Behavior of the Limited-Memory BFGS Method on Nonsmooth Optimization Problems in Theory and Practice

by

Azam Asl

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May 2020

Professor Michael L. Overton

Dedication

To my family.

Acknowledgements

I would like to express my sincere gratitude to my advisor Michael L. Overton for providing invaluable guidance throughout my graduate study. In research I have learned meticulous scrutiny and scientific approach from Michael. In writing and in particular, in composing the current dissertation, I have gained so much from his scholarly advice and endless support. Aside from work, and above and beyond anything, Michael is one of the kindest people I have ever met.

I wish to extend my gratitude to Frank E. Curtis with whom I had a chance to work and benefited much from his mathematical insight and knowledge. I also would like to extend my thanks to Margaret H. Wright for arranging funding of my studies for some parts.

My deep and genuine appreciation goes to Ernest Davis, my master advisor, whom I have known for a long time and has always been a source of reliance for me.

I am extremely thankful to my friends Alexander Golovnev, Sandro Coretti and many other friends, who made my time at Courant fun. I also thank Rosemary Amico and other staff in our department endlessly for their help throughout.

Abstract

The limited memory BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, abbreviated L-BFGS, is widely used for large-scale unconstrained optimization, but its behavior on nonsmooth problems has received little attention. In this thesis we give the first convergence analysis of the L-BFGS method applied to nonsmooth functions. We focus on the simplest version of the method, sometimes known as memoryless BFGS, which uses just one update. L-BFGS can be used with or without "scaling"; the use of scaling is normally recommended. We consider a simple class of convex piecewise linear nonsmooth functions that are unbounded below. On this class of problems, we show that memoryless BFGS with scaling, using any Armijo-Wolfe line search and initialized at any point where the objective f is differentiable, generates iterates that converge to a non-optimal point, if a certain condition relating the Lipschitz constant of f to the line search Armijo parameter holds. We also present an analysis of the ordinary gradient method with the same line search applied to the same class of functions, giving conditions under which it fails. However, scaled memoryless BFGS fails under a *weaker* condition relating the Lipschitz constant of the function to the line search Armijo parameter than that implying failure of the gradient method. Furthermore, in sharp contrast to the gradient method, if a specific standard Armijo-Wolfe bracketing line search is used, scaled memoryless BFGS fails if the Lipschitz constant is sufficiently large *regardless* of the Armijo parameter. Our experimental results suggest that our analysis is tight on this class of functions, and that similar results likely hold for L-BFGS with any fixed number of updates. In contrast, the "full" BFGS method is remarkably effective for minimizing nonsmooth functions, but it is not a practical approach when the number of variables is large.

We also conduct extensive experiments applying L-BFGS, both scaled and unscaled, with various choices for the number of updates, on other convex nonsmooth functions, ranging from artificially devised, highly ill-conditioned nonsmooth problems to eigenvalue optimization problems that are equivalent to semidefinite programming problems arising from applications. We also apply L-BFGS to smoothed versions of these problems. We find that although L-BFGS is usually a reliable method for minimizing ill-conditioned smooth problems, when the condition number is so large that the function is effectively nonsmooth, L-BFGS consistently fails. This behavior is in sharp contrast to the behavior of full BFGS, which is consistently reliable for nonsmooth optimization problems. We arrive at the conclusion that, for large-scale nonsmooth optimization problems for which BFGS and other methods are not practical, it is far preferable to apply L-BFGS to a *smoothed* variant of a nonsmooth problem than to apply it directly to the nonsmooth problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

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

The gradient method dates back to Cauchy [Cauchy, 1847]. Armijo [Armijo, 1966] was the first to establish convergence to stationary points of smooth functions using an inexact line search with a simple "sufficient decrease" condition. Wolfe [Wolfe, 1969], discussing line search methods for more general classes of methods, introduced a "directional derivative increase" condition among several others. The Armijo condition ensures that the line search step is not too large while the Wolfe condition ensures that it is not too small. Powell [Powell, 1976b] seems

to have been the first to point out that combining the two conditions leads to a convenient bracketing line search, noting also in another paper [Powell, 1976a] that use of the Wolfe condition ensures that, for quasi-Newton methods, the updated Hessian approximation is positive definite. Hiriart-Urruty and Lemarechal [Hiriart-Urruty & Lemaréchal, 1993, Vol 1, Ch. 11.3] give an excellent discussion of all these issues, although they reference neither [Armijo, 1966] nor [Powell, 1976b] and [Powell, 1976a]. They also comment (p. 402) on a surprising error in [Cauchy, 1847].

Suppose that f, the function to be minimized, is a nonsmooth convex function. An example of [Wolfe, 1975] shows that the ordinary gradient method with an exact line search may converge to a non-optimal point, without encountering any points where f is nonsmooth except in the limit. This example is stable under perturbation of the starting point, but it does not apply when the line search is inexact. Another example given in [Hiriart-Urruty & Lemaréchal, 1993, vol. 1, p. 363] applies to a subgradient method in which the search direction is defined by the steepest descent direction, i.e., the negative of the element of the subdifferential with smallest norm, again showing that use of an exact line search results in convergence to a non-optimal point. This example is also stable under perturbation of the initial point, and, unlike Wolfe's example, it also applies when an inexact line search is used. However, it is more complicated than is needed for the results we give below because it was specifically designed to defeat the steepest-descent subgradient method with an exact line search. Another example of convergence to a non-optimal point of a convex max function using a specific subgradient method with an exact line search goes back to [Dem'janov & Malozemov, 1971]; see [Fletcher, 1987, p. 385]. More generally, in a "black-box" subgradient method, the search direction is the negative of any subgradient returned by an "oracle",

which may not be a descent direction if the function is not differentiable at the point, although this is unlikely if the current point was not generated by an exact line search since convex functions are differentiable almost everywhere. The key advantage of the subgradient method is that, as long as f is convex and bounded below, convergence to its minimal value can be guaranteed even if f is nonsmooth by predefining a sequence of steplengths to be used, but the disadvantage is that convergence is usually slow. Nesterov [Nesterov, 2005] improved the complexity of such methods using a smoothing technique, but to apply it one needs some knowledge of the structure of the objective function.

The counterexamples mentioned above motivated the introduction of bundle methods by [Lemaréchal, 1975] and [Wolfe, 1975] for nonsmooth convex functions and, for nonsmooth, nonconvex problems, the bundle methods of [Kiwiel, 1985] and the gradient sampling algorithms of [Burke et al., 2005] and [Kiwiel, 2007]. These algorithms all have fairly strong convergence properties, to a nonsmooth (Clarke) stationary value when these exist in the nonconvex case (for gradient sampling, with probability one), but when the number of variables is large the cost per iteration is much higher than the cost of a gradient step. See the recent survey paper [Burke et al., 2020] for more details. The "full" BFGS method is a very effective alternative choice for nonsmooth optimization [Lewis & Overton, 2013], and its $O(n^2)$ cost per iteration (for the matrix-vector products that it requires) is generally much less than the cost of the bundle or gradient sampling methods, but its convergence results for nonsmooth functions are limited to very special cases. The limited memory variant of BFGS [Liu & Nocedal, 1989] costs only $O(n)$ operations per iteration, like the gradient method, but its behavior on nonsmooth problems is less predictable.

In Chapter 2 we analyze the ordinary gradient method with an inexact line search applied to a simple nonsmooth convex function. We require points accepted by the line search to satisfy both Armijo and Wolfe conditions for two reasons. The first is that we carry out a related analysis for the limited memory BFGS method in Chapter 3 for which the Wolfe condition is essential. The second is that the inclusion of the Wolfe condition is potentially useful in the nonsmooth case, where the norm of the gradient gives no useful information such as an estimate of the distance to a minimizer. For example, consider the absolute value function in one variable initialized at x_0 with x_0 large. A unit step gradient method with only an Armijo condition will require $O(x_0)$ iterations just to change the sign of x, while an Armijo-Wolfe line search with extrapolation defined by doubling requires only one line search with $O(\log_2(x_0))$ extrapolations to change the sign of x. Obviously, the so-called strong Wolfe condition recommended in many books for smooth optimization, which requires a reduction in the absolute value of the directional derivative, is a disastrous choice when f is nonsmooth. We mention here that in a 2017 paper on the analysis of the gradient method with fixed step sizes [Taylor et al., 2017], Taylor et al. remark that "we believe it would be interesting to analyze [gradient] algorithms involving line-search, such as backtracking or Armijo-Wolfe procedures."

The limited memory BFGS (L-BFGS) method is widely used for large-scale unconstrained optimization, but its behavior on nonsmooth problems has received little attention. We give the first analysis of an instance of the method, sometimes known as memoryless BFGS with scaling, on a specific class of nonsmooth convex problems, showing that under given conditions the method generates iterates whose function values are bounded below, although the function itself is unbounded below.

The "full" BFGS method [Nocedal & Wright, 2006, Sec. 6.1], independently derived by Broyden, Fletcher, Goldfarb and Shanno in 1970, is remarkably effective for unconstrained optimization, but even when the minimization objective $f : \mathbb{R}^n \to \mathbb{R}$ is assumed to be twice continuously differentiable and convex, with bounded level sets, the analysis of the method is nontrivial. Powell [Powell, 1976a] gave the first convergence analysis for full BFGS using an Armijo-Wolfe line search for this class of functions, establishing convergence to the minimal function value. In the smooth, nonconvex case it is generally accepted that the method is very reliable for finding stationary points (usually local minimizers), although pathological counterexamples exist [Dai, 2002, Mascarenhas, 2004].

At first glance, it might appear that, since BFGS uses gradient differences to approximate information about the Hessian of f, the use of BFGS for nonsmooth optimization makes little sense: first, because at minimizers where f is not differentiable, neither the gradient nor the Hessian exists; and secondly, even at other points where f is twice differentiable, the Hessian might appear to be meaningless: for example, for a piecewise linear function such as studied in this thesis, the Hessian is zero everywhere that it is defined. However, the way to make sense of the applicability of BFGS to a nonsmooth function is to consider its approximation by a very ill-conditioned smooth function. For example, the function $f(x) = \|x\|_2$ can be arbitrarily well approximated by the smooth function $f(x) = \sqrt{\|x\|_2^2 + \epsilon^2}$, where $\epsilon > 0$. As $\epsilon \downarrow 0$, the approximation becomes arbitrarily good — but also arbitrarily ill-conditioned. For any *fixed* $\epsilon > 0$, the BFGS convergence theory applies. As $\epsilon \downarrow 0$, it is not at all clear what impact the property of good approximation via badly conditioned functions has on the convergence theory, which, of course, does not apply when $\epsilon = 0$. Nonetheless, even for $\epsilon = 0$, the method remains well defined,

as the gradient is defined everywhere except at the minimizer (the origin). In fact, it was established recently by Guo and Lewis [Guo & Lewis, 2018] that Powell's result for smooth functions mentioned above can be extended, in a nontrivial way, to show that the iterates generated by BFGS with an Armijo-Wolfe line search, when applied to $f(x) = \|x\|_2$, converge to the origin. Even the case $n = 1$, where f is the absolute value function, is surprisingly complex; it turns out that in this case the sequence of iterates is defined by a certain binary expansion of the starting point [Lewis & Overton, 2013]. However, in this simple example it is easy to see intuitively *why* BFGS works well. The line search ensures that the iterates oscillate back and forth across the origin, giving a gradient difference equal to 2 at every iteration. As the iterates converge to the origin, the result is that the "inverse Hessian approximation" generated by BFGS converges to zero, resulting in quasi-Newton steps that also converge to zero. An important consequence is that the line search never requires many function evaluations. In contrast, when gradient descent with the same line search is applied to the absolute value function, the iterates converge to the origin, but each line search requires a number of function evaluations that increases in a manner inversely proportional to $|x|$.

More generally, if f is locally Lipschitz, BFGS is still typically well defined, because such functions are differentiable almost everywhere by Rademacher's theorem [Clarke, 1990], and hence f is differentiable at a randomly generated point with probability one. Furthermore, substantial computational experience [Lewis & Overton, 2013] shows that when f is a locally Lipschitz nonsmooth function, the method is remarkably reliable for finding Clarke stationary points (again, typically local minimizers), and furthermore, this property extends in a certain sense to constrained problems [Curtis et al., 2017]. Indeed, no non-pathological

counterexamples showing convergence to non-stationary values, meaning in particular examples where the starting point is not predetermined but generated randomly, are known. The superlinear convergence rate that holds generically for smooth functions is not attained in the nonsmooth case; instead, full BFGS is observed to converge linearly, in a sense described in [Lewis & Overton, 2013], on nonsmooth functions. Furthermore, in general one does not observe the inverse Hessian approximation converging to zero; instead, what seems to be typical is that *some* of its eigenvalues converge to zero, with corresponding eigenvectors identifying directions along which f is nonsmooth at the minimizer. See [Lewis & Overton, 2013, Sec. 6.2] for details.

The full BFGS method maintains and updates an approximation to the inverse (or a factorization) of the Hessian matrix $\nabla^2 f(x)$ at every iteration, defined by the known gradient difference information $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$ along $s_{k-1} = x_k - x_{k-1}$. The use of the Wolfe condition in the line search, requiring an increase in the directional derivative of f along the descent direction generated by BFGS, ensures that the updated inverse Hessian approximation is positive definite. The cost of full BFGS is $O(n^2)$ operations per iteration. While this was a great advance over the cost of Newton's method in the 1970s, already in the 1980s it was realized that the cost was too high for problems where n is large, and hence the limited memory version, L-BFGS, became popular, and is widely used today (see [Le et al., 2011, Lin et al., 2016], for example).

The standard version of L-BFGS, is discussed in detail in [Nocedal & Wright, 2006, Sec. 7.2]. For the earlier development of the L-BFGS method see [Liu & Nocedal, 1989] and the references therein. Let $m \ll n$ be given. Instead of maintaining an approximation to the inverse Hessian, at the kth iteration a proxy for this matrix is

implicitly defined by application of the most recent m BFGS updates (which are defined by saving y_j and s_j from the past m iterations) to a given sparse matrix H_k^0. One possible choice for H_k^0 is the identity matrix I, but a popular choice is to instead use *scaling*, defining

$$H_k^0 = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} I, \tag{1.0.1}$$

where the scaler multiplying the identity matrix is sometimes known as a Barzilai-Borwein approximation [Barzilai & Borwein, 1988]. Analysis of L-BFGS is more straightforward than analysis of full BFGS in the case that f is smooth and strongly convex, and is given in [Liu & Nocedal, 1989, Theorem 7.1], where linear convergence to minimizers is established, regardless of whether scaling is used or not. Furthermore, it is stated in [Liu & Nocedal, 1989] that scaling greatly accelerates L-BFGS, and this seems to be the currently accepted wisdom. However, as we show in Chapter 3, it is exactly the choice of scaling that may result in failure of L-BFGS on a specific class of nonsmooth functions. This situation is in sharp contrast to our experience with full BFGS on nonsmooth functions, where the same algorithm that is normally used for smooth functions works well also on nonsmooth functions.

We consider the class of convex functions

$$f(x) = a|x^{(1)}| + \sum_{i=2}^{n} x^{(i)}, \tag{1.0.2}$$

where $a \geqslant \sqrt{n-1}$. Note that although f, as is shown in Figure 1.1, is unbounded below, it is bounded below along the line defined by the negative gradient direction from any point x with $x^{(1)} \neq 0$. The specific choice of objective function f offers two advantages: one is its simplicity, but another is that there is little difficulty distinguishing in practice whether a method "succeeds" or "fails" from a given

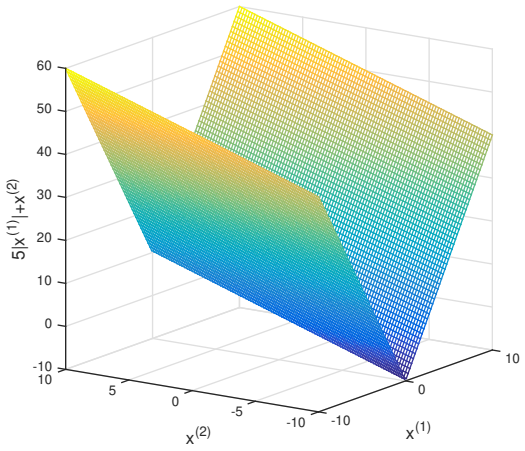


Figure 1.1: Mesh plot of function f given in (1.0.2), with $a = 5$ and $n = 2$. The function f is unbounded below.

starting point: success is associated with a sequence of function values that is unbounded below, while convergence of the sequence to a finite value implies failure.

In Chapter 2 we analyze the gradient method with *any* Armijo-Wolfe line search applied to (1.0.2). We show that if a satisfies a lower bound that depends only on the Armijo parameter, then the iterates generated by the gradient method with steps satisfying Armijo and Wolfe conditions converge to a point $\bar{x}$ with $\bar{x}^{(1)} = 0$, regardless of the starting point, although f is unbounded below. The function f defined in (1.0.2) was also used by [Lewis & Overton, 2013, p. 136] with $n = 2$ and $a = 2$ to illustrate failure of the gradient method with a specific line search, but the observations made there are not stable with respect to small changes in the initial point. The results of Chapter 2 have recently been published in [Asl & Overton, 2020b].

In Chapter 3 we analyze scaled L-BFGS with $m = 1$, i.e., with just one update — a method sometimes known as *memoryless* BFGS [Nocedal & Wright, 2006, p. 180] — applied to the function (1.0.2), and identify conditions under which

the method converges to non-optimal points. In contrast, it is known that when *full* BFGS is applied to the same function, eventually the method generates a search direction on which f is unbounded below [Xie & Waechter, 2017]; see also [Lewis & Zhang, 2015]. The results of Chapter 3 have recently been published in [Asl & Overton, 2020a].

In Chapter 4 we report on extensive experiments applying L-BFGS, both scaled and unscaled, with various choices for the number of updates, on many other classes of convex nonsmooth functions, ranging from artificially devised, highly ill-conditioned nonsmooth problems to eigenvalue optimization problems that are equivalent to semidefinite programming problems arising from applications. We also apply L-BFGS to smoothed versions of these problems. We find that although L-BFGS is usually a reliable method for minimizing ill-conditioned smooth problems, when the condition number is so large that the function is effectively nonsmooth, L-BFGS consistently fails. This behavior is in sharp contrast to the behavior of full BFGS, which is consistently reliable for nonsmooth optimization problems. We arrive at the conclusion that, for large-scale nonsmooth optimization problems for which BFGS and other methods are not practical, it is far preferable to apply L-BFGS to a *smoothed* variant of a nonsmooth problem than to apply it directly to the nonsmooth problem.

1.1 Computer Resources Used

Throughout this thesis all computations are performed in MATLAB. Most experiments are implemented in MATLAB (R2019a) running on a macOS 10.14.6 laptop computer with an Intel Core i7 processor with 2.5 GHz speed, 6MB of cache and

16GB of RAM. The larger experiments were run on the NYU HPC cluster Prince which runs Linux CentOS 7.4 operating system on its nodes. The experiments in §4.1.3 were run on partition c28 with Intel(R) IvyBridge @ 3.00GHz CPU type, on a single node with 2 CPUs. The experiments in §4.2.5 were run on partition c42 with Intel(R) Skylake @ 2.40GHz CPU type, on a single node with 10 CPUs. All BFGS/L-BFGS methods are taken from the implementation available in HANSO[1].

1.2 Funding Acknowledgment

Funding for my PhD studies was provided primarily by the Graduate School of Arts and Science, NYU, and, for several semesters, by a grant from the Simons Foundation (417314, MHW). The National Science Foundation also provided some summer support under grant DMS-1620083.

[1] www.cs.nyu.edu/overton/software/hanso/

Chapter 2

Analysis of the Gradient Method Applied to a Class of Nonsmooth Optimization Problems

In this chapter, we analyze the behavior of the gradient method applied to the simple nonsmooth convex function (1.0.2). The chapter is organized as follows. In §2.1 we establish the main theoretical results, without assuming the use of any specific line search beyond satisfaction of the Armijo and Wolfe conditions. In §2.2, we extend these results assuming the use of a bracketing line search that is a specific instance of the ones outlined by [Powell, 1976b] and [Hiriart-Urruty & Lemaréchal, 1993]. In §2.3, we give experimental results, showing that our theoretical results are reasonably tight. We discuss connections with the convergence theory for subgradient methods in §2.4. We make some concluding remarks in §2.5.

2.1 Convergence Results Independent of a Specific Line Search

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

$$A(t): \quad f(x_k + td_k) \leqslant f(x_k) + c_1 t \nabla f(x_k)^T d_k \tag{2.1.1}$$

and that it satisfies the Wolfe condition if [1]

$$W(t): \quad f \text{ is differentiable at } x_k + td_k \text{ with } \nabla f(x_k + td_k)^T d_k \geqslant c_2 \nabla f(x_k)^T d_k. \tag{2.1.2}$$

The condition $0 < c_1 < c_2 < 1$ ensures that points t satisfying $A(t)$ and $W(t)$ exist, as is well known in the convex case and the smooth case; for more general f, see [Lewis & Overton, 2013]. The results of this section are independent of any choice of line search to generate such points. Note that as long as f is differentiable at the initial iterate, defining subsequent iterates by $x_{k+1} = x_k + t_k d_k$, where $W(t)$ holds for $t = t_k$, ensures that f is differentiable at all x_k.

[1] There is a subtle distinction between the Wolfe condition given here and that given in [Lewis & Overton, 2013], since here the Wolfe condition is understood to fail if the gradient of f does not exist at $x_k + td_k$, while in [Lewis & Overton, 2013] it is understood to fail if the function of one variable $s \mapsto f(x_k + sd_k)$ is not differentiable at $s = t$. For the example analyzed here, these conditions are equivalent.

We now restrict our attention to f defined by (1.0.2), with

$$d_k = -\nabla f(x_k) = -\begin{bmatrix} \operatorname{sgn}(x_k^{(1)})a \\ \mathbb{1} \end{bmatrix}, \tag{2.1.3}$$

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

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

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

$$x_{k+1}^{(1)} = x_k^{(1)} - \operatorname{sgn}(x_k^{(1)})at_k \quad \text{and} \quad x_{k+1}^{(i)} = x_k^{(i)} - t_k \quad \text{for} \quad i = 2, \ldots, n. \tag{2.1.4}$$

We have

$$\nabla f(x_k)^T d_k = -(a^2 + n - 1) \tag{2.1.5}$$

and

$$\nabla f(x_k + t_k d_k)^T d_k = -(a^2 \operatorname{sgn}(x_{k+1}^{(1)})\operatorname{sgn}(x_k^{(1)}) + n - 1). \tag{2.1.6}$$

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

Assumption 1 *Let f be defined by (1.0.2) with $a \geqslant \sqrt{n-1}$ and define $x_{k+1} = x_k + t_k d_k$, with $d_k = -\nabla f(x_k)$, for some steplength t_k, $k = 1, 2, 3, \ldots$, where x_0 is arbitrary provided that $x_0^{(1)} \neq 0$.*

Lemma 2.1.1 *The Armijo condition $A(t_k)$ (i.e., (2.1.1) with $t = t_k$), is equivalent*

to

$$c_1 t_k(a^2 + n - 1) \leqslant f(x_k) - f(x_{k+1}) \tag{2.1.7}$$

and the Wolfe condition $W(t_k)$ *(i.e., (2.1.2) with* $t = t_k$*) is equivalent to each of the following three conditions:*

$$\operatorname{sgn}(x_{k+1}^{(1)}) = -\operatorname{sgn}(x_k^{(1)}), \tag{2.1.8}$$

$$t_k > \frac{|x_k^{(1)}|}{a} \tag{2.1.9}$$

and

$$at_k = |x_{k+1}^{(1)} - x_k^{(1)}| = |x_k^{(1)}| + |x_{k+1}^{(1)}|. \tag{2.1.10}$$

Proof: *These all follow easily from (2.1.4), (2.1.5) and (2.1.6), using* $c_2 < 1$ *and* $a \geqslant \sqrt{n-1}$. ∎

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

Theorem 2.1.2 *Suppose* t_k *satisfies* $A(t_k)$ *and* $W(t_k)$ *for* $k = 1, 2, 3, \ldots, N$ *and define* $S_N = \sum_{k=0}^{N-1} t_k$. *Then*

$$c_1(a^2 + n - 1)S_N \leqslant f(x_0) - f(x_N) \leqslant (n-1)S_N + a|x_0^{(1)}|, \tag{2.1.11}$$

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

[2]The same oscillatory behavior occurs if we replace the Wolfe condition by the Goldstein condition $f(x_k + td_k) \geqslant f(x_k) + c_2 t \nabla f(x_k)^T d_k$.

Proof: Summing up (2.1.7) from $k = 0$ to $k = N - 1$ we have

$$c_1(a^2 + n - 1)S_N \leqslant f(x_0) - f(x_N). \tag{2.1.12}$$

Using (2.1.4) we have

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

so

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

using (1.0.2). Combining this with (2.1.12) and dropping the term $a|x_N^{(1)}|$ we obtain (2.1.11), so S_N is bounded above if and only if $f(x_N)$ is bounded below. Now suppose that $f(x_N)$ is bounded below and hence S_N is bounded above, implying that $t_N \to 0$, and therefore, from (2.1.10), that $x_N^{(1)} \to 0$. Since $f(x_N) = a|x_N^{(1)}| + \sum_{i=2}^{n-1} x_N^{(i)}$ is bounded below as $N \to \infty$, and since, from (2.1.4), for $i = 2, \ldots, n$, each $x_N^{(i)}$ is decreasing as N increases, we must have that each $x_N^{(i)}$ converges to a limit $\bar{x}^{(i)}$. On the other hand, if x_N converges to a point $(0, \bar{x}^{(2)}, \ldots, \bar{x}^{(n)})$ then $f(x_N)$ is bounded below by $\sum_{i=2}^{n-1} \bar{x}^{(i)}$. ∎

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

We next observe that, because of the bounds (2.1.11), it is not possible that $S_N \to \infty$ if

$$a > \sqrt{(n-1)\left(\frac{1}{c_1} - 1\right)}$$

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

It will be convenient to define

$$\tau = c_1 + \frac{(n-1)(c_1-1)}{a^2}. \tag{2.1.13}$$

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

Corollary 2.1.3 *Suppose $A(t_k)$ and $W(t_k)$ hold for all k. If $\tau > 0$ then $f(x_k)$ is bounded below as $k \to \infty$.*

Proof: This is now immediate from (2.1.11) and the definition of τ. ∎

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

At this point it is natural to ask whether $\tau \leqslant 0$ implies that $f(x_k) \to -\infty$. We will see in the next section (in Corollary 2.2.4, for $\tau = 0$) that the answer is no. However, we can show that there is a specific choice of t_k satisfying $A(t_k)$ and $W(t_k)$ for which $\tau \leqslant 0$ implies $f(x_k) \to -\infty$. We start with a lemma.

Lemma 2.1.4 *Suppose $W(t_k)$ holds. Then $A(t_k)$ holds if and only if*

$$(1+\tau)\frac{at_k}{2} \leqslant |x_k^{(1)}|. \tag{2.1.14}$$

Proof: Suppose $x_k^{(1)} > 0$. Since $W(t_k)$ holds, using (2.1.8), we can rewrite the

Armijo condition (2.1.7) as

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

giving (2.1.14). A similar argument applies when $x_k^{(1)} < 0$. ■

Theorem 2.1.5 *Let*

$$t_k = \frac{2|x_k^{(1)}|}{(\tau+1)a}. \tag{2.1.15}$$

Then

(1) $A(t_k)$ and $W(t_k)$ both hold.

(2) if $\tau \leqslant 0$, then $f(x_k)$ is unbounded below as $k \to \infty$.

Proof: The first statement follows immediately from (2.1.9) (since $|\tau| < 1$) and Lemma 2.1.4. Furthermore, (2.1.10) allows us to write (2.1.14) equivalently as

$$(1+\tau)|x_{k+1}^{(1)}| \leqslant (1-\tau)|x_k^{(1)}|. \tag{2.1.16}$$

Since t_k is the maximum steplength satisfying (2.1.14), it follows that (2.1.16) holds with equality, so $|x_{k+1}^{(1)}| = C|x_k^{(1)}|$, where

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

and hence

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

Then, we can rewrite (2.1.15) as

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

When $-1 < \tau \leqslant 0$, we have $C \geqslant 1$, so $S_N = \sum_{k=0}^{N-1} t_k \to \infty$ as $N \to \infty$ and hence, by Theorem 2.1.2, $f(x_N) \to -\infty$. ∎

2.2 Additional Results Depending on a Specific Choice of Armijo-Wolfe Line Search

In this section we continue to assume that f and d_k are defined by (1.0.2) and (2.1.3) respectively, with $a \geqslant \sqrt{n-1}$, and that $A(t)$ and $W(t)$ are defined as earlier. However, unlike in the previous section, we now assume that t_k is generated by the Armijo-Wolfe bracketing line search given in Algorithm 1, which is taken from [Lewis & Overton, 2013, p. 147] and is a specific realization of the line searches described implicitly in [Powell, 1976b] and explicitly in [Hiriart-Urruty & Lemaréchal, 1993]. Since the line search function $s \mapsto f(x_k+sd_k)$ is locally Lipschitz and bounded below, it follows, as shown in [Lewis & Overton, 2013], that at any stage during the execution of Algorithm 1, the interval $[\alpha, \beta]$ must always contain a set of points t with nonzero measure satisfying $A(t)$ and $W(t)$, and furthermore, the line search must terminate at such a point. This defines the steplength t_k. A crucial aspect of Algorithm 1 is that, in the "while" loop, the

Armijo condition is tested first and the Wolfe condition is then tested only if the Armijo condition holds. We already know from Theorem 2.1.2 and Corollary 2.1.3

$\alpha \leftarrow 0$
$\beta \leftarrow +\infty$
$t \leftarrow 1$
while true **do**
  if $A(t)$ fails (see (2.1.1)) **then**
    $\beta \leftarrow t$
  else if $W(t)$ fails (see (2.1.2)) **then**
    $\alpha \leftarrow t$
  else
    stop and return t
  end if
  if $\beta < +\infty$ **then**
    $t \leftarrow (\alpha + \beta)/2$
  else
    $t \leftarrow 2\alpha$
  end if
end while

Algorithm 1: Armijo-Wolfe Bracketing Line Search

that, for *any* set of Armijo-Wolfe points, if $\tau > 0$, then $f(x_N)$ is bounded below. In this section we analyze the case $\tau \leqslant 0$, assuming that the steps t_k are generated by the Armijo-Wolfe bracketing line search. It simplifies the discussion to make a probabilistic analysis, assuming that $x_0 = (x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(n)})$ is generated randomly, say from the standard normal distribution. Clearly, all intermediate values t generated by Algorithm 1 are rational, and with probability one all corresponding points $x = (x_0^{(1)} - \mathrm{sgn}(x_0^{(1)})at,\ x_0^{(2)} - t, \ldots,\ x_0^{(n)} - t)$ where the Armijo and Wolfe conditions are tested during the first line search are irrational (this is obvious if a is rational but it also holds if a is irrational assuming that x_0 is generated independently of a). It follows that, with probability one, f is differentiable at these

points, which include the next iterate $x_1 = (x_1^{(1)},\ x_1^{(2)}, \ldots,\ x_1^{(n)})$. It is clear that, by induction, the points $x_k = (x_k^{(1)},\ x_k^{(2)}, \ldots,\ x_k^{(n)})$ are irrational with probability one for all k, and in particular, $x_k^{(1)}$ is nonzero for all k and hence f is differentiable at all points x_k.

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

Assumption 2 *Let f be defined by (1.0.2), with $a \geqslant \sqrt{n-1}$, and define $x_{k+1} = x_k + t_k d_k$, with $d_k = -\nabla f(x_k)$, and with t_k defined by Algorithm 1, $k = 1, 2, 3, \ldots$, where $x_k = (x_k^{(1)},\ x_k^{(2)}, \ldots,\ x_k^{(n)})$, and $x_0 = (x_0^{(1)},\ x_0^{(2)}, \ldots,\ x_0^{(n)})$ is randomly generated from the standard normal distribution. All statements in the theorems, lemmas and corollaries in this section are understood to hold with probability one.*

Lemma 2.2.1 *Suppose $\tau \leqslant 0$ and suppose $|x_k^{(1)}| > a$. Define*

$$r_k = \left\lceil \log_2 \frac{|x_k^{(1)}|}{a} \right\rceil \quad \text{so that} \quad a2^{r_k - 1} < |x_k^{(1)}| < a2^{r_k}. \tag{2.2.1}$$

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

Proof: Since $|x_k^{(1)}| > a$ any steplength $t \leqslant |x_k^{(1)}|/a$ satisfies $A(t)$ but fails $W(t)$. Starting with $t = 1$, the "while" loop in Algorithm 1 will carry out r_k doublings of t until $t > |x_k^{(1)}|/a$, i.e., $W(t)$ holds. Hence, in the beginning of stage $r_k + 1$, we have $\alpha = 2^{r_k - 1}$ (a lower bound on t_k), $t = 2^{r_k}$ and $\beta = +\infty$. At this point, t satisfies $W(t)$ and since $\tau \leqslant 0$, it also satisfies (2.1.14), i.e. $A(t)$. So $t_k = 2^{r_k}$. ∎

Theorem 2.2.2 *Suppose $\tau \leqslant 0$ and $|x_0^{(1)}| > a$. Then after $j \leqslant r_0$ iterations we have $|x_j^{(1)}| < a$, where r_0 is defined by (2.2.1), and furthermore, for all subsequent iterations, the condition $|x_k^{(1)}| < a$ continues to hold.*

Proof: For any k with $|x_k^{(1)}| > a$ we know from the previous lemma that $t_k = 2^{r_k}$ with $r_k > 0$. From (2.1.10) and (2.2.1) we get

$$|x_{k+1}^{(1)}| = at_k - |x_k^{(1)}| < a2^{r_k} - a2^{r_k-1} = a2^{r_k-1}. \tag{2.2.2}$$

See Figure 2.1 for an illustration with $n = 2$, with $x_k^{(1)} > 0$, so $-a2^{r_k-1} < x_{k+1}^{(1)} < 0$. Hence, either $|x_{k+1}^{(1)}| < a$, or $a < |x_{k+1}^{(1)}| < a2^{r_k-1}$, in which case from (2.2.1) and (2.2.2) we have

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

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

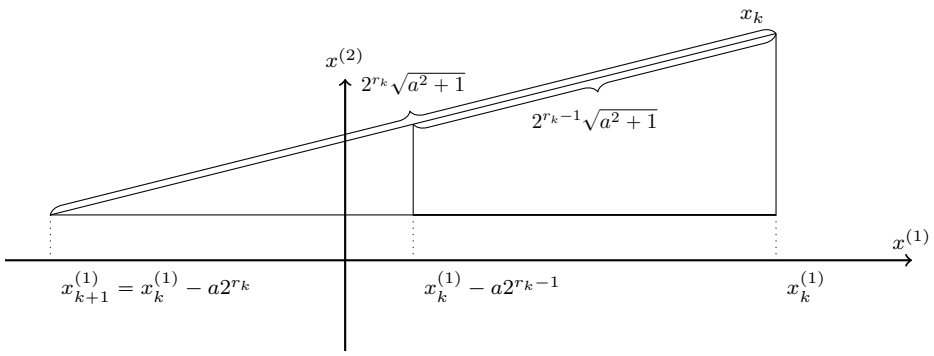


Figure 2.1: Doubling t in order to satisfy $W(t)$.

