the algorithm and then introduce some theoretical concepts that we will need.

5.1. Practical Aspects of the BFGS Algorithm.

All the examples that we present below use the line search presented in Algorithm 2.6, with the Armijo and Wolfe conditions (2.2) and (2.3) modified as follows. For the Armijo condition, we set $c_1 = 0$ (any reduction in f is acceptable). For theoretical reasons at least in the smooth case, one normally uses a positive value for the Armijo parameter, but we have never encountered difficulties due to setting $c_1 = 0$. For the Wolfe condition, we make no attempt to check the “differentiable” property. In the unlikely event that f is evaluated at a point where it is not differentiable, any “tie-breaking” rule is considered acceptable for the computed gradient; this generally produces a subgradient. In contrast with the situation for the Armijo parameter, it is important from a practical point of view to set the Wolfe parameter c_2 to a value in $(0, 1)$. Setting $c_2 = 1$ invites disaster in the BFGS update, since division by zero can occur. Setting $c_2 = 0$ is not as bad, but this choice can destroy superlinear convergence when f is smooth as steps of size 1 may never be allowed, and it can cause difficulties in the nonsmooth case too. We therefore set $c_2 = 1/2$ for all our experiments.

For the tests on functions for which the optimal value f_{opt} is known (in most cases 0 but in some cases 1) we terminate BFGS when f is reduced below $f_{\text{opt}} + 10^{-15}$. In all cases, we terminate the method when it breaks down: by this we mean that either $g^T H g \leq 0$, where g is the final gradient produced by a successful line search and H is the updated Hessian, or the line search fails to satisfy the Armijo and Wolfe conditions: this is deemed to occur if a (large) limit on the number of doubling or bisection steps is exceeded. Normally, breakdown occurs because of rounding errors, although in principle it could also occur if a point where f is not differentiable is reached. In all tests, x and H were initialized randomly except if explicitly noted otherwise. For this section only, we reserve subscripts for *components* of the vector $x \in \mathbf{R}^n$. The plots that follow require viewing in color to be fully appreciated. All experiments were conducted in MATLAB, which uses IEEE double precision (about 16 decimal digits).

5.2. Theoretical Nomenclature. Although we will not present any theoretical results in this section, we will need to refer to several standard concepts in nonsmooth analysis. We use $\partial f(x)$ to denote the Clarke subdifferential (generalized gradient) [Cla83, RW98] of f at x, which for locally Lipschitz f is simply the convex hull of the limits of gradients of f evaluated at sequences converging to x. A key property, generalizing both convexity and continuous differentiability, is *regularity* [Cla83, RW98]: a locally Lipschitz, directionally differentiable function f is *Clarke regular* at a point when its directional derivative $x \mapsto f'(x; d)$ is upper semicontinuous there for every fixed direction d. A consequence of regularity of a function f at a point x is that the Clarke stationarity condition $0 \in \partial f(x)$ is equivalent to the first-order optimality condition $f'(x, d) \geq 0$ for all directions d. Another key property is *partial smoothness* [Lew03]. A regular function f is partly smooth at x relative to a manifold $\mathcal{M}$ containing x if (1) its restriction to $\mathcal{M}$ is twice continuously differentiable near x, (2) its subdifferential ∂f is continuous on $\mathcal{M}$ near x, and (3) $\operatorname{par} \partial f(x)$, the subspace parallel to the affine hull of the subdifferential of f at x, is exactly the subspace normal to $\mathcal{M}$ at x. For convenience we refer to $\operatorname{par} \partial f(x)$ as the *V-space* for f at x (with respect to $\mathcal{M}$), and to its orthogonal complement, the subspace tangent to $\mathcal{M}$ at x, as the *U-space* for f at x. When we refer to the V-space and U-space without reference to a point x, we mean at a minimizer. For nonzero y in the V-space, the mapping $t \mapsto f(x + ty)$ is necessarily nonsmooth at $t = 0$, while for nonzero y in the U-space, $t \mapsto f(x + ty)$ is differentiable at $t = 0$ as long as f is locally Lipschitz.

For example, the norm function is partly smooth at 0 with respect to the trivial manifold $\{0\}$, the V-space at 0 is $\mathbf{R}^n$, and the U-space is $\{0\}$. When f is convex, the partly smooth nomenclature is consistent with the usage of V-space and U-space in [LOS00]. Most of the functions that we have encountered in applications are partly smooth at local optimizers with respect to some manifold, but many of them are not convex.

5.3. Polyhedral Functions. It is shown in [LO08] that the BFGS algorithm with an exact line search may fail on a polyhedral function. However, we have never observed BFGS with the inexact line search to fail in this manner when applied to polyhedral functions, including the counterexample given in [LO08] (which is unbounded below). For polyhedral functions that are bounded below, BFGS with the inexact line search typically exhibits linear, although often slow, convergence of the function values to the optimal value. We omit details, as other examples are more interesting.

5.4. The Tilted Norm Function. Consider the Euclidean norm function on $\mathbf{R}^n$ tilted by a linear term:

$$f(x) = w\|x\| + (w - 1)e_1^T x$$

where e_1 is the first coordinate vector and $w \geq 1$. The only minimizer is the origin, the V-space is $\mathbf{R}^n$ and the U-space is $\{0\}$. The case $n = 1$ is a variant of the tilted absolute value function discussed in Section 4.

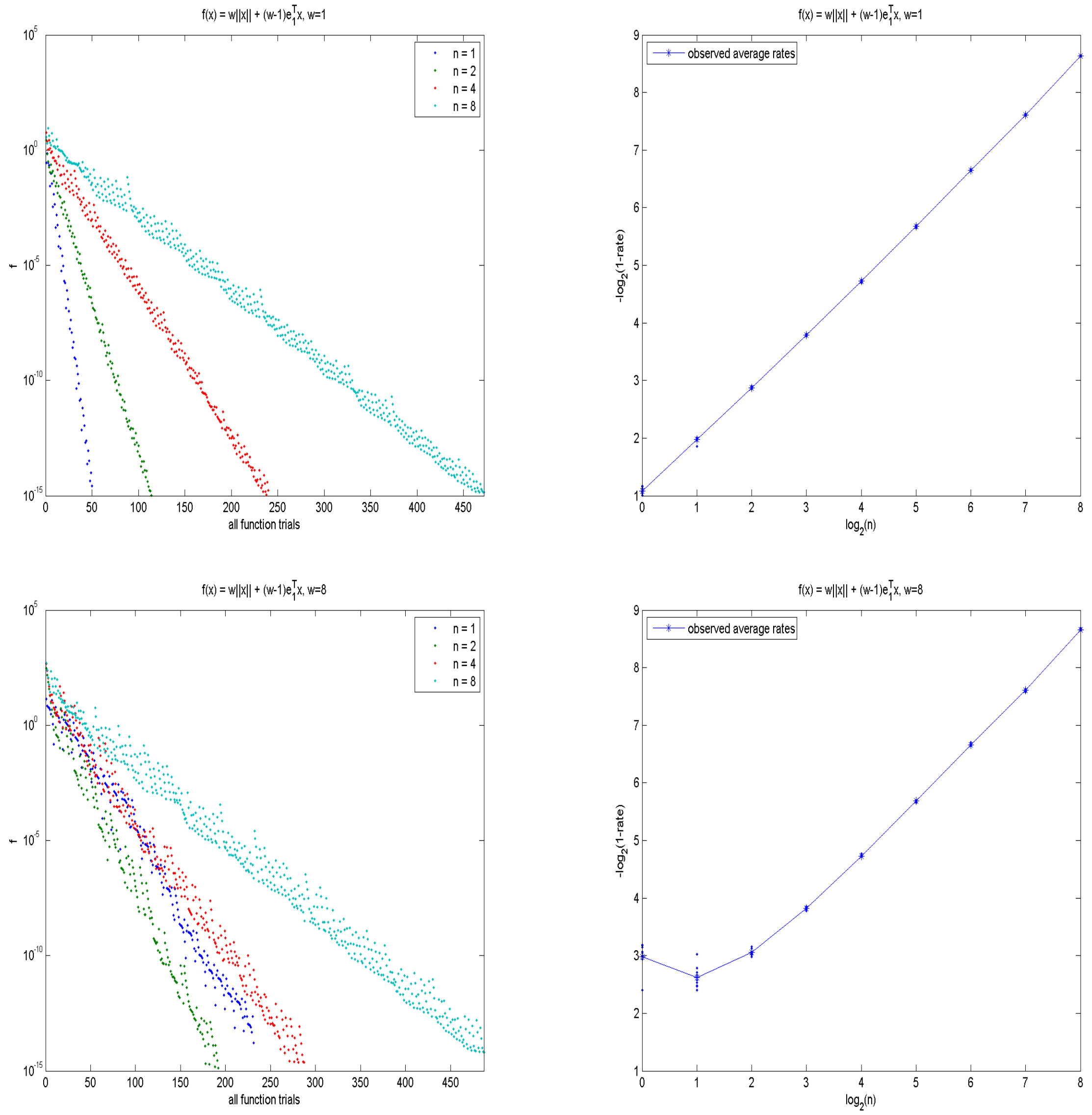

FIG. 5.1. *BFGS with the inexact line search on $f(x) = w\|x\| + (w-1)e_1^T x$ for varying n. Top left: typical runs for $n = 1, 2, 4, 8$ and $w = 1$ showing all function trial values. Top right: plots $-\log_2(1-r)$, where r is the average observed convergence rate with respect to the number of function trials, against $\log_2(n)$. Bottom left and right: same for $w = 8$.*

Figure 5.1 shows the behavior of BFGS with the inexact line search when w is fixed and n is varied. The top two panels show results for the untilted norm ($w = 1$). The top left panel shows all function values computed by the algorithm, including trial values in the line search, for typical runs for $n = 1, 2, 4$ and 8. The sequences of function trial values appear to be R-linear: in terms of a semi-log plot such as this, the convergence of a sequence is R-linear with rate $\tilde{r}$ if $\log_{10} \tilde{r}$ is the infimum of the slopes of all lines that bound the points from above. However, our real interest is in the rate of convergence of those function values that are *accepted* by the line search, taking into account nonetheless the number of function evaluations required by the line search: this rate is r if $\log_{10} r$ is the infimum of the slopes of all lines bounding the points corresponding to *accepted* function values from above. We see from the figure that, for these sequences, the rates $\tilde{r}$ and r are approximately equal. For this reason we choose to *estimate* the convergence rate of the function trial values as we explained in Section 4, using a least squares fit to the pairs (ν^k, f^k), where f^k is the function value at the end of the kth line search and ν^k is the cumulative number of function trials up to that point.