Theorem 2.2.2 shows that for any $\tau \leqslant 0$ and sufficiently large k using Algorithm 1 we always have $|x_k^{(1)}| < a$. In the reminder of this section we provide further details on the step t_k generated when $|x_k^{(1)}| < a$. In this case, the initial step $t = 1$ satisfies

$W(t)$ but not necessarily $A(t)$. So Algorithm 1 will repeatedly halve t, until it satisfies $A(t)$. See Figure 2.2 for an illustration.

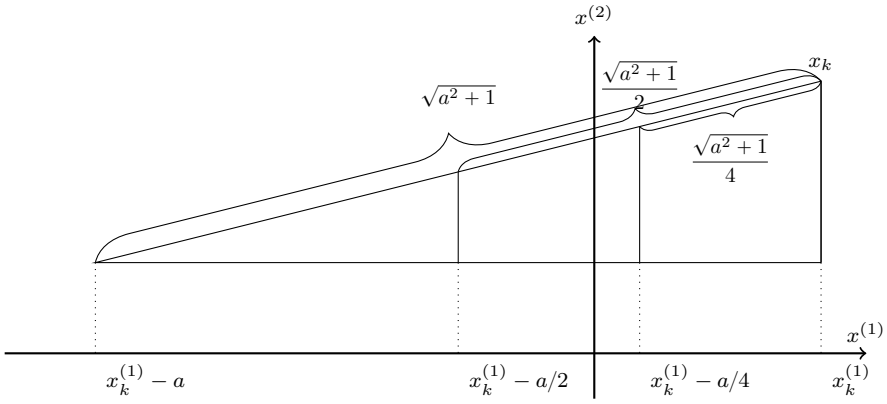


Figure 2.2: Halving t in order to satisfy $A(t)$.

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

$$p_k = \left\lceil \log_2 \frac{a}{|x_k^{(1)}|} \right\rceil \quad \text{so that} \quad \frac{a}{2^{p_k}} < |x_k^{(1)}| < \frac{a}{2^{p_k-1}}. \tag{2.2.3}$$

For example, in Figure 2.2, $p_k = 2$. So, $a/4 < |x_k^{(1)}| < a/2$. Hence $t = 1/2$ satisfies $W(t)$. In fact it also satisfies $A(t)$, because for $\tau = 0$, we have

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

which is exactly the Armijo condition (2.1.14). So, Algorithm 1 returns $t_k = 1/2$.

On the other hand if we had $\tau \leqslant -1/2$, $t = 1$ would have satisfied the Armijo condition (2.1.14) since

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

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

Theorem 2.2.3 *Suppose $\tau \leqslant 0$ and $|x_k^{(1)}| < a$. Then $t_k = \min(1, 1/2^{q_k - 1})$, where*

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

so

$$\frac{(1+\tau)a}{2^{q_k}} < |x_k^{(1)}| < \frac{(1+\tau)a}{2^{q_k - 1}}. \tag{2.2.4}$$

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

Proof: If $|x_k^{(1)}| > (1+\tau)a/2$, then $t = 1$ satisfies the Armijo condition (2.1.14) as well as the Wolfe condition, so t_k is set to 1. Otherwise, $q_k > 1$, so $1/2^{q_k - 1} < 1$ and Algorithm 1 repeatedly halves t until $A(t)$ holds. We now show that the first t that satisfies $A(t)$ is such that $|x_k^{(1)}| < at$, i.e., it satisfies $W(t)$ as well. Since $\tau \leqslant 0$, the second inequality in (2.2.4) proves that steplength $t = 1/2^{q_k - 1}$ satisfies $W(t)$. Moreover, the first inequality is the Armijo condition (2.1.14) with the same steplength. Furthermore, the second inequality in (2.2.4) also shows that $t' = 2t = 1/2^{q_k - 2}$ is too large to satisfy the Armijo condition (2.1.14). Hence $t = 1/2^{q_k - 1}$ is the first steplength satisfying both $A(t)$ and $W(t)$. So, Algorithm 1 returns $t_k = 1/2^{q_k - 1}$. ■

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

Corollary 2.2.4 *Suppose $\tau = 0$. Then x_k converges to a limit $\bar{x}$ with $\bar{x}^{(1)} = 0$.*

Proof: Assume that k is sufficiently large so that $|x_k^{(1)}| < a$. From (2.2.3) we have $a/2^{p_k} < |x_k^{(1)}|$. Using Theorem 2.2.3 we have $t_k = 1/2^{p_k - 1}$ and therefore

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

(see Figure 2.2 for an illustration). So $p_{k+1} \geqslant p_k + 1$. Using Theorem 2.2.3 again we conclude $t_{k+1} \leqslant 1/2^{p_k}$ and so $t_{k+1} \leqslant t_k/2$. The same argument holds for all subsequent iterates so $S_N = \sum_{k=0}^{N-1} t_k$ is bounded above as $N \to \infty$. The result therefore follows from Theorem 2.1.2.

■

Corollary 2.2.5 *If $\tau \leqslant -0.5$ then eventually $t_k = 1$ at every iteration, and $f(x_k) \to -\infty$.*

Proof: As we showed in Theorem 2.2.2, for sufficiently large k, $|x_k^{(1)}| < a$ and therefore $t = 1$ always satisfies the Wolfe condition, so $t_k \leqslant 1$. If $|x_k^{(1)}| > (1+\tau)a/2$, then $t = 1$ also satisfies the Armijo condition (2.1.14), so $t_k = 1$. If $|x_{k+1}^{(1)}| > (1+\tau)a/2$ as well, then $t_{k+1} = 1$ and hence $x_{k+2}^{(1)} = x_k^{(1)}$. It follows that $t_j = 1$ for all $j > k+1$. Hence, by Theorem 2.1.2, $f(x_k) \to -\infty$. Otherwise, suppose $|x_k^{(1)}| < (1+\tau)a/2$ (in case $|x_k^{(1)}| > (1+\tau)a/2$ and $|x_{k+1}^{(1)}| < (1+\tau)a/2$ just shift the index by one so that we have $|x_{k-1}^{(1)}| > (1+\tau)a/2$ and $|x_k^{(1)}| < (1+\tau)a/2$).

Since $|x_k^{(1)}| < (1+\tau)a/2$, from the definition of q_k in (2.2.4) we conclude that $2 \leqslant q_k$, i.e. $1/2^{q_k-1} \leqslant 1/2$, so from Theorem 2.2.3 we have $t_k = 1/2^{q_k-1} \leqslant 1/2$. Since $|x_k^{(1)}| < (1+\tau)a/2^{q_k-1}$ and $1 + \tau \leqslant 1/2$ we have

$$|x_k^{(1)}| < \frac{a}{2^{q_k}}. \tag{2.2.5}$$

So by (2.1.10)

$$|x_{k+1}^{(1)}| = at_k - |x_k^{(1)}| \geqslant \frac{a}{2^{q_k-1}} - \frac{a}{2^{q_k}} = \frac{a}{2^{q_k}} > \frac{(1+\tau)a}{2^{q_k-1}} \tag{2.2.6}$$

and using (2.2.4) again we conclude $q_{k+1} \leqslant q_k - 1$. So,

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

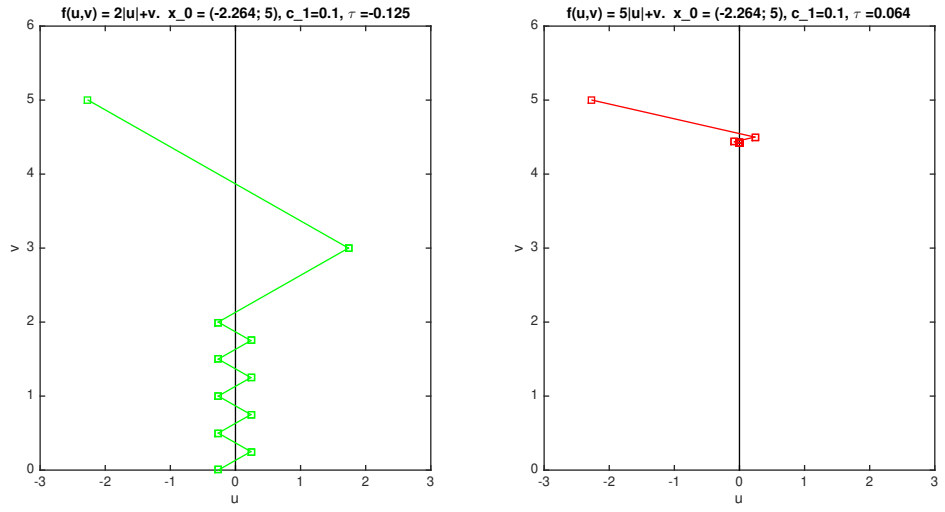
and therefore, applying this repeatedly, after a finite number of iterations, say at iteration $\bar{k}$, we must have $t_{\bar{k}} = 1$ for the first time. Furthermore, from (2.2.5) and (2.2.6) we have $|x_k^{(1)}| < |x_{k+1}^{(1)}|$, and applying this repeatedly as well we have $|x_{\bar{k}}^{(1)}| < |x_{\bar{k}+1}^{(1)}|$. From the Armijo condition (2.1.14) at iteration $\bar{k}$ we have $(1+\tau)a/2 \leqslant |x_{\bar{k}}^{(1)}|$ and therefore

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

Hence, $t = 1$ also satisfies the Armijo condition (2.1.14) at iteration $\bar{k}+1$. With $t_{\bar{k}} = 1$ and $t_{\bar{k}+1} = 1$, we conclude $x_{\bar{k}+2}^{(1)} = x_{\bar{k}}^{(1)}$. It follows that $t_j = 1$ for all $j > \bar{k}+1$. Hence $f(x_k) \to -\infty$ by Theorem 2.1.2. ∎

2.3 Experimental Results

In this section we again continue to assume that f and d_k are defined by (1.0.2) and (2.1.3) respectively. For simplicity we also assume that $n = 2$, writing $u = x^{(1)}$ and $v = x^{(2)}$ for convenience. Our experiments confirm the theoretical results presented in the previous sections and provide some additional insight. We know from Theorem 2.1.2 that when the gradient algorithm fails, i.e, x_k converges to a point $(0, \bar{v})$, the step t_k converges to zero. However, an implementation of Algorithm 1 in floating point arithmetic must terminate the "while" loop after it executes a maximum number of times. We used the MATLAB implementation in HANSO, which limits the number of bisections in the "while" loop to 30.



(a) **(b)**

Figure 2.3: Minimizing f with $n = 2$, $u = x^{(1)}$, $v = x^{(2)}$ and $c_1 = 0.1$. **Left**, with $a = 2$, so $\tau < 0$ and $f(u_k, v_k) \to -\infty$ (success). **Right**, with $a = 5$, so $\tau > 0$ and $(u_k, v_k) \to (0, \bar{v})$ (failure).

Figure 2.3 shows two examples of minimizing f with $a = 2$ and $a = 5$, with $c_1 = 0.1$ in both cases, and hence with $\tau < 0$ and $\tau > 0$, respectively. Starting from the same randomly generated point, we have $f(x_k) \to -\infty$ (success) when $\tau < 0$ and $x_k \to (0, \bar{v})$ (failure) when $\tau > 0$.

For various choices of a and c_1 we generated 5000 starting points $x_0 = (u_0, v_0)$, each drawn from the normal distribution with mean 0 and variance 1, and measured how frequently "failure" took place, meaning that the line search failed to find an Armijo-Wolfe point within 30 bisections. If failure did not take place within 50 iterations, i.e., with $k \leqslant 50$, we terminated the gradient method declaring success. Figure 2.4 shows the failure rates when (top) c_1 is fixed to 0.05 and a is varied and (bottom) when $a = \sqrt{2}$ and c_1 is varied. Both cases confirm that when $\tau > 0$ the method always fails, as predicted by Corollary 2.1.3, while when $\tau \leqslant -0.5$, failure does not occur, as shown in Corollary 2.2.5.

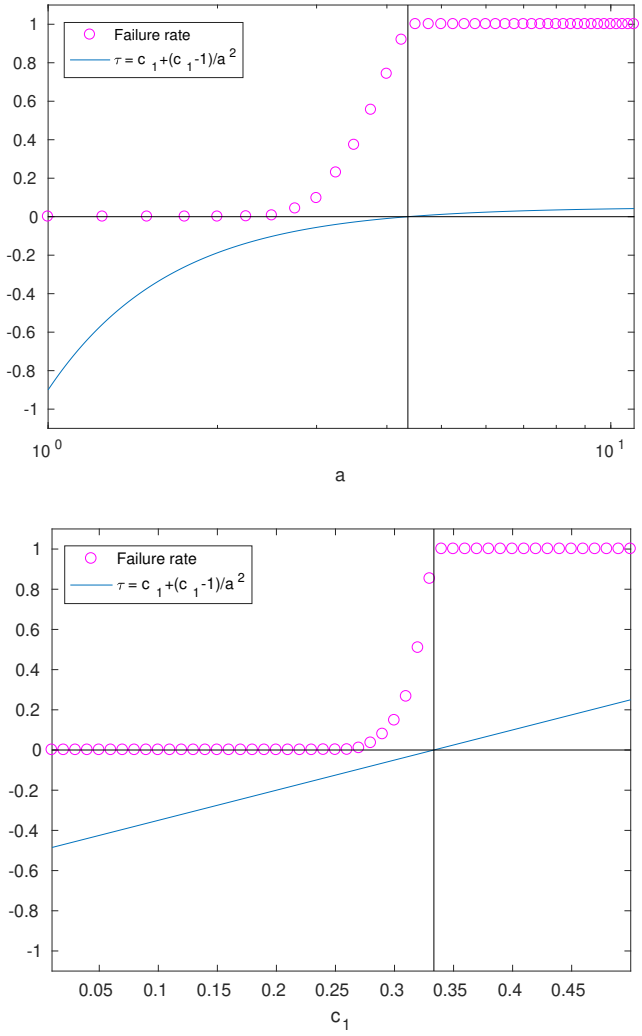


Figure 2.4: Failure rates (small circles) for f with $n = 2$ when (top) c_1 is fixed to 0.05 and a is varied and (bottom) a is fixed to $\sqrt{2}$ and c_1 is varied. The solid curves show the value of τ. Each experiment was repeated 5000 times.

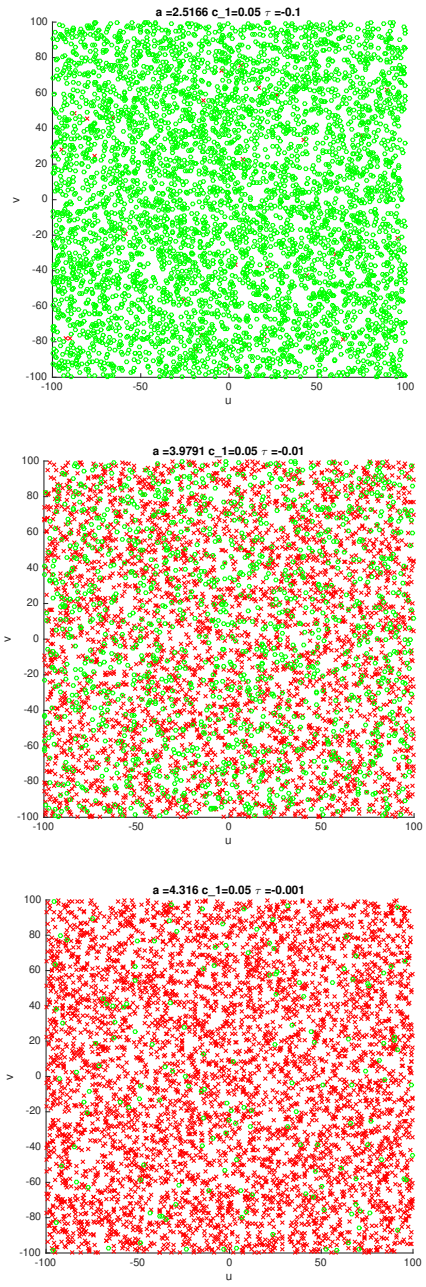


Figure 2.5: Mixed success and failure when $\tau = -0.1$ (top), $\tau = -0.01$ (middle) and $\tau = -0.001$ (bottom). Each plot shows 5000 points. The green circles show starting points for which the method succeeded, generating $x_k = (u_k, v_k) \in \mathbb{R}^2$ for which $f(x_k)$ is apparently unbounded below, while the red crosses show starting points for which the method failed, generating x_k converging to a point on the v-axis.

As Figure 2.4 shows, when $\tau < 0$ with $|\tau|$ small, the method may or may not fail, with failure more likely the closer τ is to zero. Further experiments for three specific values of τ, namely $-0.1, -0.01$ and -0.001, using a fixed value of $c_1 = 0.05$ and a defined by $a = \sqrt{(1 - c_1)/(c_1 - \tau)}$, confirmed that failure is more likely the closer that τ gets to zero and also showed that the set of initial points from which failure takes place is complex; see Figure 2.5. The initial points were drawn uniformly from the box $(-100, 100) \times (-100, 100)$.

We know from Corollary 2.2.4 that, for $\tau = 0$, with probability one $t_k \to 0$, so even if high precision were being used, for sufficiently large k an implementation in floating point must fail. It may well be the case that failures for $\tau < 0$ occur only because of the limited precision being used, and that with sufficiently high precision, these failures would be eliminated. This suggestion is supported by experiments done reducing the maximum number of bisections to 15, for which the number of failures for $\tau < 0$ increased significantly, and increasing it to 50, for which the number of failures decreased significantly.

2.4 Relationship with Convergence Results for Subgradient Methods

Let h be any convex function. The subgradient method [Shor, 1985, Bertsekas, 1999] applied to h is a generalization of the gradient method, where h is not assumed to be differentiable at the iterates $\{x_k\}$ and hence, instead of setting $-d_k = \nabla h(x_k)$, one defines $-d_k$ to be any element of the subdifferential set

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

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

We cannot apply the convergence theory of the subgradient method directly to our function f defined in (1.0.2), because f is not bounded below. However, we can argue as follows. Suppose that $\tau > 0$, so that we know (by Corollary 2.1.3) that for all x_0 with $x_0^{(1)} \neq 0$, the iterates x_k generated by the gradient method with Armijo-Wolfe steplengths applied to f converge to a point $\bar{x}$ with $\bar{x}^{(1)} = 0$. Fix any initial point x_0 with $x_0^{(1)} \neq 0$, and let $M = f(\bar{x})$, where $\bar{x}$ is the resulting limit point (to make this well defined, we can assume that the Armijo-Wolfe bracketing line search of Section 2.2 is in use). Now define

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

Clearly, the iterates generated by the gradient method with Armijo-Wolfe steplengths initiated at x_0 are identical for f and $\tilde{f}$, with f (equivalently, $\tilde{f}$) differentiable at all iterates $\{x_k\}$, and with $f(x_k) = \tilde{f}(x_k) \to M$. Furthermore, the theory of subgradient methods applies to $\tilde{f}$. One well-known result states that provided the steplengths $\{t_k\}$ are square-summable (that is, $\sum_{k=0}^{\infty} t_k^2 < \infty$, and hence the steps are "not too long"), but not summable (that is, $\sum_{k=0}^{\infty} t_k = \infty$, and hence the steps

are "not too short"), then convergence of $\tilde{f}(x_k)$ to the optimal value $M-1$ must take place [Nedić & Bertsekas, 2001]. Since this does *not* occur, we conclude that the Armijo-Wolfe steplengths $\{t_k\}$ do *not* satisfy these conditions. Indeed, the "not summable" condition is exactly the condition $S_N \to \infty$, where $S_N = \sum_{k=0}^{N-1} t_k$, and Theorem 2.1.2 established that the converse, that S_N is bounded above, is equivalent to the function values $f(x_k)$ being bounded below. This, then, is consistent with the convergence theory for the subgradient method, which says that the steps must not be "too short"; in the context of an Armijo-Wolfe line search, when c_1 is not sufficiently small, and hence $\tau > 0$, the Armijo condition is too restrictive: it is causing the $\{t_k\}$ to be "too short" and hence summable.

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

2.5 Concluding Remarks

Should we conclude from the results of this chapter that, if the gradient method with an Armijo-Wolfe line search is applied to a nonsmooth function, the Armijo parameter c_1 should be chosen to be small? Results for a very ill-conditioned convex nonsmooth function $\hat{f}$ devised by Nesterov [Nesterov, 2016] suggest that

the answer is yes. The function is defined by

$$\hat{f}(x) = \max\{|x_1|,\ |x_i - 2x_{i-1}|,\quad i = 2, ..., n\}. \tag{2.5.1}$$

Let $\hat{x}_1 = 1, \hat{x}_i = 2\hat{x}_{i-1} + 1, i = 2, ..., n$. Then $\hat{f}(\hat{x}) = 1 = \hat{f}(\mathbb{1})$ although $\|\hat{x}\|_\infty \approx 2^n$ and $\|\mathbb{1}\|_\infty = 1$, so the level sets of $\hat{f}$ are very ill conditioned. The minimizer is $x = 0$ with $\hat{f}(x) = 0$. Figure 2.6 shows function values computed by applying five different methods to minimize $\hat{f}$ with $n = 100$. The five methods are: the subgradient method with $t_k = 1/k$, a square-summable but not summable sequence that guarantees convergence; the gradient method using the Armijo-Wolfe bracketing line search of Section 2.2; the limited memory BFGS method [Nocedal & Wright, 2006] with 5 and 10 updates respectively (using "scaling"); and the full BFGS method [Nocedal & Wright, 2006, Lewis & Overton, 2013]; the BFGS variants also use the same Armijo-Wolfe line search.[3] The top and bottom plots in Figure 2.6 show the results when the Armijo parameter c_1 is set to 0.1 and to 10^{-6} respectively. The Wolfe parameter was set to 0.5 in both cases. These values were chosen to satisfy the usual requirement that $0 < c_1 < c_2 < 1$, while still ensuring that c_1 is not so tiny that it is effectively zero in floating point arithmetic. All function values generated by the methods are shown, including those evaluated in the line search. The same initial point, generated randomly, was used for all methods; the results using other initial points were similar.

For this particular example, we see that, in terms of reduction of the function

[3] In our implementation, we made no attempt to determine whether $\hat{f}$ is differentiable at a given point or not. This is essentially impossible in floating point arithmetic, but as noted earlier, the gradient is defined at randomly generated points with probability one; there is no reason to suppose that any of the methods tested will generate points where $\hat{f}$ is not differentiable, except in the limit, and hence the "subgradient" method actually reduces to the gradient method with $t_k = 1/k$. See [Lewis & Overton, 2013] for further discussion.

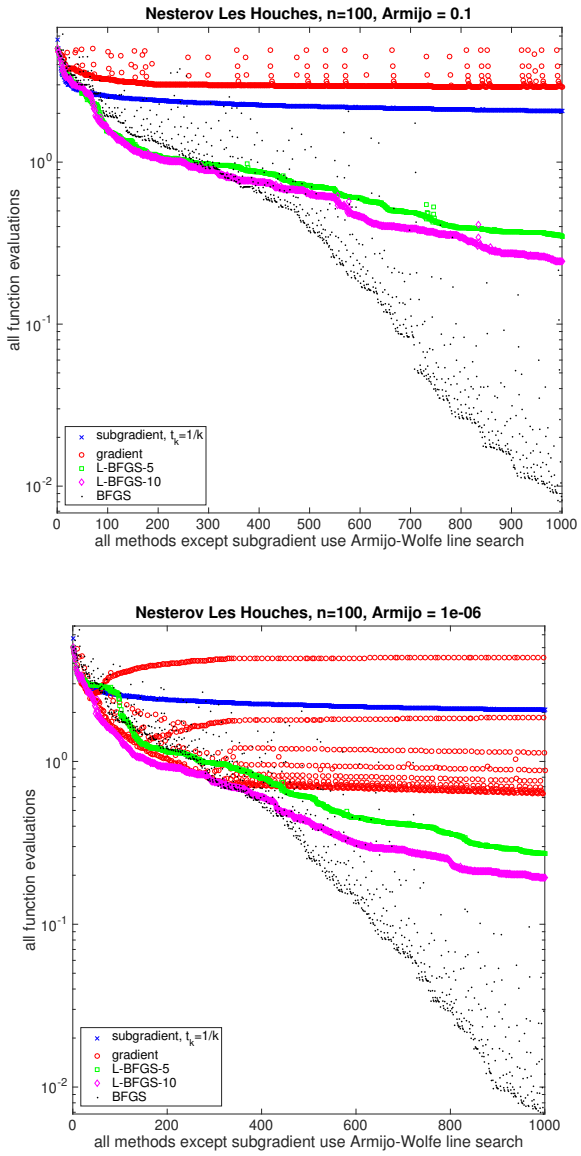


Figure 2.6: Comparison of five methods for minimizing Nesterov's ill-conditioned convex nonsmooth function $\hat{f}$. The subgradient method (blue crosses) uses $t_k = 1/k$. The gradient, limited-memory BFGS (with 5 and 10 updates respectively) and full BFGS methods (red circles, green squares, magenta diamonds and black dots) all use the Armijo-Wolfe bracketing line search. All function evaluations are shown. Top: Armijo parameter $c_1 = 0.1$. Bottom: Armijo parameter $c_1 = 10^{-6}$.

value within a given number of evaluations, the gradient method with the Armijo-Wolfe line search when the Armijo parameter is set to 10^{-6} performs better than using the subgradient method's predetermined sequence $t_k = 1/k$, but that this is not the case when the Armijo parameter is set to 0.1. The smaller value allows the gradient method to take steps with $t_k = 1$ early in the iteration, leading to rapid progress, while the larger value forces shorter steps, quickly leading to stagnation. Eventually, even the small Armijo parameter requires many steps in the line search — one can see that on the right side of the lower figure, at least 8 function values per iteration are required. One should not read too much into the results for one example, but the most obvious observation from Figure 2.6 is that the full BFGS and limited memory BFGS methods are much more effective than the gradient or subgradient methods. This distinction becomes far more dramatic if we run the methods for more iterations: BFGS is typically able to reduce $\hat{f}$ to about 10^{-12} in about 5000 function evaluations, while the gradient and subgradient methods fail to reduce $\hat{f}$ below 10^{-1} in the same number of function evaluations. The limited memory BFGS methods consistently perform better than the gradient/subgradient methods but worse than full BFGS. The value of the Armijo parameter c_1 has little effect on the BFGS variants.

These results are consistent with substantial prior experience with applying the full BFGS method to nonsmooth problems, both convex and nonconvex [Lewis & Overton, 2013, Curtis et al., 2017, Greenbaum et al., 2017, Guo & Lewis, 2018]. However, although the BFGS method requires far fewer operations per iteration than bundle methods or gradient sampling, it is still not practical when n is large. Hence, the attraction of limited-memory BFGS which, like the gradient and subgradient methods, requires only $O(n)$ operations per iteration. In the next chapter, we

investigate under what conditions the limited-memory BFGS method applied to the function f studied in this chapter generates iterates that converge to a non-optimal point, and, more generally, how reliable a choice it is for nonsmooth optimization.

Chapter 3

Analysis of the Limited Memory BFGS Method Applied to a Class of Nonsmooth Optimization Problems

In this chapter, we analyze the behavior of a specific variant of limited-memory BFGS (L-BFGS) applied to the same nonsmooth convex function (1.0.2) that we considered earlier. The chapter is organized as follows. In §3.1, we define the scaled memoryless BFGS method (scaled L-BFGS with just one update), using any line search satisfying the Armijo and Wolfe conditions, and derive some properties of the method applied to the function f in (1.0.2), initiated at any point x_0 with $x_0^{(1)} \neq 0$. In §3.1.1, we show that if $a \geqslant \sqrt{3(n-1)}$, the algorithm is well defined in the sense that Armijo-Wolfe steplengths always exist. In §3.2, we give our main theoretical results. First, in §3.2.1, we show that if $a \geqslant 2\sqrt{n-1}$, in the limit the absolute

value of the normalized search direction generated by the method converges to a constant vector. Then, in §3.2.2, we show that if a further satisfies a condition depending on the Armijo parameter, the method converges to a non-optimal point $\bar{x}$ with $\bar{x}^{(1)} = 0$. Furthermore, this condition is *weaker* than the corresponding condition for gradient method.

In §3.2.3, we show that, if the Armijo-Wolfe bracketing line search defined in Algorithm 1 in §2.2 is used, scaled memoryless BFGS converges to a non-optimal point when $a \geqslant 2\sqrt{n-1}$ *regardless* of the Armijo parameter. This is in sharp contrast to the gradient method using the same line search, for which success or failure on the function f depends on the Armijo parameter. In §3.3, we present some numerical experiments which support our theoretical results, and which indicate that the results may extend to scaled L-BFGS with any fixed number of updates m, and to more general piecewise linear functions. We make some concluding remarks in §3.4.

3.1 The Memoryless BFGS Method

First let f denote any locally Lipschitz function mapping $\mathbb{R}^n$ to $\mathbb{R}$, and let $x_{k-1} \in \mathbb{R}^n$ denote the $(k-1)$th iterate of an optimization algorithm where f is differentiable at x_{k-1} with gradient $\nabla f(x_{k-1})$. Let $d_{k-1} \in \mathbb{R}^n$ denote a descent direction, i.e., satisfying $\nabla f(x_{k-1})^T d_{k-1} < 0$. As earlier, let c_1 and c_2 denote the Armijo and Wolfe parameters, satisfying $0 < c_1 < c_2 < 1$.

To keep this chapter self-contained let us restate the Armijo and Wolfe line search conditions, using slightly different notation from that used earlier. We say

that the steplength t satisfies the Armijo condition at iteration $k-1$ if

$$f(x_{k-1} + td_{k-1}) \leqslant f(x_{k-1}) + c_1 t \nabla f(x_{k-1})^T d_{k-1} \tag{3.1.1}$$

and that it satisfies the Wolfe condition if

$$\nabla f(x_{k-1} + td_{k-1}) \text{ exists with } \nabla f(x_{k-1} + td_{k-1})^T d_{k-1} \geqslant c_2 \nabla f(x_{k-1})^T d_{k-1}. \tag{3.1.2}$$

It is known that if f is smooth or convex, and bounded below along the direction d_{k-1}, a point satisfying these conditions must exist (see [Lewis & Overton, 2013, Theorem 4.5] for weaker conditions on f for which this holds). Note that as long as f is differentiable at the initial iterate, defining subsequent iterates by $x_k = x_{k-1} + t_{k-1} d_{k-1}$, where (3.1.2) holds for $t = t_{k-1}$, ensures that f is differentiable at x_k.

We are now ready to define the memoryless BFGS method (L-BFGS with $m = 1$), also known as L-BFGS-1, with scaling, i.e., with H_k^0 defined by (1.0.1).

The algorithm is defined for any f, but its analysis will be specifically for (1.0.2).

$$d_0 = -\nabla f(x_0) \tag{3.1.3}$$

for $k = 1, 2, 3, \ldots,$ **define**

$t_{k-1} = t$ satisfying (3.1.1) and (3.1.2)

$$x_k = x_{k-1} + t_{k-1} d_{k-1} \tag{3.1.4}$$

$$s_{k-1} = x_k - x_{k-1} \tag{3.1.5}$$

$$y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1}) \tag{3.1.6}$$

$$V_{k-1} = I - \frac{y_{k-1} s_{k-1}^T}{y_{k-1}^T s_{k-1}} \tag{3.1.7}$$

$$H_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} V_{k-1}^T V_{k-1} + \frac{s_{k-1} s_{k-1}^T}{s_{k-1}^T y_{k-1}} \tag{3.1.8}$$

$$d_k = -H_k \nabla f(x_k) \tag{3.1.9}$$

end

Algorithm 2: Scaled memoryless BFGS, with input x_0

Let us adopt the convention that if no steplength t exists satisfying the Armijo and Wolfe conditions (3.1.1) and (3.1.2), the algorithm is terminated. Hence, for any smooth or convex function, termination implies that a direction d_{k-1} has been identified along which $f(x_{k-1} + t d_{k-1})$ is unbounded below.

Now let us restrict our attention to the convex function f given in (1.0.2). The question we address in this chapter is whether memoryless BFGS will succeed in identifying the fact that f is unbounded below, either because it generates a direction d for which no steplength t satisfying the Armijo and Wolfe conditions exists (in which case the algorithm terminates), or, alternatively, that it generates a sequence $\{x_k\}$ for which Armijo-Wolfe steplengths always exist, with $f(x_k) \downarrow -\infty$.

If neither event takes place, $\{f(x_k)\}$ is bounded below, which is regarded as failure, since f is unbounded below.

For the function (1.0.2), requiring t_{k-1} to satisfy (3.1.2), regardless of the value of the Wolfe parameter $c_2 \in (0,1)$, implies, via (3.1.4), the condition

$$\operatorname{sgn}(x_k^{(1)}) = -\operatorname{sgn}(x_{k-1}^{(1)}). \tag{3.1.10}$$

Via (3.1.5) we see that (3.1.10) is equivalent to the condition

$$|s_{k-1}^{(1)}| = |x_{k-1}^{(1)}| + |x_k^{(1)}|. \tag{3.1.11}$$

Without loss of generality, we assume that the initial point x_0 has a positive first component, i.e., $x_0^{(1)} > 0$, so that

$$\nabla f(x_k) = \begin{bmatrix} (-1)^k a \\ \mathbb{1} \end{bmatrix}, \tag{3.1.12}$$

where $\mathbb{1} \in \mathbb{R}^{n-1}$ is the column vector of all ones. Via (3.1.10) and (3.1.12), (3.1.6) is simply

$$y_{k-1} = \begin{bmatrix} (-1)^k 2a \\ \mathbb{0} \end{bmatrix}, \tag{3.1.13}$$

where $\mathbb{0} \in \mathbb{R}^{n-1}$ is the column vector of all zeros. Note that from (3.1.4) and (3.1.5) it is immediate that for any $k \geqslant 1$

$$s_{k-1} = t_{k-1} d_{k-1}. \tag{3.1.14}$$

For $i = 2, \ldots, n$, let

$$\theta_{k-1}^{(i)} = \arctan\left(\frac{d_{k-1}^{(i)}}{d_{k-1}^{(1)}}\right),$$

with $\theta_{k-1}^{(i)} \in [-\pi/2, \pi/2]$. Note that $|\theta_{k-1}^{(i)}|$ is the acute angle between d_{k-1} and the $x^{(1)}$ axis when it is projected onto the $(x^{(1)}, x^{(i)})$ plane. From (3.1.3) and (3.1.12) we have

$$\frac{1}{a} = \tan\theta_0^{(2)} = \tan\theta_0^{(3)} = \ldots = \tan\theta_0^{(n)}. \tag{3.1.15}$$

The assumption of the initial inverse Hessian approximation being a multiple of the identity is embedded in the definition (3.1.8), and therefore we know that d_{k-1} (and consequently s_{k-1}) is in the subspace spanned by the two gradients in (3.1.12) (see [Gill & Leonard, 2003, Lemma 2.1]). Since both gradients are symmetric w.r.t. the components $x^{(2)}, \ldots, x^{(n)}$, it follows that d_{k-1} has the same property. The same symmetry holds in the definition of the objective function (1.0.2). Since (3.1.15) holds, we conclude inductively that, for $k > 1$, $\tan\theta_{k-1}^{(2)} = \tan\theta_{k-1}^{(3)} = \ldots = \tan\theta_{k-1}^{(n)}$. So, let us simply write

$$b_{k-1} = \tan\theta_{k-1} = \frac{d_{k-1}^{(i)}}{d_{k-1}^{(1)}} = \frac{s_{k-1}^{(i)}}{s_{k-1}^{(1)}}, \quad \text{for all } i = 2, \ldots, n. \tag{3.1.16}$$

From (3.1.13) we have

$$s_{k-1}^T y_{k-1} = (-1)^k 2a s_{k-1}^{(1)}, \tag{3.1.17}$$

so we can rewrite V_{k-1} in (3.1.7) in terms of b_{k-1} as

$$V_{k-1} = \left[\begin{array}{c|c} 0 & -b_{k-1}\mathbb{1}^T \\ \hline \mathbb{0} & I_{n-1} \end{array}\right]. \tag{3.1.18}$$

This leads us to write H_k in (3.1.8) as

$$H_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} \left[\begin{array}{c|c} 0 & \mathbb{0}^T \\ \hline \mathbb{0} & b_{k-1}^2 \mathbb{1}\mathbb{1}^T + I_{n-1} \end{array}\right] + \frac{(s_{k-1}^{(1)})^2}{s_{k-1}^T y_{k-1}} \left[\begin{array}{c|c} 1 & b_{k-1}\mathbb{1}^T \\ \hline b_{k-1}\mathbb{1} & b_{k-1}^2 \mathbb{1}\mathbb{1}^T \end{array}\right].$$

From (3.1.17) we can see that the fractions in front of the first and second matrices are the same, i.e.,

$$\frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} = \frac{(s_{k-1}^{(1)})^2}{s_{k-1}^T y_{k-1}} = \frac{|s_{k-1}^{(1)}|}{2a}. \tag{3.1.19}$$

Hence, we obtain the following much more compact form

$$H_k = \gamma_k \left[\begin{array}{c|c} 1 & b_{k-1}\mathbb{1}^T \\ \hline b_{k-1}\mathbb{1} & 2b_{k-1}^2 \mathbb{1}\mathbb{1}^T + I_{n-1} \end{array}\right], \tag{3.1.20}$$

where

$$\gamma_k = \frac{|s_{k-1}^{(1)}|}{2a} \tag{3.1.21}$$

is the scale factor in (1.0.1). Finally, with the gradient defined in (3.1.12) we can compute the direction generated by Algorithm 1 in (3.1.9) as

$$d_k = -\frac{|s_{k-1}^{(1)}|}{2a} \begin{bmatrix} (-1)^k a + (n-1)b_{k-1} \\ \Big((-1)^k a b_{k-1} + 2(n-1)b_{k-1}^2 + 1\Big)\mathbb{1} \end{bmatrix}. \tag{3.1.22}$$

So, from definition (3.1.16) we can write b_k recursively as

$$b_k = \frac{(-1)^k a b_{k-1} + 2(n-1)b_{k-1}^2 + 1}{(-1)^k a + (n-1)b_{k-1}}. \tag{3.1.23}$$

3.1.1 Existence of Armijo-Wolfe Steps when $\sqrt{3(n-1)} \leqslant a$

In the next lemma we prove that if $\sqrt{3(n-1)} \leqslant a$, then the $\{b_k\}$ alternate in sign with $|b_k| \leqslant 1/a$.

Lemma 3.1.1 *Suppose* $\sqrt{3(n-1)} \leqslant a$. *Define* b_k *as in (3.1.23) with* $b_0 = 1/a$. *Then* $|b_k| \leqslant 1/a$ *and furthermore* $\{b_k\}$ *alternates in sign with*

$$|b_k| = \frac{1 + (n-1)b_{k-1}^2}{a - (n-1)|b_{k-1}|} - |b_{k-1}|. \tag{3.1.24}$$

Proof: Suppose $\sqrt{3(n-1)} \leqslant a$. Using a change of variable such that $\beta_k = b_k$ when k is even, and $\beta_k = -b_k$ when k is odd, (3.1.23) becomes

$$\beta_k = \frac{1 + (n-1)\beta_{k-1}^2}{a - (n-1)\beta_{k-1}} - \beta_{k-1}. \tag{3.1.25}$$

From (3.1.15) we have $\beta_0 = 1/a$. Using induction we prove that $0 < \beta_k \leqslant 1/a$. This is clearly true for $k = 0$. Suppose we have $0 < \beta_{k-1} \leqslant 1/a$. Hence

$$\beta_{k-1} < \frac{1}{a - (n-1)\beta_{k-1}} < \frac{1 + (n-1)\beta_{k-1}^2}{a - (n-1)\beta_{k-1}},$$

so, dropping the middle term and moving β_{k-1} to the R.H.S., we get exactly the definition of β_k according to (3.1.25). So, we have $0 < \beta_k$. Next, starting from

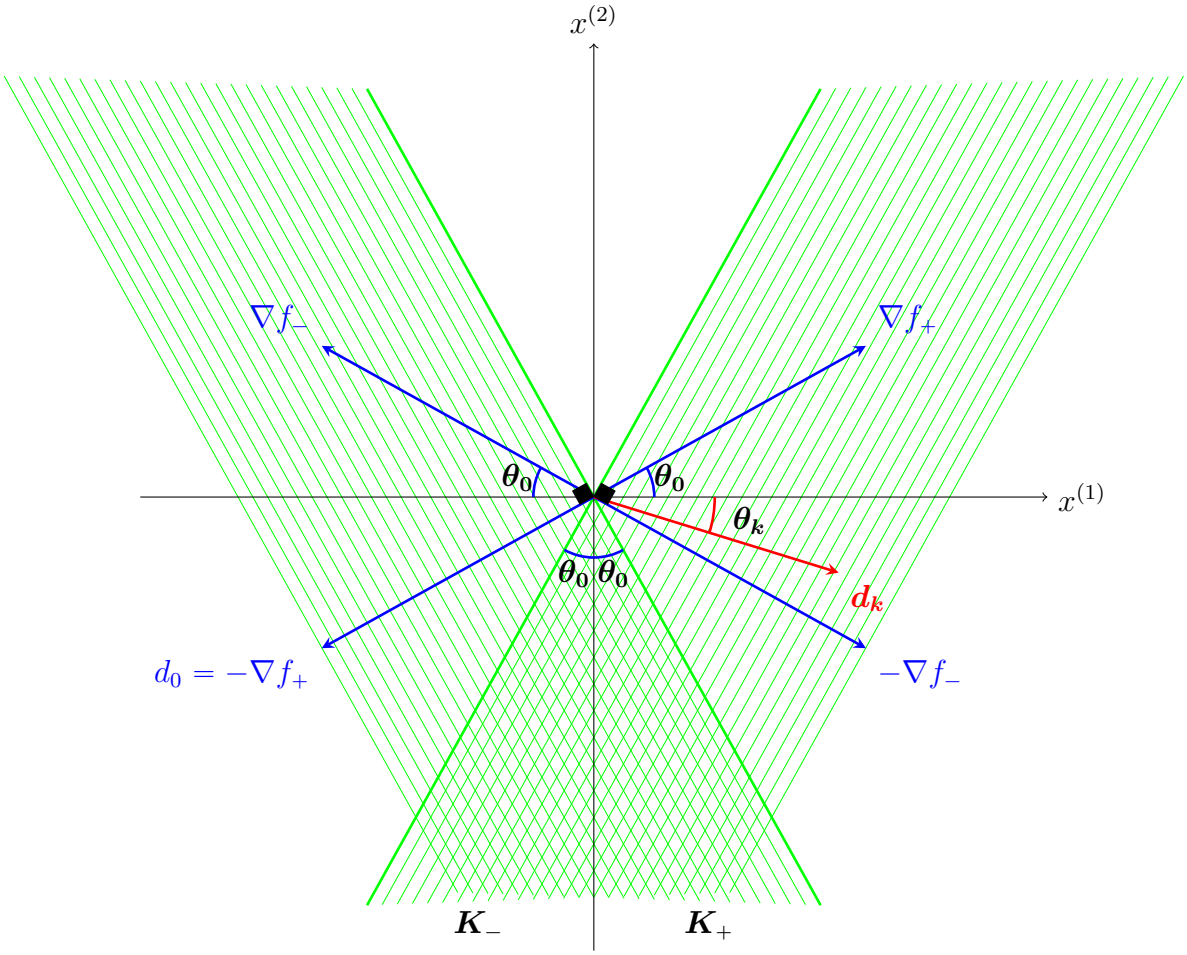


Figure 3.1: Angles of Search Directions. Let $n = 2$, let $\nabla f_+ = [a\ 1]^T$ and let $\nabla f_- = [-a\ 1]^T$, so, since $x_0^{(1)} > 0$ by assumption, we have $d_0 = -\nabla f_+$. It follows from Lemma 3.1.1 that $b_k = d_k^{(2)}/d_k^{(1)}$ alternates in sign for $k = 1, 2, \ldots$, with absolute value bounded above by $1/a$, and hence that $\theta_k = \arctan(b_k)$ alternates in sign for $k = 1, 2, \ldots$, with $|\theta_k|$, the acute angle between the $x^{(1)}$ axis and the search direction d_k, bounded above by θ_0. Furthermore, Lemma 3.1.2 states that the function f is unbounded below along all directions in the open cones K_- and K_+, and bounded below along all other directions (except the vertical axis). Note, however, that points satisfying the Wolfe condition may exist along directions $d \in K_+$ emanating from iterates on the left side of the $x^{(2)}$ axis, but not along directions $d \in K_-$ emanating from the left side, because the former cross the $x^{(2)}$ axis and the latter do not, and vice versa. Finally, Theorem 3.1.3 implies that, under the assumption $a \geqslant \sqrt{3}$, we have $|\theta_k| \leqslant \theta_0 \leqslant \pi/6$, for all $k > 0$ (see the discussion after the theorem), so d_k does not lie in K_- or in K_+ and hence the algorithm does not terminate.

$\sqrt{3(n-1)} \leqslant a$, we show that $\beta_k \leqslant 1/a$:

$$\begin{aligned}
&\frac{3(n-1)}{a} \leqslant a \Rightarrow \\
&\frac{(n-1)}{a} + 2(n-1)\beta_{k-1} \leqslant a \Rightarrow \\
&\frac{a^2+n-1}{a} \leqslant 2(a-(n-1)\beta_{k-1}) \Rightarrow \\
&\frac{a^2+n-1}{a(a-(n-1)\beta_{k-1})} \leqslant 2.
\end{aligned}$$

Multiplying both sides by β_{k-1} we get

$$\frac{a\beta_{k-1}+1}{a-(n-1)\beta_{k-1}} - \frac{1}{a} \leqslant 2\beta_{k-1},$$

and finally by moving $1/a$ to the right and $2\beta_{k-1}$ to the left we get

$$\frac{1+(n-1)\beta_{k-1}^2}{a-(n-1)\beta_{k-1}} - \beta_{k-1} \leqslant \frac{1}{a}.$$

The L.H.S. is β_k as it's defined in (3.1.25), so $\beta_k \leqslant 1/a$. Recalling the change of variable in the beginning of the proof it follows that $\beta_k = |b_k|$. So, from (3.1.25) we get (3.1.24). ∎

Putting (3.1.23) and (3.1.24) together we can rewrite (3.1.22) as

$$d_k = -\frac{|s_{k-1}^{(1)}|}{2a}(a-(n-1)|b_{k-1}|) \begin{bmatrix} (-1)^k \\ |b_k|\mathbb{1} \end{bmatrix}. \tag{3.1.26}$$

Before stating the main result of this section we give the following simple lemma.

Lemma 3.1.2 *Let $x \in \mathbb{R}^n$ be given, define*

$$d_+ = -\begin{bmatrix} 1 \\ \beta\mathbb{1} \end{bmatrix} \quad \text{and} \quad d_- = -\begin{bmatrix} -1 \\ \beta\mathbb{1} \end{bmatrix}, \tag{3.1.27}$$

where $\beta > 0$, and define f by (1.0.2). Let d be either d_+ or d_-. Then $h(t) = f(x + td) - f(x)$ is unbounded below if and only if $\dfrac{a}{n-1} < \beta$.

Proof: We have

$$h(t) = a|x^{(1)} \pm t| - a|x^{(1)}| - (n-1)\beta t.$$

So,

$$(a - (n-1)\beta)\, t - 2a|x^{(1)}| < h(t) < (a - (n-1)\beta)\, t.$$

The result follows. ■

Note that stating that h is unbounded below is not equivalent to saying that Armijo-Wolfe points do not exist along the direction d emanating from x. Such points may exist if the sign of $d^{(1)}$ is opposite to the sign of $x^{(1)}$.

Theorem 3.1.3 *When Algorithm 1 is applied to (1.0.2) with $\sqrt{3(n-1)} \leqslant a$, using any Armijo-Wolfe line search, with any starting point x_0 such that $x_0^{(1)} \neq 0$, the method generates directions d_k that are nonnegative scalar multiples of d_+ or d_-, defined in (3.1.27), with $\beta < a/(n-1)$. It follows that the steplength t_k satisfying the Armijo and Wolfe conditions (3.1.1) and (3.1.2) always exists and hence the method never terminates.*

Proof: The proof is by induction on k. Without loss of generality assume $x_0^{(1)} > 0$, so $d_0 = -\nabla f(x_0) = a d_+$ with $\beta = 1/a$. Since $\sqrt{3(n-1)} \leqslant a$, we have

$1/a < a/(n-1)$ and hence the initial Armijo-Wolfe steplength t_0 exists by Lemma 3.1.2. Now, suppose that the result holds for all $j < k$, so d_k in (3.1.26) is well defined. Since by Lemma 3.1.1 we know that $|b_{k-1}| \leqslant 1/a \leqslant a/(n-1)$, the leading scalar in (3.1.26) is negative and therefore d_k is a nonnegative scalar multiple of d_+ or d_- with $\beta = |b_k| \leqslant 1/a < a/(n-1)$. Hence f is bounded below along the direction d_k emanating from x_k and so there exists t_k satisfying the Armijo and Wolfe conditions at iteration k, which implies that the algorithm does not terminate at iteration k. ■

Using Figure 3.1 we can provide an alternative informal geometrical proof for Theorem 3.1.3. We have

$$\frac{1}{a} \leqslant \frac{1}{\sqrt{3}} \quad \Rightarrow \quad \theta_0 = \arctan\frac{1}{a} \;\leqslant\; \arctan\frac{1}{\sqrt{3}} = \frac{\pi}{6}.$$

According to Lemma 3.1.1, we have $|b_k| \leqslant 1/a$, and so, $|\theta_k| \leqslant \theta_0$ and hence,

$$2\theta_0 + |\theta_k| \leqslant \frac{\pi}{2}.$$

It follows (see Figure 3.1) that $d_k \notin K_+ \cup K_-$. This means that the method never generates a direction along which f is unbounded below.

However, Theorem 3.1.3 does not imply that Algorithm 1 converges to a non-optimal point under the assumption that $\sqrt{3(n-1)} \leqslant a$, because the existence of Armijo-Wolfe steps t_k for all k does not imply that the sequence $\{f(x_k)\}$ is bounded below. This issue is addressed in the next section.

3.2 Failure of Scaled Memoryless BFGS

3.2.1 Convergence of the Absolute Value of the Normalized Search Direction when $2\sqrt{n-1} \leqslant a$

Define

$$b = \frac{a - \sqrt{a^2 - 3(n-1)}}{3(n-1)} \tag{3.2.1}$$

and note that when $\sqrt{3(n-1)} \leqslant a$, then

$$\frac{1}{2a} \leqslant b \leqslant \frac{1}{a}.$$

Next we show the sequence $\{|b_k|\}$ converges to b under a slightly stronger assumption.

Theorem 3.2.1 *For $2\sqrt{n-1} \leqslant a$ the sequence defined by (3.1.24) converges and moreover*

$$\lim_{k\to\infty} |b_k| = b.$$

Proof: We continue to use the same change of variable as before, that is $\beta_k = b_k$ when k is even, and $\beta_k = -b_k$ when k is odd. In this way, (3.1.25) is equivalent to (3.1.24), and we prove that if $2\sqrt{n-1} \leqslant a$, then $\{\beta_k\}$ converges. From a little rearrangement in (3.1.25) we can easily get

$$a(\beta_k + \beta_{k-1}) = 1 + 2(n-1)\beta_{k-1}^2 + (n-1)\beta_{k-1}\beta_k, \tag{3.2.2}$$

and by moving $(n-1)\beta_{k-1}\beta_k$ to the left and adding 1 to both sides we get

$$a(\beta_k + \beta_{k-1}) - (n-1)\beta_{k-1}\beta_k + 1 = 2\Big(1 + (n-1)\beta_{k-1}^2\Big). \tag{3.2.3}$$

For further simplification we define

$$\rho_k = \frac{1 + (n-1)\beta_k^2}{a - (n-1)\beta_k}, \tag{3.2.4}$$

so we can rewrite (3.1.25) as

$$\beta_{k+1} = \rho_k - \beta_k. \tag{3.2.5}$$

By applying (3.2.5) recursively we obtain

$$\beta_{k+1} - \beta_{k-1} = \rho_k - \rho_{k-1}. \tag{3.2.6}$$

Note that from (3.2.4) we have

$$\begin{aligned}
\rho_k - \rho_{k-1} &= \frac{1 + (n-1)\beta_k^2}{a - (n-1)\beta_k} - \frac{1 + (n-1)\beta_{k-1}^2}{a - (n-1)\beta_{k-1}} \\
&= \frac{\Big(1 + (n-1)\beta_k^2\Big)\Big(a - (n-1)\beta_{k-1}\Big) - \Big(1 + (n-1)\beta_{k-1}^2\Big)\Big(a - (n-1)\beta_k\Big)}{\Big(a - (n-1)\beta_k\Big)\Big(a - (n-1)\beta_{k-1}\Big)} \\
&= \frac{(\beta_k - \beta_{k-1})(n-1)\Big(a(\beta_k + \beta_{k-1}) - (n-1)\beta_{k-1}\beta_k + 1\Big)}{\Big(a - (n-1)\beta_k\Big)\Big(a - (n-1)\beta_{k-1}\Big)}.
\end{aligned} \tag{3.2.7}$$

The last factor in the numerator is the L.H.S. in (3.2.3), so

$$\rho_k - \rho_{k-1} = \frac{(\beta_k - \beta_{k-1})(n-1)2\Big(1 + (n-1)\beta_{k-1}^2\Big)}{\Big(a - (n-1)\beta_k\Big)\Big(a - (n-1)\beta_{k-1}\Big)}. \tag{3.2.8}$$

Hence, since all of the factors in this product except $(\beta_k - \beta_{k-1})$ are known to be positive, we have

$$(\rho_k - \rho_{k-1})(\beta_k - \beta_{k-1}) \geqslant 0. \tag{3.2.9}$$

Putting (3.2.6) and (3.2.9) together we conclude

$$(\beta_{k+1} - \beta_{k-1})(\beta_k - \beta_{k-1}) \geqslant 0. \tag{3.2.10}$$

As the next step we will show that

$$(\beta_{k+1} - \beta_k)(\beta_k - \beta_{k-1}) \leqslant 0. \tag{3.2.11}$$

Since $a \geqslant 2\sqrt{n-1}$ and using $1/a \geqslant \beta_{k-1}$ we get

$$\begin{aligned}
&\Big(a^2 - 4(n-1)\Big)\Big(a^2 + (n-1)\Big) \geqslant 0 \Rightarrow \\
&a^2 - 3(n-1) \geqslant \frac{4(n-1)^2}{a^2} \Rightarrow \\
&a^2 - 3(n-1) \geqslant 4(n-1)^2\beta_{k-1}^2 \Rightarrow \\
&a^2 - 3(n-1) - 4(n-1)^2\beta_{k-1}^2 \geqslant 0.
\end{aligned}$$

By adding and deducting $2(n-1)^2\beta_k\beta_{k-1}$ to the L.H.S. above we get

$$a^2 - 2(n-1)\Big(1 + 2(n-1)\beta_{k-1}^2 + (n-1)\beta_{k-1}\beta_k\Big) + 2(n-1)^2\beta_k\beta_{k-1} - (n-1) \geqslant 0.$$

By combining this with (3.2.2) we get

$$a^2 - 2(n-1)a(\beta_k + \beta_{k-1}) + 2(n-1)^2\beta_k\beta_{k-1} - (n-1) \geqslant 0.$$

By moving some of the terms to the R.H.S. and factorizing the L.H.S. we get

$$\Big(a - (n-1)\beta_k\Big)\Big(a - (n-1)\beta_{k-1}\Big) \geqslant a(n-1)(\beta_k + \beta_{k-1}) - (n-1)^2\beta_k\beta_{k-1} + (n-1),$$

which we can write as

$$1 \geqslant \frac{(n-1)\Big(a(\beta_k+\beta_{k-1})-(n-1)\beta_k\beta_{k-1}+1\Big)}{\Big(a-(n-1)\beta_k\Big)\Big(a-(n-1)\beta_{k-1}\Big)}. \tag{3.2.12}$$

Now, suppose $\beta_k - \beta_{k-1} \geqslant 0$. Multiplying both sides of the inequality (3.2.12) by $\beta_k - \beta_{k-1}$, according to (3.2.7) we get

$$\beta_k - \beta_{k-1} \geqslant \rho_k - \rho_{k-1},$$

so,

$$\rho_{k-1} - \beta_{k-1} \geqslant \rho_k - \beta_k$$

which means that via (3.2.5) we have shown $\beta_k \geqslant \beta_{k+1}$. Alternatively, if we had $\beta_k - \beta_{k-1} \leqslant 0$ above, then we would get $\beta_k \leqslant \beta_{k+1}$. Hence, we always have $(\beta_{k+1} - \beta_k)(\beta_k - \beta_{k-1}) \leqslant 0$, which is exactly inequality (3.2.11).

Since we start with $\beta_0 = 1/a$, according to Lemma 3.1.1 we have $\beta_1 \leqslant \beta_0$. Using (3.2.11) inductively we get

$$\beta_1 - \beta_0 \leqslant 0, \quad 0 \leqslant \beta_2 - \beta_1, \quad \beta_3 - \beta_2 \leqslant 0, \ldots$$

and from applying (3.2.10) to each one of these inequalities we conclude

$$\beta_2 - \beta_0 \leqslant 0, \quad 0 \leqslant \beta_3 - \beta_1, \quad \beta_4 - \beta_2 \leqslant 0, \ldots$$

which shows that we can split $\{\beta_k\}$ into two separate monotonically decreasing and

increasing subsequences:

$$0 < \ldots \beta_4 \leqslant \beta_2 \leqslant \beta_0 = 1/a,$$
$$0 < \beta_1 \leqslant \beta_3 \leqslant \beta_5 \ldots < 1/a.$$

By the bounded monotone convergence theorem we conclude that each one of these subsequences converge, i.e.

$$\lim_{k\to\infty} |\beta_{k+2} - \beta_k| = 0,$$

and recalling (3.2.6) we get

$$\lim_{k\to\infty} |\rho_{k+1} - \rho_k| = 0.$$

On the other hand, looking at the equality in (3.2.7) we know that except $(\beta_{k+1}-\beta_k)$ all the factors in the numerator and denominator are bounded away from zero. So therefore we must have

$$\lim_{k\to\infty} |\beta_{k+1} - \beta_k| = 0,$$

and hence, since the even and odd sequences both converge, they must have the same limit. Using the definition of β_{k+1} in (3.1.25) we get

$$\lim_{k\to\infty} \left| \frac{1 + (n-1)\beta_k^2}{a - (n-1)\beta_k} - 2\beta_k \right| = 0.$$

Since the denominator is bounded away from zero we must have

$$\lim_{k\to\infty} 3(n-1)\beta_k^2 - 2a\beta_k + 1 = 0.$$

The two roots of the limiting quadratic equation are

$$\frac{a \pm \sqrt{a^2 - 3(n-1)}}{3(n-1)}.$$

The smaller root is b as defined in (3.2.1) and the larger root is greater than $1/a$, which according to Lemma 3.1.1 is not possible. Hence,

$$\lim_{k\to\infty} \beta_k = \lim_{k\to\infty} |b_k| = b.$$

■

Note that the convergence result established in this theorem does not require any assumption of symmetry with respect to variables $2, 3, \ldots, n$, in the initial point x_0. The only assumption on x_0 is that $x_0^{(1)} > 0$. We need $x_0^{(1)} \neq 0$ so that f is differentiable at x_0; the assumption on the sign is purely for convenience.

Assumption 1. For the subsequent theoretical analysis we assume that

$$2\sqrt{n-1} \leqslant a.$$

With this assumption, as a direct implication of Theorem 3.2.1, for any given positive ϵ there exists K such that for $k \geqslant K$ we have

$$||b_k| - b| < \frac{\epsilon}{n-1}. \tag{3.2.13}$$

As we showed in Lemma 3.1.1, for $k \geqslant 0$ we have $|b_k| \leqslant 1/a$ and therefore

$$\frac{3(n-1)}{a} \leqslant a - \frac{n-1}{a} \leqslant a - (n-1)|b_k|. \tag{3.2.14}$$

Thus, $a-(n-1)|b_k|$ is positive and bounded away from zero.

Since $|b_k|$ converges by Theorem 3.2.1, we see that in the limit the normalized direction $d_k/\|d_k\|_2$ alternates between two limiting directions. For an illustration, see Figures 3.2 and 3.3. It is this property that allows us to establish, under some subsequent assumptions, that scaled memoryless BFGS generates iterates x_k for which $f(x_k)$ is bounded below even though f is unbounded below.

3.2.2 Dependence on the Armijo Condition

Combining (3.1.12) and (3.1.26) we get

$$\nabla f(x_k)^T d_k = -|d_k^{(1)}| \begin{bmatrix} (-1)^k a \\ \mathbb{1} \end{bmatrix}^T \begin{bmatrix} (-1)^k \\ |b_k|\mathbb{1} \end{bmatrix} = -|d_k^{(1)}| \left(a + (n-1)\,|b_k|\right), \tag{3.2.15}$$

so the Armijo condition (3.1.1) with $t = t_k$ at iteration k is

$$c_1 t_k |d_k^{(1)}| \left(a + (n-1)\,|b_k|\right) \leqslant f(x_k) - f(x_k + t_k d_k). \tag{3.2.16}$$

If t_k satisfies the Wolfe condition, i.e. t_k is large enough that the sign change (3.1.10) occurs, then we must have

$$|x_k^{(1)}| < t_k |d_k^{(1)}|. \tag{3.2.17}$$

Given this we can derive $f(x_k) - f(x_k + t_k d_k)$ using the definition of b_k in (3.1.16) as follows:

$$f(x_k) - f(x_k + t_k d_k) = 2a|x_k^{(1)}| - (a-(n-1)|b_k|)\, t_k |d_k^{(1)}|. \tag{3.2.18}$$

By defining φ_k as follows

$$\varphi_k = \frac{c_1\left(a + (n-1)|b_k|\right) + a - (n-1)|b_k|}{2a}, \tag{3.2.19}$$

we can restate the Armijo condition in the following lemma.

Lemma 3.2.2 *Suppose t_k satisfies the Wolfe condition (3.1.10). Then for t_k to satisfy the Armijo condition (3.2.16) we must have*

$$\varphi_k t_k |d_k^{(1)}| \leqslant |x_k^{(1)}|. \tag{3.2.20}$$

Proof: Combining (3.2.18) and (3.2.16) we get

$$c_1 t_k |d_k^{(1)}| \left(a + (n-1)\,|b_k|\right) \leqslant 2a|x_k^{(1)}| - \left(a - (n-1)\,|b_k|\right) t_k |d_k^{(1)}|,$$

and using the definition of φ_k in (3.2.19), (3.2.20) follows. ∎

From (3.2.17) and (3.2.20) we see that φ_k is the ratio of the lower bound and the upper bound on the steplength t_k provided by the Wolfe and Armijo conditions respectively. The next lemma provides bounds on φ_k.

Lemma 3.2.3

$$\frac{(n-1)|b_k|}{a} < \varphi_k. \tag{3.2.21}$$

Proof: Using Lemma 3.1.1 we know $3(n-1)|b_k| \leqslant a$ for all k, and so

$$2(n-1)|b_k| \leqslant a - (n-1)|b_k|,$$

and since

$$\frac{a-(n-1)|b_k|}{2a} = \varphi_k - c_1 \frac{a+(n-1)|b_k|}{2a},$$

and $c_1 > 0$, (3.2.21) follows. ■

Corollary 3.2.4 *For $k \geqslant 1$ we have*

$$|s_k^{(1)}| \leqslant |s_{k-1}^{(1)}| \frac{1-\varphi_{k-1}}{\varphi_k}. \tag{3.2.22}$$

Proof: Summing the Armijo inequality (3.2.20) for two consecutive iterations we obtain

$$|s_{k-1}^{(1)}|\varphi_{k-1} + |s_k^{(1)}|\varphi_k \leqslant |x_{k-1}^{(1)}| + |x_k^{(1)}|,$$

and noticing that the R.H.S., according to (3.1.11), is equal to $|s_{k-1}^{(1)}|$ we get (3.2.22).