The top right panel of Figure 5.1 shows the estimated linear convergence rates r computed in this way, averaged over 10 runs. As in the previous section, we plot $-\log_2(1-r)$ against $\log_2(n)$. The observed convergence rates are amazingly consistent and we see that r is well described by $1 - 1/(2n)$. It is interesting to compare this to the convergence rate with respect to the number of *exact* line searches for the same problem, which was observed in [LO08] to be somewhat greater than $1 - 1/\sqrt{2n}$. The discrepancy between these rates is due to the fact that the average number of function trials needed in an inexact line search grows with n, as can be seen in the top left panel of Figure 5.1.

The bottom left and right panels of Figure 5.1 show the same information for $w = 8$. We observe that, as we saw in Section 4 for $n = 1$, the larger value of w causes deterioration in the rate of convergence for small n, but this deterioration vanishes rapidly as n grows. This observation is confirmed by other experiments that are not shown here.

We also carried out a number of experiments minimizing $f(Ax)$ where A is a non-singular matrix. Remarkably, we found that the results were essentially independent of A, for fixed n. One might suspect that this property extends to any positively homogenous function, but experiments with polyhedral examples indicate that this is not the case.

5.5. A Convex Partly Smooth Function.

Consider the function

$$f(x) = \sqrt{x^T A x} + x^T B x.$$

where A and B are positive semidefinite and at least one of them is positive definite, so the origin is the unique minimizer. If we take $A = I$ and $B = 0$, f reduces to $\|x\|$. Now let $A = \mathrm{diag}(1, 0, 1, 0, \ldots)$ and choose B to be positive definite. Then f is partly smooth at 0 with respect to the manifold

$$\mathcal{M} = \{x : x_{2j-1} = 0, j = 1, \ldots, \frac{n}{2}\}.$$

This manifold is linear, so the U-space is $\mathcal{M}$ and the V-space is $\mathcal{M}^\perp$.

Figure 5.2 shows the behavior of BFGS when $B = I$, the identity matrix. As previously, x and H were initialized randomly. In the top left panel, we see convergence of the function trial values to zero for typical runs for $n = 2, 4, 8$. At the top right, we see the convergence of the iterate components x_j, $j = 1, \ldots, n$, for the case $n = 8$. Note that the odd components converge to zero in advance of the even components, reflecting the property that f grows away from the origin with the absolute value of the odd coordinates, as opposed to the square of the even coordinates. In the bottom left panel, we see that four of the eigenvalues of H converge to zero, and the other four remain bounded away from zero. The eigenvectors of the final H corresponding to the tiny eigenvalues span the V-space $\mathcal{M}^\perp$ and the eigenvectors corresponding to

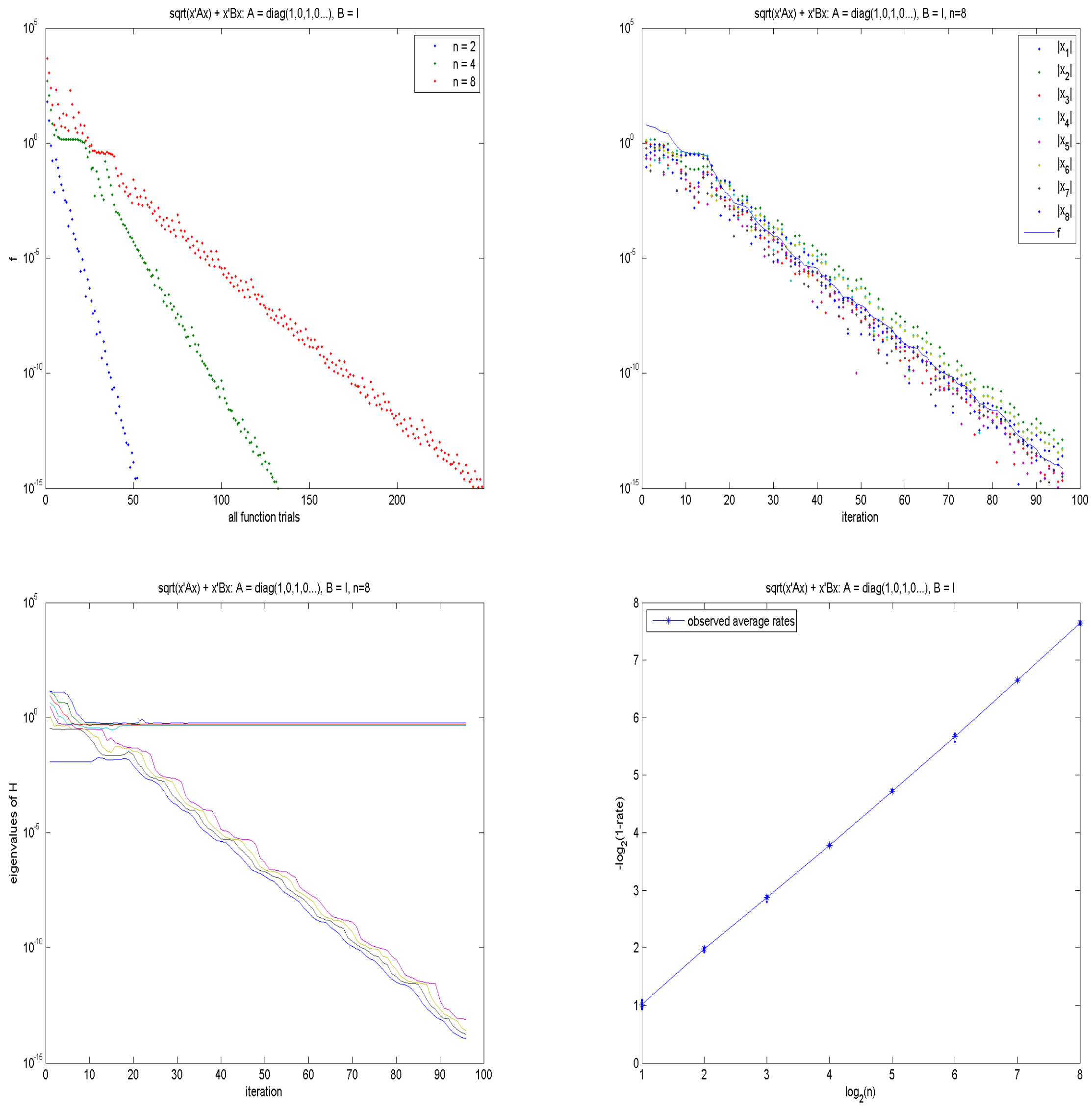

FIG. 5.2. *BFGS with the inexact line search on* $f(x) = \sqrt{x^T Ax} + x^T Bx$, *with* $A = diag(1, 0, 1, 0, \ldots)$ *and* $B = I$. *Top left: typical runs for* $n = 1, 2, 4, 8$, *showing all function trial values. Top right:* $|x_j|$, $j = 1, \ldots, n$, *after each line search, for* $n = 8$. *Bottom left: eigenvalues of* H *after each line search, for the same run with* $n = 8$. *Bottom right: plots* $-\log_2(1 - r)$, *where* r *is the average observed convergence rate with respect to the number of function trials, against* $\log_2(n)$.

the eigenvalues bounded away from zero span the U-space $\mathcal{M}$ (up to rounding error, not shown in the figures). The bottom right panel shows how the convergence rate with respect to the number of function trials varies with n. This time we see that the rate r is approximately $1 - 1/n$. This is consistent with the result for the norm function in the sense that in both cases, the rate is approximately $1 - 1/(2d)$ where d is the dimension of the V-space.