■

Lemma 3.2.5 *For any given $\epsilon > 0$ let K be the smallest integer such that for any $k \geqslant K$, (3.2.13) holds. Then for all $N > K$ we have*

$$f(x_K) - f(x_N) < a|x_K^{(1)}| + ((n-1)\,b + \epsilon) \sum_{k=K}^{N-1} |s_k^{(1)}|. \tag{3.2.23}$$

Proof: Using $t_k d_k = s_k$ and $x_{k+1} = x_k + s_k$ in (3.2.18) and then applying (3.2.13) we obtain

$$f(x_k) - f(x_{k+1}) < 2a|x_k^{(1)}| - a|s_k^{(1)}| + ((n-1)\,b + \epsilon)\,|s_k^{(1)}|. \tag{3.2.24}$$

Summing up (3.2.24) from $k = K$ to $k = N-1$ and recalling (3.1.11), we get

$$f(x_K) - f(x_N) <$$
$$a\sum_{k=K}^{N-1} |s_k^{(1)}| + a|x_K^{(1)}| - a|x_N^{(1)}| - a\sum_{k=K}^{N-1} |s_k^{(1)}| + ((n-1)\,b + \epsilon)\sum_{k=K}^{N-1} |s_k^{(1)}|.$$

Canceling the first and fourth terms and dropping $-a|x_N|$, we arrive at (3.2.23). ■

From applying Theorem 3.2.1 to the definition of φ_k in (3.2.19) it is immediate that $\{\varphi_k\}$ converges. Let

$$\varphi = \frac{c_1\,(a + (n-1)b) + a - (n-1)b}{2a}, \tag{3.2.25}$$

so

$$\lim_{k\to\infty} \varphi_k = \varphi. \tag{3.2.26}$$

Lemma 3.2.6 *Assume*

$$0 < \epsilon \leqslant \frac{\sqrt{a^2 - 3(n-1)}}{3}, \tag{3.2.27}$$

and let K be defined as in Lemma 3.2.5. Then for any $k \geqslant K$ we have

$$\left|\frac{1-\varphi_{k-1}}{\varphi_k} - \frac{1-\varphi}{\varphi}\right| < \frac{15}{a}\epsilon. \tag{3.2.28}$$

Proof: By rearranging terms in (3.2.1) and using (3.2.27) we get

$$(n-1)b + \epsilon \leqslant (n-1)b + \frac{\sqrt{a^2 - 3(n-1)}}{3} = \frac{a}{3}. \tag{3.2.29}$$

Using (3.2.13) and (3.2.29), for $k \geqslant K$ we have

$$0 < a - (n-1)b - \epsilon < a - (n-1)|b_k|.$$

Combining this with (3.2.21) we get

$$0 < \frac{a - (n-1)b - \epsilon}{2a} < \varphi_k < 1.$$

Hence,

$$1 < \frac{1}{\varphi_k} < \frac{2a}{a - (n-1)b - \epsilon} \leqslant \frac{2a}{a - \frac{a}{3}} = 3.$$

Since $0 < c_1 < 1$, from (3.2.13), (3.2.19), (3.2.25) and (3.2.26) we get

$$|\varphi_k - \varphi| < \frac{(1 + c_1)\epsilon}{2a} < \frac{\epsilon}{a}.$$

So,

$$\begin{aligned} \left| \frac{1 - \varphi_{k-1}}{\varphi_k} - \frac{1 - \varphi}{\varphi} \right| &= \left| \frac{1}{\varphi_k} - 1 + \frac{\varphi_k - \varphi_{k-1}}{\varphi_k} - \frac{1}{\varphi} + 1 \right| \\ &< \left| \frac{\varphi - \varphi_k}{\varphi_k \varphi} \right| + \left| \frac{\varphi_k - \varphi_{k-1}}{\varphi_k} \right| < \frac{\epsilon}{a\varphi_k} \left(\frac{1}{\varphi} + 2 \right). \end{aligned}$$

Note that $1 < 1/\varphi_k < 3$ applies to all φ_k (as well as the limit φ) with $k \geqslant K$, and therefore we conclude (3.2.28). ∎

Let

$$\psi_\epsilon = \frac{1 - \varphi}{\varphi} + \frac{15}{a}\epsilon. \tag{3.2.30}$$

If Lemma 3.2.6 applies then from (3.2.22) and (3.2.28) we conclude

$$|s_k^{(1)}| < \psi_\epsilon |s_{k-1}^{(1)}|. \tag{3.2.31}$$

That is to say, with ϵ satisfying (3.2.27), after at most K iterations, (3.2.31) holds. Consequently, with the additional assumption $\psi_\epsilon < 1$, we obtain

$$\sum_{k=K}^{N-1} |s_k^{(1)}| < |s_K^{(1)}| \frac{1}{1-\psi_\epsilon}. \tag{3.2.32}$$

Now we can prove the main result of this subsection. Recall that $c_1 < 1$.

Theorem 3.2.7 *Suppose c_1 is chosen large enough that*

$$\frac{1}{c_1} - 1 < \frac{a}{(n-1)b} \tag{3.2.33}$$

holds. Then, using any Armijo-Wolfe line search with any starting point x_0 with $x_0^{(1)} \neq 0$, scaled memoryless BFGS applied to (1.0.2) fails in the sense that $f(x_N)$ is bounded below as $N \to \infty$.

Proof: It follows from (3.2.33) and (3.2.25) that $\varphi > 1/2$. Therefore, using (3.2.30), we can choose ϵ small enough such that $\psi_\epsilon < 1$ holds in addition to (3.2.27). Applying Lemmas 3.2.5 and 3.2.6, we conclude that there exists K such that for for any $N > K$, (3.2.32) holds, and, substituting this into (3.2.23) we get

$$f(x_K) - f(x_N) < a|x_K^{(1)}| + |s_K^{(1)}| \frac{(n-1)\,b + \epsilon}{1-\psi_\epsilon}. \tag{3.2.34}$$

This establishes that $f(x_N)$ is bounded below for all $N > K$. ∎

Using (3.2.1) we see that the failure condition (3.2.33) for scaled memoryless BFGS

with any Armijo-Wolfe line search applied to (1.0.2) is equivalent to

$$\frac{1-c_1}{c_1}(n-1) \;<\; a^2 + a\sqrt{a^2-3(n-1)}. \tag{3.2.35}$$

The corresponding failure condition for the gradient method on the same function, again using any Armijo-Wolfe line search, is, as we showed in Chapter 2

$$\frac{1-c_1}{c_1}(n-1) \;<\; a^2. \tag{3.2.36}$$

Hence, scaled memoryless BFGS fails under a *weaker* condition relating a to the Armijo parameter than the condition for failure of the gradient method on the same function with the same line search conditions. Indeed, Assumption 1 implies

$$a^2 + a\sqrt{a^2-3(n-1)} \geqslant 4(n-1) + 2\sqrt{n-1}\sqrt{n-1} = 6(n-1).$$

So, if the Armijo parameter $c_1 \geqslant 1/7$, then (3.2.35) holds. In contrast, the same assumption implies that if $c_1 \geqslant 1/5$, then (3.2.36) holds. So, scaled memoryless BFGS with any Armijo-Wolfe line search applied to (1.0.2) fails under a weaker condition on the Armijo parameter than the gradient method does.

3.2.3 Results for a specific Armijo-Wolfe line search, independent of the Armijo parameter

Considering only the first component of the direction d_k in (3.1.26) we have

$$\frac{2a}{a-(n-1)|b_{k-1}|}|d_k^{(1)}| = |s_{k-1}^{(1)}|. \tag{3.2.37}$$

Using (3.1.14), it follows that if

$$t_k < \frac{2a}{a-(n-1)|b_{k-1}|}, \tag{3.2.38}$$

we have $|s_k^{(1)}| < |s_{k-1}^{(1)}|$. Note that the R.H.S. of (3.2.38) is greater than two. However, as shown in the next lemma, except at the initial iteration ($k = 0$), $t = 2$ is always large enough to satisfy the Wolfe condition, implying that there exists $t \leqslant 2$ satisfying both the Armijo and Wolfe conditions.

Lemma 3.2.8 *For $k \geqslant 1$, the steplength $t_k = 2$ always satisfies the Wolfe condition (3.1.10), i.e., we have*

$$|x_k^{(1)}| < 2|d_k^{(1)}|. \tag{3.2.39}$$

Proof: Since $k \geqslant 1$, we know that the Armijo and Wolfe conditions hold at iteration $k-1$ by definition of Algorithm 1. So, using (3.2.20) and (3.1.14) we have

$$\varphi_{k-1}|s_{k-1}^{(1)}| \leqslant |x_{k-1}^{(1)}|. \tag{3.2.40}$$

Using the inequality (3.2.21) in the L.H.S. and the equality (3.1.11) in the R.H.S. we get

$$\frac{(n-1)|b_{k-1}|}{a}|s_{k-1}^{(1)}| < |s_{k-1}^{(1)}| - |x_k^{(1)}|,$$

i.e.

$$|x_k^{(1)}| < |s_{k-1}^{(1)}|\frac{a-(n-1)|b_{k-1}|}{a}.$$

Substituting (3.2.37) into the R.H.S., we obtain (3.2.39). ∎

Consider the Armijo-Wolfe bracketing line search Algorithm 1 in §2.2. It is known from the results in [Lewis & Overton, 2013] that provided f is bounded

below along d_{k-1} (as we already established must hold for directions generated by Algorithm 2 in §3.1), the Armijo-Wolfe bracketing line search will terminate with a steplength t satisfying both conditions. In the following lemma we show that if we use this line search, it always generates $t_k \leqslant 2$ for $k \geqslant 1$.

Lemma 3.2.9 *When scaled memoryless BFGS is applied to (1.0.2), using Algorithm 1 in §2.2 it always returns steplength $t_k \leqslant 2$ for $k \geqslant 1$.*

Proof: *The line search begins with the unit step. If this step, $t = 1$, does not satisfy the Armijo condition (3.1.1), then the step is contracted, so the final step is less than one. On the other hand, if $t = 1$ satisfies (3.1.1), then the line search checks whether the Wolfe condition (3.1.2) is satisfied too. If it is, then the line search quits; if not, the step is doubled and hence the line search next checks whether $t = 2$ satisfies (3.1.2). At the initial iteration ($k = 0$), several doublings might be needed before (3.1.2) is eventually satisfied. But for subsequent steps ($k \geqslant 1$), we know that $t = 2$ must satisfy the Wolfe condition, so the final step must satisfy $t_k = 2$ (if $t = 2$ satisfies (3.1.1)) or $t_k < 2$ (otherwise). Thus, for $k \geqslant 1$ we always have $t_k \leqslant 2$.* ∎

Now we can present the main result of this subsection: using a line search with the property just described, the optimization method fails.

Theorem 3.2.10 *If scaled memoryless BFGS is applied to (1.0.2), using an Armijo-Wolfe line search that satisfies $t_k \leqslant 2$ for $k \geqslant 1$, such as Algorithm 1 in §2.2 then the method fails in the sense that $f(x_N)$ is bounded below as $N \to \infty$.*

Proof: Recalling $t_{k+1}d_{k+1}^{(1)} = s_{k+1}^{(1)}$ again, using (3.2.37) and $t_{k+1} \leqslant 2$ we find that

$$|s_{k+1}^{(1)}| \leqslant \frac{a - (n-1)|b_k|}{a}|s_k^{(1)}|. \tag{3.2.41}$$

Let $\epsilon > 0$ satisfy

$$\delta_\epsilon \equiv \frac{a-(n-1)b}{a} + \frac{\epsilon}{a} < 1.$$

Define K as in Lemma 3.2.5, so that (3.2.13) holds, and hence

$$\frac{a-(n-1)|b_k|}{a} < \delta_\epsilon.$$

Applying this inequality to (3.2.41) we get

$$|s_{k+1}^{(1)}| \leqslant \delta_\epsilon |s_k^{(1)}|, \tag{3.2.42}$$

and since $\delta_\epsilon < 1$ we have

$$\sum_{k=K}^{N-1} |s_k^{(1)}| < |s_K^{(1)}| \frac{1}{1-\delta_\epsilon}. \tag{3.2.43}$$

By substituting this into (3.2.23) we get

$$f(x_K) - f(x_N) < a|x_K^{(1)}| + |s_K^{(1)}| \frac{(n-1)\,b+\epsilon}{1-\delta_\epsilon},$$

which shows $f(x_N)$ is bounded below. ∎

Finally, we have the following corollary to Theorems 3.2.7 and 3.2.10. Recall that γ_k is the scale factor (see (3.1.21)).

Corollary 3.2.11 *If the assumptions required by either Theorem 3.2.7 or 3.2.10 hold, then*

$$\lim_{N\to\infty} \gamma_N = 0 \tag{3.2.44}$$

and x_N converges to a non-optimal point $\bar{x}$ such that

$$\bar{x} = [0, \bar{x}^{(2)}, \ldots, \bar{x}^{(n)}]^T. \tag{3.2.45}$$

Proof: It is immediate from (3.2.32) or (3.2.43) that $|s_N^{(1)}| \to 0$ as $N \to \infty$, so from (3.1.21), we conclude (3.2.44). Also due to (3.1.11) we have $|x_N^{(1)}| \to 0$, and since $f(x_N) = a|x_N^{(1)}| + \sum_{i=2}^{n-1} x_N^{(i)}$ is bounded below, so is $\sum_{i=2}^{n-1} x_N^{(i)}$. Due to (3.2.14) and (3.1.26), we have $d_{N-1}^{(i)} < 0$, for $i = 2, 3, \ldots, n$, so $t_{N-1} d_{N-1}^{(i)} = x_N^{(i)} - x_{N-1}^{(i)} < 0$, and therefore $x_N^{(i)}$ is strictly decreasing as $N \to \infty$. Hence, $x_N^{(i)}$ converges to a limit $\bar{x}^{(i)}$. ■

Due to the symmetry we discussed earlier, the total decrease along each component, $x_0^{(i)} - \bar{x}^{(i)} = \sum_{k=0}^{N} s_k^{(i)}$, is the same for $i = 2, 3, \ldots, n$.

Finally, note that it follows from Corollary 3.2.11 together with (3.1.20) that, when the assumptions hold, the matrix H_N converges to zero. In contrast, when full BFGS is applied to the same problem, it is typically the case that a direction is identified along which f is unbounded below within a few iterations, and that at the final iterate, one eigenvalue of the inverse Hessian is much smaller than the others, with its corresponding eigenvector close to the first coordinate vector, along which f is nonsmooth.

3.3 Experiments

Our experiments were conducted using the BFGS / L-BFGS MATLAB code in HANSO. This uses the Armijo-Wolfe bracketing line search given in Algorithm 1 in §2.2. Consequently, according to the results of §3.2.3, scaled memoryless BFGS

(L-BFGS with $m = 1$) should fail on function (1.0.2) when a satisfies Assumption 1: $2\sqrt{n-1} \leqslant a$. This is illustrated in Figure 3.2, which shows an experiment where we set $a = 3$ and $n = 2$ and ran scaled memoryless BFGS, the gradient method, and full BFGS, starting from the same randomly generated initial point. We see that scaled memoryless BFGS fails, in the sense that it converges to a non-optimal point, while the gradient method succeeds, in the sense that it generates iterates with $f(x_k) \downarrow -\infty$. In contrast to both, full BFGS succeeds in the sense that it finds a direction along which f is unbounded below in just five iterations. These three different outcomes respectively illustrate the three different ways that the HANSO code terminated in our experiments: (i) convergence to a non-optimal point, which is detected when the steplength upper bound β in Algorithm 1 in §2.2 converges to zero indicating that Armijo-Wolfe points exist, but the line search terminates without finding one due to rounding errors; (ii) divergence of the $f(x_k)$ to $-\infty$ although the line search always finds Armijo-Wolfe steplengths; and (iii) generation of a direction along which f is apparently unbounded below, which is detected when β in Algorithm 1 in §2.2 remains equal to its initial value of ∞ while the lower bound α is repeatedly doubled until a limit is exceeded.[1] In the results reported below for function (1.0.2), termination (i) is considered a failure while terminations (ii) and (iii) are considered successes. We note that, provided $\sqrt{n-1} \leqslant a$, the gradient method can never result in termination (iii), and whether it results in termination (i) or (ii) depends on the Armijo parameter as was shown in Chapter 2. In our experiments, L-BFGS, with or without scaling and with one or more updates, always resulted in termination (i) or (iii), while full BFGS invariably resulted in

[1] Although in principle the code would alternatively terminate if a termination tolerance was met or an upper bound on the number of iterations was exceeded, we set these so small and large respectively that they virtually never caused termination.

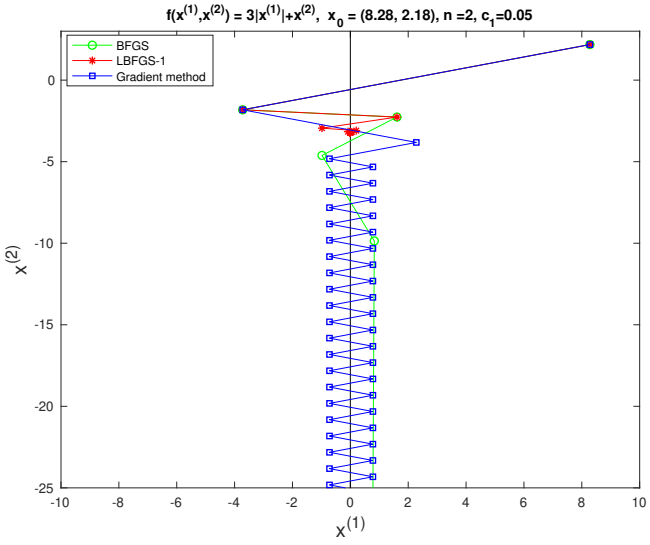


Figure 3.2: Full BFGS (green circles), scaled memoryless BFGS (red asterisks) and the gradient method (blue squares) applied to the function (1.0.2) defined by $a = 3$ and $n = 2$. Scaled memoryless BFGS fails while full BFGS and the gradient method succeed.

termination (iii) (as we know it must from the results in [Xie & Waechter, 2017]).

Although the proof of Theorem 3.2.1 does require Assumption 1 we observed that $\sqrt{3(n-1)} \leqslant a$ suffices for $\{|b_k|\}$ and consequently $|d_k|/\|d_k\|_2$ to converge. In Figure 3.3 we repeat the same experiment with $a = \sqrt{3}$ and $n = 2$, showing that scaled memoryless BFGS still fails. In this case, as noted in Section 3.2, the normalized direction is the same as the normalized direction generated by the gradient method, but unlike in the gradient method, the magnitude of the directions d_k converge to zero so scaled memoryless BFGS fails.

However, if we set a to $\sqrt{3} - 0.001$ the method succeeds. This is demonstrated in Figure 3.4: observe that although one at first has the impression that x_k is converging to a non-optimal point, a search direction is generated on which f is unbounded below "at the last minute".

Extensive additional experiments verify that the condition $\sqrt{3(n-1)} \leqslant a$, as

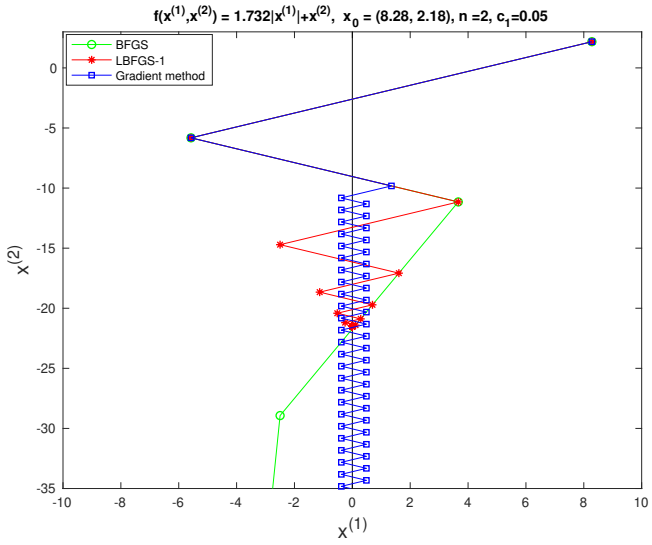


Figure 3.3: Full BFGS (green circles), scaled memoryless BFGS (red asterisks) and the gradient method (blue squares) applied to the function (1.0.2) defined by $a = \sqrt{3}$ and $n = 2$. Scaled memoryless BFGS fails while full BFGS and the gradient method succeed.

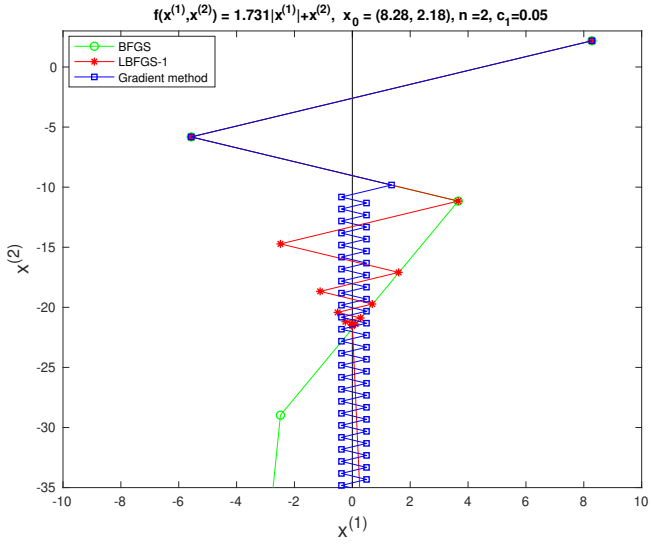


Figure 3.4: Full BFGS (green circles), scaled memoryless BFGS (red asterisks) and the gradient method (blue squares) applied to the function (1.0.2) defined by $a = \sqrt{3} - 0.001$ and $n = 2$. All methods succeed.

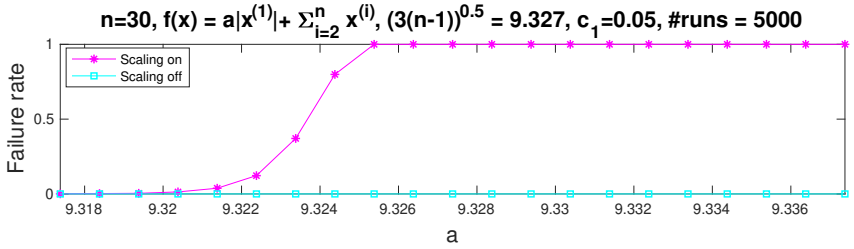


Figure 3.5: The failure rate of memoryless BFGS with scaling (magenta asterisks) and without scaling (cyan squares) applied to function (1.0.2) with $n = 30$ and 21 different values of a, initiating the method from 5000 random points. With scaling, the failure rate is 1 for $9.327 \leqslant a$. Without scaling, the failure rate is 0 regardless of a.

opposed to Assumption 1, is sufficient for failure, as illustrated by the magenta asterisks in Figure 3.5. Starting from 5000 random points generated from the normal distribution, we called scaled memoryless BFGS to minimize function (1.0.2) with $n = 30$ and for values of a ranging from 9.317 to 9.337, since for $n = 30$, $\sqrt{3(n-1)} \approx 9.327$. We see that for $9.327 \leqslant a$ the failure rate is 1 (100%), while for $9.32 > a$ the failure rate is 0. In comparison to the similar experiment in Figure 2.4 in Chapter 2 for the gradient method, the transition from failure rate 0 to 1 is quite sharp here. This might be explained by the fact that the gradient method fails because the steplength $t_k \to 0$, whereas for scaled memoryless BFGS, t_k does not converge to zero; it is the scale γ_k and consequently the norm of d_k which converges to zero. Hence, rounding error prevents the observation of a sharp transition in the results for the gradient method, as explained in Chapter 2; by comparison, rounding error plays a less significant role in the experiments reported here.

The cyan squares in Figure 3.5 show the results from the same experiment for memoryless BFGS *without* scaling, i.e., with $H_k^0 = I$ instead of (1.0.1), using the same 5000 initial points. In this case, the method is successful regardless of the value of a.

Experiments suggest that the theoretical results we presented for scaled L-BFGS

with only one update might extend, although undoubtedly in a far more complicated form, to any number of updates. In Figure 3.6 we show results of experiments with a variety of choices of m and a, running scaled L-BFGS-m (L-BFGS with m updates) initiated from 1000 randomly generated points for each pair $(m,\ a)$. The horizontal axis shows m, the number of updates, while the vertical axis shows the observed failure rate. We set the Armijo parameter $c_1 = 0.01$ and $n = 4$, so that $\sqrt{3(n-1)} = 3$, and show results for values of a ranging from 2.99 to 300. Figure 3.7 shows results from the same experiment except that $c_1 = 0.001$. The results shown in Figure 3.8 use a different objective function; instead of (1.0.2), we define $f(x) = a|b_1^T x| + b_2^T x$, where b_1 and b_2 were each chosen as a random vector in $\mathbb{R}^{10}$ and normalized to have length one. The Armijo parameter was set to $c_1 = 0.01$. In all of Figures 3.6, 3.7 and 3.8 we observe that as a gets larger for a fixed m, the failure rate increases. On the other hand, as m gets larger for a fixed a, the failure rate decreases. Comparing Figures 3.6 and 3.7, we see that the results do not demonstrate a significant dependence on the Armijo parameter c_1; in particular, as we established in Section 3.2.3, there is no dependence on c_1 when $m = 1$ because we are using the line search in Algorithm 1 in §2.2. However, we do observe small differences for the larger values of m, where the failure rate is slightly higher for the larger Armijo parameter. This is consistent with the theoretical results in §3.2.2 as well as those in Chapter 2 where, if a is relatively large, then to avoid failure c_1 should not be too large.

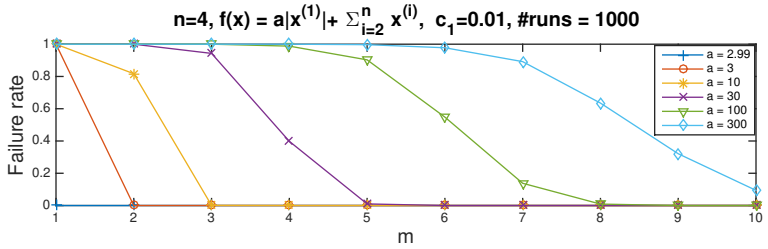


Figure 3.6: The failure rate for each scaled L-BFGS-m, where the number of updates m ranges from 1 to 10, applied to function (1.0.2) with $a = 2.99$ (blue pluses), $a = 3$ (orange circles), $a = 10$ (yellow asterisks), $a = 30$ (purple crosses), $a = 100$ (green triangles) and finally $a = 300$ (cyan diamonds), with $c_1 = 0.01$ and $n = 4$ and hence $\sqrt{3(n-1)} = 3$, and with each experiment initiated from 1000 random points.

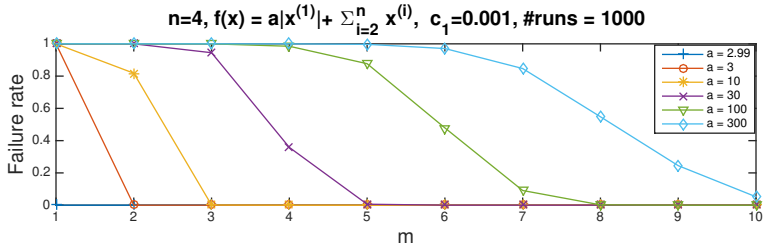


Figure 3.7: The same experiment as in Figure 3.6 except that $c_1 = 0.001$.

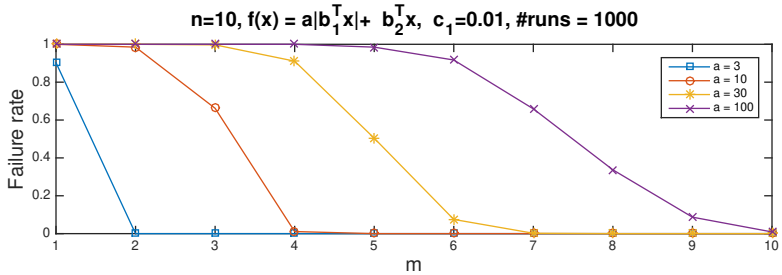


Figure 3.8: The same experiment as in Figure 3.6 except that $f(x) = a|b_1^T x| + b_2^T x$ where $b_1, b_2 \in \mathbb{R}^{10}$ were chosen randomly.

3.4 Concluding Remarks

We have given the first analysis of a variant of L-BFGS applied to a nonsmooth function, showing that the scaled version of memoryless BFGS (L-BFGS with just one update) applied to (1.0.2) generates iterates converging to a non-optimal

point under simple conditions. One of these conditions applies to the method with any Armijo-Wolfe line search and depends on the Armijo parameter. The other condition applies to the method using a standard Armijo-Wolfe bracketing line search and does not depend on the Armijo parameter. Experiments suggest that extended results likely hold for L-BFGS with more than one update, though clearly a generalized analysis would be much more complicated.

We do not know whether L-BFGS without scaling applied to the same function can converge to a non-optimal point, but numerical experiments suggest that this cannot happen. Furthermore, we observed that L-BFGS without scaling obtains significantly more accurate solutions than L-BFGS with scaling when applied to a more general piecewise linear function that is bounded below. In the next chapter, we further investigate whether scaling is generally inadvisable when applying L-BFGS to nonsmooth functions, despite its apparent advantage for smooth optimization.

Chapter 4

Experiments

In this chapter we present extensive numerical experiments investigating the behavior of L-BFGS on a variety of convex nonsmooth optimization problems, as well as some experiments on *smooth* approximations to nonsmooth problems. L-BFGS is applicable to nonconvex problems, too, but we decided to restrict our investigation to convex functions.

The functions to be minimized in this chapter are all bounded below. Consequently, terminations of kind (ii) and (iii) discussed in § 3.3 are not possible, and the only possible terminations are: (a) convergence to some point, possibly optimal, which is detected when the steplength upper bound β in Algorithm 1 in § 2.2 converges to zero indicating that Armijo-Wolfe points exist, but the line search terminates without finding one due to rounding errors: this termination is called type (i) in Chapter 3; (b) a limit on the number of iterations (or the number of function evaluations) is exceeded (not relevant in Chapter 3 as the limit was set so large that it was never active, but relevant in this chapter) and (c) a termination tolerance was met (not relevant in either Chapter 3 or here because this was set so

small that it was never active). In this chapter, if termination (a) occurs far from an optimal solution, we sometimes refer to this as "break-down".

We emphasize that in these experiments we follow the same philosophy used throughout this thesis: in the presence of floating point arithmetic, it does not make sense to try to determine whether, at a given point x, the function f being minimized is actually differentiable at x. Convex functions (and indeed all locally Lipschitz functions) are differentiable almost everywhere, though frequently not at optimal solutions, and since none of the methods being tested are biased towards generating points where f is actually nonsmooth, except in the limit, computing gradients without first checking for differentiability makes sense. Indeed, it is the huge variation in gradients near points where a function is nonsmooth that provides BFGS with the information it needs to effectively minimize nonsmooth functions: see [Lewis & Overton, 2013] for further discussion of this point. Note also that this means, as noted already in § 2.5, that our implementation of the subgradient method is equivalent to a gradient method with predetermined stepsizes.

4.1 Piecewise-Linear Functions

We first consider piecewise linear functions. In §4.1.1 we consider randomly generated problems, while in §4.1.2 we consider a class of highly ill-conditioned nonsmooth problems devised by Nesterov.

4.1.1 Randomly Generated Problems

Consider the following max function

$$f(x) = \max_{i=1,...p} \{b_i^T x - r_i\}, \tag{4.1.1}$$

where $x \in \mathbb{R}^n$ and $b_1, ..., b_p$ are uniformly randomly generated vectors in $\mathbb{R}^n$ and $r_1, ..., r_p$ are random scalars. These quantities were fixed for the experiment reported here but similar results were obtained for other choices. Note that, under the tacit assumption that f is differentiable at x just discussed, i.e., that the index j attaining the maximum value of $b_i^T x - r_i$ is unique, we have $\nabla f(x) = b_j$. We set $n = 10$ and $p = 50$, obtaining a problem that, unlike those studied in previous chapters, is bounded below. For the experiments in this subsection, the iteration limit was set large enough (5000) that the only form of termination was type (a): convergence to some point, possibly optimal, which is detected when the steplength upper bound β in Algorithm 1 in § 2.2 converges to zero indicating that Armijo-Wolfe points exist, but the line search terminates without finding one due to rounding errors. Consequently, we evaluated how successful the runs were by comparing the final function value to the optimal value f_* that we obtained via linear programming using MOSEK[1] with the tolerance set to 10^{-14}. Figure 4.1 shows the median accuracy obtained by L-BFGS-m, for $m = 1, \ldots, 10$, with and without scaling, over 5000 random starting points independently generated from the standard normal distribution. L-BFGS with scaling does not achieve a median accuracy better than 10^{-2}, even when $m = 10$. Without scaling, the accuracy of the results improves substantially, to a median accuracy of about 10^{-9} with $m = 9$. Strangely, for this

[1] https://www.mosek.com/

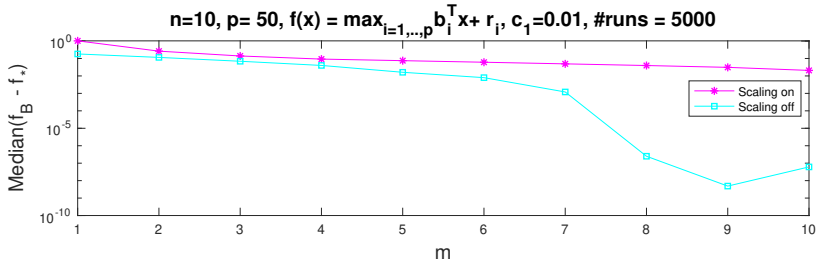


Figure 4.1: Median accuracy of the solution f_B found by L-BFGS-m with $m = 1, \ldots, 10$ for the piecewise linear function defined in (4.1.1), with $n = 10$ and $p = 50$, compared with the value f_* obtained from the linear optimizer in MOSEK using high accuracy. Scaled L-BFGS-m does not obtain accurate solutions even with $m = 10$. In contrast, with scaling off, L-BFGS-9 obtains a median accuracy of about 10^{-9}.

problem, and many different instances of it that we tried, L-BFGS-10 performs worse than L-BFGS-9. The median accuracy of the solution found by full BFGS (with or without scaling the initial inverse Hessian approximation) is significantly better: about 10^{-14}.