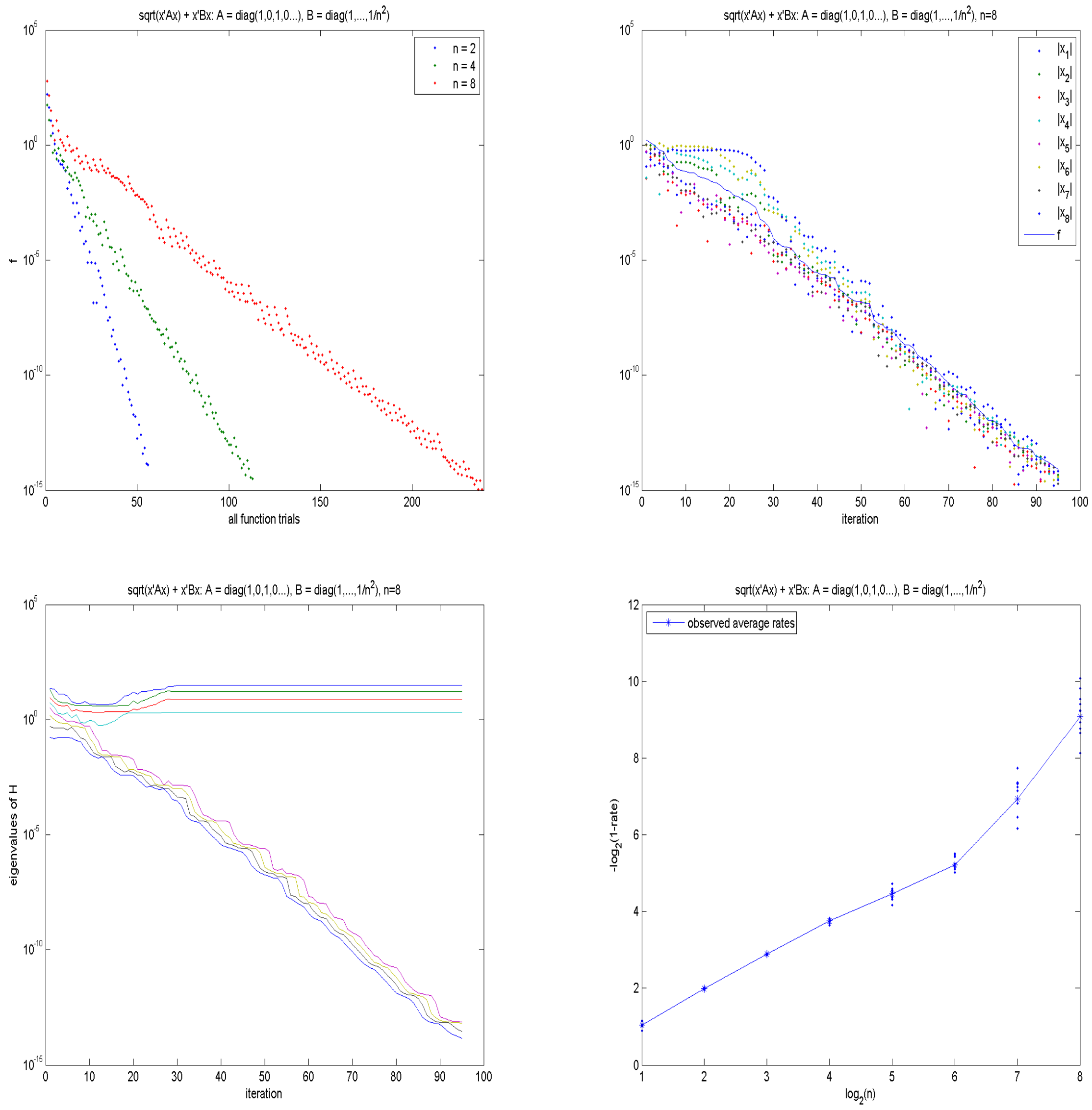

FIG. 5.3. *Same as Figure 5.2, except that* $B = diag(1, \ldots, 1/n^2)$.

Figure 5.3 shows the same information for the case $B = \mathrm{diag}(1, \ldots, 1/n^2)$. The top right panel shows, as previously, that the odd components x_j converge to zero in advance of the even components, but this time the even components lag further behind. We explain this as follows: the initial changes in H reflect the more important odd components, but once these are resolved it still takes some time to resolve the even components, as B is no longer the identity. Likewise, in the bottom left we see that it takes some time to resolve the different magnitudes of the eigenvalues of H that are bounded away from zero. Qualitatively similar figures are obtained repeatedly for runs with different random initializations. The convergence rates observed in the

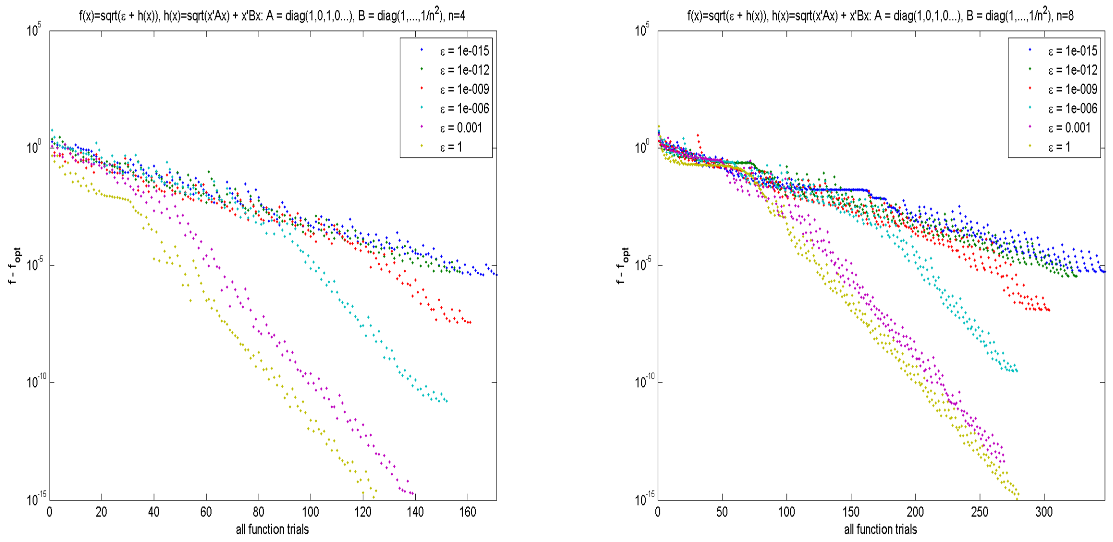

FIG. 5.4. *BFGS with the inexact line search on* $f(x) = (\epsilon + (x^T Ax)^{1/2} + x^T Bx)^{1/2}$*, with* $A = diag(1, 0, 1, 0, \ldots)$ *and* $B = diag(1, \ldots, 1/n^2)$*. Left: typical runs for* $n = 4$ *with* $\epsilon = 10^k$*,* $k = -15, -12, \ldots, 0$*, showing all function trial values. Top right: same for* $n = 8$*.*

bottom right panel are not as consistent as they are for $B = I$, perhaps because of rounding errors.

5.6. A Nonconvex Partly Smooth Function. Consider the function

$$f(x) = \sqrt{\epsilon + \sqrt{x^T Ax} + x^T Bx}$$

where $A = \text{diag}(1, 0, 1, 0, \ldots)$, $B = \text{diag}(1, \ldots, 1/n^2)$ and $\epsilon > 0$. This function is nonconvex for $\epsilon < 1$, and its Lipschitz constant is $O(\epsilon^{-1/2})$ as $\epsilon \downarrow 0$. It is partly smooth with respect to the same manifold as in the previous example.

Figure 5.4 shows the behavior of BFGS with the inexact line search on this example for varying ϵ, with $n = 4$ on the left and $n = 8$ on the right. Remarkably, the convergence rates appear to be independent of ϵ, though for smaller values of ϵ, rounding errors limit the achievable accuracy. The value $\epsilon = 10^{-15}$ is near the machine precision and hence the function is effectively non-Lipschitz; nonetheless, BFGS is able to reduce f to about 10^{-5}.

5.7. A Nonsmooth Rosenbrock Function. Consider the following nonsmooth variant of the well known Rosenbrock function in two variables:

$$f(x) = w|x_2 - x_1^2| + (1 - x_2)^2,$$

which is partly smooth with respect to the manifold $\mathcal{M} = \{x | x_2 = x_1^2\}$. This manifold is not linear, as was the case in the previous example. The iterates rapidly approach $\mathcal{M}$ and then, much more slowly, "track" $\mathcal{M}$, that is they follow a path close to but not on $\mathcal{M}$, converging to the minimizer $[1, 1]^T$. This is clearly seen in the contour plot for $w = 8$ at the top left of Figure 5.5. The different colored points and line segments indicate the path taken from 7 randomly generated initial points to the minimizer,

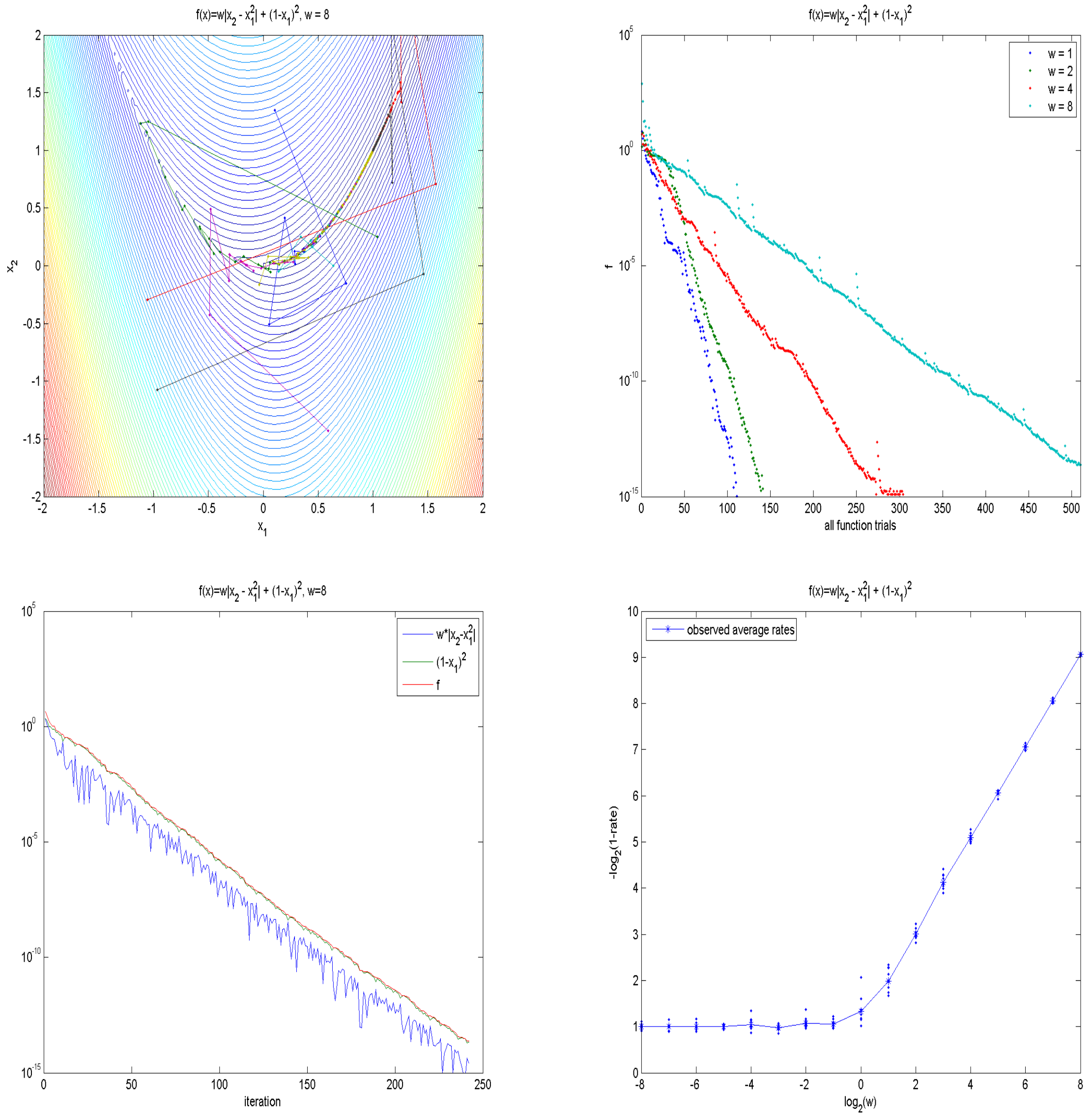

Fig. 5.5. *BFGS with the inexact line search on the nonsmooth Rosenbrock function* $f(x) = w|x_2 - x_1^2| + (1 - x_2)^2$*. Top left: contour plot for* $w = 8$*, showing the iterates generated (colored points connected by line segments) for 7 different starting points. Top right: typical runs for* $w = 1, 2, 4$ *and 8 showing all function values. Bottom left: nonsmooth and smooth components of* f *after each line search. Bottom right: plots* $-\log_2(1 - r)$*, where* r *is the average observed convergence rate with respect to the number of function trials, against* $\log_2(w)$*.*

with H initialized to the identity matrix so that the first step is in the direction of steepest descent. Note that the colors plotted later (black being the latest) overwrite previously plotted points. At the top right, we see linear convergence with respect to the number of function trials for typical runs with $w = 1, 2, 4$ and 8. At the bottom left, we see the evolution of the smooth and nonsmooth components of f after each line search for $w = 8$. At the bottom right, observed convergence rates are shown as a function of w. We see that the rate is approximately $1 - 1/(2w)$ for large w, and close to $1/2$ for small positive w. When $w = 0$, the convergence is superlinear, but this is not the case for any positive value. Of course, if w is sufficiently small, superlinear convergence is apparently observed due to limits of machine precision.

5.8. Nesterov's Chebyshev-Rosenbrock Functions. Nesterov recently introduced the following smooth function:

$$\tilde{f}(x) = \frac{1}{4}(x_1 - 1)^2 + \sum_{i=1}^{n-1}(x_{i+1} - 2x_i^2 + 1)^2.$$

A nonsmooth variation is

$$\hat{f}(x) = \frac{1}{4}(x_1 - 1)^2 + \sum_{i=1}^{n-1}|x_{i+1} - 2x_i^2 + 1|.$$

In both cases the only minimizer is $\bar{x} = [1, 1, 1, \ldots, 1]^T$. Consider the point $\hat{x} = [-1, 1, 1, \ldots, 1]^T$ and the manifold

$$\mathcal{M} = \{x : x_{i+1} = 2x_i^2 - 1, \quad i = 1, \ldots, n-1\}$$

which contains both $\bar{x}$ and $\hat{x}$. For $x \in \mathcal{M}$,

$$x_{i+1} = 2x_i^2 - 1 = T_2(x_i) = T_{2^i}(x_1) = \cos(2^i \cos^{-1}(x_1)), \quad i = 1, \ldots, n-1,$$

where $T_i(x)$ denotes the ith Chebyshev polynomial.

The functions $\tilde{f}$ and $\hat{f}$ are both sums of a quadratic term and a nonnegative sum whose zero set is the manifold $\mathcal{M}$. Minimizing either function is equivalent to minimizing the first quadratic term on $\mathcal{M}$. Typically, BFGS generates iterates that, as in the Rosenbrock example, approach $\mathcal{M}$ relatively rapidly and then track $\mathcal{M}$ to approximate the minimizer. The iterates do not track $\mathcal{M}$ exactly, even if they are initialized at $\hat{x}$, but because they typically follow the highly oscillatory manifold $\mathcal{M}$ fairly closely, particularly in the nonsmooth case, this tracking process requires many iterations. To move from $\hat{x}$ to $\bar{x}$ along $\mathcal{M}$ *exactly* would require x_n to trace the graph of the 2^{n-1}th Chebyshev polynomial, which has $2^{n-1} - 1$ extrema in $(-1, 1)$, as x_1 increases from -1 to 1.

Indeed, for $n = 8$, initializing x to $\hat{x}$ and H to the identity matrix, BFGS with the inexact line search requires about 6700 iterations to reduce the *smooth* function $\tilde{f}$ below 10^{-15}, and for $n = 10$, nearly 50,000 iterations are needed.

Minimizing the nonsmooth function $\hat{f}$ is, not surprisingly, much more difficult. This function is partly smooth with respect to $\mathcal{M}$ at all points in $\mathcal{M}$. The codimension of $\mathcal{M}$ is $n - 1$, so the dimension of the U and V-spaces at $\bar{x}$ are 1 and $n - 1$ respectively. To run BFGS on $\hat{f}$, we cannot use $\hat{x}$ for the initial point as the method immediately breaks down, $\hat{f}$ being nondifferentiable at $\hat{x}$. Instead, we initialize x randomly, retaining the identity matrix for initializing H. We find that we can usually solve the problem reasonably accurately (reducing f to about 10^{-8} before breakdown occurs) when $n = 3$, but not when $n = 4$, for which the method typically breaks down far from $\bar{x}$. We conjecture that the reason for this is rounding error, not a failure of the method to converge in theory. The final iterate x is very close to $\mathcal{M}$, and the final matrix H has $n - 1$ tiny eigenvalues as expected, but the method is unable to track $\mathcal{M}$ to minimize $\hat{f}$.