4.1.2 An Ill-conditioned Problem from Nesterov

In this section, we revisit Nesterov's ill-conditioned Les-Houches problem that we introduced in § 2.5. In that section, our focus was on the behavior of the gradient method with different choices of the Armijo parameter, but we also included a comparison of the L-BFGS-5 and L-BFGS-10 methods with scaling, as well as the full BFGS method and the subgradient method with prescribed steplength $t_k = 1/k$. There, Figure 2.6 showed results for all these methods on the Les-Houches problem with $n = 100$, displaying all function values that were computed, including those generated in the line search. Here, in Figure 4.2, we show results for L-BFGS-1 (memoryless BFGS), with and without scaling, as well as full BFGS and the subgradient method with $t_k = 1/k$, for the Les-Houches problem with $n = 500$. Again, we display all the function values that were computed. We put a limit of

10,000 function evaluations for each method. The starting point x_0 (used by all the methods) was drawn randomly from the ball of radius 0.1 centered at the vector of all ones, using the normal distribution.

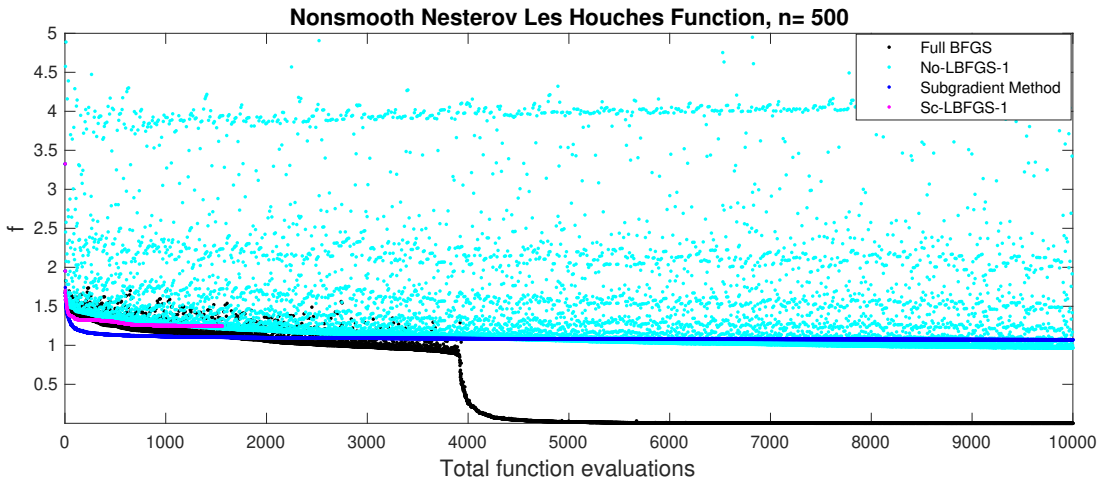


Figure 4.2: Comparing BFGS, L-BFGS-1 with scaling on and off and the subgradient method on the ill-conditioned Les-Houches problem (2.5.1) with $n = 500$.

We see from Figure 4.2 that scaled L-BFGS-1 (magenta dots) breaks down, with failure in the line search (termination (a)) after fewer than 2000 function evaluations. In contrast, unscaled L-BFGS-1 (cyan) runs for the full 10,000 function evaluations. However its scattered plot indicates that the method performs many function evaluations per iteration (in the line search), indicating that, perhaps not surprisingly given its name, the search directions it generates are not well scaled. Despite this, the method obtains a somewhat lower answer than the subgradient method (dark blue). Full BFGS (black) performs much better than any of the other methods, reducing the function value to about 10^{-5} (recall that the optimal value is zero). It is interesting to note that its convergence rate picks up rapidly right after it has lowered the function value down to 1. We do not know the reason for this.

We now increase the number of updates m from 1 to 20 and repeat this experiment: see the top plot in Figure 4.3. Unlike scaled L-BFGS-1, scaled L-BFGS-20 does not quit early, and furthermore it also demonstrates a suddenly faster convergence rate toward the end of the experiment similar to that of full BFGS (although the final answer it obtains is not nearly as accurate as full BFGS). It obtains a function value of size 0.47, whereas unscaled L-BFGS-20 gets a final answer of about 0.998 and the subgradient method obtains 1.083. This experiment shows that increasing the number of updates enhanced the performance of scaled L-BFGS far more than it did for unscaled L-BFGS.

The middle panel of Figure 4.3 shows the same plot as in the top panel, except in log scale, making it easier to see how the fast phase of scaled L-BFGS compares to that of full BFGS.

The bottom panel is a zoomed-in view of the middle plot, focused at the point where scaled L-BFGS is transitioning to its fast convergence phase. What seems interesting is that before this point, scaled L-BFGS behaved similarly to the subgradient method, performing almost exactly one function evaluation per iteration, and hence, using a stepsize of one (see the Armijo-Wolfe bracketing line-search Algorithm 1 in § 2.2). Eventually, a stepsize of one no longer satisfies the Armijo-Wolfe conditions and therefore the line search requires multiple function evaluations before it finds a satisfactory step. In general, in its fast phase, scaled L-BFGS-20 performs more than one function evaluation per iteration, although not as many as full BFGS does.

The conclusion from both experiments is that with a small m, unscaled L-BFGS-m performs better and with a larger m it is the scaled variant which performs better. However, neither method performs nearly as well as full BFGS.

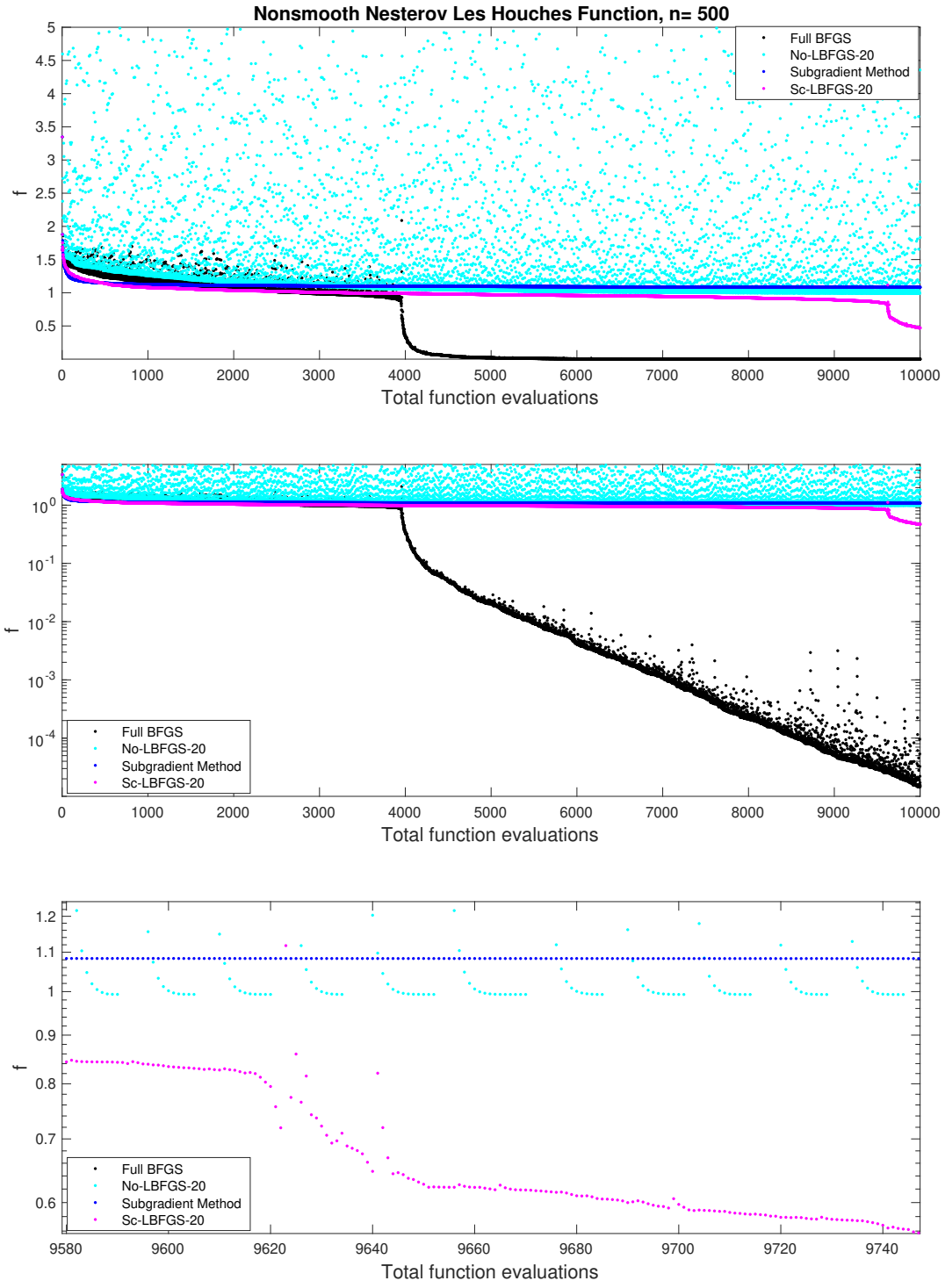


Figure 4.3: Top: Comparing BFGS, L-BFGS-20 with scaling on and off and the subgradient method on the ill-conditioned Les-Houches problem given in (2.5.1) with $n = 500$. **Middle**: The same plot as in the top except in semi-log-y scale. **Bottom**: The same plot as in the middle, when zoomed-in around the point where scaled L-BFGS transitions to a faster convergence rate.

4.1.3 Smoothed Versions of Nesterov's Ill-conditioned Problem

We revisit function (2.5.1). Suppose $A \in \mathbb{R}^{n\times n}$ with

$$A = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -2 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -2 & 1 \end{bmatrix}.$$

Let $y = Ax$. Then, (2.5.1) is equivalent to the following vector-max function $g : \mathbb{R}^n \to \mathbb{R}$

$$g(y) = \max_i \{y_i\ , -y_i\} \quad \text{for} \quad i = 1, 2, ..., n, \tag{4.1.2}$$

that is

$$f(x) = g(Ax). \tag{4.1.3}$$

Consider the Nesterov smoothing [Nesterov, 2005] of the vector-max problem (4.1.2):

$$g_\mu(y) = \mu \log \sum_{i=1}^{n} \left(e^{y_i/\mu} + e^{-y_i/\mu}\right) - \mu \log(2n). \tag{4.1.4}$$

Without the constant term $-\mu \log(2n)$, this function is also known as the softmax function [Boyd & Vandenberghe, 2004]. The unique minimizer of $g_\mu(y)$ is $y^*_\mu = 0$ with $g^*_\mu = 0$. Since A is full-rank (although one of its singular values converges to zero as $n \to \infty$), via (4.1.3) we know that the unique minimizer of $f_\mu(x)$, x^*_μ, is also 0 and with the same optimal value $f^*_\mu = 0$.

To see how the conditioning of (4.1.4) is dependent on μ we derive its gradient and Hessian. For brevity let s be the sum inside the log in (4.1.4), i.e., $s =$

$\sum_{i=1}^{n}\left(e^{y_i/\mu}+e^{-y_i/\mu}\right)$. Let σ and $\bar{\sigma}$ respectively be the vectors with elements $\sigma_i = \dfrac{e^{y_i/\mu}}{s}$ and $\bar{\sigma}_i = \dfrac{e^{-y_i/\mu}}{s}$. The partial derivative of (4.1.4) w.r.t. y is

$$\frac{\partial g_\mu}{\partial y_i} = \frac{e^{y_i/\mu} - e^{-y_i/\mu}}{s} = \sigma_i - \bar{\sigma}_i.$$

So we have $\nabla_y g_\mu = \sigma - \bar{\sigma}$. Note that at $y = 0$, we get $\sigma = \bar{\sigma}$, so $\nabla g_\mu(0) = 0$. The second derivative $\nabla_y^2 g_\mu$ is

$$\nabla_y^2 g_\mu = \frac{1}{\mu}\Big(\mathbf{Diag}\,(\sigma + \bar{\sigma}) - (\sigma - \bar{\sigma})(\sigma - \bar{\sigma})^T\Big). \tag{4.1.5}$$

Since $(\sigma - \bar{\sigma})(\sigma - \bar{\sigma})^T$ is a rank-one positive semidefinite matrix we have

$$\nabla_y^2 g_\mu \preccurlyeq \frac{1}{\mu}\mathbf{Diag}\,(\sigma + \bar{\sigma})\,. \tag{4.1.6}$$

At $y = 0$, this inequality holds with equality. In fact, it is not hard to see that at $y = 0$, since $s = 2n$ we have $\nabla_y^2 g_\mu(0) = \dfrac{1}{n\mu} I$.

The diagonal of the matrix in the right hand side in (4.1.6) sums up to one: $\sum_{i=1}^{n}(\sigma_i + \bar{\sigma}_i) = 1$, so its largest eigenvalue is at most one, i.e.,

$$\|\nabla_y^2 g_\mu\| \leqslant \frac{1}{\mu}. \tag{4.1.7}$$

Turning to the variation of f_μ w.r.t. the original variable x, the gradient is $\nabla_x f_\mu = A^T \nabla_y g_\mu$, and the Hessian is

$$\nabla_x^2 f_\mu = A^T \nabla_y^2 g_\mu A, \tag{4.1.8}$$

so applying (4.1.7) to the right hand side we get

$$\|\nabla_x^2 f_\mu\| \leqslant \frac{1}{\mu}\|A\|^2.$$

Hence, the Lipschitz constant of the gradient $\nabla_x f_\mu$ is $L = \frac{1}{\mu}\|A\|^2$. From applying the triangle inequality to $\|A\|$ we know that $\|A\| \leqslant 3$, so we get $L \leqslant \frac{9}{\mu}$. Note that although the bound on the norm of the Hessian is unbounded as $\mu \to 0$, the norm is only large near points where the original function f is nonsmooth. For example, let x_0 be the same randomly generated starting point that was used in the previous subsection § 4.1.2.

We find that the norm of $\nabla_x^2 f_\mu(x_0)$ decays exponentially as $\mu \to 0$, with $\|\nabla^2 f_\mu(x_0)\| \approx 10^{-7}$ even for $\mu = 10^{-2}$. This phenomenon should not be surprising. For example, very ill-conditioned smooth approximations to the absolute value function have a large second derivative at zero but a nearly zero second derivative everywhere else.

If we follow the standard approach [Boyd & Vandenberghe, 2004, Sec 9.1] to defining the condition number of the strongly convex function f_μ as

$$\kappa(f_\mu) = \left(\max_{x \in S} \|\nabla^2 f_\mu(x)\|\right)\left(\max_{x \in S} \|(\nabla^2 f_\mu(x))^{-1}\|\right) \tag{4.1.9}$$

where $S = \{x : f(x) \leqslant f(x_0)\}$, then we find

$$\kappa(f_\mu) \gg \frac{1}{\mu}$$

as $\mu \to 0$ since the first factor in (4.1.9) is at least $\|\nabla^2 f_\mu(0)\| = \|A\|^2/\mu$, while the second factor is enormous as all eigenvalues of $\nabla^2 f_\mu(x_0)$ are tiny for $\mu \leqslant 10^{-2}$.

We now report on experiments we conducted applying full BFGS and L-BFGS, with and without scaling, to the smoothed function (4.1.4), with $n = 500$ as before. We did not include the subgradient method in the comparison. The top panel of Figure 4.4 shows the final function value computed by full BFGS and L-BFGS-1 with and without scaling as a function of the smoothing parameter μ using a log-log scale. Let us focus first on the results for full BFGS (black circles).

BFGS always finds a solution with magnitude smaller than 10^{-15}, even for a very small μ, when the function is extremely ill conditioned. This is a remarkable property of full BFGS: its accuracy does not deteriorate significantly[2] as the condition number $\kappa(f_\mu)$ of the smoothed problem blows up with $\mu \to 0$. In fact, when μ is sufficiently small, say $\mu = 10^{-16}$ (which is approximately the rounding unit in IEEE double precision used by MATLAB).

The smoothed problem is precisely equivalent to the original nonsmooth problem when rounding errors are taken into account, so the top panel shows the remarkable transition of the accuracy of full BFGS from smoothed variants of the problem to the limiting nonsmooth problem. The bottom panel shows the number of iterations that were required, again as a function of the smoothing parameter μ and again using a log-log scale. The maximum number of iterations (not function evaluations) was set to 10^4 for each μ. Remarkably, we see that the number of iterations required for full BFGS to accurately minimize f_μ does not significantly increase as $\mu \to 0$, even though the condition number $\kappa(f_\mu)$ blows up as μ decreases to zero, and the number required for the effectively nonsmooth instance $\mu = 10^{-16}$ is not much more than the number required for much better conditioned smoothed problems arising

[2]Surprisingly, the accuracy increases somewhat as μ decreases, but this is at the level of rounding errors and could perhaps be explained by a rounding error analysis. Certainly the scatter at the bottom left corner of the plot is a consequence of rounding error.

from moderate values of μ.

The results for L-BFGS-1 are very different. Unscaled L-BFGS-1 (cyan squares) finds an accurate answer for $\mu \geqslant 10^{-3}$, but the number of iterations required increases rapidly as μ is decreased so the maximum number allowed is reached for μ ranging from 10^{-4} to 10^{-9}. However, starting with $\mu = 10^{-10}$, unscaled L-BFGS-1 breaks down before reaching the maximum number of iterations allowed. The behavior of scaled L-BFGS-1 (magenta asterisks) is similar except that it breaks down only for $\mu = 10^{-16}$. Note that in this subsection, since we limit the number of iterations, not the number of function evaluations, the performance of unscaled L-BFGS-1 looks better than it really is: the scaled version is computing substantially fewer function evaluations per line search.

When we increase the number of updates to $m = 20$, scaled L-BFGS reacts much better than unscaled L-BFGS; see the top panel in Figure 4.5. Unscaled L-BFGS-20 finds accurate answers for $\mu \geqslant 10^{-4}$ before hitting the iteration limit, while the scaled version does so for $\mu \geqslant 10^{-5}$. Furthermore, when the maximum iteration limit is reached, scaled L-BFGS-20 achieves an answer of magnitude $\approx 10^{-2}$, whereas unscaled L-BFGS-20 is still giving an answer of magnitude $\approx 10^{0}$, similar to unscaled L-BFGS-1. But overall, both are still doing poorly compared to full BFGS.

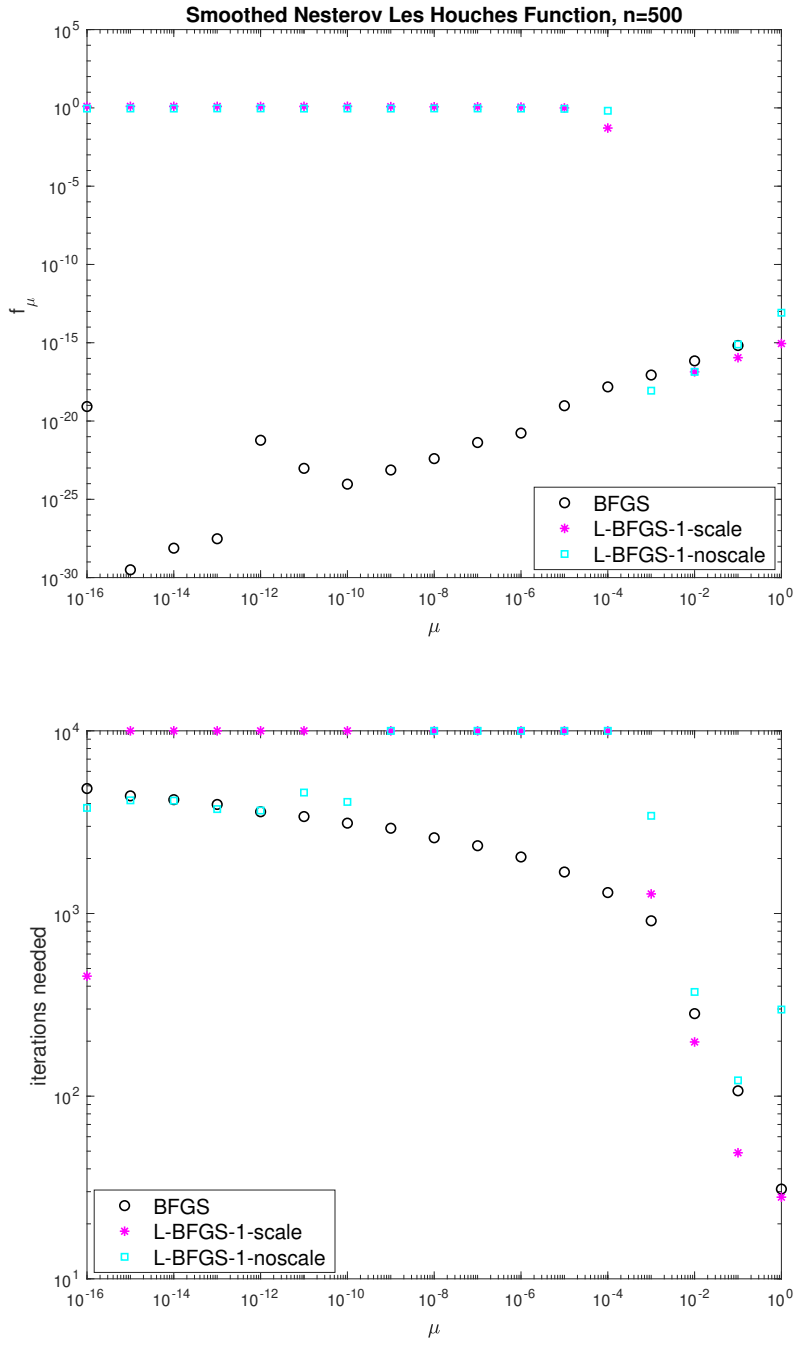


Figure 4.4: Comparing BFGS and L-BFGS-1 with scaling on and off on the smoothed Nesterov-Les-Houches (4.1.4) for $n = 500$. The top panel shows the final function value and the bottom panel shows the iteration count, both as a function of the smoothing parameter μ. The maximum number of iterations is set to 10^4.

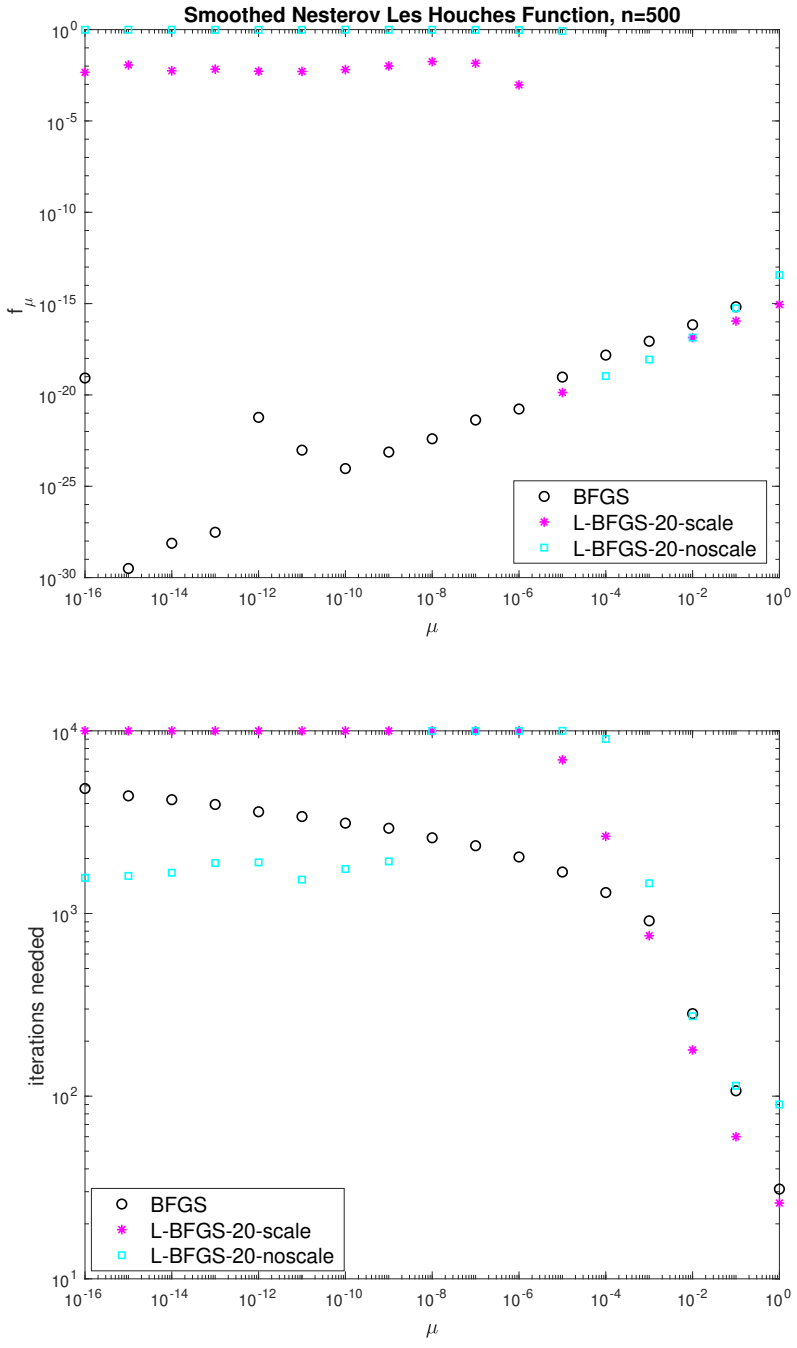


Figure 4.5: Comparing BFGS and L-BFGS-20 with scaling on and off on the smoothed Nesterov-Les-Houches problem (4.1.4) for $n = 500$. The top panel shows the final function value and the bottom panel shows the iteration count, both as a function of the smoothing parameter μ. The maximum number of iterations is set to 10^4.

Finally, we performed experiments for a much smaller variant of the problem, with $n = 5$, and with the maximum iteration limit set to 10^6, so we could see how well L-BFGS-1 performs when it terminates without reaching the iteration limit. Scaled L-BFGS-1 now performs remarkably well, finding a good approximation to the optimal solution for $\mu \geqslant 10^{-14}$, with the required number of iterations increasing log-linearly as μ is decreased. Interestingly though, and in contrast to full BFGS, it breaks down in the nonsmooth limit. On the other hand, unscaled L-BFGS-1 performs poorly, finding an accurate solution only for $\mu \geqslant 10^{-7}$.

Our conclusions from this subsection are consistent with the generally accepted wisdom concerning L-BFGS. For smooth problems, even very ill-conditioned ones, it is best to use the scaled version of L-BFGS, and choosing the number of updates m to be larger rather than smaller gives better performance, although, in contrast to full BFGS, the number of iterations required increases significantly with the conditioning of the problem. Again in contrast with full BFGS, when the ill-conditioning increases to the nonsmooth limit implicit in consideration of rounding errors, scaled L-BFGS generally fails to converge to an optimal solution. For the smooth but ill-conditioned problems considered in this subsection, unlike the nonsmooth problems considered in Chapter 3 and in Figures 4.1 and 4.2 in the current chapter, unscaled L-BFGS offers no advantage compared to scaled L-BFGS. The most important conclusion is that, while applying full BFGS directly to nonsmooth problems works remarkably well, this is not the case for L-BFGS; at least for the Les-Houches problem, it is far preferable to apply scaled L-BFGS to a smoothed approximation to the nonsmooth problem.

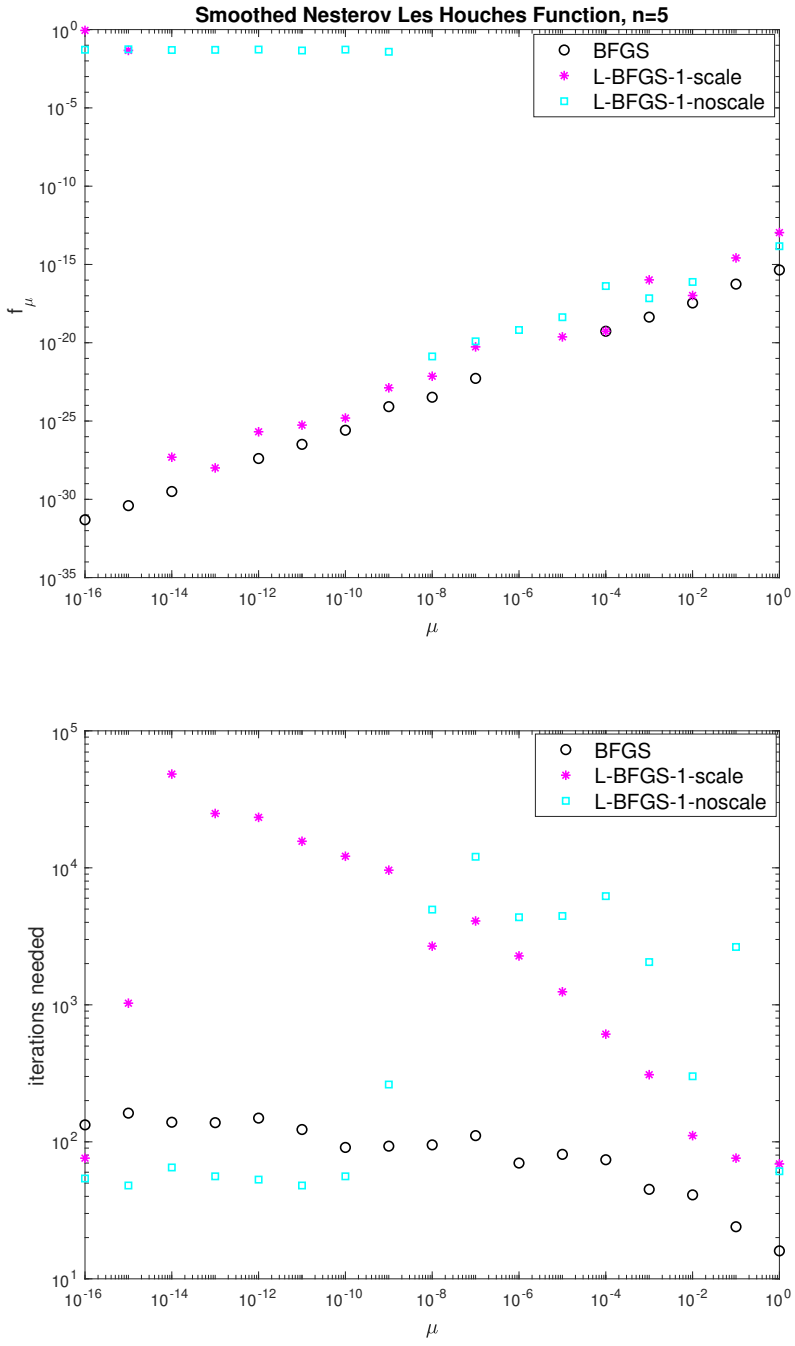


Figure 4.6: Comparing BFGS and L-BFGS-1 with scaling on and off on the smoothed Nesterov-Les-Houches problem (4.1.4) for $n = 5$. The top panel shows the final function value and the bottom panel shows the iteration count, both as a function of the smoothing parameter μ. Maximum number of iterations is set to 10^6.

4.2 Eigenvalue Optimization and Semidefinite Programming

In this section we consider two closely related problems: eigenvalue optimization and semidefinite programming. We begin with the first.

4.2.1 Max Eigenvalue Problem

Let S^N denote the space of $N \times N$ real symmetric matrices and let $\mathcal{A} : S^N \to \mathbb{R}^n$ denote a linear operator acting on X as follows:

$$\mathcal{A}X = \begin{bmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_n, X \rangle \end{bmatrix}, \tag{4.2.1}$$

with $A_i \in S^N$ for $i = 1., \ldots, n$. Its adjoint operator; $\mathcal{A}^T : \mathbb{R}^n \to S^N$, is defined by

$$\mathcal{A}^T y = \sum_{i=1}^{n} y_i A_i. \tag{4.2.2}$$

The Max Eigenvalue problem is to minimize the function

$$f(y) = \lambda_{\max}(C - \mathcal{A}^T y), \tag{4.2.3}$$

where $C \in S^N$ and $\lambda_{\max} : S^N \to \mathbb{R}$ denotes largest eigenvalue of its argument. It is well known that $\lambda_{\max}$ is a convex function on S^N . Early papers on Eigenvalue Optimization include [Overton, 1988].

Assuming the maximum eigenvalue of $C - \mathcal{A}^T y$ is simple, the gradient of f is

$$\nabla f(y) = -\mathcal{A}(qq^T) = -[q^T A_1 q, \cdots, q^T A_n q]^T,$$

where q is a normalized eigenvector corresponding to $\lambda_{\max}(C - \mathcal{A}^T y)$. As earlier in the thesis, our philosophy in implementing algorithms for nonsmooth optimization is not to attempt to determine whether f is differentiable at a given point, which is essentially impossible in the presence of rounding errors, as already explained in the beginning of Chapter 4, so we will tacitly assume that f is differentiable at algorithm iterates. However, at optimal solutions, we generally expect that $C - \mathcal{A}^T y$ has a multiple largest eigenvalue and hence f is not differentiable. As is well known, eigenvalue optimization problems are instances of semidefinite programs, and hence small problems can be solved using CVX [Grant & Boyd, 2014],[Grant & Boyd, 2008].

Using the standard normal distribution, we generated a random instance of this problem, defining $\mathcal{A}$ and C with $N = 50$ and $n = 49$. Figure 4.7 shows the performance of full BFGS, L-BFGS-1 with and without scaling, and the subgradient method (as before with $t_k = 1/k$) for minimizing (4.2.3). All methods are terminated after 10^4 function evaluations. Each function evaluation $f(y)$ requires a call to the MATLAB function `eig` to compute all the eigenvalues of $C - \mathcal{A}^T y$.