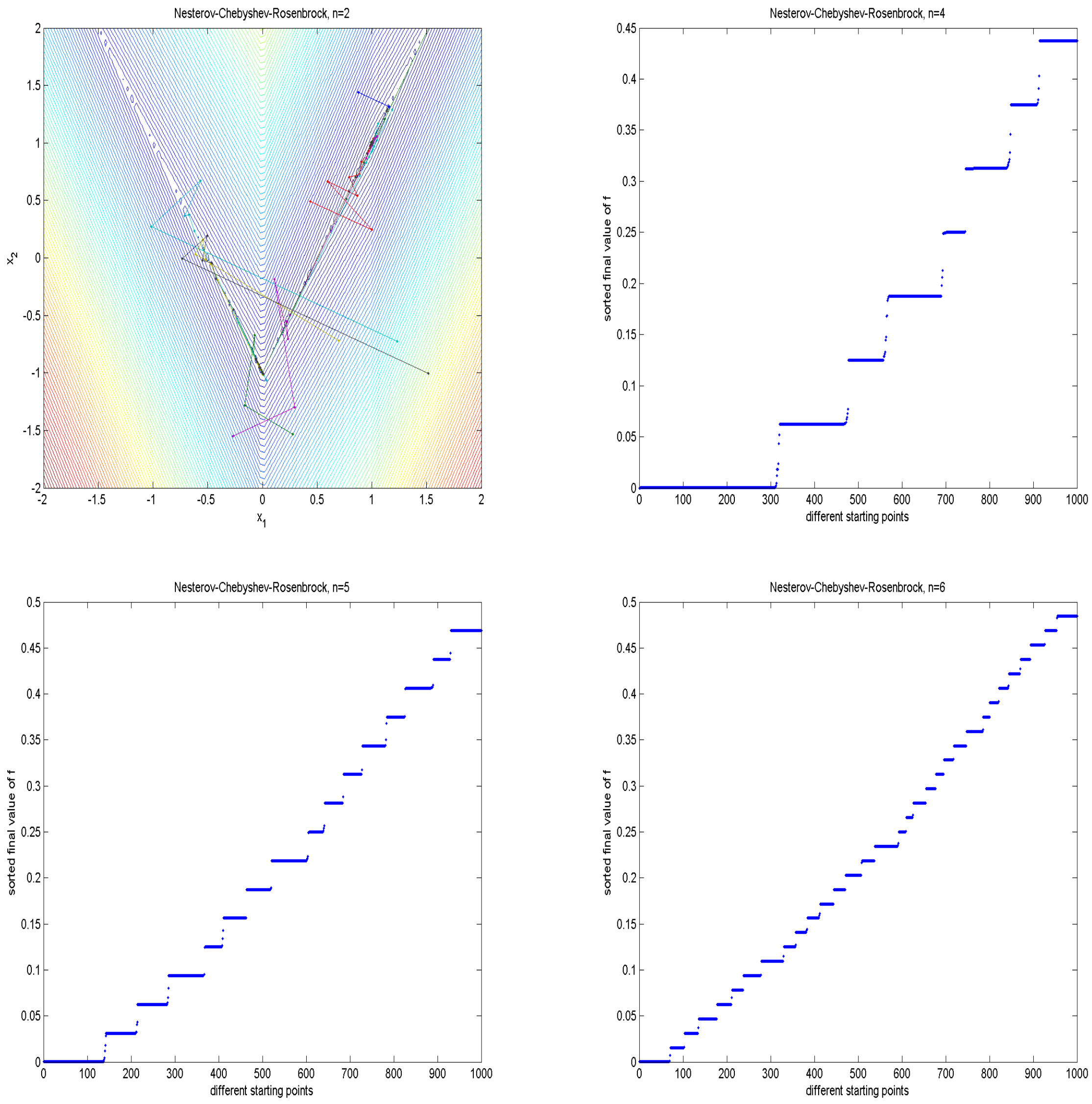

FIG. 5.6. *Top left: contour plot for Nesterov's second nonsmooth Chebyshev-Rosenbrock function f, with $n = 2$. Colored points connected by line segments show the iterates generated by BFGS for 7 different starting points. The Clarke stationary points at $[0, -1]^T$ and $[1, 1]$ are both attractors for the iteration, but only the latter is a minimizer. Top right: sorted final values of f for 1000 randomly generated starting points, $n = 4$. Bottom left and right: same for $n = 5$ and $n = 6$ respectively.*

Now consider a second nonsmooth variant, also suggested by Nesterov:

$$f(x) = \frac{1}{4}|x_1 - 1| + \sum_{i=1}^{n-1} |x_{i+1} - 2|x_i| + 1|.$$

Again, the only minimizer is $\bar{x}$. Consider the set

$$S = \{x : x_{i+1} = 2|x_i| - 1, \quad i = 1, \dots, n-1\}.$$

Minimizing f is equivalent to minimizing its first term on S, and BFGS typically generates iterates that rapidly approach S and then need to track S to approximate $\bar{x}$. Like $\mathcal{M}$, the set S is highly oscillatory, but it has "corners": it is not a manifold around any point x where any of the components $x_1, \dots, x_{n-1}$ vanishes. For example, consider the case $n = 2$, for which a contour plot is shown at the top left of Figure 5.6. It is easy to verify that the point $[0, -1]^T$ is Clarke stationary (zero is in the convex hull of gradient limits at the point), but not a local minimizer ($[1, 2]^T$ is a direction of linear descent from $[0, -1]^T$). Thus, f is not regular at $[0, -1]^T$. The contour plot shows the path of the iterates generated by BFGS using 7 random starting points, plotted in 7 different colors (as previously, the points plotted later overwrite many of those plotted earlier near the attractors). Most of the runs converge to the minimizer $[1, 1]^T$, but some converge to the Clarke stationary point $[0, -1]^T$.

It is not hard to see that, in general, there are $2^{n-1} - 1$ points in S where x_j vanishes for some $j < n$. For $n \le 6$, given enough randomly generated starting points, BFGS finds all these points, in addition to the minimizer. The top right, bottom left and bottom right panels plot final values of f found by 1000 runs of BFGS starting with random x and $H = I$, sorted into increasing order, for the cases $n = 4, 5$ and 6 respectively. Most runs find either the minimizer or one of the $2^{n-1} - 1$ nonminimizing points described above, although a few runs break down earlier. For $n = 7$, the method usually breaks down far away from these points, again presumably because of the limitations of machine precision. It seems likely that each of these $2^{n-1} - 1$ points is Clarke stationary and not Clarke regular, although we have not verified this algebraically.

Our work on this problem is in response to suggestions from both Y. Nesterov and K. Kiwiel. The latter informed us that this example is the only one he knows for which his bundle code converges to nonminimizing Clarke stationary points, thus raising our interest in this issue.

5.9. An Eigenvalue Product Application. All previous examples are contrived, chosen to illustrate various points but for which the solution is known. Our final example is a nonconvex relaxation of an entropy minimization problem arising in an environmental application [AL04]. Let $\mathcal{S}^N$ denote the space of real symmetric N by N matrices. The function f to be minimized is

$$f(X) = \log E_K (A \circ X), \tag{5.1}$$

where $E_K(X)$ denotes the product of the K largest eigenvalues of a matrix X in $\mathcal{S}^N$, A is a fixed matrix in $\mathcal{S}^N$, and $\circ$ denotes the Hadamard (componentwise) matrix product, subject to the constraints that X is positive semidefinite and has diagonal entries equal to 1. If the requirement was to minimize the *sum* of the largest eigenvalues instead of the product, this would be equivalent to a semidefinite program, but the product of the largest K eigenvalues is not convex. This problem was one of the examples in [BLO05]; in the results reported there, the objective function was defined without the logarithm and we enforced the semidefinite constraint by an exact penalty function. It turns out to be much more favorable for the convergence of BFGS to impose the constraint by the substitution $X = VV^T$, where V is square. The constraint on the diagonal of X then translates to a requirement that the rows of V have norm one, a constraint

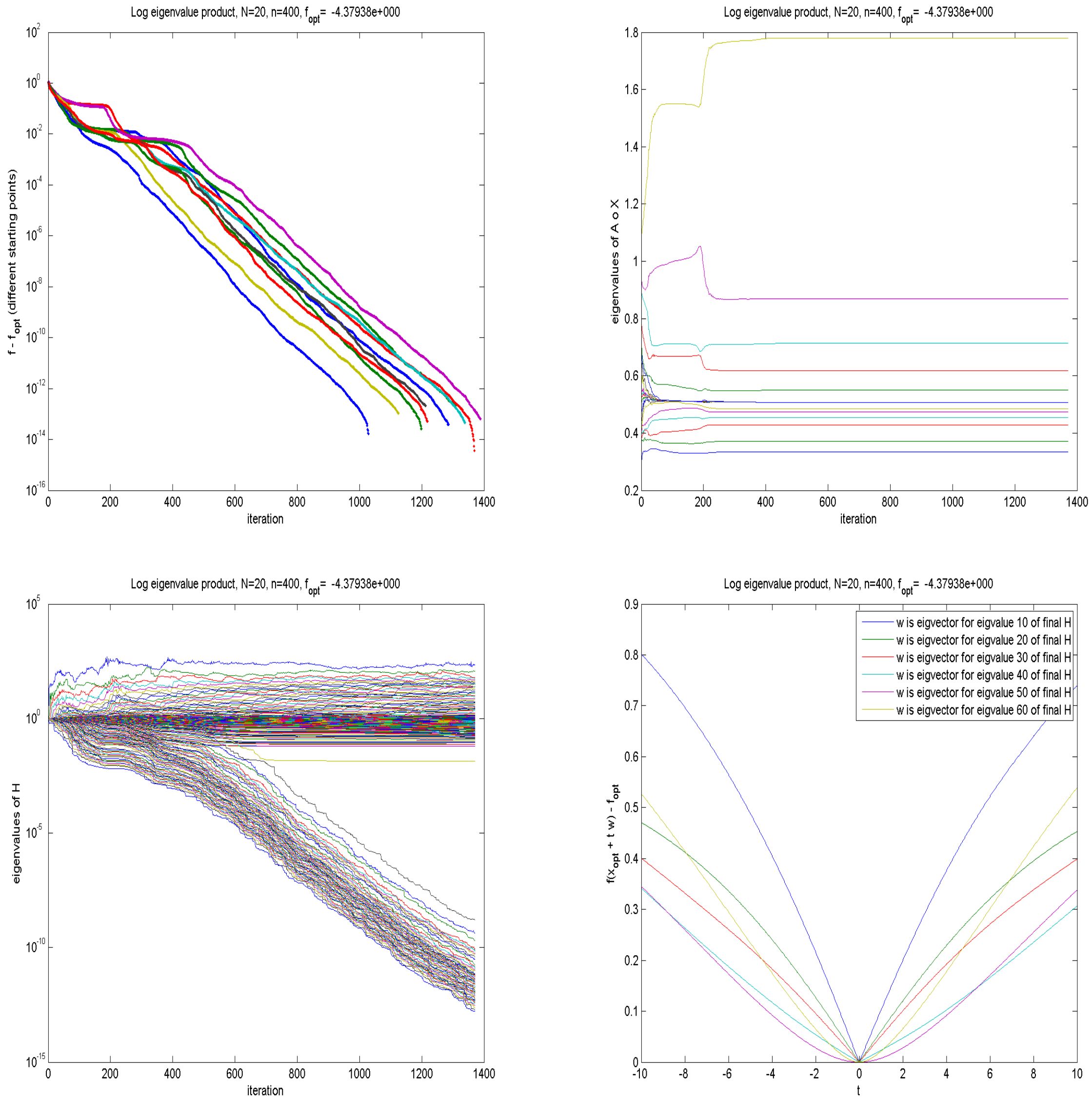

FIG. 5.7. . *Results for minimizing the eigenvalue product, $N = 20$, $n = 400$. Top left: the function values after each line search for 10 randomly generated starting points, shifted by $f_{\rm opt}$, the minimal value found. Top right: eigenvalues of $A \circ X$ after each line search for one run. Bottom left: eigenvalues of H for same run: 44 of these converge to zero. Bottom right: plots $f - f_{\rm opt}$ along a line $x_{\rm opt} + tw$, where $x_{\rm opt}$ is the computed minimizer and w is the eigenvector of H associated with its jth smallest eigenvalue, for $j = 10, 20, \ldots, 60$. The function f is "V-shaped" along the eigenvectors associated with tiny eigenvalues of H, and "U-shaped" along the others.*

that can be easily removed from the problem, redefining f appropriately. Thus, the problem is converted to the unconstrained minimization of a nonsmooth function f over $\mathbf{R}^n$ with $n = N^2$ (the variable being $x = \mathrm{vec}(V)$, the vector representation of the matrix V). In principle, one might expect multiple local minimizers with different minimal values, but at least with the data we have been using, this situation seems to occur relatively rarely, although occasionally runs from different starting points find two different minimal values. (Multiple local minimizers with the same minimal value frequently arise because of the redundant variables in the parametrization.)

Let $\lambda_i(Y)$ denote the ith largest eigenvalue of $Y \in \mathcal{S}^N$ and, for given Y, define an active set $I(Y) = \{i : \lambda_i(Y) = \lambda_K(Y)\}$. Then E_K is partly smooth at Y with respect to the manifold $\widetilde{\mathcal{M}}(Y) = \{Z \in \mathcal{S}^N : \lambda_i(Z) = \lambda_K(Z),\ \forall i \in I(Y)\}$. It is known from matrix theory that the codimension of $\widetilde{\mathcal{M}}(Y)$ is $m(m+1)/2 - 1$, where m is the multiplicity $|I(Y)|$ [Lax97, p.141]. Now consider the manifold in $\mathbf{R}^n$ defined by

$$\mathcal{M}(\bar{x}) = \left\{x : A \circ \mathrm{vec}(x)\mathrm{vec}(x)^T \in \widetilde{\mathcal{M}}\left(A \circ \mathrm{vec}(\bar{x})\mathrm{vec}(\bar{x})^T\right)\right\},$$

where $\bar{x}$ is a minimizer of f. To conclude that f is partly smooth with respect to $\mathcal{M}$ at $\bar{x}$, and that the codimension of $\mathcal{M}$ is $m(m+1)/2 - 1$, where $m = |I(A \circ \mathrm{vec}(\bar{x})\mathrm{vec}(\bar{x})^T)|$, requires a transversality condition [Lew03]; let us assume that this holds. For the results reported below, A is set to the leading $N \times N$ submatrix of a 63×63 covariance matrix [AL04], scaled so that the largest entry is 1.

Figure 5.7 shows the results for $N = 20$. At the top left, the values of f after each line search are plotted, shifted by f_{opt}, an estimate of the optimal value, for 10 different randomly generated starting points. Since the exact optimal value is not known, we set f_{opt} to the best value found in these 10 runs; the apparent superlinear convergence of f to the optimal value in this run is an artifact of this choice. We see that all starting points lead to nearly the same final value of f with the same rate of convergence. At the top right, we see the eigenvalues of $A \circ X$ as a function of the iteration count. Observe that after just a few iterations, $\lambda_6(A \circ X), \ldots, \lambda_{14}(A \circ X)$ have coalesced together to plotting accuracy ($\lambda_{15}, \lambda_{16}$ and λ_{17} are slightly smaller). This computed multiplicity–9 eigenvalue suggests that the manifold $\mathcal{M}(\bar{x})$ has codimension $9(10)/2 - 1 = 44$; if so, this is the dimension of the V-space at $\bar{x}$. Indeed, this is confirmed by the bottom left plot: exactly 44 eigenvalues of the inverse Hessian approximation matrix H converge to zero! Furthermore, at the bottom right we see the function $f - f_{\mathrm{opt}}$ plotted along lines through the computed minimizer x_{opt} parallel to the eigenvectors corresponding to the jth *smallest* eigenvalue of H, for $j = 10, 20, \ldots, 60$. We see that f is V-shaped in the first four of these directions and U-shaped in the last two, again consistent with our conclusion that the V-space has dimension 44. This is compelling evidence that BFGS *automatically* identifies the V and U-spaces at the minimizer.

As for the nonsmooth Rosenbrock example and the first nonsmooth Nesterov function $\hat{f}$, the manifold $\mathcal{M}(\bar{x})$ is nonlinear. Thus, one expects rapid convergence nearly to the manifold and much slower convergence tracking the manifold to the minimizer. Indeed, all 10 runs produce an eigenvalue of $A \circ X$ with multiplicity 9 to about 14 digits, about as many as one could expect given the precision being used, but the numerical value of this eigenvalue found by the 10 runs is consistent to only about 8 digits (0.50662367), indicating that although all runs find the optimal manifold to high precision, their approximation to the solution $\bar{x}$ has lower precision. If we were to modify the algorithm to impose $x \in \mathcal{M}$ as a *constraint*, we would surely be able to find $\bar{x}$ to higher precision. We could indeed take this approach for this problem since we know the equation defining $\mathcal{M}$ given the observed multiplicity. What is remarkable is that BFGS finds such a good approximation to $\bar{x}$ *without* any modifications of this sort.

Our work on this problem is due to suggestions from K. Anstreicher and J. Lee. It was on this example that we first observed the surprisingly consistent behavior of

BFGS on nonsmooth problems.

6. Alternative Methods and Software. The purpose of this paper is to explore the behavior of BFGS on nonsmooth functions, not to benchmark it against other methods. Nonetheless, we make some brief comments in this direction. We start by noting that huge nonsmooth convex problems are routinely solved in many applications by a variety of methods, notably bundle and interior point methods. While it is possible that a limited memory variant of BFGS might have a useful role to play in the nonsmooth convex case, this has not been systematically investigated. We are therefore concerned here only with algorithms applicable to small to medium-sized, nonsmooth, nonconvex problems.

6.1. Shor's R-Algorithm. Because of its simplicity, the easiest method to compare with BFGS is the Shor R-algorithm [Sho85]. Like BFGS, this can be viewed as a variable metric method, but one that does not satisfy the secant equation. In [BLO08], we presented the first proof that this algorithm is linearly convergent on some problems; however, the proof is limited to quadratics (using an exact line search) when $n = 2$. In contrast with the method of steepest descent, the rate of convergence is apparently independent of the conditioning of the quadratic. Not including the cost of function and gradient evaluations, the overhead per iteration is the same as BFGS: $O(n^2)$, for matrix-vector products.

We carried out experiments with the Shor R-algorithm using the same inexact line search that has already been described. We found that it works poorly compared to BFGS. One difficulty is that the algorithm requires the choice of a rather arbitrary parameter β [Sho85], equivalently $1 - \gamma$ [BLO08], in $(0, 1)$; convergence is slow if β is close to 0 or 1 and it is not clear how to best choose its value. A second difficulty is that if the Wolfe parameter c_2 is set to a positive number, as we argued is favorable for BFGS, the Shor algorithm often fails to converge even on simple examples. This is well known, as is the remedy: set $c_2 = 0$ to ensure that the directional derivative always changes sign in the line search, a property that BFGS does not require. Even using, say, $\beta = 1/2$ and $c_2 = 0$, the Shor algorithm often requires significantly more function trials in the line search compared to BFGS. For the Shor algorithm, the unit step is often not well scaled, as even the example $f(x) = |x|$ shows, but no other choice is obviously better.

Figure 6.1 shows results for the Shor R-algorithm, with $\beta = 1/2$ and $c_2 = 0$. In the top left and bottom left panels we show results for the tilted norm function of Section 5.4, with $A = I$, and w set to 1 and 8 respectively. These may be compared to the results for the same problem using BFGS shown in the top left and bottom left panels of Figure 5.1. In several cases the number of line search trials made by the Shor method increases steadily with the iteration number k. In such a case the convergence rate for all function trial values is not the same as the convergence rate (still with respect to all function trials) for those function values that are accepted by the line search.

The top right panel of the same figure shows results for the convex partly smooth problem of Section 5.5, while the bottom right shows results for the nonsmooth Rosenbrock problem of Section 5.7. On these problems, the Shor algorithm does not take an excessive number of steps in the line search, but the convergence is slower than BFGS nonetheless, and does not appear to be R-linear, especially for the Rosenbrock problem. Compare with the results for BFGS in the top left panel of Figure 5.3 and the top right panel of Figure 5.5 respectively.