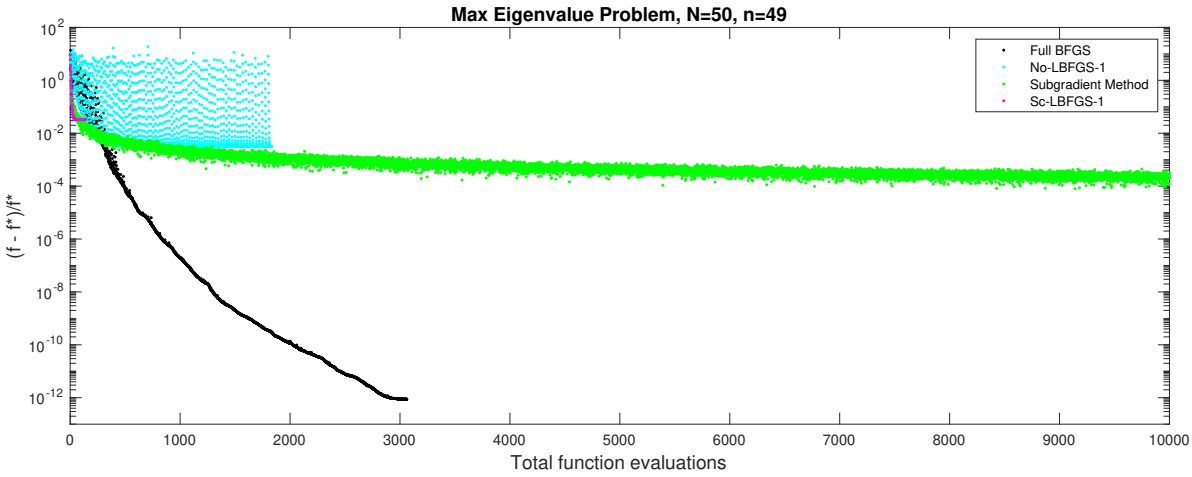


Figure 4.7: Comparing BFGS, L-BFGS-1 with scaling on and off and the subgradient method on a randomly generated Max Eigenvalue problem (4.2.3) with $N = 50$ and $n = 49$. All of the methods are terminated after 10^4 function evaluations.

The vertical axis shows the final value of the relative error $|(f - f^*)/f^*|$, where we used the SDPT3 solver in CVX to obtain the optimal solution f^* with accuracy 10^{-14}. As in the experiment on the nonsmooth Les-Houches problem reported in Figure 4.2, scaled L-BFGS-1 (magenta dots) breaks down early. Unlike in that experiment, however, here unscaled L-BFGS-1 (cyan dots) also breaks down early, and as a result the subgradient method (green dots) obtains a better answer, though not nearly as good as full BFGS (black dots).

It's also of interest to examine the multiplicity of the eigenvalues of $C - \mathcal{A}^T y$ at the optimal solution y^* and its computed approximations. From SDPT3, we know that the optimal multiplicity for this problem is 5. Table 4.1 shows the top 6 eigenvalues of the final answer found by each method. Besides SDPT3, only BFGS and the subgradient method are able to determine the correct optimal multiplicity. BFGS finds a solution with 11 correct digits and the subgradient method one with 3 correct digits. Both variants of L-BFGS-1 converge to answers with multiplicity

4. Although the multiplicity is wrong, the unscaled L-BFGS-1 gets a better answer than its scaled counterpart, with 2 correct digits, albeit in more CPU time.

SDPT3	BFGS	Sc L-BFGS-1	No L-BFGS-1	subgradient
7.82702970305352	7.82702970306035	8.08455876518360	7.85155000878711	7.82953885641783
7.82702970305349	7.82702970306035	8.08455876518359	7.85155000044960	7.82746561454846
7.82702970305348	7.82702970306034	8.08197715145863	7.85154996050940	7.82673229360627
7.82702970305346	7.82702970306031	8.05541534062475	7.85141043900471	7.82472408900949
7.82702970305334	7.82702970306017	7.84362627205676	7.69075549655739	7.82286893229481
7.70350538019538	7.70350432059448	7.56258926523925	7.48734455288558	7.70188538367848

Table 4.1: Top 6 eigenvalues of $C-\mathcal{A}^T y$ for Max Eigenvalue problem (4.2.3) where y is computed by SDPT3, full BFGS, scaled/unscaled L-BFGS-1 and the subgradient method for a randomly generated problem with $N = 50$ and $n = 49$. The optimal multiplicity is 5. The maximum number of function evaluations is set to 10^4.

Next, we repeat this experiment on the same problem, increasing the number of L-BFGS updates from $m = 1$ to $m = 20$. See Figure 4.8 as well as Table 4.2 which presents the top 6 eigenvalues for the final answer obtained by scaled and unscaled L-BFGS-20. Both methods find the right multiplicity, with scaled L-BFGS-20 obtaining 3 correct digits and unscaled L-BFGS-20 obtaining 4 correct digits.

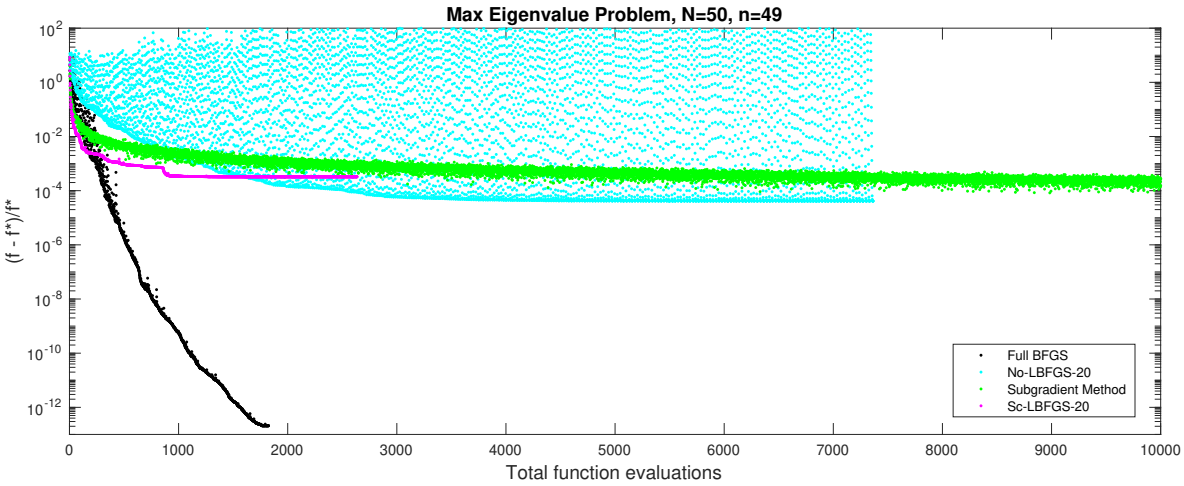


Figure 4.8: Comparing BFGS, L-BFGS-20 with scaling on and off and subgradient method on a randomly generated Max Eigenvalue problem (4.2.3) with $N = 50$ and $n = 49$. All of the methods are terminated after 10^4 function evaluations.

Sc L-BFGS-20	No L-BFGS-20
7.82959659952176	7.82735384547039
7.82959659952176	7.82735380191247
7.82959659952174	7.82735377961470
7.82959659952166	7.82735377236995
7.82959659950328	7.82735372682267
7.62982269438813	7.69181664817458

Table 4.2: Top 6 eigenvalues of $C - \mathcal{A}^T y$ for Max Eigenvalue problem (4.2.3) where y is computed by scaled and unscaled L-BFGS-20 for the same problem reported in Table 4.1. The optimal multiplicity is 5.

In summary, we observe that for the Max Eigenvalue problem, unlike the Les-Houches problem, increasing m from 1 to 20 does not result in scaled L-BFGS doing better than unscaled L-BFGS.

4.2.2 Smoothed Max Eigenvalue Problem

Consider now Nesterov smoothing of the Max Eigenvalue problem (4.2.3), taken from [Vandenberghe, 2019]:

$$f_\mu(y) = \mu \log \sum_{i=1}^{N} \exp(\lambda_i(C - \mathcal{A}^T y)/\mu) - \mu \log N, \tag{4.2.4}$$

where $\lambda_1(W) \geqslant \lambda_2(W) \geqslant \ldots \geqslant \lambda_N(W)$ denote the ordered eigenvalues of a symmetric matrix $W \in S^N$. Thus, λ_1 is equivalent to $\lambda_{\max}$. Unlike the Les-Houches problem, where the nonsmooth optimal value is equal to the smoothed optimal value, that is $f^* = f^*_\mu = 0$, for any μ as $\mu \to 0$, the same statement is not true for the Max Eigenvalue problem in general. The Smoothed Max Eigenvalue problem requires a complete eigendecomposition in order to get every eigenvalue for the given matrix, and since CVX does not allow such functions but only those that it knows to be convex such as the maximum eigenvalue function, we could not compute the optimal value f^*_μ from CVX. Instead, we use full BFGS with the max number of iterations set to a large number (10^5 in the following experiments) to minimize (4.2.4) to high accuracy: we denote this computed value by f^B_μ.

In Figure 4.9 we report on an experiment using the smoothed version of the same instance of the randomly generated Max Eigenvalue problem as in the previous part with $n = 49$ and $N = 50$, using L-BFGS-1 to minimize (4.2.4). The top panel shows the final value computed by scaled (magenta asterisks) and unscaled (cyan squares) L-BFGS-1 shifted by f^B_μ (the answer found by full BFGS), as a function of the smoothing parameter μ, in log-log scale. The bottom panel shows the number of iterations as a function of μ, also in log-log scale. The maximum number of iterations is 10^5.

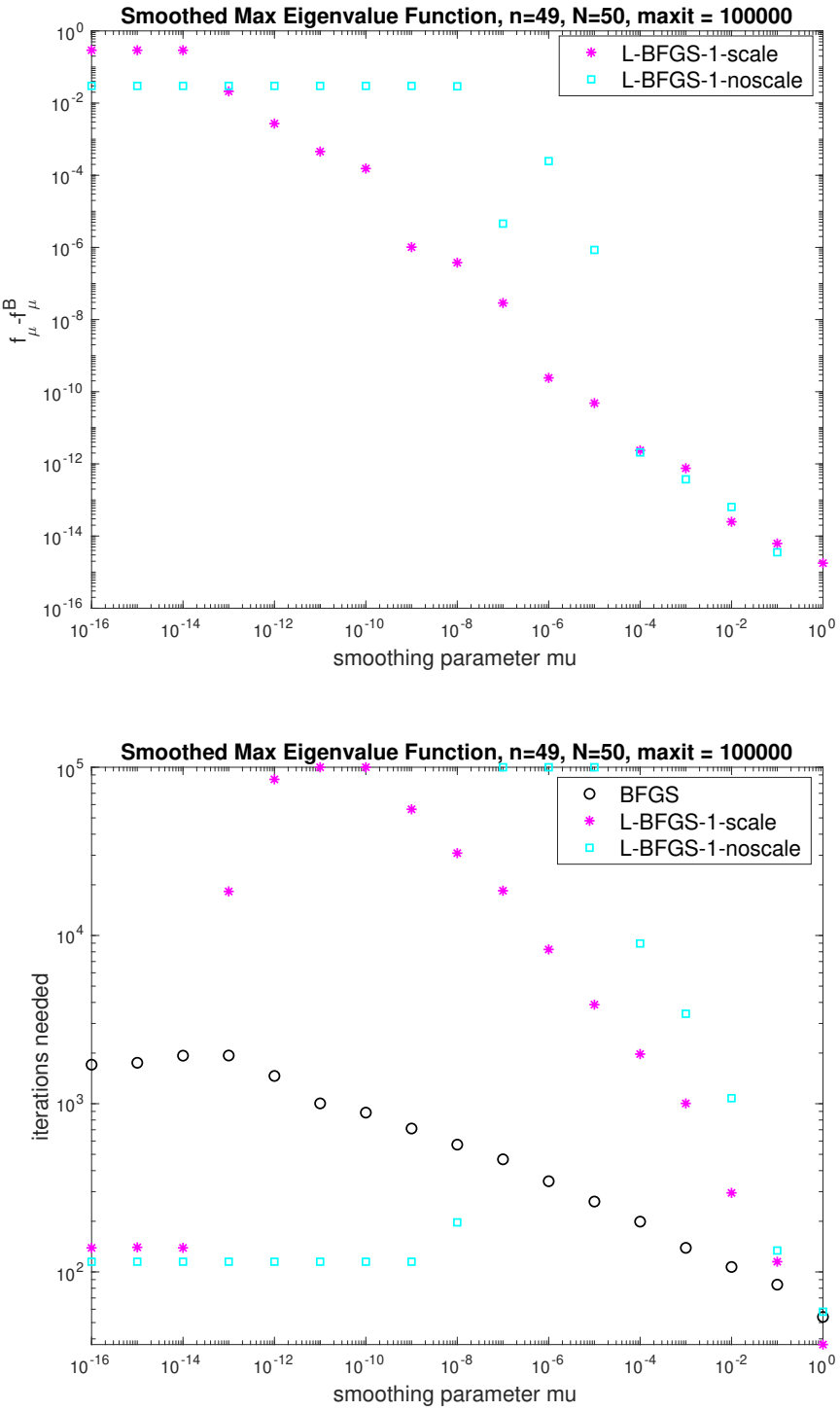


Figure 4.9: Comparing L-BFGS-1 with scaling on and off on the smoothed Max Eigenvalue problem (4.2.4) for $N = 50$ and $n = 49$. The top panel shows the final function value, shifted by f_μ^B, the optimal value computed by BFGS, and the bottom panel shows the iteration count, both as a function of the smoothing parameter μ. The maximum number of iterations is set to 10^5.

In the top panel we see that for $\mu = 1$ down to $\mu = 10^{-4}$ both methods yield about the same accuracy as each other, but that this deteriorates as μ decreases. Scaled L-BFGS-1 continues to obtain a reasonable approximation to the presumed accurate solution f_μ^B for μ down to 10^{-9}, although this accuracy continues to decrease as μ is reduced. Looking at the bottom panel, we see that starting with $\mu = 10^{-10}$ scaled L-BFGS-1 hits the maximum iteration limit and starting with 10^{-12} it breaks down before reaching the maximum iteration limit. In contrast, unscaled L-BFGS-1 hits the maximum iteration limit for $\mu = 10^{-5}$ and breaks down for $\mu \leqslant 10^{-8}$.

Table 4.3 shows the top 6 eigenvalues of the final answer found by BGFS and L-BFGS-1 for the smoothed Max Eigenvalue problem with $\mu = 10^{-7}$. We also repeat the optimal top 6 eigenvalues of the solution to the original nonsmooth function f found by SDPT3 for the sake of comparison. Note that the result computed by applying scaled L-BFGS-1 to the smoothed problem agrees with the nonsmooth optimal value f^* to 8 digits, compared to 0 digits when applied directly to the nonsmooth problem (Table 4.1).

SDPT3	BFGS	Sc L-BFGS-1	No L-BFGS-1
7.82702970305352	7.82702976093363	7.82702978971152	7.82703432112405
7.82702970305349	7.82702968960443	7.82702971836333	7.82703424950540
7.82702970305348	7.82702966768912	7.82702969644540	7.82703422776180
7.82702970305346	7.82702963829977	7.82702966707913	7.82703419808905
7.82702970305334	7.82702954704101	7.82702957585856	7.82703410710019
7.70350538019538	7.70350539089203	7.70374015675041	7.69886385782543

Table 4.3: Top 6 eigenvalues of $C - \mathcal{A}^T y$ for Max Eigenvalue problem where y is computed by applying BFGS, scaled L-BFGS-1 and unscaled L-BFGS-1 to f_μ with $\mu = 10^{-7}$, for the same instance of the randomly generated Max Eigenvalue problem as in Table 4.1. The optimal multiplicity is 5. The first column gives the top 6 eigenvalues of the solution to the original nonsmooth problem.

We repeat this experiment with $m = 20$, reported in Figure 4.10, and as one would hope, we observe that the slope of the magenta line in the top panel in Figure 4.9 decreases, indicating that the loss of accuracy in L-BFGS as a function of μ is less pronounced with 20 updates. Roughly speaking, overall the error decreases by a factor of 10^{-2}. In the bottom panel we see that neither scaled nor unscaled L-BFGS-20 reaches the maximum iteration limit, but that both methods break down for sufficiently small μ. The top eigenvalues produced by L-BFGS-20 for $\mu = 10^{-7}$ are shown in Table 4.4: compared with those obtained by L-BFGS-1, they agree slightly more closely with those found by full BFGS (see Figure 4.3).

In summary, as with the Les Houches problem, it is much more effective to apply L-BFGS to the smoothed max eigenvalue problem than directly to the nonsmooth problem. As earlier, this is in sharp contrast to the behavior of full BFGS.

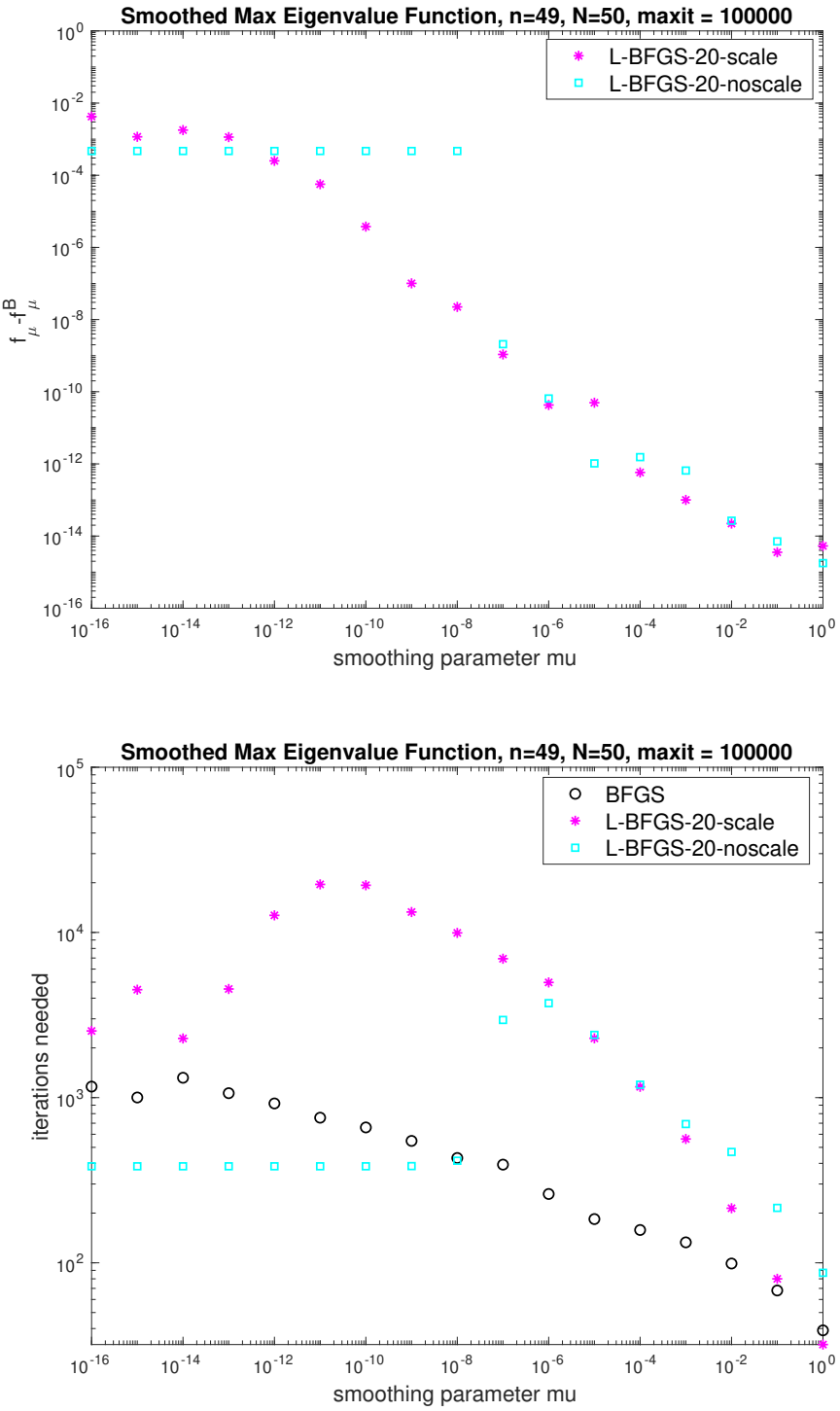


Figure 4.10: Comparing L-BFGS-20 with scaling on and off on the smoothed Max Eigenvalue problem (4.2.4) for $N = 50$ and $n = 49$. The top panel shows the final function value, shifted by f_μ^B, the optimal value computed by BFGS, and the bottom panel shows the iteration count, both as a function of the smoothing parameter μ. The maximum number of iterations is set to 10^5.

Sc L-BFGS-20	No L-BFGS-20
7.82702976201688	7.82702976326908
7.82702969067290	7.82702969273000
7.82702966877706	7.82702966880138
7.82702963938214	7.82702964120678
7.82702954813253	7.82702954355970
7.70348252862186	7.70344821498698

Table 4.4: Top 6 eigenvalues of $C - \mathcal{A}^T y$ for Max Eigenvalue problem where y is computed by applying BFGS, scaled L-BFGS-20 and unscaled L-BFGS-20 to f_μ with $\mu = 10^{-7}$, for the same instance of the randomly generated Max Eigenvalue problem as in Table 4.1. The optimal multiplicity is 5.

4.2.3 Semidefinite Programming

Consider the following primal and dual semidefinite problems (SDP) in standard form [Helmberg & Rendl, 2000, Helmberg et al., 2014]

$$\max_{X \in S^N} \quad \langle C, X \rangle \tag{4.2.5}$$

$$\text{subject to} \quad \mathcal{A}X = b \quad \text{and} \quad X \in S^N_+,$$

$$\min_{y \in \mathbb{R}^n} \quad b^T y \tag{4.2.6}$$

$$\text{subject to} \quad Z = \mathcal{A}^T y - C \quad \text{and} \quad Z \in S^N_+,$$

where $b \in \mathbb{R}^n$, $C \in S^N$ and $\mathcal{A} : S^N \to \mathbb{R}^n$ is a linear operator as defined in (4.2.1),(4.2.2). Here $S^N_+ \subseteq S^N$ denotes the cone of positive semidefinite $N \times N$ matrices. Let us assume that strong duality holds and the optimal values are attained, so that the optimal primal and dual values are the same. It follows that if X^* is an optimal solution to the primal problem (4.2.5) and Z^* is an optimal solution to the dual problem (4.2.6), we have $X^* Z^* = 0$. Further assume that X^* is nonzero, and consequently, Z^* has at least one eigenvalue equal to zero.

Then the dual problem (4.2.6) is equivalent to the following unconstrained eigenvalue optimization problem

$$\min_{y \in \mathbb{R}^n} \quad f(y), \tag{4.2.7}$$

with the exact penalty dual function [Ding et al., 2019]

$$f(y) = b^T y + \alpha \max\{\lambda_{\max}(C - \mathcal{A}^T y),\ 0\}, \tag{4.2.8}$$

for sufficiently large α, where $\lambda_{\max}$ denotes maximum eigenvalue as earlier. Note that this exact penalty function differs from the eigenvalue optimization formulation in [Helmberg & Rendl, 2000], namely

$$b^T y + \alpha \lambda_{\max}(C - \mathcal{A}^T y)$$

which does not include the $\max\{\cdot\ ,\ 0\}$ operator. In that formulation, to give a correct equivalence α must be exactly equal to a critical value, as opposed to greater than or equal to this value. For the SDP problems we consider in the following we already know valid lower bounds for α. Note that at an optimal solution y^* the maximum eigenvalue of $-Z^* = C - \mathcal{A}^T y^*$ is zero, often with multiplicity greater than one, and hence f is nonsmooth at y^*.

4.2.4 Max Cut Problem

Our first example of semidefinite programming (SDP) arises from the Max Cut problem. This subsection and a subsequent one on the Matrix Completion problem were motivated by the recent paper [Ding et al., 2019] and the observation made

there that the first-order algorithms they used to minimize the penalized dual function (4.2.8) arising from Max Cut SDPs were slow. Here we compare full BFGS, scaled and unscaled L-BFGS and the subgradient method (again with $t_k = 1/k$) on penalized dual functions arising from Max Cut SDPs. We note that one of the key ideas in [Ding et al., 2019] is that, when the primal SDP optimal solution X^* has rank much less than N, an accurate estimate of the optimal value of the SDP obtained from minimizing the penalized dual function allows the use of a novel method for obtaining efficient low-rank solutions to the *primal* SDP even when N is large.

Let A be the adjacency matrix of an undirected simple graph G with the set of vertices $\mathbb{V}$ and edges $\mathbb{E}$. The matrix A is square with size $N = |\mathbb{V}|$; its off-diagonal entries a_{ij} are the non-negative weights of the edge (i, j) and its diagonal entries are zero. For an unweighted graph the adjacency matrix is a zero-one matrix: entry (i, j) is zero if there is no edge from vertex v_i to v_j, and otherwise is one. The max-cut problem is to divide $\mathbb{V}$ into two disjoint sets, say S and $\mathbb{V}\backslash S$, such that the total weight of the edges crossing from S to its complement set is maximized. Thus, we would like to maximize $\sum_{(i,j)\in\mathbb{E},\ i\in S,\ j\in\mathbb{V}\backslash S} a_{ij}$. If we label the vertices belonging to S by $+1$, and the rest by -1, then we can equivalently maximize $\sum_{(i,j)\in\mathbb{E}} a_{ij}(1-l_il_j)/2$, where l_i is the label of vertex v_i. We can impose the constraint $l_i = \pm 1$ by writing $l_i^2 = 1$, so we obtain

$$\begin{aligned} \max \quad & \frac{1}{2}\sum_{(i,j)\in\mathbb{E}} a_{ij}(1 - l_il_j) \\ \text{subject to} \quad & l_i^2 = 1, \quad l_i \in \mathbb{R}. \end{aligned} \tag{4.2.9}$$

This problem is NP-hard. But we can relax it by lifting variables l_i to higher

dimension $\mathbb{R}^N$, and replacing the scalar multiplication $l_i l_j$ with the dot product $\langle l_i, l_j \rangle$ in (4.2.9) and further noticing that the resulting matrix with entry (i, j) equal to $\langle l_i, l_j \rangle$ is positive semidefinite. This relaxed problem can be expressed as an SDP. To do so, we introduce the degree matrix D of G, a square diagonal matrix of size N where d_{ii} is the degree of the vertex v_i, when G is unweighted, and the total weight coming out of vertex v_i when G is weighted. Define the Laplacian matrix $L = D - A \in S^N$. The primal SDP Max Cut problem and its dual are then as follows [Bandeira, 2015]:

$$\begin{aligned} &\max_{X} \quad \frac{1}{4}\langle L, X \rangle && (4.2.10) \\ &\text{subject to} \quad \mathbf{Diag}\,(X) = \mathbb{1} \quad \text{and} \quad X \in S_+^N, \\ &\min_{y \in \mathbb{R}^n} \quad \mathbb{1}^T y && (4.2.11) \\ &\text{subject to} \quad Z = \mathbf{Diag}\,(y) - \frac{1}{4}L \quad \text{and} \quad Z \in S_+^N. \end{aligned}$$

Note that these are instances of the primal and dual SDP introduced in (4.2.5) and (4.2.6), respectively. By definition for the SDP Max Cut problem we have $n = N$. The penalized dual function (4.2.8) for the Max Cut problem is

$$f(y) = \mathbb{1}^T y + \alpha \ \max\{\lambda_{\max}(L - \mathbf{Diag}\,(y)),\ 0\}. \tag{4.2.12}$$

Due to the constant trace property of the primal Max Cut SDP (4.2.10), the trace (nuclear) norm of the primal optimal solution is known, i.e. we have $\|X^*\|_* = N$, and hence any solution y^* to the penalized dual max cut problem (4.2.12) with $\alpha \geqslant \|X^*\|_*$, is also a solution to the dual SDP (4.2.11) and vice versa [Ding et al., 2019, Lem. 6.1].

We picked graph $G1$ from the Gset group in the sparse matrix collection [Gse, 2014] for the following experiment. $G1$ is an unweighted graph with $N = 800$ vertices and the adjacency matrix A is a sparse symmetric matrix with 38352 nonzero entries (they are all equal to 1). Since N is relatively small we can apply the SDPT3 solver via CVX to the primal SDP (4.2.10), obtaining the optimal primal and dual value $f^* = 12083.19765$. The rank r^* of the optimal primal solution X^* is 13, and strict complementarity holds, so the nullity of the dual solution Z^* is also 13.

The experiment is presented in Figure 4.11. We compare the performance of full BFGS, L-BFGS-5 with and without scaling, and the subgradient method (with $t_k = 1/k$) to minimize the penalized dual function (4.2.12) with $\alpha = 2N = 1600$. As before, the maximum allowed number of function evaluations is set to 10^4. In contrast to the two experiments presented in Figure 4.7 and 4.8 respectively, for the nonsmooth Max Eigenvalue problem, the results of this experiment are in favor of L-BFGS when compared to the subgradient method. Both scaled L-BFGS-5 (magenta dots) and unscaled L-BFGS-5 (cyan dots) reduce the relative error down to $\approx 10^{-2}$ where as the subgradient method (blue dots) gives an answer of $\approx 10^0$. Full BFGS (black dots) reduces the error to $\approx 10^{-6}$.

In Figure 4.12, we show the *negative* of the top 20 eigenvalues of the final negative dual slack matrix $-Z$ (equivalently, the smallest 20 eigenvalues of Z) obtained by the four methods, along with values obtained by SDPT3. It is interesting to note that full BFGS approximates the eigenvalues well, in the sense that it clearly separates the first 13 approximately zero eigenvalues from the approximations to the nonzero eigenvalues, although not as decisively as SDPT3. However, the final eigenvalues obtained by L-BFGS-5 are not clearly separated.

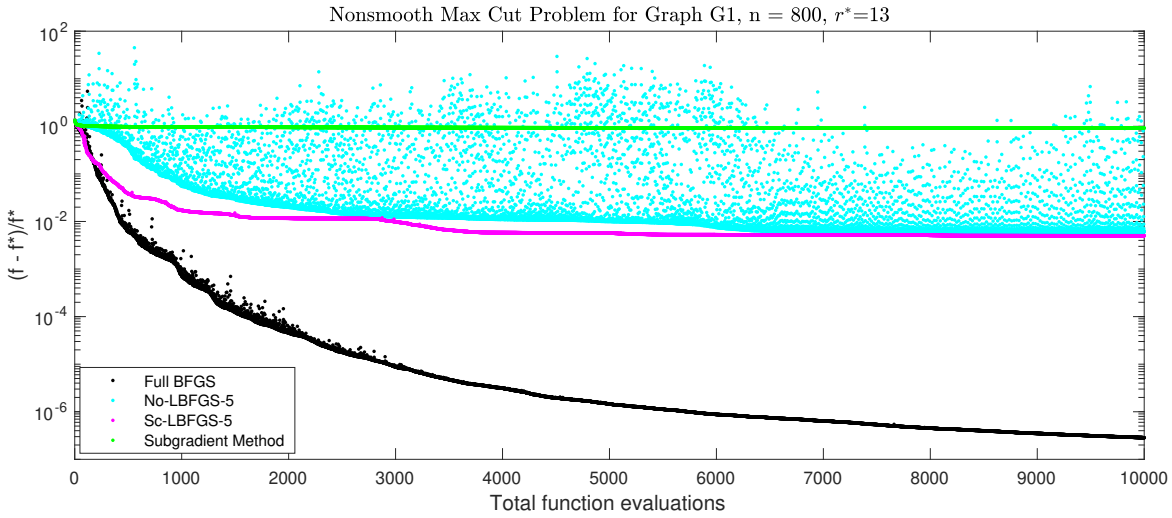


Figure 4.11: Comparing BFGS, L-BFGS-5 with scaling on and off and the subgradient method on penalized dual Max Cut problem (4.2.12).

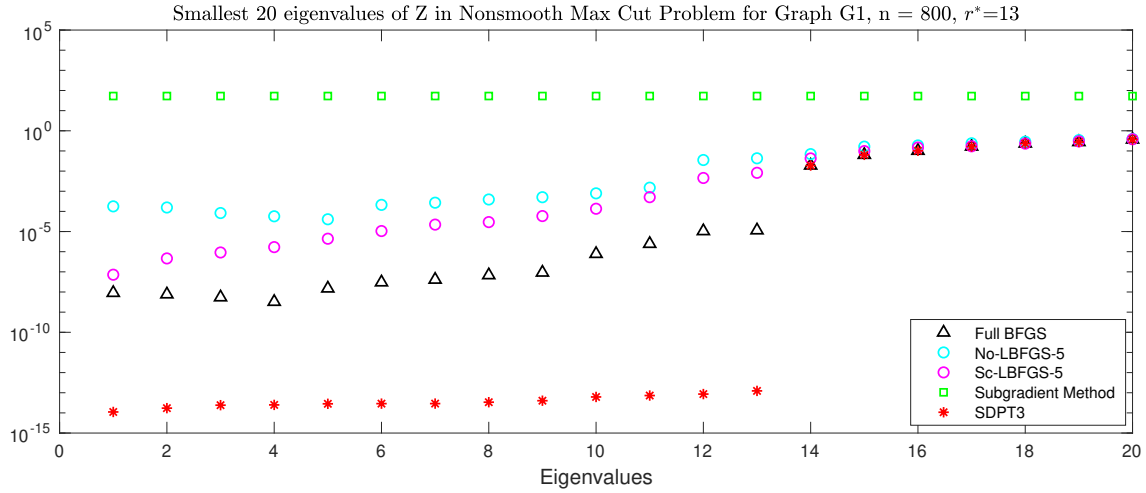


Figure 4.12: Comparing smallest 20 eigenvalues of the dual slack matrix Z obtained by BFGS, L-BFGS-5 with scaling on and off and the subgradient method on penalized dual Max Cut problem (4.2.12) for $G1$ graph with $n = 800$. The maximum number of function evaluations is 10^4. The nullity of the optimal dual slack matrix Z^* is 13. The smallest 20 eigenvalues obtained from SDPT3 are shown as well. The lack of monotonicity at the left end of some of the plots occurs because we actually plotted the absolute values of the ordered largest eigenvalues of $-Z$, and some of these eigenvalues are positive, either because of rounding errors or insufficient accuracy in the optimization.