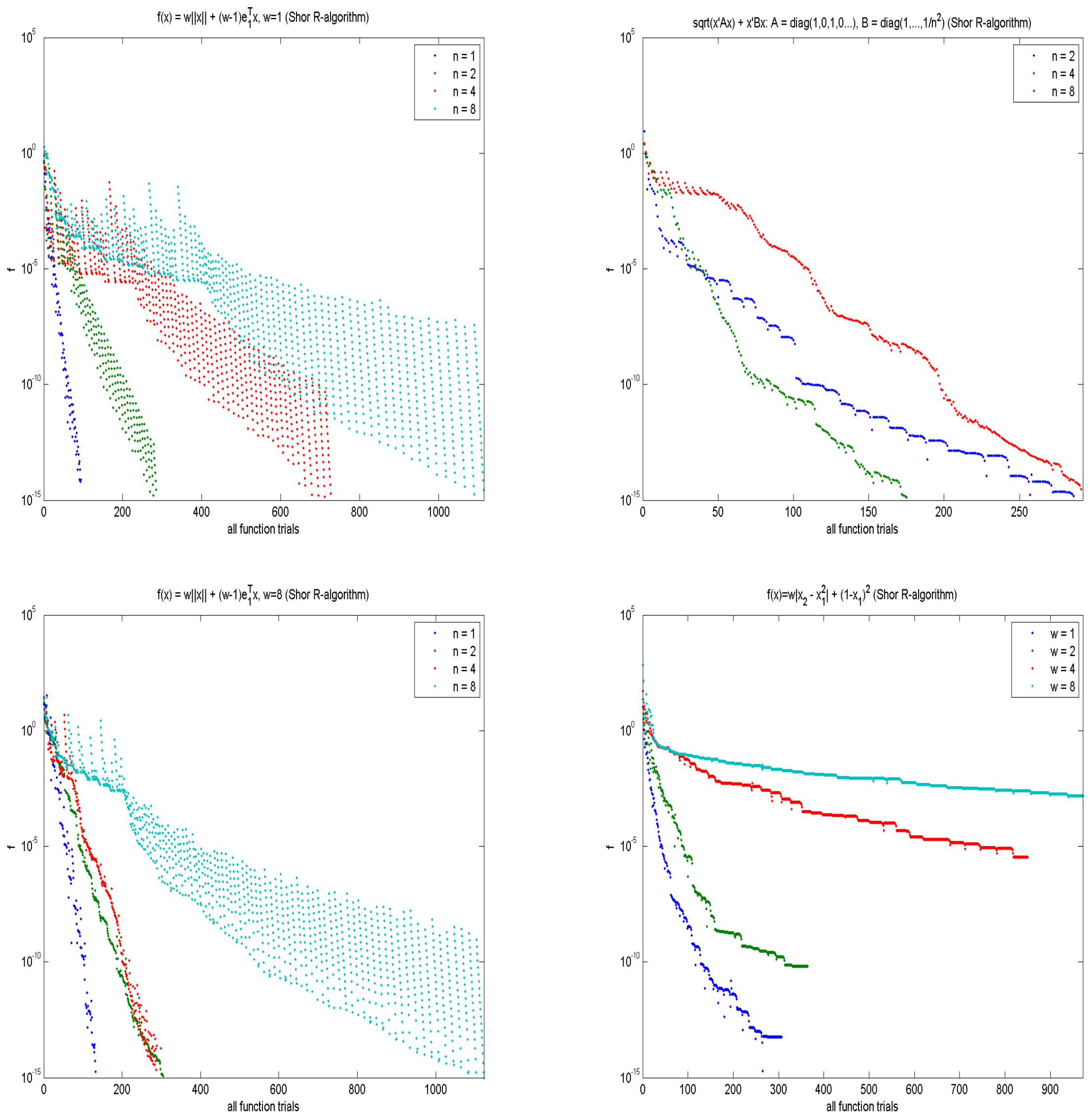

FIG. 6.1. *Results using the Shor R-Algorithm instead of BFGS, with $c_2 = 0$ in the inexact line search. Top and bottom left: same problem as in Figure 5.1, top and bottom left. Top right: same problem as in Figure 5.3, top left. Bottom right: same problem as in Figure 5.5, top right.*

We also experimented with a version of the Shor algorithm where we initialized the line search by scaling the search direction by the step taken the previous iteration. This seems to improve the method, but not a great deal.

We emphasize that these results are for the basic version of the Shor R-algorithm as defined in [Sho85, BLO08]. It may be that some of the ideas developed, for example, in [KK00] would result in improved performance.

6.2. Gradient Sampling. The gradient sampling algorithm enjoys fairly strong convergence results for locally Lipschitz functions [BLO05, Kiw07], although no rate of convergence has been established. Unfortunately, it becomes increasingly impractical as n grows, both because of its computational requirements (multiple gradient evaluations and the overhead of solving a convex quadratic program) and for theoretical reasons [Sha05].

In practice we find that BFGS is generally superior, particularly as n increases, although for difficult functions with large Lipschitz constants or that are non-Lipschitz or even discontinuous, as can often happen in applications, gradient sampling seems to have an advantage in robustness.

6.3. Bundle Methods. In Section 1, several references were given to bundle methods, especially those that incorporate variable metric updates. In the nonconvex case these methods are not easy to implement so we do not attempt to make comparisons here. The overhead in most bundle methods is the cost of solving a convex quadratic program, but as noted in Section 1, methods with lower overhead have been developed during the last decade; Haarala [Haa04] gives a good overview. We speculate that on medium-sized partly smooth problems with relatively small-dimensional V-spaces, there may be little to be gained by using the more complex bundle-variable-metric methods in preference to standard BFGS. At the opposite extreme, BFGS converges very slowly on large polyhedral functions while bundle methods terminate finitely. Thus, it does seem compelling that a method for general large-scale use should incorporate key ingredients of bundle methods.

6.4. HANSO. In Section 1, the quote from Lemaréchal alluded to the difficulty of assessing the result provided by BFGS. Our approach is simple: we run the method until it breaks down because of rounding errors, or until a preset iteration or time limit is exceeded, and view the resulting point as a candidate for a local minimizer. If the number of variables is not too large, one can then run a local bundle iteration to try to verify local optimality. If this fails, another possibility is to initiate gradient sampling, which, like ordinary bundle methods, returns information that can be used to assess how well the final point may approximate a local minimizer. Our freely available MATLAB code HANSO (Hybrid Algorithm for Non-Smooth Optimization)[2] is based on these ideas. HANSO is used by our code HIFOO (H-Infinity Fixed-Order Optimization)[3] [BHLO06] to design low-order controllers for linear dynamical systems, an important source of small-dimensional but difficult nonsmooth, nonconvex optimization problems. The BFGS code in HANSO has been used to solve other nonsmooth optimization problems as well, including the "condition geodesic" problem [BD08].

7. A Conjecture. This paper raises far more questions than it answers. We hope that we have made a convincing case that BFGS is a practical and effective method for nonsmooth optimization, and we have tried to give insight into *why* it works as well as it does.

In practice, for functions with bounded level sets, when initialized randomly, BFGS always seems to generate function values converging linearly to a Clarke stationary value. We speculate that, for some broad class of reasonably well-behaved functions, this behavior is almost sure. In framing a conjecture, let us first rule out the worst kinds of pathology by considering objective functions whose graphs stratify into analytic manifolds. (A variety of dynamical systems associated with such

[2]http://www.cs.nyu.edu/overton/software/hanso/
[3]http://www.cs.nyu.edu/overton/software/hifoo/

functions are known to behave well.) To be concrete, we consider the class of semi-algebraic functions, which includes all the examples given in this paper. Now let us consider appropriately random initial data: the precise distributions are irrelevant, providing they are absolutely continuous with respect to Lebesgue measure. Again to be concrete, let us assume a normally distributed intial point and an initial positive definite inverse Hessian approximation sampled from a Wishart distribution (that is, $H = XX^T$ where X is square with normally distributed entries.) In fact, this is how x and H were initialized in the experiments reported above. We now consider the BFGS method, in exact arithmetic, using the inexact line search defined in Section 2, for any fixed Armijo and Wolfe parameters satisfying $0 < c_1 < c_2 < 1$. Let us adopt the convention that if the algorithm generates a trial point at which f is not differentiable, then it terminates.

Conjecture 7.1. *Consider any locally Lipschitz, semi-algebraic function f, and choose x_0 and H_0 randomly. With probability one, the BFGS method generates an infinite sequence of iterates. Furthermore, any cluster point $\bar{x}$ of this sequence is Clarke stationary, that is $0 \in \partial f(\bar{x})$, and the sequence of all function trial values converges to $f(\bar{x})$ R-linearly.*

Acknowledgment. Mille grazie a F. Facchinei e agli altri membri del Dipartimento di Informatica e Sistemistica dell' Università di Roma "La Sapienza", dove gran parte di questo lavoro è stato eseguito, per avermi fornito un ambiente piacevole e stimolante.

Appendix A. Alternating Series Representation. Section 3 studies the behavior of the BFGS algorithm applied to the absolute value function, using the line search described in Section 2. This behavior turns out to depend on how the initial point can be represented as the sum of an alternating series of decreasing powers of two. We discuss this representation here, starting with a simple tool.

Lemma A.1. *If*

$$y = \sum_{k=0}^{m} (-1)^k 2^{-b_k},$$

for some $m = 2, 3, 4, \ldots$ or ∞ and some strictly increasing sequence of integers (b_j), then

$$2^{-b_0-1} < y < 2^{-b_0}.$$

Proof First notice that since y is the sum of an alternating series of strictly decreasing positive terms, we have

$$2^{-b_0} - 2^{-b_1} < y < 2^{-b_0}.$$

But $b_1 \geq b_0 + 1$, so

$$2^{-b_0} - 2^{-b_1} \geq 2^{-b_0} - 2^{-b_0-1} = 2^{-b_0-1},$$

whence the result. $\square$

The following routine result describes the representation we use.

Theorem A.2. *Any number $x \in \mathbf{R}_{++}$ has a unique* **alternating binary** *representation as the sum of a finite or infinite alternating series of strictly decreasing* *powers of two: that is, there is a unique number $m = 0, 1, 2, \ldots$ or ∞ and a unique sequence of integers $a_0 < a_1 < a_2 < \cdots$ (with $m+1$ terms if $m < \infty$) such that*

$$x = \sum_{k=0}^{m} (-1)^k 2^{-a_k}. \tag{1.3}$$

Proof Consider the following iterative procedure.