Next we repeat the same experiment in Figures 4.13 and 4.14, only with $m = 20$.

The result for BFGS and the subgradient method are shown again for comparison.

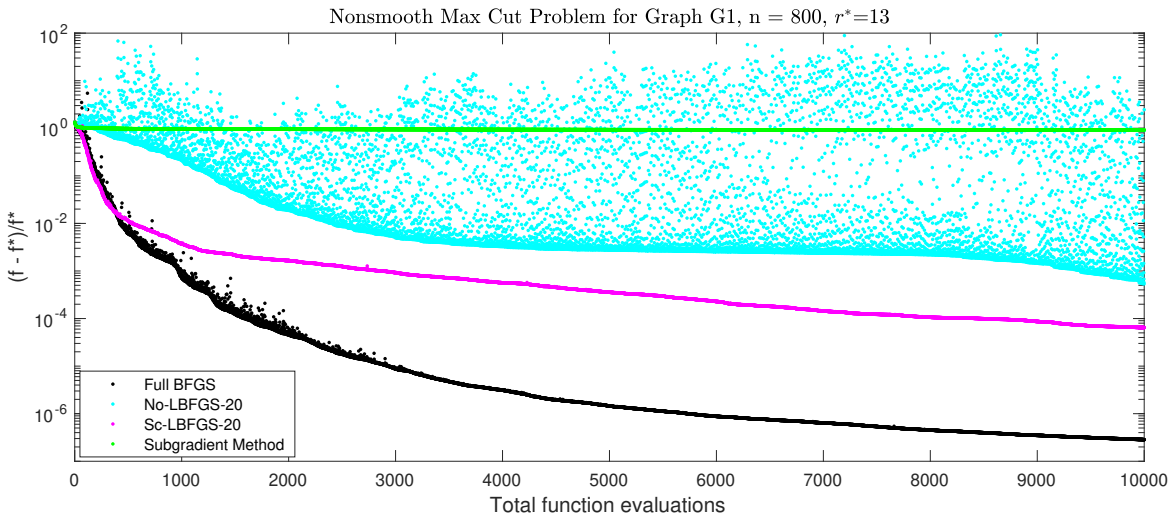


Figure 4.13: Comparing BFGS, L-BFGS-20 with scaling on and off and the subgradient method on penalized dual Max Cut problem (4.2.12).

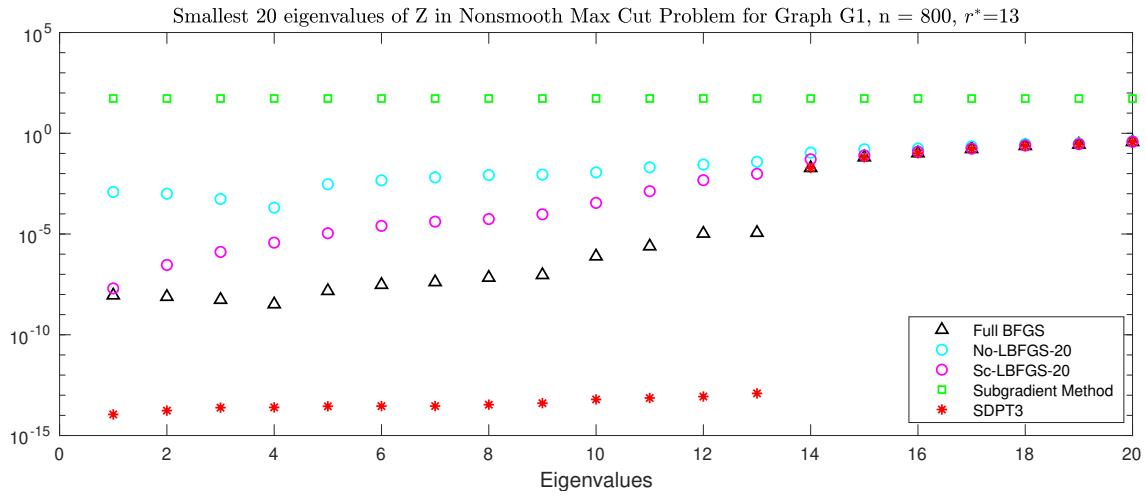


Figure 4.14: Comparing smallest 20 eigenvalues of the dual slack matrix Z obtained by BFGS, L-BFGS-5 with scaling on and off and the subgradient method on penalized dual Max Cut problem (4.2.12) for $G1$ graph with $n = 800$. The maximum number of function evaluations is 10^4. The nullity of the optimal dual slack matrix Z^* is 13. The smallest 20 eigenvalues obtained from SDPT3 are shown as well. See legend of Figure 4.12 regarding eigenvalue monotonicity.

We see that scaled L-BFGS-20 now reduces the relative error down to 10^{-4} and

unscaled to about 10^{-3}, compared to about 10^{-2} for scaled and unscaled L-BFGS-5. However, the eigenvalue plot is similar to the corresponding plot for L-BFGS-5: neither variant is able to discover that the nullity of the optimal dual slack matrix Z^* is 13.

4.2.5 Smoothed Max Cut Problem

Consider now Nesterov smoothing of the penalized dual Max Cut problem (4.2.12):

$$f_\mu(y) = \mathbb{1}^T y + \alpha\mu \log\left(1 + \sum_{i=1}^{n} \exp\left(\lambda_i(L - \mathbf{Diag}\,(y))/\mu\right)\right) - \alpha\mu\log(n+1). \quad (4.2.13)$$

Note the presence of the term "1" which does not appear in (4.2.4): this reflects the presence of the $\max\{\cdot\ ,\ 0\}$ operator in the penalty function (4.2.8). This smoothing requires a complete eigendecomposition and hence it is expensive to evaluate (for a matrix of size N, it is $O(N^3)$). Since we are interested in solving Max Cut problems for large N, we now introduce an approximation of this function which only needs the top K eigenvalues of the given matrix, here the negative dual slack matrix: $-Z = L - \mathbf{Diag}\,(y)$:

$$f_{\mu,K}(y) = \mathbb{1}^T y + \alpha\mu \log\left(1 + \sum_{i=1}^{K} \exp(\lambda_i(L - \mathbf{Diag}\,(y))/\mu)\right) - \alpha\mu\log(K+1). \quad (4.2.14)$$

The justification is that, since the eigenvalues are ordered algebraically, the largest ones dominate the smoothed approximation. The function $f_{\mu,K}$ is guaranteed to be smooth at y only if $\lambda_K(L - \mathbf{Diag}\,(y))$ is larger than $\lambda_{K+1}(L - \mathbf{Diag}\,(y))$ [Overton & Womersley, 1993]. This implies that K should be no smaller than the optimal dual nullity (primal rank) r^*. Except as noted below, we use MATLAB's

`eigs` to compute the largest K eigenvalues via the Lanczos method. This is in contrast to the nonsmooth experiment in Figures 4.11 and 4.13 where we used MATLAB's `eig`, which calls a backwards stable algorithm from LAPACK to compute all the eigenvalues, in order to determine the maximal eigenvalue.

In Figures 4.15 and 4.16, we show several experiments, each one applying full BFGS, scaled L-BFGS-5 and scaled L-BFGS-20 to (4.2.14) with $K \in \{5, 15, 20\}$ and with $\mu = 10^{-7}$ for all of them. We do not include the unscaled variants of L-BFGS because it is clear that these offer no advantage on smooth problems. In all cases, the underlying Max Cut problem, $G1$, is the same instance as in the nonsmooth experiment in Figure 4.11, with $n = 800$, $f^* = 12083.19765$ and the optimal primal rank and dual nullity $r^* = 13$. All of the methods start form the same initial point used in the previous section. The plots show the relative error $(f_{\mu,K} - f^*)/f^*$ as a function of the iteration count, where f^* is the optimal value of the *nonsmooth* instance of the primal problem. The maximum number of iterations is set to 10^3.

In these experiments we used `eigs` except in the bottom plot in Figure 4.15, where we computed the eigenvalues of $-Z$ by `eig`, using only the top 5 of them, in order to demonstrate the difference it makes in the result for this experiment. When $K = 5$, all the methods break down early when computing the eigenvalues via `eigs`, but when using the more stable `eig`, none of them break down. The reason for the breakdown may be that `eigs` is computing the eigenvalues less accurately than `eig`, perhaps exacerbated by the failure to make K, the number of eigenvalues requested from `eigs`, larger than the expected multiplicity at the optimal solution. When function values are not computed accurately, the typical result is failure of the line search to satisfy the Armijo condition, resulting in termination. Indeed, BFGS performs much better for $K = 5$ when using `eig` instead of `eigs`. Increasing

K to 15, using `eigs`, results in both full BFGS and L-BFGS-20 finding a better answer, most likely because when $K = 5 < r*$, the function $f_{\mu,K}(y)$ is not a valid smoothing, as noted above. However, L-BFGS-5 continues to break down early, even when K is further increased to 20, and full BFGS computes a worse answer with $K = 20$ than it does for $K = 15$. The conclusion we reach from this is that results using `eigs` are somewhat inconsistent and unpredictable, which is perhaps not surprising, given the general reputation of the reliability of `eigs` versus `eig`. However, when we turn to much larger graphs, increasing $n = N$ substantially, we have little choice: the only option is to use the more efficient L-BFGS, not full BFGS, and the more efficient `eigs`, not `eig`, as we do in the final experiments in this section.

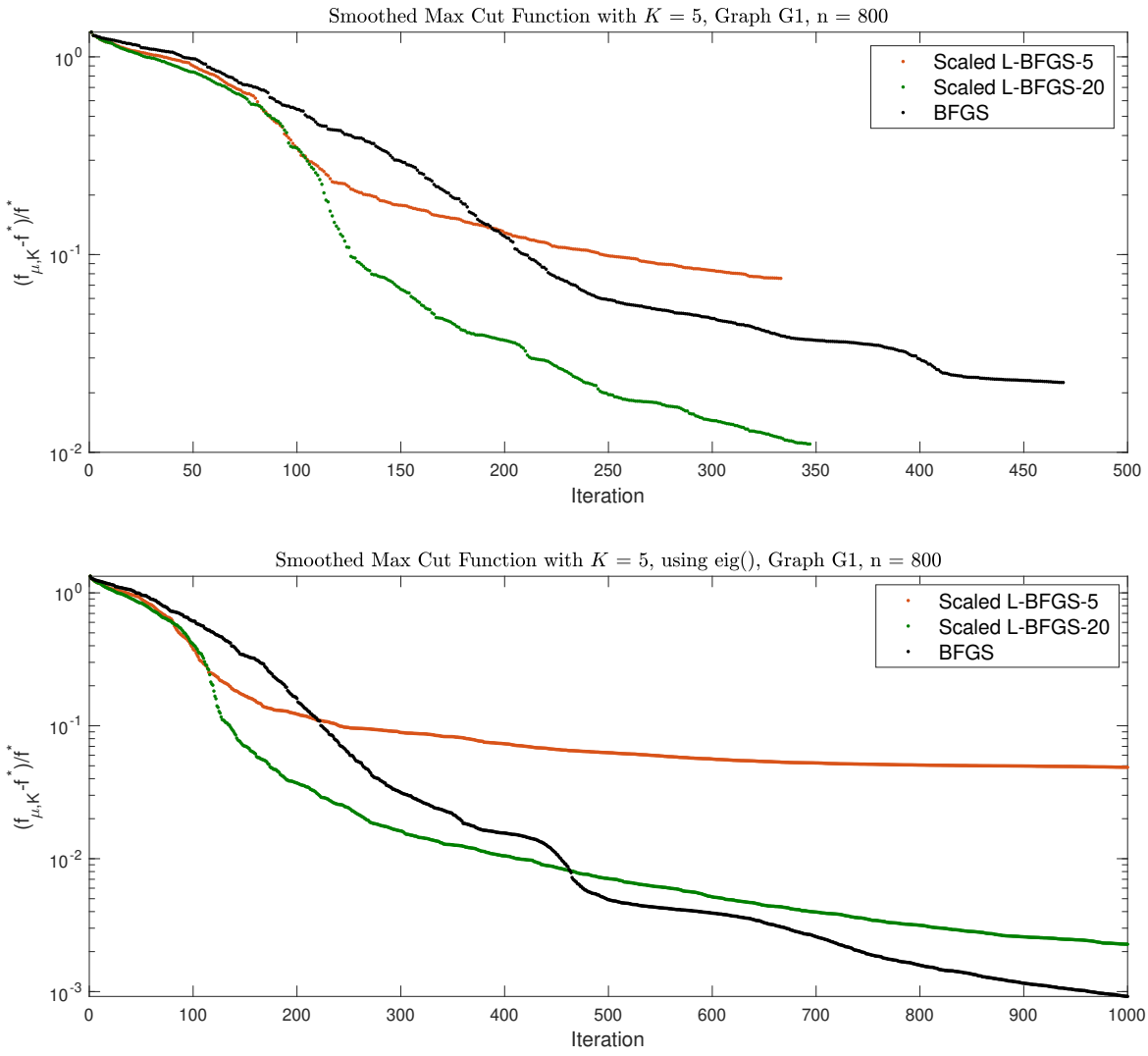


Figure 4.15: Comparing scaled L-BFGS-5, scaled L-BFGS-20 and BFGS on the smoothed approximate Max Cut problem (4.2.14) for $G1$ graph with $n = 800$ for $K = 5$. $f^* = 12083.19765$. The smoothing parameter is $\mu = 10^{-7}$. The maximum number of iterations is 10^3. **Top:** Using MATLAB's `eigs`. **Bottom:** Using MATLAB's `eig`.

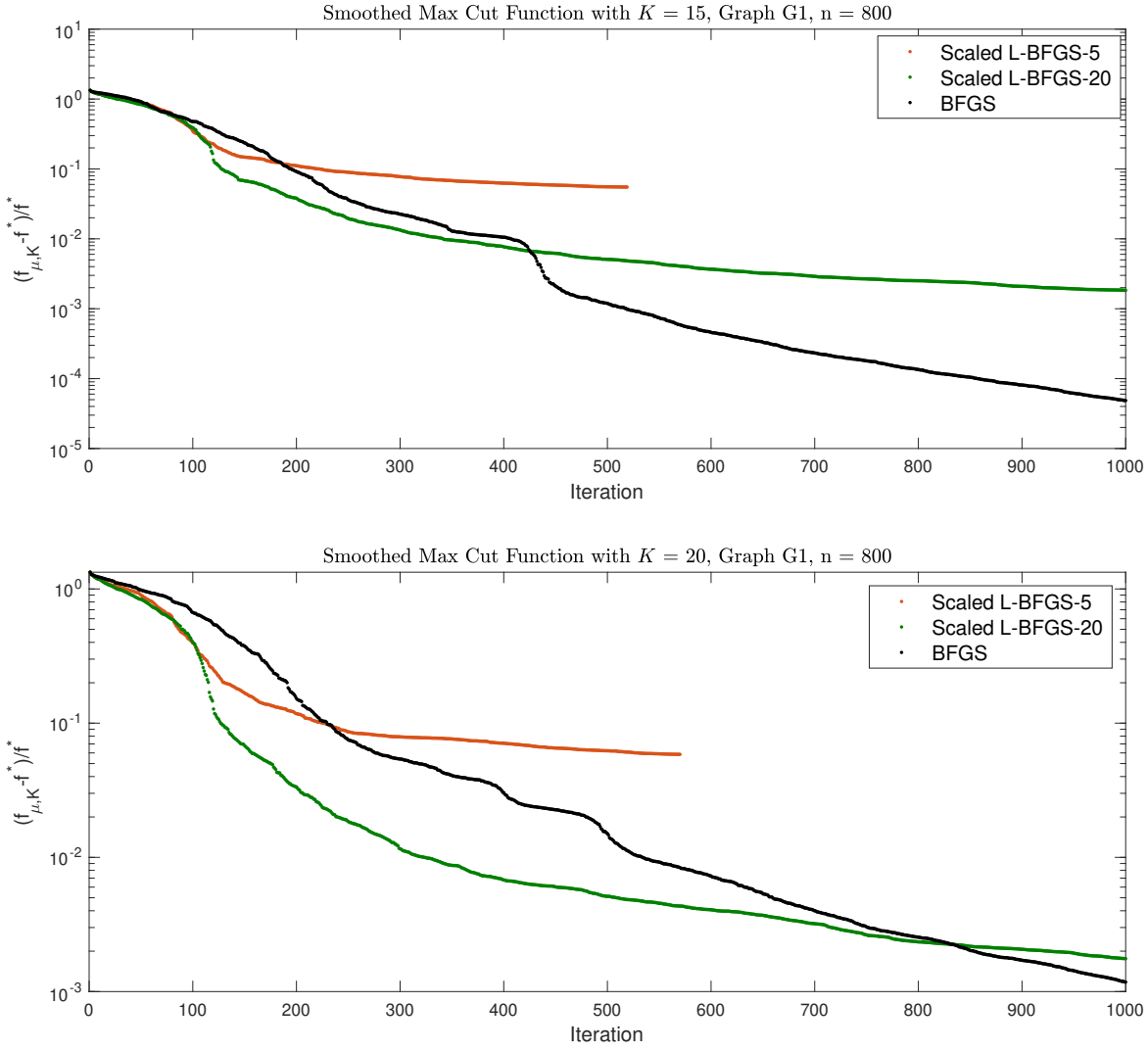


Figure 4.16: Comparing scaled L-BFGS-5, scaled L-BFGS-20 and BFGS on the smoothed approximate Max Cut problem (4.2.14) for $G1$ graph with $n = 800$ for different values of K. $f^* = 12083.19765$. The smoothing parameter is $\mu = 10^{-7}$. The maximum number of iterations is 10^3. **Top:** $K = 15$. **Bottom:** $K = 20$.

In Figure 4.17, we plot the negative of the top K eigenvalues of the matrix $-Z = L - \mathbf{Diag}(y)$ which we get from the final answer y obtained by each method. Equivalently, these are the smallest K eigenvalues of Z. Notice that, as with the

experiments on the original nonsmooth problem, full BFGS is able to separate the approximately zero first 13 eigenvalues of Z from the rest, for $K = 15$ and, less clearly, $K = 20$, but L-BFGS is not.

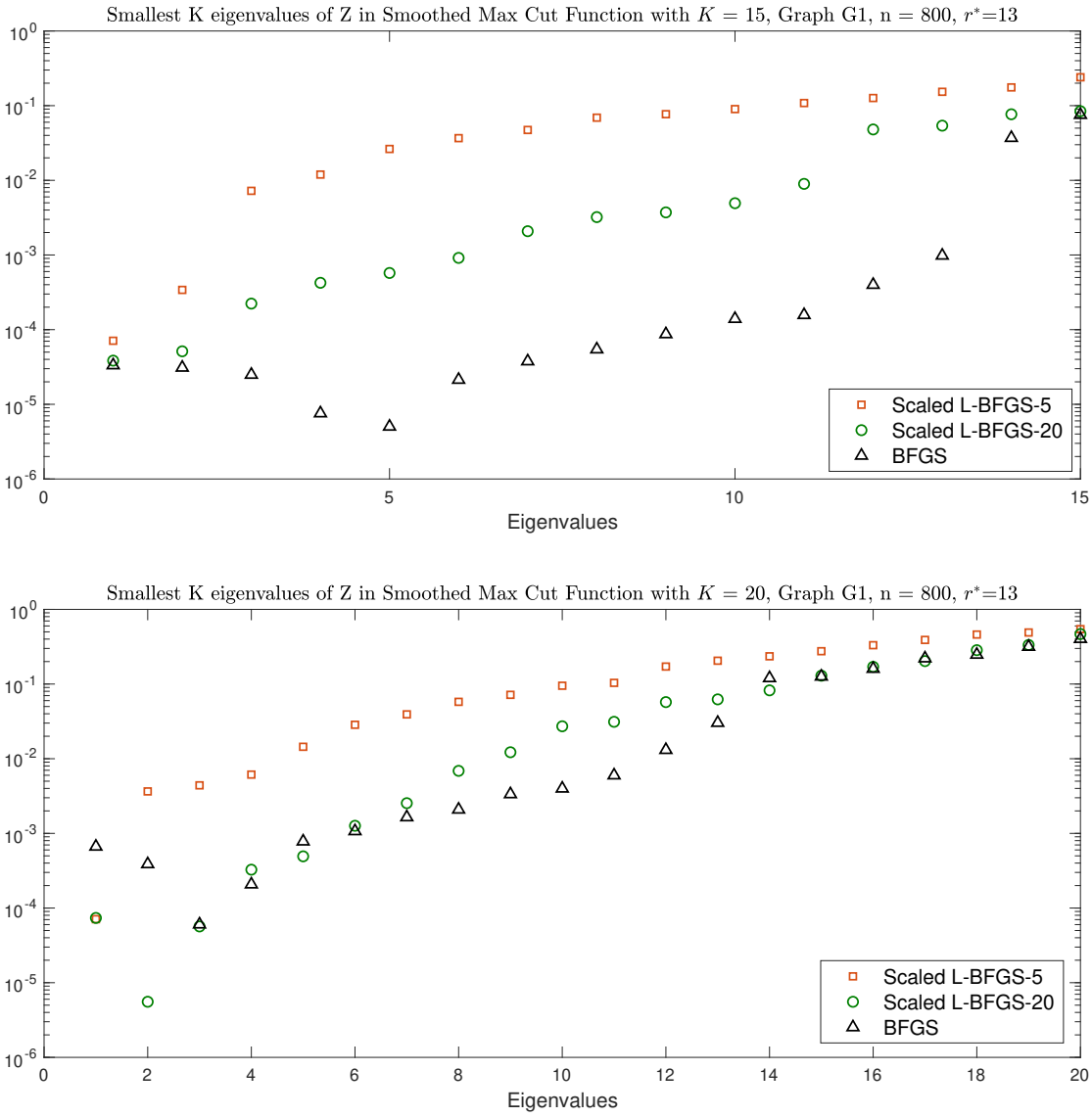


Figure 4.17: Comparing scaled L-BFGS-5, scaled L-BFGS-20 and BFGS on the smoothed approximate Max Cut problem (4.2.14) for $G1$ graph with $n = 800$ for different values of K. $f^* = 12083.19765$. The smoothing parameter is $\mu = 10^{-7}$. The maximum number of iterations is 10^3. **Top:** Smallest $K = 15$ eigenvalues of Z. **Bottom:** Smallest $K = 20$ eigenvalues of Z.

For the last experiment on the Max Cut problem we select a much larger graph from Gset [Gse, 2014], Graph G55. The adjacency matrix is a random sparse binary matrix with 0.05% uniformly distributed density and $N = 5000$. In the following we present the result of applying scaled L-BFGS-5 and L-BFGS-20 to the smoothed Max Cut function (4.2.14) with $K = 100$ and $\mu = 10^{-6}$, using `eigs` to compute the eigenvalues. Figure 4.18 shows the function value $f_{\mu,K}$ at each iteration.

Figure 4.19 shows the 100 smallest eigenvalues of Z computed from the final answer we get from each method. We see that L-BFGS-20 is able to reduce the smallest eigenvalue of Z to about 10^{-3}, but none of the eigenvalues obtained by L-BFGS-5 are close to zero, indicating that the computed solution is nowhere near optimal.

So, we increased the maximum number of iterations to 10^4 to allow the experiment run for longer. See Figures 4.20 and 4.21. Now we see that the result for L-BFGS-5 improves significantly, but even so, the result it obtains after 10^4 iterations is still worse than the result L-BFGS-20 obtained after 10^3 iterations. In contrast, the result obtained by L-BFGS-20 after 10^4 iterations is not much better than it obtained in 10^3 iterations. Since we don't know the optimal value for this problem, further investigation would be needed to investigate the accuracy of the result obtained by L-BFGS-20. Furthermore, it is difficult to draw any conclusion about the optimal nullity of Z from the eigenvalue plot.

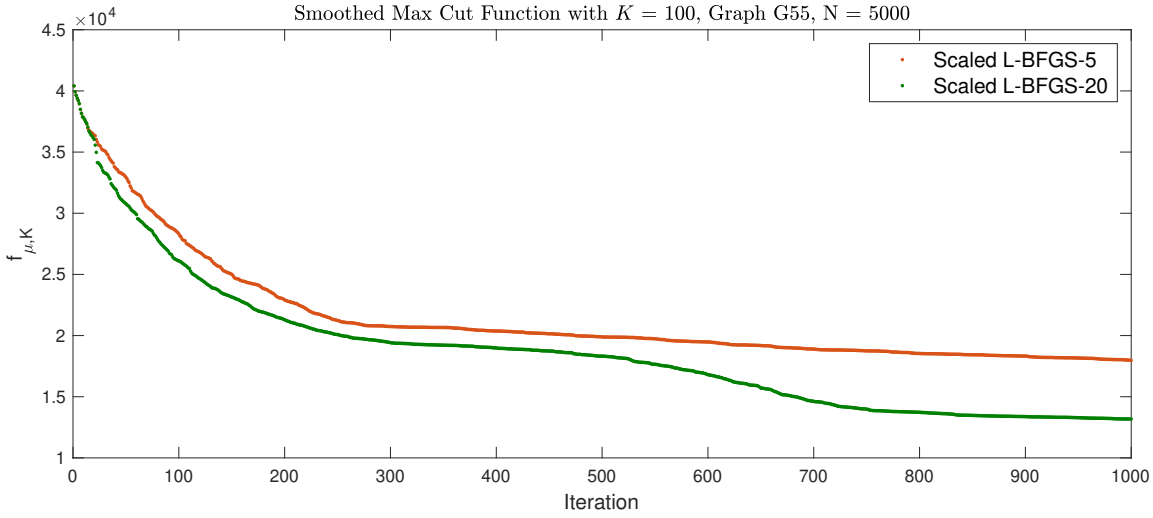


Figure 4.18: Comparing scaled L-BFGS-5 and scaled L-BFGS-20 on smoothed Max Cut problem (4.2.14) with $K = 100$ for $G55$ graph with $N = 5000$. The smoothing parameter is $\mu = 10^{-6}$. The maximum number of iterations is 10^3.

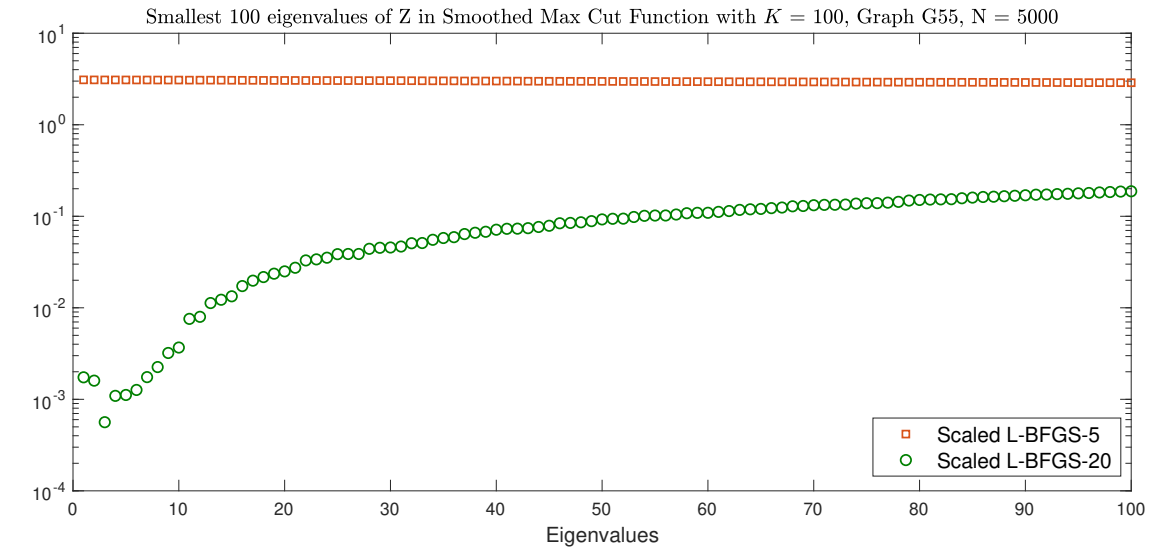


Figure 4.19: Comparing smallest 100 eigenvalues of the dual slack matrix Z obtained by scaled L-BFGS-5, and scaled L-BFGS-20 on the smoothed Max Cut problem (4.2.14) with $K = 100$ for $G55$ graph with $N = 5000$. The smoothing parameter is $\mu = 10^{-6}$. The maximum number of iterations is 10^3.

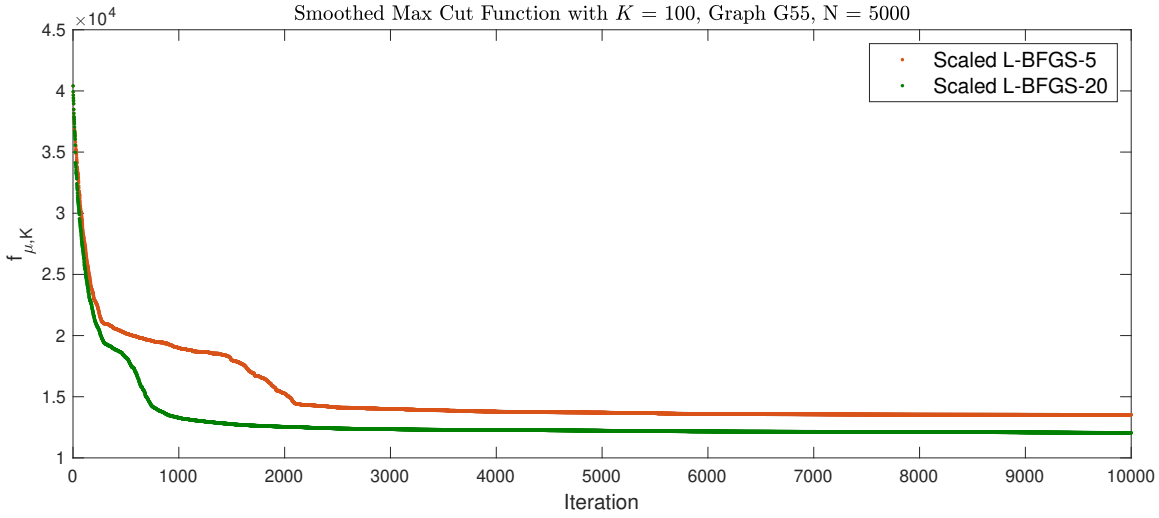


Figure 4.20: Comparing scaled L-BFGS-5 and scaled L-BFGS-20 on smoothed Max Cut problem (4.2.14) with $K = 100$ for $G55$ graph with $N = 5000$. The smoothing parameter is $\mu = 10^{-6}$. The maximum number of iterations is 10^4.

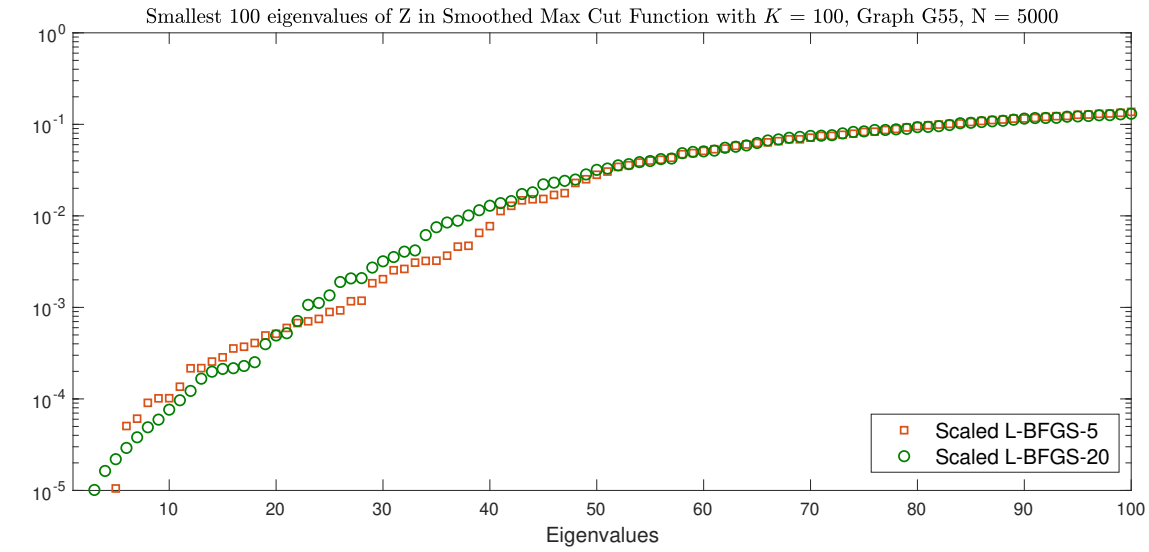


Figure 4.21: Comparing smallest 100 eigenvalues of the dual slack matrix Z obtained by scaled L-BFGS-5, and scaled L-BFGS-20 on the smoothed Max Cut problem (4.2.14) with $K = 100$ for $G55$ graph with $N = 5000$. The smoothing parameter is $\mu = 10^{-6}$. The maximum number of iterations is 10^4.