Algorithm A.4 (alternating binary representation).

```
    $s_{-1} \leftarrow 0$
    $m \leftarrow +\infty$
    $k \leftarrow 0$
    repeat
            $a_k \leftarrow \lfloor -\log_2[(-1)^k (x - s_{k-1})] \rfloor$
            $s_k \leftarrow s_{k-1} + (-1)^k 2^{-a_k}$
            if $s_k = x$, $m = k$, STOP
            $k \leftarrow k + 1$
    end(repeat)
```

We claim first that the algorithm is well-defined: that is, the partial sums

$$s_k = \sum_{j=0}^{k} (-1)^j 2^{-a_j} \tag{1.5}$$

satisfy

$$(-1)^k (x - s_{k-1}) > 0 \tag{1.6}$$

for all $k < m$. Notice that the sequence of integers (a_k) is uniquely defined by the inequalities

$$2^{-a_k - 1} < (-1)^k (x - s_{k-1}) \le 2^{-a_k}, \quad \text{for all finite } k \le m. \tag{1.7}$$

To prove our claim (1.6), note that the case $k = 0$ follows immediately from the inequalities (1.7). On the other hand, if (1.6) holds for some $k < m$, then inequalities (1.7) hold, so

$$\begin{aligned} (-1)^{k+1}(x - s_k) &= (-1)^{k+1}(x - [s_{k-1} + (-1)^k 2^{-a_k}]) \\ &= -(-1)^k (x - s_{k-1}) + 2^{-a_k} \ \ge \ 0, \end{aligned}$$

and in fact the inequality is strict since $k \ne m$. Our claim (1.6) now follows by induction.

We turn to proving the monotonicity of the sequence (a_k). Assuming $k < m$, by applying inequalities (1.7) once as they appear and once with k replaced by $k+1$, we deduce

$$\begin{aligned} 2^{-a_{k+1} - 1} \ < \ (-1)^{k+1}(x - s_k) &= (-1)^k (s_k - s_{k-1}) - (-1)^k (x - s_{k-1}) \\ &< (-1)^k (-1)^k 2^{-a_k} - 2^{-a_k - 1} \ = \ 2^{-a_k - 1}. \end{aligned}$$

Thus $a_{k+1} > a_k$, as required. Finally, as a consequence, notice that the inequalities (1.7) imply $s_k \to x$ as $k \to \infty$ if $m = \infty$, so the representation (1.3) follows.

To prove uniqueness of the expression (1.3), suppose

$$x = \sum_{k=0}^{\infty} (-1)^k 2^{-b_k}, \tag{1.8}$$

for some strictly increasing sequence of integers (b_j). We then claim $b_k = a_k$ for all indices k. (The proof for finite sums is similar.)

First notice that

$$2^{-b_0-1} < x < 2^{-b_0},$$

using Lemma A.1. Since inequalities (1.7) in the case $k = 0$ define a_k uniquely, we deduce $b_0 = a_0$.

We now proceed by induction. Suppose $b_j = a_j$ for all $j = 0, 1, 2, \ldots, k-1$. Then we have

$$s_{k-1} = \sum_{j=0}^{k-1} (-1)^j 2^{-b_j}$$

and hence

$$x - s_{k-1} = \sum_{j=k}^{\infty} (-1)^j 2^{-b_j}.$$

Using Lemma A.1 again now shows

$$2^{-b_k-1} < (-1)^k (x - s_{k-1}) < 2^{-b_k}.$$

Again, since inequalities (1.7) define a_k uniquely, we deduce $b_k = a_k$. □

REFERENCES

[AL04] K. Anstreicher and J. Lee. A masked spectral bound for maximum-entropy sampling. In A. di Bucchianico, H. Läuter, and H. P. Wynn, editors, *MODA 7 - Advances in Model-Oriented Design and Analysis*, pages 1–10. Springer, Berlin, 2004.

[BD08] P. Boito and J.-P. Dedieu. The condition metric in the space of rectangular full rank matrices. 2008. Submitted to SIAM Journal on Matrix Analysis and Applications.

[BGLS95] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal. A family of variable metric proximal methods. *Mathematical Programming*, 68:15–48, 1995.

[BHLO06] J.V. Burke, D. Henrion, A.S. Lewis, and M.L. Overton. HIFOO - a MATLAB package for fixed-order controller design and H_∞ optimization. In *Fifth IFAC Symposium on Robust Control Design, Toulouse*, 2006.

[BLO05] J.V. Burke, A.S. Lewis, and M.L. Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15:751–779, 2005.

[BLO08] J.V. Burke, A.S. Lewis, and M.L. Overton. The speed of Shor's R-algorithm. *IMA Journal on Numerical Analysis*, 28:711–720, 2008.

[Cla83] F. H. Clarke. *Optimization and Nonsmooth Analysis.* John Wiley, New York, 1983. Reprinted by SIAM, Philadelphia, 1990.

[Haa04] M. Haarala. *Large-Scale Nonsmooth Optimization: Variable metric bundle method with limited memory.* PhD thesis, University of Jyväskylä, Finland, 2004.

[HUL93] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms.* Springer-Verlag, New York, 1993. Two volumes.

[Kiw85] K.C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization.* Lecture Notes in Mathematics 1133. Springer-Verlag, Berlin and New York, 1985.

[Kiw07] K.C. Kiwiel. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 18:379–388, 2007.

[KK00] F. Kappel and A. Kuntsevich. An implementation of Shor's r-algorithm. *Computational Optimization and Applications*, 15:193–205, 2000.

[Lax97] P. D. Lax. *Linear Algebra.* John Wiley, New York, 1997.

[Lee98] J. Lee. Constrained maximum-entropy sampling. *Operations Research*, 46:655–664, 1998.

[Lem81] C. Lemaréchal. A view of line searches. In *Optimization and optimal control (Proc. Conf. Math. Res. Inst., Oberwolfach, 1980)*, pages 59–78, Berlin and New York, 1981. Springer. Lecture Notes in Control and Information Sciences, 30.

[Lem82] C. Lemaréchal. Numerical experiments in nonsmooth optimization. In E.A. Nurminski, editor, *Progress in Nondifferentiable Optimization*, pages 61–84, Laxenburg, Austria, 1982.

[Lew03] A.S. Lewis. Active sets, nonsmoothness and sensitivity. *SIAM Journal on Optimization*, 13:702–725, 2003.

[LF01] D.-H. Li and M. Fukushima. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 11:1054–1064, 2001.

[LO08] A.S. Lewis and M.L. Overton. Behavior of BFGS with an exact line search on nonsmooth examples. 2008. Submitted to SIAM Journal on Optimization.

[LOS00] C. Lemaréchal, F. Oustry, and C. Sagastizábal. The U-Lagrangian of a convex function. *Trans. Amer. Math. Soc.*, 352:711729, 2000.

[LS94] C. Lemaréchal and C. Sagastizábal. An approach to variable metric bundle methods. In *IFIP Proceedings, Systems Modeling and Optimization*, 1994.

[LV99] L. Lukšan and J. Vlček. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications*, 102:593–613, 1999.

[LV01] L. Lukšan and J. Vlček. Variable metric methods for nonsmooth optimization. Technical Report 837, Academy of Sciences of the Czech Republic, May 2001.

[Mif77] R. Mifflin. An algorithm for constrained optimization with semismooth functions. *Mathematics of Operations Research*, 2:191–207, 1977.

[MSQ98] R. Mifflin, D. Sun, and L. Qi. Quasi-Newton bundle-type methods for nondifferentiable convex optimization. *SIAM Journal on Optimization*, (2):583–603, 1998.

[NW06] J. Nocedal and S. Wright. *Nonlinear Optimization.* Springer, New York, second edition, 2006.

[Pow76] M.J.D. Powell. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In *Nonlinear Programming*, pages 53–72, Providence, 1976. Amer. Math. Soc. SIAM-AMS Proc., Vol. IX.

[RF00] A.I. Rauf and M. Fukushima. Globally convergent BFGS method for nonsmooth convex optimization. *Journal of Optimization Theory and Applications*, 104:539–558, 2000.

[Roy63] H.L. Royden. *Real Analysis.* Macmillan, New York, 1963.

[RW98] R.T. Rockafellar and R.J.B. Wets. *Variational Analysis.* Springer, New York, 1998.

[Sha05] A. Shapiro. Comments on "generalized derivatives and nonsmooth optimization, a finite dimensional tour" by j. dutta. *Sociedad de Estadustica e Investigacion Operativa, TOP*, 13:302–306, 2005.

[Sho85] N.Z. Shor. *Minimization Methods for Non-differentiable Functions.* Springer-Verlag, New York, 1985.

[VL01] J. Vlček and L. Lukšan. Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. *Journal of Optimization Theory and Applications*, 111:407–430, 2001.

[Wol75] P. Wolfe. A method of conjugate subgradients for minimizing nondifferentiable functions. *Math. Programming Stud.*, 3:145–173, 1975. In: Nondifferentiable Optimization, M.L. Balinski and P. Wolfe, eds.

[YVGS08] J. Yu, S.V.N. Vishwanathan, S. Günther, and N. Schraudolph. A quasi-Newton approach to non-smooth convex optimization. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[ZBN97] C. Zhu, R.H. Byrd, and J. Nocedal. Algorithm 778, L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23:550–560, 1997.

[ZZA$^+$00] S. Zhang, X. Zou, J. Ahlquist, I. M. Navon, and J. G. Sela. Use of differentiable and nondifferentiable optimization algorithms for variational data assimilation with discontinuous cost functions. *Monthly Weather Review*, 128:4031–4044, 2000.