An issue that would be worth investigating in the future is whether passing an

initial starting vector to `eigs` based on the previous iterate would result in significant improvements in either running time or accuracy; [Mitchell & Overton, 2016] suggests using the average of all the vectors returned by `eigs` in the previous function call.

4.2.6 Matrix Completion Problem

The Matrix Completion problem is as follows. Suppose $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2}$ denotes a low-rank matrix for which we only have access to some of its entries and would like to recover entirely by minimizing the rank over all matrices whose entries agree with the known values. This rank minimization problem is NP-hard, so we relax it by minimizing a well-known convex surrogate for the rank: the nuclear norm (sum of all the singular values). Let Ω be the set of pairs (i, j) for which $\mathcal{X}_{ij}$ is known. Then the nuclear norm (a.k.a. trace norm) minimization problem can be expressed as the following SDP [Recht et al., 2010]

$$\begin{aligned} \max_{X \in S^{(N_1+N_2)}} \quad & -\operatorname{Tr}(W_1) - \operatorname{Tr}(W_2) \\ \text{subject to} \quad & U_{ij} = \mathcal{X}_{ij}, \ (i,j) \in \Omega, \\ & X = \begin{bmatrix} W_1 & U \\ U^T & W_2 \end{bmatrix} \in S_+^{(N_1+N_2)}. \end{aligned} \tag{4.2.15}$$

We write the primal problem in the *max* form in order to be consistent with the SDP form (4.2.5), with $N = N_1 + N_2$. Define the constraint $U_{ij} = \mathcal{X}_{ij}$ for $(i, j) \in \Omega$ in linear operator form $\mathcal{B}(U) = b$, where $b \in \mathbb{R}^n$ with $n = |\Omega|$ is the vector consisting of the known entries of $\mathcal{X}$ in some prescribed order and $\mathcal{B} : \mathbb{R}^{N_1 \times N_2} \to \mathbb{R}^n$, with $\mathcal{B}^T$

its adjoint operator. The dual SDP is

$$\begin{aligned} &\min_{y\in\mathbb{R}^n} \quad b^T y \\ &\text{subject to} \quad Z = \begin{bmatrix} \mathbb{0}_{N_1\times N_1} & \mathcal{B}^T(y) \\ (\mathcal{B}^T(y))^T & \mathbb{0}_{N_2\times N_2} \end{bmatrix} - \big(-I_{(N_1+N_2)}\big),\ Z \in S_+^{(N_1+N_2)}. \end{aligned}$$

The dual can be represented more compactly as:

$$\begin{aligned} &\min_{y\in\mathbb{R}^n} \quad b^T y \\ &\text{subject to} \quad Z = \begin{bmatrix} I_{N_1} & \mathcal{B}^T(y) \\ \big(\mathcal{B}^T(y)\big)^T & I_{N_2} \end{bmatrix},\quad Z \in S_+^{(N_1+N_2)}, \end{aligned} \tag{4.2.16}$$

The exact penalty dual function (4.2.8) for the Matrix Completion problem is then [Ding et al., 2019]

$$f(y) = b^T y + \alpha \max\left\{\lambda_{\max}\left(-\begin{bmatrix} I_{N_1} & \mathcal{B}^T(y) \\ \big(\mathcal{B}^T(y)\big)^T & I_{N_2} \end{bmatrix}\right),\ 0\right\}. \tag{4.2.17}$$

For the experiment in this part, we generated a low-rank random matrix $\mathcal{X}$ of size $N_1 = 20$ by $N_2 = 160$ with rank $\mathcal{R} = 3$. We then selected the ordered pairs in Ω randomly with the probability of each (i, j) being included set to 0.2. For this problem instance we got $|\Omega| = n = 587$. We then applied the various methods to minimize (4.2.17) with $\alpha = 2\|X^*\|_* = 2\,\mathrm{Tr}\,(X^*) = -2f^* = 3.0796$, where $f^* = -1.5398$ and $X^* \in S_+^{(N_1+N_2)}$ were obtained from solving the SDP (4.2.16) via SDPT3. Note that the optimal value is negative because of the minus sign in the '*max* formulation of the primal SDP.

Figure 4.22 and Figure 4.23 show the performance of full BFGS, L-BFGS-5 with and without scaling, and the subgradient method (with $t_k = 1/k$). The y-axis in Figure 4.22 shows the relative error $(f - f^*)/|f^*|$ and the maximum allowed number of function evaluations is set to 10^4. As is evident from the plot, both variants of L-BFGS-5 outperform the subgradient method, even though scaled L-BFGS-5 quits early before 5000 evaluations and unscaled L-BFGS-5 right before 10000 evaluations.

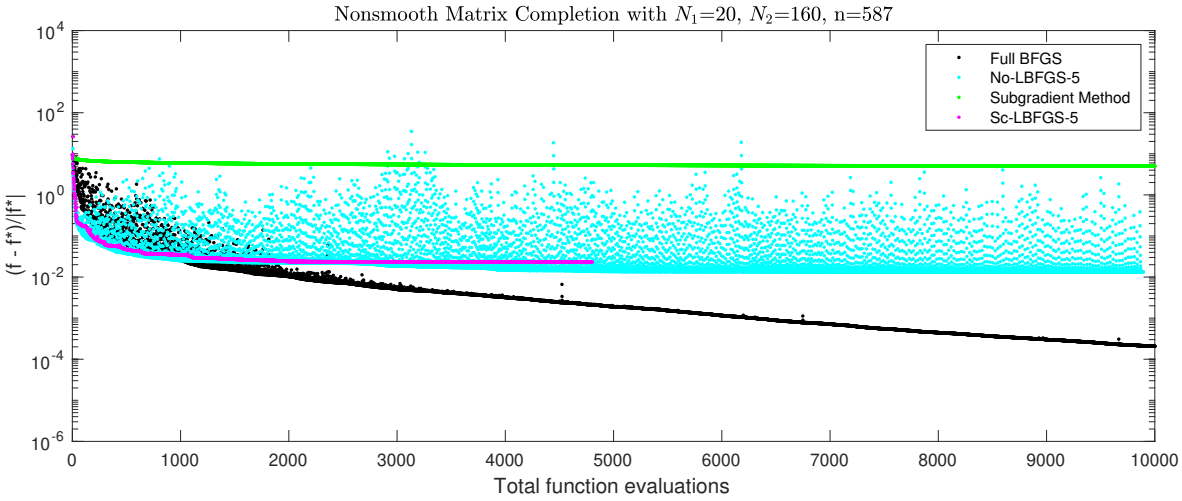


Figure 4.22: Comparing BFGS, LBFGS-5 with scaling on and off and the subgradient method on penalized dual Matrix Completion problem (4.2.17). The maximum number of function evaluations is 10^4.

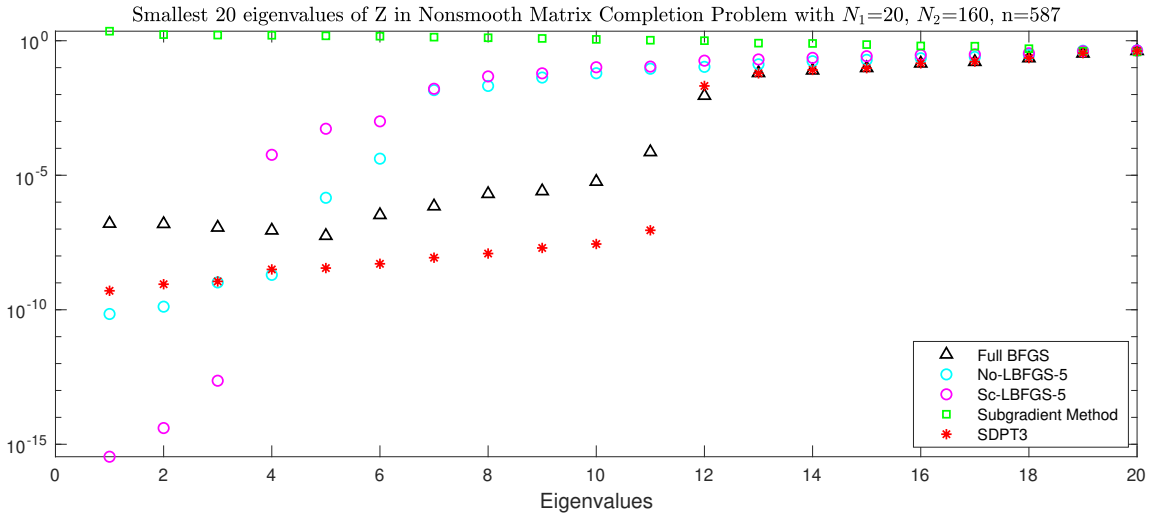


Figure 4.23: Comparing smallest 20 eigenvalues of the dual slack matrix Z obtained by BFGS, L-BFGS-5 with scaling on and off and the subgradient method on penalized dual Matrix Completion problem (4.2.17). See legend of Figure 4.12 regarding eigenvalue monotonicity.

Figure 4.23 presents the negative of the top 20 eigenvalues of the final negative dual slack matrix $-Z$, or equivalently, the 20 smallest eigenvalues of Z, obtained by the four methods, along with values obtained by SDPT3. As before, BFGS is able to separate the zero and nonzero eigenvalues of Z^*, agreeing with SDPT3 that the nullity of Z^* is effectively 11. Note that this is larger than $\mathcal{R} = 3$, the rank of the original matrix $\mathcal{X}$, implying that 20% was not enough observations to reconstruct $\mathcal{X}$. It is interesting that the eigenvalues of the solution found by scaled L-BFGS-5 do suggest a nullity of 3, but this may just be a coincidence.

In the following experiment in Figures 4.24 and 4.25, we increase m to 20 and again we compare the relative error and the smallest 20 eigenvalues of the dual slack matrix, respectively. In both plots the result from BFGS and the subgradient method are repeated for comparison. In Figure 4.24, neither L-BFGS-20 method quits early this time; the unscaled variant gets a slightly lower answer. The eigenvalues shown for L-BFGS-20 in Figure 4.25 do not suggest any conclusion

about the nullity of Z^*.

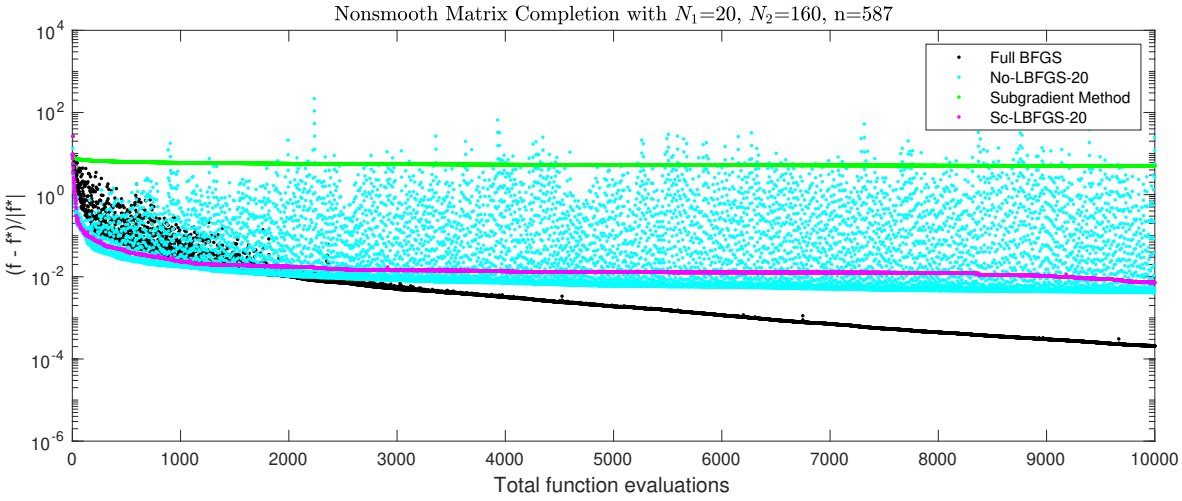


Figure 4.24: Comparing BFGS, LBFGS-20 with scaling on and off and subgradient method on penalized dual Matrix Completion problem (4.2.17). The maximum number of function evaluations is 10^4.

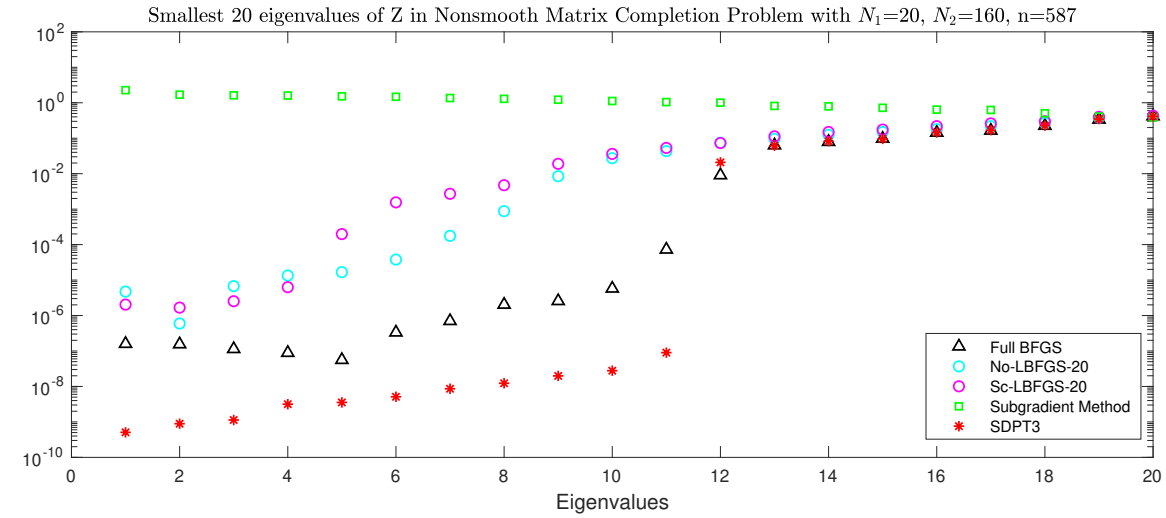


Figure 4.25: Comparing smallest 20 eigenvalues of the dual slack matrix Z obtained by BFGS, L-BFGS-20 with scaling on and off and the subgradient method on penalized dual Matrix Completion problem (4.2.17). See legend of Figure 4.12 regarding eigenvalue monotonicity.

4.2.7 Smoothed Matrix Completion Problem

Nesterov smoothing of the penalized dual Matrix Completion problem (4.2.17) gives

$$
\begin{aligned}
f_\mu(y) &= b^T y \\
&+ \alpha\mu \log\left(1 + \sum_{i=1}^{N_1+N_2} \exp\left(\lambda_i\left(-\begin{bmatrix} I_{N_1} & \mathcal{B}^T(y) \\ \left(\mathcal{B}^T(y)\right)^T & I_{N_2} \end{bmatrix}\right)/\mu\right)\right) \\
&- \alpha\mu \log(N_1 + N_2 + 1).
\end{aligned} \tag{4.2.18}
$$

In Figure 4.26 followed by Figure 4.27, we show the result of applying full BFGS, scaled L-BFGS-5 and scaled L-BFGS-20 to (4.2.18) with $\mu = 10^{-7}$. As with the smoothed max-cut experiments, we do not include the unscaled L-BFGS variants because it is clear that they offer no advantage on smooth problems. The underlying reference matrix, $\mathcal{X}$, is the same matrix as in the nonsmooth experiment in Figure 4.22, with $N_1 = 20$, $N_2 = 160$, $f^* = -1.5398$. We set $\alpha = 3.0796$ as before. All of the methods start from the same initial point used in the nonsmooth section. Unlike the experiments in the smoothed Max Cut sections, where we used MATLAB's `eigs` to compute the top K eigenvalues of the negative dual slack matrix $-Z$, here we used `eig` to compute all of the eigenvalues of $-Z$ in (4.2.18), so here it makes sense to compare the objective value obtained by scaled L-BFGS-5 and L-BFGS-20 in Figures 4.22 and 4.24, respectively, performed on the nonsmooth Matrix Completion problem (4.2.17), with the smoothed experiment which we are about to present. The maximum number of function evaluations is set to 10^4. Figure 4.27 shows the 20 smallest eigenvalues of Z computed from the final answer we get from each method.

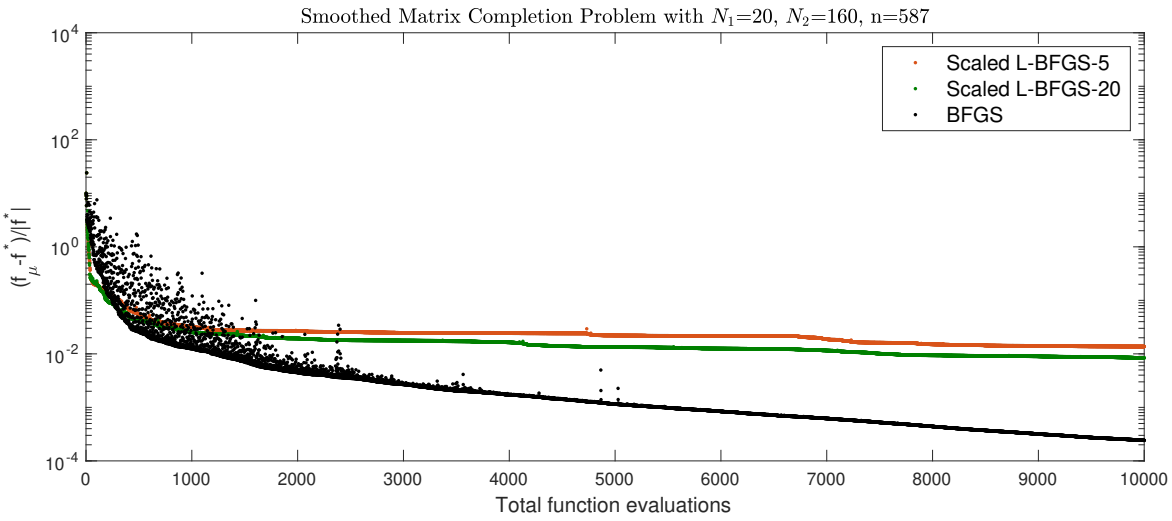


Figure 4.26: Comparing scaled L-BFGS-5, scaled L-BFGS-20 and BFGS on Smoothed Matrix Completion problem (4.2.18) for the same problem as in the Nonsmooth Matrix Completion problem. The smoothing parameter is $\mu = 10^{-7}$. The maximum number of function evaluations is 10^4.

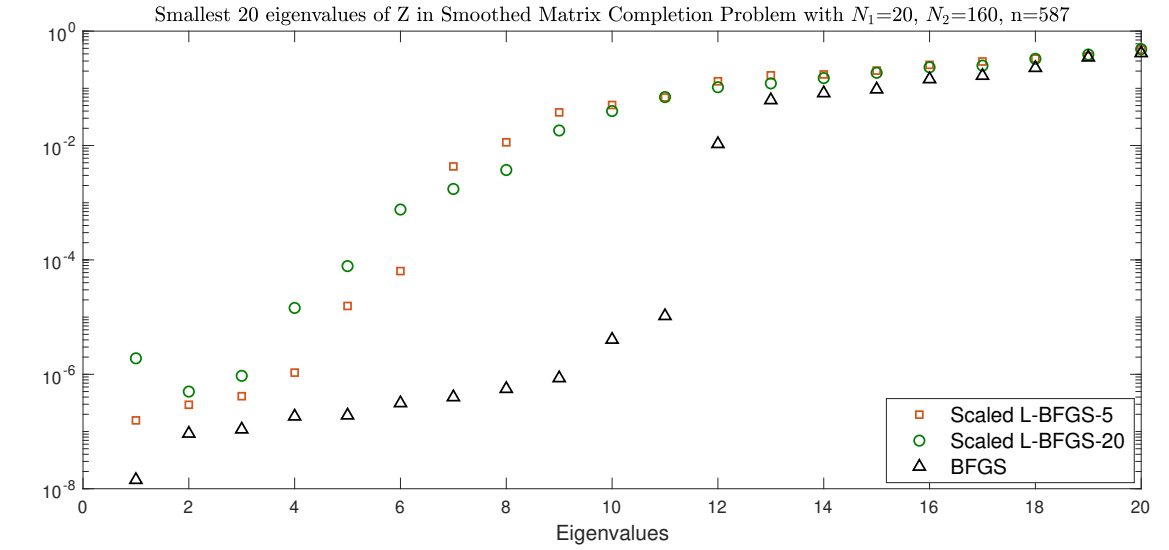


Figure 4.27: Comparing scaled L-BFGS-5, scaled L-BFGS-20 and BFGS on Smoothed Matrix Completion problem (4.2.18) for the same problem as in the Nonsmooth Matrix Completion problem. The smoothing parameter is $\mu = 10^{-7}$. The maximum number of function evaluations is 10^4. See legend of Figure 4.12 regarding eigenvalue monotonicity.

We see in Figure (4.26) that although the relative error obtained by scaled L-BFGS-20 and full BFGS are about the same as when applied to the nonsmooth function, scaled L-BFGS-5 gets a lower error when it is applied to the smoothed function, and more importantly, it does not break down early. Table 4.5 shows the final answers we obtained from full BFGS, scaled L-BFGS-20 and scaled L-BFGS-5 on the smoothed and nonsmooth Matrix Completion problem. The optimal SDP value f^* is also shown for comparison.

SDP-optimal	-1.53978555575175
BFGS-smooth	-1.53941270679104
BFGS-nonsmooth	-1.53946718487809
L-BFGS-20-smooth	-1.52686081626082
L-BFGS-20-nonsmooth	-1.52864965852708
L-BFGS-5-smooth	-1.51868588317043
L-BFGS-5-nonsmooth	-1.50422184883968

Table 4.5: Final objective value we obtained from full BFGS, scaled L-BFGS-20 and scaled L-BFGS-5 on the smoothed and nonsmooth Matrix Completion problem present in Figures 4.26, 4.24 and 4.22 respectively. The optimal SDP value f^* is also shown for comparison.

We see from the eigenvalue plot in Figure 4.27 that, as in the nonsmooth case reported in Figure 4.25, BFGS is able to separate the zero and nonzero eigenvalues of Z^*, agreeing with SDPT3 that the nullity of Z^* is effectively 11, but L-BFGS-5 an L-BFGS-20 are not.

4.3 Concluding Remarks

In this chapter we have presented many different nonsmooth problems and investigated the behavior of L-BFGS both on the given nonsmooth function and on smoothed approximations to it. When applied to the nonsmooth function directly, L-BFGS, especially its scaled variant, often breaks down early. Unscaled L-BFGS

conducts far more function evaluations per iteration than scaled L-BFGS does, and thus it is slow. Nonetheless, it is often the case that both variants obtain better results than the provably convergent, but slow, subgradient method.

On the other hand, when applied to the smoothed function, scaled L-BFGS invariably obtains a lower value than unscaled L-BFGS, often obtaining good results even when the problem is quite ill-conditioned. In particular, scaled L-BFGS seems to be a reasonable approach to minimizing smoothed exact penalty dual functions arising in large-scale semidefinite programs, although further investigation is needed to investigate the practicality of this approach and to compare it with other approaches, in particular the spectral bundle method [Helmberg & Rendl, 2000]. Minimization of the SDP exact penalty dual function is a key component of a recently proposed method for solving large-scale SDPs with low-rank primal solutions [Ding et al., 2019].

Most importantly, we find that although L-BFGS is often a reliable method for minimizing ill-conditioned smooth problems, when the condition number is so large that the function is effectively nonsmooth, L-BFGS consistently fails. This behavior is in sharp contrast to the behavior of full BFGS, which is consistently reliable for nonsmooth optimization problems. We arrive at the conclusion that, for large-scale nonsmooth optimization problems for which BFGS and other methods are not practical, it is preferable to apply L-BFGS to a *smoothed* variant of a nonsmooth problem than to apply it directly to the nonsmooth problem.

Bibliography

[Gse, 2014] (2014). The university of florida sparse matrix collection: Gset group. http://www.cise.ufl.edu/research/sparse/matrices/Gset/index.html (accessed 2019-10-10). 103, 112

[Armijo, 1966] Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific J. Math.*, 16, 1–3. 1, 2

[Asl & Overton, 2020a] Asl, A. & Overton, M. L. (2020a). Analysis of limited-memory BFGS on a class of nonsmooth convex functions. *IMA Journal of Numerical Analysis*. drz052. 10

[Asl & Overton, 2020b] Asl, A. & Overton, M. L. (2020b). Analysis of the gradient method with an Armijo–Wolfe line search on a class of non-smooth convex functions. *Optimization Methods and Software*, 35(2), 223–242. 9

[Bandeira, 2015] Bandeira, A. S. (2015). Approximation Algorithms and Max-Cut. Lecture Note for Topics in Mathematics of Data Science, MIT Open Courseware, http://ocw.mit.edu/courses/mathematics/18-s096-topics-in-mathematics-of-data-science-fall-2015/lecture-notes/MIT18_S096F15_Ses21.pdf, (accessed 2020-2-10). 102

[Barzilai & Borwein, 1988] Barzilai, J. & Borwein, J. M. (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1), 141–148. 8

[Bertsekas, 1999] Bertsekas, D. (1999). *Nonlinear Programming*. Athena Scientific, second edition. 30

[Boyd & Vandenberghe, 2004] Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. 80, 82

[Burke et al., 2020] Burke, J. V., Curtis, F. E., Lewis, A. S., Overton, M. L., & Simões, L. E. A. (2020). *Gradient Sampling Methods for Nonsmooth Optimization*. Cham: Springer International Publishing. 3

[Burke et al., 2005] Burke, J. V., Lewis, A. S., & Overton, M. L. (2005). A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM J. Optim.*, 15(3), 751–779. 3

[Cauchy, 1847] Cauchy, A. (1847). Méthode générale pour la résolution des systèmes d'équations simultanées. *Comp. Rend. Sci. Paris.*, 25, 135–163. 1, 2

[Clarke, 1990] Clarke, F. H. (1990). *Optimization and nonsmooth analysis*, volume 5 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition. 6

[Curtis et al., 2017] Curtis, F. E., Mitchell, T., & Overton, M. L. (2017). A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles. *Optimization Methods and Software*, 32(1), 148–181. 6, 35

[Dai, 2002] Dai, Y.-H. (2002). Convergence properties of the BFGS algorithm. *SIAM J. Optim.*, 13(3), 693–701 (2003). 5

[Dem'janov & Malozemov, 1971] Dem'janov, V. F. & Malozemov, V. N. (1971). The theory of nonlinear minimax problems. *Uspehi Mat. Nauk*, 26(3(159)), 53–104. 2

[Ding et al., 2019] Ding, L., Yurtsever, A., Cevher, V., Tropp, J. A., & Udell, M. (2019). An Optimal-Storage Approach to Semidefinite Programming using Approximate Complementarity. arXiv:1902.03373. 100, 101, 102, 116, 124

[Fletcher, 1987] Fletcher, R. (1987). *Practical methods of optimization*. A Wiley-Interscience Publication. John Wiley & Sons, Ltd., Chichester, second edition. 2

[Gill & Leonard, 2003] Gill, P. E. & Leonard, M. W. (2003). Limited-Memory Reduced-Hessian Methods for Large-Scale Unconstrained Optimization. *SIAM Journal on Optimization*, 14(2), 380–401. 42

[Grant & Boyd, 2008] Grant, M. & Boyd, S. (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, & H. Kimura (Eds.), *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences (pp. 95–110). Springer-Verlag Limited. http://stanford.edu/~boyd/graph_dcp.html. 90

[Grant & Boyd, 2014] Grant, M. & Boyd, S. (2014). CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx. 90

[Greenbaum et al., 2017] Greenbaum, A., Lewis, A. S., & Overton, M. L. (2017). Variational analysis of the Crouzeix ratio. *Math. Program.*, 164(1-2, Ser. A), 229–243. 35

[Guo & Lewis, 2018] Guo, J. & Lewis, A. (2018). Nonsmooth variants of Powell's BFGS convergence theorem. *SIAM Journal on Optimization*, 28(2), 1301–1311. 6, 35

[Helmberg et al., 2014] Helmberg, C., Overton, M., & Rendl, F. (2014). The spectral bundle method with second-order information. *Optimization Methods and Software*, 29(4), 855–876. 99

[Helmberg & Rendl, 2000] Helmberg, C. & Rendl, F. (2000). A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3), 673–696. 99, 100, 124

[Hiriart-Urruty & Lemaréchal, 1993] Hiriart-Urruty, J.-B. & Lemaréchal, C. (1993). *Convex analysis and minimization algorithms. I*, volume 305 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin. 2, 12, 19

[Kiwiel, 1985] Kiwiel, K. C. (1985). *Methods of descent for nondifferentiable optimization*, volume 1133 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin. 3

[Kiwiel, 2007] Kiwiel, K. C. (2007). Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 18(2), 379–388. 3

[Le et al., 2011] Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. Y. (2011). On optimization methods for deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11 (pp. 265–272). USA: Omnipress. 7

[Lemaréchal, 1975] Lemaréchal, C. (1975). An extension of Davidon methods to non differentiable problems. *Math. Programming Stud.*, (3), 95–109. 3

[Lewis & Overton, 2013] Lewis, A. S. & Overton, M. L. (2013). Nonsmooth optimization via quasi-Newton methods. *Math. Program.*, 141(1-2, Ser. A), 135–163. 3, 6, 7, 9, 13, 19, 33, 35, 39, 62, 74

[Lewis & Zhang, 2015] Lewis, A. S. & Zhang, S. (2015). Nonsmoothness and a variable metric method. *J. Optim. Theory Appl.*, 165(1), 151–171. 10

[Lin et al., 2016] Lin, H., Mairal, J., & Harchaoui, Z. (2016). An Inexact Variable Metric Proximal Point Algorithm for Generic Quasi-Newton Acceleration. *arXiv e-prints*, (pp. arXiv:1610.00960). 7

[Liu & Nocedal, 1989] Liu, D. C. & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)), 503–528. 3, 7, 8

[Mascarenhas, 2004] Mascarenhas, W. F. (2004). The BFGS method with exact line searches fails for non-convex objective functions. *Math. Program.*, 99(1, Ser. A), 49–61. 5

[Mitchell & Overton, 2016] Mitchell, T. & Overton, M. L. (2016). Hybrid expansion–contraction: a robust scaleable method for approximating the H_∞ norm. *IMA Journal of Numerical Analysis*, 36(3), 985–1014. 115

[Nedić & Bertsekas, 2001] Nedić, A. & Bertsekas, D. P. (2001). Incremental subgradient methods for nondifferentiable optimization. *SIAM J. Optim.*, 12(1), 109–138. 32

[Nesterov, 2005] Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Math. Program.*, 103(1, Ser. A), 127–152. 3, 80

[Nesterov, 2016] Nesterov, Y. (2016). Private communication. Les Houches, France. 32

[Nocedal & Wright, 2006] Nocedal, J. & Wright, S. J. (2006). *Numerical Optimization*. New York: Springer, 2nd edition. 5, 7, 9, 33

[Overton, 1988] Overton, M. (1988). On minimizing the maximum eigenvalue of a symmetric matrix. *SIAM J. Matrix Anal. Appl.*, 9, 256–268. 89

[Overton & Womersley, 1993] Overton, M. & Womersley, R. (1993). Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. *Mathematical Programming*, 62(1-3), 321–357. 106

[Powell, 1976a] Powell, M. J. D. (1976a). Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In *Nonlinear Programming* (pp. 53–72). Providence: Amer. Math. Soc. SIAM-AMS Proc., Vol. IX. 2, 5

[Powell, 1976b] Powell, M. J. D. (1976b). A view of unconstrained optimization. In *Optimization in action (Proc. Conf., Univ. Bristol, Bristol, 1975)* (pp. 117–152).: Academic Press, London. 1, 2, 12, 19

[Recht et al., 2010] Recht, B., Fazel, M., & Parrilo, P. A. (2010). Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. *SIAM Review*, 52(3), 471–501. 115

[Shor, 1985] Shor, N. Z. (1985). *Minimization Methods for Non-differentiable Functions*. Springer Series in Computational Mathematics, Springer. 30

[Taylor et al., 2017] Taylor, A. B., Hendrickx, J. M., & Glineur, F. (2017). Exact worst-case performance of first-order methods for composite convex optimization. *SIAM Journal on Optimization*, 27(3), 1283–1313. 4

[Vandenberghe, 2019] Vandenberghe, L. (2019). Optimization Methods for Large-Scale Systems. http://www.seas.ucla.edu/~vandenbe/236C/lectures/smoothing.pdf, Lecture Note for ECE236C, (accessed 2019-10-10). 94

[Wolfe, 1969] Wolfe, P. (1969). Convergence conditions for ascent methods. *SIAM Rev.*, 11, 226–235. 1

[Wolfe, 1975] Wolfe, P. (1975). A method of conjugate subgradients for minimizing nondifferentiable functions. *Math. Programming Stud.*, (3), 145–173. 2, 3

[Xie & Waechter, 2017] Xie, Y. & Waechter, A. (2017). On the convergence of BFGS on a class of piecewise linear non-smooth functions. arXiv:1712.08571. 10, 67